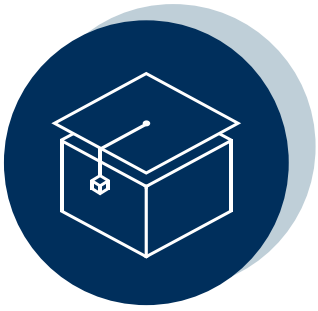

BISE Student

https://bise-student.io

DIPLOMA THESIS

Analysis, conceptual design and implementation of a web-application providing patient-friendly aggregation and refinement of information in patient information leaflets

Publication Date: 2022-02-22

Author
Tobias DEHLING
Karlsruhe Institute of Technology
Karlsruhe, Germany
dehling@kit.edu
0x6ca495d7b55310954f4b5f2986E5C11b6ed397E2

Abstract

Analysis, conceptual design and implementation of a web-application providing patient-friendly aggregation and refinement of information in patient information leaflets

Keywords: Diploma Thesis

Submission Date: 2022-02-22
Submission Contract: 0xe87F6011620DE19bAD5A4218d2ecDCEb353d4c90

Tobias Dehling

Analysis, conceptual design and implementation of a web-application providing patient-friendly aggregation and refinement of information in patient information leaflets

Diplomarbeit im Fach Wirtschaftsinformatik

Themensteller: Prof. Dr. Ali Sunyaev

Vorgelegt in der Diplomprüfung im Studiengang Wirtschaftsinformatik der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität zu Köln

Köln 2012

Table of Contents

Index of Abbreviations....V
Illustration Index....VII
Index of Tables....VIII
1 Introduction....1
1.1 Problem and Objective....1
1.2 Methodology....2
1.3 Text Structure....3
2 Preliminary Draft....4
2.1 Derivation of Objectives....5
2.1.1 Information in Patient Information Leaflets....5
2.1.2 Features and Drawbacks of Patient Information Leaflets....6
2.1.3 Prospective Users....8
2.1.4 Data Security/Reliability Concerns....8
2.1.5 Convenience....10
2.2 The Preliminary Draft....11
2.2.1 GUI Prototype....12
2.2.2 GUI Prototype Design Decisions....14
3 Requirements....21
3.1 Interview Design and Execution....21
3.1.1 Software Requirements Specification....21
3.1.2 Interview Characteristics....23
3.1.3 Interview Design....25
3.1.4 Interview Execution and Processing....26
3.2 Elicited Requirements....28
3.2.1 Usability....29
3.2.2 Functionality....33
3.2.3 Software System Attributes....36
3.2.4 Concluding Remarks....37
3.3 Requirements Evaluation....38
3.3.1 Problematic Requirements....39
3.3.2 Elicited Requirements versus Objectives for Preliminary Draft....40

4 Conceptual Design....42
4.1 Vaadin....42
4.2 Application Modules....45
4.3 Used Technologies....48
4.3.1 Tomcat....48
4.3.2 MySQL....48
4.3.3 Hibernate....49
4.4 Physical Infrastructure....49
4.5 Implementation Limitations....51
4.6 Conceptual Design versus Gathered Requirements....52
5 Implementation....53
5.1 Graphical User Interface....53
5.1.1 Reorganisation of Graphical User Interface Data Management....53
5.1.2 Realisation of Requirements....56
5.2 Data Access....60
5.2.1 Making the Data Available....60
5.2.2 Database Mapping....65
5.2.3 Provided Functionality....66
5.2.4 Realisation of Requirements....69
5.2.5 Concluding Remarks....70
5.3 Additional Functionality....71
5.3.1 Pop-up for Selection of Pharmaceuticals....71
5.3.2 Detection of Adverse Drug Reactions....72
5.3.3 Listing of Side Effects....75
5.3.4 Display of Alternative Dosage Forms....75
5.3.5 Display of Pharmaceuticals with same Active Pharmaceutical Agents....76
5.3.6 Comparison of Pharmaceuticals....77
5.3.7 Printing Information on Pharmaceuticals....80
5.3.8 Explanation of Technical Terms....81
5.3.9 Further changes/additions....82
5.3.10 Realisation of Requirements....83
5.4 Concluding remarks....86

6 Final Considerations and Remarks....87
6.1 Achievement of Objectives....87
6.2 Patient-Friendliness....89
6.3 Chosen Approach....89
6.4 Comparison with Similar Services....90
6.5 Further Research....93
6.5.1 Improvement of Data Quality....93
6.5.2 German Healthcare Telematics Infrastructure....94
6.5.3 Different Perspectives....95
Bibliography....96
7 Appendix....99
Appendix....99
Erklärung....152
Curriculum Vitae....153

Index of Abbreviations

ADR	Adverse Drug Reaction
API	Application Programming Interface
ATC	Anatomical Therapeutic Chemical (classification system)
CPU	Central Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
GUI	Graphical User Interface
GWT	Google Web Toolkit
HQL	Hibernate Query Language
HTML	HyperText Markup Language
HTTP(S)	HyperText Transfer Protocol (Secure)
ICD	International Classification of Diseases
ID	Identification Key
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LVS	Linux Virtual Server
MVC	Model View Controller
n.d.	no date
n.p.	no place
no.	number
p.	page
PDF	Portable Document Format
publ.	publisher
PZN	Pharmazentralnummer
UI	User Interface
UIDL	User Interface Definition Language

UML	Unified Modelling Language
URL	Uniform Resource Locator
SQL	Structured Query Language
vol.	volume
W3C	World Wide Web Consortium

Illustration Index

Fig. 1-1: Used development process....2
Fig. 2-1: Percentage of surveyed people that consider the risks to be serious....9
Fig. 2-2: Screenshot of the GUI prototype....12
Fig. 2-3: Screenshot of the tab providing information on pharmaceuticals....15
Fig. 3-1: Characteristics describing the group of interviewed people....27
Fig. 3-2: Grouping of requirements which were mentioned at least four times....29
Fig. 4-1: General architecture of Vaadin....43
Fig. 4-2: Client- and server-side Vaadin architecture....44
Fig. 4-3: Application Modules and their relationships....46
Fig. 4-4: Network diagram of application....50
Fig. 5-1: Class diagram: MVC for component visibility....54
Fig. 5-2: Graphical User Interface: PanelToggler and ComponentSelectionWindow....55
Fig. 5-3: Navigation section of the drug information tab....58
Fig. 5-4: Entity relationship diagram of used MySQL database....62
Fig. 5-5: Advanced search tab....67
Fig. 5-6: Tables storing data on adverse drug reactions....73
Fig. 5-7: Tab for composing a set of pharmaceuticals....72
Fig. 5-8: Final GUI and drug information tab design....79
Fig. 6-1: Layout of GELBE LISTE PHARMINDEX....92
Fig. 7-1: Mapping of relevant requirements to objectives for preliminary draft....108
Fig. 7-2: Complete and final entity relationship diagram of database model....146

Index of Tables

Tab. 2-1: Objectives implied by information in patient information leaflets.....6
Tab. 2-2: Objectives derived from patient information leaflets.....8
Tab. 2-3: Objectives implied by prospective users.....8
Tab. 2-4: Objectives implied by data security concerns.....10
Tab. 2-5: Objectives related to convenience aspects.....10
Tab. 2-6: Mapping of design characteristics to the objectives they are based on.....20
Tab. 3-1: Requirements: 'Drug Information Tab' subgroup.....30
Tab. 3-2: Requirements: 'Readability' subgroup.....31
Tab. 3-3: Requirements: 'Intuitive Operation of Web Application' subgroup.....32
Tab. 3-4: Requirements: 'Comprehensibility' subgroup.....33
Tab. 3-5: Requirements: 'Linking to Pharmaceuticals' subgroup.....33
Tab. 3-6: Requirements: 'Search for Pharmaceuticals' subgroup.....34
Tab. 3-7: Requirements: 'Additional Functionality' subgroup.....35
Tab. 3-8: Requirements: 'Contact to Operating Authority' subgroup.....35
Tab. 3-9: Requirements: 'User Accounts' subgroup.....36
Tab. 3-10: Requirements: 'Software System Attributes' group.....37
Tab. 5-1: Summary of realised requirements during GUI implementation.....59
Tab. 5-2: Summary of requirements realised by module 'Data Access'.....69
Tab. 5-3: Requirements abandoned during development of module 'Data Access'.....70
Tab. 5-4: Summary of requirements: additional functionality/associated adoptions.....85
Tab. 7-1: List of objectives on which the preliminary draft is based.....99
Tab. 7-2: Requirements mentioned most often during requirements elicitation.....107

1 Introduction

1.1 Problem and Objective

Patient information leaflets have poor readability and comprehensibility.[1] Especially for elderly patients, the font size is too small, the structure is not suitable to satisfactorily retrieve desired information, the used language and words are not appropriate, and the leaflets contain information that is irrelevant for the individual patient. These deficits lead to undesirable consequences: Patients harm themselves through wrong use of pharmaceuticals or get scared because of a lack of understanding, throw the pharmaceuticals away and do not take them at all. A web application providing pharmaceutical information could be more suitable for serving the required information and offer additional services. In order to facilitate the design and development of a web application providing aggregation and refinement of information in patient information leaflets three research questions need to be answered:

1. What are the requirements of a web application providing aggregation and refinement of information in patient information leaflets?
2. How can the information required for the web application be obtained and displayed?
3. How can the web application be realised?

To the best of our knowledge no web application providing aggregation and refinement of information in patient information leaflets for patients exists. Such a web application would be beneficial since it could mend the insufficiencies of patient information leaflets if it is well designed and implemented: In contrast to patient information leaflets, web applications are not limited to a sheet of paper, can be easily updated, can point the users to further information more comfortably, can be easily accessed prior to purchase of the pharmaceutical, and can provide a customisable structure and interactive content.

The aim of this thesis is to design and develop a prototype of the proposed web application. Therefore, it is necessary to determine the requirements of the web application, to provide and prepare the necessary data, to design the web application, and to implement the prototype of the web application.

---

[1] See regarding this and the following 3 sentences Nink, Schröder (2005), p. 13, 58-70.

1.2 Methodology

We decided to follow the waterfall model[2] because the time and manpower restrictions of this thesis do not allow for iterations in the development process, like repeated customer contact to ensure compliance with user expectations. Moreover, we are developing a prototype to demonstrate/evaluate the benefits of the proposed web application which will be refined before/if it is put into live operation. Another fitting problem characteristic is that we are developing a new system. Therefore, it is useful to elicit and analyse requirements and to finish the design prior to implementation in order to get a better vision of the application as well as to avoid unnecessary development effort. Especially requirements elicitation and analysis will provide valuable input that specifies the application and ensures that it conforms with user expectations. Figure 1-1 depicts the development process. The activities 'Integration and System Test' and 'Live Operation and Maintenance' listed in Sommerville (2007)[3] are replaced by the activity 'Evaluation' since we are developing a stand-alone prototype not intended for live operation whose usefulness needs to be evaluated.

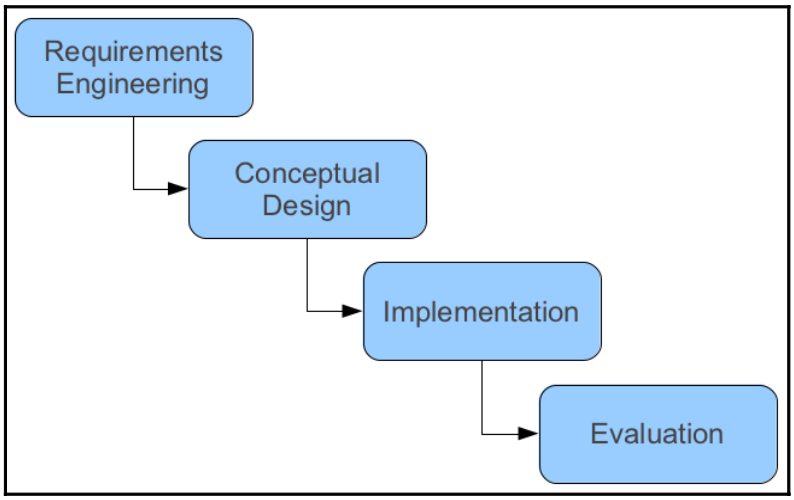

Fig. 1-1: Used development process[4]

Initially, it is necessary to determine the requirements. In order to prevent that the requirements get out of hand, at first, a preliminary draft of the web application will be created. The preliminary draft will be based on objectives that will be determined by examining different aspects of the proposed web application. Afterwards, prospective

---

[2] See Sommerville (2007) p. 96-98 for a short description of the waterfall model.

[3] See Sommerville (2007), p.96-98.

[4] Adopted from Sommerville (2007), p. 97.

users and stakeholders will be interviewed to derive further requirements based on the preliminary draft. Subsequently, the requirements will be analysed and utilised to create the conceptual design. In the following step, necessary data will be retrieved and preprocessed and suitable technologies for implementation will be chosen. Finally, the prototype of the web application will be implemented and its benefits evaluated.

1.3 Text Structure

The thesis is structured as follows: In the chapter '2 Preliminary Draft' different aspects of the proposed web application are examined and objectives as well as a preliminary draft are derived. In the following chapter '3 Requirements' the requirements are determined, analysed, and evaluated. In the chapter '4 Conceptual Design' the conceptual design is derived and illustrated. Chapter '5 Implementation' describes the implementation and the realised web application. Finally, the thesis finishes with the evaluation of the web application and the conclusion in chapter '6 Final Considerations and Remarks'.

2 Preliminary Draft

Accurate and fitting requirements are crucial for successful development of software since the requirements, their interpretation, and their utilisation shape the resulting application. Additionally, reducing the effort put into requirements engineering increases the risk of missing user expectations.[5] Rupp (2002) differentiates six groups of requirements elicitation techniques: creative techniques ('Kreativitätstechniken'), monitoring techniques ('Beobachtungstechniken'), questioning techniques ('Befragungs-techniken'), retrospective techniques ('Vergangenheitsorientierte Techniken'), feedback techniques ('Feedback-Techniken'), and supporting techniques ('Unterstützende Techniken'). Creative techniques are useful for deriving rough objectives. Therefore, application of creative techniques is not necessary because in this case, as demonstrated in the following chapter '2.1 Derivation of Objectives', sufficing objectives can be determined by contemplation or be derived from literature. Monitoring techniques and retrospective techniques require the existence of processes and applications that are similar to those that are to be implemented. Since, to the best of our knowledge, no such processes/applications already exist, monitoring techniques and retrospective techniques are not appropriate either. As supporting techniques are usually used in combination with other techniques, questioning techniques and feedback techniques are left for requirement elicitation.

Repeated customer contact to gain feedback would be too much effort for the development of the prototype. Therefore, we decided to initially devise a preliminary draft on our own and present this to potential users in an interview to gain feedback and additional requirements. An interview is an appropriate technique because consulting only a few potential stakeholders/users suffices for the scope of this thesis, matches the available resources, and allows for immediate enquiry by the interviewer. Along with the provision of feedback, the preliminary draft has further advantages: Users get a better idea of the web application that is to be developed, its possibilities, and its limitations. Additionally, a better understanding and image of the application leads to potentially more concise and reasonable requirements.

---

[5] See regarding this and the following 3 sentences Rupp (2002), p. 92, 109-129.

2.1 Derivation of Objectives

To facilitate the creation of the preliminary draft and to establish a basic understanding of the application to be developed, we will consider different aspects of the application and derive objectives in the following.

2.1.1 Information in Patient Information Leaflets

The main objective of the web application is the aggregation and refinement of information in patient information leaflets. Hence, we need to identify the information that patient information leaflets have to provide:

- Information for identification of a pharmaceutical[6]
  - Identifier
  - Substance group, indication group, or mode of application
- Field of application
- Information to be considered before taking the pharmaceutical
  - Contraindications
  - Necessary precautions
  - Possible adverse drug reactions
  - Warning messages
- Taking instructions
  - Dosage
  - Mode of application
  - Taking frequency
  - Treatment duration
  - Information regarding over- or underdosage
  - Suggestion to contact pharmacist or physician in case of further questions
- Side effects and possible counteractive measures

---

[6] See regarding this listing Bundestag/Bundesrat (1976), p. 16-18.

- Pointer to expiry date
  - Warning to not take a pharmaceutical after expiry
  - Information regarding storage
  - Indications signalling spoilage
- Qualitative and quantitative listing of ingredients
- Available dosage forms and package sizes
- Name of pharmaceutical company
- Name of manufacturer or importer
- Date of last revision

The list above comprises the main information that needs to be provided for common pharmaceuticals. Some explanations might be helpful: Substance group is the group to which the active pharmaceutical agent, the substance responsible for the intended effect, belongs. Indication group refers to the therapeutic application. Contraindications mention situations in which the pharmaceutical should not be used. Adverse drug reactions refer to negative effects that can occur if a pharmaceutical is taken in combination with another pharmaceutical.

The required functionality to aggregate and refine information in patient information leaflets implies a few objectives. Information that should be displayed needs to be accessible by the application and requires an appropriate user interface. Capabilities for aggregation and refinement of information need to be implemented and accessible in the user interface as well.

#	**Objective**
1	Make relevant information in patient information leaflets available for the user.
2	Provide aggregation functionality for information in patient information leaflets.
3	Provide refinement functionality for information in patient information leaflets.

Tab. 2-1: Objectives implied by information in patient information leaflets

2.1.2 Features and Drawbacks of Patient Information Leaflets

Since the web application is an alternative information source to patient information leaflets, it should offer applicable features of patient information leaflets and avoid drawbacks of patient information leaflets. A survey by Nink and Schröder identified

three main troubles that patients have with patient information leaflets: content, readability, and comprehensibility.[7] Patients argue that the leaflets are too long and provide too much irrelevant content. Readability is perceived as poor because of the length and the confusing structure that impedes discovery of desired information. In combination with the inadequate, formal language and the use of technical terms, the deficits in content and readability are found to impair the comprehensibility. Suggestions for improvement of patient information leaflets were captured in the survey as well. Only important, generally understandable information should be provided in the patient information leaflet. However, different patients desire different information. One interviewee, for example, regarded the active pharmaceutical agent as unimportant information, whereas, another interviewee wished for further information regarding the active pharmaceutical agent. Readability could be improved by a larger font size and a well-arranged layout, which eases the discovery of desired information and bundles related information. In order to avoid the drawbacks of patient information leaflets and to incorporate the suggestions for improvement, the user should be able to select the information to be displayed, the layout should be well-arranged, the interface should be zoomable, and technical terms should be replaced or at least explained.

Recommendations for the design of patient information leaflets to improve readability are provided by the European Commission as well.[8] Different font sizes should be used to stress key information and the font should be easy to read. Contrast between text and background colour should be sufficient and the text alignment 'justified' should not be used so that the text is easy to read. Headings are deemed important for navigation and should be in a bold type face or accentuated with a different colour. However, headings of the same level should have the same design. To avoid confusing the readers, simple words of few syllables and short sentences should be used. Abbreviations, acronyms, and scientific symbols should be avoided whenever possible, otherwise, they should be explained. Some of the recommendations are also mentioned in the survey of Nink and Schröder. New recommendations that should be kept in mind for the design of the web application are the contrast between text and background colour and the formatting of the headings.

---

[7] See regarding this and the following 8 sentences Nink, Schröder (2005), p. 58, 61-75.

[8] See regarding this and the following 6 sentences European Commission (2009), p. 7-11.

#	Objective
4	Offer functionality for the selection of the displayed information.
5	Structure the layout according to user wishes and needs.
6	Improve structuring with accentuated and consistent formatting of headings.
7	Provide zoom capabilities or a sufficient font size.
8	Avoid or explain technical terms or notations.
9	Obtain readability with clear contrast between text and background colour.

Tab. 2-2: Objectives derived from patient information leaflets

2.1.3 Prospective Users

Intended users of the web application are persons who want to get information regarding pharmaceuticals licensed in Germany. Indicators like age, education, or profession do not restrict the user group, hence, user web application operating proficiency ranges from low to high. Accordingly, the application has to provide assistance for inexperienced users but should not bother or slow down experienced users. Assistance can be provided through features like tooltips, instructions, a configurable interface, or different views. A configurable interface avoids scaring off users with too much information by displaying only basic, aggregated information for basic users and offering additional, detailed information when demanded by more proficient users. Different views are preconfigured interfaces that support the user in a similar way to a configurable interface.

#	Objective
10	Provide assistance for inexperienced users.
11	Avoid hindering or slowing down experienced users.

Tab. 2-3: Objectives implied by prospective users

2.1.4 Data Security/Reliability Concerns

Figure 2-1 represents results of a representative survey of the German population and shows that data security considerations should not be neglected in a software development project. About half of the surveyed people deem data security related risks like abuse of personal data or online fraud as serious risks. This means, that these risks are considered by more people as serious than risks like terrorism, acts of violence, or seriously contagious diseases. Additionally, the group of surveyed people is a

representative subset of prospective users, which are all people mature enough to deal with pharmaceuticals, because the group of surveyed people is a representative subset of the German population older than 16. Obviously, data security concerns need to be considered in the development process if 50% of prospective users see it as a serious risk. Fortunately, the main purpose of the proposed web application is the provision and not the collection of information. Accordingly, collection and especially storage of additional sensitive data, e.g. data about user behaviour, should be avoided because it is not necessary for the main purpose and collecting less data reduces the effort necessary for data protection. Data that needs to be collected is data representing the information in patient information leaflets. It is important that the data is correct and reliable to avoid misinformation of patients, which could lead, at the worst, to serious repercussions like death, if, for example, wrong dosages are listed or contraindications are omitted. To provide correct and reliable information, it is important to use only trustworthy sources of information, to prevent tampering with data, and to provide a

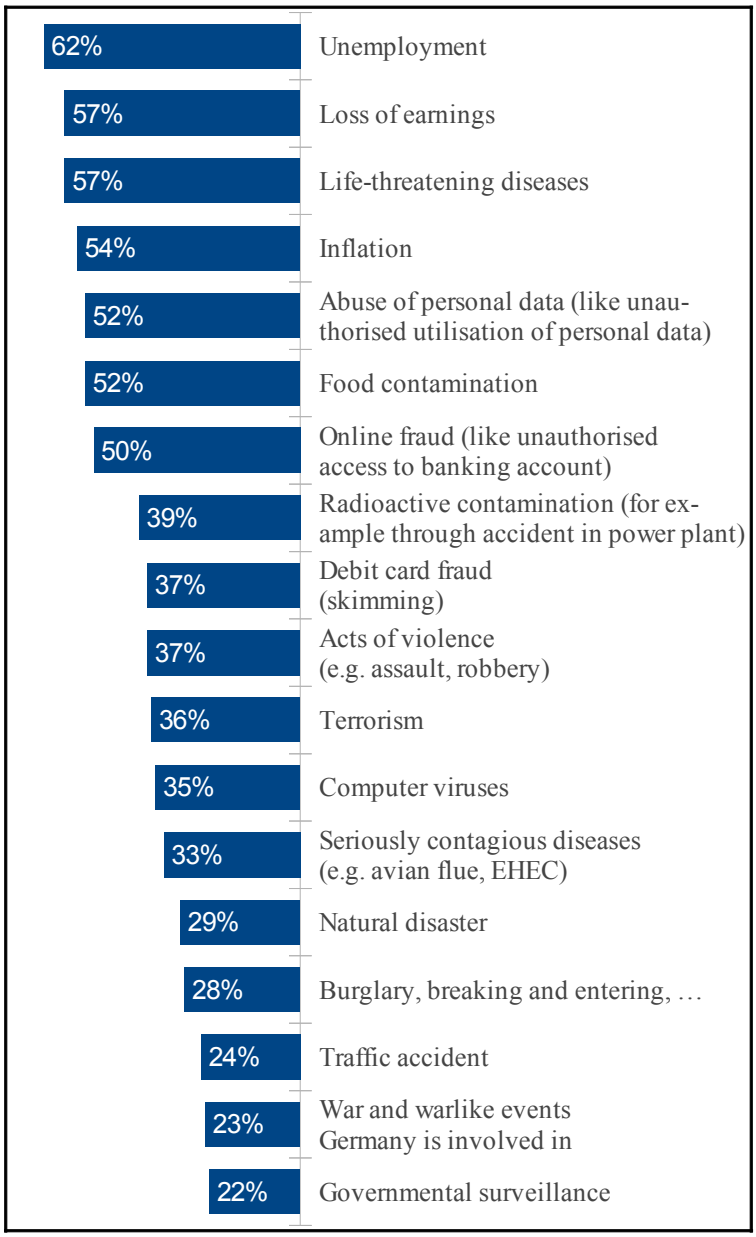

Fig. 2-1: Percentage of surveyed people that consider the risks (listed on the right) to be serious[9]

[9] Adopted from Deutsche Telekom (2011), p. 15.

mechanism to flag potentially wrong information. Furthermore, the user should be informed about the data sources and potential risks so that he can decide on his own whether he finds the information trustworthy or not.

#	Objective
12	Do not collect and store data that is not relevant for the display of information in patient information leaflets.
13	Use only trustworthy sources of information.
14	Protect data against tampering.
15	Provide a mechanism to report potentially wrong information.
16	Inform users about data sources and potential risks associated with the application.

Tab. 2-4: Objectives implied by data security concerns

2.1.5 Convenience

A web application providing information about pharmaceuticals needs to be accessible around the clock so that users can retrieve information when they need to. Additionally, response times need to be short, almost unnoticeable, in order to provide the required information in a convenient way and as fast as possible. This requires a scalable architecture because in productive use many users might use the web application concurrently. It should be possible to easily increase performance of the web application by running it in more threads on one or multiple machines. Performance will not be a problem for the prototype, which will have only a few users, but scalability should be kept in mind and easily be realisable. Additional functionality, like for example provision of information regarding price developments or suppliers of pharmaceuticals, might be useful. Therefore, it is necessary that the architecture is expandable as well so that additional features can be easily implemented later on.

#	Objective
17	The web application needs to be accessible around the clock.
18	Response times should be short (almost unnoticeable).
19	The architecture needs to be scalable.
20	The architecture needs to be expandable.

Tab. 2-5: Objectives related to convenience aspects

2.2 The Preliminary Draft

As mentioned above, the main purpose of the preliminary draft is to give users to be interviewed a better idea of the web application, its limitations, and its possibilities in order to improve conciseness and quality of the requirements to be elicited. Hence, we developed a prototype for the graphical user interface (GUI), which is more suitable to convey the look and feel of the intended web application than alternatives like textual description or unified modelling language (UML) diagrams. The prototype can be classified as an exploratory, presentational prototype[10] that is executable and demonstrates a possible, fully functional graphical user interface, but lacks further functionality not associated with the presentation layer, like provision of actual information about pharmaceuticals. Purpose of the prototype is mainly the communication of a clear picture of the preliminary draft for the web application. In contrast to envisioning the application based on text or diagrams, a GUI prototype enables the user to interact with an actual application and provides a clear foundation for requirement elicitation. Another advantage is that room for misinterpretation is reduced since interviewer and interviewee can relate their statements to or clarify them based on the GUI prototype. With text or diagrams describing the application, misunderstandings are more likely because of more room for interpretation and a greater distance to the intended product.

A survey of nine major industrial projects by Bäumer et al. supports the utilisation of prototypes as well.[11] Results of the survey are that prototypes support communication between developers and end users and are suitable for development and communication of a vision of the intended application. A further finding was a tendency for throwaway prototypes that are implemented as quickly and cheaply as possible and are not implemented well enough to be reused. Nevertheless, implementing the prototype with the same technologies as the final product and using a modular architecture improves the likelihood to be able to reuse the prototype or at least parts of it, which would reduce the implementation effort. Additionally, modules can be easily adopted, expanded or added if necessary.

---

[10] See Bäumer et al. (1996), p. 532-533.

[11] See regarding this and the following 2 sentences Bäumer et al. (1996), p. 535-537.

2.2.1 GUI Prototype

We decided to use the Vaadin[12] Framework, a Java[13] framework for the development of web-based user interfaces, for the development of the GUI prototype. Vaadin hides web technologies like HTML[14], JavaScript[15] or AJAX[16] from the developer so that the user interface can be implemented by writing Java classes. Developing with Vaadin leads to a thin-client architecture, where everything except the display of the user interface and the capture of user interaction is handled on the server side. Since the application logic is hidden from the client side security is increased. Additionally, executing the application logic mainly on the server side eases integration of further components, like the database providing information on pharmaceuticals, because the integration affects only the server side. Rendering of the user interface on the client side is handled by an abstraction tier that ensures compatibility with common browsers and requires only interaction with the abstraction tier from the developers, which relieves them of effort

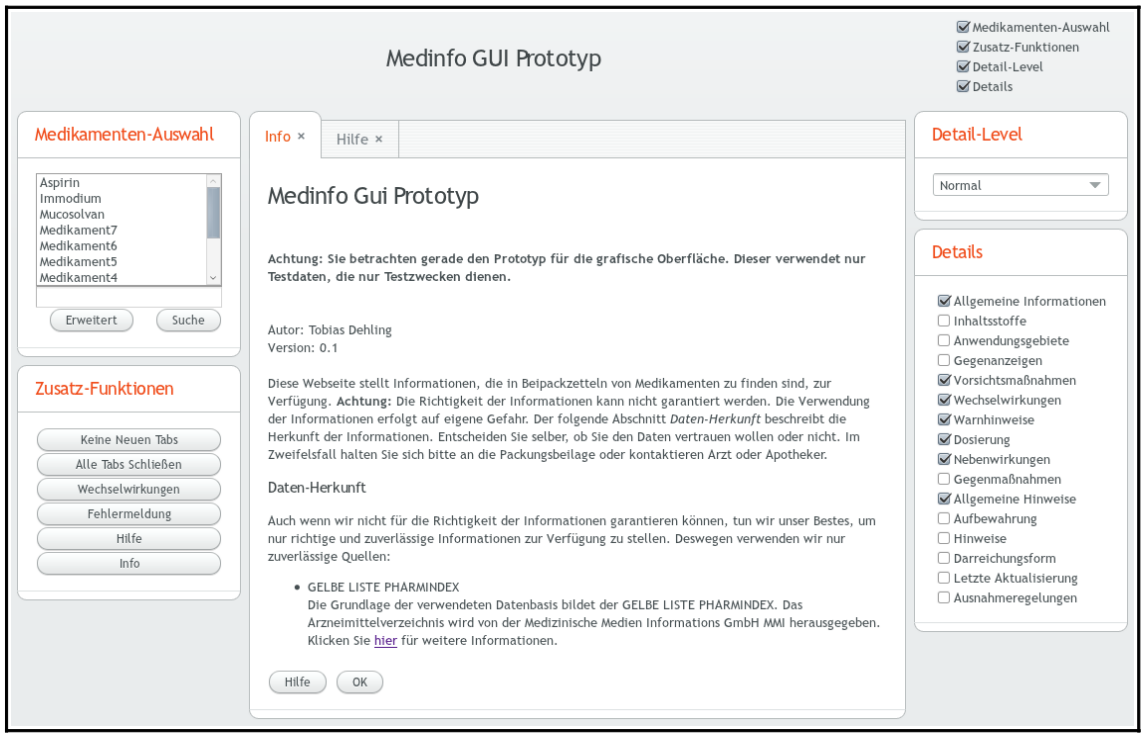

Fig. 2-2: Screenshot of the GUI prototype

---

[12] See 'https://vaadin.com/home' for more information on Vaadin.

[13] See 'http://java.com/en/about' for more information on Java.

[14] See 'http://dev.w3.org/html5/spec/spec.html' for information on HTML.

[15] See 'https://developer.mozilla.org/en/A_re-introduction_to_JavaScript' for information on JavaScript.

[16] See 'http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications' for information on AJAX.

for managing the client side like handling browser compatibility issues. Therefore, developers can compose the user interface by adding, adopting, and expanding widgets[17] provided by the framework. The Vaadin framework will be described in more detail later on in the chapter '4.1 Vaadin'.

Figure 2-2 depicts the user interface[18] structure of the GUI prototype. The user interface is separated into four main components:

- The 'Header' on the top, which displays the name of the application and provides check boxes to switch the displayed sub-panels off or on.
- The 'SearchBar' on the left with the two sub-panels 'Medikamenten-Auswahl' and 'Zusatz-Funktionen'.
  - The panel 'Medikamenten-Auswahl' is used to search for pharmaceuticals for which information should be displayed. Pharmaceuticals can be selected directly in the list-select, searched by name in the textbox below the list-select, or an advanced search mask can be opened by clicking on the button 'Erweitert'. The advanced search mask can be used to search for pharmaceuticals by name, PZN[19], ATC[20] code, ICD[21] code, field of application or active pharmaceutical agent.
  - The panel 'Zusatz-Funktionen' provides functionality beyond the search for pharmaceuticals. Functionality like closing all open tabs, turning creation of new tabs on or off, or displaying help or about information. Additional functionality like an adverse drug reaction tool or price comparison tool for pharmaceuticals could be made accessible via this panel.

---

[17] Widgets are GUI modules, like buttons, text boxes, or lists, that can be combined to create the user interface.

[18] The user interface is in German because the web application should provide information on German pharmaceuticals whose users are more likely to speak German than English.

[19] PZN refers to Pharmazentralnummer, which is a German unique identifier for pharmaceuticals. The PZN has to be printed on the packaging of every German pharmaceutical.

[20] ATC refers to the Anatomical Therapeutic Chemical classification system. An ATC code identifies a pharmaceutical based on the anatomical application area, its therapeutic characteristics, and its chemical characteristics. Accordingly, two pharmaceuticals from different brands can have the same ATC code, if they have the same anatomical application area, therapeutic characteristics, and chemical characteristics.

[21] ICD refers to the International Classification of Diseases. ICD codes identify diseases and can be used in the proposed web application to search for pharmaceuticals associated with a particular disease or a group of diseases.

- The 'SettingsBar' on the right with the two sub-panels 'Detail-Level' and 'Details'.
  - The panel 'Detail-Level' is used to set levels of detail for the displayed information on pharmaceuticals. With a higher level of detail more information on pharmaceuticals will be displayed and with a lower level of detail less information on pharmaceuticals will be displayed.
  - The panel 'Details' contains check boxes for all categories of details, like general information, contraindications, field of application, et cetera. By checking or unchecking the check boxes the user can decide in detail which information should be displayed. If a check box is checked the associated information will be displayed and if a check box is unchecked the associated information will not be displayed. The levels of detail that can be set in the panel 'Detail-Level' are actually presets for settings of the check boxes.
- The 'TabSheet' in the centre of the application. The 'TabSheet' can hold tabs for search masks, search results, information on pharmaceuticals, and all other responses on user interaction that can not be handled by the other main components. In the panel 'Zusatz-Funktionen' the user can toggle whether new tabs should replace the current tab or be displayed as an additional tab.

2.2.2 GUI Prototype Design Decisions

Purpose of this chapter is to validate that the GUI prototype fulfils the objectives derived in chapter '2.1 Derivation of Objectives' by linking the objectives to design characteristics. As stated in objective 1, the application needs to make the relevant information provided by patient information leaflets available to the user. Provision of the required information is realised with a tab that can display the information for a selected pharmaceutical. Based on a literature review, Nink and Schröder developed a structuring for patient information leaflets that incorporates wishes and needs of patients.[22] They recommend to group information that is associated with regard to content. The groupings have a common question as heading that characterises the information provided. In accordance with objective 5, we adopted the proposed structuring in order to organise the information based on wishes and needs of users and to provide the relevant information listed in chapter '2.1.1 Information in Patient

[22] See regarding this and the following 2 sentences Nink, Schröder (2005), p. 88-90.

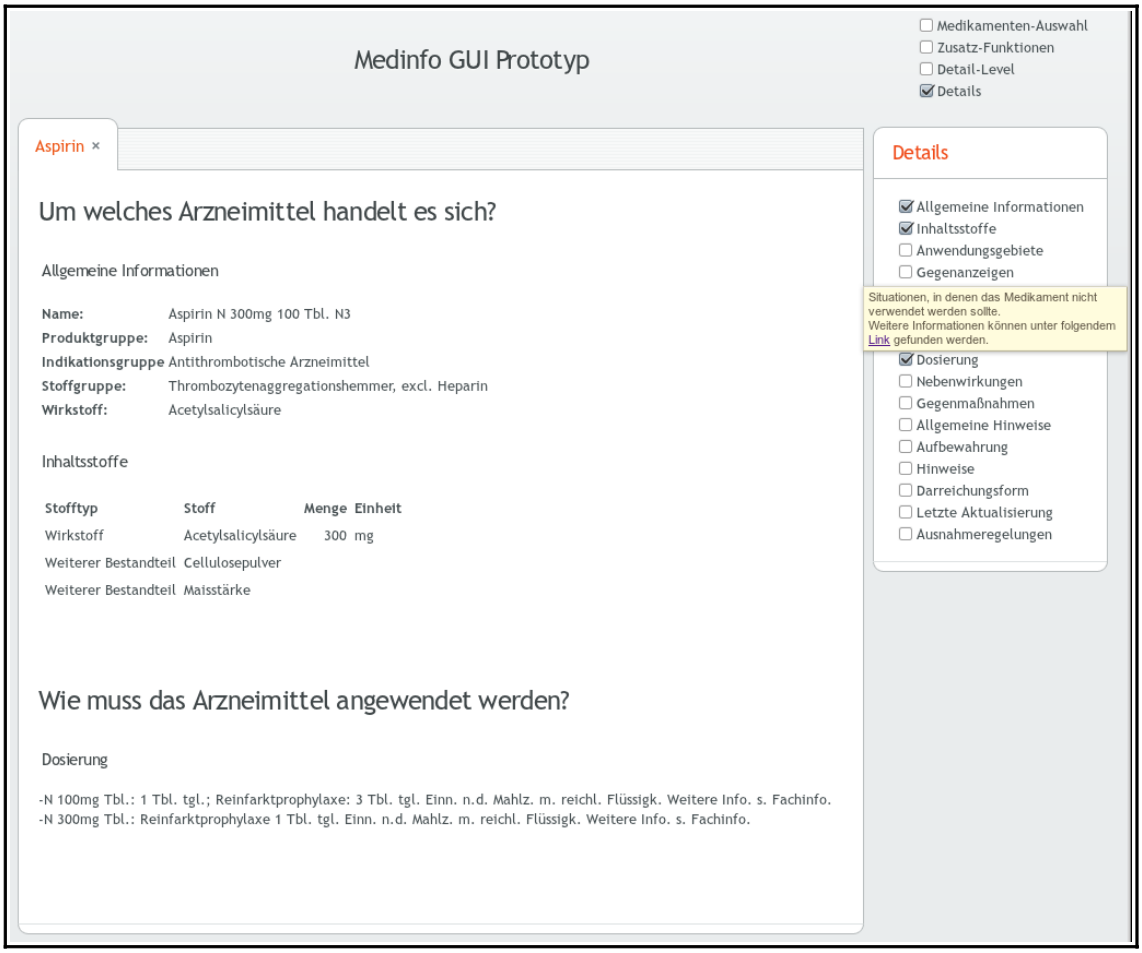

Fig. 2-3: Screenshot of the tab providing information on pharmaceuticals

Information Leaflets'. Information on pharmaceutical company and manufacturer or importer is omitted because multiple surveys of patients or specialists observed that this information is deemed unimportant[23]. Moreover, the user can, as demanded by objective 4, decide which information should be displayed by toggling check boxes in the 'Details' panel or selecting presets in the 'Detail-Level' panel. As illustrated in figure 2-2 and figure 2-3, the check boxes in the 'Header', which enable the user to hide not needed panels, generate further configuration potential. In figure 2-2 all panels are enabled and in figure 2-3 only the 'Details' panel is active.

Figure 2-3 is a screenshot of the tab that provides the information on pharmaceuticals. In the screenshot a custom set of information is selected so that only general information on the pharmaceutical, the ingredients and the dosage are given. Additionally, the screenshot illustrates that, in compliance with objectives 6 and 9, the headers are accentuated and consistently formatted and the contrast between text and

[23] See Nink, Schröder (2005), p. 57-58, Fuchs, Hippius, Schaefer (2003), p. 303-304, and Fuchs et al. (2007), p. 168-169.

background colour is sufficient. Objective 10 advocates support for inexperienced users, which is supplied by multiple features. Tooltips offering assistance are implemented for every component and can be raised by hovering over the component with the cursor. Figure 2-3 depicts a tooltip explaining contraindications in a short sentence and a link to Wikipedia for further information. In addition, a help tab that presents general usage information and detailed guidance concerning single page components can be accessed via the panel 'Zusatz-Funktionen'. Further assistance is given through the simple access to the web application because no installation or other preparatory effort is required, instead, the user just needs a current browser supporting JavaScript. To take objective 11 into account, we considered potential needs of experienced users as well. Information can be accessed with a few clicks or key-strokes, tooltips can be easily ignored, the web application does not perform unrequested actions, the advanced search accepts more specific search keys, like PZN or ATC code, and information that might not be relevant for the common user, like the ingredients, can be displayed. Since common browsers like Firefox, Opera, Chrome or Internet Explorer already provide sophisticated zoom capabilities, which are mentioned in objective 7, we advise the user to use the zoom functionality of the browser instead of implementing application-specific zoom capabilities. To avoid complications while zooming, the GUI components are configured to provide scrollbars when necessary and not to overlap as long as enough space is available.

Objectives 17 and 19 require scalability and high availability of the web application. Vaadin applications can be deployed as servlets[24], therefore, the scalability and accessibility objectives can be easily achieved by running the servlet in a cluster of Tomcat application servers, which manages load balancing and failover[25]. Load balancing refers to assigning pending work to servers as efficiently as possible and failover refers to re-routing unfinished work from crashed servers. Up until now, the GUI prototype displays only exemplary data so that no database access is required. However, the web application should provide useful information and requires access to a database. Hence, the database needs to be available around the clock as well as scalable because the web application cannot provide information without a working database and scaling up the performance of the Tomcat cluster increases the required

---

[24] Servlets are web components that can be run on compatible web servers and provide dynamic content. Vaadin currently supports Servlet 2.3. See JSR 154 for more information (http://download.oracle.com/otndocs/jcp/7840-servlet-2.3-spec-oth-JSpec)

[25] See Apache Software Foundation (n.d.), p. 1-4.

performance of the database. Scalability and high availability of the database could be realised by, for example, using a MySQL database with replication in a master-slave setting[26] in combination with Linux Virtual Server (LVS) for load balancing and failover management[27]. With the utilisation of a Tomcat cluster and MySQL replication in combination with LVS increasing performance demands should be manageable.

Vaadin development is based on Java and Java is object-oriented so that objective 20 can be easily handled: The architecture is expandable because the modular structure induced by object-orientation facilitates modification and addition of classes – the actions required for provision of additional functionality. Further functionality can be added to classes without affecting other classes, further objects of existing or new classes can be integrated, and inheritance eases the introduction of further, more specialised objects. The GUI prototype generally shows subsecond response times, which are addressed in objective 18. However, database queries, which will be necessary later on, will increase response times and it is already possible to create slow response times in some browsers by requesting repainting of many objects at once. In order to display the application the client has to render JavaScript, which takes some time. We kept the GUI simple and the amount of necessary objects at a minimum to reduce the amount of JavaScript that needs to be rendered, but we will have to wait for a complete version and user feedback to determine satisfaction with response times. If the users are satisfied with the response times of the application it would be a waste of effort to try to reduce them even further. By writing well performing code, considering SQL performance, keeping the necessary rendering of JavaScript in mind, and leveraging scalability to provide sufficient computing power on the server side, response times should be manageable.

In conformance with objectives 12 and 14, intended data collection is restricted on data that is provided by patient information leaflets and the server-oriented architecture of Vaadin prevents data tampering. Since the user does not need direct access to the database, the database can be connected via a private network, which makes it harder for an attacker to access the database. Additionally, the application requires only read access to the database so that accounts used by the application are not able to tamper with data. However, user inputs need to be checked for SQL injections[28] so that a user cannot gain access to the database by running unintended SQL queries. Complying with

---

[26] See Oracle (2011), p. 1423-1431.

[27] See Zhang (2000), p. 1.

objective 13, only sources that are deemed trustworthy by general opinion will be used. As advised by objective 16, an info-tab displayed at application startup describes the sources of information, indicates that users should decide on their own whether they trust the information, and informs users that they can use the information only at their own risk.

Avoiding or explaining technical terms to obey objective 8 is problematic. In the user interface itself we use only common expressions, explain them in tooltips, and link to further information, but improving the information provided by the database is complex. As basic source of information, we intend to use GELBE LISTE PHARMINDEX[29], which primarily targets medical professionals. The orientation on medical professionals leads to an excessive use of abbreviations and technical terms which deteriorate comprehensibility. These abbreviations and technical terms cannot be easily replaced because their meaning depends on context and is sometimes ambiguous. We will address this problem later on when we have analysed the requirements and have a more precise idea of the web application so that we can better evaluate alternatives for improving the data or accessing other sources of information. Objective 15, provision of report functionality for potentially wrong information, requires further information and decisions as well. In the GUI prototype we display just a message that users should report wrong information by mail when they click on the associated button. With a further developed design it should be possible to support objective 15 with more sophisticated functionality that still fits the rest of the application. Inadequate decisions would be, for example, requiring a user administration for the reporting of wrong information, if it is not necessary for any other functionality or providing elaborate functionality to report wrong information if information is unlikely to be wrong because of other arrangements.

The primary feature of the web application is the provision of aggregation and refinement functionality for information in patient information leaflets, which are listed in objectives 2 and 3. This functionality should support users and satisfy needs of users so that it will be determined during the interviews for requirement elicitation. In the GUI prototype only some basic aggregation and refinement functionality is implemented: Users can select which information should be displayed, users can look

---

[28] SQL injection refers to SQL code that is given as application input by a malicious user, executed by the application, and intended to gain access to the database or tamper with the data.

[29] See 'http://www.gelbe-liste.de' for further information.

pharmaceuticals up by specific properties like ATC code, terminology like contraindications, side effects, and so on are explained, and links to further information are provided.

Table 2-6 summarises the objectives and associated design decision. With the exception of a few objectives that have to be dealt with later on, the GUI prototype honours the objectives and should represent an acceptable preliminary draft of the web application.

Category	#	Objective	Design
Data	1	Provide information on pharmaceuticals	Tab providing information on pharmaceuticals
	12	No unnecessary data collection	No unnecessary data collection
	13	Trustworthy sources of information	Utilisation of sources that are deemed trustworthy in general opinion
	14	Data tampering	Server-oriented architecture, only read access for application, connection via private network
	16	User information: risks and data sources	Info tab
Interface Design Principles	5	Structure of layout	Groupings of associated information, questions easily convey information provided in groupings
	6	Formatting of headings	Consistent formatting of headings
	8	Technical terms	Avoidance and explanation in GUI, improvement of data in database will be considered later on
	9	Contrast and readability	Black font on fair background colour
	10	Inexperienced users	Tooltips, tab providing help information, links to further information, easy application access with a browser
	11	Experienced users	Low amount of clicks/key-strokes, help information can be easily ignored, specific search keys, display of detailed information

Category	#	Objective	Design
Functionality	2	Aggregation of information	Selection of displayed information, lookup by specific characteristics, requirements elicitation will determine more
	3	Refinement of information	Explanations, links to further information, requirements elicitation will determine more
	4	Selection of displayed information	Check boxes to select displayed information, check boxes to toggle displayed GUI components
	7	Zoom capabilities	Hint to use zoom capabilities of browser
	15	Wrong information	Report by mail, will be further considered later on
Application Design	17	Accessibility	Clustering, failover
	18	Response times	Simple interface, well performing code and SQL, scalability
	19	Scalability	Clustering, load balancing
	20	Expandability	Object orientation

Tab. 2-6: Mapping of design characteristics to the objectives they are based on. (Objectives are mentioned in key words. See Tab. 7-1 in the appendix or chapter '2.1 Derivation of Objectives' for a detailed listing of the objectives.)

3 Requirements

The requirements engineering process is comprised of three steps. At first we will elicit the requirements with interviews. After interview analysis, we will analyse the gathered requirements by formulating them, writing them down, sorting, grouping and deciding which should be realised. Afterwards we will validate the requirements by checking whether they adhere to the characteristics of good requirements described below in chapter '3.1.1 Software Requirements Specification'.

3.1 Interview Design and Execution

At first, we determined aspects of good requirements as well as important interview characteristics to improve the interview design and execution and to enhance the quality of the gathered requirements. In the following, we will describe these aspects of interview preparation in addition to the actual design and execution of the interview.

3.1.1 Software Requirements Specification

With IEEE (1998) the Institute of Electrical and Electronics Engineers (IEEE) presents a recommended practice for the composition of software requirements specifications. Although we are not going to create a dedicated requirements specification document, IEEE (1998) contains useful information for the compilation of requirements. A dedicated software requirements specification is not necessary because the thesis itself describes the relevant requirements. Additionally, a novel proof of concept prototype of a web application is to be developed so that we do not need a software requirements specification to serve as a binding contract with a client. Our motive for requirement engineering is to determine user expectations and wishes to increase user acceptance and patient-friendliness. In the following, we will, on the basis of IEEE (1998), specify aspects that need to be considered during requirements engineering for the scope of this thesis.

The gathered requirements should address the aspects user interface, functions, and software system attributes.[30] The user interface (in web applications the part of the application displayed in the browser) is the part of the application visible to the user,

[30] See regarding this and the following 5 sentences IEEE (1998), p. 15-18.

therefore, the user interface should especially meet user expectations. Functions refer to features that should be provided by the web application. In our case, we do not need requirements for subordinate functions that are transparent to the user like database connections. However, we need requirements for desired top-level functionality to improve user satisfaction, like detection of possible adverse drug reactions. Software system attributes are attributes like reliability, availability and security. These attributes should illustrate user expectations. Users, for example, probably expect the web application to be accessible when they need information and do not want the application to abuse their provided input.

A requirement is "a condition or capability needed by a user to solve a problem or achieve an objective"[31]. Hence, requirements should not specify design or implementation details and should not restrict the software in other ways.[32] The gathered requirements should be correct and unambiguous, i.e., specify a condition or capability that is actually needed and allows for only one interpretation. In addition, requirements need to be complete so that they provide all necessary information. Requirements should be consistent so that they do not contradict specified properties of real-world objects, do not contradict each other, and do not refer to the same real-world object with different terms. For example, a requirement specifying that a table titled contraindications should list possible negative effects during treatment with a pharmaceutical would not be consistent because it contradicts properties of a real-world object. A contraindication, the real-world object, is actually a situation where the pharmaceutical should not be used for treatment, e.g., a patient who is allergic to the pharmaceutical. Additionally, requirements should be verifiable; it must be possible to decide whether a requirement has been met or not. A requirement stating to provide as much information on pharmaceuticals as possible would be unverifiable because it is always possible to research more information. Requirement statements should be tangible and quantifiable to ensure verifiability, like a requirement demanding to list all contraindications listed in the respective patient information leaflet. Furthermore, requirements need to be traceable, that is having a clear source and enabling referencing. Thus, requirements need a unique identifier so that it is possible to link requirements with each other and decisions/further developments based on it.

---

[31] IEEE (2010), p. 301.

[32] See regarding this and the following 10 sentences IEEE (1998), p. 4-7.

3.1.2 Interview Characteristics

As determined above, we will use interviews for requirement elicitation. This decision is reaffirmed by Davis et al. (2006), which is a systematic review of empirical studies on the effectiveness of elicitation techniques. This review led to the result that "interviews, preferentially structured, appear to be one of the most effective elicitation techniques in a wide range of domains and situations"[33]. Another interesting finding is that prototypes seem to prevent gathering of information that cannot be associated with the prototype itself because attention of the user is focused on the prototype.[34] Nevertheless, we will use the GUI prototype developed in chapter '2 Preliminary Draft' to leverage its ability for gathering feedback. In order to avoid preventing information from being gathered, we will not present the GUI prototype in the interview until we have covered general questions about ideas, wishes, and expectations concerning the web application to be developed.

Interviews can be classified in a range from structured to unstructured[35]. In structured interviews a set of predetermined questions is employed. Semi-structured interviews allow for more flexibility and adoption to the course of interview. In semi-structured interviews the interviewer has still a set of predetermined questions, but he can rephrase them, choose between them or introduce further questions. Unstructured interviews need no envisioned course of action. After a short introduction, interviewer and interviewee see where the interview takes them. Thus, we will conduct semi-structured interviews because we want to elicit new requirements, for which it might not be possible to predetermine suitable questions, but we still want to stay on a topic concerning requirements and investigate specific subjects.

To improve interview effectiveness, we will utilise the personal and reported experiences from various interviews outlined in Hove, Anda (2005). Hove and Anda identified four crucial areas of concern for interview design: effort necessary for the interview, interviewer skills, interview tools, and interaction with the interviewee.[36] Besides the conduction of the interview, necessary effort for scheduling, collection of background information, preparation of interview guides, and analysis need to be considered. In

---

[33] Davis et al. (2006), p. 185.

[34] See Davis et al. (2006). p. 184.

[35] See regarding this and the following 5 sentences Cohene, Easterbrook (2005), p. 96-97.

[36] See regarding this paragraph Hove, Anda (2005), p. 23-31.

addition to software requirements engineering skills, interviewers need to leverage interviewing skills. To gain trust, interviewees should be assured of confidentiality and anonymity, the purpose of the interview should be explained, and permission for audio recording should be requested. The interviewers should pay attention, let the interviewee talk, express themselves clearly and be courteous, non-judgemental, sensitive, gentle, and well prepared. An audio recorder is a common, useful tool for an interview. If interviewees agree to recording of the interview, the interviewer needs to focus less on taking notes and can pay more attention to the actual conversation. Moreover, interviewers can refer to the actual statements said during interview analysis, are less dependent on their notes and memory, and omission of details is less likely. Visual aids were experienced to be a useful tool too. Visual aids make it easier to ask follow-up questions related to them and interviewees might provide more rich and enlightening information. These findings coincide with our reasoning for the employment of the GUI prototype (preliminary draft), which is a form of visual aid. Interaction with the interviewee is susceptible to a few problems. If the interviewee is uncommunicative it is useful to rephrase questions so that they cannot be answered with 'yes' or 'no'. To raise confidence of interviewees, they should be ensured that there are no right or wrong answers. Shy interviewees might be motivated to talk by warming them up with topics they are interested in or letting them explain issues to the interviewer. Interviewees that talk to much and get off topic, on the other hand, should be gently directed back to topic. By either giving them cues like stopping to take notes or interjecting new questions when they pause, or by interrupting them if nothing else works.

Another influence on interaction with interviewees are the interview questions.[37] Questions related to behaviour or experience, where interviewees should describe how they do something, often lead to rich information. Reflexive questions that address opinions or values of interviewees by dealing with past experiences are likely to lead to useful information as well. Further helpful suggestions are that questions should not be too detailed because detailed questions can be difficult to answer and that questions should not assume conditions because if the assumption is not true, the question and follow-up questions are nearly useless. In Cohene, Easterbrook (2005) the successful application of two additional useful interview tactics is described. Hypothetical

---

[37] See regarding this and the following 3 sentences Hove, Anda (2005), p. 29-30.

experience questions are based on a fictional scenario.[38] A scenario provides context and creates shared meaning so that interviewees better understand the intention of the question and can be steered in the right direction which enables them to provide more valuable information. Additionally, hypothetical experience questions tend to reduce personal biasses because the interviewee can answer the question from the perspective of the hypothetical user in the scenario which reduces possible pressures to please the interviewer or to not offend him. Another tactic is protocol analysis that overcomes enunciative difficulties. Instead of being asked to describe how they would do something, interviewees are asked to show interviewers how they do it. This tactic can be useful to discover usability difficulties of the GUI prototype.

3.1.3 Interview Design

We developed the interview based on the recommendations, experiences, and characteristics described in chapters '3.1.1 Software Requirements Specification' and '3.1.2 Interview Characteristics'. The interview is divided into five sections. At first, the interview is initiated with introductory remarks that include introducing the interviewer, asking for permission to record the interview, describing the web application and the purpose of the interview, and gathering some general information on interviewees to be able to characterise the group of interviewees later on. The second section addresses the imagination of interviewees and is intended to elicit requirements without bias from familiarity with the GUI prototype. Interviewees are asked about their utilisation of package information leaflets and about inconveniences they associate with package information leaflets. Then, they are asked to describe how they picture the application based on the explanations in the first section and how they expect the application to facilitate the provision of information in patient information leaflets. In the third section, the GUI prototype is introduced and interviewees are asked to familiarise themselves with the GUI prototype. Simultaneously, the interviewer offers assistance and monitors the behaviour of the interviewees to identify problems. Hesitant interviewees are asked to perform simple tasks or to explain to the interviewer how they would accomplish a task in order to motivate them to become acquainted with the application.

---

[38] See regarding this and the following 4 sentences Cohene, Easterbrook (2005), p. 99-102.

Afterwards, in the fourth section, interviewees are asked to compare the GUI prototype to the expectations they had before they knew the GUI prototype and to elaborate on differences and characteristics they missed or would prefer to be different. In order to identify aspects of the prototype the interviewees do not like but did not mention before, a scenario that introduces a hypothetical patient that prefers using patient information leaflets over using the web application is provided. Based on this scenario, interviewees are asked for the most likely reasons of the hypothetical patient to prefer patient information leaflets over the web application. Then, ideas how the reasons for preferring patient information leaflets could be nullified are elicited. In the following, interviewees are asked whether they can think of further functionality that could be useful and to mention further useful functionality they know from other applications. Subsequently, interviewees are prompted for useful functionality concerning aggregation and refinement of information in patient information leaflets. Finally, interviewees are confronted with questions about quality aspects like reliability of the provided information and performance, availability and usability of the system. The fifth and final section is used for further questions, ideas, comments, and feedback that were not mentioned previously. Additionally, before concluding remarks from the interviewer interviewees are asked whether they would use the final fully implemented version of the application.

3.1.4 Interview Execution and Processing

We interviewed twelve people in eleven interviews. Elicited characteristics describing the group of interviewed people are plotted in figure 3-1. Interviewees can be classed in two age groups. The first group consists of people that are 45 to 50 years old and knew a world without the internet. Interviewees from age 20 to 23, which grew up with the internet, constitute the second group. This is useful to broaden different interviewee attitudes towards and experiences with the internet. In order to avoid gender-specific bias we interviewed an equal number of female and male interviewees. In general, interviewees had a positive attitude towards internet utilisation so that they prefer to use services provided over the internet. However, two interviewees stated that they use internet applications only reluctantly, which leads to another perspective on the application. The application to be developed is mainly influenced by the two disciplines 'Information Technology' and 'Health Care'. Besides being a potential user of the application, some interviewees had an occupational background in one of the disciplines

which provides potential to elicit more specific requirements or requirements that an interviewee without such expert knowledge would not think of. Interviewing more people would lead to more information and improve the results but for the development of the prototype the group of interviewees is diverse, adequate and large enough to provide sufficient applicable and feasible requirements.

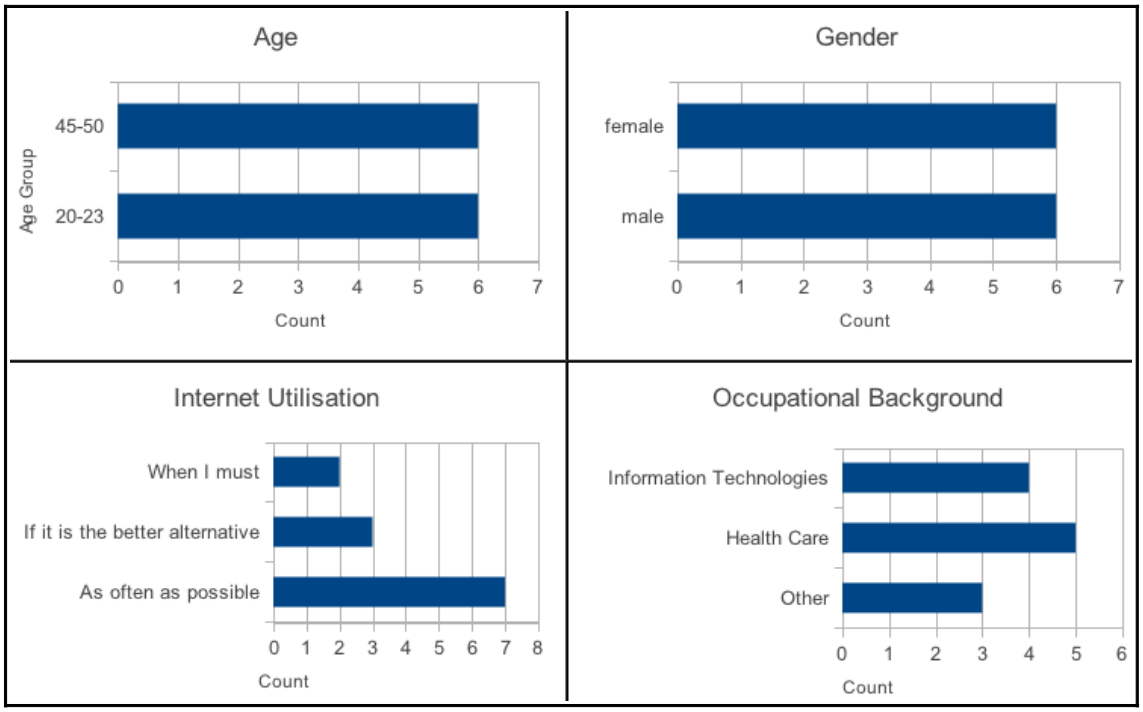

Fig. 3-1: Characteristics describing the group of interviewed people.

Interviewees were asked questions based on the structure mentioned in chapter '3.1.3 Interview Design'. However, the question sequence was often spontaneously adjusted because interviewees addressed subjects that were initially scheduled for a later phase of the interview. During the interview, an interview guide was used to support the interviewer. Its purpose was to depict the general structure of the interview and to suggest exemplary questions that could be rephrased to adapt to the needs of the interviewees. Additionally, an audio recorder was used to tape the interviews so that the interviewer could focus on the interview instead of taking notes. Notes were just made to capture ideas, remarks, requirements, or problems for later reference in the course of the interview and to capture relevant observations that would not be reflected on the tape. Besides higher focus on the interview, recording turned out to ease processing of the interviews because everything said could be replayed and interview processing depended less on recollection.

The interviews were processed by listening to the individual recordings and writing down the mentioned ideas and requirements for the web application. Thereby, similar ideas or requirements were directly consolidated. This led to one document for every interview listing the rudimentary requirements. We call these requirements rudimentary because by definition[39] they might not be exactly requirements and they are just loosely formulated. Each requirement is assigned a unique ID[40] and is further described in a description and a comment section. The description section describes the requirement and clarifies uncertainties, and the comment section lists comments of the interviewees like personal views or implementation suggestions. In the following, the rudimentary requirements were further refined and consolidated by merging similar requirements, improving their formulation, adding missing descriptions, and listing and grouping them in a single document that contains the final requirements. Moreover, a sources section that lists the IDs[41] of the rudimentary requirements on which a final requirement is based was added to the final requirements. As well as the rudimentary requirements the final requirements were assigned a unique ID[42].

3.2 Elicited Requirements

Since the objective of this thesis is the development of a prototype, requirements do not need to be handled very strictly. In the scope of this thesis requirements are rather ideas and features that would be beneficial for users than conditions/capabilities that the software has to fulfil/provide. Therefore, requirement formulations do not have to be very accurate as long as the ideas, expectations, or benefits elicited in the interviews are conveyed. Nevertheless, we tried to use unambiguous and meaningful formulations for the requirements.

Altogether, the list of final requirements, which can be found in the section 'Complete list of Elicited Requirements' of the appendix, is comprised of 141 requirements. Some of the elicited requirements do not fit the scope of the thesis so that they were removed

---

[39] See elaboration on requirements in chapter '3.1.1 Software Requirements Specification'.

[40] The format of the IDs of the rudimentary requirements is 'xx-yy'; 'xx' is the number of the interview and 'yy' is a sequential number counting the requirements per interview.

[41] To avoid confusion with IDs of final requirements IDs listed in the sources section were prefixed with 'INT-'.

[42] The format of the IDs of the final requirements is 'aa-bb-...-uu-vv'; 'aa' is the number of the main group, 'bb' is the number of the highest subgroup, 'uu' is the number of the lowest subgroup, and 'vv' is a unique number among the requirements and groups in the lowest subgroup.

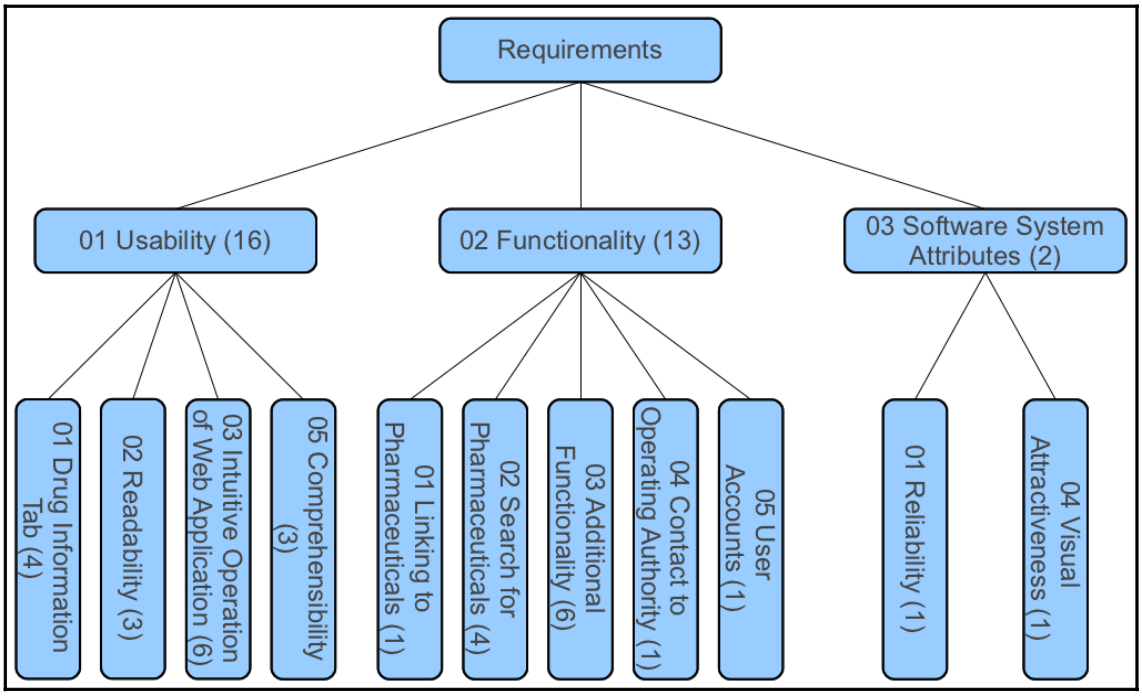

Fig. 3-2: Grouping of requirements which were mentioned at least four times (The numbers in front of the name identify the group and the number in brackets behind the name give the number of requirements in the group.)

from the list of requirements. These were requirements like an interface with an online pharmacy for purchase of pharmaceuticals, a search for suitable medical practitioners, or a directory of pharmacies. Removing out-of-scope requirements reduces the number of requirements to 132. In order to get a manageable amount of requirements, we additionally removed all requirements that were mentioned in less than four interviews because requirements mentioned more often by potential users are probably important to more users than requirements mentioned less often. Removing the requirements that have only a few sources left us with a controllable set of 31 requirements. Figure 3-2 depicts the tree structure of the groups containing the 31 remaining requirements, which will be described in the following.

3.2.1 Usability

Requirements in the group 'Usability' are intended to improve ease of use of the application by, for example, increasing readability or comprehensibility. These requirements are arranged in four subgroups 'Drug Information Tab', 'Readability', 'Intuitive Operation of Web Application', and 'Comprehensibility'.

Drug information tab[43] refers to the tab that provides the information on pharmaceuticals. Available alternative dosage forms of a pharmaceutical represent additional useful information that the drug information tab should provide. This, for example, would be useful if patients had problems with swallowing a pill so that they would rather dissolve the pharmaceutical in a fluid and drink it. Locating of desired information in the drug information tab would be improved by providing a way to jump directly to sections of information. This could be achieved by providing links to the sections of information at the top of the drug information tab or in a side bar. A disadvantage of package information leaflets is that they have to provide information which is irrelevant for some patients. The web application should not have the same drawback and provide only the information the user needs. Side effects are such information that many patients are not interested in until they occur. Therefore, the web application should offer functionality to filter side effects by frequency of occurrence so that user can, for example, select only side effects that occur very often or often for display.

ID	Requirement
01-01-04-02	Display alternative dosage forms of a pharmaceutical.
01-01-05	Enable the user to go to a specific section of drug information.
01-01-07	Do not display more information than the user needs.
01-01-07-02	Allow users to filter side effects by frequency of occurrence.

Tab. 3-1: Requirements: 'Drug Information Tab' subgroup

Readability is another object for consideration to achieve a patient-friendly web application. An often requested feature to improve readability is the adjustment of the font size. With the interviews, we rebutted the assumption in chapter '2.2.2 GUI Prototype Design Decisions' that browser zoom capabilities suffice for the adjustment of the font size because most interviewees were not aware of the zoom functionality offered by browsers. Accordingly, the web application needs to provide an easy and intuitive way for adjustment of the font size, like listing examples of the available font size. Another requirement is that the text should have a clear structure, layout, and markup. This entails, for example, a consistent layout, splitting long text segments in bullet points, and listing sections of information below instead on the side of each other. Accentuation of the sub-headings in the drug information tab was identified as a further

[43] The drug information tab is depicted in figure 2-3 on the left.

measure to increase clarity of the structure. The term 'sub-heading' refers to the headings of the sections of information in the drug information tab like side effects or contraindications. These could, for example, be accentuated with fitting icons or another colour.

ID	Requirement
01-02-01	Allow for adjustment of font size.
01-02-02	Clear structuring of text.
01-02-02-01	Accentuate sub-headings in drug information tab.

Tab. 3-2: Requirements: 'Readability' subgroup

To improve usability, operation of the web application should be intuitive. When using desktop applications like a mail application, users normally do not mind spending some time on configurations like setting up different connections for their mail accounts or configuring mail filters. When using web applications, on the other hand, users normally expect more intuitive operation. The Google search engine[44] is a perfect example for intuitive operation because users only have to enter a search string and get fitting results. If necessary, one can also use advanced search options or spend some time studying manuals[45] to learn a dedicated search language for more potent search requests. In the interviews we discovered that different users prefer different GUI designs. Inexperienced users seem to prefer a simple GUI, which provides only the most important controls and settings. On the contrary, more experienced users prefer fast access to the available functionality and do not mind a GUI that displays more settings and controls at once. To satisfy both preferences and to avoid confusion of inexperienced users, the GUI should hide all functionality except the most important controls and settings. Additionally, more experienced users should be enabled to display further functionality. Inexperienced users should be further assisted with an optional guide for the retrieval of information on pharmaceuticals. Instead of letting the users provide the input and configure the options for the output by themselves, the guide should ask them for required input, their preferences for the output and reduce the amount of initiative required from the user. Further assistance with the operation of the web application should be offered by a start page that outlines the purpose and basic operation of the application in a succinct way.

---

[44] The Google search engine can be accessed with the URL 'www.google.com'.

[45] See 'http://support.google.com/websearch/?hl=en' for a collection of manuals for the Google search engine.

Further requirements related to the operation of the web application refer to insufficiencies of the GUI prototype. The input field for the basic search needs to be clearly noticeable. In the GUI prototype, as depicted in the top left corner of figure 2-2, the input field is located below the list-select for pharmaceuticals. At this position many interviewees overlooked it or thought it was a filter for the list-select. All objects that can be closed need to indicate where the user can click to close them. A pop-up in the GUI prototype misses such indication. This confused some users and they did not know where to click to close the pop-up. Moreover, some more intuitive functionality for creation of new tabs needs to be found. While many interviewees deemed tabs a useful feature, they did not understand how tabs could be created and the interviewer had to explain it. Therefore, it is necessary to find a way for intuitive creation of new tabs that still allows users who do not want tabs to use the application without tabs.

ID	Requirement
01-03-01-02	Hide functionality that might confuse inexperienced users.
01-03-01-02-04	Guide inexperienced users through the application rather than offering them many different controls and settings.
01-03-01-03	Make the search field in panel 'Medikamenten-Auswahl' clearly noticeable.
01-03-03	Display close buttons/icons for everything that can be closed.
01-03-04-01	Intuitive functionality for creation of new tabs.
01-03-06-01	Display introductory information at application startup.

Tab. 3-3: Requirements: 'Intuitive Operation of Web Application' subgroup

Requirements related to the comprehensibility of the provided information are grouped in the subgroup 'Comprehensibility'. An important feature is the provision of explanations for technical terms because the information on pharmaceuticals contains many words that people who are not medical professionals are not familiar with. Accordingly, these words inhibit the comprehensibility of the provided information so that they need to be explained in a comfortable way. Consequently, explanatory text, like the tooltips for buttons, should be in common wording as well. For example, instead of using the English term 'Header' the German term 'Kopfzeile' should be used when referring to the header. Furthermore, usage of abbreviations should be avoided because they might be ambiguous and complicate the mental processing of the provided information. In patient information leaflets abbreviations might be a viable option due to space restrictions but a web page has no space restrictions so that usage of abbreviations is more bothersome than useful.

ID	Requirement
01-05-01	Use common wording in explanatory text.
01-05-02	Avoid abbreviations in text.
01-05-03	Explain technical terms.

Tab. 3-4: Requirements: 'Comprehensibility' subgroup

3.2.2 Functionality

The group 'Functionality' contains requirements that are concerned with the functionality the web application should offer. The group is divided in the five subgroups 'Linking to Pharmaceuticals', 'Search for Pharmaceuticals', 'Additional Functionality', 'Contact to Operating Authority', and 'User Accounts'.

The subgroup 'Linking to Pharmaceuticals' holds requirements that deal with functionality to access information on pharmaceuticals that have a relationship with the currently displayed pharmaceutical. Only one requirement in this group was mentioned in more than three interviews: When viewing a pharmaceutical users should be able to select pharmaceuticals with the same active pharmaceutical agent as the currently displayed pharmaceutical. This could be accomplished by, for example, providing a list of pharmaceuticals with the same active pharmaceutical agent when the user clicks on the active pharmaceutical agent displayed in the drug information tab. Such functionality would be useful to find cheaper alternatives to a pharmaceutical or to find a pharmaceutical with different further ingredients.

ID	Requirement
02-01-01-01	Display pharmaceuticals with same active pharmaceutical agent as the currently displayed pharmaceutical.

Tab. 3-5: Requirements: 'Linking to Pharmaceuticals' subgroup

Requirements specifying functionality to search for pharmaceuticals are grouped in the subgroup 'Search for Pharmaceuticals'. Required search parameters are the name, field of application, and the active pharmaceutical agent of the pharmaceutical. A field of application describes medical conditions, like for example fever, in which a pharmaceutical is appropriate for treatment. Searching for a pharmaceutical by name could be problematic if users do not know the exact name of a pharmaceutical. To solve this problem, the application should also provide a way to find a specific pharmaceutical

without knowing the exact name. If the user does not know other specifics like the field of application or the active pharmaceutical agent, the user could be assisted by a list that contains all available pharmaceuticals.

ID	Requirement
02-02-01-01	Search for pharmaceuticals by field of application.
02-02-01-02	Search for pharmaceuticals by name.
02-02-01-04	Search for pharmaceuticals by active pharmaceutical agent.
02-02-03-01	Find specific pharmaceuticals without knowing the exact names.

Tab. 3-6: Requirements: 'Search for Pharmaceuticals' subgroup

The title of the group 'Additional Functionality' refers to functionality beyond the access to and provision of information in patient information leaflets. This functionality leverages the capabilities of a web application and is important to distinguish the web application from patient information leaflets which cannot offer such functionality. Obviously, aggregation of information on multiple pharmaceuticals is functionality that can hardly be achieved with patient information leaflets. In the interviews, we elicited a few requirements for functionality that aggregates information on multiple pharmaceuticals. Users desire a feature that checks a set of pharmaceuticals for adverse drug reactions. This would require the user to specify a set of pharmaceuticals and the application would then check the specified pharmaceuticals for known adverse drug reactions and indicate whether the pharmaceuticals can safely be taken with each other, should not be taken with each other, or whether taking these pharmaceuticals with each other requires some precautions. Further functionality based on a set of pharmaceuticals is the listing of all side effects of multiple pharmaceuticals. Such a feature could be useful to check whether symptoms a user is experiencing, like fatigue, can be associated with a pharmaceutical the user is taking.

Moreover, interviewees mentioned functionality to compare pharmaceuticals by price. Such information could be useful to find the cheapest pharmaceutical in cases where pharmaceuticals that do not need prescription, like a headache tablet, are required. Another feature for comparison of similar pharmaceuticals is desired as well. It should be possible for users to select two similar pharmaceuticals for comparison. The application should then offer some functionality to aid users in comparing the selected pharmaceuticals. This could be useful to establish characteristics in which two pharmaceuticals that seem identical differ so that users can decide which one is more appropriate. Another problem mentioned by the interviewees, for which problem

solving could be assisted by the application, is the identification of pharmaceuticals. If users, for example, keep pharmaceuticals in a pill box or remove a blister from the packaging and find such pharmaceuticals later on, they might not be able to identify the pharmaceutical. To provide assistance in such cases, the web application should provide images of the pharmaceutical in its dosage form, the blister or other inner packaging, and the outer packaging so that users can guess the kind of pharmaceutical and verify their guesses through comparison with the provided images. Integrating images in the drug information tab would also make the application look more lively and less text heavy. The interviewees stated additionally that it would be useful to print the selected information of the currently displayed pharmaceutical. This would be useful to create a sort of personal patient information leaflet that provides only the required information and has a satisfactory font size. Obviously, the print out should only contain the information on the pharmaceutical and not GUI elements.

ID	Requirement
02-03-01-01	Check a set of pharmaceuticals for adverse drug reactions.
02-03-01-02	List all side effects of a set of pharmaceuticals.
02-03-01-04-01	Offer price comparison of similar pharmaceuticals.
02-03-01-04-02	Compare similar pharmaceuticals.
02-03-02	Support identification of pharmaceuticals.
02-03-05	Provide functionality to print the selected information of currently displayed pharmaceutical.

Tab. 3-7: Requirements: 'Additional Functionality' subgroup

In the interviews, we found out that users desire some way to contact the operating authority of the web application; this is expressed in the subgroup 'Contact to Operating Authority'. Accordingly, some functionality to get in touch with the operating authority needs to be implemented and the users should be informed how they can use this functionality. Interviewees deemed such functionality useful for tasks like getting help with the application, giving feedback on the application, or reporting errors found in the provided information.

ID	Requirement
02-04-01	Inform users how the owner/operating authority of the web application can be contacted.

Tab. 3-8: Requirements: 'Contact to Operating Authority' subgroup

Requirements categorised in the subgroup 'User Accounts' are related to functionality requiring user accounts. As in the subgroup 'Contact to Operating Authority', only one requirement of the 'User Accounts' subgroup was mentioned in at least four interviews: The application should allow users to post their experiences with the application or specific pharmaceuticals, and to access experiences provided by other users. This functionality requires user accounts to inhibit spam at least a little and to increase motivation for users to post correct information. A forum or a comment section could be a possible implementation of this requirement. Realisation of this requirement would, for example, be beneficial for users to get questions answered, to get in contact with people in similar situations, or to decide whether the application is trustworthy.

ID	Requirement
02-05-02	Provide experiences of other users with the application or specific pharmaceuticals.

Tab. 3-9: Requirements: 'User Accounts' subgroup

3.2.3 Software System Attributes

In contrast to the groups 'Usability' and 'Functionality', only two requirements related to software system attributes were mentioned in more than three interviews. We already addressed software system attributes in a dedicated interview section but the responses were rather general remarks on software system attributes of web applications than requirements. Interviewees expect web applications to respond quickly to their requests but seem to be used to occasional latency caused by their internet connection. Availability of web applications needs to be high so that users can access web applications whenever they want. Operation of web applications needs to be intuitive so that user can work with a web application right away without having to study a manual or tutorial. A manual is seen as useful to discover advanced functionality once users are satisfied with the basic functionality. However, as already mentioned above, at application startup some users expected a short introduction to get an idea about the purpose of the application and to get very basic instructions. Only a few interviewees mentioned security and reliability concerns with the provided information by themselves. Anyhow, when such concerns were directly addressed interviewees expected the application to provide reliable information, to abstain from profiling, and to avoid misuse of their provided information. Methods mentioned to increase perceived reliability are a professional appearance, provision of information that is consonant with

information provided by patient information leaflets, no collection of unnecessary data, patronage of some reliable organisation, and simply stating that data privacy will be honoured. The latter was stated in more than three interviews. Users should be informed that they will not be profiled while using the application and that collected data will only be utilised by the web application and not misused. Another desired characteristic of the application is a design that is more diversified, attractive and appealing than the GUI prototype design while still maintaining a professional and respectable appearance. Interviewees do not demand a major change of design but some minor changes that liven up the design.

ID	Requirement
03-01-02-02	Inform users that they will not be profiled and that collected data will not be applied beyond the scope of the application.
03-04-02	Make GUI design more diversified/attractive/appealing

Tab. 3-10: Requirements: 'Software System Attributes' group

3.2.4 Concluding Remarks

In the following, we will refer to the 31 requirements presented above with the term 'relevant requirements'. The relevant requirements answer the first research question: What are the requirements of a web application providing aggregation and refinement of information in patient information leaflets? The relevant requirements reveal the requirements of a web application providing aggregation and refinement of information in patient information leaflets. Obviously, the 31 requirements do not represent a complete list of requirements for a web application providing aggregation and refinement of information in patient information leaflets; with the interviews alone, we elicited more than a hundred further requirements. However, the 31 requirements were mentioned in more than a third of the interviews. Thus, it is likely that out of all elicited requirements these requirements are the requirements that are important to a large subset of all potential users. Therefore, they should serve as a good basis for the implementation of the prototype targeted in this thesis. It needs to be taken into consideration that the relevance assessments of requirements 01-01-04-02, 02-02-01-01, and 02-03-01-02 might be biased because we used similar requirements as examples during the interviews. Nevertheless, we decided to keep these requirements because users could still make up their own mind about the usefulness of the requirements.

Furthermore, implementing a requirement that users deemed useful is more reasonable than refraining from a requirement because of possible bias which might be insignificant.

Remaining requirements that were not listed above and are not beyond the scope of this thesis will be taken into account if they give particulars on requirements to be realised, or help solving problems or uncertainties that occur during design/implementation. They might be useful for later increments of the web application so that they are listed in the section 'Complete list of Elicited Requirements' of the appendix, but in this thesis they will not be processed any further.

3.3 Requirements Evaluation

All relevant requirements[46] fit the definition of requirements because they represent “a condition or capability needed by a user to solve a problem or achieve an objective”[47] and feature the characteristics of good requirements mentioned in chapter '3.1.1 Software Requirements Specification'. They are correct because they were elicited from potential users so that the required capabilities or conditions are actually needed. Meaningful phrasing and the associated descriptions prevent ambiguity. The requirements are complete because they describe the top level functionality the prototype should provide and necessary associated information. Since the requirements do not contradict each other and do not refer to the same real-world object with different terms, the requirements are consistent. Although some requirements are not quantifiable, in the scope of this thesis the requirements can be considered verifiable. For the development of the prototype, the requirements need to indicate ideas and expectations of potential users so that they do not necessarily need to be quantifiable. It must be possible to check whether a requirement was taken into account and implemented, but there is no especial need to verify whether a requirement has been met exactly. The requirements are traceable as well because they have a unique identifier and reference their sources. However, since the sources provide no relevant information they are not specified in this thesis; only the number of sources for each requirement, which were used to filter the requirements, are specified in the appendix. Furthermore, the relevant requirements address the targeted aspects of the web application: user interface/usability, functionality

---

[46] A list containing all relevant requirements can be found in the appendix.

[47] IEEE (2010), p. 301.

and software system attributes. Only two requirements regarding software system attributes is a little sparse, but, as described in '3.2.3 Software System Attributes', the interviews pictured general expectations of the software system attributes.

3.3.1 Problematic Requirements

We established that the relevant requirements fit the definition of requirements and sufficiently embody good characteristics of requirements. Nevertheless, checking the relevant requirements for potential problems they might cause in the future led to the decision to abandon two further requirements.

Requirement 02-03-01-04-01 states that the application should provide a feature that enables users to compare similar pharmaceuticals by price. Such functionality is quite useless for pharmaceuticals that require prescription because in these cases physicians and health insurance companies decide which pharmaceutical a patient gets. If pharmaceuticals require no prescription other factors like availability, lead time, or accustomed sources of supply might be more important than the price of the pharmaceutical. However, the main reason for abstaining from provision of price information is the need of a high quality data base. In order to provide good, complete and nicely formatted information on pharmaceuticals we need a data base that holds the required data. Without a medical background single-handedly creating such a data base is error prone and it might be problematic to get access to the required information. It would be advantageous if the information was provided by medical professionals, ideally, by the manufacturers and pharmaceutical companies that are already responsible for patient information leaflets. This would also increase the perceived reliability of the web application due to the fact that the information provided by the web application is provided by the same organisation that provides the information in patient information leaflets. Providing price information for pharmaceuticals would probably reduce the interest of pharmaceutical companies to support the application. A neutral platform that provides only information on pharmaceuticals and no subliminal recommendations, which would be the case if price information were provided, seems to be more likely to attract support of pharmaceuticals companies so that we decided to not provide a price comparison of pharmaceuticals and to remove requirement 02-03-01-04-01.

Requirement 02-05-02 is the other requirement that fell victim to our check for potential problems. Kaletsch, Sunyaev (2011) points out that serious trouble can be caused for patients if unauthorised persons gain access to their health information.[48] Especially social functions can breach a patient's privacy and are considered a top threat: Patients might expose sensitive information on their own or provide wrong information, which could cause harm to other patients. Additionally, with case studies of systems for the management of personal health records Kaletsch and Sunyaev established that no analysed system was unrestrainedly recommendable. Therefore, we decided to refrain from implementing functionality that requires user accounts in order to avoid collection of sensitive data. It is better to store sensitive data in an application that is specifically designed for that purpose and to interface such application to provide functionality that requires user accounts. The German healthcare telematics infrastructure, which is currently being established, in combination with the electronic health card, which is currently being released to all German citizens, would be a suitable candidate. Similar to our prototype it is designed for use in Germany and not subject to foreign privacy laws. All German citizens will have access to the telematics infrastructure and can use the electronic health card as unique identifier. Consequently, we decided not to implement functionality that requires user accounts until recommendable applications for management of personal health records exist. Requirement 02-05-02 is especially problematic because it asks for social functionality where patients can exchange personal experiences. Using this functionality could bring them to unintentionally provide sensitive or false information.

3.3.2 Elicited Requirements versus Objectives for Preliminary Draft

Abandoning requirements 02-03-01-04-01 and 02-05-02 leaves us with 29 relevant requirements for the further design and implementation of the prototype. Mapping the relevant requirements to the objectives for the preliminary draft[49] brings out that most elicited relevant requirements are related to functionality for the aggregation and refinement of information in patient information leaflets. In the GUI prototype we implemented/insinuated only some basic functionality for the aggregation and refinement of information in patient information leaflets because we deemed such

---

[48] See regarding this and the following 2 sentences Kaletsch, Sunyaev (2011), p. 2, 5-6.

[49] A mapping of the relevant requirements to the objectives for the preliminary draft can be found in the appendix: 'Mapping of relevant requirements to objectives for preliminary draft'.

functionality too dependent on user preferences. Thus, it is an advantageous result that we elicited many requirements that illustrate the aggregation and refinement functionality. The other objective that seemed to be in need of user input to improve the achievement status was concerned with the reporting of wrong information. However, the interviews showed that for reporting of wrong information users are satisfied with functionality to contact the operating authority of the web page. As already mentioned in '3.2.3 Software System Attributes', the application design objectives were not further specified with requirements. Interviewees expect the web application to behave like any other popular web application and mentioned no special requirements regarding the application design objectives.

The relevant requirements uncover some deficiencies of the GUI prototype as well. These aspects were either not considered in the objectives for the preliminary draft or wrong assumptions were made. In contrast to our prior considerations, the interviews showed that the zoom functionality of the browser is not sufficient for the adjustment of the font size because many potential users do not seem to know this feature. Functionality for creation of tabs needs to be more intuitive. The search field for the basic search needs to be in a more prominent position. Some users prefer a short introduction and instructions at application startup. Additionally, it should be possible to print the information provided on pharmaceuticals. Otherwise, the objectives for the preliminary draft were affirmed by the elicited requirements and their realisation further specified. Required changes and extensions of the GUI prototype do not seem to be too severe and contradictory so that it should be possible to reuse much code of the GUI prototype.

4 Conceptual Design

Purpose of this chapter is to draw up a system design of the prototype. The resulting design will serve as a basis for the implementation and as a foundation for the description of the implementation. However, we will not go into details and draft every last bit of the application. The resulting design should just provide a general understanding of the application, demonstrate how the application performs its tasks and serve as guiding framework for the implementation. Detailed descriptions for important aspects of the application will be provided in chapter '5 Implementation'. Thus, only the basic architecture of the application is described in this chapter. Additionally, we will describe technologies used for running the application or implementing the application because those influence the design and provide/aid in providing required functionality. Furthermore, we will point out limitations of the design or abstractions made due to the fact that we are developing a prototype and not an application intended for live operation.

4.1 Vaadin

We already used Vaadin for the implementation of the GUI prototype and in the interviews we got no negative feedback that concerned Vaadin or gave a reason to refrain from using Vaadin. Furthermore, using Vaadin also for the development of the application eases reuse of code that was written for the GUI prototype. Therefore, we are going to use Vaadin for the development of the application as well. Since using Vaadin has a great influence on application design Vaadin is described first. In chapter '2.2.1 GUI Prototype', we have already provided a short description of Vaadin and listed a few additional benefits of using Vaadin: abstraction of web technologies and associated browser compatibility issues, increased security, and easy integration of further components. Figure 4-1 depicts the general architecture of Vaadin. In the following we will provide a more detailed description of the Vaadin framework by explicating the architecture components depicted in figure 4-1 from left to right.

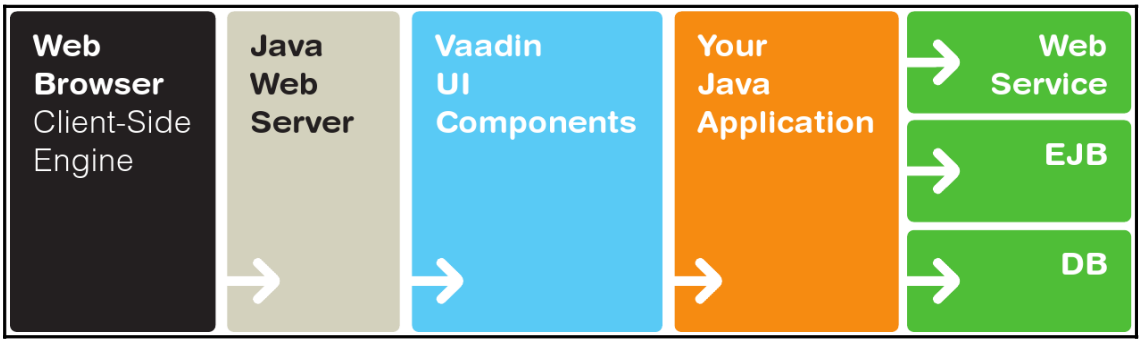

Fig. 4-1: General architecture of Vaadin[50]

The client-side engine uses the Google Web Toolkit (GWT[51]) to render the user interface in the web browser of the user.[52] GWT is a software development kit for the implementation of widgets in Java, which are compiled to JavaScript, so that they can be rendered in the browser. All user interface components (widgets) provided by Vaadin are implemented with GWT. Capturing of user interaction and rendering the widgets composing the user interface according to application state are tasks of the client-side engine. User interactions with the widgets that compose the user interface are initially captured by the GWT widgets and then relayed to the server side by the client-side engine. Correspondingly, responses from the server side are relayed to the GWT widgets by the client-side engine so that the user interface reflects the application state. As long as the widgets provided by Vaadin are sufficient and no additional GWT widgets need to be implemented or integrated, developers do not need to concern themselves with the client-side engine because client-server-side communication and appropriate rendering of the user interface is handled by Vaadin.

Java Web Server, the next architecture component, is the platform on which the application is executed. Vaadin applications can be deployed as Java web applications. A Java "web application is a collection of servlets, html pages, classes, and other resources that make up a complete application on a web server"[53]. "A servlet is a Java technology based web component, managed by a container, that generates dynamic content. Like other Java-based components, servlets are platform independent Java classes that are compiled to platform neutral bytecode that can be loaded dynamically

---

[50] Grönroos (2011), p. 4.

[51] See 'http://code.google.com/webtoolkit' for more information on GWT.

[52] See regarding this paragraph Grönroos (2011), p. 30-35, 230.

[53] Sun Microsystems (2001), p. 59.

into and run by a Java enabled web server"[54]. Therefore, the application can be executed on any application server that supports the version of the Java Servlet API used by Vaadin[55] and has access to a compatible Java runtime environment (JRE). Possible application servers are, for example, Apache Tomcat[56], which we will use to deploy the application, or Jetty[57], which we use during development because of its comfortable integration with Eclipse[58].

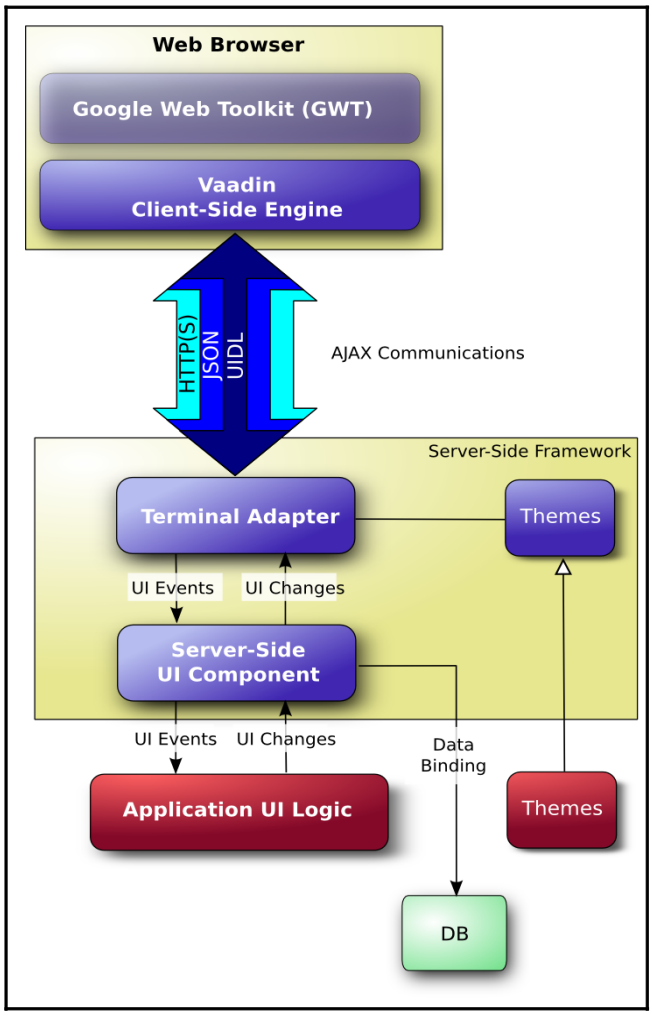

Fig. 4-2: Client- and server-side Vaadin architecture[59]

As stated before, the Vaadin user interface (UI) components are used to compose the user interface. The previously described UI components that are handled by the client-side engine have a counterpart on the server side, as depicted in figure 4-2, which illustrates the Vaadin architecture in more detail than figure 4-1. User events registered on the client side are relayed to an abstraction layer called Terminal Adapter and the corresponding changes of the user interface are sent from the Terminal Adapter to the client side.[60]

---

[54] Sun Microsystems (2001), p. 17.

[55] Vaadin 6.7.0 supports Java Servlet API 2.3 (https://vaadin.com/features).

[56] See chapter '4.3.1 Tomcat' or 'http://tomcat.apache.org/' for more information on Apache Tomcat.

[57] See 'http://www.eclipse.org/jetty/' for more information on Jetty.

[58] Eclipse offers among other things an integrated development environment which we used for the development of the web application. See 'http://www.eclipse.org/home/newcomers.php' for more information on Eclipse.

[59] Grönroos (2011), p. 30.

[60] See regarding this sentence and the rest of this paragraph Grönroos (2011), p. 30-31.

Communication is accomplished by asynchronous HTTP[61] or HTTPS[62] requests using the JSON[63] based User Interface Definition Language (UIDL[64]). User events relayed from the client side to the Terminal Adapter are forwarded to the corresponding server-side UI components and, accordingly, the changes made by server-side components are interpreted by the Terminal Adapter and communicated to/realised by the client-side engine. A benefit of the Terminal Adapter is that it handles all communication with the client side so that the client side could be implemented with completely different technology and the application, which only communicates with the Terminal Adapter, would still work.

The Java web application we will develop runs on the server and uses a set of Vaadin UI components to provide the GUI. Furthermore, the application provides the functional logic and access to databases or other useful services.

4.2 Application Modules

The three main modules 'Graphical User Interface', 'Application Logic', and 'Data Access' that form the application are depicted in figure 4-3. A module is “a logically separable part of a program”[65]. With the term 'main module' we refer to the top-level modules that are necessary to perform the tasks of the application.

Functionality that is not GUI related and does not directly interact with the database is provided by the module 'Application Logic'. This module bundles auxiliary functionality like, for example, consolidation of results from multiple SQL statements or compilation of SQL statements based on inputs made by the user. Another example would be the processing and layout of information required for printing. Additionally, more complex routines for tasks like checking a set of pharmaceuticals for adverse drug reactions or comparing information provided on similar pharmaceuticals will be

---

[61] The HyperText Transfer Protocol (HTTP) is the basic protocol for data communication in the world wide web. See 'http://tools.ietf.org/html/rfc2616' for more information.

[62] The HyperText Transfer Protocol Secure (HTTPS) combines HTTP with encryption to provide encrypted communication. See 'http://tools.ietf.org/html/rfc2818' for more information.

[63] The JavaScript Object Notation (JSON) is a lightweight, human- and machine-readable data-interchange format. See 'http://www.json.org/' for more information.

[64] The User Interface Definition Language (UIDL) is used to create (describe) complex user interfaces and to communicate asynchronously with the server side. See 'http://www.uidl.net/doc/' for more information.

[65] IEEE (2010), p. 223.

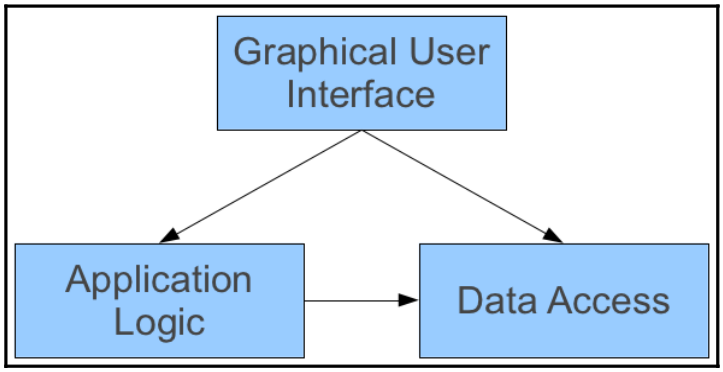

Fig. 4-3: Application Modules and their relationships. (The module to which the arrow points is used by the module from which the arrow originates.)

provided by the 'Application Logic' module. Consequently, the module provides functionality required by the 'Graphical User Interface'. Some routines implemented in the 'Application Logic' module need access to the database and, therefore, need to employ functionality provided by the 'Data Access' module. A module that concentrates the application logic is useful because, otherwise, interchangeable functionality is easily implemented redundantly in multiple classes.

Similar to the 'Application Logic' module the 'Data Access' module does not need functionality provided by the 'Graphical User Interface' module and provides functionality required by both other modules. In order to provide information on pharmaceuticals we have to retrieve the information somehow. Hence, we will store the required data in a database. Access to the database will be provided by the module 'Data Access'. This module will consist of a set of classes that retrieve required data from the database and perform simple preprocessing of the data so that it can be directly processed by classes of the other modules. Preprocessing jobs that are employed in multiple cases or are laborious constitute application logic and, accordingly, they will be handled by the 'Application Logic' module. Handling access to the database in a dedicated module is beneficial because it eases the handling of database related code. Other modules or classes do not need individual implementations for database access and database related code is stored in one place, which reduces coding and maintenance effort. Moreover, common functionality can be easily reused and performance gains might be achieved through a reduction of SQL connections and bundling of SQL statements.

The user interface is provided by the module 'Graphical User Interface'. Since we will use Vaadin and there is no immediate need for modifications on the client side, classes that implement the server-side UI components constitute the 'Graphical User Interface' module. These classes adopt and extend the UI components provided by Vaadin so that they fit the needs of the application. Moreover, the UI components are combined to provide side bars for navigation or settings, and individual tabs that provide, for example, search results, search input fields, or information on pharmaceuticals. Additionally, the classes implementing the GUI need to handle user events and forward these to the respective classes or adopt the GUI accordingly. Application behaviour is controlled by the user through UI events. Accordingly, the 'Graphical User Interface' module does not provide functionality for the other two modules, instead, the other two modules are needed to process the requests made by the users and provide appropriate results.

We decided to use a modular architecture because it can be easily extended and adapted. For example, if we were to offer a dedicated user interface for mobile devices in the future, we could implement additional modules implementing the user interfaces for the various platforms like Android or iOS[66] and still use the 'Application Logic' and 'Database Access' modules to handle the user events. Additionally, we could easily use another database. A change of the database just requires an adoption of the 'Database Access' module so that it can access the new database and retrieve the required data. As long as the other modules still require the same data they can remain untouched.

---

[66] Android and iOS are operating systems for mobile devices, which are presently popular and widely used.

4.3 Used Technologies

In order to ease implementation and deployment of the application we utilised some subsidiary technologies. Vaadin was already introduced in chapter '4.1 Vaadin'. Further used technologies will be described in the following.

4.3.1 Tomcat

As described in the chapter '4.1 Vaadin', Vaadin applications need a Java web server on which they are executed. We decided to use Apache Tomcat, which is such a Java web server and is easy to use. For local testing one basically only needs to copy the Tomcat directory to a folder, paste the .war-archive containing the application and start the server. Still, Tomcat can be configured[67] for productive use and offers, for example, settings for encrypted communication or a manager for deployment/undeployment of applications during live operation. As already elaborated in chapter '2.2.2 GUI Prototype Design Decisions', Tomcat is scalable because it can be run in a cluster, which also increases availability of the application because if a node fails another node will take over. Moreover, Tomcat is widely used and offers elaborate documentation so that it is easy to obtain help in solving problems concerning Tomcat.

4.3.2 MySQL

MySQL is a relational database management system[68] which we will use to store and retrieve the information on pharmaceuticals. We will use MySQL because it is sufficient for our needs since it can be used to store arbitrary data in a self-defined system of relations/tables. Additionally, as mentioned in chapter '2.2.2 GUI Prototype Design Decisions', MySQL is scalable because it is simple to run it in a cluster. Furthermore, MySQL is widely used and well-documented so that it is easy to obtain help when encountering problems with MySQL. MySQL can be obtained without a licensing fee and as open source so that its source code can be viewed and improved by everyone.[69]

---

[67] See the Apache Tomcat 7 documentation (http://tomcat.apache.org/tomcat-7.0-doc/index.html) for more information on Apache Tomcat.

[68] See 'http://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm' for a short tutorial that explains relational database management systems.

[69] See regarding this and the following 2 sentences Oracle (2011), p, 3-4, 6, 2756.

Establishment of a connection between Java and MySQL is straightforward. Moreover, the database can be comfortably configured with a graphical administration interface. There are certainly other database management systems as suitable as MySQL, but since MySQL is sufficient for our needs we decided to use MySQL.

4.3.3 Hibernate

Hibernate[70] is a Java library that provides object-relational mapping. Object-relational mapping refers to associating (Java) objects with tables stored in a relational database. This is, for example, useful to persist objects, that is storing objects in a relational database when the application exits and restoring them in memory at application startup. Persisting of objects is, for example, beneficial to keep the application state across multiple sessions. However, for the prototype it is not necessary to persist objects, but mapping objects to tables in the relational database is useful. It is more comfortable to work with Java objects and having Hibernate handle communication with the database than importing data with SQL queries. Additionally, it is easy to change to another database. In such situations, only the XML files that map the objects to tables and the connection information in the Hibernate configuration XML file would require adopting. All application logic could remain untouched. This decreases dependence on the database. Besides improving expandability through decreased dependence on a specific database, the Hibernate query language (HQL) is quite similar to SQL, which makes Hibernate easy to use if one already knows SQL.

4.4 Physical Infrastructure

As depicted in figure 4-4, clients access the application over the internet. Clients can access the application from any device with an AJAX-capable browser. We have already addressed the handling of scalability and availability issues in chapter '2.2.2 GUI Prototype Design Decisions'. When a client starts the application a session is initiated on a Tomcat application server. In order to service many clients simultaneously and to match the available computing power to a changing number of clients the application runs on a cluster of Tomcat servers. Each node in the cluster hosts the servlet and a load balancing node redirects client requests based on a load balancing strategy like, for

---

[70] See 'http://www.hibernate.org/' for detailed information on Hibernate.

example, redirecting clients to the node with the lowest load. By adding further nodes to or removing nodes from the cluster the available computing power can be matched to the demand of clients. Additionally, the load balancer avoids situations in which some nodes are overburdened while others are idling. Availability is increased through session replication so that sessions and tasks of a crashed node can be handled by other nodes. The Tomcat nodes access the database over a private network. Similar to the Tomcat cluster, database services are provided by a Linux Virtual Server (LVS) to meet scalability and availability needs. The LVS consists of multiple physical servers that host a replicated version of the MySQL database. One server serves as a master and processes write requests, which are replicated to the other servers. The other servers process read requests, which, in our case, represent most accesses to the database. Based on the current requirements, write accesses are only necessary to update the provided information and not required for the handling of client sessions. Hence, one server is sufficient to handle database write operations. If a future extension requires more write operations initiated by users, like storage of personal settings in the database, than one server can handle, it will be possible to adopt the replication strategy accordingly. With the exception of redirecting write accesses to the master version of the database the load balancer of the LVS works like the load balancer of the Tomcat cluster. Scalability is facilitated through adding or removing of physical servers and availability is ensured to a high degree because SQL statements assigned to crashed servers can be handled by available instances of the database.

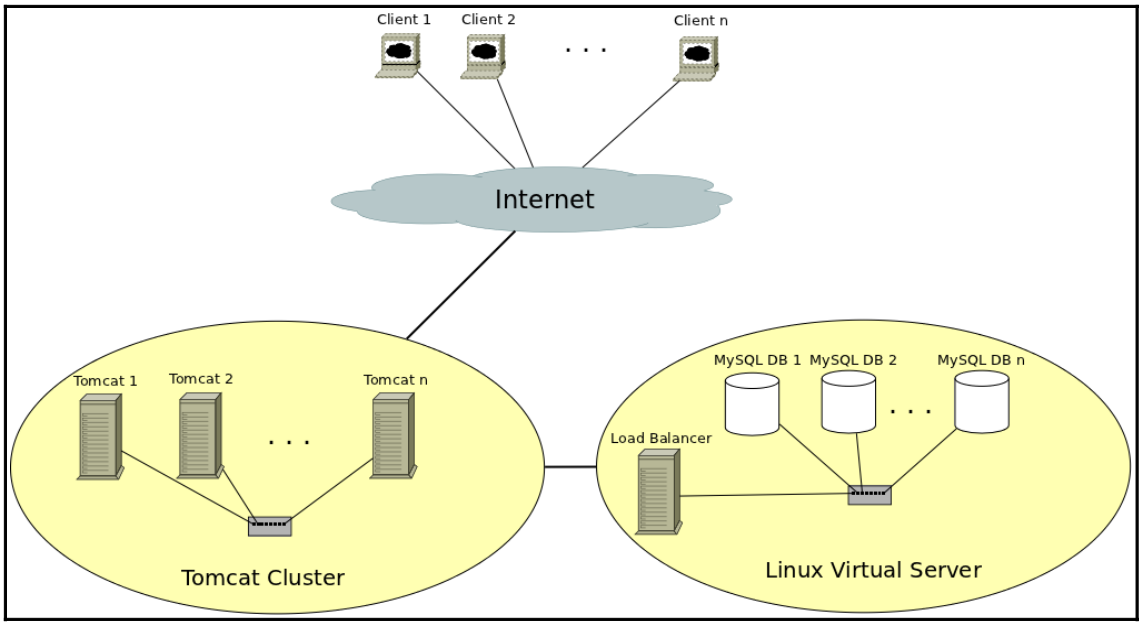

Fig. 4-4: Network diagram of application.

4.5 Implementation Limitations

Since we are developing a prototype and not an application intended for live operation we imposed some limitations on the software. In order to ensure high availability, it would be beneficial to upgrade the physical infrastructure so that the application runs on multiple clusters in different geographical locations. However, to run the prototype we will even downgrade the physical architecture and deploy the prototype to a single Tomcat running on a server which simultaneously runs the MySQL database. This should be sufficient to handle the small amount of users which will test the application. If the application runs well on a single server, running the application on a cluster will only require the necessary hardware and some configuration effort. In order to run the application in a cluster classes need to be serialisable. If a class is serialisable its objects can be written to an output stream and recreated by reading an input stream. This is necessary to transfer the application state between nodes of the cluster. Since the prototype will run only on a single node we did not make the classes serialisable because it was not necessary and it requires only some writing effort.

Additionally, we will not focus on security. Basic security principles, like avoiding SQL injections, will be honoured but it is more useful to implement and demonstrate useful functionality than putting much effort in the security of a system that is not intended for live operation and, thus, collects no sensitive data. Implemented functionality will additionally not be subject to elaborate legal considerations, instead, the focus is on provision of useful functionality. Besides avoiding obvious legal infringements, spending effort on legal considerations is more useful if it is planned that the application will actually go live. As already mentioned, the number of implemented requirements is limited as well. Accordingly, we will not provide a legal info[71] because the application is not intended for productive use and, thus, will only be accessed by a few people for testing purposes. We will realise only the most important requirements to stay within the scope and time restrictions of the thesis. For the same reasons we decided to forgo repetitive gathering of user feedback, which would have been useful to ensure that the application stays in congruence with user expectations. Another aspect that will not be subject to much consideration is the administration of the application. Since we intend to develop a proof of concept prototype and not an application for live operation, features like performance monitoring, inserting new information into the database, or

[71] With legal info we refer to the German term 'Impressum' which has to be provided on every web side by law and provides information on the operating authority.

updating the database are of less importance than features that improve the user experience. However, through the modular architecture such features could be easily added later on.

4.6 Conceptual Design versus Gathered Requirements

The gathered requirements have only little influence on the conceptual design. Elicited user expectations regarding availability, performance, and responsiveness of the application were taken into account. Accordingly, we ensured that the physical architecture is scalable and leads to a high availability. However, other requirements like checking a set of pharmaceuticals for adverse drug reactions have, besides obvious implications like necessity of a database, no direct influence on the architecture. This is probably caused by the circumstance that users have only contact with the user interface of web applications so that everything else does not matter as long as sufficient results are provided. Hence, we designed the application not based on the requirements, which could be accomplished with various architectures, instead, we focused on expandability and fulfilment of performance and availability needs. User interface design might be considered part of the conceptual design as well and is obviously influenced by the requirements. However, we decided that the development of the user interface is more related to implementation than conceptual design so that we will address this topic in chapter '5 Implementation'.

5 Implementation

Chapter '4.2 Application Modules' describes the modular architecture of the application. The implementation was conducted in corresponding steps. At first, the GUI was implemented and then the 'Data Access' module. The 'Application Logic' module has a supporting role and its functionality is implemented as needed by the other modules. This chapter is structured as follows. First, the provision of the main functionality – provision of information in patient information leaflets – is described, by illustrating the initial implementation of the modules 'Graphical User Interface' and 'Data Access'. Afterwards, the implementation of additional functionality which offers further services to users is described. Implementation of additional functionality entails extending the initial implementations of the 'Graphical User Interface' and 'Data Access' modules.

5.1 Graphical User Interface

Implementation of the GUI was aided by the GUI prototype, which we developed for requirements elicitation. Large parts of code from the GUI prototype could be reused and the analysis of the interviews led to a good understanding of user expectations.

5.1.1 Reorganisation of Graphical User Interface Data Management

In the GUI prototype every GUI component had a special purpose that was not served by an alternative GUI component. Therefore, every component could hold its own data and other components would be notified about changes of that data if they were affected by them. However, in order to fulfil a requirement like, for example, hiding functionality which might confuse inexperienced users (requirement 01-03-01-02) such data management is not useful. Inexperienced users might prefer to be asked for their preferences by the application when it is necessary, while other users might prefer a component for changing settings that is always displayed. This requires two different components which manage the same data. An example for such data is the selection of visible GUI components. Redundant management of the data is superfluous and bothersome. Keeping the data consistent consumes precious computing power and

becomes more complex with a rising number of components that work with the same data. Hence, we adopted the 'Model View Controller' (MVC[72]) pattern to manage the data required by the GUI components.

MVC basically consists of three entities for data management, which are called model, view, and controller. The model is used to store data in a single location. Views visualise the data stored in the model in similar or completely different ways for the users. User interaction with a view is captured and expressed in the model by the controller, conversely, the controller translates and expresses changes of the model in the views. Accordingly, we split the classes implementing GUI components into two classes. One serves as a model that holds the current configuration for the GUI component and the other serves as a view and visualises the data for users. Obviously, multiple views can use the same model. Duties and responsibilities of the controller are mostly handled by Vaadin.

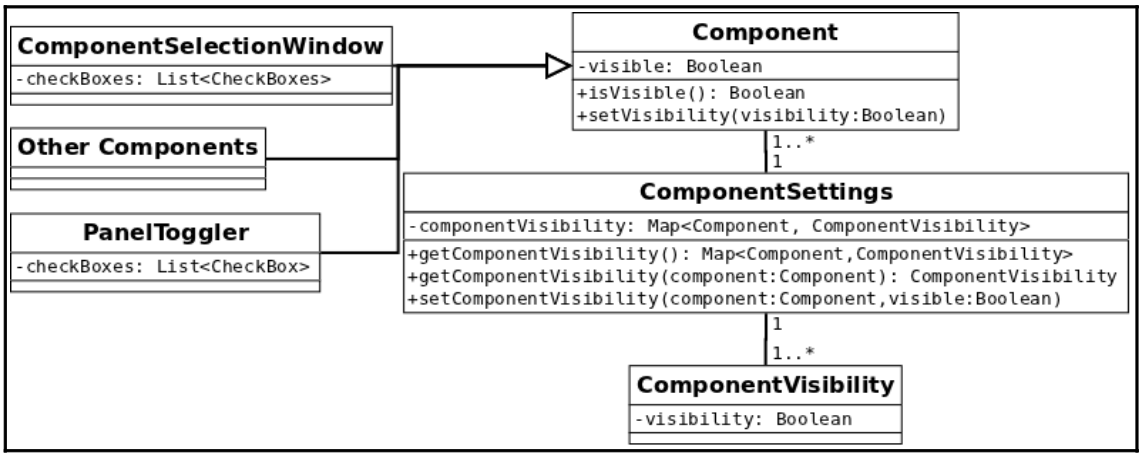

Fig. 5-1: Class diagram: MVC for component visibility.

As an example for our MVC implementation the management of the visibility of the individual GUI components is illustrated in figure 5-1. The classes ComponentSettings and ComponentVisibility represent the model. In ComponentSettings every GUI component whose visibility can be configured by the user is associated with a ComponentVisibility object which stores the visibility of the component. PanelToggler is a view for the model and offers check boxes that users can utilise to toggle the visibility of a selection of GUI components. In the top right corner of figure 2-2, which is a screenshot of the GUI prototype, the PanelToggler is depicted. Now the PanelToggler is positioned similarly, as depicted in figure 5-2, but it offers an additional

[72] See 'http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html' for more information on the Model View Controller pattern.

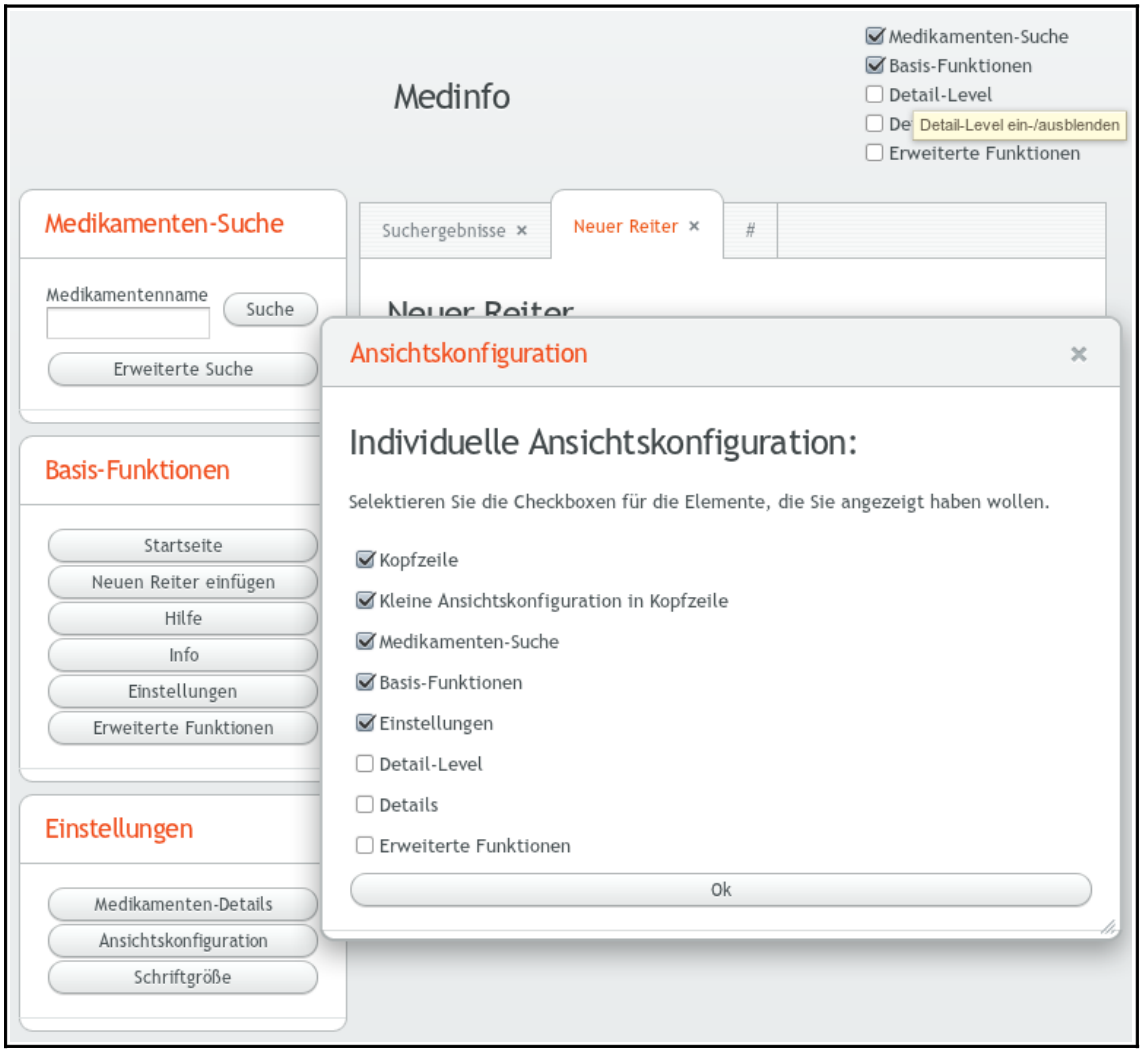

Fig. 5-2: Graphical User Interface: PanelToggler and ComponentSelectionWindow.

check box and the check boxes use ComponentVisibility objects instead of a PanelToggler attributes as data source. Another view is the ComponentSelection-Window, which, in contrast to the PanelToggler, offers check boxes for all GUI components that can be hidden and opens as a pop-up. Figure 5-2 shows the PanelToggler and the ComponentSelectionWindow simultaneously. The screenshot illustrates that only GUI components whose associated check boxes are checked are displayed and that check boxes for the same GUI component in the PanelToggler and in the ComponentSelectionWindow are set to the same value since they are both linked to the same data source. Vaadin manages the state of the check boxes and sets the flag in the ComponentVisibility objects accordingly. Another controller task, which the application needs to handle, is the actual enabling/disabling of the GUI components. This task is handled by the ComponentSettings object as well so that it serves as model and controller simultaneously. The ComponentSettings object already has the references

to the components and toggles the visibility of these based on events that are fired by the ComponentVisibility objects and indicate a changed visibility value. Data necessary for other GUI objects is managed in the same manner.

5.1.2 Realisation of Requirements

Enabling users to change the visibility of GUI components was necessary for the realisation of requirement 01-03-01-02. In order to hide functionality that might confuse inexperienced users, we display only the header and a single tab at application startup that displays introductory information according to requirement 01-03-06-01. From now on we will refer to this tab with the term 'home tab'. Displaying just the two components should not confuse any user and users are provided with basic information about the purpose and usage of the application. Additionally, the home tab provides buttons to configure the displayed GUI components. With the buttons the application can be set into standard view, expert view, or individual view. The standard view targets inexperienced users and displays only the panel 'Medikamenten-Suche' that is used to initiate a search for pharmaceuticals and a panel 'Basis-Funktionen' that offers basic functionality like displaying the home tab, help, or information about the application, and inserting a new tab. Accordingly, the standard view provides only the components that are required for operation of the application. Further GUI components can be turned on when they are needed. Switching to expert view makes all GUI components visible so that users who are familiar with the application can easily access all offered functionality. Clicking on the button for the individual view spawns a ComponentSelectionWindow so that the users can configure which GUI components they want to be displayed. Using less GUI components might also be useful to improve the performance of the application because less JavaScript needs to be rendered if less GUI components are used.

A comparison of the panel 'Medikamenten-Auswahl' in figure 2-2 and the panel 'Medikamenten-Suche' in figure 5-2 demonstrates the realisation of requirement 01-03-01-03. The list-select for the direct selection of pharmaceuticals is removed because such functionality is better provided with a tab or pop-up window, especially for a large amount of pharmaceuticals. Through the removal of the list-select, the search field is now clearly noticeable and the search field does not seem to belong to the list-select

anymore. Additionally, the search field is further emphasised through the search button placed to its right. Moreover, we added a caption to the search field that clearly indicates that the search field is used to find pharmaceuticals by name.

Another rather substantial modification of the GUI implementation was caused by the fulfilment of requirement 01-03-04-01. Instead of switching between a mode where new tab objects replace the current tab and a mode where new tab objects are inserted as a new tab, new tab objects always replace the current tab. If users want to use multiple tabs they can add an additional tab with a button in the panel 'Basis-Funktionen' or by selecting the rightmost tab with the caption '#', which is also shown in figure 5-2. Users can then select the tab they want to use and display any tab object they want in it. This can be search results, help information, information on pharmaceuticals et cetera. This methodology of tab management is more intuitive because it is similar to other applications, like for example Firefox, so that many users are already accustomed to it.

Requirement 01-01-07 was already fulfilled in the GUI prototype. Users can select which information on pharmaceuticals is displayed in the drug information tab with check boxes in the panel 'Details' or by selecting a detail level in the panel 'Detail-Level', which are depicted in figure 2-2 on the right. Additionally, we implemented another view for the selection of displayed information. If users do not want to display the panel 'Details' or 'Detail-Level' they can alternatively open a pop-up window by clicking on the button 'Medikamenten-Details' in the panel 'Einstellungen', which is depicted on the bottom left of figure 5-2. The pop-up for the selection of the displayed sections of information on pharmaceuticals can also be opened directly from the home tab.

According to requirement 01-01-05 we implemented a further modification related to the drug information tab. As illustrated in figure 5-3, we introduced a navigation section at the top of the drug information tab and below each section of information we inserted links to go back to the top of the drug information tab. The links in the navigation section can be used to jump to a specific section of information on pharmaceuticals. Targeted sections are made visible if necessary. Users can go back to the navigation section by clicking the links that titled 'Gehe zum Seitenanfang' and positioned below each section.

To incorporate requirement 01-02-01, users can set the font size with buttons on the home tab or in a pop-up which can be accessed from the panel 'Einstellungen'. In order to display as much information as possible without scrolling, changes of the font size

affect only the text displayed in the tabs that are in the centre of the application and not the text in the navigation components on the left and right side of the application. If users want to resize the navigation components as well, they can do this with the zooming functionality provided by their browser.

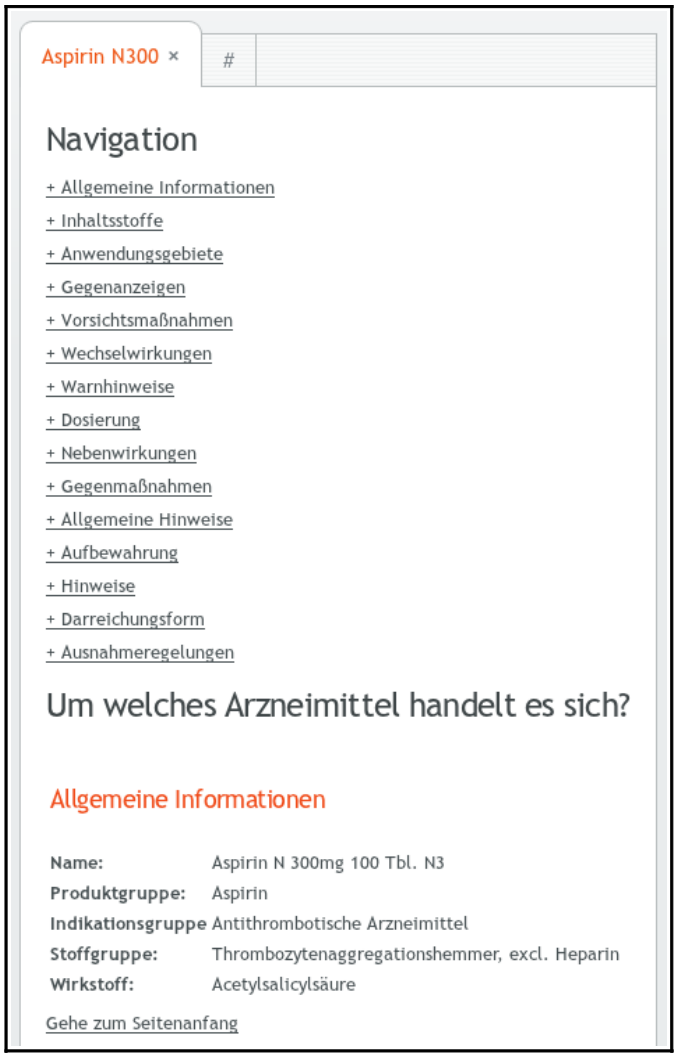

Fig. 5-3: Navigation section of the drug information tab.

With the GUI implementation, we additionally fulfilled the following minor requirements: In compliance with requirement 01-03-03, every GUI component that can be closed indicates this with a close button, icon, or link. To meet requirement 01-02-02-01 we accentuated the sub-headings in the drug information tab by increasing the font size a little and changing the colour from black to orange. This modification can be observed by comparing a sub-heading in figure 2-3 to one in figure 5-3. Another feature that eases locating a section of information is the already mentioned navigation section. A further improvement of the text structuring, as mentioned in requirement 01-02-02, is the change of the vertical alignment in tables from middle to top so that users can clearly identify which columns are in the same row. Additionally, we utilised definition lists instead of unordered lists in the help information in order to clearly separate the headings from the explanatory information. All requirements realised during GUI implementation are summarised in table 5-1.

ID	Requirement	Realisation
01-01-05	Enable the user to go to a specific section of drug information.	Navigation section at top of drug information tab, links to go to top of drug information tab below each section
01-01-07	Do not display more information than the user needs.	Selection of detail level or displayed information in panel 'Details', panel 'Detail-Level' and a pop-up window
01-02-01	Allow for adjustment of font size.	Buttons on home tab, pop-up window
01-02-02	Clear structuring of text.	Vertical alignment in tables set to top, utilisation of definition lists instead of unordered lists
01-02-02-01	Accentuate sub-headings in drug information tab.	Increased font size, colour changed from black to orange
01-03-01-02	Hide functionality that might confuse inexperienced users.	Only header and home tab displayed at startup, selection of preferred view in home tab, setting of component visibility in PanelToggler or a pop-up window
01-03-01-03	Make the search field in panel 'Medikamenten-Auswahl' clearly noticeable.	Removal of list-select for pharmaceuticals, placement of search button to the right of search field
01-03-03	Display close buttons/icons for everything that can be closed.	Display of close buttons, icons or links
01-03-04-01	Intuitive functionality for creation of new tabs.	No automatic tab creation, allowing users to insert new tabs with a button in panel 'Basis-Funktionen' or by selecting the rightmost tab
01-03-06-01	Display introductory information at application startup.	Displaying information on purpose and utilisation of application at application startup

Tab. 5-1: Summary of realised requirements during GUI implementation

5.2 Data Access

The database and database access are a crucial part of the application because the database holds the information that is provided by the web application and, moreover, the display of information in patient information leaflets is the main objective of the application. To be able to provide information in patient information leaflets, the following steps were necessary for the implementation of the 'Data Access' module. At first the necessary data had to be collected and made retrievable in a database. Functionality that enables the application to query the database had to be implemented. Finally, the routines that conduct specific operations like searching a pharmaceutical or retrieving information on a specified pharmaceutical were implemented and integrated in the GUI.

5.2.1 Making the Data Available

When thinking about acquisition of the required data three alternatives come to mind: digitising the information on one's own, data input by third parties or utilisation of existing data. Developing an individual data model and digitising information in patient information leaflets on one's own is probably the most effective way to ensure that the provided data fits the needs of the application and achieves desired quality levels. However, this approach is tedious, wasteful, and ineffective, especially in the scope of this thesis, since much time would be wasted on data input. Data input by third parties seems quite promising; the workload could be distributed between multiple entities, data quality could be ensured by the input interfaces, and an own data model that soundly supports the web application could be used. Yet, third parties might not be motivated to help create a database for a web application in its infancy. Getting support from third parties is more likely once the web application is operational and third parties deem it beneficial. This might be useful to improve the provided information later on. Wikipedia[73] is a good example where such an approach works. Another problem that occurs in our case is that we need to provide reliable information in order to avoid misinforming users on pharmaceuticals which could obviously lead to serious and harmful consequences. Since checking reliability of information with a software is too complex and too difficult, data input by third parties would require too much effort for

[73] See 'http://www.wikipedia.org/' for more information on Wikpedia.

checking the reliability and validity of the entered information. Hence, we decided to use existing data. We chose to use the GELBE LISTE PHARMINDEX[74] because it offers large parts of the required information for many pharmaceuticals and we had access to the database. A downside is that we have little influence on the formatting of the data and need to cope with used abbreviations, technical terms, or concatenations of information as well as the general data model and inconsistencies in the data.

The version of the GELBE LISTE PHARMAINDEX that we had access to and used is a little chaotic. It contains many tables and some tables are even duplicates of each other. Foreign key restrictions are not specified and data inconsistencies provoke crashes of SQL statements. All in all, it was difficult to make sense of the cluttered database. Therefore, we developed a similar data model, created our own MySQL database and copied useful data into the MySQL database. We sifted through the GELBE LISTE PHARMINDEX and replicated useful information in our MySQL database step by step. This led to a data model that resembles the data model of the GELBE LISTE PHARMINDEX since they represent the same information and similarities ease the transfer from one database to another. However, we added primary key restrictions and foreign key relationships so that connections between data are expressed and enforced on the database level. Moreover, we removed inconsistencies and transferred only the information we deemed useful. We designed the data model with MySQL Workbench[75] and the resulting data model is illustrated in figure 5-4. We found the data model to be a great help for structuring the database, identifying available information, and detecting missing information. The original database on the other hand provided only many tables with no apparent relationships that had to be sought out and verified with SQL. MySQL Workbench is distributed with MySQL. Besides offering capabilities to design the data model, it can also handle the forward engineering of the database from the entity relationship model. In combination with a little Java program we wrote, this enables us to rapidly deploy changes of the database. The Java program runs a sequence of routines that get the required information from the GELBE LISTE PHARMINDEX, perform necessary filtering and transformation work, and, finally, insert the information into the MySQL database.

---

[74] See 'http://www.gelbe-liste.de' for further information.

[75] See Oracle (2011), p. 2756 for more information on MySQL Workbench.

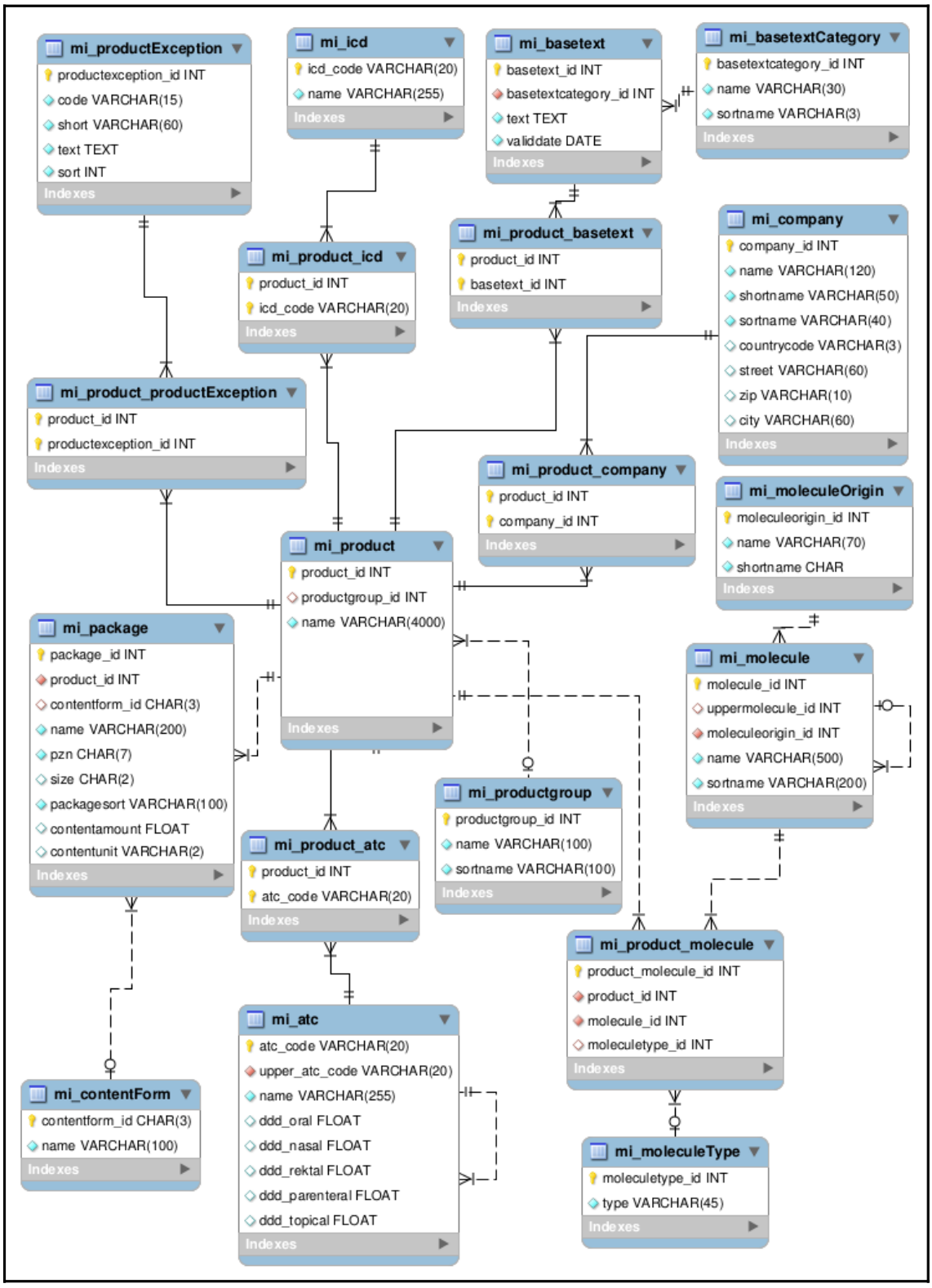

Fig. 5-4: Entity relationship diagram of used MySQL database.

The MySQL database provides information on about 100,000 pharmaceuticals and offers detailed information for a quarter of them. While this is not enough for productive use it is a good start and should suffice for the prototype we are developing. As depicted

in figure 5-4, mi_product is the central table of the data model and contains products. A product is a pharmaceutical with a specific name, dosage form, and strength. 'Mucosolvan Infektionslösungskonzentrat' is obviously a different product than 'Aspirin N 100mg Tbl.', but 'Aspirin N 100mg Tbl.' is, for example, also a product different from 'Aspirin protect 100mg Tbl.' or even 'Aspirin N 300mg Tbl.'. We inherited this definition of a product from the GELBE LISTE PHARMINDEX. Other interpretations of a product might be possible and more suitable for our application, but the previously described definition of a product should suffice for our application. Changing the interpretation of a product would probably cause more work than benefits since, for example, most of the relationships and associations in the database depend on it. A product is sold in different packages, which are stored in the table mi_package, and have a specified size like, for example, 20 or 50 pills. Additionally, mi_package provides further information like the dosage form, PZN, or the name associated with a package.

Products that differ only in strength from each other comprise a product group which are represented in mi_productgroup. 'Aspirin N 100mg Tbl.' and 'Asprin N 300mg Tbl.' are, for example, in the same product group. The remaining tables represent either relationships between tables or hold patient information leaflet information. Ingredients of a product are represented in the tables mi_molecule, mi_product_molecule, mi_moleculeType, and mi_moleculeOrigin. The tables mi_product_atc and mi_atc link pharmaceuticals to ATC codes. Information on companies associated with pharmaceuticals are held in mi_product_company and mi_company. Relationships between pharmaceuticals and ICD codes are expressed in mi_product_icd and mi_icd. Situations in which health insurance companies subsidise taking of pharmaceuticals that do not require prescription are stored in mi_product_productException and mi_productException. Most patient information leaflet information is captured in the table mi_basetext that holds strings that provide information on pharmaceuticals, which is ordered in different categories, like side effects, application, dosage, or contraindications.

The brief description of the data model shows that we have information regarding most information that patient information leaflets, as described in chapter '2.1.1 Information in Patient Information Leaflets', need to provide. Only for a few sections of information, which are listed in the following, no information is provided by the database. In many cases, information regarding dosage is not differentiated by age groups, like for example different taking regulations for children and adults. No information regarding

countermeasures for problems caused by wrong dosage or other circumstances is provided. How a pharmaceutical should be taken is not described distinctly, but such information is sometimes bundled with the dosage information. Moreover, we have no information regarding the issue date of the patient information leaflet and the use-by date of the pharmaceutical, but these dates are not necessary because the web application is not a patient information leaflet distributed with the pharmaceutical. Yet, entries in mi_basetext are associated with an attribute valid date that gives a point of time where the information was valid. We display the valid date in the drug information tab so that users have more input to decide on the trustworthiness of the application.

A further deficit of the data is that most of the information is captured in strings which consist of whole sentences. All side effects of an pharmaceutical are, for example, stored in one string. Dosage information, contraindications, and all other categories of information in mi_basetext are also captured in one string per category per product. This causes, as demonstrated in the following example, unnecessary effort during the search for pharmaceuticals. In order to search for pharmaceuticals by field of application it would be useful to have associations between the pharmaceuticals and only the relevant information like 'cough' or 'headache'. Instead, since the information is stored in strings, we need to check many additional filler words, like 'field', 'application', 'of', 'are', 'and' et cetera, which hold no relevant information. Additionally, in a reasonable way, only the whole string can be presented to users. In order to filter side effects by frequency of occurrence, it would be more convenient if the side effects were stored in multiple records with an associated frequency of occurrence. Since the strings do not have sufficient syntactical commonalities they cannot be automatically converted in a reliable way and have to be used the way they are. Some of the strings use, in contrast to others, a lot of abbreviations that due to ambiguities cannot be computationally replaced. Furthermore, the formatting of the strings with HTML ranges from non-existent to very sophisticated. While this leads to good presentations of the information for some pharmaceuticals, the presentational differences are undesirable. It would have been better to be able to present the information consistently and to build the representation for the user from isolated records. However, manual revision of the strings could not be squeezed into the time frame of this thesis and revising the strings with a program would probably cause more problems than it solves due to ambiguities and insufficient syntactical commonalities. Removal of the markup tags could easily be performed with

a simple program that matches them and replaces them with the empty string. Nevertheless, the strings that were composed based on the markup tags are better readable with them. Hence, we will have to work with the strings as they are.

On the other hand, benefits of the chosen approach are that, by creating our own database, we could chose a data model that fits the needs of our application and by getting the information from GELBE LISTE PHARMINDEX we have access to data which is already in productive use and, thus, is likely to offer satisfactory quality. Moreover, we have access to large parts of the relevant information. Another benefit related to using our own database is that, if necessary, we can easily add further information from other databases by extending our data model and adapting the Java program that collects the required information and populates the MySQL database.

In conclusion, we have built a database that is sufficient for the purposes of the prototype to be developed. The database can be easily extended if this is necessary for future improvements and the quality of the data should be adequate to demonstrate the merits of the web application.

5.2.2 Database Mapping

Besides having the information in the database the web application needs to be able to access the database so that it can present the information to users. One way to accomplish this is to use the standard Java application programming interface[76] (API) for accessing and working with data in relational databases. We chose this approach for the simple Java program we used to populate the MySQL database. We established a connection with the original database and the MySQL database and wrote multiple short SQL statements that either get data from the original database or insert data into the MySQL database. These statements were quite short and involved only a few tables at once. However, working with the database and offering useful information to the user entails querying data from almost all available tables and selecting pharmaceuticals based on characteristics stored in different tables. This would require more complex SQL statements. In order to ease the retrieval of the necessary information we used Hibernate, which we already introduced in chapter '4.3.3 Hibernate'.

---

[76] An API to access relational database is provided in the package java.sql. See the Javadocs (http://docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html) for further information.

With Hibernate database connections do not need to be managed manually and can be administered more effectively by a connection pool which reduces the workload on the database. Hibernate needs just an XML file that specifies the connection parameters and then sessions that can be used to retrieve data and require no further configuration can be obtained whenever database access is necessary. When using Hibernate, it is no longer necessary to directly access the database. Instead, Java classes whose objects represent the necessary data[77] need to be implemented. Additionally, for each such class an XML file needs to be written that specifies in which tables Hibernate can find the associated data and how the database is organised. Accordingly, we implemented classes for all needed tables and wrote corresponding XML configuration files. Afterwards, we could obtain data from the database without SQL, instead, the data was retrieved using HQL. With HQL we could directly query objects and read their attributes to present information to the user without worrying about data type conversion, database connections, or join conditions. The XML files specify what information is stored in which tables as well as how to join the tables. Unless told otherwise, Hibernate gets only the information that is actually used and joins tables automatically if necessary. This behaviour eases writing of well performing code because it is only required to specify which objects should be selected under which conditions and the actual information, which can be accessed with the attributes of the objects, is loaded as needed later on.

5.2.3 Provided Functionality

The Hibernate mappings in form of the classes that represent the data and the XML configuration files make up a large part of the lines of code in the 'Data Access' module, but the interesting facets of the the module are the classes that actually work on the data.

A search for pharmaceuticals is initiated in the GUI module. Users can choose between two different ways of searching. They can either use a simple search to find pharmaceuticals by name or use the advanced search that allows for various search characteristics. The simple search is initiated in the panel 'Medikamenten-Suche', which is depicted in figure 5-2. Users need to enter the name of the pharmaceutical they are looking for into the text field and start the search by clicking on the search button.

---

[77] The classes need to specify the attributes which should hold the desired information. Relationships with other tables that are not unique are represented with collections that hold objects of the class representing the other table. Additionally, the classes need a trivial constructor, another constructor to set all the attributes and getter and setter for all attributes.

When users click on the button the entered name is checked for validity. If users specify no string or less than three letters an error message, which points out that the user has to specify a search parameter with a length of at least 3, is raised. Otherwise, an HQL statement selects all pharmaceuticals where the name specified in mi_product fits the specified search string. The resulting pharmaceuticals are then listed in a tab so that users can select them to display the patient information leaflet information. In order to find a pharmaceutical name in larger strings, the SQL wildcard '%' is added as prefix and suffix to the string specified by the user.

The advanced search is initiated by clicking on the button 'Erweiterte Suche' in the panel 'Medikamenten-Suche'. When users click on it an advanced search tab, which is displayed in figure 5-5, is opened. In the advanced search tab users can specify six different search characteristics. Pharmaceuticals can be searched by name, PZN, field of application, active pharmaceutical agent, ATC code, or ICD code. While searching for pharmaceuticals by ATC or ICD code is probably only useful for users with a background in healthcare, the other four characteristics are useful for common users. If a user is looking for a pharmaceutical similar to another pharmaceutical he could find one with a search by active pharmaceutical agent. Searching for pharmaceuticals by field of application is useful when looking for a pharmaceutical that helps with a concrete ailment. Users could, for example, search for the terms 'cough' or 'headache'. Looking up information on a specific pharmaceuticals is supported by the search characteristics name and PZN. Users can either enter the name of the pharmaceutical or copy the seven digit PZN from the packaging which might be shorter. Users start the actual search by clicking on the button 'Suche'. The specified parameters are then checked for validity. Methods for checking input strings are implemented in the 'Application Logic' module so that they can be easily utilised by different classes. Strings specified for name, field

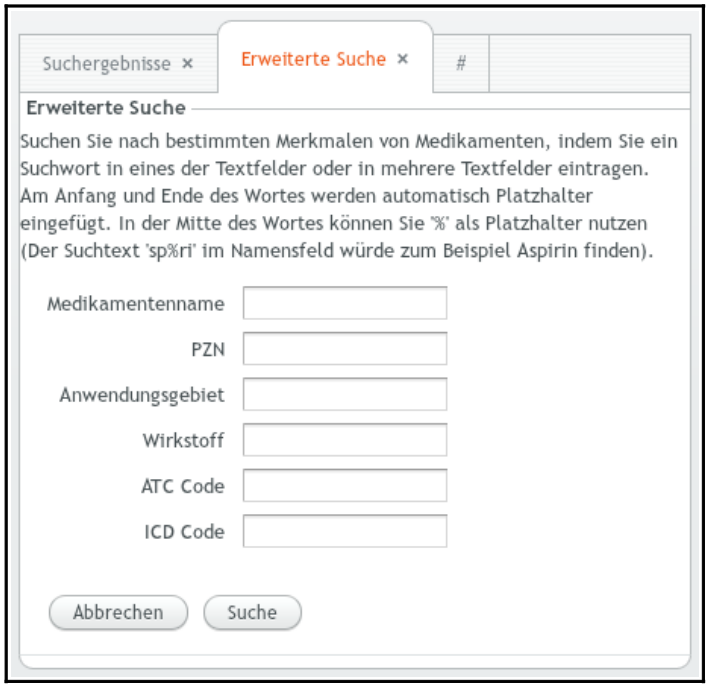

Fig. 5-5: Advanced search tab.

of application, and active pharmaceutical agent need to have a length of at least 3. The PZN needs to be an integer with exactly seven digits. Input for an ICD code needs to have a length between 3 and 6 and a string specified as ATC code should not have more than 7 characters. If users enter no parameters, an error message, which informs them that they need to specify input parameters, will be raised. If no search terms are specified correctly, an error message will point out how the falsely entered terms need to be provided. In a situation where some parameters are specified correct and others wrong, pharmaceuticals will be searched with the correct parameters and a warning message that informs users about the wrong input will be displayed.

As already insinuated, the advanced search allows for combinations of search criteria. In order to perform the search, an HQL query is constructed according to the correctly specified parameters. The query joins the corresponding objects and filters the results by the entered criteria. The name parameter has to match the name specified in mi_product. The ATC code parameter has to match associated ATC codes in mi_atc. ICD codes stored in mi_icd have to match the ICD code parameter. The active pharmaceutical agent parameter has to match an associated molecule in mi_molecule and the relationship has to be classified as active pharmaceutical agent relationship in mi_moleculeType. Other possible classifications are, for example, colourant or preservative agent. The number specified in the PZN field needs to fit a PZN in mi_package. User input for the field of application is compared to diseases corresponding to ICD codes and strings with the category 'field of application' in mi_basetext. Fitting pharmaceuticals are then listed in a search results tab, where users can access patient information leaflet information by clicking on them. Except for the PZN parameter, which should match exactly, all parameters are wrapped in wildcards so that occurrences in larger strings are matched as well. Furthermore, users can use wildcards for more complex parameter specifications[78].

All performed matches are case-insensitive so that results are not omitted because of different case sensitivity. Furthermore, the application remembers the last user input in a search field so that users can easily modify a search without having to retype previously entered criteria. In order to prevent SQL injections we used named parameters so that user input is escaped and cannot be interpreted as HQL commands. To avoid waiting for user input while having an open database connection that requires resources, we open a session at the begin of a search and close it as soon as we have the results. Sessions are

[78] 'Aspirin%100%mg' would for example find all variants of Aspirin with a specified strength of 100 mg.

objects provided by Hibernate. Through the Hibernate configuration Hibernate knows how to establish a connection, but sessions need to be created in order to indicate that a database connection is needed by the application. The resulting objects representing the pharmaceuticals are then passed as a list to the search results tab to be presented to the user. If a user selects a pharmaceutical in the search result tab, the corresponding object is passed to a drug information tab to be displayed. Since it would be wasteful to load information for results which will not be displayed from the database, only the name, which is displayed in the search results tab, and the product_id is definitely initialised in objects that are passed to a drug information tab. Depending on the previously conducted search other attributes might be initialised as well, but necessary attributes concerning, for example, the dosage information or contraindications are not initialised. Moreover, we already closed the session associated with the object. In order to obtain the required information we added a further class to the 'Data Access' module that provides functionality to retrieve all information to be displayed in the drug information tab based on the product_id of a pharmaceutical. This approach is quite effective because we need no sessions to remain open unnecessarily and load only the data that we really need from the database.

5.2.4 Realisation of Requirements

With the various search criteria that can be used for the advanced search we fulfilled requirements 02-02-01-01, 02-02-01-02, and 02-02-01-04 that demanded searching for pharmaceuticals by field of application, name, and active pharmaceutical agent, respectively. Moreover, the application can now serve its primary objective and provides information in patient information leaflets to users. We did not collect many requirements for the 'Data Access' because users have no direct contact with the database. However, the 'Data Access' module is an important module because the functionality it provides is necessary for satisfactory operation of the application.

ID	Requirement	Realisation
02-02-01-01	Search for pharmaceuticals by field of application.	Field of application parameter for advanced search
02-02-01-02	Search for pharmaceuticals by name.	Name parameter for simple and advanced search
02-02-01-04	Search for pharmaceuticals by active pharmaceutical agent.	Active pharmaceutical agent parameter for advanced search

Tab. 5-2: Summary of requirements realised by module 'Data Access'.

On the other hand, implementation of the 'Data Access' module showed that realisation of some requirements is inhibited by properties of the data. Since the strings in mi_basetext cannot be modified/broken down in a satisfactory manner, we have to use them as they are. This entails displaying them with included HTML markup tags and not being able to display less structured strings in a more readable way like, for example, splitting them up in a list. This is problematic for achieving requirement 01-02-02 because we simply cannot structure some of the data ourselves. However, this affects only some data and some strings are even well structured. The consequences for requirement 01-01-07-02 and 01-05-02 are much worse because realisation of these requirements had to be abandoned. The information in the strings holding information regarding side effects cannot be filtered by frequency occurrence because the strings have no sufficiently common structure. Accordingly, we cannot provide functionality to filter displayed side effects by frequency of occurrence. Moreover, abbreviations cannot be removed reliably. The used abbreviations are ambiguous and inserting wrong replacements would probably worsen readability and comprehensibility more than sticking with the abbreviations. These requirements have to be realised by improving the database later on. In the scope of this thesis, requirements 01-01-07-02 and 01-05-02 will not be implemented.

ID	Requirement	Reason
01-01-07-02	Allow users to filter side effects by frequency of occurrence.	Not enough syntactical commonalities in strings, no relationship between side effects and frequency of occurrence expressed on database level
01-05-02	Avoid abbreviations in text.	Abbreviations too ambiguous, consequences of wrong substitutions probably worse than keeping the abbreviations.

Tab. 5-3: Requirements abandoned during development of module 'Data Access'.

5.2.5 Concluding Remarks

The implementation of the 'Data Access' module described in chapter '5.2 Data Access' serves as answer to the second research question: How can the information required for the web application be obtained and displayed? The necessary information can be obtained in multiple ways, like digitisation on one's own, data input by third parties, or utilisation of existing data, which each pose their own challenges. For the prototype that is to be developed in this thesis, we chose to leverage existing data. Independent of the

method chosen to obtain the data, databases are a useful tool to make the data available to the application. We inserted the necessary information into a database and implemented a module which provides functionality to access the data so that it can be displayed or processed by other modules.

5.3 Additional Functionality

After the initial implementation of the modules 'Graphical User Interface' and 'Data Access' the web application enables users to access patient information leaflet information. However, to create further benefits for users, the web application can offer further services beyond the mere representation of information in patient information leaflets. The term 'additional functionality' refers to such services. In the course of this chapter we will illustrate the implementation of additional functionality provided by the web application.

5.3.1 Pop-up for Selection of Pharmaceuticals

In the GUI prototype the panel 'Medikamenten-Auswahl' included a list that contained all available pharmaceuticals. During the initial implementation of the GUI module we removed that list since it was not suitable to display many entries because the list was too small and offered no filters. In order to realise requirement 02-02-03-01 we implemented a similar list. The new list is displayed in a pop-up window because it needs to be displayed only for the selection of a pharmaceutical and can be closed afterwards. In order to make the list accessible we added a new button to the panel 'Medikamenten-Suche'. The pop-up gives a short introduction how the user can list pharmaceuticals and provides two text fields to filter the pharmaceuticals, which are displayed in a table, by name. To filter the displayed pharmaceuticals users need to enter a string in one text field or both text fields and click on a button to display matching pharmaceuticals. The first text field is used to specify a prefix and the second text field is used to filter by a string that may be contained anywhere in the name. The list of pharmaceuticals is filtered with the simple search functionality we described in chapter '5.2.3 Provided Functionality'. When the user selects a pharmaceutical in the pop-up the pop-up is closed and a drug information tab is opened for the selected pharmaceutical.

Besides searching for pharmaceuticals by name prefix the pop-up provides not more functionality than a simple search by name. Yet, the pop-up is a useful feature because users see the results in the same window and can easily adopt their search parameters to find the pharmaceuticals they are looking for. Furthermore, in contrast to the simple search, the pop-up does not limit the length of the strings entered for filtering. Therefore, the application meets another requirement. If users do not know the exact name of a pharmaceutical they can either conduct a search with a partial name or use the pop-up for pharmaceutical selection, which displays the results in the same window, can search for prefixes and has no input restrictions.

5.3.2 Detection of Adverse Drug Reactions

Taking multiple pharmaceuticals in combination might lead to some undesired effects. One pharmaceutical might, for example, lessen the effectiveness of another pharmaceutical. These effects are called adverse drug reactions. Checking a set of pharmaceuticals for adverse drug reactions can, for example, be useful for users if they want to take a pharmaceutical that requires no prescription like Aspirin and would like to know whether the pharmaceutical can be taken in combination with other pharmaceuticals they are taking on a regular basis. Reading all relevant package information leaflets would be a time-consuming task. The web application, on the other hand, is well suited for this task because it can access and process relevant information much faster as long as the information is stored in the database.

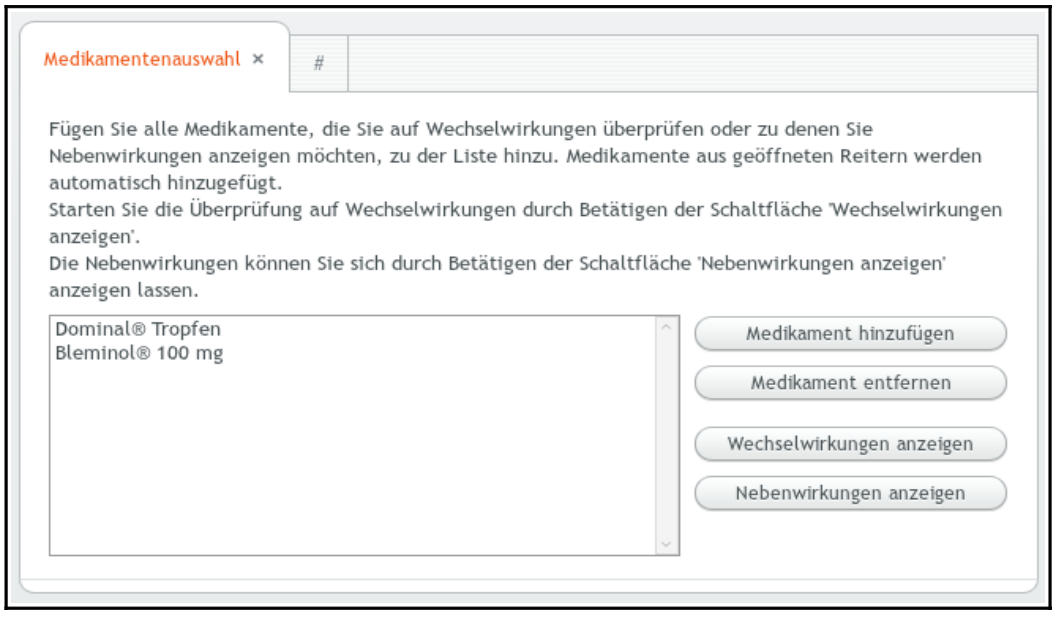

Fig. 5-7: Tab for composing a set of pharmaceuticals

At first we implemented a new tab, which is depicted in figure 5-7, that enables users to compose the set of pharmaceuticals they want to check for adverse drug reactions. In the following we will refer to this tab with the term 'set composition tab'. When users open the set composition tab and concurrently display information on pharmaceuticals in other tabs, these pharmaceuticals are automatically added to the set. In order to add additional pharmaceuticals to the set users can use the button 'Medikament hinzufügen', which opens the pop-up described in chapter '5.3.1 Pop-up for Selection of Pharmaceuticals'. This pop-up can be configured to perform different tasks when a pharmaceutical is selected. In order to use the pop-up to add pharmaceuticals to the set, we just needed to pass it another object for selection handling that adds selected pharmaceuticals to the set instead of opening a new drug information tab. Users can remove pharmaceuticals from the set with the button 'Medikament entfernen'. After composition of the pharmaceutical set the check for adverse drug reactions is started with the button 'Wechselwirkungen anzeigen'.

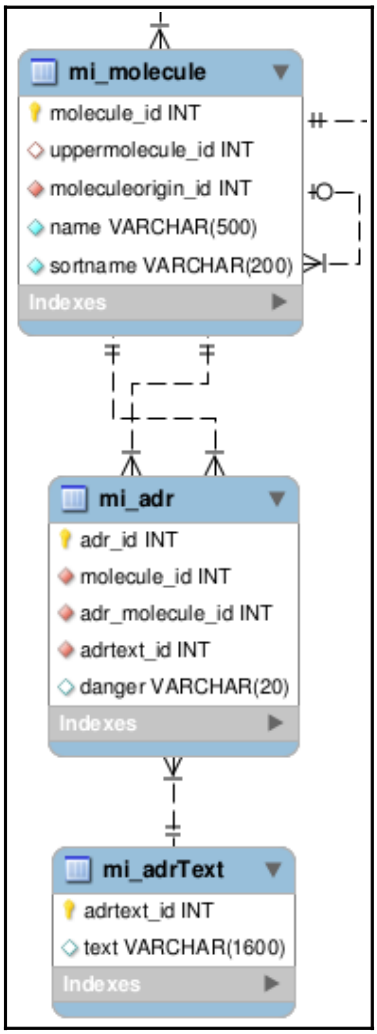

Fig. 5-6: Tables storing data on adverse drug reactions

In order to detect adverse drug reactions we added two further tables mi_adr and mi_adrText, which are depicted in figure 5-6, to the data model. Similar to the other tables, the data hold in these tables is compiled from tables in the GELBE LISTE PHARMINDEX. Adverse drug reactions are stored based on ingredients of pharmaceuticals. Each recorded adverse drug reaction specifies a pair of molecules[79] that cause the adverse drug reaction. Additionally, a rating for the graveness of an adverse drug reaction and a text describing the adverse drug reaction are provided by the tables. The description is stored in mi_adrText.

Since the tables specify adverse drug reactions based on ingredients, we need to check for adverse drug reactions based on the ingredients of the pharmaceuticals that comprise the set prepared in the set composition tab. We implemented a suitable algorithm in the

[79] molecule_id specifies one ingredient and adr_molecule_id specifies the other.

module 'Application Logic'. We decided to perform necessary comparisons of molecules (ingredients) on the application server and not in the database because the comparisons concern only a subset of all molecules. This avoids overhead for finding a molecule in the set of all molecules and reduces network traffic as well as database accesses. Hence, this should be the faster alternative.

The algorithm works as follows: At first it is necessary to retrieve the required data from the database. For every selected pharmaceutical in the set composition tab a set with all associated molecules that are an ingredient of the pharmaceutical is constructed. Moreover, additional sets are constructed for every molecule of a chosen pharmaceutical. In the following we will refer to these molecules as 'primary molecules'. The sets contain the adverse drug reaction (ADR) molecules associated with the primary molecules. With the term 'ADR molecules' we refer to a molecule that has an adverse drug reaction with a primary molecule. After the initialisation we iterate over all pharmaceuticals chosen in the set composition tab and check whether any set of ADR molecules of its ingredients intersects with the ingredients of another pharmaceutical that was also selected in the set composition tab. Intersections other than the empty set constitute detected adverse drug reactions because an ingredient of a pharmaceutical that has an adverse drug reaction with an ingredient of another pharmaceutical has been found. The algorithm is correct because we compare the ingredients of all chosen pharmaceuticals with the ingredients of all other chosen pharmaceuticals. The algorithm is finite because it stops after all possible checks. To tweak the performance of the algorithm, we leveraged the condition that an adverse drug reaction with molecule_id A and adr_molecule_id B is also stored as adverse drug reaction with molecule_id B and adr_molecule_id A. Accordingly, it is sufficient to check only for adverse drug reactions between pharmaceutical X and pharmaceutical Y and it is not necessary to additionally check for adverse drug reactions between pharmaceutical Y and pharmaceutical X. This cuts the number of conducted comparisons in half. Runtime of the algorithm rises fast with the number of checked pharmaceuticals. Nevertheless, since it is unlikely that users check dozens of pharmaceuticals for adverse drug reactions, performance of the algorithm should be sufficient.

Whenever an adverse drug reaction is determined the involved pharmaceuticals and the according adverse drug reaction record (as a Hibernate mapping object) are stored in a list. After the algorithm has finished a table containing the detected adverse drug

reactions is displayed. The table shows the gravity rating and the involved pharmaceuticals for every detected adverse drug reaction. Users can access further information regarding an adverse drug reaction with a pop-up window that is displayed when they click on a row of the table. If no adverse drug reactions were found users are notified and the table shows no entries. The pharmaceutical set compiled in the set composition tab is stored for the duration of a session so that users can easily add further pharmaceuticals without having to re-enter pharmaceuticals.

This functionality shows that the web application is suitable to offer more complex services that aggregate information on multiple pharmaceuticals. As long as the required data is available or can be made available, the web application can provide useful additional information that cannot as easily be provided by patient information leaflets.

5.3.3 Listing of Side Effects

Figure 5-7 shows a further, previously unmentioned button 'Nebenwirkungen anzeigen' in the set composition tab. This button is used to display all side effects of a set of pharmaceuticals in a single tab. This might, for example, be useful if users experience headaches and want to check if this can be a side effect of one pharmaceutical they are currently taking.

We decided to place the buttons for the adverse drug reaction check and the listing of side effects in the same tab and run both routines on the same set because users probably want to run an adverse drug reaction check and display side effects for the same set of pharmaceuticals. Such a set could, for example, be comprised of the pharmaceuticals they are currently taking.

5.3.4 Display of Alternative Dosage Forms

Displaying alternative dosage forms seems to be easy: It is just necessary to iterate over all pharmaceuticals in the same product group and select the distinct dosage forms. However, this simple approach will not work. As described in chapter '5.2.1 Making the Data Available', in our case pharmaceuticals are identified as products that have a specific name, dosage form and strength. Additionally, a product group contains products with same name and dosage form so that the products in a product group have the same dosage form.

Therefore, we devised a different approach. Products are associated with ATC codes and ATC codes basically specify the purpose of a pharmaceutical. Correspondingly, pharmaceuticals that have the same ATC code have the same purpose and are possible alternatives for each other. Hence, we determined the ATC codes of the primary product and selected all other products with a fitting ATC code. The term 'primary product' refers to the product for which alternative dosage forms should be determined. Subsequently, we used a simple bucket sort algorithm to order the products with fitting ATC codes by dosage form.

A problem of this approach is that products can be associated with multiple ATC codes if they have multiple indications. If this is the case our approach will also include dosage forms of pharmaceuticals that have only indications the user is not interested in. Moreover, determination of the alternative dosage forms might take some time, if many other products with matching ATC codes exist. However, this approach is useful because users can determine alternative dosage forms. If they want to switch to a pharmaceutical with another dosage form they just need to check if it is really suitable for the desired application.

In order to avoid longer load times for drug information tabs, users can start the determination of alternative dosage forms by clicking on a link in the dosage form section of the drug information tab. Thus, alternative dosage forms are identified only when necessary. Detected alternative dosage forms are displayed in a pop-up where users can display the similar products that have a certain dosage form by clicking on a dosage form in the pop-up. If users select one of the similar products a drug information tab for this product is displayed.

5.3.5 Display of Pharmaceuticals with same Active Pharmaceutical Agents

A further additional functionality accessible from the drug information tab is a feature that displays pharmaceuticals with the same active pharmaceutical agents as the primary pharmaceutical. The term 'primary pharmaceutical' refers to the pharmaceutical displayed in the drug information tab.

In order to find suitable pharmaceuticals, we collected the active pharmaceutical agents of the primary pharmaceutical. Active pharmaceutical agents can be determined by checking all associations between mi_product and mi_molecule of the respective pharmaceutical for a moleculetype_id in mi_product_molecule that specifies the

molecule as active pharmaceutical agent of the corresponding pharmaceutical[80]. Subsequently we selected all other pharmaceuticals with a set of active pharmaceutical agents that is a superset of the set of active pharmaceutical agents of the primary product. These products are the products we need to display because they have the same active pharmaceutical agents as the primary product.

For primary products with only one active pharmaceutical agent, matching products could easily be determined with an HQL/SQL query using the operator 'in' that checks whether an object is an element of a set. With this operator we can check whether the active pharmaceutical agent of the primary pharmaceutical is in the set of active pharmaceutical agents of the currently examined pharmaceutical. However, the operator 'in' cannot be used to determine whether a set with a cardinality greater than 1 is a subset of another set. As a solution, we dynamically added further conditions for all other individual active pharmaceutical agents of the primary pharmaceutical that ensure that each active pharmaceutical agent of the primary product is an active pharmaceutical agent of the currently examined pharmaceutical.

The functionality can be accessed with a link in the ingredients section of the drug information tab and matching pharmaceuticals are displayed in a pop-up window. This way loading time of drug information tabs is not increased through the determination of pharmaceuticals with the same active pharmaceutical agents and these pharmaceuticals are only identified when it is necessary. If users select a pharmaceutical in the pop-up, a drug information tab for this product is displayed.

5.3.6 Comparison of Pharmaceuticals

Users might want to compare two pharmaceuticals to choose the more fitting alternative of two pharmaceuticals or might want to compare a pharmaceutical they have been taking in the past to an alternative pharmaceutical they recently got prescribed. This could be achieved by opening information on both pharmaceuticals in two different tabs and comparing the pharmaceuticals by switching back and forth between the tabs while reading the information provided on the pharmaceuticals. Viewing information on pharmaceuticals in multiple tabs is already supported by the application. However, it would be more convenient and user-friendly, if related information on both

---

[80] The entities and relationships are illustrated in the entity relationship diagrams depicted in figure 5-4 and 7-2.

pharmaceuticals were accessible simultaneously and displayed coherently so that users would not have to switch between tabs and would not have to identify/locate related information themselves. Hence, we implemented a further version of the drug information tab that offers such functionality.

At first we implemented a tab that enables users to select two pharmaceuticals they want to compare. This tab is accessible from the panel 'Erweiterte Funktionen', which is depicted in figure 5-8. After selection of the pharmaceuticals, users can open the tab for pharmaceutical comparison by clicking on a button. From now on we will refer to the tab for pharmaceutical comparison with the term 'drug compare tab'. The drug compare tab is derived from the drug information tab so that useful functionality already implemented in the class for the drug information tab can be used. This entails functionality that handles the visibility of sections providing information on pharmaceuticals or that composes the labels displaying the information on pharmaceuticals. Accordingly, only the layout management of the drug information tab and the handling of the buttons that display alternative dosage forms or display pharmaceuticals with the same active pharmaceutical agent need to be implemented in the class for the drug compare tab.

Adoption of the methods handling user events of the previously mentioned buttons was accomplished by passing the correct object representing a pharmaceutical to the routines that determine alternative dosage forms or pharmaceuticals with the same active pharmaceutical agent. In contrast to a drug information tab, a drug compare tab obviously works with two pharmaceuticals instead of one.

The layout of the drug compare tab uses some methods of the drug information tab as well. At the top of the page the drug compare tab offers the same navigation list as the drug information tab. The questions heading groups of information on pharmaceuticals are the same. Additionally, we display the same footer that provides the button to go to the top of the page below each section of information. The differences to the drug information tab are the sections that provide the information on the pharmaceuticals. Instead of a single label we display two labels next to each other that provide the information for the first and the second pharmaceutical. Hence, the design should not cause confusion for users because it is consistent with the drug information tab. Moreover, it allows users to view information on two pharmaceuticals in parallel so that

both pharmaceuticals can be easily compared to each other. Moreover, with the configurations that are also used for drug information tabs, users can select only those sections of information for display in which they are interested.

Fig. 5-8: Final GUI and drug information tab design.

An approach that might have been even more convenient for users, would have been to programmatically detect differences between the information provided for the two pharmaceuticals. However, comparing the information on a character basis is not useful because the information to be inferred from the strings of characters needs to be compared. This is problematic because it depends on context, it is not clear how many characters convey some information, different strings of characters can hold the same information, and it is not known which information is contained in the strings. Therefore, since the information on pharmaceuticals provided by the application is only stored in strings, the database is not suitable to programmatically compare the provided information. Devising and implementing an algorithm for semantic comparison of the provided information probably entails enough challenges to be handled in a thesis of its own. Furthermore, with our approach users do not have to trust the algorithm to make

no mistakes like filtering relevant information that should be displayed. They can access all the information that they can also view in the drug information tab and they can determine the importance/relevance of the differences they identified and of the provided information on their own.

5.3.7 Printing Information on Pharmaceuticals

Printing the information provided on pharmaceuticals can be useful as well. Information can be printed in a well-readable font size and users can choose on their own which sections of information are important to/relevant for them and should be printed.

Obviously, a printout should not contain GUI elements because they do not provide relevant information and cannot be used on a sheet of paper. Accordingly, only the information provided on pharmaceuticals and not the sidebars, header, tabsheet, navigational elements in the drug information tab et cetera should be printed. This requires a new layout of the provided information without all GUI elements. Offering a HTML document with the respective information constitutes one possibility. However, representation of the HTML document depends on the browser. Additionally, the print functionality of the browser might also change the representation of the HTML document. To avoid such problems and due to the fact that we had to reorganise the information to be displayed in any case, we decided to offer the information in a file with the Portable Document Format (PDF[81]) instead of printing it directly. PDF files are designed to consistently represent documents independent of the creation or representation environment.[82] Thus, PDF files provide exactly the functionality we need – a consistent environment independent representation of a document. To create PDF files we used the iText[83] library, which provides functionality for PDF creation and can also parse HTML strings. To export the displayed information to a PDF file, users can click on the PDF icon in the top-right corner of drug information tabs, which is depicted in figure 5-8. Afterwards, functionality provided by the module 'Application Logic' creates a PDF document using the iText library. Basically, creation of the PDF is accomplished by parsing the HTML code representing the visible sections of information on pharmaceuticals and writing it to the PDF file instead of using it as

---

[81] For more information on the Portable Document Format see Adobe Systems Incorporated (2008).

[82] See Adobe Systems Incorporated (2008), p. 1.

[83] See 'http://itextpdf.com/itext.php' for more information on iText.

content of the labels in the drug information tab. The resulting PDF file is then made available to users as download and contains the information currently visible in the active drug information tab. Users can then either print the PDF document or use the PDF document directly.

5.3.8 Explanation of Technical Terms

As mentioned in chapter '2.1.2 Features and Drawbacks of Patient Information Leaflets', technical terms impede the comprehensibility of the provided information. The most effective way to avoid problems caused by technical terms would be to replace them with generally understandable expressions. However, this task should be performed by medical professionals with sound knowledge of the technical terms to ensure that correct substitutions are used and the meaning of the provided information is preserved. Additionally, users might be familiar with technical terms used to provide information that concerns them. For example, diabetics are likely to know that they should monitor their blood sugar levels if they are told to monitor their Glucose level and such information is probably irrelevant for other patients. Usage of technical terms might even increase comprehensibility in such situations because of an increased precision. Talking about shortness of breath instead of using the term 'asthma' would probably be more confusing than helpful for people used to the term 'asthma'.

Therefore, we decided not to replace technical terms. Instead, we provide a link titled 'Fachbegriff erläutern' below each section of information in the drug information tab next to the link that scrolls to the top of the page. This link can be seen on the bottom of figure 5-8. This link opens a pop-up where users can enter the technical term that should be explained. Afterwards, explanations for the technical term are provided in a new browser view that displays a corresponding entry on Wikipedia. This way users uninterested in explanations for a technical term cannot be bothered with unnecessary explanations. Moreover, users have access to more explanations than the web application could provide on its own. Furthermore, it is likely that at least one of the links will be displayed if users need an explanation for a technical term. Displaying the link in the header would, for example, be less convenient because users would have to go to the top of the page every time they want to look something up. The functionality could be easily adopted to use another source for explanations, like an own lexicon of technical terms or a different external encyclopaedia instead of Wikipedia/in addition to Wikipedia. We decided to link to Wikipedia because links corresponding to technical

terms can be easily determined and it offers information for a wide spectrum of terms. Development of an own lexicon cannot easily meet the breadth and depth offered by Wikipedia and should also be written by medical professionals for the previously mentioned reasons.

In order to avoid complications with technical words in the texts that provide application related information in the tooltips and the help tab, we tried to avoid technical terms and Anglicisms and used common alternatives. While these might seem odd to users adept at using online applications, which are mostly in English, inexperienced users, which need explanatory information the most, cannot be confused by technical terms in the explanatory texts.

5.3.9 Further changes/additions

A further additional feature can be seen in figure 5-8: An image that illustrates the packaging, blister, and tablet is provided between the navigation list and the PDF icon. This is, for example, useful for the identification of pharmaceuticals. If users find an unmarked red tablet, they might guess that it is a neo-angin tablet and can verify this guess with the web application. Since we did not have access to pictures for all pharmaceuticals we made only the one example picture, which is displayed in every drug information tab. With a lot of work, similar pictures could be made for every pharmaceutical and be stored in the database so that a picture could be provided for every pharmaceutical to support identification of pharmaceuticals.

Another advantage of such images is that they bring more colour to the drug information tab and liven up the design. This endeavour is further supported by the increased font size and changed colour of the drug information section headings as mentioned in chapter '5.1.2 Realisation of Requirements'. Additionally, we changed the design of the button for search initiation in the panel 'Medikamenten-Suche' to an icon of a magnifying lens. The new button can be seen in figure 5-8 as well as two further new buttons in the header. These buttons use also icons to convey to users that they can be used to open the home tab and the help tab. Hence, we made a few changes to spice up the GUI while still maintaining a conservative, legitimate, professional and trustworthy appearance. A too colourful design of a web application providing medical information might lead users to believe that the web application is unreliable and not

trustworthy. Still, the header should be redesigned as well and display some kind of banner or logo instead of just the simple plain heading 'Medinfo', but this is a job for an designer and not important for the prototype.

For users that want/need to contact the owner/operating authority of the web application in order to report mistakes in the provided information or ask a question about the web application we offer a contact pop-up that provides the necessary information. Additionally, we extended the tab providing information on the sources of the data and added some text explaining that users will not be profiled and collected data will not be applied beyond the scope of the application. The web application does not really collect information on the users, but it needs to handle the data necessary to process user events. Additionally, the log files of the application server and the database might store some information. However, these logs are necessary for maintenance and monitoring of performance. Hence, it would be inaccurate to tell the user that no information is stored.

5.3.10 Realisation of Requirements

With the implementation of the additional functionality and associated adoptions we realised the remaining relevant requirements. A search for a pharmaceutical with a partial name as well as the pop-up for selection of pharmaceuticals support the functionality demanded in requirement 02-02-03-01 and offer convenient ways for users to find specific pharmaceuticals without knowing the exact names. Additionally, the pop-up is useful for other features where the selection of pharmaceuticals is required. Checking a set of pharmaceuticals for adverse drug reactions (requirement 02-03-01-01) is realised with an appropriate algorithm and a tab for the composition of the set of pharmaceuticals. Moreover, the set composition tab makes functionality required by requirement 02-03-01-02 available that lists all side effects of the composed set of pharmaceuticals. Requirement 01-01-04-02 and 02-01-01-01 are realised by corresponding features that gather the required information and display alternative dosage forms, similar pharmaceuticals with alternative dosage forms, and pharmaceuticals with the same active pharmaceutical agents as the currently displayed pharmaceutical in pop-up windows. Comparison of similar pharmaceuticals, as mentioned in requirement 02-03-01-04-02 is facilitated with a tab derived from the drug information tab that displays sections of information for two pharmaceuticals in parallel. Alternatively, pharmaceuticals can be compared by viewing them in different

tabs. Printing of pharmaceuticals, as stated in requirement 02-03-05, is supported with a PDF export of the visible information in the currently viewed drug information tab. Requirement 01-05-03 that requests explanations for technical terms is realised with a pop-up that links to explanations for entered technical terms on Wikipedia. As postulated in requirement 01-05-01 we used common wording in the explanatory text. To support identification of pharmaceuticals (requirement 02-03-02), we demonstrated with a sample picture how pictures could be used to help with the identification of pharmaceuticals. In combination with icons and a changed colour of headings in drug information tabs, the pictures were also used to make the GUI design more attractive as mentioned in requirement 03-04-02. In accordance with requirements 02-04-01 and 03-01-02-02 the web application informs users how they can contact the operating authority and informs them about the sources as well as the usage of the provided/collected data.

Hence, we implemented features for all requirements except of requirement 01-03-01-02-04 that proposes to guide users through the application instead of offering them many different controls and settings. A problem with guiding users through the application is that it is easy to annoy users by asking them questions or asking them for configurations they are not interested in. Additionally, the home tab offers a short introduction and the possibility to configure the application. Furthermore, the GUI elements offer instructions for their utilisation with informative labels or tooltips and further help can be obtained from the help tab. Moreover, inexperienced users can use the application in the default view that offers only the basic functionality so that users should be easily able to comprehend the operation of the application and no guide is required. Later on, they can access and activate further functionality in the user interface with self-explanatory buttons or change the application configuration in the home tab. Accordingly, the application serves as an implicit guide through the realisation of other requirements and no extra feature that guides users has to be implemented.

ID	Requirement	Realisation
01-01-04-02	Display alternative dosage forms of a pharmaceutical.	Routine that obtains the necessary information, pop-up windows that display the results
01-03-01-02-04	Guide inexperienced users through the application rather than offering them many different controls and settings.	Instructions on home tab, different views, informative texts in GUI elements, tooltips, et cetera
01-05-01	Use common wording in explanatory text.	Use of common wording in explanatory text
01-05-03	Explain technical terms.	Pop-up for entry of technical terms that links to Wikipedia
02-01-01-01	Display pharmaceuticals with same active pharmaceutical agent as the currently displayed pharmaceutical.	Routine that obtains the necessary information, pop-up window that displays the resulting pharmaceuticals
02-02-03-01	Find specific pharmaceuticals without knowing the exact names.	Pop-up for selection of pharmaceuticals, search for pharmaceuticals with a partial name
02-03-01-01	Check a set of pharmaceuticals for adverse drug reactions.	Set composition tab, appropriate algorithm
02-03-01-02	List all side effects of a set of pharmaceuticals.	Set composition tab, listing of side effects in a tab
02-03-01-04-02	Compare similar pharmaceuticals.	Drug compare tab, displaying pharmaceuticals in different tabs
02-03-02	Support identification of pharmaceuticals.	Picture of pharmaceutical
02-03-05	Provide functionality to print the selected information of currently displayed pharmaceutical.	PDF export
02-04-01	Inform users how the owner/operating authority of the web application can be contacted.	Provision of corresponding information
03-01-02-02	Inform users that they will not be profiled and that collected data will not be applied beyond the scope of the application.	Provision of corresponding information
03-04-02	Make GUI design more diversified/attractive/appealing	Picture of pharmaceutical, icons, changed colour of headings

Tab. 5-4: Summary of requirements: additional functionality/associated adoptions

5.4 Concluding remarks

The preceding elaborations serve as a long answer to the third research question: How can the web application be realised? Our chosen approach started with the determination of relevant aspects and framework conditions as well as the development of a basic understanding of the proposed web application. This led to a set of objectives that drew an outline of the web application and were incorporated in the GUI prototype/the preliminary draft of the application. Afterwards, interviews were conducted to validate and improve the concept with user input/feedback and to elicit requirements for the web application. The gathered requirements were then sorted, consolidated, analysed and filtered. Subsequently, the requirements were prioritised and those mentioned most often were chosen for realisation. Thereafter, we developed a suitable basic design and architecture and selected fitting supportive technologies for implementation and deployment of the web application.

The actual implementation of the web application was then accomplished in three major phases. At first, we implemented the GUI. The user feedback and requirements collected in the interviews led to many improvements to the GUI of the GUI prototype. Yet, we could reuse and adopt large parts of the GUI prototype code and, thus, ease the implementation process by benefiting from previous efforts. In the next phase, we assembled the required information on pharmaceuticals in a database and implemented the necessary functionality for data retrieval and processing. Additionally, we integrated the database related functionality into the GUI so that, at the end of this implementation phase, the web application could serve its basic objective: enabling users to search and view information on pharmaceuticals usually provided in patient information leaflets. The final phase was dedicated to the implementation of additional functionality and the realisation of the remaining relevant requirements. While the previous implementation phases were necessary to make the web application operational, the third implementation phase led to the realisation of most of the features that leverage opportunities of providing the information with a web application to generate benefits for users.

6 Final Considerations and Remarks

Purpose of this chapter is to assess the resulting prototype as well as its development process and to propose further enhancements.

6.1 Achievement of Objectives

The motivation for development of the prototype was to demonstrate that a web application is suitable to provide information in patient information leaflets and offer additional functionality without exhibiting the drawbacks of patient information leaflets.

As described in chapter '2.1.2 Features and Drawbacks of Patient Information Leaflets', patients are troubled by the content, readability, and comprehensibility of patient information leaflets. Insufficiencies regarding the content of patient information leaflets are provision of too much content and too much irrelevant information. In the web application, we solved this problem by allowing users to select which information should be displayed so that users can decide which information is or is not relevant. Along with the provision of too much information, the small font size and the confusing structure, which impedes discovery of desired information, deteriorate the readability of patient information leaflets. The web application prevents these deficits by offering an adjustable font size, a well-arranged layout, which bundles related information, and the navigation header that allows users to bring desired information into view with a single click. Furthermore, a feature that exports the visible patient information leaflet information to a PDF document enables users to benefit from the refined representation offline.

Provision of the information with a web application is not the right approach to solve the comprehensibility troubles. Inadequate, formal language and usage of technical terms are problems rooted in the data itself and should be solved by improving the data instead of devising workarounds and manipulating the data to make it presentable. However, we improved comprehensibility a little by enabling users to retrieve explanations for technical terms. Additionally, by using common expressions and terms in the application-related explanatory information, we avoided creating similar problems ourselves. Data related issues will be discussed in more detail in chapter '6.5.1 Improvement of Data Quality'.

A better presentation of the information is not the only advantage of providing the information in patient information leaflets with a web application. While a patient information leaflet is focused on a single pharmaceutical, the web application can process and aggregate information on multiple pharmaceuticals. The developed prototype demonstrates this with a few features: Pharmaceuticals can be searched by different parameters. Besides searching for a pharmaceutical to view the patient information leaflet information, users can, for example, also use the search functionality to find and list pharmaceuticals for a specific field of application. Pharmaceuticals can also be searched by partial names, which will, for example, be useful if users do not know the exact name. Moreover, the application enables users to find alternative dosage forms of a pharmaceutical by listing all dosage forms of similar pharmaceuticals. Additionally, pharmaceuticals with the same active pharmaceutical agents as another pharmaceutical can be easily determined.

Furthermore, the application provides support for users who take multiple pharmaceuticals in parallel: The web application presents information on all pharmaceuticals in a consistent layout. Retrieving the desired information from multiple patient information leaflets, on the other hand, is less convenient and comfortable because they are likely to be structured in different ways so that identification of the desired information is impeded. Through the support for tabs, the web application additionally enables users to switch efficiently between information on different pharmaceuticals. A further feature that displays information on two pharmaceuticals in parallel and arranges congruent sections of information for both pharmaceuticals next to each other can be utilised to compare two pharmaceuticals. Users can also check a set of pharmaceuticals, which could, for example, be comprised of all the pharmaceuticals they are taking, for adverse drug reactions or display all associated side effects in a single tab. These features satisfy needs identified and mentioned most often in the interviews. Further aggregation functionality can be added as long as the database provides the required information or can be extended in an adequate way.

6.2 Patient-Friendliness

The web application should be beneficial for patients, hence, it should provide functionality and information patients need. Additionally, it should be easy as well as comfortable to use and should not require special training or skills.

In chapter '6.1 Achievement of Objectives', we already mentioned that we tried to avoid problems patients associate with patient information leaflets to enhance the provision of information in patient information leaflets. Furthermore, the conducted interviews were useful to improve the alignment of the application with patients expectations and needs. Through the interviews, we were able to identify expectations and requirements regarding the application and gathered feedback on usability, necessary instructions, and user preferences. A problem, we identified in the interviews, is that users have a varying proficiency in using web applications. Less skilled users are easily confused by too many possibilities and options. On the other hand, more skilled users are easily bothered with an inefficient GUI in which the required functionality is not directly accessible. To satisfy most kinds of users, we implemented a modular GUI that users can configure to their liking. Inexperienced users can start with a basic view and enable further GUI components step by step/when they need to. The possible configurations are displayed in combination with introductory information on usage and purpose of the application at application startup. In combination with the tooltips, instructive information in the tabs, and the help tab users are supplied with sufficient information to be able to use the application.

Hence, the application can be considered patient-friendly since we provide functionality elicited from prospective users and the application is easy to use and can be adopted to the needs of users.

6.3 Chosen Approach

The chosen approach can be deemed suitable because it took needs of users/patients into account and led to a proper, working prototype within the given time frame. Thus, it led to the accomplishment of our objective. Establishment of objectives for the preliminary draft and the implementation of the GUI prototype were useful to become acquainted with the problem and subject area as well as the Vaadin framework and the Java programming language. The interviews were important to validate and improve the

initial solution, to elicit further functionality desired by potential users, and to determine usability problems. Being able to interact with interviewees and observe their interaction with the GUI prototype was an additional advantage of the interviews. It enabled us to ensure that interviewees understood the questions, stayed on topic, and could clarify their responses and to address potential problems that were observed by us but not mentioned by the interviewees. However, conducting the interviews and especially the following analysis created a lot of effort. This might be improved by noting requirements directly during the interview instead of listening to recordings and extracting requirements later on. On the other hand, such approach would probably lead to omission of requirements and worsen the interview quality because the interviewer needs to focus more on taking notes.

Normally, the design and architecture should be specified in more detail, but in our case this was not necessary since we did not have to coordinate multiple developers, the modular architecture did not lead to many interdependencies, and it did not cause problems during implementation. Implementing the user interface first was beneficial because, this way, a working version of the application could always be tested. Additionally, further functionality could be integrated and tested step by step in its targeted environment and it was not necessary to integrate a lot of features at once, which could cause unforeseen problems with many potential sources.

6.4 Comparison with Similar Services

We found two freely accessible online services that offer similar functionality: PhamNet.bund[84] and GELBE LISTE PHARMINDEX[85].

PharmNet.bund offers patient information leaflets as download. These contain the same information as those distributed with pharmaceuticals but have an improved readability because of a larger font size. Hence, using PharmNet.bund is not much different from using a patient information leaflet and it does not provide additional services related to the information in patient information leaflets and cannot be used to view the information in a browser.

---

[84] See 'http://www.pharmnet-bund.de/dynamic/de/am-info-system/index.html' for further information.

[85] See 'www.gelbe-liste.de' for further information.

GELBE LISTE PHARMINDEX is more similar to our application and enables users to search for pharmaceuticals and display patient information leaflet information in the browser. The provided information is similar to the information our application provides since we use the same database. Without a user account GELBE LISTE PHARMINDEX provides only information on dosage form, available package sizes, ingredients, ATC code, and an image of the pharmaceutical in its dosage form. However, further information and functionality can be accessed by creating a user account. With a user account the same information on pharmaceuticals as our application provides can be accessed. Additionally, pharmaceuticals link to pharmaceuticals with a matching active pharmaceutical ingredient and the price of a pharmaceutical can be compared with prices of similar pharmaceuticals.

The main difference to our prototype is that GELBE LISTE PHARMINDEX targets medical professionals instead of patients. Besides not requiring a user account our prototype has some advantages. First of all, users can choose which information they want to see so that irrelevant information is not displayed. As depicted in figure 6-1, GELBE LISTE PHARMINDEX lists only headings and strings of information below each other. With our layout the information is arranged more clearly because only the selected information is displayed and information with related content is ordered in groups with helpful and meaningful questions as heading. Additionally, readability is improved because the layout is less dense and information is structured with lists and tables wherever it is useful, reasonable, and possible. The prototype also enables users to change the font size and configure the GUI to their liking. To support users, the prototype provides help information regarding the usage of the application as well as the meaning of the different sections of information and can link to explanations for technical terms. Moreover, the prototype offers more functionality that potential users deemed useful: Users can efficiently switch between information on multiple pharmaceuticals by using the tabs. Two pharmaceuticals can be compared in a single tab. A set of pharmaceuticals can be checked for adverse drug reactions. Side effects can be displayed for a set of pharmaceuticals. Furthermore, alternative dosage forms of a pharmaceutical can be easily determined.

---

[86] Screenshot of web page 'http://www.gelbe-liste.de'. Taken on 27.01.2012.

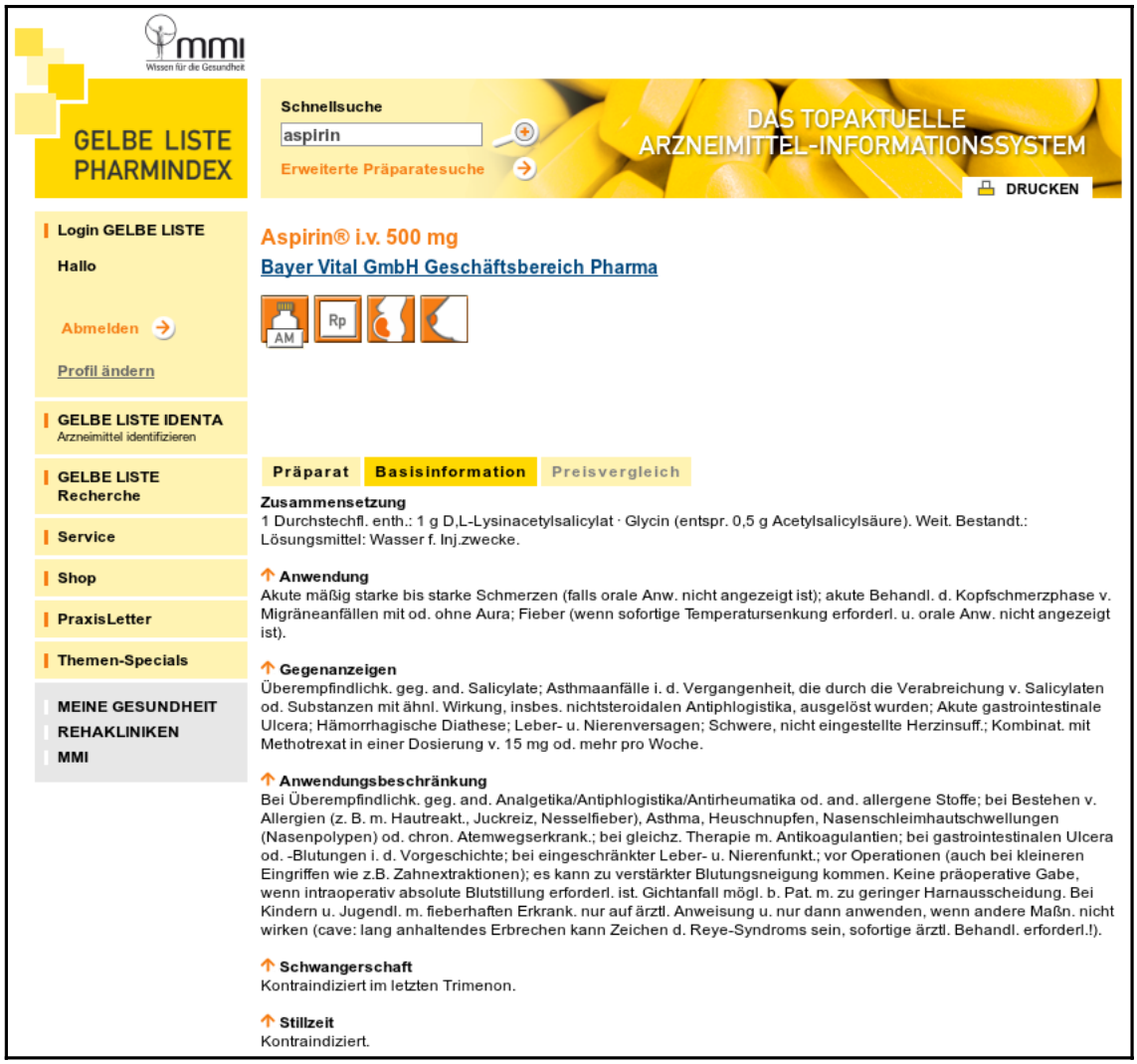

Fig. 6-1: Layout of GELBE LISTE PHARMINDEX[86]

In short, the developed prototype is more useful for patients because it is designed for patients and not for medical professionals. Thus, it exhibits an improved usability, its operation is more intuitive, and it does a better job mending the troubles patients have with patient information leaflets, as described in chapter '6.1 Achievement of Objectives'.

6.5 Further Research

We developed a prototype for the patient-friendly provision of information in patient information leaflets in a web application. Besides requiring tasks like further validation with user feedback or realisation of the adoptions necessary to run the application in the target environment, the prototype can serve as foundation for further improvements and considerations. In the following we will briefly propose some possibilities.

6.5.1 Improvement of Data Quality

While the current quality and quantity of the data is sufficient for the purposes of the prototype there is room left for improvement: For live operation a satisfying amount of information should be provided for all pharmaceuticals in the database. In the prototype only roughly a quarter of the pharmaceuticals are associated with content for most sections of information in the drug information tab. This coverage could be improved by identifying data associated with pharmaceuticals that is also suitable for other pharmaceuticals that are not linked with the data. The prototype provides, for example, a satisfying amount of information for 'Aspirin 0.5g Tbl. kohlpharma' and only basic information for 'Aspirin 0.5g Gerke Tbl.'. The information provided for the 'kohlpharma' version is probably also suitable for the 'Gerke' version. If such relationships could be identified in a reliable way the coverage of the provided information could be easily increased. In order to avoid providing wrong information, such relationships should be identified or approved by medical professionals.

Updating the data in the database and adding further data on pharmaceuticals is an interesting aspect as well. The web application could be supplemented with a further application for data input. Such functionality could, for example, be designed in a way that allows to determine additional semantic aspects of the entered information. It would, for example, be useful to associate side effects with their frequency of occurrence in order to enable users to select only side effects with a certain frequency for display. Another example is associating taking instructions with parts of the day. This would be useful to display icons that visualise the information and illustrate that a pharmaceutical should be taken in the morning, in the morning and in the evening, after a meal, et cetera. It would also be useful to implement relationships that can be used to display informative icons/information at the top of the drug information tab, like 'Do not operate heavy machinery' or 'Do not use during pregnancy'. With the current database

we can only inform the user that the application provides some information for a certain topic, but we cannot deduce whether the information is important or relevant for the user. Hence, in our case, icons or pictograms are not really useful because users already see whether the sections contain text.

In any case, added information needs to originate from a reliable source or should be verified by experts to ensure that the application provides correct information. Accordingly, programmatic improvement of the data is complicated because the information depends on context and is provided by ambiguous words so that it is difficult to ensure that an algorithm performs reliable and correct transformations. Yet, the sheer mass of necessary information calls for an efficient approach. A possible solution would be to obtain the required information from the respective pharmaceutical companies in electronic form and to implement input routines that update the database accordingly. However, some companies might be reluctant to spend the additional effort for making the information available. A crowd sourcing approach is a further possible solution. Wikipedia is a good example that such an approach can work. Nevertheless, crowd sourcing is problematic in combination with medical information because provision of wrong information could lead to serious consequences. A crowd sourcing approach requires definitely an additional mechanism that ensures the reliability of the information. This could be achieved by requiring a certain amount of approved experts to confirm new or updated entries prior to publishing them.

6.5.2 German Healthcare Telematics Infrastructure

Integration of the web application with the German healthcare telematics infrastructure that is currently being established and the electronic health card currently being issued to German citizens could lead to further promising applications of the prototype. The application could, for example, retrieve information for users who identify themselves with their electronic health card and offer personalised information. The application could, for example, automatically offer information on the pharmaceuticals a user is taking. Special medical conditions of users could be considered by the application so that users can, for example, be informed that taking the currently viewed pharmaceutical is not advisable because of a medical condition they have. Users could also be informed that the currently viewed pharmaceutical has known adverse drug reactions with pharmaceuticals they are currently taking. Furthermore, physicians could supplement the presented information with additions for individual patients like an alternative

dosage pattern, symptoms a patient should look for, or just substantiate why a patient should take a certain pharmaceutical. If patients want to use such functionality and identify themselves, it will also be possible to offer further functionality that requires user accounts. Users could for example define a preferred view and a set of tabs that should be displayed on application startup. However, such functionality raises data security concerns that need to be managed. It would, for example, be necessary to ensure that patients and others can only access the information they provided/the personalised information provided for them.

6.5.3 Different Perspectives

The application should also be evaluated from different perspectives. It would be worthy of consideration to determine and weigh the advantages and disadvantages of providing easier access to information on pharmaceuticals. The application could, for example, improve self medication or encourage undesirable forms of self medication. Compliance might be improved through the application because of easier access to information or patients might get scared for the same reasons.

A legal perspective is interesting as well. It should, for example, be clarified who is responsible if a user harms himself by mistake because he followed the taking instructions for the wrong pharmaceutical. A further interesting aspect is whether the operating authority of the web application, which offers the information, or the source, which created the information, is liable for providing wrong information. A related aspect is how users are best informed that wrong information might be provided and that they should take this into account.

Especially when personalised functionality is added security aspects of the application should be considered to ensure that third parties cannot gain unauthorised access to information on users/patients. Security measures to ensure that the provided information cannot be manipulated should be specified as well. Currently, the database can be configured to allow access only for nodes in the local network, but this gets more complicated if servers in different geographical locations are used. Moreover, it might be reasonable to protect the individual nodes against physical manipulations.

The prior suggestions for further research touch some aspects that came to mind while working on the thesis. The developed prototype can possibly serve as foundation for many more related and supplemental projects.

Bibliography

Adobe Systems Incorporated (2008)

Adobe Systems Incorporated: Document management – Portable document format – Part 1: PDF 1.7. http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf, retrieved on 24.01.2008

Apache Software Foundation (n.d.)

Apache Software Foundation: The Apache Tomcat Connector – Generic HowTo. Workers HowTo. http://tomcat.apache.org/connectors-doc/generic_howto/workers.html, retrieved on 23.10.2011

Bäumer et al. (1996)

Dirk Bäumer, Walter R. Bischofberger, Horst Lichter, Heinz Züllinghoven: User Interface Prototyping – Concepts, Tools, and Experience. In: IEEE Computer Society (publ.): Proceedings of the 18th International Conference on Software Engineering (ICSE '96), 25-30 March, 1996, Berlin. Washington, DC, USA 1996, p. 532-541

Bundestag/Bundesrat (1976)

Bundestag/Bundesrat: Arzneimittelgesetz in der Fassung der Bekanntmachung vom 12. Dezember 2005 (BGBI. I S. 3394), das zuletzt durch Artikel 1 der Verordnung vom 19. Juli 2011 (BGBI. I S. 1398) geändert worden ist. n.p. 1976

Cohene, Easterbrook (2005)

Tira Cohene, Steve Easterbrook: Contextual Risk Analysis for Interview Design. In: IEEE Computer Society (publ.): Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE '05), 29 August – 2 September, 2005, Paris, France. Washington, DC, USA 2005, p. 95-104

Davis et al. (2006)

Alan Davis, Oscar Dieste, Ann Hickey, Natalia Juristo, Ana M. Moreno: Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. In: IEEE Computer Society (publ.): Proceedings of the 14th IEEE International Requirements Engineering Conference (RE '06), 11-15 September, 2006, Minneapolis/St. Paul, MN, USA. Washington, DC, USA 2006, p. 179-188

Deutsche Telekom (2011)

Deutsche Telekom: Sicherheitsreport 2011. http://www.download-telekom.de/dt/StaticPage/10/84/08/T-Systems_Sicherheitsreport_2011.pdf_1084082.pdf, retrieved on 19.09.2011

European Commission (2009)

European Commission: Guideline on the Readability of the Labelling and Package Leaflet of Medicinal Products for Human Use. Brussels, 2009

Fuchs et al. (2007)

Jörg Fuchs, Stefanie Banow, Nancy Görbert, Marion Hippius: Importance of Package Insert Information in the European Union. In: Pharmazeutische Industrie. no. 2, vol. 69, 2007, p. 165-172

Fuchs, Hippius, Schaefer (2003)

Jörg Fuchs, Marion Hippius, Marion Schaefer: Gestaltung von Packungsbeilagen für Arzneimittel. In: Pharmazeutische Industrie. no. 4, vol. 65, 2003, p. 302-306

Grönroos (2011)

Marko Grönross: Book of Vaadin. 4th edition. https://vaadin.com/download/book-of-vaadin/current/pdf/book-of-vaadin.pdf, retrieved on 07.12.2011

Hove, Anda (2005)

Siw E Hove, Bente Anda: Experiences from Conducting Semi-Structured Interviews in Empirical Software Engineering Research. In: IEEE Computer Society (publ.): Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS '05), 19-22 September, 2005, Como, Italy. Washington, DC, USA 2005, p. 23-32

IEEE (1998)

IEEE (publ.): IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998. New York, NY, USA 1998

IEEE (2010)

IEEE (publ.): Systems and software engineering – Vocabulary. ISO/IEC/IEEE 24765-2010(E). New York, NY, USA 2010

Kaletsch, Sunyaev (2011)

Alexander Kaletsch, Ali Sunyaev: Privacy Engineering: Personal Health Records in Cloud Computing Environments. In: no publ.: ICIS 2011 Proceedings, 04-07 December, 2011, Shanghai, China. n.p. 2011, paper 2

Nink, Schröder (2005)

Katrin Nink, Helmut Schröder: Zu Risiken und Nebenwirkungen: Lesen Sie die Packungsbeilage?. Bonn 2005

Oracle (2011)

Oracle (publ.): MySQL 5.5 Reference Manual. Including MySQL Cluster 7.2 NDB Reference Guide. revision 27785; Redwood City, CA, USA 2011

Rupp (2002)

Chris Rupp: Requirements-Engineering und -Management. Professionelle, iterative Anforderungsanalyse für die Praxis. 2nd edition, München, Wien, 2002

Sommerville (2007)

Ian Sommerville: Software Engineering. 8th edition, München, 2007

Sun Microsystems (2001)

Sun Microsystems (publ.): Java Servlet Specification, Version 2.3. JSR 53. Palo Alto, CA, USA 2001

Zhang (2000)

Wensong Zhang: Linux Virtual Server for Scalable Network Services. In: no publ.: Proceedings of the Linux Symposium, 19-22 July, 2000, Ottawa. n.p., 2000, p. 1-10

Appendix

List of objectives

Category	#	Objective
Data	1	Make relevant information in patient information leaflets available for the user.
	12	Do not collect and store data that is not relevant for the display of information in patient information leaflets.
	13	Use only trustworthy sources of information.
	14	Protect data against tampering.
	16	Inform users about data sources and potential risks associated with the application.
Interface Design Principles	5	Structure the layout according to user wishes and needs.
	6	Improve structuring with accentuated and consistent formatting of headings.
	8	Avoid or explain technical terms or notations.
	9	Obtain readability with clear contrast between text and background colour.
	10	Provide assistance for inexperienced users.
	11	Avoid hindering or slowing down experienced users.
Functionality	2	Provide aggregation functionality for information in patient information leaflets.
	3	Provide refinement functionality for information in patient information leaflets.
	4	Offer functionality for the selection of the displayed information.
	7	Provide zoom capabilities or a sufficient font size.
	15	Provide a mechanism to report potentially wrong information.
Application Design	17	The web application needs to be accessible around the clock.
	18	Response times should be short (almost unnoticeable).
	19	The architecture needs to be scalable.
	20	The architecture needs to be expandable.

Tab. 7-1: List of objectives on which the preliminary draft is based.

Interview Guide

Purpose of the interview guide is to guide the interviewer in the interview. It depicts the general structure of the interview and suggests questions that can be rephrased to adapt to the needs of the interviewee. Additionally, the interview guide is useful to take short notes during the interview and to associate the notes directly with the questions. The interview was conducted in German, therefore, the interview guide is in German too.

A) Einführung

I. Begrüßung/Vorstellung

II. Dank für Teilnahme

III. Bitte um Erlaubnis zur Aufzeichnung

IV. Beschreibung der Arbeit

Entwicklung einer Web-Applikation zur patientenfreundlichen Darstellung von Informationen in Beipackzetteln von Medikamenten

V. Beschreibung des Zweckes des Interviews

(1) Erhebung von Anforderungen/Ideen, um die Webseite benutzerfreundlicher zu gestalten

Anforderung: Bedingung oder Fähigkeit, die ein Benutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen.

1. Keine weiteren Einschränkungen
2. Keine Implementierungsdetails
3. Korrekt
4. Eindeutig
5. Komplett
6. Konsistent
7. Verifizierbar

(2) Um später Anforderungen genau aufschreiben zu können, werde ich teilweise um Detaillierung einiger Aussagen bitten

(3) Es gibt keine falschen Antworten

(4) Anonymität, Vertraulichkeit

(5) Bitte um freie Meinungsäußerung

(6) Ungefährer Zeitaufwand für das Interview: 60-90 Minuten

VI. Beschreibung des Ablaufs des Interviews

(1) Allgemeine Fragen zur Web-Applikation

(2) Vorstellung eines auf meinen Annahmen basierenden Prototypen

(3) Weitere Fragen zur Anwendung mit Berücksichtigung des GUI-Prototypen

(4) Möglichkeit für weitere Kommentare, Ideen oder Anmerkungen

VII. Allgemeine Daten zur Beschreibung der Gruppe der Befragten

(1) Geschlecht

(2) Alter

(3) Beruf

(4) Internetnutzung

(a) gar nicht

(b) wenn ich muss

(c) wenn es die bessere Alternative ist

(d) so oft es geht

(e) anderes

B) Fragen vor Präsentation von GUI-Prototyp

I. Welche Probleme gibt es bei der Benutzung von Beipackzetteln für Medikamente?

(1) Wann haben Sie zuletzt einen Beipackzettel gelesen?

(2) Hat Sie dabei etwas gestört?

II. Wie könnten die Probleme durch eine Web-Applikation zur Darstellung von Informationen in Beipackzetteln behoben werden?

III. Wie stellen Sie sich die beschriebene Web-Applikation vor?

IV. Was müsste die Web-Applikation leisten, damit diese benutzt wird?

(1) In welchen Situationen kann die beschriebene Web-Applikation nützlich sein?

(2) Szenario: Ein Patient möchte vor der Einnahme eines Medikamentes die Web-Applikation nutzen, um sich über ein Medikament zu informieren.

(a) Wie sollten die Informationen dargestellt werden, um den Patienten dabei zu unterstützen?

(b) Wie sollten die gesuchten Informationen ausgewählt werden?

(c) Mit welchen weiteren Maßnahmen könnte man den Patienten unterstützen?

(3) Welche zusätzliche Funktion neben der reinen Darstellung von Informationen in Beipackzetteln können Sie sich vorstellen?

C) Präsentation von GUI-Prototyp

I. Start des Prototypen auf PC des Befragten oder alternativ auf eigenem Laptop und parallel Erklärungen zu dem Prototyp

(1) http://alturl.com/j6t2e

(2) Basiert auf meinen Ideen zur Gestaltung der Web-Applikation

(3) Soll ein Gefühl für die Richtung der Entwicklung vermitteln

(4) Dient nur zu Testzwecken

(5) Stellt nur Testinformationen dar

(6) Man kann bei der Benutzung nichts zerstören

II. Aufforderung, die Web-Applikation zu benutzen bzw. ein Gefühl für diese zu bekommen

(1) Erkundung der Web-Applikation nach Belieben

(2) Aufforderung zum Stellen von Fragen, falls Probleme auftreten

(3) Falls der Befragte zögerlich mit der Web-Applikation interagiert

(a) Aufforderung, Informationen zu einem Medikament aufzurufen

(b) Aufforderung, die unterschiedlichen Funktionen zu testen

(c) Fragen, ob die einzelnen Funktionen der GUI-Komponenten erkannt werden.

Beispiel: Wie würden Sie vorgehen, wenn sie Informationen zu einem Medikament aufrufen möchten?

D) Fragen nach Präsentation von GUI-Prototyp

I. Abweichungen zwischen GUI-Prototyp und vorherigen Antworten

Entspricht der GUI-Prototyp Ihrer Vorstellung, bevor Sie den Prototyp gesehen haben?

(a) Was haben Sie sich anders vorgestellt?

(b) Was sollte anders gemacht werden?

(c) Was kann weggelassen werden?

II. Spezifische Fragen zu Aspekten der Anwendung

(1) Kann man mit dem Prototypen benutzerfreundlich Informationen zu Medikamenten aufrufen?

(a) Szenario: Ein Patient nutzt die Web-Applikation, um sich über Medikamente zu informieren. Nach ein paar Sitzungen entscheidet er aber doch lieber Beipackzettel zu lesen.

1. Was sind die wahrscheinlichsten Dinge, die dem Patienten an der Web-Applikation nicht gefallen?

2. Wie kann man diese vermeiden?

(b) Wenn Sie an die Benutzerführung in anderen Programmen, die Sie benutzen, denken: Fallen Ihnen dann Maßnahmen ein, wie man die Benutzerführung verbessern könnte?

(c) Gibt es Funktionalität, die Sie vermisst haben?

(2) Aggregation und Aufarbeitung

(a) Aggregation von Informationen kann als die Zusammenfassung von einzelnen Informationen beschrieben werden.

Inwiefern kann die Aggregation/Zusammenfassung von Informationen im Rahmen der Web-Applikation hilfreich sein?

- Beispiel: Liste aller Medikamente, die bei Kopfschmerzen helfen
- Beispiel: Anzeige der verschiedenen Darreichungsformen von Aspirin
- Beispiel: Auflistung der Nebenwirkungen aller Medikamente, die ein Patient parallel einnimmt

(b) Wie könnten die dargestellten Informationen verbessert werden?

1. Wie könnte man die Informationen benutzerfreundlicher darstellen?
   - Ist die Sortierung der Informationen in Ordnung?
2. Mit welchen Zusatz-Informationen, die über den Inhalt von Beipackzetteln hinaus gehen, könnte man den Patienten besser informieren?
3. Wie könnte man die Qualität der bereitgestellten Informationen verbessern?

III. Nutzung

(1) Aspekte:

(a) Sicherheitsbedenken:

1. Korrektheit der Informationen
   - Für wie verlässlich halten Sie die Informationen?
   - Wie könnte man die wahrgenommene Verlässlichkeit erhöhen?
   - Wie könnten User auf falsche Informationen aufmerksam machen?

2. Aktualität der Informationen
3. Unerwünschte Datenerfassung

(b) Performanz:

Sollte die Web-Applikation schneller auf Benutzereingaben reagieren?

(c) Verfügbarkeit:

Wie würden Sie reagieren, wenn die Web-Applikation mal nicht erreichbar ist, wenn Sie diese benutzen möchten?

(d) Benutzerfreundlichkeit:

Wie würden Sie unerfahrene Benutzer unterstützen?

- Dabei aber keine Beeinträchtigung von erfahrenen Benutzern.

E) Freie Kommentare/Diskussion

I. Haben Sie noch weitere Fragen oder Bemerkungen?

II. Angenommen, die Webseite ist fertiggestellt und bietet vernünftige Informationen: Würden Sie die Web-Applikation nutzen?

F) Endbemerkungen

G) Feedback

List of Requirements

ID	Requirement
01-01-04-02	Display alternative dosage forms of a pharmaceutical.
01-01-05	Enable the user to go to a specific section of drug information.
01-01-07	Do not display more information than the user needs.
01-01-07-02	Allow users to filter side effects by frequency of occurrence.
01-02-01	Allow for adjustment of font size.
01-02-02	Clear structuring of text.
01-02-02-01	Accentuate sub-headings in drug information tab.
01-03-01-02	Hide functionality that might confuse inexperienced users.
01-03-01-02-04	Guide inexperienced users through the application rather than offering them many different controls and settings.
01-03-01-03	Make the search field in panel 'Medikamenten-Auswahl' clearly noticeable.
01-03-03	Display close buttons/icons for everything that can be closed.
01-03-04-01	Intuitive functionality for creation of new tabs.
01-03-06-01	Display introductory information at application startup.
01-05-01	Use common wording in explanatory text.
01-05-02	Avoid abbreviations in text.
01-05-03	Explain technical terms.
02-01-01-01	Display pharmaceuticals with same active pharmaceutical agent as the currently displayed pharmaceutical.
02-02-01-01	Search for pharmaceuticals by field of application.
02-02-01-02	Search for pharmaceuticals by name.
02-02-01-04	Search for pharmaceuticals by active pharmaceutical agent.
02-02-03-01	Find specific pharmaceuticals without knowing the exact names.
02-03-01-01	Check a set of pharmaceuticals for adverse drug reactions.
02-03-01-02	List all side effects of a set of pharmaceuticals.
02-03-01-04-01	Offer price comparison of similar pharmaceuticals.
02-03-01-04-02	Compare similar pharmaceuticals.
02-03-02	Support identification of pharmaceuticals.
02-03-05	Provide functionality to print the selected information of currently displayed pharmaceutical.
02-04-01	Inform users how the owner/operating authority of the web application can be contacted.
02-05-02	Provide experiences of other users with the application or specific pharmaceuticals.

ID	Requirement
03-01-02-02	Inform users that they will not be profiled and that collected data will not be applied beyond the scope of the application.
03-04-02	Make GUI design more diversified/attractive/appealing

Tab. 7-2: Requirements mentioned most often during requirements elicitation.

Mapping of relevant requirements to objectives for preliminary draft

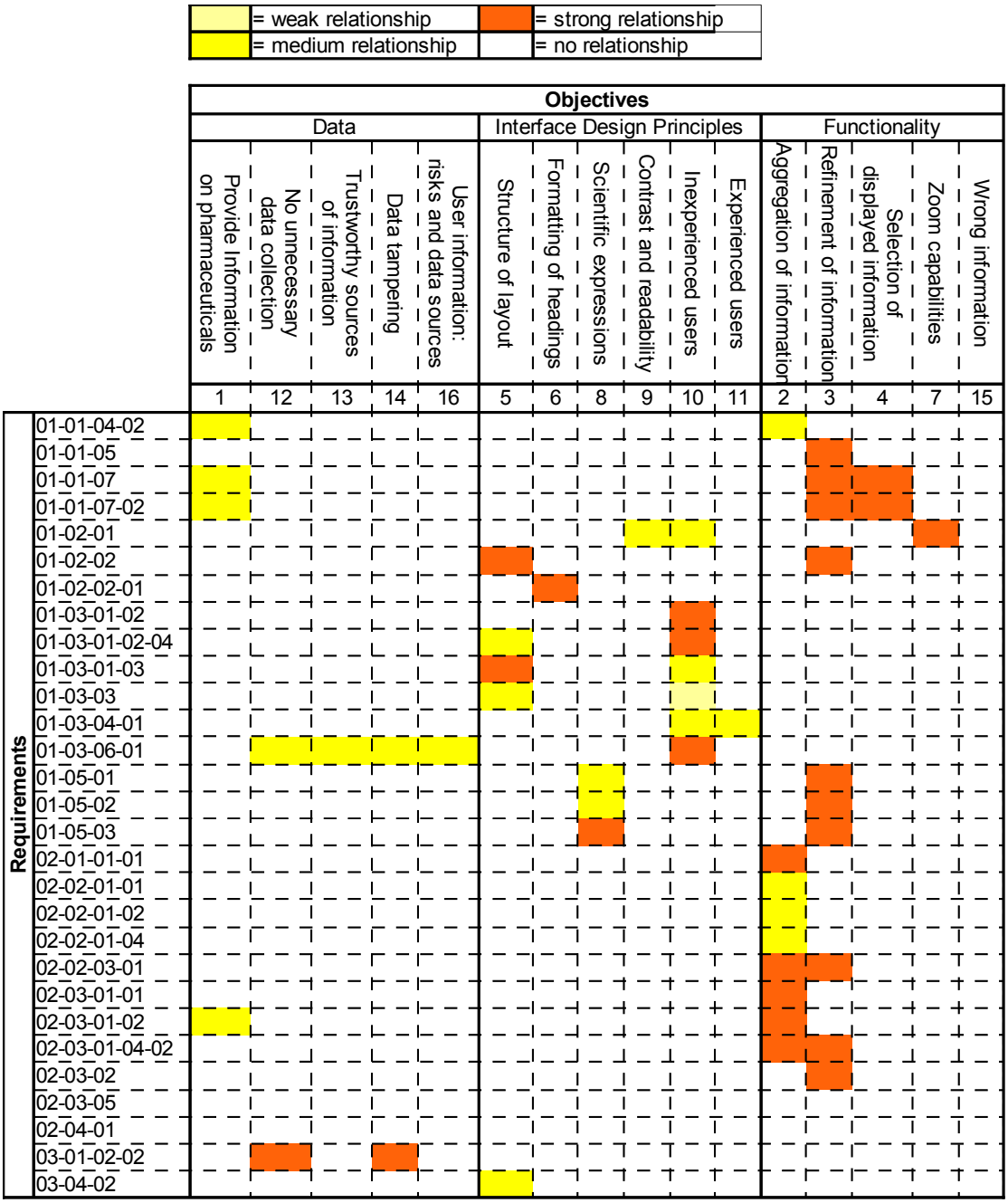

Fig. 7-1: Mapping of relevant requirements to objectives for preliminary draft. (Objectives for application design are not listed because we collected no requirements for them.)

Complete list of Elicited Requirements

- Usability – 01
  - Drug Information Tab – 01
    - Structuring – 01
      - Requirement 01-01-01-01
        - Well-arranged ordering of information on pharmaceuticals
        - Sources: 1
        - Description:
          - Order information in drug information tab in a way that makes sense
          - Example: Use the ordering of package information leaflets
          - Example: Order information in sequence of relevance, i.e., information necessary prior to intake, information concerning intake, information relevant after intake, ...
      - Requirement 01-01-01-02
        - Provide short, summarised information on top of drug information tab
        - Sources: 1
        - Description:
          - Summarise information provided in drug information tab
          - Display the summarised information on top of drug information tab
          - Example: Provide an icon on top of the drug information tab if information concerning pregnancy is available.
          - Example: Provide an icon on top of the drug information tab if ability to drive can be affected by pharmaceutical.

- Requirement 01-01-01-03
  - Replace question 'Um welches Arzneimittel handelt es sich?' with name of pharmaceutical.
  - Sources: 1
  - Description:
    - The user wants to be able to clearly identify the pharmaceutical so that the name of the pharmaceutical is the most important information.
    - The question 'Um welches Arzneimittel handelt es sich?' is not necessary because the name of the pharmaceutical can serve a similar purpose.
- Requirement 01-01-01-06
  - Reposition section dosage form
  - Sources: 2
  - Description:
    - The dosage form is related to the dosage information so that dosage form and dosage information should be listed next to each other.
- Requirement 01-01-01-07
  - Do not contain dosage form and package size in name of pharmaceutical
  - Sources: 1
  - Description:
    - Do not list Aspirin N300 as 'Aspirin N300' rather than 'Aspirin N300 Tbl. N3'

◆ Requirement 01-01-02

- Highlight information related to medical conditions or allergies specified by the user
- Sources: 1

- Description:
  - Information on a pharmaceutical concerning medical conditions or allergies of a user is highly relevant for the user.
  - Specify this information in another colour, underlined, greater font size or special section.

◆ Requirement 01-01-03

- Emphasise warning messages
- Sources: 1
- Description:
  - Warning messages are important for the user.
  - Users are used to recognise messages written in red font colour as warning messages.
  - Thus, display warning messages in red font colour.

◆ Provided Information - 04

- Requirement 01-01-04-01
  - Display alternative package sizes of a pharmaceutical
  - Sources: 1
  - Description:
    - Inform the user if the pharmaceutical is also available in different package sizes.
    - Enable the user to see the different package sizes.
- Requirement 01-01-04-02
  - Display alternative dosage forms of a pharmaceutical
  - Sources: 7
  - Description:
    - Inform the user that the pharmaceutical is also available in other dosage forms.
    - Enable the user to see the other available dosage forms.

- Requirement 01-01-04-03
  - Display whether pharmaceutical requires prescription
  - Sources: 2
  - Description:
    - User might be interested in knowing whether he can freely procure the pharmaceutical or needs to contact a physician first.
- Requirement 01-01-04-04
  - Provide information on mode of action of pharmaceutical
  - Sources: 1
  - Description:
    - Describe how a pharmaceutical works
- Requirement 01-01-04-05
  - Display pharmaceuticals with different intensity
  - Sources: 1
  - Description:
    - Inform the user that the pharmaceutical is also available with a different intensity.
    - Enable the user to see the versions with different intensity.
    - Different intensity = Higher or lower used amount
- Requirement 01-01-04-06
  - Inform user whether it is allowed to pestle, divide, or dissolve a pharmaceutical to ease taking of pharmaceutical
  - Sources: 2

  - Description:
    - Some people might have problems to swallow a pharmaceutical
    - It would be useful to know whether a capsule can be opened and its content can be dissolved in a fluid or a tablet can be divided or pesteled to ease taking of the pharmaceutical

- Requirement 01-01-04-08
  - Provide a list of pharmaceuticals that contain a specified active pharmaceutical agent or comprise a group of pharmaceuticals
  - Sources: 1
  - Description:
    - If a contraindication states that a pharmaceutical should not be taken with another pharmaceutical that contains a distinct active pharmaceutical agent the user should be able to easily determine the pharmaceuticals containing this active pharmaceutical agent. For example by clicking on the listed active pharmaceutical agent.
    - When a group of pharmaceuticals, like anti-inflammatory drugs, is mentioned the user should be able to list the pharmaceuticals that comprise this group.
- Requirement 01-01-04-09
  - Provide information on price of pharmaceutical
  - Sources: 1
- Requirement 01-01-04-10
  - Provide information on follow-up pharmaceuticals
  - Sources: 1
  - Description:
    - Inform user which pharmaceuticals might be useful after taking a pharmaceutical

        - A pharmaceutical might be only applicable for a certain amount of time so that an alternative should be taken afterwards
        - Another pharmaceutical might be useful to help with side effects
    - Requirement 01-01-04-12
        - Provide information on suitable dosage
        - Sources: 1
        - Description:
            - Inform the user about suitable dosage based on age or weight.
    - Requirement 01-01-04-13
        - Provide information on relation in which two pharmaceuticals should be taken
        - Sources: 1
        - Description:
            - Inform the user in which sequence two pharmaceuticals should be taken.
            - Example: If you take drug A and drug B take drug B two hours before you take drug A
- Requirement 01-01-05
    - Enable the user to go to a specific section of drug information
    - Sources: 7
    - Description:
        - In order to ease navigation in the drug information panel enable the user to directly navigate to a specific section of drug information

- Requirement 01-01-06
  - Offer functionality to go to top of drug information page below each section of information
  - Sources: 1
  - Description:
    - Enable the user to navigate directly to the top of the drug information page.
- Requirement 01-01-07
  - Do not display more information than the user needs
  - Sources: 9
  - Description:
    - Enable the user to configure which information he is interested in and hide information he is not interested in.
  - Requirement 01-01-07-02
    - Allow the user to filter side effects by frequency of occurrence
    - Sources: 4
    - Description:
      - Example: Display only side effects that occur often or very often
      - Example: Enable users to hide side effects which occur seldom
  - Requirement 01-01-07-03
    - Provide functionality to select all/no information to be displayed
    - Sources: 1
    - Description:
      - Manually enabling/disabling all information can be tedious
      - Offer information to directly enable/disable all displayed information

- Requirement 01-01-07-04
  - Enable user to turn off/on information concerning pregnancy
  - Sources: 2
  - Description:
    - Since pregnancy concerns only a few people it should be possible to hide this information
- Requirement 01-01-07-05
  - Filter displayed information by age
  - Sources: 1
  - Description:
    - Hide information irrelevant for age-groups
    - Example: Hide dosage information for children, if information for adults is requested.
- Requirement 01-01-07-06
  - Expand sections of information
  - Sources: 1
  - Description:
    - Enable users to expand/fold sections of information
    - Display only the headings of the sections and provide a toggle where the user can expand or fold the section of information

- Readability – 02
  - Requirement 01-02-01
    - Allow for adjustment of font size
    - Sources: 8
    - Description:
      - Enable the user to adjust the font size through functionality provided by the application

- Requirement 01-02-02
  - Clear structuring of text
  - Sources: 5
  - Description:
    - Improve readability through clear structuring/markup/layout of text
  - Requirement 01-02-02-01
    - Accentuate sub-headings in drug information tab
    - Sources: 4
    - Description:
      - In order to ease retrieval of information, sub-headings in drug information tab, like side effects or contraindications, should be clearly recognisable
  - Requirement 01-02-02-02
    - Transform text into bullet lists if possible.
    - Sources: 1
    - Description:
      - Instead of lengthy text display information in bullet lists.
      - Example: Make a bullet point for every listed adverse drug reaction instead of listing them as continuous text.
  - Requirement 01-02-02-03
    - Group contraindications
    - Sources: 1
    - Description:
      - Example: One group for medical conditions and one group for hypersensitivities to active pharmaceutical agents

- Requirement 01-02-03
  - Ensure that text-fields have at least a length of twelve words of average length
  - Sources: 1
  - Description:
    - To improve readability ensure that lines of text are not too short.
- Requirement 01-02-04
  - Maintain readability by ensuring that columns of tables have enough margin
  - Sources: 2
  - Description:
    - Clearly delineate different objects in drug information interface with margins.
- Requirement 01-02-05
  - Do not use distracting fonts or backgrounds
  - Sources: 1
  - Description:
    - Do not use too colourful fonts or backgrounds
- Requirement 01-02-06
  - Provide functionality to turn off all control elements so that the user can focus on reading the information on pharmaceuticals
  - Sources: 1
  - Description:
    - Display only the central drug information tab so that the user is not distracted while reading the provided information
    - Hence, hide the sidebars

- Requirement 01-02-07
  - Replace text with graphics
  - Sources: 1
  - Description:
    - Provide graphics that convey the same information as the replaced text so that the user can easier process the provided information.

- Intuitive operation of application – 03
  - Confusion of users – 01
    - Requirement 01-03-01-01
      - Display only relevant GUI elements to keep the application simple
      - Sources: 3
      - Description:
        - Display only GUI elements if it makes sense to use them
        - Example: Do not display selection of details provided on pharmaceuticals if the user is not viewing information on pharmaceuticals
    - Requirement 01-03-01-02
      - Hide functionality that might confuse inexperienced users
      - Sources: 6
      - Description:
        - Do not confuse inexperienced users with too many GUI elements
        - Provide a simple view for inexperienced users as default and allow expert users to turn on further functionality

- Requirement 01-03-01-02-01
  - Allow experienced users to turn on hidden GUI elements
  - Sources: 1
- Requirement 01-03-01-02-02
  - Hide additional functionality for inexperienced users
  - Sources: 1
  - Description:
    - Do not initially enable inexperienced users to access additional functionality
    - Inform users that they can turn on access to additional functionality
- Requirement 01-03-01-02-03
  - Display only introductory information at application startup
  - Sources: 2
  - Description:
    - Provide only introductory information at application startup
    - No display of GUI elements that are not related to the introductory information
- Requirement 01-03-01-02-04
  - Guide inexperienced users through the application rather than offering them many different controls and settings
  - Sources: 4
  - Description:
    - Ask users for relevant input
    - Display outputs accordingly
    - Do not require much initiative from the user

- Requirement 01-03-01-03
  - Make the search field in panel 'Medikamenten-Auswahl' clearly noticeable
  - Sources: 7
  - Description:
    - During requirement elicitation interviews many interviewees did not realise that they could enter search terms in the search field
    - The GUI should be designed more clearly so that users immediately realise that they can use the search field to initiate a search for pharmaceuticals
- Requirement 01-03-01-04
  - Use clear and meaningful names for GUI elements
  - Sources: 2
- Requirement 01-03-01-05
  - Do not offer different functionality under the same name
  - Sources: 1
  - Description:
    - Do not confuse the user by providing multiple GUI elements with same name but different associated functionality

◆ Instructions – 02

- Requirement 01-03-02-01
  - Easy locating and accessing of help
  - Sources: 1
  - Description:
    - Provide a link to the help menu that is positioned in such way that the user directly realises how he can access the help menu

- Requirement 01-03-02-02
  - Tooltips should not obstruct the user
  - Sources: 1
  - Description:
    - Avoid situation where the user is obstructed by tooltips that cover something
    - Example: Avoid situations where the user clicks on the tooltip instead of a button he wanted to click on
    - Example: Avoid situations where the user cannot read something because a tooltip floats above it.
- Requirement 01-03-02-03
  - Explain detail level selection
  - Sources: 1
  - Description:
    - Explain to the user how the detail level selection works and what is its intended purpose
- Requirement 01-03-02-04
  - Explain functioning of details panel
  - Sources: 3
  - Description:
    - Explain to the user how the details panel work and what is its intended purpose

◆ Requirement 01-03-03

- Display close buttons/icons for everything that can be closed
- Sources: 5
- Description:
  - Indicate where the user can close a GUI element

- Tabs – 04
  - Requirement 01-03-04-01
    - Intuitive functionality for creation of new tabs
    - Sources: 7
    - Description:
      - During requirement elicitation many interviewees did not understand what is meant with tab and how the creation of tabs work
      - Find a way that users can still view pharmaceuticals in different tabs but intuitively grasp how the creation and handling of tabs work
  - Requirement 01-03-04-02
    - Manual creation of new tabs
    - Sources: 1
    - Description:
      - Provide functionality to manually create a new empty tab
- Enabling/Disabling Panels – 05
  - Requirement 01-03-05-01
    - Avoid accidental disabling of GUI elements
    - Sources: 2
    - Description:
      - Design GUI in such a way that users cannot turn off GUI elements by accident
      - At least prevent that users do not know how to turn them back on

- Requirement 01-03-05-02
  - Explain how to disable/enable panels
  - Sources: 1
  - Description:
    - Provide explanations how users can enable/disable single panels.
- Requirement 01-03-05-03
  - Instead of check boxes in the header, provide an icon that minimises panels and displays them as small icons on the side of the application
  - Sources: 1

◆ Homepage – 06

- Requirement 01-03-06-01
  - Display introductory information at application startup
  - Sources: 6
  - Description:
    - Inform the user about the purpose of the application
    - Give short usage instructions
- Requirement 01-03-06-02
  - Display start page if all other tabs are closed
  - Sources: 3
  - Description:
    - Never display no tab. Display start page with instructions, if all other tabs are closed.
- Requirement 01-03-06-03
  - Ask users for general settings on start page
  - Sources: 1

- Description:
  - Let the user perform basic configuration of the application on the start page
  - Example: Setting for normal or advanced view
- Requirement 01-03-06-04
  - Provide a button to get back to home page and call it 'Startseite'
  - Sources: 1

- Accessing information on pharmaceuticals – 04
  - Requirement 01-04-01
    - Easy access to desired information
    - Sources: 3
    - Description:
      - Do not obstruct the user with unnecessary typing or clicking
      - Reduce tipping effort by offering predetermined options.
  - Requirement 01-04-02
    - Display more search characteristics in panel 'Medikamenten-Auswahl' instead of listbox
    - Sources: 1
    - Description:
      - Remove the listbox from panel 'Medikamenten-Auswahl'
      - Use the freed space for more search characteristics

- Comprehensibility – 05
  - Requirement 01-05-01
    - Use common wording for explanatory text
    - Sources: 4

- Description:
  - Use only words in explanatory text that a common user would understand
  - Example: Avoid computer slang words: Header → Kopfzeile

◆ Requirement 01-05-02

- Avoid abbreviations in text
- Sources: 8
- Description:
  - Replace abbreviations if possible.

◆ Requirement 01-05-03

- Explain technical terms
- Sources: 9
- Description:
  - Provide somehow explanations for technical terms
  - Explanations should be provided intuitively and simply
  - If possible it should not require opening a new window
  - Explanations should not introduce new technical terms

◆ Requirement 01-05-04

- Replace mathematical notations
- Sources: 1
- Description:
  - Example: Replace '>' with 'größer als' (greater than)

◆ Requirement 01-05-05

- Provide links to further information
- Sources: 1

- Description:
  - Example: Link to sources of information that provide more detailed information.

- Requirement 01-07
  - Open links to further information in a new window
  - Sources: 1

- Functionality – 02
  - Linking to pharmaceuticals – 01
    - Similar pharmaceuticals – 01
      - Requirement 02-01-01-01
        - Display pharmaceuticals with same active pharmaceutical agent as the currently displayed pharmaceutical
        - Sources: 4
        - Description:
          - Provide functionality to list pharmaceuticals that have the same active pharmaceutical agent as the currently displayed pharmaceuticals
      - Requirement 02-01-01-02
        - Display of pharmaceuticals with same application as currently displayed pharmaceutical
        - Sources: 3
        - Description:
          - Provide functionality to list pharmaceuticals that have the same application as the currently displayed pharmaceutical.
      - Requirement 02-01-01-03
        - Display further pharmaceuticals in product group of current pharmaceutical
        - Sources: 1

        - Description:
            - Example: When viewing Aspirin N 300, list all available pharmaceuticals in product group Aspirin

- Search for pharmaceuticals – 02
    - Search parameters – 01
        - Requirement 02-02-01-01
            - Search for pharmaceuticals by field of application
            - Sources: 8
        - Requirement 02-02-01-02
            - Search for pharmaceuticals by name
            - Sources: 10
        - Requirement 02-02-01-03
            - Search for pharmaceuticals by PZN
            - Sources: 2
        - Requirement 02-02-01-04
            - Search for pharmaceuticals by active pharmaceutical agent
            - Sources: 4
        - Requirement 02-02-01-05
            - Adapt to minor typos in search string
            - Sources: 1
            - Description:
                - A search should also find results that are typographically close to the input parameters.
                - Example: A search for 'Asisrin' should still find 'Aspirin'
        - Requirement 02-02-01-06
            - Search for pharmaceuticals by pharmaceutical company/manufacturer
            - Sources: 3

- Requirement 02-02-01-07
  - Use combinations of search parameters
  - Sources: 1
  - Description:
    - Allow specify a search request that consists of multiple parameters
- Requirement 02-02-01-08
  - Search for synonyms
  - Sources: 1
  - Description:
    - Entered search parameters might have synonyms. The search should consider the synonyms as well.
- Requirement 02-02-01-09
  - Search for pharmaceuticals by general name
  - Sources: 1
  - Description:
    - Example: Search for 'headache tablet' to find tablets that help with headaches
- Requirement 02-02-01-10
  - Order advanced search by relevance for user
  - Sources: 1
  - Description:
    - Order the input fields for search parameters in advanced search by relevance for the user.
    - Example: A common user is more likely to search for pharmaceuticals by name than by ATC-code

- Requirement 02-02-01-11
  - Search for matches in whole database
  - Sources: 1
  - Description:
    - Offer a search parameter that tries to find matches in all fields of the database
    - Example: Do not just search in names of pharmaceuticals.

◆ Search filter – 02

- Requirement 02-02-02-01
  - Filter search by requirement of prescription
  - Sources: 1
  - Description:
    - Specify whether pharmaceuticals that require/do not require a prescription should be listed
- Requirement 02-02-02-03
  - Filter search results by dosage form
  - Sources: 2
- Requirement 02-02-02-04
  - Filter search results by keywords mentioned in contraindications
  - Sources: 3
  - Description:
    - Do not display pharmaceuticals as search results that mention a specified keyword in category contraindications
    - Example: Do not display pharmaceuticals where 'liver' is mentioned in category contraindications.
- Requirement 02-02-02-05
  - Filter search results by allergies
  - Sources: 3

- Description:
  - Example: Do not display pharmaceuticals with a certain ingredient

- Requirement 02-02-02-06
  - Filter search results by age group
  - Sources: 2
  - Description:
    - Display only pharmaceuticals as search results that are suitable for a specific age group
    - Example: Display only pharmaceuticals suitable for children.
- Requirement 02-02-02-07
  - Provide mask for search filters before searching
  - Sources: 1

◆ Selection of input – 03

- Requirement 02-02-03-01
  - Find specific pharmaceuticals without knowing the exact name.
  - Sources: 4
  - Requirement 02-02-03-01-01
    - Provide list to select search parameters so that one can select the most common synonym
    - Sources: 1
- Requirement 02-02-03-02
  - Filter entries displayed in a listbox by an entered string
  - Sources: 2
  - Description:
    - Example: Enter initial letters of name of pharmaceuticals and adopt the list of matching pharmaceuticals with each newly entered letter.

- Requirement 02-02-03-03
  - Provide functionality to open information on pharmaceuticals that are being checked for adverse drug reactions
  - Sources: 1
  - Description:
    - Let the user open the drug information tab for pharmaceuticals he added to a check for adverse drug reactions
    - Example: Open a new tab with the drug information for a pharmceutical if the user clicks on its name in the menu for a adverse drug selection check

◆ Requirement 02-02-04
- Optionally initiate search by pressing enter in search field
- Sources: 3
- Description:
  - Provide a button to start search, but start search as well if a user presses enter in a search field

◆ Search results – 05
- Requirement 02-02-05-01
  - Sort search results by prices
  - Sources: 1
  - Description:
    - Order search results by price
    - Lowest price on top
    - If a pharmaceutical does not require a prescription patients probably want the cheapest pharmaceutical
- Requirement 02-02-05-02
  - Display alternative pharmaceuticals directly in search results
  - Sources: 1

- Description:
  - List pharmaceuticals that have same application or the same compounds in search results
  - Example: When search for Aspirin list other headache tablets as well.

- Additional functionality – 03
  - Description:
    - The core functionality of the application is the patient-friendly provision of information provided in patient information leaflets
    - The category additional functionality is used for functionality beyond the core functionality
  - Aggregation – 01
    - Requirement 02-03-01-01
      - Check a set of pharmaceuticals for adverse drug reactions
      - Sources: 10
      - Description:
        - Let the user specify a set of pharmaceuticals
        - Check the specified pharmaceuticals for adverse drug reactions with each other
    - Requirement 02-03-01-02
      - List all side effects of a set of pharmaceuticals
      - Sources: 5
      - Description:
        - User specifies a number of pharmaceuticals and all associated side effects are listed

- Requirement 02-03-01-03
  - Display a warning message if a pharmaceutical currently being viewed has an adverse drug reaction with a previously viewed pharmaceutical
  - Sources: 1
- Comparison of pharmaceuticals – 04
  - Requirement 02-03-01-04-01
    - Offer price comparison of similar pharmaceuticals
    - Sources: 4
    - Description:
      - Let the user see the prices of similar pharmaceuticals
      - Enable the user to identify the cheapest alternative.
  - Requirement 02-03-01-04-02
    - Compare similar pharmaceuticals
    - Sources: 5
    - Description:
      - Let the user select two pharmaceuticals for comparison
- Requirement 02-03-01-05
  - Provide general information on classes of pharmaceuticals
  - Sources: 1
  - Description:
    - Provide information common to most variants in a class of pharmaceuticals
    - Example: Specify common side effects of headache tablets
- Requirement 02-03-01-06
  - Check a set of pharmaceuticals for similar application
  - Sources: 1

- Description:
  - Inform the user if he takes two pharmaceuticals with same application
  - Let the user specify the pharmaceuticals he is currently taking
  - Inform the user if he is taking pharmaceuticals that are possibly redundant

- Requirement 02-03-01-07
  - Provide a list of pharmaceuticals that should not be taken under a specific condition
  - Sources: 1
  - Description:
    - Let the user specify some conditions
    - List all pharmaceuticals that should not be take under this condition
    - Example: List all pharmaceuticals that should not be taken during pregnancy

- Requirement 02-03-01-08
  - Check a set of pharmaceuticals for compatibility with allergies
  - Sources: 2
  - Description:
    - Let the user specify a set of pharmaceuticals and a set of allergies
    - Check whether the pharmaceuticals are compatible with those allergies

◆ Requirement 02-03-02

- Support identification of pharmaceuticals
- Sources: 5

- Description:
  - Enable the user to identify an unknown pharmaceutical
  - Example: Provide images of packages, pill, and blister so that the user can check whether the pharmaceutical he wants to identify looks like the one in the images

◆ Out of scope of thesis – 03

- Requirement 02-03-03-01
  - Link to pharmacies that sell the pharmaceutical
  - Sources: 5
  - Description:
    - Provide links to pharmacies where the user can buy the currently viewed pharmceutical
- Requirement 02-03-03-02
  - Provide a list of pharmacies that are open and in the vicinity of the user
  - Sources: 1
  - Description:
    - Enable the user to find pharmacies in his vicinity where he can purchase a desired pharmaceutical
- Requirement 02-03-03-03
  - Provide information on homespun remedies for common illnesses.
  - Sources: 1
  - Description:
    - Example: Inform users to drink chicken soup if they come down with a cold.
- Requirement 02-03-03-04
  - Search for suitable medical practitioners
  - Sources: 2

- Description:
  - Let the user specify an illness
  - Provide a list of suitable medical practitioners
- Requirement 02-03-03-05
  - Provide a dedicated interface for mobile phones
  - Sources: 2
- Requirement 02-03-03-06
  - Make appointments with specific medical professionals
  - Sources: 1
  - Description:
    - Interface with medical practitioners so that users can directly make appointments with medical practitioners
- Requirement 02-03-03-07
  - Provide information on current widespread diseases.
  - Sources: 1
  - Description:
    - Example: Inform user about a current outbreak of flu
- Requirement 02-03-03-08
  - Provide pharmaceutical news to attract users.
  - Sources: 1
- Requirement 02-03-03-09
  - Integration with German healthcare telematics infrastructure
  - Sources: 1

◆ Requirement 02-03-04

- Provide functionality to go back to previously viewed pharmaceuticals
- Sources: 1

- Description:
  - Make the buttons for 'Go back one page'/'Go forward one page', which are provided by browsers, work

◆ Requirement 02-03-05

- Provide functionality to print the selected information of currently displayed pharmaceutical
- Sources: 4
- Description:
  - Provide some functionality to enable the user to print information on currently selected pharmaceutical
  - Print just the information the user has currently selected
  - Do not print the GUI elements

◆ Requirement 02-03-06

- Enable users to create a list of pharmaceuticals they are taking and to print the list
- Sources: 1
- Description:
  - Hospitals or retirement homes require often a list of the pharmaceuticals one is taking
  - Provide some functionality to compile such a list

● Contact to Website Owner/Operating Authority – 04

◆ Requirement 02-04-01

- Inform users how the owner/operating authority of the web application can be contacted
- Sources: 4

- Description:
  - Allow users to ask questions concerning operation or other aspects of the application
  - Example: Provide a menu where the user can write an email, specify his email address and send the email

- User Accounts – 05
  - Requirement 02-05-01
    - Offer possibility to store user settings
    - Sources: 1
    - Description:
      - Let users save their current configurations so that they can easy view the application with a familiar setting
  - Requirement 02-05-02
    - Provide experiences of other patients with pharmaceuticals or with the application
    - Sources: 4
    - Description:
      - Provide some sort of forum or comment section where users can communicate with each other
      - Could be useful to get questions answered or read experiences of other users with a pharmaceutical
  - Requirement 02-05-03
    - Personal pharmaceutical management
    - Sources: 1
    - Description:
      - Enable users to specify the pharmaceuticals they are taking
      - Store the pharmaceuticals
      - Check whether a new pharmaceutical has adverse drug reactions with current pharmaceuticals

- Automatically inform the user on adverse drug reactions of specified pharmaceuticals
- Specify allergies and medical conditions to optimise searches for pharmaceuticals

- Requirement 02-05-03-01
  - Remind users to take pharmaceuticals
  - Sources: 1
  - Description:
    - Example: Send users an email to remind them to take a pharmceutical

- Requirement 02-05-03-02
  - Inform users to reorder pharmaceuticals
  - Sources: 1
  - Description:
    - User specifies the time he starts using a pharmaceutical
    - Some time before the pharmaceutical supply should run out, the application reminds the user to reorder

■ Software System attributes – 03

● Reliability – 01

◆ Provided Information – 01

- Requirement 03-01-01-01
  - Inform users on last update of provided information
  - Sources: 3
  - Description:
    - Display date of last update next to sections of provided information on pharmaceuticals
    - Date of last update can be always displayed in drug information tab because it does not require much space and provides possibly useful information

- Requirement 03-01-01-02
  - Enable users to report errors in provided information
  - Sources: 3
  - Description:
    - Functionality that should increase the reliability of the application
    - Provide users with a way to report wrong application provided by the application, if they discover such information
  - Requirement 03-01-01-02-01
    - Inform users on processing state of submitted notifications concerning wrong information.
    - Sources: 1
    - Description:
      - Show users that reports of wrong information, which is provided by the application, are processed
      - Provide some functionality where users can access the status of wrong information reports
      - Provide some functionality that captures the status of wrong information reports
- Requirement 03-01-01-03
  - Get a reliable company/association to host the application to increase perceived reliability
  - Sources: 1
  - Description:
    - Increase perceived reliability of the application through operation by company/association that is perceived as reliable
- Requirement 03-01-01-04
  - Let an independent entity check the provided information
  - Sources: 2

  - Description:
    - Increase perceived reliability of the application through validation by a independent entity
    - Example: A group of physicians/pharmacists
- Requirement 03-01-01-05
  - Inform the user about arrangements made to ensure reliability of provided information
  - Sources: 1
  - Description:
    - Increase perceived functionality of the application by informing the user about arrangements made to increase reliability
- Requirement 03-01-01-06
  - Keep provided information up to date
  - Sources: 3
  - Description:
    - Increase perceived reliability by providing information that is up to date
- Requirement 03-01-01-07
  - Describe sources of information
  - Sources: 3
  - Description:
    - Functionality to increase perceived reliability
    - Describe the sources from which the information provided on pharmaceuticals was taken so that users can decide for themselves whether they trust the provided information or not
- Requirement 03-01-01-08
  - Link to information on pharmaceutical provided by manufacturer
  - Sources: 1

- Description:
  - Provide links to information that the manufacturer provides on a pharmaceutical
  - Do this to provide an easy way for the user to validate provided information

◆ Data collection – 02

- Requirement 03-01-02-01
  - Do not collect irrelevant data that could identify users.
  - Sources: 1
  - Description:
    - Ensure anonymity of users by avoiding to collect irrelevant data
    - Example: Do map connected IPs to information requests
- Requirement 03-01-02-02
  - Inform users that they will not be profiled and that collected data will not be applied beyond the scope of the web application
  - Sources: 5
  - Description:
    - Increase perceived reliability by telling the users that they will not be profiled when using the application

◆ Requirement 03-01-03

- Provide functionality where users can give feedback/rate the application
- Sources: 1
- Description:
  - Let users report their experiences with and attitudes toward the application
  - This way users can access the reliability of the application with comments of other users

- Performance – 02
  - Requirement 03-02-01
    - Display 'loading notification' if loading takes longer than 0.5 seconds
    - Sources: 1
    - Description:
      - Inform users that the application is working on their requests if responses take some time
  - Requirement 03-02-02
    - Response times should be less than 5 seconds
    - Sources: 2
    - Description:
      - Do not annoy users with too slow response times
- Legal aspects – 03
  - Requirement 03-03-01
    - Provide information on author and operating authority ('Impressum')
    - Sources: 3
    - Description:
      - In Germany websites have to provide an 'Impressum'
  - Requirement 03-03-02
    - Tell users to contact physician in case of uncertainties
    - Sources: 3
    - Description:
      - Tell users that they should contact a physician in case of uncertainty
      - Try to make sure that self medication does not go wrong

- Visual Attractiveness – 04
  - Description:
    - Non-functional aspects that are intended to make the application more attractive for the user
  - Requirement 03-04-01
    - More attractive design of header.
    - Sources: 1
    - Description:
      - Design the header more attractive (Not only text in the header)
  - Requirement 03-04-02
    - Make GUI design more diversified/attractive/appealing
    - Sources: 4
    - Description:
      - Increase attractiveness of application with a diversified design
      - Do not provide a black/white experience
  - Requirement 03-04-03
    - Provide symbols for the categories of information
    - Sources: 1
    - Description:
      - Diversify the GUI and ease identification of sections of information with meaningful icons for the sections of information
  - Requirement 03-04-04
    - Provide different themes for application so that users can change to a visual design they like
    - Sources: 1
    - Description:
      - Let the user configure the application in a way they like by offering different colour themes

Complete and Final Database Model

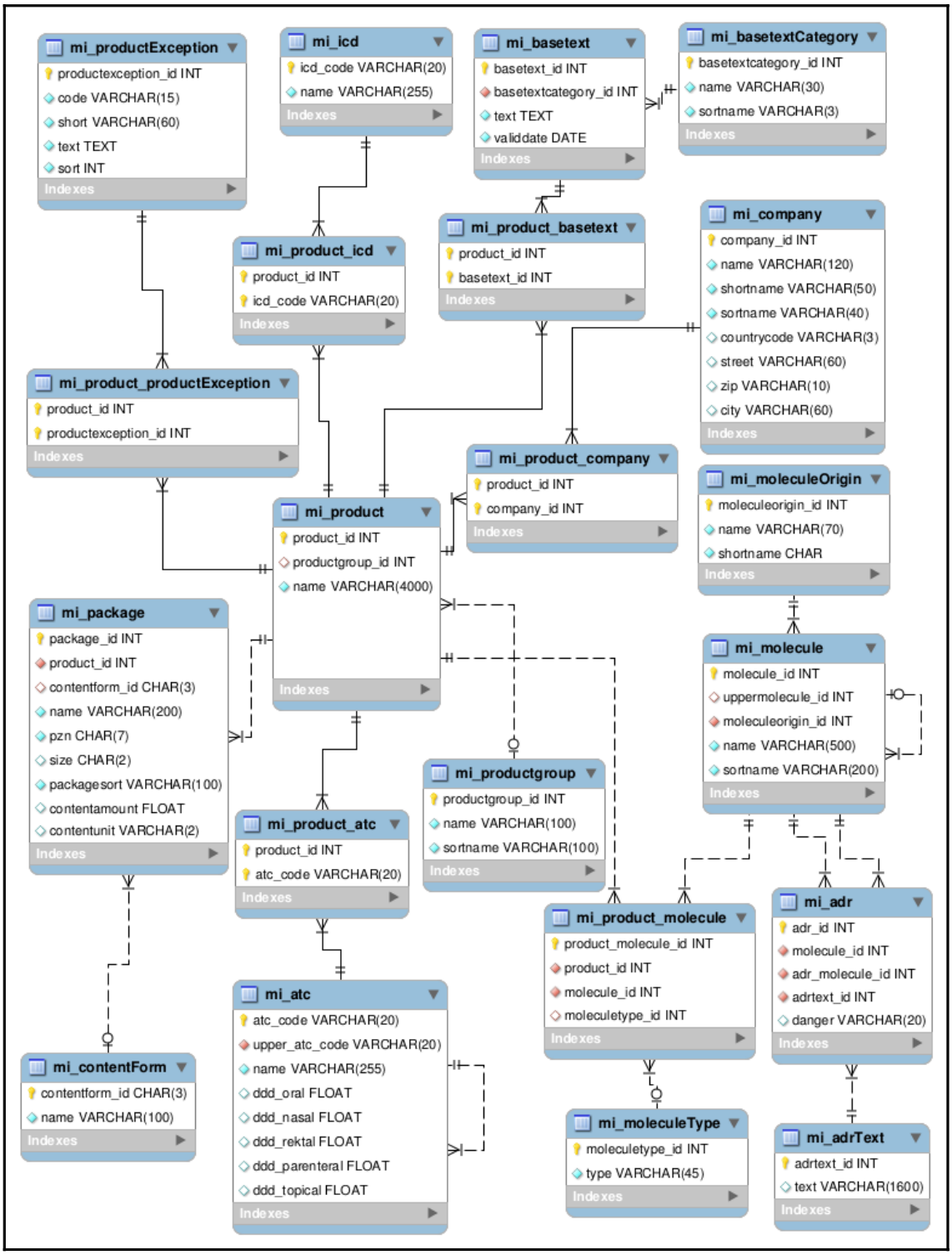

Fig. 7-2: Complete and final entity relationship diagram of database model.

Installation Instructions

1.) Web Access

If you do not want to install the web application yourselves, you can access the application on the web server we set up with the following URLs:

1. GUI Prototype: 'http://alturl.com/j6t2e' or 'http://dehling.dyndns-ip.com:8080/medinfoGuiPrototype/'
2. Final Version: 'http://alturl.com/aszda' or 'http://dehling.dyndns-ip.com:8080/medinfo/'

2.) Installation

If the application runs to slow on the web server (more than 10 year old linux box and just a home DSL connection) or you want to install the application for some other reasons, follow the following instructions:

1. In order to install the GUI prototype you need to preform the following tasks:
    1. Install Tomcat application server and deploy the war file
2. In order to install the final version you need to perform the following tasks
    1. Install and import MySQL Database
    2. Install Tomcat application server and deploy war file
3. The specified version numbers are the version we used. Newer versions should work as well.
4. If you perform the described steps correctly the GUI prototype/the final version should work on your computer. Otherwise, check what went wrong.

3.) Install and import MySQL Database

1. If you are installing the GUI prototype you can skip this step.
2. Get MySQL Community Server 5.1 ('http://dev.mysql.com/downloads/mysql/5.1.html#downloads')
   We chose the Windows x86 32-bit MSI Installer. Make sure that you select a version that fits to your operating system.

3. Run the setup wizard after download
    1. Your can use the custom setup to install MySQL to a directory of your choice
    2. Remember the folder where you installed MySQL
    3. You can also omit installation of the documentation and developer components
    4. Check the checkbox for the configuration wizard on the final screen of the installation wizard
4. Configuration Wizard
    1. Choose the standard configuration
    2. Check the checkbox to run MySQL as windows service
    3. Set a root password of your choice and remember it
    4. Set the configuration by clicking on execute.
5. Data Import
    1. Extract the zip file with the MySQL dump that can be found on the cd in 'app/db/medinfo.mysql.db.sql.zip' to the directory where you installed MySQL. The default location is 'C:\program files\mysql\mysql server 5.1'. (We will refer to this directory with $MYSQL_HOME from now on).
    2. Press 'Windows-Key' and 'R-Key' simultaneously, enter 'cmd', and press enter to open the command prompt.

        Use a user that has access to $MYSQL_HOME.
    3. Navigate to $MYSQL_HOME\bin ('PartitionLetter:' to change the partition, 'cd directoryName' to change the directory and 'cd ..' to go one directory level up)
        1. Hence, the command sequence 'F:', cd 'programme', 'cd mysql' would get you to the directory 'F:\programme\mysql'.

4. Run the command 'mysql.exe -h localhost -u root -pYOURSPECIFIEDPASSWORD < ..\medinfo.mysql.db.sql' to import the import the required data. If you did not extract the MySQL dump to $MYSQL_HOME adopt the last argument accordingly. Keep in mind that there is no space between -p and the password.
5. Enter 'services.msc' and restart the MySQL service so that the imported tables are known to the application. (As an alternative you can restart your PC)
6. Create the user that can read the data.
    1. Run the command 'mysql.exe -h localhost -u root -pYOURSPECIFIEDPASSWORD' in $MYSQL_HOME\bin
    2. Upon successful login type the commands listed below and press enter after each ';':
        1. use mysql;
        2. CREATE USER 'mysql_user'@'localhost' IDENTIFIED BY 'my5q1u532';
        3. GRANT SELECT ON medinfo.* TO 'mysql_user'@'localhost';
        4. quit;
    3. Avoid typing errors because the application needs this user to access the database

4.) Install Tomcat application server and deploy war file

This section describes how to deploy a .war-file to a Tomcat application server. Deploying a .war-file to Tomcat is necessary, if you want to run the web application locally. It should work with various versions of Tomcat as long they use a 32-bit Java Virtual Machine (JVM). Since we tested these instructions with Windows 7, a 32-bit CPU, a 32-bit JVM (JRE 1.6.0_29) and Tomcat-7.0.22, the instructions are written according to those characteristics. Feel free to adopt them to your setup. There should not be much change.

1. Make sure that you have installed Java Runtime Environment (JRE) 6 or later. If you need to install it, download it from 'www.oracle.org' and install it.
2. Make sure that you have set the environment variable JRE_HOME. If you need to set it go to 'Control Panel' → 'System' → 'Advanced system settings' → 'Advanced' → 'Environment Variables' and click on the button 'New...' in the section for system variables. Set the 'Variable name' to JRE_HOME and the 'Variable value' to 'Disk:\path\to\jre' (e.g. 'C:\Program Files\java\jre6'). As an alternative you can use JAVA_HOME instead of JRE_HOME.
3. Go to 'tomcat.apache.org' and download 'apache-tomcat-7.0.22-windows-x86.zip'.
4. Extract 'apache-tomcat-7.0.22-windows-x86.zip' to a folder of your choice. We will call the folder extracted from the .zip-file CATALINA_HOME.
5. Copy the .war-file to CATALINA_HOME\webapps

   The WAR files can be found on the CD in the directory 'app/war/' and are called medinfoGuiPrototype.war for the GUI prototype and medinfo.war for the final version.
6. To start Tomcat, run CATALINA_HOME\bin\startup.bat

   Do not close the Tomcat window as long as you need the server
7. Open your browser and access the web application with the URL 'http://localhost:8080/applicationName'. The application name is usually the file name of the .war-file without '.war'. The application name of 'medinfoGuiPrototype.war' would be 'medinfoGuiPrototype' and the application name of 'medinfo.war' is 'medinfo'.
8. To shutdown Tomcat, run CATALINA_HOME\shutdown.bat.
9. If another version of the application is already deployed to Tomcat, make sure to remove 'CATALINA_HOME\webapps\applicationName.war' and the associated folder 'CATALINA_HOME\webapps\applicationName' prior to copying the new .war-file (Step 5).

5.) Get the source code:

1. The source code is contained in the .WAR and .JAR files in the app/war/ folder of the cd. Just open the archives with an archiving software like WinZip and view the source files.
    1. GUI prototype: medinfoGuiPrototype.war
    2. Final version: medinfo.war
    3. Tool for DB creation: medinfoDBCreation.jar
2. Additionally, the sources are stored in the archive app/src/src.zip
    1. src.zip/dbCreation: Sources of tool that builds the MySQL database.
    2. src.zip/medinfoguiprototype: Sources of GUI prototype.
    3. src.zip/medinfo: Sources of final version.
3. As an alternative you can check out the sources from the svn repository in app/src/svn. The repository is called medinfo and requires no user accounts.