Master's Thesis

Utilizing New Technologies for the Improvement of Problem Based Learning in Medical Education

Georg Kitz
g.kitz@student.tugraz.at

Graz University of Technology

Institute for Information Systems and Computer Media
Advisor: Asoc. Prof PHD Martin Ebner

Graz, November 2013

Masterarbeit

Verwendung neuer Technologien um Problembasiertes Lernen im Medizinischen Bereich zu verbessern

Georg Kitz
g.kitz@student.tugraz.at

Technische Universität Graz

Institut für Informationssysteme und Computer Medien
Advisor: Univ.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner

Graz, November 2013

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

”Computers themselves, and software yet to be developed,
will revolutionize the way we learn.”
Steve Jobs

Acknowledgement

Finishing this Master's thesis and finishing University wouldn't have been possible with the help of several people.

First of all I want to thank Martin Ebner for giving me the opportunity to write this thesis and for helping me with his valuable supervision as well as providing me with advice, even during difficult phases of this project.

My beloved fiancé Judith, who was a great support while writing this thesis (and did not banish me when I was whining). No need to mention that she was motivating, supportive, patient and quite thorough when she reviewed my writings.

My mum and dad who gave me the possibility to study and supported me all along the way. They believed in me even when I was giving them a hard time. Special thanks to my dad, who was and still is my role model whilst trying to find my way outside academia.

In fact, this thesis wouldn't have been written without the constant grouchiness of 3 women. This *thank you* goes to my mum Dorothea, my fiancé Judith and my mother in law Tina, who consistently insisted that finishing University is something I will benefit from. I guess time will show… but if not for the sake of knowledge finishing my studies definitely taught me endurance and reinforced my decision that academia is just not for me.

Finally a special salut to Daniel Lichtenegger, Patrick Thonhauser and Richard Marktl who have been great companions during the time at University. It was really a pleasure to work with you on the numerous group-works we've encountered. As the saying goes: Work hard, play hard! I am glad I've made such bad-ass friends for life.

Georg Kitz, Wolfsberg, 2014

Abstract

In a time where medical education seems to be more complex than ever, facing horrendous student numbers and an increasing accessibility of information, the question of how to ensure the most efficient learning experience for medical students remains a vivid point of discussion. Active Learning and its subconcept of Problem Based Learning represent a promising candidate for future improvements in the area of medical education. The utilisation of the concept of Problem Based Learning, paired with the latest inventions in web and mobile technologies, finally provides the necessary media to build a possible solution, satisfying the needs of ”New Age Education”. This Master’s thesis discusses and describes the successful development of a novel reporting system, which focusses on the reliable and fast processing of patient case files during a student’s supervised period of compulsory hospital field work. This novel system enables an improved and fast way of supervisor/student interaction and communication, enables an uncomplicated data transfer between all parties and thus successfully demonstrates the implementation of state of the art technology to the field of medical education. Finally the system was tested by student volunteers using a method called ”thinking aloud”. In the frame of these tests smaller ”flaws” as well as ideas for further improvement have been identified and the huge potential of this novel system, as well as its practical applicability, has been highlighted by the testers and the merely positive feedback.

Keywords: Problem Based Learning, Active Learning, Web Technologies, Mobile Technologies, iOS

Kurzfassung

In einer Zeit in der die medizinische Ausbildung komplexer ist als je zuvor und in der Universitäten unbeschreiblich hohen Studierendenzahlen und der stetig wachsende Informationsvielfalt gegenüberstehen, bleibt die Frage wie man die effizienteste Lernerfahrung für Studierende sicherstellen kann, ein reger Diskussionpunkt. Aktives Lernen und das Subkonzept des Problembasierten Lernens repräsentieren einen vielversprechenden und modernen Ansatz um Lernen in der medizinischen Ausbildung zukünftig zu verbessern. Wenn man das Konzept des Problembasierten Lernens mit der modernsten Technologie im Bereich Web und Mobile kombiniert, kann man schnell das große Potential einer solchen Paarung als Lösungsansatz, welcher den Erfordernissen der sogenannten ”New Age Education” genügt, erkennen. Im Rahmen dieser Masterarbeit wurde ein neuartiges Kommunikations- und Reportsystem entwickelt, dessen Fokus auf der schnellen und zuverlässigen Verarbeitung von Patientenakten während des praktischen Jahres im Medizinstudium liegt. Das entwickelte Softwaresystem, welches die erfolgreiche Implementation moderner Technologien im Anwendungsfeld der medizinischen Ausbildung demonstriert, gewährleistet eine vereinfachte und beschleunigte Kommunikation bzw. Interaktion zwischen Student und Supervisor und ermöglicht einen raschen und unkomplizierten Datenaustausch zwischen den beteiligten Parteien. Schlussendlich wurde das System von Studenten mit Hilfe der ”Thinking Aloud” Methode getestet. Im Rahmen dieses Tests wurden kleinere Kritikpunkte und Verbesserungsvorschläge identifiziert und das große Potential dieses neuartigen Systems, sowie dessen Anwendbarkeit in der Praxis, wurde durch die Tester und das überwiegend positive Feedback unterstrichen.

Keywords: Problem Based Learning, Active Learning, Web Technologies, Mobile Technologies, iOS

Contents

List of Figures

Listings

List of Abbreviations

API	Application Programming Interface
CBL	Case Based Learning
CSS	Cascading Style Sheets
DOM	Document Object Model
DRY	Don't Repeat Yourself
GRAT	Group Readiness Assurance Test
IRAT	Integrated Readiness Assurance Test
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MVC	Model View Controller
PBL	Problem Based Learning
REST	REpresentational State Transfer
TBL	Team Based Learning
UML	Unified Model Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

1. Introduction

Ever since, learning plays a key role in adapting to new situations and in dealing with life in general. Our generation lives in an information driven world in which the use of new technologies influences our success and the gathering of quick information is a prerequisite for staying competitive compared to our peers. As a matter of fact, the technologies and information available to us is increasing steadily and so is the pressure of keeping abreast [Ebner, 2013]. In combination with a growing number of students, who eagerly want to learn new things, the field of education faces a quite challenging problem.

"How are students supposed to learn efficiently and make the most out of the information overload available to them?"

Another issue is the fact that learning is only one part of the problem. The other aspect is the efficient use of the provided information, in order to successfully apply the gained knowledge for solving a problem. Most students tend to learn for a specific exam in a very short time period, storing the necessary information in their short-term memory and forgetting it immediately after their examinations are done. This is also because they don't have enough time to transfer the gathered knowledge from their short to their long-term memory [Banikowski, 1999]. Although most teachers prepare their lectures at great expense, the thoroughly prepared material, which is later on learned by the students, is already forgotten after a short timespan. Researchers found out that after 90 days students forget around 90% of everything they've heard and learned during a lecture [Smilovitz, 1996].

This evokes the need for a different approach when dealing with a large number of students, the quickly changing information and the relation between the information and its practical use. A possible solution, which could tackle the mentioned issues, is the concept of Problem Based Learning (PBL), which is derived from the renowned concept of Active Learning.

The highly popular field of medical education, from which the concept of PBL originates from, is obviously no exception to these problems. Consequently,

this thesis is dedicated to answering the question of how a better implementation of PBL into the field of medical education can be achieved by employing new technologies e.g. web applications/services and mobile devices. The focus lies on ensuring an efficient processing of medical case files during a student's academic doctoral training, by introducing a novel reporting system. The developed system, which will be explained in the course of this thesis, comprises a web portal, useable by both, students and supervisors. To ensure the flexibility needed for students who are working with different patient cases in a hospital, the student's side of the web portal is additionally linked to a mobile device. This provides the students and supervisors with the flexibility to get the most out of the concept of Problem Based Learning and ensures a smooth and regular interaction between both parties.

2. Overview of Learning Methods

In general learning methods can be divided into two different learning styles; Active Learning and Passive Learning. Passive Learning is what happens in almost all classrooms nowadays. It's a presentation based learn style, where a teacher is giving a lecture and the pupils are expected to learn from what has been presented to them. This approach assumes that pupils learn just by listening to the teacher and that those students are an "empty" vessel which sits in class in order to be filled with knowledge [Herr, 2008].

Active learning on the other hand requires the student to actively take part in the learning process. As such the student gets involved in all the steps of the learning process and the teacher takes on the role of a guide rather than a presenter. The basic idea is to stimulate the students' interest and curiosity into learning about a topic by actively integrating them into the subject [Bonwell, 1991].

There are several different approaches to motivate students. Benware [Benware, 1984] conducted a study in which students were divided into two groups, learning about the same subject but using different learning approaches. The first group was using a passive learning approach. In this group the students were aware that they will be tested about the presented course syllabus at the end of the course. In contrast to the first approach, the second group was assigned with fulfilling active tasks. Their assignment was to prepare the same topic on their own and to teach other students about the topic. So one group was basically learning for a test while the other was learning in order to be able to pass on the newly gained knowledge to their classmates. In his study Benware found that the students who were assigned with teaching approached the subject in a completely different manner. They showed increased motivation and paid much more attention to detail when preparing the syllabus. Benware based this observation on the importance of a psychological component, which seems to play a crucial role in this behaviour as he thinks that students who prepare for teaching want to leave a good impression.

In literature [Larmer, 2013] there are several different models that are derived from the concept of Active Learning.

The models listed below represent some of the more persistent Active Learning models and thus are briefly discussed in more detail in the following sections.

- Case Based Learning
- Team Based Learning
- Problem Based Learning [1]

2.1. Case Based Learning

As suggested by the name ”Case Based Learning”, in this approach a teacher employs case studies to deliver knowledge to the students. Teachers provide their students with cases, including real-life cases or fictional cases, for discussion in class. The students then need to prepare and work through those case files on their own. This includes work in the classroom as well as at home. In Case Based Learning (CBL) it's crucial that the students are provided with clear instructions about their case files and the involved tasks and questions [Allchin, 2013]. They need to know whether the case file contains enough information for them to prepare themselves properly or whether they have to do research on their own to learn more about the topic. If they have to do their own research the supervisor has to provide them with information about where to find the necessary additional resources for their preparation. After they've individually prepared their case files and gathered enough knowledge about it, they get together in small groups to discuss their findings. At this stage the teacher needs to be present and active during all further steps, not only for guidance and for answering questions, but also to foster active discussions within the small groups as well as with the whole class [Srinivasan, 2007].

The steps a student has to take, when a teacher employs Case Based Learning in his/her class, can be summarised in three points:

- Exploring a problem by analysing the case and sorting relevant facts
- Developing conclusions about the topic

[1] Which is the model of main interest to this thesis

- Presenting their findings in front of their classroom and instructors

It's also important that the teachers familiarise themselves with this learning method before applying it in their classes. They have to be experienced with identifying and selecting suitable cases as well as with preparing questions, which lead the students in all the possible directions of a subject. This is necessary to explore the case from every possible direction and as a consequence to foster later discussions in the groups.

A problem one might encounter with this approach is the natural fear of students to ask questions. CBL builds its base on top of open discussions. Subsequently, if students fear to ask questions or to generate their own assumptions, CBL isn't going to work and all its benefits are negligible. Thus, teachers need to make clear that there are no *stupid* questions and that everyone will benefit from asking questions [Stanford, 1994].

The main aim of Case Based Learning is to highly engage students in the learning process and to encourage them to find out as much as possible about a certain topic by carrying out self-initiated research and discussing their findings with the rest of the class. Well structured questions provided by their supervisors should help them to consider aspects of the topic they haven't thought of before.

2.2. Team Based Learning

The idea of Team Based Learning (TBL) was discussed and formulated by Larry Michaelsen in 1970, during his time as a faculty member of the University of Oklahoma, and was mainly targeting the education of business students [Michaelsen, 2002].

Like other models based on Active Learning, TBL is centred around the active work of students, which is led by an instructor. This method however fosters group activity, where teams of five to a maximum of seven people are built. This method can also be applied to classes with a higher number of participants as you simply end up with a higher number of individual teams.

During the team building phase, the teacher should ensure a high diversity amongst the people within an individual team. The aim should be to bring together a group of people with a broad range of different experiences in one group. The maximum diversity in a team is important, as students with different backgrounds approach a certain problem in a way other students would never think of. Hence this will uncover new possibilities for the group and further lead to a better overall learning experience. However, it's important to sustain the teams' comparability during this process, which means that the teams have to be well balanced after this phase [Michaelsen, 2007]. Haidet conducted a study where he brought medical and nursing students together in groups to solve patient safety problems and found that this collaboration helped to build a better inter-professional teamwork which helped to understand both groups better and which led to better case solving results [Haidet, 2011].

Basically Team Based Learning is grouped in three different repeating phases which build the learning environment.

1. Pre-class Study

   In this prior learning phase students have to study the material they've received from their instructor in advance. The course material is provided by the teacher together with clear instruction about the scope of the issued material, whether the handouts cover all the aspects of the course or whether additional reading is necessary. In this regard it's quite similar to the CBL approach. (Section 2.1.)

2. Readiness Assurance

   This phase focuses on the knowledge obtained during phase one and can be further divided into four steps.
   The first step is to test the individual students about the knowledge they've gathered during phase one. Usually a multiple choice test is used in order to do so. This step is called Individual Readiness Assurance Test (IRAT). The purpose of this test is to focus on the internalisation of the material which was given to the student rather than trying to apply the knowledge

gathered from it. After every student of the team has completed this test, they sit down together and try to solve the test as a group. This second step is called Group Readiness Assurance Test (GRAT). It has to be pointed out that usually the group performs better at the test than the best student in the group individually. This can be simply explained by the greater pool of knowledge available within a group of people. The third step is used to give immediate feedback to the group. They check their individual results and try to work out everything they didn't know. They also get the chance to note down every open question which they want to discuss later. This is followed by the fourth and last step where they get feedback from their supervisor(s) and have the chance to ask and discuss their questions in order to close all their remaining knowledge gaps.

3. Application

   During this phase students are faced with problems of the problem domain which has been previously covered in the material they've got. Through the whole process they remain in the same team they've started with. The problems are the same for all teams and all the teams work to find a solution for smaller problems within the given problem domain. If every group has finished their work, they are told to present their findings in front of the other groups. After those presentations usually a discussion is started where teams have to defend their answers and conclusion in front of the other teams which don't necessarily agree with their work. The instructor takes on the role of a chairmen during this discussions.

A very important point of the Team Based Learning model is the individual evaluation of each member within a group. Students get feedback about their work, performance within the team and learn more about their skill set and contributions within a team.

2.3. Problem Based Learning

Problem Based Learning (PBL) is another model derived from the concept of Active Learning. PBL intends to promote a student's engagement in developing advanced problem solving abilities, by focusing on intentional learning and

decision making. [Lacuesta, 2009] [Grabinger, 2002] [Akinoglu, 2007]
Over the past 30 years Problem Based Learning has gained increasing popularity, as it seems to be one of the most promising candidates for ensuring an effective learning experience. The concept of PBL continues to obtain acceptance and gets widely adapted to various scientific fields. It's a constructive and instructive approach, with the learner in its focus. This model enables the students to perform research, make use of all different kinds of theoretical knowledge, previously obtained experiences and finally helps them to apply all the newly obtained skills to finding a possible solution for an ill-structured problem. This learning model aims to simulate real-life problems, and thus helps students to learn how to successfully tackle problems they might face during their future work positions and career. It also represents the core learning model applied within this thesis.

PBL, as it is generally known today, evolved from innovative health science curricula, introduced in North America over 30 years ago. Medical education, with its intensive pattern of basic science lectures, followed by an equally exhausting clinical teaching program, was rapidly becoming an ineffective and inhumane way to prepare students, given the explosion in medical information, new technologies and the rapidly changing demands of future practice.

The medical faculty at McMaster University in Canada introduced the tutorial process, not only as a specific instructional method but also as central part of their philosophy for structuring an entire curriculum to promote the idea of student-centred, multidisciplinary education and lifelong learning in professional practice [Feletti, 1998]. Barrows stated that a medical student has to learn a lot of facts, but learning all of those facts doesn't make the student an excellent doctor. The student might or might not remember the hard facts he/she has learned. The only thing a student really learns, is to quickly memorise a huge amount of information, regardless if he/she is able to use and apply it in real-life scenarios or not. Consequently, he wanted to change the way medical education is taught completely. Teaching in a traditional way didn't help students to understand the relation between the theoretical content and the actual context they were facing during practical clinical applications [Barrows, 1980] [Barrows, 1994] [Barrows, 1996].

The concept of PBL puts the student into the position of a problem solver rather than a strict learner. It is a method, which develops the student's problem solving strategies and knowledge gathering skills, as and when they are confronted with complex problems. The problem-solving aspect within this learning model engages students actively, which enhances their motivation to learn and carry out self-initiated research. Usually there are two main roles in education; the teacher and the student. In comparison to traditional learning styles these roles are completely altered when applying the model of Problem Based Learning. The teacher is no longer the person who tries to transport the knowledge to the learner by simply presenting the syllabus to the students, he/she is more a tutor, supervisor or moderator, which guides the students and leads them in the right direction when they're off track. He/She helps the students to identify the key subject of the problem, rather than being a sheer content expert. However, this doesn't mean that the teacher, in his role as tutor, becomes a simple observer of the students' actions. They still have an active role and have to provide the students with continuous feedback as soon as they come up with new findings or if they want to discuss certain things about a topic or problem. This is vital because the supervisor is still supposed to keep the students on track [Albanes, 2000].

There are several aspects, which actually define what PBL is:

- Students are responsible for their own learning progress

  As mentioned before, it's a learner-centred approach. Students engage with the problem and are responsible for solving it on their own. This results in an increased motivation for finding a solution to the given problem. It must be accepted by the supervisors that students are allowed to consult them in all areas they need to gain more information about in order to be able to solve the problem or if they're stuck and need some extra guidance.

- The problems must be ill-structured

  The problems should be as close as possible to real-world problems and therefore they have to be ill-structured. A well-structured problem doesn't illustrate reality. In fact, it's most likely that a student will never face

a well-structured problem in real-life. Moreover, students are less engaged when they have to solve well-structured problems.

- Collaboration is important

  During and after their education students have to deal with other people. It's not possible to circumvent group work and that's why it's crucial to develop group-working skills. Information needs to be shared within the group. It's also possible that tutors question all group members about certain information, to check whether the questioned topic is evenly distributed within a group or not.

- A final analysis must be made

  After solving a problem students should do a final analysis. This will reflect what they've actually learned during the process of solving a problem. This helps them to understand the new information they've gathered during PBL, reflects what they've finally achieved and how they've performed within this complex environment [Barrows, 1988].

- The assessments must measure the progress

  The final assessment from a supervisor must cover the detailed progress of problem solving. Learners need to be assessed on a regular basis, which ensures that they really benefit from PBL and that they actually cover the given topic thoroughly.

3. The problem domain

The last chapter dealt with learning methods and especially with Problem Based Learning, which is the method of interest to this project. This chapter will now cover the next step of this thesis and shed some light on the actual problem domain of the conducted work. Since this Master's thesis is carried out in cooperation with the Medical University of Graz (Austria) the problem domain is especially tailored to their specific needs. When students attend their 5th year of medical training at the Medical University of Graz (Austria) they are sent to hospitals to work with real patients and on real medical cases. There are several different fields they could be assigned with and end up working in. This includes e.g. surgery, geriatrics and emergency medicine. What they are supposed to learn during this practical year is how to solve and process medical cases on their own. They've to deal with those case files just like they're the leading doctor responsible for the case. Nonetheless, as they aren't fully trained doctors yet, they still need some sort of tutor or supervisor, which overlooks them and guides them in the right direction during the processing of their patients' case files. This is supposed to provide them with the necessary skills to approach medical problems on their own in their future careers as specialised doctors or surgeons.

The current approach of processing case files, which the Medical University of Graz makes use of, is explained in the following:

- A supervisor allocates a case to a student, who then starts an investigation
- The students are responsible for making the right diagnosis
- They write down all of their findings in a standardised paper report form (medical case file).
- They have to hand in their reports once per month and the supervisor checks them and provides the students with feedback.

Recalling the previous Section 2.3., which defined what Problem Based Learning is, it becomes clear that the approach and the curriculum of the Medical University of Graz is already based on the concept of Project Based Learning. Nonetheless, the current approach has multiple pitfalls, which literally scream for an optimisation.

Students as well as supervisors face several problems, which are explained in the following.

First the issues students are facing are discussed.

- Keeping efficient and comprehensive records of their findings

  Student surveys have shown that, although students are required to have their report forms with them at any time and fill them out whilst working with the patients, most of them don't take notes during their sessions with the patients or loose their notes until they actually manage to write the reports for their supervisor(s). In the end, this leads to incomplete and in-comprehensive reports, where things don't fit together, crucial information lacks or different case files get mixed up.

- The time span that it takes for obtaining feedback

  One common major problem is the time that it takes from the point where students experience problems during the processing of a patient's case file, to the point where a supervisor is able to give them feedback. Often this process of identifying a problem and gaining the required supervision corresponds to several weeks, because every time students request help they have to set up a meeting, hand in and present the case file. This often leads to the problem that an enquiry made by a student gets lost in the amount of other requests, which are still awaiting processing.

On the other side there are the tutors/supervisors, who, besides being professionally accountable for fulfilling their everyday job duties in the hospital, are also responsible for numerous students, all of them in need of constant supervision.

- Providing fast feedback and advice for several supervisees

  Every month, at a previously set deadline, the supervisors receive all the reports of their students. Each student worked on multiple cases and usually a supervisor has a group of 20-30 people. This makes it difficult

to provide fast advice and feedback for all of them. Let's assume that each student has five cases to work on within a month. With an average student count of 25 people, this means a supervisor has to go through 125 cases in one week, including the provision of feedback and the final assessment of the students based on their reports. This is far from ideal, especially as it needs to be considered that a supervisor still has account for his/her every day duties in the hospital.

- Coordinating supervision between multiple supervisors

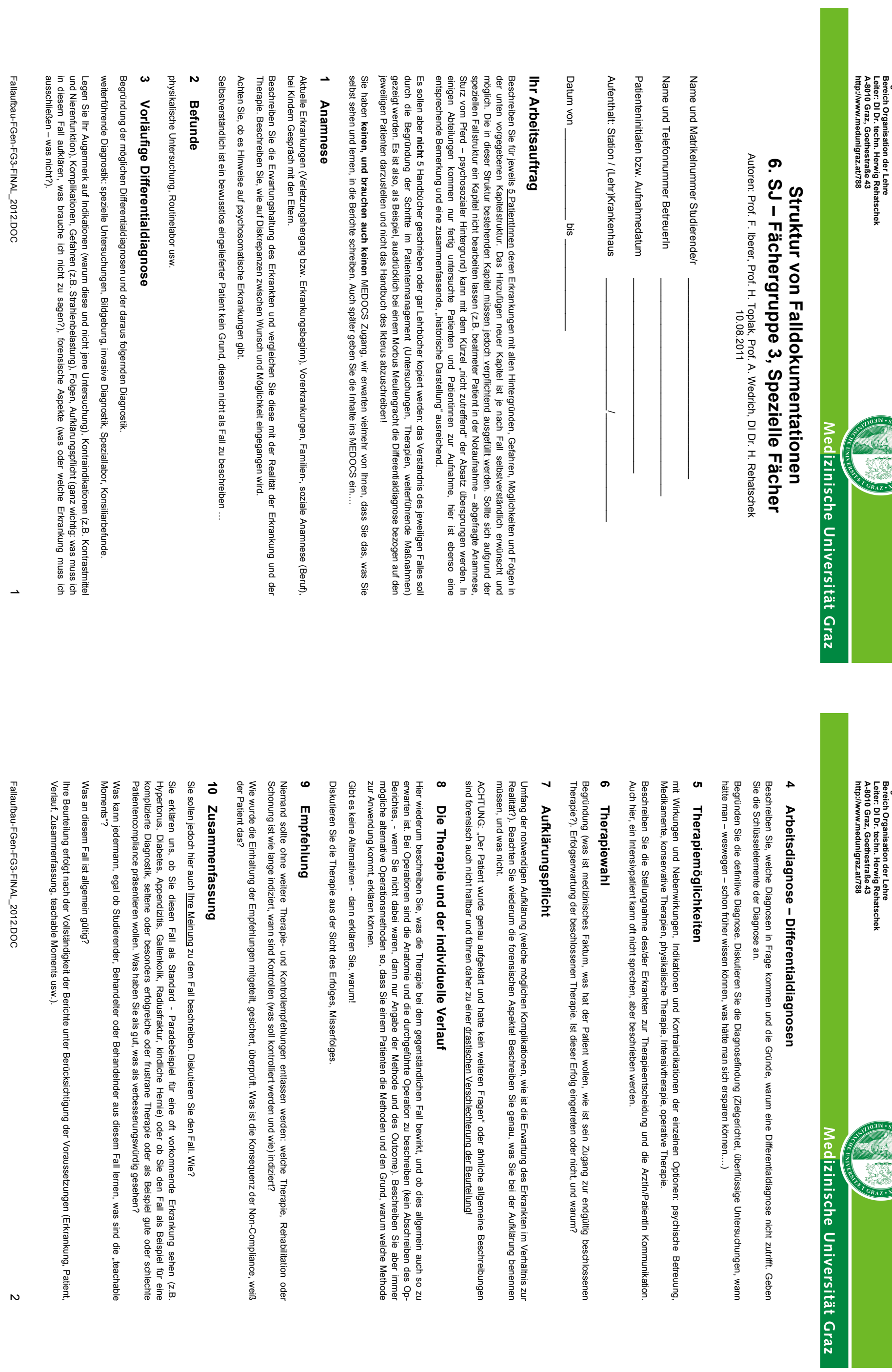
  Often there are up to three supervisors involved in the revision of a student's report. This supervisors are assigned hierarchically. Accordingly they have hierarchic rights, meaning that one supervisor holds superior rights over another supervisor and the one in the higher position is always able to overrule and edit input provided by a colleague in an inferior position. The communication between supervisors in paper form is quite challenging as they always have to hand in their work to each other in order to satisfy the hierarchic order of supervision. This leads to longer processing times of the students' reports.

After analysing this findings, three things can be identified, which both groups would tremendously benefit of:

- Continuous reporting, where students write down their findings and send all problems they encounter to the supervisor as and when they occur.
- Continuous feedback, meaning supervisors are able to give feedback and add comments to their students' reports on a regular basis and guide them in the right direction if they are completely off the track.
- An improved communication and data exchange between all participating supervisors.

That's why those things need to be improved in order to obtain a better student/-supervisor interaction and consequently to achieve a better report quality, faster feedback and in the end a higher quality of doctors in our hospitals.

Figure 1 is an example of the before mentioned standardised paper form the student has to hand in. In the figure you can see the headlines which are basically the main points of interest of this form. Every headline targets a specific influence factor of the case. Students are meant to setup a report like this, using the headlines and fill in the blanks in between them. This is done either by filling in the details by hand or on the computer as a Word document.



Organisationseinheit für Studium und Lehre
Bereich Organisation der Lehre
Leiter: DI Dr. techn. Herwig Rehatschek
A-8010 Graz, Goethestraße 43
http://www.medunigraz.at/788

Medizinische Universität Graz

Struktur von Falldokumentationen
6. SJ – Fächergruppe 3, Spezielle Fächer

Autoren: Prof. F. Iberer, Prof. H. Toplak, Prof. A. Wedrich, DI Dr. H. Rehatschek
10.08.2011

Name und Matrikelnummer Studierende/r ______________________

Name und Telefonnummer BetreuerIn ______________________

Patienteninitialen bzw. Aufnahmedatum ______________________

Aufenthalt: Station / (Lehr)Krankenhaus ______________/____________

Datum von ____________ bis ____________

Ihr Arbeitsauftrag

Beschreiben Sie für jeweils 5 PatientInnen deren Erkrankungen mit allen Hintergründen, Gefahren, Möglichkeiten und Folgen in der unten vorgegebenen Kapitelstruktur. Das Hinzufügen neuer Kapitel ist je nach Fall selbstverständlich erwünscht und möglich. Die in dieser Struktur bestehenden Kapitel müssen jedoch verpflichtend ausgefüllt werden. Sollte sich aufgrund der speziellen Fallstruktur ein Kapitel nicht bearbeiten lassen (z.B. beatmeter Patient in der Notaufnahme – abgefragte Anamnese, Sturz vom Pferd – psychosozialer Hintergrund) kann mit dem Kürzel „nicht zutreffend" der Absatz übersprungen werden. In einigen Abteilungen kommen nur fertig untersuchte Patienten und Patientinnen zur Aufnahme, hier ist ebenso eine entsprechende Bemerkung und eine zusammenfassende, „historische Darstellung" ausreichend.

Es sollen aber **nicht** 5 Handbücher geschrieben oder gar Lehrbücher kopiert werden: das Verständnis des jeweiligen Falles soll durch die Begründung der Schritte im Patientenmanagement (Untersuchungen, Therapien, weiterführende Maßnahmen) gezeigt werden. Es ist also, als Beispiel, ausdrücklich bei einem Morbus Meulengracht die Differentialdiagnose bezogen auf den jeweiligen Patienten darzustellen und nicht das Handbuch des Ikterus abzuschreiben!

Sie haben **keinen, und brauchen auch keinen** MEDOCS Zugang, wir erwarten vielmehr von Ihnen, dass Sie das, was Sie selbst sehen und lernen, in die Berichte schreiben. Auch später geben Sie die Inhalte ins MEDOCS ein….

1 Anamnese

Aktuelle Erkrankungen (Verletzungshergang bzw. Erkrankungsbeginn), Vorerkrankungen, Familien-, soziale Anamnese (Beruf), bei Kindern Gespräch mit den Eltern.

Beschreiben Sie die Erwartungshaltung des Erkrankten und vergleichen Sie diese mit der Realität der Erkrankung und der Therapie. Beschreiben Sie, wie auf Diskrepanzen zwischen Wunsch und Möglichkeit eingegangen wird.

Achten Sie, ob es Hinweise auf psychosomatische Erkrankungen gibt.

Selbstverständlich ist ein bewusstlos eingelieferter Patient kein Grund, diesen nicht als Fall zu beschreiben …

2 Befunde

physikalische Untersuchung, Routinelabor usw.

3 Vorläufige Differentialdiagnose

Begründung der möglichen Differentialdiagnosen und der daraus folgernden Diagnostik.

weiterführende Diagnostik: spezielle Untersuchungen, Bildgebung, invasive Diagnostik, Speziallabor, Konsiliarbefunde.

Legen Sie Ihr Augenmerk auf Indikationen (warum diese und nicht jene Untersuchung), Kontraindikationen (z.B. Kontrastmittel und Nierenfunktion), Komplikationen, Gefahren (z.B. Strahlenbelastung), Folgen, Aufklärungspflicht (ganz wichtig: was muss ich in diesem Fall aufklären, was brauche ich nicht zu sagen?), forensische Aspekte (was oder welche Erkrankung muss ich ausschließen – was nicht?).

Fallaufbau-FGen-FG3-FINAL_2012.DOC 1

Organisationseinheit für Studium und Lehre
Bereich Organisation der Lehre
Leiter: DI Dr. techn. Herwig Rehatschek
A-8010 Graz, Goethestraße 43
http://www.medunigraz.at/788

Medizinische Universität Graz

4 Arbeitsdiagnose – Differentialdiagnosen

Beschreiben Sie, welche Diagnosen in Frage kommen und die Gründe, warum eine Differentialdiagnose nicht zutrifft. Geben Sie die Schlüsselelemente der Diagnose an.

Begründen Sie die definitive Diagnose. Diskutieren Sie die Diagnosefindung (Zielgerichtet, überflüssige Untersuchungen, wann hätte man – weswegen – schon früher wissen können, was hätte man sich ersparen können….)

5 Therapiemöglichkeiten

mit Wirkungen und Nebenwirkungen, Indikationen und Kontraindikationen der einzelnen Optionen: psychische Betreuung, Medikamente, konservative Therapien, physikalische Therapie, Intensivtherapie, operative Therapie.

Beschreiben Sie die Stellungnahme des/der Erkrankten zur Therapieentscheidung und die ArztIn/PatientIn Kommunikation. Auch hier, ein Intensivpatient kann oft nicht sprechen, aber beschrieben werden.

6 Therapiewahl

Begründung (was ist medizinisches Faktum, was hat der Patient wollen, wie ist sein Zugang zur endgültig beschlossenen Therapie?). Erfolgserwartung der beschlossenen Therapie. Ist dieser Erfolg eingetreten oder nicht, und warum?

7 Aufklärungspflicht

Umfang der notwendigen Aufklärung (welche möglichen Komplikationen, wie ist die Erwartung des Erkrankten im Verhältnis zur Realität?). Beachten Sie wiederum die forensischen Aspekte! Beschreiben Sie genau, was Sie bei der Aufklärung benennen müssen, und was nicht.

ACHTUNG: „Der Patient wurde genau aufgeklärt und hatte kein weiteren Fragen" oder ähnliche allgemeine Beschreibungen sind forensisch auch nicht haltbar und führen daher zu einer drastischen Verschlechterung der Beurteilung!

8 Die Therapie und der individuelle Verlauf

Hier wiederum beschreiben Sie, was die Therapie bei dem gegenständlichen Fall bewirkt, und ob dies allgemein auch so zu erwarten ist. Bei Operationen sind die Anatomie und die durchgeführte Operation zu beschreiben (kein Abschreiben des Op-Berichtes, - wenn Sie nicht dabei waren, dann nur Angabe der Methode und des Outcome). Beschreiben Sie aber immer mögliche alternative Operationsmethoden so, dass Sie einem Patienten die Methoden und den Grund, warum welche Methode zur Anwendung kommt, erklären können.

Gibt es keine Alternativen - dann erklären Sie, warum!

Diskutieren Sie die Therapie aus der Sicht des Erfolges, Misserfolges.

9 Empfehlung

Niemand sollte ohne weitere Therapie- und Kontrollempfehlungen entlassen werden: welche Therapie, Rehabilitation oder Schonung ist wie lange indiziert, wann sind Kontrollen (was soll kontrolliert werden und wie) indiziert?

Wie wurde die Einhaltung der Empfehlungen mitgeteilt, gesichert, überprüft. Was ist die Konsequenz der Non-Compliance, weiß der Patient das?

10 Zusammenfassung

Sie sollen jedoch hier auch Ihre Meinung zu dem Fall beschreiben. Diskutieren Sie den Fall. Wie?

Sie erklären uns, ob Sie diesen Fall als Standard - Paradebeispiel für eine oft vorkommende Erkrankung sehen (z.B. Hypertonus, Diabetes, Appendizitis, Gallenkolik, Radiusfraktur, kindliche Hernie) oder ob Sie den Fall als Beispiel für eine komplizierte Diagnostik, seltene oder besonders erfolgreiche oder frustrane Therapie oder als Beispiel gute oder schlechte Patientencompliance präsentieren wollen. Was haben Sie als gut, was als verbesserungswürdig gesehen?

Was kann jedermann, egal ob Studierender, Behandelter oder Behandelnder aus diesem Fall lernen, was sind die „teachable Moments"?

Was an diesem Fall ist allgemein gültig?

Ihre Beurteilung erfolgt nach der Vollständigkeit der Berichte unter Berücksichtigung der Voraussetzungen (Erkrankung, Patient, Verlauf, Zusammenfassung, teachable Moments usw.).

Fallaufbau-FGen-FG3-FINAL_2012.DOC 2

Figure 1. Shows the layout of the patient case file form, which the Medical University of Graz provides their students with, in order for them to summarize their findings.

4. A possible solution

The basic idea of this Master's project is to build a system, comprising of a web portal, usable for both, students and supervisors, with the students side of the portal being linked to a mobile device. The current generation is living in a world where mobile devices are everywhere, it seemed quite reasonable to make use of the flexibility provided by such devices and to adapt it for our purposes. Therefore, it seemed reasonable that the use of an iPad in exchange of a normal patient case file is a proper solution for the students side of the system. This quite handy device provides several advantages (e.g. flexibility and connectivity), which makes it the perfect candidate to serve a student's needs. By making use of a combination between an iPad application and a web portal, the students are enabled to note down and address their problems and comments immediately. The novel purpose of the developed system is to give students and supervisors an instrument, which allows for regular interactions between both parties without first having to hand in the case file and find an appointment of mutual agreement. With the new system both parties shall be able to directly document medical cases, within their side of the web portal and in the case of students also in the iPad application. The information stored in the iPad app gets automatically submitted to the additionally provided web portal for students and supervisors. Subsequently, students are provided with the possibility to add text as well as photos to their case files and to address problems as and when they occur, using the iPad application. If necessary, even while still standing in a patient's room. When a supervisor opens his/her side of the web portal, the system automatically displays a list of notifications, containing information about newly created cases, recent updates or deleted case files of their students and enables him/her to browse through all the details of every case. Subsequently, he/she is able to read through all the ideas of the students, learn how they would solve or approach a certain case and most importantly he/she is able to leave comments to every entry a student has made. This gives the supervisor the possibility and flexibility to effectively provide supervision and advice at any time, by immediately pointing the students in the right direction. After the supervisor submits his/her suggestions to a certain case file, the students receive direct notifications about the recent changes to their files, via the web portal or iPad app. The students can make use of the supervisor's suggestions and implement

them directly on the iPad or revise their reports on their computer, using the web portal. The developed novel reporting system deals with all previously mentioned issues of the current approach and provides a more effective way of applying the concept of Problem Based Learning to the field of medical education. Further this new system shall serve as the first part of a completely new teaching system (Figure 2) used by the Medical University of Graz, which consists of several other parts like a key-competence catalog, student journal and student portfolio. As such, the findings of this thesis are meant to serve as a foundation on which all further parts will be built on.

The core part of the proposed new teaching system is case conference (blue box), which represents the work in this thesis. The part on the left of the core system is intended to help the student during the learning phase, where the current case is matched against a set of multiple choice questions, which helps the student to get more knowledge about the problem he/she is working on, which also fits nicely in the concept of Problem Based Learning, where gathering additional knowledge about a certain topic plays a key role. The right part shows how the student could build his portfolio, by using a keyword match with the case report, a skill set for the user is generate. The idea is that the supervisor confirms this skill set, which than transforms itself into the portfolio of the student, which in turn could be used by the student to apply for a specific position.

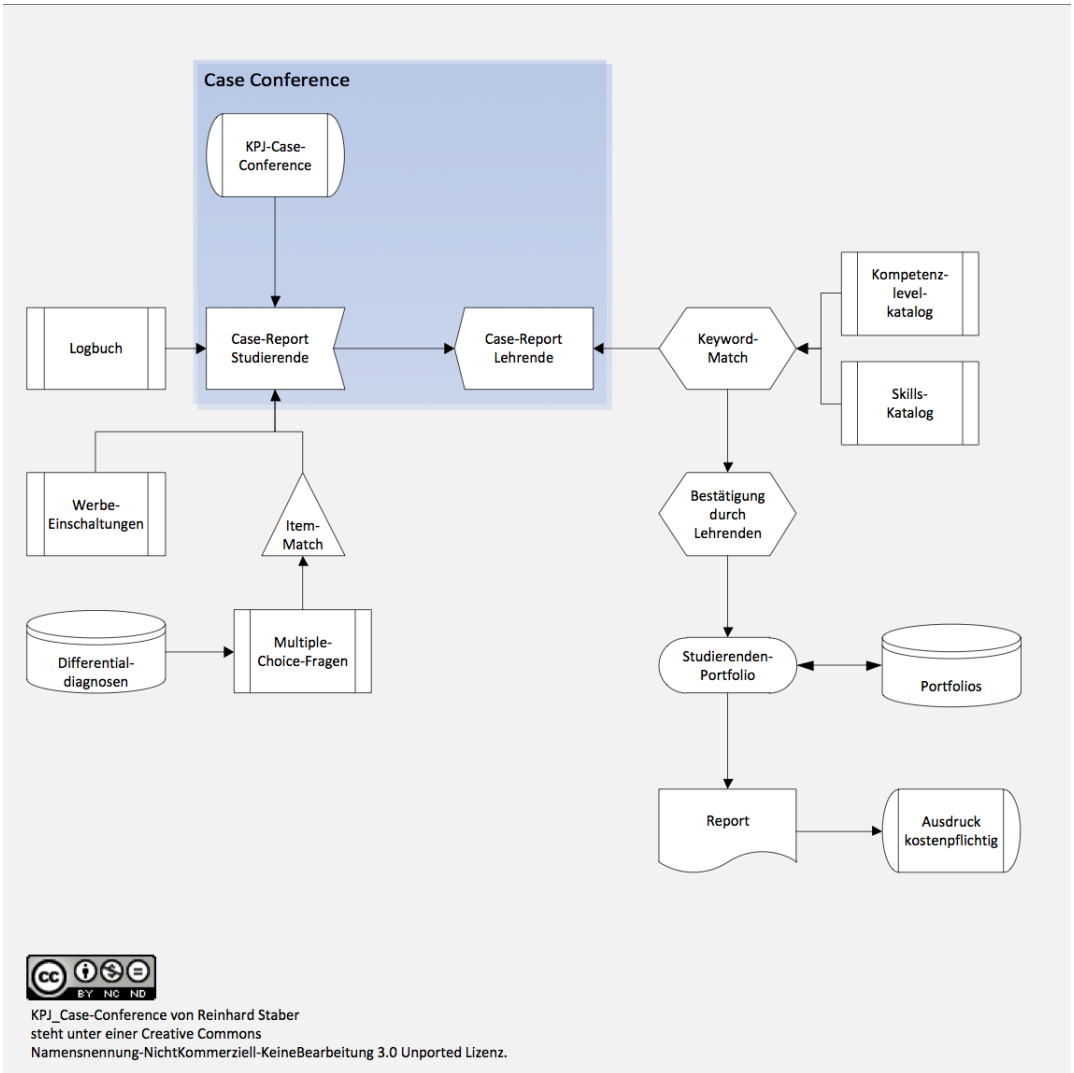


Figure 2. Shows the overall concept of the proposed new teaching system which should be built in the long term. The blue rectangle marks the core part of the system; the developed reporting system.

5. The technology stack

This chapter gives a brief overview of the technology stack which was used to build the previously described report system. The main goal was to utilise state of the art technologies to build a robust and user friendly system. After doing some research the following technologies and programming languages have been identified as suitable building blocks for the software behind the proposed solution, which will be discussed in detail later.

5.1. Python

Python is a so-called interpreter programming language which was created by Guido van Rossum in 1989 as a successor of the programming language ABC [van Rossum, 2009]. Interpreter language means that there is no need to compile your code before you can run it. You just execute your program and the content is interpreted and executed.

One key concept of Python is that the language itself doesn't define how you have to use it. It basically is just depending on the specific problem which needs to be solved as Python supports functional, structured and object oriented programming and thus enables a broad range of applications. Like many modern programming languages, Python supports dynamic type, instead of the old fashioned static type you have in languages like C.

It also supports integration of modules, which comes in handy if you have to deal with time critical code. One can write this code in Assembler or C and integrate the module to execute it in Python and use it to boost the performance. There are certainly other areas where one might want to make us of this. The interested reader can find more about this versatile programming language in the following source [Pyt, 2013].

5.2. Django

Django is a web application framework which is built on top of Python (see Section 5.1.). It was invented by Adrian Holovaty and Simon Willison in 2003 who

used it for the Lawrence Journal-World newspaper's backend and frontend. This Python-based web framework, which focusses on a quick and clean development and maximum of automatisation, was open sourced in 2005 and has gotten it's name from famous guitarist Django Reinhardt. Since 2008 the Django Software Foundation supports and maintains this popular web framework on a professional and non-profit basis [Dja, 2013].

Django is a framework based on several design philosophies. Three of them are described in the following:

- Louse coupling

  This means that individual components in the system are supposed to have just as little "knowledge" about each other as possible. This independence of individual parts of the framework makes the reuse of the code easier and leads to a cleaner overall software design.

- Don't Repeat Yourself (DRY)

  Don't repeat yourself during coding. One piece of code should only exist in one place. Having it in in multiple places makes it difficult to maintain.

- Consistency

  Django is supposed to be consistent from low level Python APIs up to the high level Django APIs with no exceptions to this rule.

Although it's represented in a different way, Django is tightly bound to the Model-View-Controller principle (5.2.). Django uses Model, View and Template to store and represent the data. The model employs data relation mapping, which provides all the benefits of object oriented programming, like aggregation and inheritance, and maps it to a relational database to store data. Thereby, the view serves as the callback function which is used to render the content if you access a specific url. The template determines how the data is finally presented to the user. Obviously the given description lags the controller keyword, as neither of the mentioned components in Django is actually called controller nor does one of these components

fulfill the job of a controller. Within the Django documentation the framework itself kind of takes on the function of the controller. In fact, it is the framework itself which puts together all the individual parts and helps them to communicate with each other. Further information about this web framework can be found in [MVC, 2013].

Model View Controller (MVC)

This concept is a design pattern which helps engineers to maintain a better separation of specific components. It is used for implementing the user interface to a developed system and, like suggested by it's name, it breaks down a software application into three components; the model, view and controller. The basic interaction between those components can be described as follows. The user sends commands and edits the model by operating the controller. The altered state of the model then updates all the corresponding views. The views then generate a visual representation of the updated data stored in the model, which is then finally visible for the user.

PyCharm

For writing code in Python a suitable integrated development environment (IDE) is needed. The environment of choice for this thesis is the popular PyCharm [pyc, 2013] environment because it works for many different platforms and thus also provides support for Django. This makes it very easy to write and debug Python code for our purposes.

5.3. PostgreSQL

After evaluating several different methods of storing the generated data, PostgreSQL was used. It's a very well known open source relational database system. A thorough description of the functionality of data storage systems like PostgreSQL exceeds the scope of this thesis and thus won't be described in detail. However, the interested reader might find the following source [Pos, 2013] useful.

5.4. iOS

Within the proposed software solution the students are provided with a mobile application for an iPad. Thus the mobile app is build on top of Apple's iOS operating system, which was introduced in 2007 and is now one of the two most popular mobile operating systems worldwide, besides Google's Android. The reason for choosing iOS over Android simply originates in our previous experience with building apps on this platform. iOS is a mobile operating system which is based on Apple's computer operating system OSX. [ios, 2013a] [ios, 2013b]

6. Building the solution

This chapter will consider key aspects of the proposed solution, how the main tasks of building the novel report system were approached on a technical level, the problems which occured during its realisation and provides insight into how the problem domain has been transformed into the final software solution presented within the frame of this thesis. As such, this chapter aims to introduce the reader to the core points of the implementation and describes crucial features in more detail.

6.1. The Layout

First the blueprint of the hospital's standardised report form was taken, which can be seen in Figure 1 and used as a template. From this template the most important parts were extracted and implemented into the frontend of our developed web portal and iPad application. The frontend itself will be discussed in it's own dedicated Section 7.
However, it should be mentioned that the whole functionality and layout of the web portal, which has been based on the paper report form, is mirrored by the iPad app, which makes it easy for students to create and edit cases on both platforms.

6.2. The Model

At the very beginning of every new software system one has to think about a suitable data model and data structures which are used to comprehend and solve a given problem. It was crucial to build a data model, which covers the following model relationships (also see Figure 3):

- Professor - Subject
- Professor - Case
- Student - Case
- Case - Entries of a case
- Professor - Professor

Of all these mentioned model relationships, the Professor-Professor relationship is the most complex one, as it always involves three supervisors per subject. It's important to point out that those supervisors don't act on the same hierarchy level, which means that one supervisor is superior to another and thus also holds superior rights. This means a higher ranked supervisor must be able to overrule an inferior one, but not the other way around. With regards to being able to supervise and comment on a student's case file, a well thought through solution is needed in order to prevent that the reporting system turns into a discussion platform for supervisors who might not agree about a certain topic or have different opinions. In order to solve this problem a solution based on a simple hierarchical approach is used, meaning that as soon as a superior supervisor has commented on a case file, the inferior one does not have the rights to comment anymore. Further every supervisor has a unique color to indicate which of them already commented on a case. The coloring is based on what is referred to as a "traffic light system", with green representing the supervisor with the highest rank, blue the supervisor of medium rank and red the lowest ranking. The implementation of this solution can be found in Section 6.6.

One other important factor which needs to be considered while working on the model strategy is that the processing of a case file holds a work in progress status until a student finally submits it. Subsequently students are provided with a versioning system (see Section 7. and 6.5. for further details), which allows them to store and revert to previous versions whenever they want or need to. The ability of accessing the version history also provides a valuable tool for the supervisors as they're able to track back the whole working progress of their students.

Last but not least the individual models are incorporated to the data model. This builds the foundation for our Model View Controller concept, which is described in Section 5.2. Figure 3 represents an UML[2]-like diagram, which displays how these models work together within the data model structure. An expert of UML class diagrams will quickly notice that the diagram doesn't a hundert percent conform to the usual UML standard. However, the legit reason for this is that Django, the web framework which is used for the backend, [Dja, 2013] uses object

[2]Unified Modeling Language

relation mapping for storing the data. This means that the classes in our model are automatically mapped to tables in the database, which in turn explains why our UML diagram looks more like a class diagram.
Basically the system consists of the following data objects:

- Subject
  Subject represents the subject of a case (like surgery, geriatrics and emergency medicine) which is marked with the subject - case relationship. Further a subject consists of three users which are the supervisors for a specific subject and which are in turn allowed to comment on a specific case with this specific subject.

- CaseUser
  CaseUser represents a user in the system. The system uses the *is_student* flag to distinguish between a student and a supervisor. CaseUser has a connection to the subject of case to outline the user - subject relation, a connection to case to outline the student - case relationship and a connection to comments to outline the supervisor - comment relationship.

- Case
  Case represents the case file in the system. There is a user - case relation which represents the student who creates the case. Further a subject - case relationship is made to outline that every case must have a subject and thus every case has three supervisors.

- Comment
  Comment represents a comment for an entry in the case file. The comment stores the text which was entered and the case entry this comment belongs to. A user - comment relationship is made which basically is the last supervisor who altered the comment.

- Anamnese, Befunde etc.
  Those items represent an specific entry of a case file.

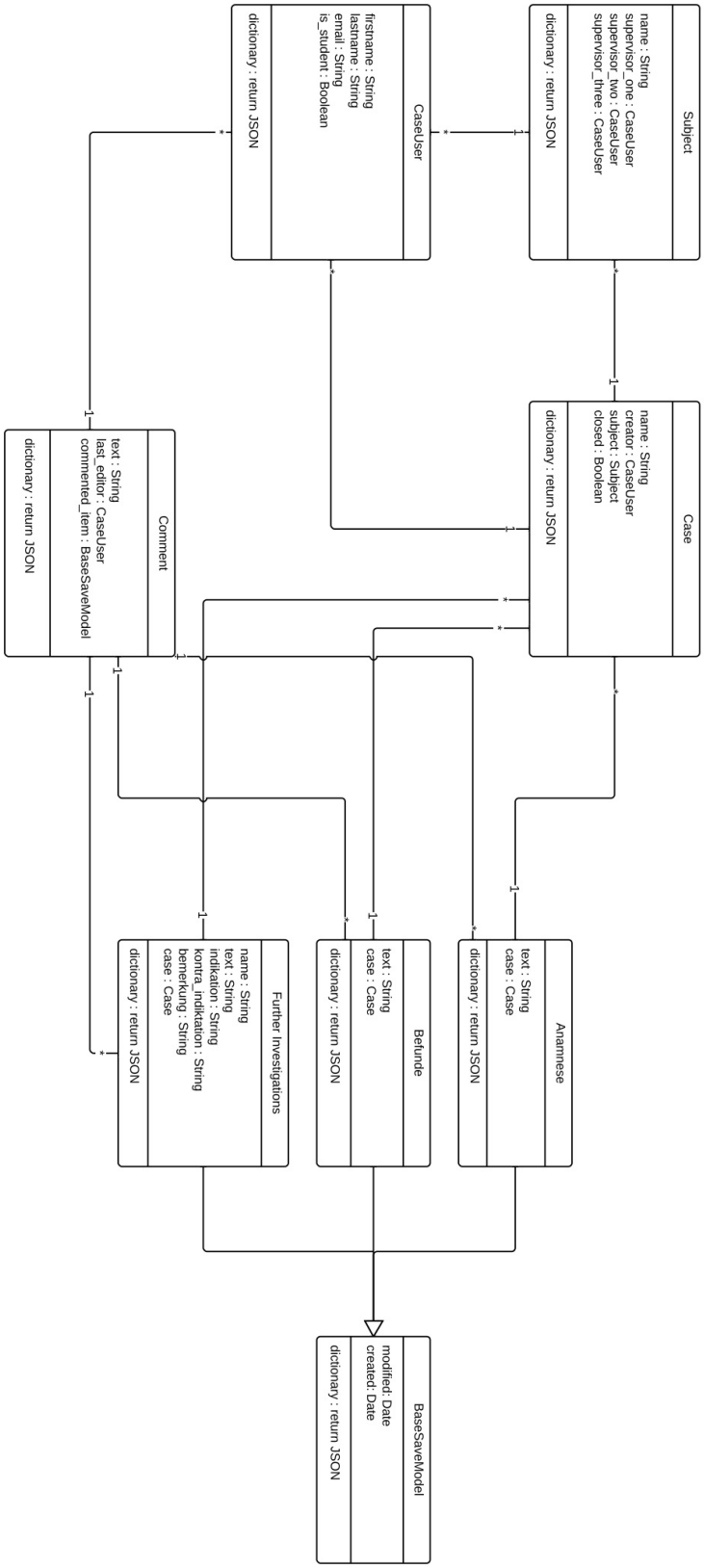


Figure 3. UML like diagram of our models

6.3. User Authentication

The web framework Django has a built in user authentication system, which handles user accounts, groups, permissions and a cookie based user session system. The main focus at this stage is the user object which represents a single user within our system. Basically our system features three different user groups:

- The student user group
- The supervisor user group
- The admin user group

The admin user group is only used for the admin interface which is further used for rights management. There is no direct access from the admin interface to the backend. Further, for reasons of data security, no admin has access to the data which is stored by students and supervisors.

In order to provide a reliable service for our students and supervisors it has to be guaranteed, that at every point within the application only authenticated users can access the content and that the correct content is rendered for the respective authenticated user group. A student should of course be able to access and see the student version of the web app and a supervisor the dedicated supervisor version. Listing 1 shows the code, which is responsible for the reliable validation of the current user's rights to access the respective content.

```
1 def studentOpen(request):
2
3     if not request.user.is_authenticated() or not request.
        user.get_profile().is_student:
4         return HttpResponseRedirect('../account/login')
5     else:
6         return render_to_response('student/open.html',{'
            cases': cases}, context_instance =
            RequestContext(request))
7
```

```
8
9  def professorOpen(request):
10
11      if not request.user.is_authenticated() or request.user
           .get_profile().is_student:
12          return HttpResponseRedirect('../account/login')
13      else:
14          return render_to_response('student/open.html',{'
               cases': cases}, context_instance =
               RequestContext(request))
```

Listing 1: Authentication example for accessing the open cases of a supervisor/student.

The first part of the condition uses the built-in *is_authenticated()* functionality of the user object to check if the user needs to be logged in or not. For the second part the built-in user model had to be extended in order to link it to a profile. This profile can be of any class and allows anyone to finally customize and add certain attributes to the user object. This can be achieved quite easily by simply creating a new model and using it as user profile. Listing 2 shows what the created user profile looks like.

```
1  class CaseUser(models.Model):
2
3      user = models.OneToOneField(User)
4      is_student = models.BooleanField(default=False)
5
6      def __unicode__(self):
7          if not self.user.first_name and not self.user.
               last_name:
8              return self.user.username
9          return self.user.first_name + " " + self.user.
               last_name
```

Listing 2: User Profile Model to extend the user object.

One remaining question is how to trigger the creation of the user profile at the same time the new user is created. Django includes a quite powerful *signal dispatcher* which notifies parts of the framework when another part of it changes. This happens without a tight connection between those two parts. One can simply imagine this process as a sender, which notifies a range of receivers, provided some action has taken place. This is quite handy if many different places want to get notified about a specific change. Needless to say this comes with a major drawback, as your code gets more difficult to maintain the more you rely on such methods.
Django comes with a complete set of predefined signals and even allows you to extend it with your own signals. In this case every *save* of a user object should be tracked. Django provides two signals for this action. The first one, which is triggered before the item is saved (*pre_save*) and the second one which is triggered after the item is saved (*post_save*). For this purpose *post_save* is utilised, as the idea is to create the profile after the user object gets saved for the first time. Listing 3 shows what the callback function for the system looks like and how this function gets registered as *post_save* callback.

```
1 #callback function for the signal
2 def create_user_profile(sender, **kw):
3     user = kw["instance"]
4     if kw["created"]:
5         up, created = CaseUser.objects.get_or_create(user=
              user)
6         up.save()
7
8 #register the callback function
9 post_save.connect(create_user_profile, sender=User)
```

Listing 3: User Profile Model to extend the user object.

This effectively solves our problems and allows us to extend the user model pre-built by Django according to our system requirements.

As sessions only exist in web browsers this doesn't help when a user tries to access his/her data from a mobile device (like an iPad) via an Application Programming Interface (API). For the demo implementation HTTP Basic Authentication [htt, 2013] is used. However, Django doesn't support this kind of authentication out of the box. Nonetheless, there are easy and smart ways to achieve this. For example, one could either extend the *RemoteUserMiddleware* class or write a decorator for the view that shall be accessed. Within the developed system a view decorator was used for the authentication, as this method has been shown to work nicely within previous projects. Listing 4 shows how the implementation of the decorator has been realised.

```
1  def view_or_basicauth(view, request, test_func, realm = ""
       , *args, **kwargs):
2
3      if 'HTTP_AUTHORIZATION' in request.META:
4          auth = request.META['HTTP_AUTHORIZATION'].split()
5          if len(auth) == 2:
6
7              if auth[0].lower() == "basic":
8                  uname, passwd = base64.b64decode(auth[1]).
                       split(':')
9                  user = authenticate(username=uname,
                       password=passwd)
10                 if user is not None:
11                     if user.is_active:
12                         login(request, user)
13                         request.user = user
14                         return view(request, *args, **
                               kwargs)
15
16     response = HttpResponse()
17     response.status_code = 401
18     response['WWW-Authenticate'] = 'Basic realm="%s"' %
           realm
19     return response
```

```
20
21 def logged_in_or_basicauth(realm = ""):
22     @logged_in_or_basicauth
23     def your_view:
24     def view_decorator(func):
25         def wrapper(request, *args, **kwargs):
26             return view_or_basicauth(func, request,
27                 lambda u: u.is_authenticated(),
28                 realm, *args, **kwargs)
29         return wrapper
30     return view_decorator
```

Listing 4: Decorator which is used for HTTPBasicAuthentication.

Subsequently, every API function in the system's backend gets decorated with *logged_in_or_basicauth* to support authentication.

6.4. Displaying the content

In order to understand how the content is finally displayed for the user, a closer a look at all the small pieces and how they work together needs to be taken. Starting by sending the request, processing the request and rendering the data on the user's screen. This can be broken down into 3 functionalities:

- *Url Dispatcher*
- *View Functions*
- Django *Templates*

For a better understanding of their functionality and role within the system, all the points will be discussed in the following.

URL dispatching (Uniform Resource Locator)

Django contains a very powerful URL dispatcher, which makes it very easy to design the URLs for a system. A correct and understandable URL scheme is a key

prerequisite for every successful web application. For this purpose Django builds a mapping table for URLs and views. This mapping table should then be called for all the URLs in the system. The developer needs to register a specific mapping in the *urls.py* file, otherwise a request for this URL would end in a 401[3], if it's not registered. Figure 5 shows the URL mapping which is used for in this thesis.

```
1  urlpatterns = patterns('',
2      url(r'^$', 'casestudy.views.index', name='index'),
3      url(r'^index/$', 'casestudy.views.index', name='index'
           ),
4      url(r'^create/$', 'casestudy.views.create', name='
           create'),
5      url(r'^case/$', 'casestudy.views.case', name='case'),
6      url(r'^student/open/case$', 'casestudy.views.
           studentCase', name='studentCase'),
7      url(r'^professor/open/case$', 'casestudy.views.
           professorCase', name='professorCase'),
8      url(r'^student/closed/case$', 'casestudy.views.
           studentClosedCase', name='studentClosedCase'),
9      url(r'^professor/closed/case$', 'casestudy.views.
           professorClosedCase', name='professorClosedCase'),
10     url(r'^open/$', 'casestudy.views.open', name='open'),
11     url(r'^student/open/$', 'casestudy.views.studentOpen',
           name='studentOpen'),
12     url(r'^student/open/upload$', 'casestudy.views.
           studentUploadImage', name='studentUploadImage'),
13     url(r'^student/open/add', 'casestudy.views.
           studentOpenCaseAddItem', name='
           studentOpenCaseAddItem'),
14     url(r'^professor/open/$', 'casestudy.views.
           professorOpen', name='professorOpen'),
15     url(r'^student/close_case/$', 'casestudy.views.
           studentCloseCase', name='studentCloseCase'),
```

[3]401 is a HTTP status code, find a complete list of codes here http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

```
16      url(r'^closed/$', 'casestudy.views.closed', name='
            closed'),
17      url(r'^student/closed/$', 'casestudy.views.
            studentClosed', name='studentClosed'),
18      url(r'^professor/closed/$', 'casestudy.views.
            professorClosed', name='professorClosed'),
19      url(r'^student/history/$', 'casestudy.views.history',
            name='history'),
20      url(r'^professor/history/$', 'casestudy.views.history'
            , name='history'),
21      url(r'^account/login/$', 'casestudy.views.login', name
            ='login'),
22      url(r'^account/create/$', 'casestudy.views.createUser'
            , name='create'),
23      url(r'^account/logout/$', 'casestudy.views.logoutUser'
            , name='logoutUser'),
24      url(r'^api/login/$', 'casestudy.api.login', name='
            login'),
25      url(r'^api/case/$', 'casestudy.api.studentCases', name
            ='studentCases'),
26      url(r'^api/case/(?P<case-id>\w+)$', 'casestudy.api.
            caseDetails', name='caseDetails'),
27      url(r'^api/caseitem/(?P<item-type>\w+)/(?P<item-id>\w
            +)$', 'casestudy.api.caseItemDetails', name='
            caseItemDetails'),
28      url(r'^api/caseitem/$', 'casestudy.api.updateCaseItem'
            , name='updateCaseItem'),
29      url(r'^api/case/(?P<case-id>\w+)/history/$', '
            casestudy.api.caseItemHistory', name='
            caseItemHistory'),
30      url(r'^api/subjects/$', 'casestudy.api.subjects', name
            ='subjects'),
31  )
```

Listing 5: *urls.py* used in the solution.

The first part of the mapping could be any regular expression[4] which suits a system's needs. The second part is simply the specific Python function which is called for a specified URL. When requesting a specific URL, Django iterates over all the patterns in the system and the first match it finds is then returned and executes the underlying function.

View Functions

The *view* code needs to live somewhere, that's why Django has chosen the *views.py*-file for it by convention. If a specific URL is called, which is mapped against a *view function*, the function gets called with one parameter. This parameter is a *HTTPRequest* object, which contains all the usual request information, including e.g. request type, request body etc. The function is then expected to return a *HTTPResponse* object, which represents the page that gets finally rendered for the user. Listing 6 shows what this rendering might look like, using an example taken from the problem domain. The chosen example concerns the *open case view* from the students side of the system, which renders all open cases of a student. The condition simply performs a check of whether the current user is authenticated and whether it's a student or not, as only students should see the student version of the cases.

```
1 def studentOpen(request):
2
3     if not request.user.is_authenticated() or not request.
        user.get_profile().is_student:
4         return HttpResponseRedirect('../account/login')
5     else:
6         cases = Case.objects.filter(closed=False,creator=
            request.user.get_profile()).order_by('-created'
            )
7         return render_to_response('student/open.html',{'
            cases': cases}, context_instance =
            RequestContext(request))
```

Listing 6: Example of how content is rendered in Django.

[4] http://www.regular-expressions.info/tutorial.html, accessed 2014-01-08

Line 6 loads all the open cases of the current user, sorted by their creation time stamp. Line 7 utilises a shortcut function to render the response. The first part of this function is the template which should be rendered as a response and the second part is the data that should be handed over to the template. This data can be used in the template to render content depended things.

Templates

Django provides the quite convenient possibility to adapt and use their template language in the HTML[5] code. This is quite helpful, as generic view templates can be build which can be filled with data. This provides several benefits, because it requires significantly less code, provides higher reusability and allows for an easier overall maintenance. To avoid ambiguity, Listing 7 will reuse the example from the previous section to explain how the templating system can be used to render a list of all student case files, by iterating over all *cases* in our view function and by adding a *div*, which displays the name and the created timestamp to the DOM[6]-tree.

```
1 {% for case in cases %}
2     <div class="case">
3         <a href="/case?id={{ case.id }}">{{ case.name }}
              erstellt am {{ case.created }}</a>
4     </div>
5 {% endfor %}
```

Listing 7: Example of how the template system renders a list of cases.

[5]Hypertext Markup Language, accessed 2014-01-08
[6]Document Object Model, http://www.w3.org/DOM/, accessed 2014-01-08

Figure 4 finally sums up how all of the three parts work together as a functioning process.

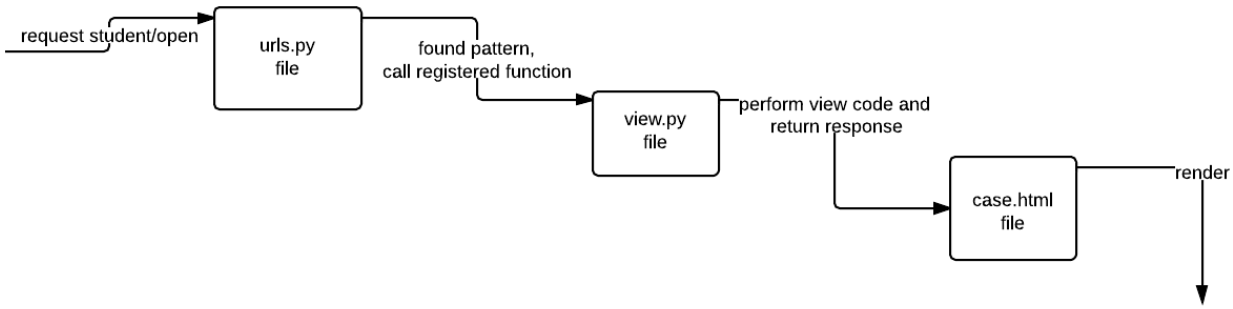


Figure 4. HTTP request flow which renders a HTML website successfully

6.5. Handling different versions

The system is designed in such a way, that it's capable of handling multiple versions of every entry in a case file. A new version should be created automatically every time the student changes the text of the current entry he/she is editing. However, since reloading the whole webpage after every change made by a user is not conform with a good user experience, the concept of *Ajax* is used to improve it.

Ajax

Ajax which means Asynchronous Javascript and XML[7], is a concept which allows the developer to exchange data between the server and the browser in an asynchronous way. This means that it allows to send HTTP requests to the server, while loading the requested data and changing the currently presented web page

[7] eXtensible Markup Language, http://www.w3schools.com/xml/, accessed 2014-01-08

and most importantly without leaving it or without having to reload the whole page every single time. This paves the way to an improved and smooth user experience, as this allows the website to behave more like a software a user would use on his/her computer.[aja, 2013]

In the project's case this means that whenever a user edits an existing field and hits the save button, an Ajax request is created and sent to the system server. In order to indicate that something will be created *POST* is used as http method. Listing 8 demonstrates how our Ajax request is generated.

```
1  .ajax({
2      type: "POST",
3      url: "/student/open/case",
4      traditional: true,
5      data: {
6          'csrfmiddlewaretoken': '{{ csrf_token }}',
7          'case_id': {{ case.id }},
8          'class': class_name,
9          'class_id':class_id,
10         'text_name': item_names,
11         'text_value' : item_values
12     },
13     success:function(data){
14     //handle the callback...
15     }
16 }
```

Listing 8: Example Ajax request for creating a new version.

Line 3 defines which endpoint should be called, lines 6-11 contain the data sent as the *POST* body. This data includes all the needed information to create a new version of the updated entry, like e.g. the ID of the current entry version, the class to be changed, which value of the class needs to be changed and of course what the new value is. Effectively a new version of an entry is only created when the user

hits ”save” and provided the new text is different from the old one. Subsequently, if both versions of an entry are equal and do not differ at all, the system just maintains the old entry.

For the sake of completeness it should be mentioned what the *csrf_token* is. This token is used by Django to simply protect itself against *cross-site-request--forgery*. For further details about *cross-site-request-forgery* the interested reader might find the following literature [cro, 2013, dja, 2013] informative.

6.6. Managing multiple supervisors

As previously mentioned (see Section 6.2.), every case file has three supervisors of different rank and hierarchy level. Although they are assigned with a different hierarchy status and considering the superiority of one supervisor over another, which includes the ability to overrule inferior colleagues, they nonetheless all serve the same purpose. Their main task is to guide the students and help them whenever they need feedback, input or perspective during the processing of a case. Consequently, if a supervisor of superior rank adds a comment, one from a lower rank automatically loses the right to comment to this specific item of the case. However, if the student creates a new version of the same entry, every supervisor in the hierarchy obtains the right to comment to it back until a superior colleague leaves his/her comments again. The idea of this system is based on a simple traffic light system, which is discussed in Section 6.2. This might seem confusing at first sight but there are in fact several reasons why it is implemented this way.

The main reason for choosing a solution like this can be summarized as follows:

- Usually supervisors on higher levels have more experience,
- further they have a better knowledge and background of the subject they are lecturing
- and finally, a discussion should be avoided, because a permanent alteration of the supervisor comments could lead to a lot of confusion for the students who rely on adequate suggestions and advice.

First of all let's take a look at Listing 9 where the supervisors are stored in our data model. Every case file has a subject (like surgery, geriatrics and emergency medicine) and every subject has three supervisors.

```
1  class Subject(models.Model):
2      name = models.CharField(max_length=200)
3
4      supervisor_one = models.ForeignKey(CaseUser,
           related_name="supervisor_one")
5      supervisor_two = models.ForeignKey(CaseUser,
           related_name="supervisor_two")
6      supervisor_thr = models.ForeignKey(CaseUser,
           related_name="supervisor_three")
7
8      def __unicode__(self):
9          return self.name
10
11     def dictionary(self):
12         return {
13             'id': self.id,
14             'name': self.name,
15             }
```

Listing 9: Shows the implementation of the *Subject* model.

The supervisors stored in the subject are allowed to comment on a case with the same subject. Managing the multiple supervisor interaction and comments section of the system needed for a solution on two different levels. Partly on the view level and partly on the template level. Let's start with the *view.py* and in particular with the corresponding *allowEditing* function. Firstly, all the items of a case are loaded. Next, the system performs a check for every item the student has created. During this check the system looks whether an item has any comments and wether the current user holds the right to comment on the respective section. For every case item a comment dictionary is returned (see Listing 10), which contains information

about the rights of the current user (e.g. his authority to comment), who has already commented on this item, all other existing supervisor comments and the current user's own comment.

```
1 return {
2     'allow_comment':allowEditingComment(commented_item,
        user, subject),
3     'commenters': commenters(commented_item, subject),
4     'other_comments': Comment.objects.filter(
        commented_item=commented_item).exclude(last_editor=
        user),
5     'own_comment': Comment.objects.filter(commented_item=
        commented_item, last_editor=user),
6 }
```

Listing 10: Comment dictionary which is returned for every item in a case.

The *allow_comment* key gets its value from the *allowEditingComment* function, which checks who of the supervisors has been the last one to comment on a case and compares it to the currently logged in user. The following conditions illustrate the szenarios that have to come into action for an user to lose his/her authorization to comment on a specific task.

- *Supervisor One*, who takes on the role of the lead supervisor, was the last editor and the current user is not this user.
- *Supervisor Two*, who holds the next highest rank, was the last editor and the current user is not this user nor the head supervisor (*Supervisor One*), who holds the most rights.

All other szenarios than the ones mentioned above lead to an allowance and a positive return of the function as seen in Listing 11 and thus grant the current user permission to comment on a case item.

```
1 def allowEditingComment(commented_item, user, subject):
2     comment = Comment.objects.filter(commented_item=
          commented_item, last_editor=subject.supervisor_one)
3     if comment and comment[0].last_editor.pk != user.pk:
4         return False
5
6     if comment and comment[0].last_editor.pk != user.pk
          and user.pk != subject.supervisor_one.pk:
7         return False
8
9     return True
```

Listing 11: Checks wether the current user is allowed to comment on the item.

This means, in the system's template it is simply checked whether the *allow_comment* flag is set or not. If it is set, a text area for the supervisor to add or change comments is displayed. However, if the *allow_comment* flag is not set, the system just shows the comment of the previous (superior) supervisor, without giving the current user a possibility to change it (see Listing 12).

```
1 {%  if case_comment.allow_comment %}
2 //show text area with the comment
3 {% else %}
4 //show paragraph with the comment
5 {% endif %}
```

Listing 12: Shows how the template language is utilised to present/hide the comment field.

6.7. Interface for the mobile client

As students should be provided with a mobile client, in form of an iOS companion app, to enter the cases right in front of the patient, an interface is needed which allows this client to communicate with the system's server. For this purpose a REST[8] API, which handles the communication between the mobile application and the server, was developed.

Representational State Transfer (REST)

Representational State Transfer is an architectural style for building a web service. It was first described in a dissertation by Roy Fieldings. In his thesis Fieldings summarizes the key attributes of REST as follows.

"REST emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems." [Fielding, 2002]

Nowadays one might easily describe it as the best practice for designing a webservice.

In the app mostly *GET* requests are used to retrieve the data from the backend. *POST* requests are then used to add a new version of an entry (generated within the app) to our server's database. JSON is used[9] as format for encoding the data which is sent from the server to the app and from the app to the server.
The following list shows all the endpoints and describes what they are used for. This functions assume that the user which is using the app is already logged in to the service. If there is no valid user object, all of these functions return a 401[10]. Information about the realisation of the user authentication is given in Section 6.3.

[8] Representational State Transfer

[9] JavaScript Object Notation, http://www.json.org/, accessed 2014-01-08

[10] 401 is a HTTP status code, find a complete list of codes here http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html, accessed 2014-01-08

- *api/case*

  The GET request returns a list of all open cases of the current user. All other request types fail.

```
[
  {
    "closed": 0,
    "id": 1,
    "title": "Fall 1, Gebrochene Nase"
  },
  {
    "closed": 1,
    "id": 3,
    "title": "Fall 2, Ausgekugelte Schulter"
  }
]
```

Listing 13: Shows an example result for *api/case*.

- *api/case/<case-id>*

  The *GET* request returns the details of a specific case. The POST request creates a new case. All other request types fail.

```
[
    {
        "class": "Anamnese",
        "id" : 19,
        "title" : "Anamnese",
    },
    {
        "class" : "Befunde",
        "id" : 20,
```

```
10          "title" : "Befunde",
11      },
12      {
13          "class" : "CurrentDifferentialDiagnose",
14          "id" : 21,
15          "title" : "Vorlaeufige Differentialdiagnose",
16      },
17      {
18          "class" : "FurtherInvestigations",
19          "id" : 22,
20          "title" : "Weitere Untersuchungen",
21      },
22      {
23          "class" : "WorkingDiagnose",
24          "id" : 23,
25          "title" : "Arbeitsdiagnose",
26      },
27      {
28          "class" : "DifferentialDiagnose",
29          "id" : 24,
30          "title" : "Differentialdiagnose",
31      },
32      {
33          "class" : "DiscussionOfDiagnose",
34          "id" : 25,
35          "title" : "Diskussion der Diagnose",
36      },
37      {
38          "class" : "PossibleTherapies",
39          "id" : 26,
40          "title" : "Moegliche Therapieansaetze",
41      },
42      {
43          "class" : "TherapyChoice",
44          "id" : 27,
45          "title" : "Therapiewahl",
```

```
46        },
47        {
48            "class" : "PreOperationDiscussion",
49            "id" : 28,
50            "title" : "Aufklaerungsgespraech",
51        },
52        {
53            "class" : "TherapyProgress",
54            "id" : 29,
55            "title" : "Therapieverlauf",
56        },
57        {
58            "class" : "Recommendation",
59            "id" : 30,
60            "title" : "Empfehlungen",
61        },
62        {
63            "class" : "SummaryAndDiscussion",
64            "id" : 31,
65            "title" : "Zusammenfassung und Diskussion",
66        }
67    ]
```

Listing 14: Shows an example result for *api/case/1*.

- *api/caseitem/<item-type>/<item-ID>*

  The *GET* request returns a specific case-item with a specific item-ID, like *api/caseitem/befunde/1* returns a *befunde* object with the ID *1*. All other request types fail.

```
1  {
2      "comments":[
```

```
3                   {
4               "comment": "Ein erste Kommentar von mir.",
5               "commenter": "prof1@prof.at",
6               "date": "2013-11-29 17:22:29",
7           },
8                   {
9               "comment": "Ich bin mit der Meinung nicht
                    einverstanden.",
10              "commenter": "prof2@prof.at",
11              "date": "2013-04-23 09:25:48",
12          }
13      ],
14      "field":[
15                  {
16              "title": "Befunde",
17              "value": "Das ist der Befund"
18          }
19      ],
20      "images":       [
21          "static/upload/1/Befunde/
                a8061725b39505cc1a5352170cf43088.png"
22      ]
23  }
```

Listing 15: Shows an example result for *api/caseitem/befunde/1*.

- *api/caseitem*

  The *POST* request creates a new case-item. The POST body contains the case-ID, the new text for this item and which item type it is. All other request types fail.

```
1  {
2    "fields":[
```

```
3            "Das ist eine neuer Befund"
4        ],
5        "item_id": 20,
6        "item_type": "Befunde"
7    }
```

Listing 16: Shows an example body for *api/caseitem* which updates the *Befunde* item.

- *api/subjects*

  The *GET* request returns a list of all subjects in the system because subjects are needed for the general case creation. All other request types fail.

```
1  [
2      {
3          "id": 1,
4          "name": "Chiru 2"
5      },
6      {
7          "id": 2,
8          "name": "Pato 1"
9      }
10 ]
```

Listing 17: Shows an example for *api/subjects*.

- *api/case/<case-id>/history*

  The *GET* request returns the history for this specific case. All other request types fail.

```
1  [
2      {
3      "date": "2013-12-19 20:22:45",
4      "fields":[
```

```
5                {
6                    "title": "Befunde",
7                    "value": "Befunde Version Zwei"
8                }
9            ]
10       },
11       {
12       "date": "2012-09-20 19:19:37",
13       "fields":[
14               {
15                   "title": "Befunde",
16                   "value": "Befunde Version Eins"
17               }
18           ]
19       }
20   ]
```

Listing 18: Shows an example for *api/case/20/history* which returns the history for Befunde

7. Exploring the frontend of the developed system

As mentioned before, a system should be built that resembles the usual template style of the old paper version of the report forms on both platforms. The main reason for adapting the renowned layout of the paper form to our system is based on one simple consideration.

By implementing the blueprint of the old standarized paper form, one makes not only sure that the portals feature all the key aspects of the previous form but also that the users are familiar with the layout and processing.

For adapting the paper form layouts and generating our frontend, HTML 5 and CSS (Cascading Style Sheets) was employed. This is how the frontend gets it's final appearance. Figure 5 on page 60 shows what the creation of a new case file on the student's side of the website looks like and Figure 6 shows the corresponding iPad version of the portal.

With every new case opened by a student, he/she is immediately provided with this familiar view and is able to enter and submit the initial data to our system.

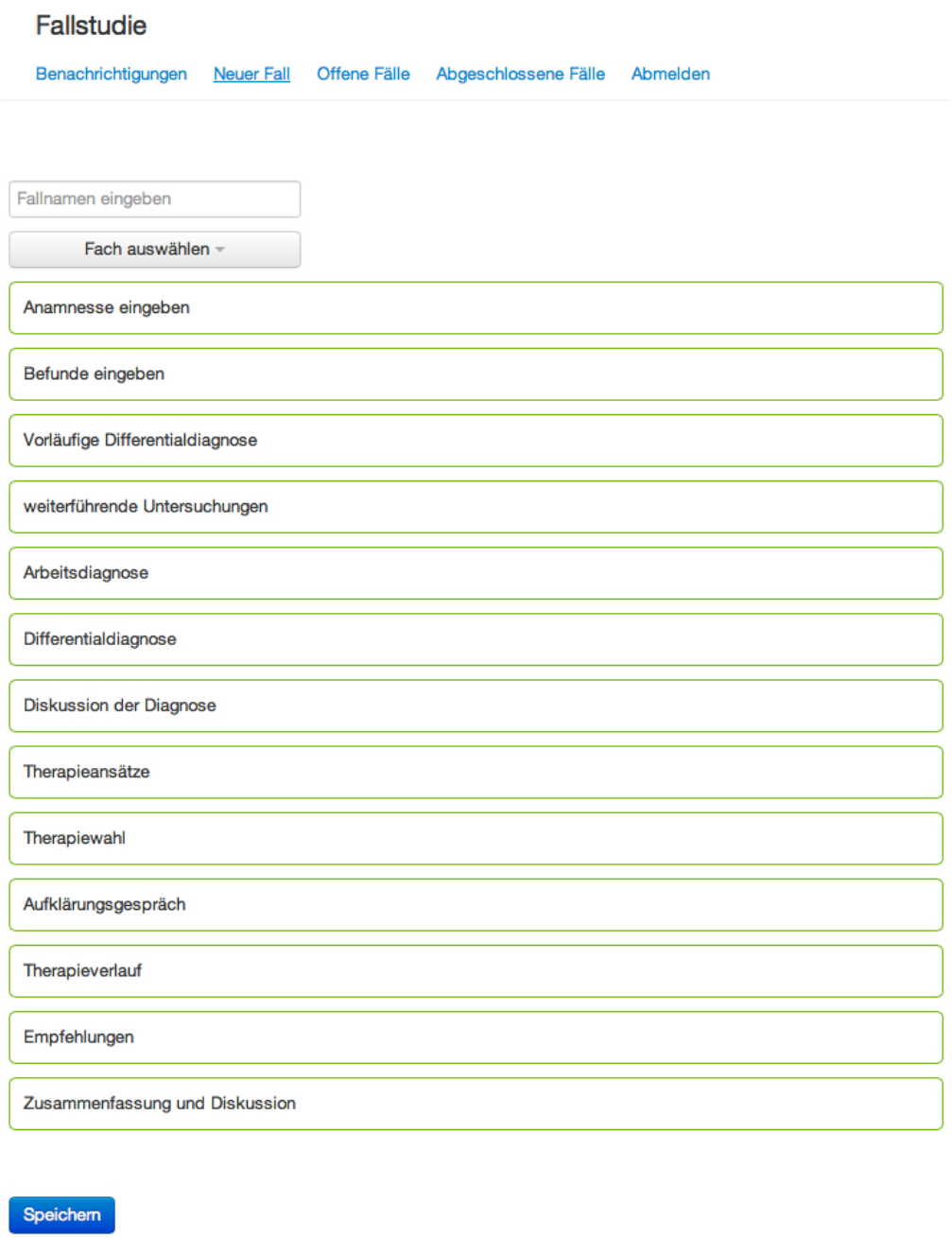


Figure 5. Report (case file) layout, displaying all the relevant entry fields one would also find on a paper report form.

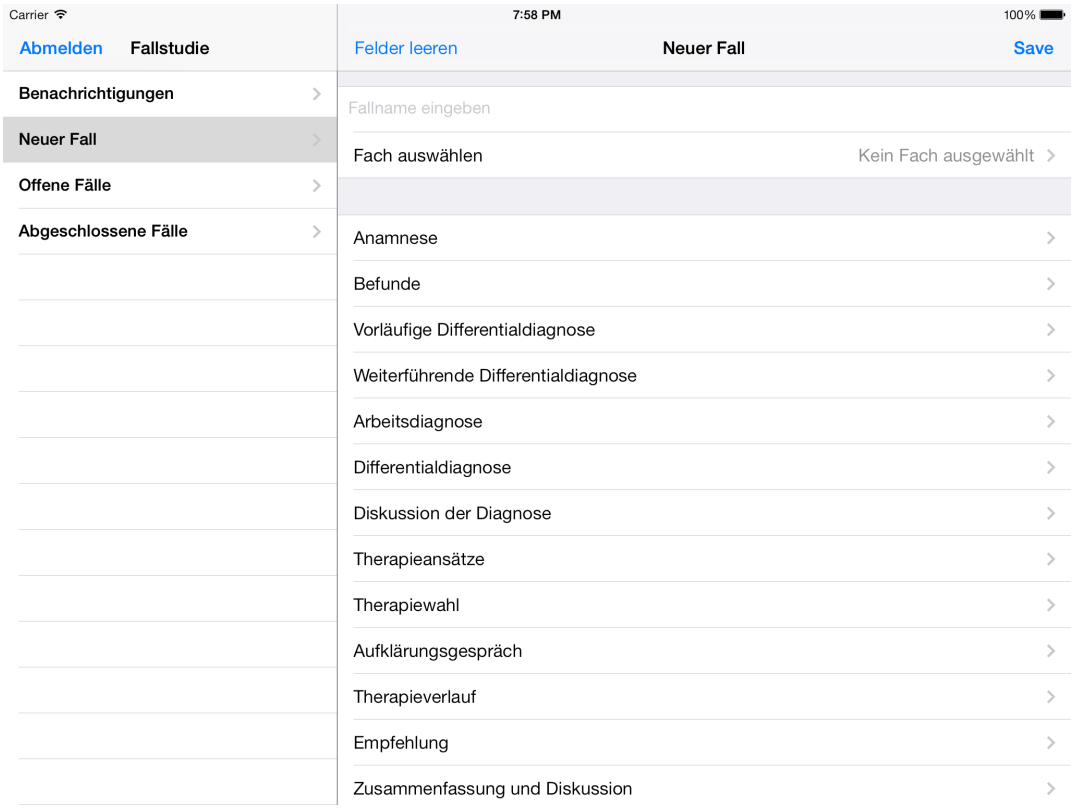


Figure 6. Report (case file) layout on an iPad, displaying the same entry fields as the web portal.

In Problem Based Learning, the learner usually does not have a complete understanding of the problem he/she is facing, at the point when he/she actually starts working on it. While independently working on a case file, students are undergoing a steady learning process, which is designed to increase their knowledge over time. As they obtain a deeper understanding of what the patient is suffering from, how complex the disease actually is and how to treat it, they might not agree with the findings they've made when they initially created the case or they might want to add additional information, which they didn't have in the first place. Therefore, every report has several fields. Those fields need to be filled in and can be edited at every time during the processing of the case.

For this reason, as described in Section 6.5., the reporting system allows the students to create multiple versions of one report, which makes it easier for them

and their supervisors to track the progress.

Figure 7 shows the possibility of changing a current version by pressing the ”Editieren” button of the corresponding field. A user can access the previously saved versions through the links beneath the label ”Alte Versionen”.

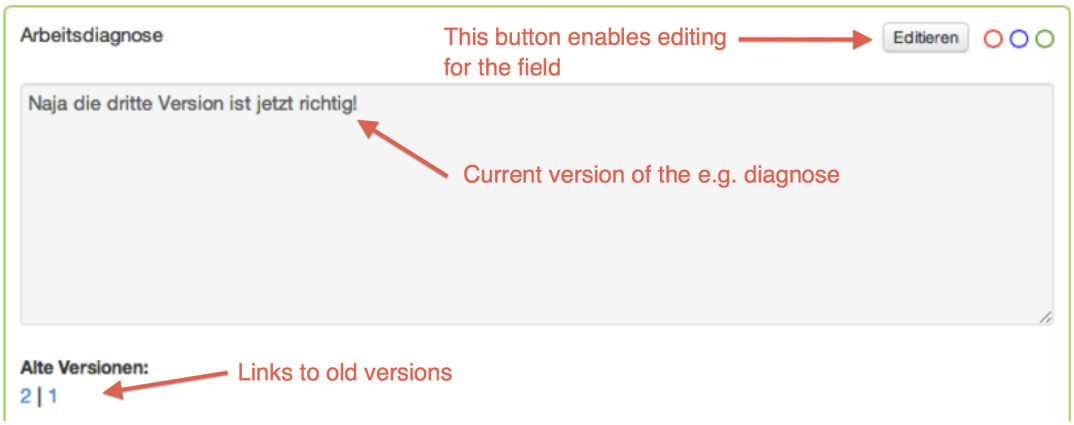


Figure 7. Illustration of the editing view, including the tracking possibility of older versions in the left corner.

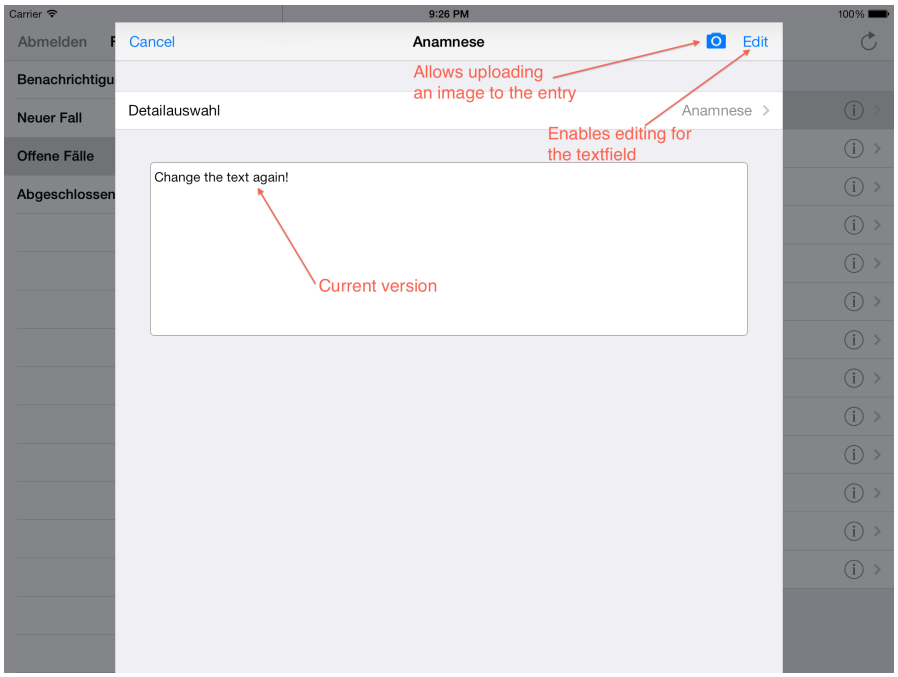


Figure 8. Illustration of the editing view on the iPad

For example, if a student makes a first diagnosis which he/she feels pretty confident about at the beginning, but later on, after investigating the case more precisely and gaining more insight, he/she decides that the initial diagnosis wasn't complete or even wrong, the student just opens the report (with the web portal or iPad app), navigates to the field he/she wants to edit, enters the new data and saves the updated case file again. The system stores the edited field(s) automatically on both platforms. In fact, as long as the student is working on a case and the case is open, the system keeps a history (Figure 9) of all versions of an entry field the student has ever saved. Subsequently, the students as well as supervisors can browse through the changes and track the complete process of coming to the final conclusions.

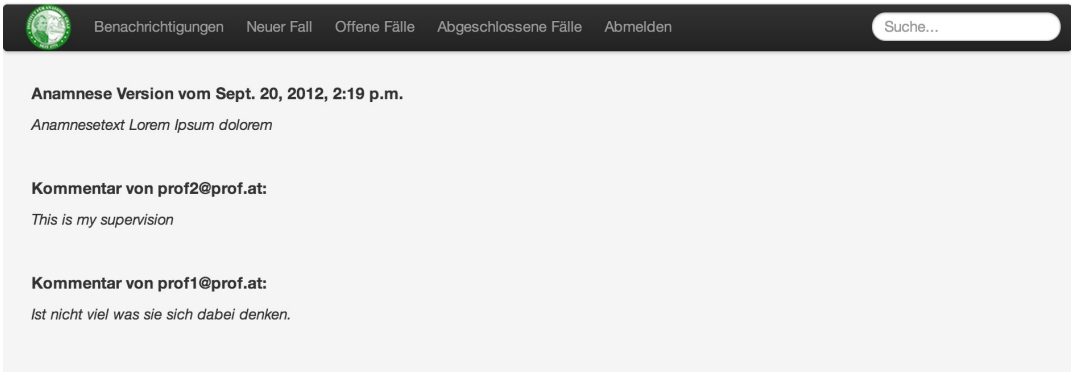


Figure 9. Shows what the history for the "Anamnese" entry field for a case would look like.

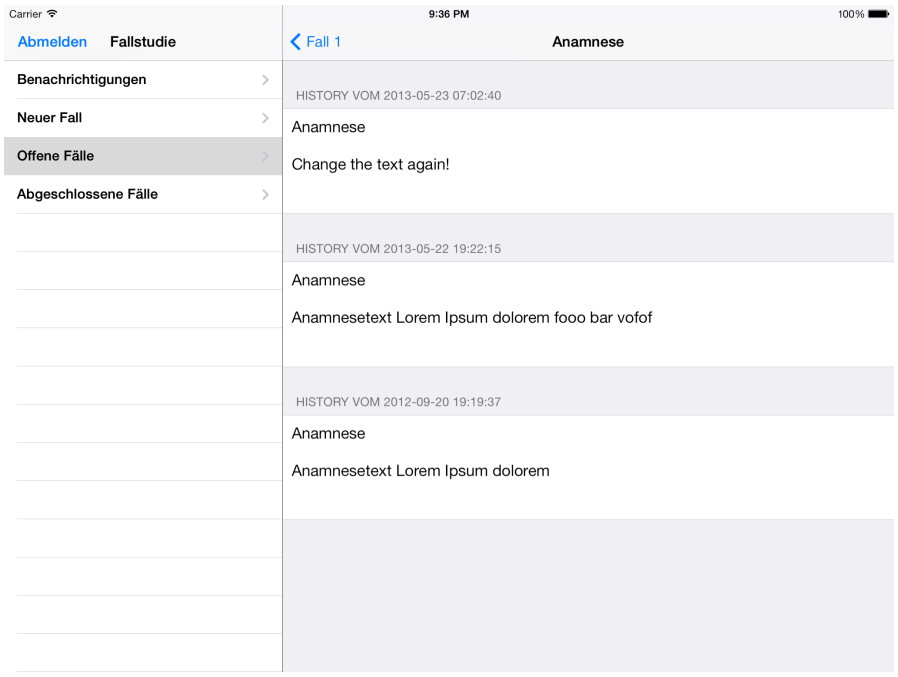


Figure 10. Shows what the history for the ”Anamnese” entry field for a case would look like on an iPad.

This is especially of value to the supervisor, because the ready to hand tracking of the student’s changes provides them with insight in their student’s individual thinking process.

So let’s make a short recap of what has been achieved on the students’ side so far.

Thus far,

- the students were given the possibility to continuously report their findings and add them to the web portal.
- they were provided with the possibility to change the report later on and to save new versions. This includes the possibility keep to track of their older versions.

Next the supervisors' version of the web portal will be discussed and how they benefit of this novel reporting system.

At the end of every month a supervisor gets an overwhelming amount of reports to revise and comment on. Hence, they are provided with the ability of getting a quick and easy overview of the changes a particular student has made to his/her report and of students who have created new files. As solution a notifications dashboard was developed and integrated to both sides of the reporting platform. The student gets notified as soon as the supervisor adds a comment to a case.
Every time a professor logs him/herself in to the web portal, he/she receives an updated list of all notifications, which have been passed on from the students' side of the system. This provides an easy way to check what is actually new and what needs to be done, without loosing track of all tasks. Figure 11 shows a representative view of received notifications.

The notification dashboard contains 3 different types of notifications:

- Notifications about newly created case files of each student
- Notifications about updates of individual case files (new versions)
- Comments of other supervisors (With the possibility of having up to 3 supervisors involved in the reviewing of a report)

As of now, supervisors are able to keep up to date with the work of their supervisees from the early beginning to the final report, ready for assessment. However, they still have to be able to provide feedback on a regular basis in order to provide the best PBL experience possible.

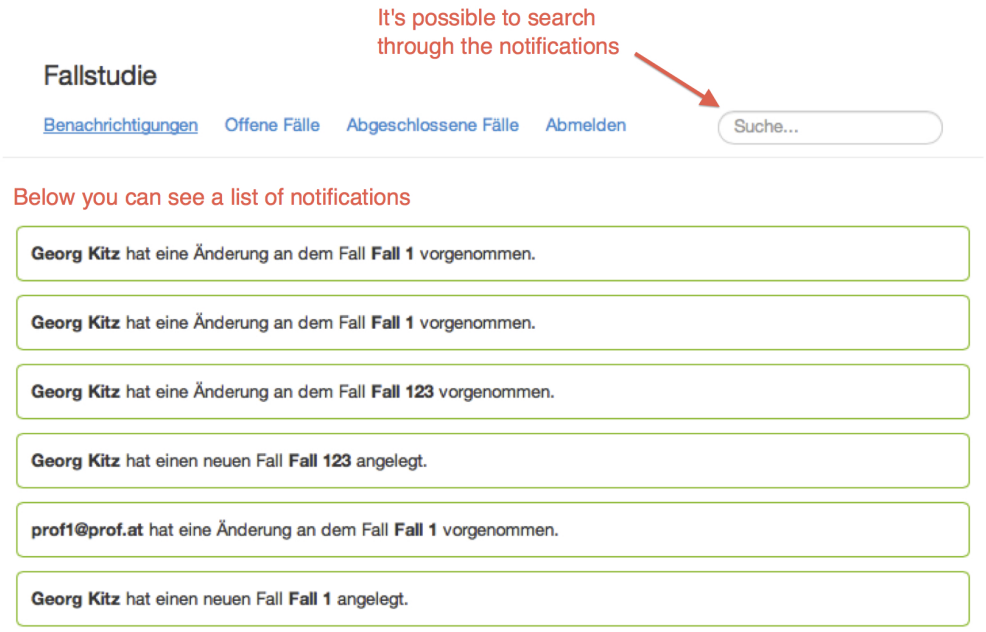


Figure 11. Representative view of the notification dashboard for supervisors.

Subsequently, they are provided with the possibility to comment on every field in the report, as it is an indispensable prerequisite of PBL to have the ability to comment on individual sections of a report and to push the learners in the right direction if they're off track. Figure 12 demonstrates what the reporting screen in the frontend looks like.

This gives the supervisors the flexibility to interact with their students during the whole process of working on a case and to give immediate feedback to a student's ideas and findings.

Figure 12. Representation of the Reporting screen for supervisors.

Same as the report editing option for the students, the supervisors have the possibility to browse through all the older versions of their comments. This gives them the possibility to overlook how the students make use of their instructions and comments when finding a solution. This plays a major role in the final assessment of a student's performance.

Further the implemented hierarchical comment feature, which is covered in more detail in Section 6.6., indicates which of the involved supervisors has already commented on a specific entry in the case.

It is based on a simple traffic light system in which the different lights indicate the individual hierarchical level of a specific supervisor, with green being the highest editor right, blue the medium and red the lowest. If one in a superior position has already answered a question and has given feedback, there is no possibility for one in an inferior position to note down his/her advice.

Figure 13 shows the traffic light system in use. In this case the supervisor in the highest position (green) and the one in the medium position (blue) both have left a comment meaning that there is no further supervision required. This also means that the supervisor with the lowest rank has automatically lost the right to comment.

Figure 13. Shows the traffic light system in work

Let's assume the supervisor with the medium rights (blue) is the first person to comment on a case. The traffic light system will display a blue light, showing the inferior supervisor that there is no possibility for him/her to comment on this specific entry anymore. The supervisor with the highest rights (green) on the contrary, is still able to edit or overrule the mid-level supervisor's suggestions.

This is version based, which means, as soon as the student adds a new version, all of the supervisors have the chance to give new feedback. However, if the supervisor with the highest hierarchic ranking is the first to comment on a case, the others lose their chance to do so. Moreover the traffic light system helps to keep track of sections which still require supervision. A section displaying a green light is consequently finished.

As identified before, one major aspect of the concept of Problem Based Learning is the continuous guidance of students and overlooking their progress. Our reporting system deals exactly with this aspect and enables the supervisors to keep track

of their supervisees, providing them with the flexibility to work on the reports and giving feedback whenever they find the time to. Besides all the other time consuming responsibilities that are associated with being a doctor at a hospital, this presents a priceless asset to ensuring smooth and regular interactions between students and supervisors. As a matter of fact, it makes reviewing the student reports more comprehensive and easier to deal with, as new reports or report versions are immediately updated to the supervisors side of the system and don't have to be handed in all at once, at the end of the month.

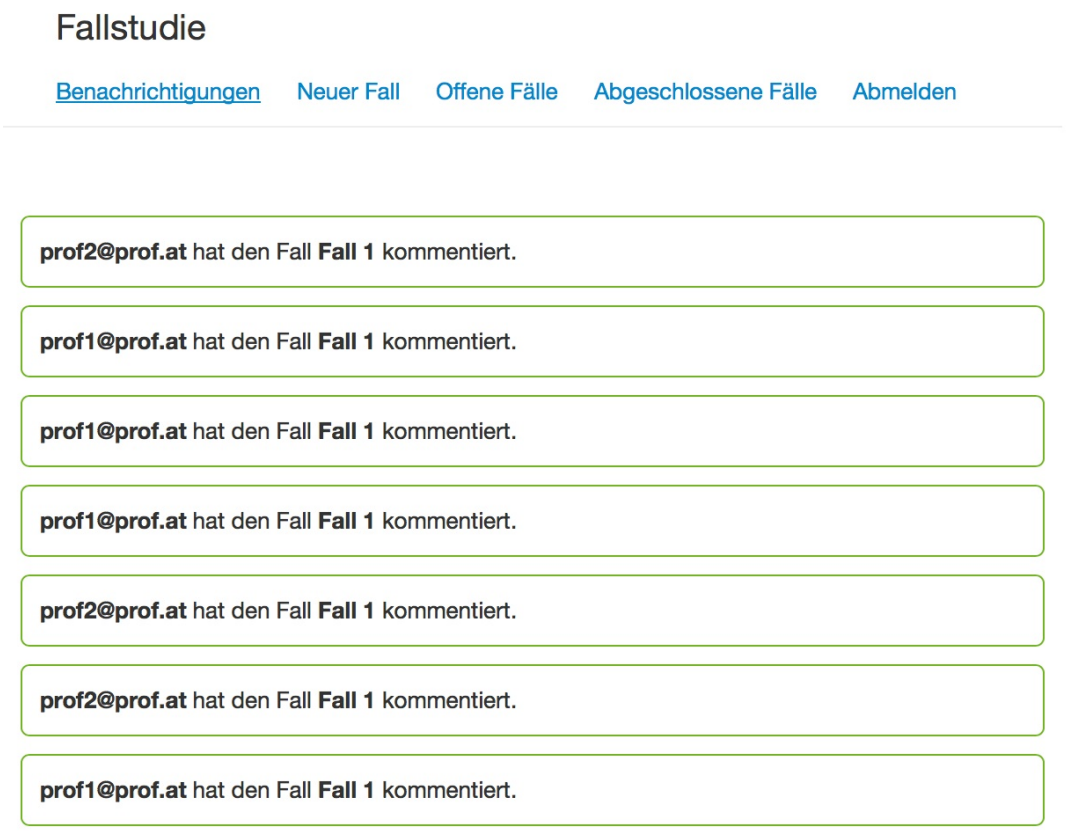


Figure 14. The student's notification dashboard.

As soon as a supervisor has added comments to a report, the student is able to make use of their own notification dashboard (Figure 14), which functions in the same way the notification service on the supervisor's side does and which makes navigating and searching for newly added things straight forward. The students can make use of the supervisor's suggestions and implement them directly on the iPad or revise their reports on their computer, using the web portal.

8. Evaluation

After building the final product, user experience tests were conducted, which should give some insights about the usability of the product and if it withstands a real-life proof.

8.1. Usability

This chapter is designated to explain why usability is important, how the requirements for program development have changed, why usability matters and how one can quantify the usability of a product.

8.1.1. A historically generated problem

Software engineers tend to focus on finishing the product and forget to see the overall picture, which includes a good user experience. This happens mostly because programmers and software engineers prioritise to solve a given problem.

When the first programs were written and people started to actually use them, this worked quite well because most of the users were experts themselves and had the same expectations and wishes like the developers who have written the program in the first place. There was simply no reason to learn how the regular enduser thinks, as this person was much like themselves. Further the wide spread philosophy was that people have to adopt their behaviour to conform with the software, rather than building a software which is easy to use. From a historical point of view, the reason for hiring software engineers has always been their technical expertise and problem solving skills. Consequently, in the early years of software engineering, there was simply no reason to question the developer's approach and so there was also no reason to simplify a system to make it fit for a non-expert user's needs. Although usability is a key factor in the field of software engineering nowadays, it seems rather understandable that it has not always been a prerequisite of software development. In fact, in the early generations it was quite difficult to write software. The main challenge was to get things running in the first place. However, over time programming languages got easier and easier so this excuse did not hold up anymore.

Later on, when computers became affordable, and software became available to a broader audience, which didn't necessarily have an engineering background or analytical way of thinking, software developers were faced with new challenges and a paradigm shift was introduced. It wasn't all about solving the problem anymore. People who write software started to think about usability and ways to simplify the use of software for the regular enduser. User experience evolved to a core task within the community and making the use of software an enjoyable experience for the user is gaining more and more significance in the way software is developed nowadays.

8.1.2. What exactly is Usability?

Basically usability is a measurement for the quality of the product. There are five key factors which influence the quality and ease-of-use for a specific product [Nielson, 2013a] [Rubin, 2008]:

- Learnability
  This questions wether it's easy for a user to accomplish a given task if he/she encounters it for the first time?

- Efficiency
  Once the user has learned how the design works, this questions whether it's easy to solve a given task or whether it is very time consuming and difficult (e.g. are there many steps involved or just a few)?

- Memorability
  This questions the reusability of a product after a certain time of user inactivity has occurred. This means whether a user is actually able to reuse a product easily, even after a long time of inactivity, or whether there is a new learning process involved every time they did not employ a product for a certain timespan.

- Errors
  This questions the intuitivity of a product, meaning the probability of a user to make errors while using the product. This also covers the degree of error they're facing and whether they are actually able to recover from it alone or whether they do need assistance.

- Satisfaction
  This questions whether a product provides a pleasant user experience or not.

- Utility
  This questions whether a product actually does what the user expects it to do. No one cares wether a product is easy to use if it isn't useful in the first place and vice versa.

Finally this can be broken down to the following statement:

A product needs to provide the features a user actually needs and the features must be easy to find and use!

8.1.3. Importance of Usability

As already pointed out in Section 8.1., the main target audience does not comprise of other technic-affine and software experienced persons anymore. Depending on the software product, the main user might be a regular office employee, banker, mum or in our case a (medical) student. Thus, when trying to sell software products, usability is nowadays a key factor which can't be ignored. If a product is hard to use or understand, very slow or does not behave like the user would expect, the user will stop using it, give bad reviews and search for alternatives [Rubin, 2008]. Obviously, unsatisfied users result in decreasing sale figures and a loss of money for the company. This isn't desirable as competition in the software development field is fierce. However, usability is not only important for selling a product to external users but also if a company builds a product for internal use. A study from the Nielson Norman Group [Nielson, 2013c] found out that a company should spend roughly ten percent of their design budget on usability testing as this in exchange doubles the quality of the end product. Although this means that a company needs to invest ten percent more when developing a product this pays off quickly because an easier to use end product decreases the time it takes for the employees to become familiar with it. This in turn saves the company money as their employees are able to employ the new product quickly.

8.1.4. Usability Testing

User-centered design describes techniques, processes and methods to design usable software with the user in the center of the system. Engineers shouldn't only think about the sheer purpose of a product and how they can build it easily. They also have to think about how the user experiences a product whilst using it. The main ingridient must be usability. The development always has to start with the user in the center of the process, with understanding who the user is and all other parts, like the design and development, should be built on top of that. As soon as a first working prototype is available it should be tested with a group of regular users. This is of crucial importance when wanting to understand how their behaviour correlates with the functionality of the offered product. After that the data of those tests needs to be analysed and all conceptual flaws which have been identified during the testing procedure should be examined and if necessary completely redesigned according to the user needs [Krahmer, 2004].

A very common way of performing usability tests is by employing the so-called *Thinking Aloud* testing method. Beside its use in software engineering it's also used in studying cognitive processes, in reading, writing or even problem solving. It represents a very basic but extremely useful way of getting a deeper understanding of how users make use of a product.
The *Thinking Aloud* testing method can be divided into three steps [Nielson, 2013b] [Nielson, 2013a]:

- Find representative users
  You need to set up a user group which comprises of real-life users of your product. For a product website this would include consumers who represent the target audience and for a company intern software solution, this would include the employees who are actually going to work with the end product.

- Perform representative tasks
  The users have to perform representative tasks during the test session. This means, that in order to get the most out of such a test, it is crucial that the user group is provided with detailed instructions and tasks which will also help to identify a product's usability. You certainly won't get any meaningful feedback data from randomly performed clicks by your user group. Conse-

quently, it should be well-defined what the users should accomplish in the frame of the test (e.g. create a user account for this website).

- Observe
  Observe what the tester is doing. Observe what problems pop up, what works well for the users and where you need to improve your design. During the whole test sessions the users are encouraged to express their thoughts and ideas about the product. The observer should note and/or record everything what the user says for later analysis.

It is important that the users are isolated from any disturbing interruptions while they are performing the test. This means that it's not allowed to give hints or suggestions to the user. Everything that might influence their thinking process or spoils their thinking strategy will inevitably affect the test results in a negative way. For instance, as soon as you tell the user to perform a specific task in a specific way, he/she will always keep this in the back of his/her mind because "if the last task worked like that, the next one could work in a similar way, so let's try to click here". Hence, the observer basically contaminated the test result.
A sensible group size for such a test is around five participants. In this case sensibel means a good balance between obtaining significant test results and the involved costs when carrying out such a test. Moreover, a reasonably sized group allows you to repeat a certain test case more often and also to test various iterations of the design.

8.2. Evaluation of the proposed system

In order to find out whether the created product is good, which besides fulfilling all its requirements also provides the user with a pleasant experience, the web app for supervisors and students as well as the student iPad app was tested.

Unfortunately, it has been impossible to find a supervisor willing to volunteer for testing the supervisor side of the system (see Section 9. for details). Consequently it was not possible for us to test the supervisor side of the system within in the frame of this thesis. However, luckily three medical students were found as participants, all of them in different stages of their studies, who agreed to test the student portals for us. One student had already passed the practical year

and thus knew how to fill in the old standardised paper forms, which is discussed in Section 3. This turned out to be quite beneficial, as he was able to compare the new system to the old-fashioned paper forms and thus contributed a different perspective.

8.3. The Evaluation Strategy

Five test tasks for the students were created in order to evaluate our system's usability.

- Task 1: Create a completely new case
  This task included the finding of the *create case* entry in the navigation, selecting a subject and filling in the according fields.

- Task 2: Change an existing entry of the case and save it as a new version
  This task included the selecting of a specific entry, clicking the *Editieren* button, changing the text and afterwards clicking on *Speichern*.

- Task 3: Upload an image for an entry
  Selecting an entry, clicking the upload image button, selecting an image from the hard drive and clicking the *Speichern* button.

- Task 4: Access a previous version of an entry
  This included the task to open an item, which already has a version history, and to find the history entry at the bottom of the item in order to select one of the history items.

- Task 5: Access the comment of an entry
  Find out which item of the case was commented and check what the supervisor has written. Due to a lack of supervisor volunteers the author & conductor of the thesis took this role for this test case and commented on an entry of the participant.

The same test szenarios were conducted for both portals, the web portal for students as well as the iPad version of the student portal. Since the test has been conducted within the frame of the so-called *Thinking Aloud* method, our student testers have

been encouraged to comment on their actions and to express their thoughts and critics during the whole duration of the test. In order to collect the test data for a later evaluation all the test sessions have been recorded and things have been noted like the year of study of the student and the time it took the student to complete every single task. After the participants had absolved their tests the recordings have been analyzed and results of the feedback, which will be discussed in the next section, evaluated.

8.3.1. Outcome of the Evaluation

The overall feedback has been merely positive and all students agreed that they prefer the new system over a paper form approach. All participants where able to solve the tasks without major problems and even more interestingly, besides the fact that one had the advantage of knowing the paper form procedure in more detail, they all needed approximately the same amount of time for finishing them.

Nonetheless, there has also been constructive feedback which helped to identify several improvement strategies for a possible successor version of the developed report system.

Points raised by the student testers regarding the web portal included the following:

- Uploading an image at the point of case creation

  Two of our testers were wondering why it is not possible to upload images to an entry at the actual point of case creation. Besides saving some time while using the portal, this also holds the benefit that one does not have to access the case again before being able to add images. Giving them the possibility to add images right at the point of case creation also holds the benefit that adding images won't create a new notification for the supervisors anymore, which at the moment is a quite unpleasant side affect of the system's image adding feature. Consequently, this small change would increase the overall performance of our system tremendously.

- Indicator lights

All of the participants noticed the traffic lights (Figure 16) but seemed to have troubles with understanding what those actually mean. As explained in Section 6.6. the main purpose of the traffic light system is to indicate wether a supervisor has already commented on an entry and which supervisor in the hierarchy has added an entry. After discussing this feature with the testers they've agreed that it is in fact quite handy and that it shouldn't be dismissed. Nonetheless, they have pointed out that it would be good to add something like a legend or a tooltip which breaks down its purpose for them.

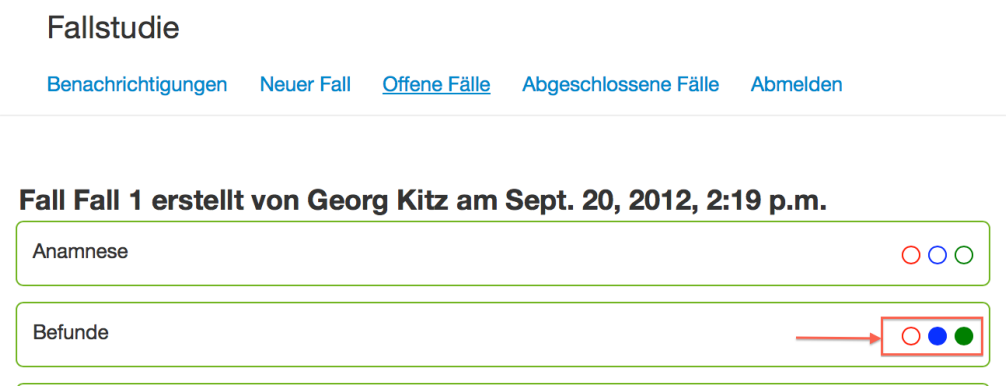


Figure 15. Shows the traffic light system, which would benefit form a legend which explains its purpose

- One entry should always be visible

  If you open a case, you see a listing of all possible entry fields and all of them are collapsed. One of our participants mentioned that it would be better to show the first field, with the actual entry area folded out. This is a better indicator that you can actually click all the entries and insert text. An even better idea is to show an animation when opening the portal for the first time. This gives the user more context about what is happening on the screen.

- Mark unprocessed entries

  Another point that has been raised is that it would be good to have some sort of indicator that reveals whether an entry has already been processed or whether it still requires processing. At the moment the user has to tap on every entry field to see if it already contains data or not. One solution might be to add the date and/or time when the user has last edited the entry right next to the entry field's name. Unprocessed entries would simply miss a timestamp and thus indicate that the field hasn't been processed yet.

- Drag & Drop Image Upload

  All of the users wanted to drag and drop images to upload them, as they knew this quite popular user paradigm from various well known websites. Being able to add multiple attachments at once is also a desirable addition for this feature.

With regards to the iPad version of our portal only one issue has been raised, which regards 2 small inconsistencies between both systems.

- Missing Traffic Light System

  There is one particular difference between the iOS app and the web app. The traffic lights, which serve as an indicator, are missing on the iOS version of the portal. As noted above, this traffic light system, makes it easier to see which entries have already been commented on and which supervisors have left a comment.

- History Button Inconsistency

  In contrast to the web portal, the iPad version provides the user with an *info*-button (see Figure16) to access the version history of a case file. The problem with this difference between the 2 systems is that you have to tap the *info*-button in the mobile portal at least once to know what happens. This is actually fine, as the user will memorise it quite quickly, but also a button named *History* could be used instead, which might be more intuitive because it has an intrinsic meaning.

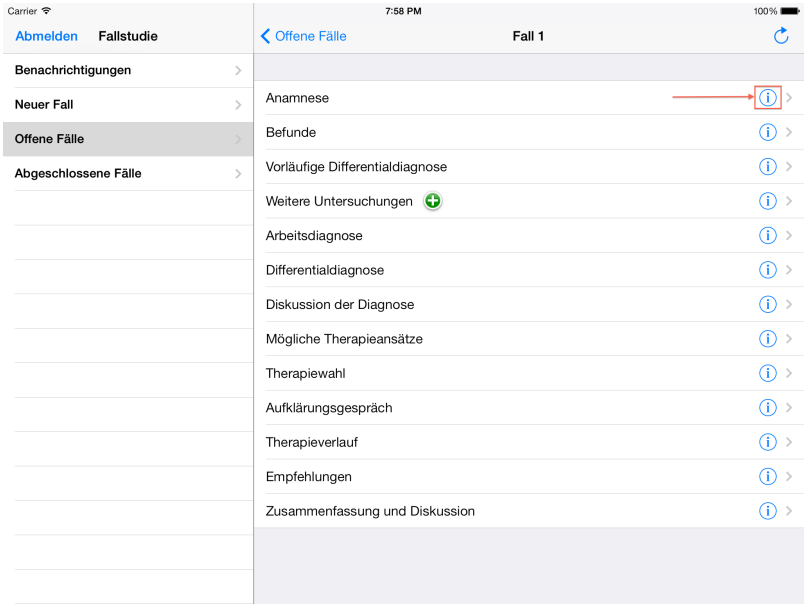


Figure 16. Undefined behaviour of the info button.

To sum up, the overall feedback from the students regarding the 2 portals was quite positive. All of them were able to fulfill all given tasks without any assistance. The problems and improvements which they've pointed out are very useful and worth considering, as they all demonstrate quite beneficial additions to our current solution and would certainly improve the overall performance of our system. Furthermore all students thought that the system is a quite handy alternative to the old paper approach and the student, who had already finished his practical year, was convinced that this would be a huge improvement for the practical year as a whole because *"it is a tremendous timesaver in comparison to the paper and pen approach and makes the duty of filling in case files quite enjoyable"*.

9. Encountered Problems

Several problems were encountered during the testing phase, which required some adaption to the evaluation approach. The first intension was to test the system within one term's time in some kind of combined "as you go" fieldwork-feedback approach. One user group, including students and their supervisors, was supposed to follow the old approach with pencil and paper and the other group was supposed to use our newly established system. The aim of this field test would have been to gather more information about how the system works in its dedicated real-life application and to compare it with the conventional method which is used at the moment.

Unfortunately it wasn't possible to perform this field test due to several reasons.

- Test Setup

  When dealing with patient data there are obviously very high security standards involved. Consequently, there had to be a server behind the firewall of the Medical University of Graz, which had to be configured in the way that it was working with the designed system (Python, Django and PostgresSQL). Since this intension had to be authorized by several authorities within the University the realisation took quite some time and resulted in a first setback of our progress, which made a one term lasting test of the system difficult.

- Partners

  Within the University, there was one dedicated correspondence partner which served as our central point of communication, including advice on all problems and questions and cooperative work during the test phase. Unfortunately, after the prototype was finalized and deployed on the test server, said partner dropped out of the project for unknown reasons and wasn't accessible for any further correspondence or help with the initial plan for the usability tests anymore.
  Without a project partner within the University it appeared to be rather difficult to get hold of any of the supervisors which are actively working with

the students during their practical year. Though understandable because of the quite significant workload they are facing, this was rather disappointing.

10. Conclusion

In the frame of this work a novel reporting system was developed, which is able to improve the PBL experience in the field of medical education tremendously. By providing the students and supervisors with platforms to store and process their work and by linking the systems with each other, using individual notification dashboards, it is possible to ensure a regular and easy interaction between supervisors and supervisees. The problem of taking notes while in the hospital and working on different cases with different patients, has been solved by using a flexible mobile device (iPad), which supplies the students with the flexibility they need to process and work several cases. This also holds the advantage that they don't have to carry around several paper report forms anymore. The supervisors are immediately updated about the progress of their supervisees via real-time notifications. The same applies to the comments a supervisor gives back to the students. This guarantees a continuous supervision and guidance of the students throughout their education at the hospital and ensures an extremely effective way of profiting of the sophisticated concept of PBL. The usability tests which have been performed showed that the system is usable and would be a huge benefit for the students, as compared to the current approach they are using. Although certain points of critics have been raised, none of those "flaws" need to be considered as critical as there has not been one single case were a student has not been be able to recover on his/her own. Nevertheless, for the future work on this system, the constructive feedback, provided by our testers needs to be taken very serious and the identified "flaws" need to be fixed as a next step. Also it would be good to be able to conduct another evaluation of the supervisor side of the portal in order to be able to capture the full usability of our system.

The message that I was able to take away with me whilst working on this thesis is that crossing the boundaries of different fields is an important step towards innovation and improvement of which all involved parties are able to profit. I am pretty sure that in the future, be it in academia, education or industry, interdisciplinarity will continue to establish and grow and that we will see a lot of similar applications, which are based on such an multifaceted approach. In the field of education, those applications have the potential to improve the work

with students and supervisors in many different ways. Nonetheless, in the end interdisciplinary project cooperations are dependent on a mutual effort and the willingness to make it work.

References

[Obj, 2013] (2013). About objective-c. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html. Accessed: 2013-04-16.

[aja, 2013] (2013). Ajax. http://www.w3schools.com/ajax/ajax_intro.asp. Accessed: 2013-12-19.

[cro, 2013] (2013). Cross site forgery. http://www.squarefree.com/securitytips/web-developers.html\#CSRF. Accessed: 2013-12-19.

[Dja, 2013] (2013). Django. http://www.djangoproject.org. Accessed: 2013-04-16.

[dja, 2013] (2013). Django csrf token. https://docs.djangoproject.com/en/dev/ref/contrib/csrf/. Accessed: 2013-12-19.

[htt, 2013] (2013). Http basic authentication. http://www.infosys.tuwien.ac.at/teaching/courses/WebEngineering/HTTP_Authentication.pdf. Accessed: 2013-12-19.

[ios, 2013a] (2013a). ios. http://www.apple.com/at/ios/. Accessed: 2013-12-19.

[ios, 2013b] (2013b). ios developer portal. https://developer.apple.com/devcenter/ios/index.action. Accessed: 2013-12-19.

[MVC, 2013] (2013). Model view controller concept. http://de.wikipedia.org/wiki/Model_View_Controller. Accessed: 2013-04-16.

[Pos, 2013] (2013). Postgresql. http://www.postgresql.org. Accessed: 2013-04-16.

[pyc, 2013] (2013). Pycharm. {http://www.jetbrains.com/pycharm/}. Accessed: 2013-12-19.

[Pyt, 2013] (2013). Python programming language. http://www.python.org. Accessed: 2013-04-16.

[Akinoglu, 2007] Akinoglu, O. (2007). The effects of problem-based active learning in science education on students' academic achievement, attitude and concept learning. *Eurasia Journal of Mathematics, Science & Technology Education*, 3:71–81.

[Albanes, 2000] Albanes, M. (2000). Problem based learning: why curricula are likely to show little effect on knowledge and clinical skills. *Med Educ*, 34:729–738.

[Allchin, 2013] Allchin, D. (2013). Problem– and case–based learning in science: An introduction to distinctions, values and outcomes. *CBE–Life Science Education*, 1.

[Banikowski, 1999] Banikowski, A. (1999). Strategies to enhance memory based on brain-research. *Focus on Exceptional Children*, 32(2).

[Barrows, 1980] Barrows, H. (1980). *Problem based learning: An approach to medical education*. Springer Publishing Company. ISBN 0826128416.

[Barrows, 1988] Barrows, H. (1988). *The tutorial process*. SIU School of Medicine. ISBN 0931369258.

[Barrows, 1994] Barrows, H. (1994). Practice-based learning: Problem based learning applied to medical education. *Biochemical Education*, 22:218.

[Barrows, 1996] Barrows, H. (1996). Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning*, 1996:3–12.

[Benware, 1984] Benware, C. (1984). Quality of learning with an active vs passive motivational set. *American Educational Research Journal*, 21(4):755 – 765.

[Bonwell, 1991] Bonwell, C. (1991). *Active Learning: Creating Excitement in the Classroom*. Jossey-Bass.

[D.Bedard, 2012] D.Bedard (2012). Problem-based and project-based learning in engineering and medicine: Determinants of students' engagement and persistance. *Interdisciplinary Journal of Problem-based Learning*, 6:8–30.

[Ebner, 2013] Ebner, M. (2013). "architecture students hate twitter and love dropbox" ordoes the field of study correlates with web 2.0 behavior? *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 43–53.

[Feletti, 1998] Feletti, B. . (1998). *The challenge of problem-based learning*. Routledge. ISBN 0749425601.

[Fielding, 2002] Fielding, R. T. (2002). Principled design of the modern web architecture. *University of California*, 2:115–150.

[Grabinger, 2002] Grabinger, S. (2002). Problem-based learning as an example of active learning and student engagement. *Lecture Notes in Computer Science Volume*, 2457:375–384.

[Guersul, 2009] Guersul, F. (2009). The effects of online and face to face problem based learning environments in mathematics education on student's academic achievment. *Procedia Social and Behavioral Sciences*, 3:71–81.

[Haidet, 2011] Haidet, P. (2011). Team-based learning. *The Journal of the International Association of Medical Science Educators*, 4(21):385–387.

[Herr, 2008] Herr, N. (2008). *The Sourcebook for Teaching Science*. Jossey-Bass.

[Hills, 2001] Hills, H. (2001). *Team-based Learning*. Gower Publishing Ltd.

[Hosseini, 2010] Hosseini, M. B. (2010). Life-long learners through problem-based and self directed learning. *Procedia Computer Science*, 3:1446–1453.

[Hrynchak, 2012] Hrynchak, P. (2012). The educational theory basis of team-based learning. *Medical Teacher*, 1(34):796–801.

[Krahmer, 2004] Krahmer, E. (2004). Thinking about thinking aloud: A comparison of two verbal protocols for usability testing. *IEEE Transactions on professional communication*, 1(2):105–117.

[Lacuesta, 2009] Lacuesta, R. (2009). Active learning through problem based learning methodology in engineering education. *Frontiers in Education Conference, 2009. FIE '09. 39th IEEE*, 3:1–6.

[Larmer, 2013] Larmer, J. (2013). Project based learning vs. problem based learning vs. xbl. http://biepbl.blogspot.ch/2013/11/problem-based-vs-project-based-learning.html. Accessed: 2013-12-11.

[Michaelsen, 2002] Michaelsen, L. (2002). *Team-based Learning: A Transformative Use of Small Groups*. Praeger Publishers.

[Michaelsen, 2007] Michaelsen, L. (2007). *Team-Based Learning for Health Professions Education: A Guide to Using Small Groups for Improving Learning*. Stylus Publishing.

[Myers, 2012] Myers, G. (2012). *The art of software testing*. Wiley Publishing Inc., 3 edition.

[Nielson, 2013a] Nielson, J. (2013a). Introduction to usability. http://www.nngroup.com/articles/usability-101-introduction-to-usability/. Accessed: 2013-12-08.

[Nielson, 2013b] Nielson, J. (2013b). Recruiting test participants for usability studies. http://www.nngroup.com/articles/recruiting-test-participants-for-usability-studies/. Accessed: 2013-12-08.

[Nielson, 2013c] Nielson, J. (2013c). Usability roi declining, but still strong. http://www.nngroup.com/articles/usability-roi-declining-but-still-strong/. Accessed: 2013-12-08.

[Pereira, 2012] Pereira, L. (2012). Traditional learning and problem-based learning: self-perception of preparedness for internship. *Rev Assoc Med Bras*, 58:594–599.

[Rubin, 2008] Rubin, J. (2008). *Hanbook of Usability*. Wiley Publishing Inc., 2 edition.

[Savery, 2006] Savery, J. (2006). Overview of problem-based learning:definitions and distinctions. *Interdisciplinary Journal of Problem-based Learning*, 1:10–20.

[Smilovitz, 1996] Smilovitz, R. (1996). *If not now when?: Education not schooling*. Morris Publishing. ISBN 1575020149.

[Srinivasan, 2007] Srinivasan, M. (2007). Comparing problem-based learning with case-based learning: Effects of a major curricular shift at two institutions. *Academic Medicine*, 82(1):74–82.

[Stanford, 1994] Stanford, U. (1994). Teaching with case studies. *Stanford University Newsletter*, 5(2).

[Turan, 2009] Turan, S. (2009). Evaluating the role of tutors in problem-based learning sessions. *Procedia Social and Behavioral Sciences*, 1:5–8.

[van Rossum, 2009] van Rossum, G. (2009). From abc to python. http://python-history.blogspot.co.at/2009/02/early-language-design-and-development.html. Accessed: 2014-08-01.