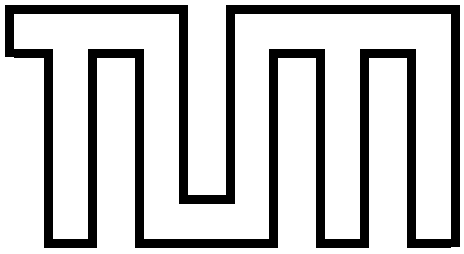


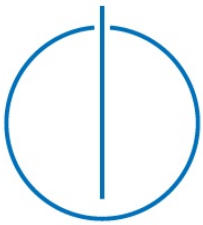
Technische Universität München

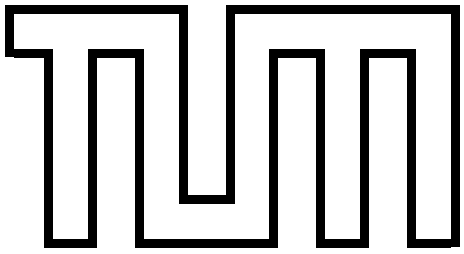
Fakultät für Informatik

Master's Thesis in Wirtschaftsinformatik

Entwurf einer Echtzeit-Web-Anwendung für kollaborative Anmerkungen in der Telemedizin

Junus Ergin





Technische Universität München

Fakultät für Informatik

Master's Thesis in Wirtschaftsinformatik

Design of a Real-Time Web Application for collaborative annotations in telemedicine

Entwurf einer Echtzeit-Web-Anwendung für kollaborative Anmerkungen in der Telemedizin

Autor: Junus Ergin

Supervisor: Prof. Dr. rer. nat. Florian Matthes

Advisor: Adrian Hernandez-Mendez

Submission: 15.11.2016

I assure the single handed composition of this master's thesis only supported by declared resources.

München, 15.11.2016

(Junus Ergin)

Acknowledgment

First of all, I want to thank Prof. Dr. Florian Matthes for giving me the opportunity to write my Master's Thesis at the chair of Software Engineering for Business Information Systems (sebis). During my studies the chair deepened my expertise on developing applications on a professional level and gave me a lot of extra support. Furthermore, I express my deep gratitude to Mr. Adrian Hernandez-Mendez who really improved my expertise in producing high quality academic work.

In addition, I want to thank Patrick Palacin for providing me the possibility to work in the field of telemedicine. Thank you for inspiring me with your great knowledge of web technologies.

Apart from that, I would like to thank my parents Karin and Mehmet and my girlfriend Sarah for motivating me during the last half year and helping me to keep on working on my thesis.

Abstract

Design of a Real-Time Web Application for collaborative annotations in telemedicine. Functional and non-functional requirements were derived from scenarios which describe a common usage of the tool and also from the technical and legal environment. The requirements and the tool design was validated using a survey. Within the survey, 22 medical experts which at least three months of experience in tele-consultation were asked. Afterwards, the final app was validated using a "field test", including real doctors and real users in order to proof its applicability under real conditions.

Design einer Echtzeit-Webapplikation für gemeinsames annotieren verschiedener Dateien in der Telemedizin. Funktionale und nicht-funktionale Anforderungen wurden von Szenarien abgeleitet, die den Einsatz des Tools beschreiben sollten. Weiterhin wurden Anforderungen von der technischen und rechtlichen Umgebung abgeleitet. Die Wichtigkeit der Anforderungen und ein angemessenes Design der Applikation wurden mit einer Experten-Umfrage ermittelt. Alle befragten Experten haben eine medizinische Ausbildung und mindestens 3 Monate Erfahrung im Bereich der Tele-Konsultation. Anschließend wurde die Applikation mit einem "Field-Test" mit real praktizierenden Ärzten und Benutzern validiert und unter tatsächlichen Bedingungen in Bezug auf ihre Anwendbarkeit validiert.

Contents

Chapter 1

Introduction

1.1 Motivation

During the last decades, the bandwidth of private internet accesses has been increased continuously [Coff 02] [Bund 15]. Due to software as Skype, Google Hangouts, Netflix and Amazon Instant Video, video streaming and conversations already became usual for the typical internet user [Bund 16].
Even the elaboration of collaborative solutions over the internet became widespread in many domains due to software as Google Docs and Microsoft SharePoint [Supp 16] [Goog 16]. Fast internet connections and smart solutions for transmitting data enable users to notice the collaboration as real time.
In order of finding collaborative solutions and editing files together over the internet, the annotation of different files as images becomes more important in businesses.

In different domains of medicine, e.g. in radiology, video consultation has already become widespread[Desh 00]. In consultation for instance, whiteboards and annotations on images are used to enhance the communication between patients and doctors. In order of enhancing the video consultation, it is mandatory to make use of different collaborative consultation tools as a digital annotation tool.
If a patient is in a video consultation with a doctor it might be important for him to upload an image and collaboratively annotate it with the doctor over the internet. That supports the patient and the doctor finding a solution for the patient's problem.

Furthermore, users prefer running software within the environment of a web browser (SAAS). Downloading and installing software only is necessary if performance issues occur in the web browser or if the content should be available offline. Hence, in the case of a medical video consultation it is not necessary to make a consultation software available offline.

A frequently used method of transmitting video streams is the WebRTC protocol. **W**eb **R**eal **T**ime **C**ommunication is an API, providing protocols for real time communication between many clients using a webserver [John 12].

Through the enhancement of telemedicine, doctors and patients save a lot of time and money. Waiting time, driving time and unnecessary visits of a doctor may be reduced. Therefore, all components of the health system will be improved.

In general, video consultation in the field of medicine is not common so far. The development of supporting collaboration tools, as a digital annotation tools, which supports real-time collaboration should be based on scientifically proven requirements to ensure quality, safety and performance. A digital White Board should help users to handle and collaborative edit different streams, as notes, webcam videos, annotations to images and much more.

1.2 Research objectives

The research objectives of this thesis are to determine the state of the art and requirements of a collaborative web-based digital annotation tool in the field of telemedicine. The requirements will be derived from scenarios which explain a typical usage of the tool within an existing environment of a company which offers telemedical consultation. The tool should be applicable in the field of telemedicine so that a doctor can collaboratively annotate a file with another participator, as a patient, a nurse or a doctor, together.

Within the given environment all participants share a workspace, which is a surface that allows the doctor to share different contents and annotate it with different information. The annotation tool should be designed to support the process of telemedical consultation. For this, medical experts who have experience in telemedicine will be interviewed by using a survey.

The discovered requirements will be validated through a prototypic implementation of an annotation tool. The design of that tool will be derived from the understanding of applicability by the doctors. The design guidelines of the tool will be derived from the conducted survey, as well.

Furthermore, several concepts of implementing such a tool and architectural solutions which help the tool fitting into the given environment will be elaborated.

Later, the annotation tools applicability will be tested within a real environment by genuine users and medical experts.

1.3 Proceeding and research questions

During this thesis, the following research questions will be answered.

1.3.1 RQ1: What is the state of the art for real-time collaborative web based annotation tools?

This question will be answered using a literature research. Different papers concerning web based collaboration tools will be analyzed. In this context utilized implementation technologies will be investigated. In addition, it will be analyzed which of the researched solutions works web based only. Furthermore, the discovered technologies will be analyzed in terms of the scope of their application and functionality.

1.3.2 RQ2: What are the requirements for web based component design based on the given scenarios in order to create real-time annotations over the internet?

Requirements will be derived from given scenarios which explain a typical usage of the tool. The scenarios are derived from an expected usage of the tool within the companies' environment and the expectations of doctors. Furthermore, the requirements will be validated by experts who will use the tool during tele-consultations in the future using a survey.

1.3.3 RQ3: How can the tool be designed to be understandable, and fulfill all functional and non-functional requirements?

To obtain a better understanding of the doctor's needs, a survey will be conducted to evaluate their design wishes and the importance of the functional requirements. Based on the results and on nun-functional requirements, a mock-up will be created. Afterwards, design concepts will be elaborated to fulfill all requirements and the experts' preferences from the survey.

1.3.4 RQ4: How far does the implemented prototype satisfy the requirements derived in RQ2 und how is an additional benefit guaranteed against other typical solutions?

For validation purposes, a prototypic annotation tool will be implemented as a typical collaborative tool for telemedical support. The tool will be deployed with web based technologies only.

The annotation tool will be validated against the discovered requirements from RQ2, letting doctors use the implemented annotation tool in the scope of telemedical consultation. Afterwards it will be determined whether the implemented prototype provides additional benefits against common ways of annotating data during telemedical consultation.

1.4 User process of teleconsultation

In the following section a typical user story of using an annotation tool will be explained. The user story will explain the environment scope and will also focus the challenges which are given by the business environment.

As mentioned before, the annotation tool which should be designed within this master's thesis should run within given constraints of a business environment. In that case, the environment is a web based consultation tool for telemedicine which allows a patient to video call medical employees. The patient will receive medical advice from nurses, general doctors and experts over video call in real time to his device.

1.4.1 User Process in the Existing System

Let's assume a patient who already signed up to the system. He is currently traveling in India, having a broken hand. A local doctor took an X-Ray image of his hand and the patient copied it on his laptop. Now he wants a second opinion from a German expert.

The patient signs in to www.teleclinic.com and simply clicks on a button to start a video call. He gets connected to a random nurse immediately. The nurse interviews the patient and determines to which doctor he needs to talk. Afterwards, a doctor gets invited to conversation and continues the process of medical advice.

During this process a medical case gets created, which contains all information about the patient suffers and the doctor's advice. The case gets created by the nurse, who already

fills the form with information she receives during the patient interview. After the doctor is invited, he also has the possibility to edit the form.

During this process, the patient or the doctor have the possibility to upload all kind of files regarding the current case. The patient decides to upload a file, as e.g. the X-Ray image of his hand he recently received from his doctor in India to share it with his German doctor.

The patient, the doctor and the nurse now can discuss the uploaded image and find a solution for the patients' problem.

1.4.2 Use of an embedded Annotation tool

Discussing an image can be quite hard if there is voice only. Imagine, the patient wants to highlight a section where he feels pain. Furthermore, the nurse might annotate a section of the uploaded X-Ray image to explain the patients suffer to the doctor.

To annotate the uploaded image, all conversation participants shall get the possibility to click on an annotation button. Now, a tool bar opens within the browser, offering several tools to the participants of the conversation. If a participant start to annotate the image, every participant of the conversation sees the annotations immediately. Now the doctor can see, where exactly the patient feels the pain in his hand. Also, the doctor can add more annotations or edit the patient's annotations.

Due to the annotations of the uploaded file, the communication between the doctor and the patient gets enhanced. The description of the problem from the patient's side, the explanation for the doctor from the nurse, and the anamnesis itself will be much easier to conduct an annotation tool.

Let's apply the new possibility of a supporting annotation tool to current scenario where a patient is traveling in India, having a broken hand. He already obtained a digital X-ray image of his hand from a doctor in India. Now, he is in a web meeting with a nurse and tries to explain his suffer. To make it more clear, he uploads his x-ray image during the call and annotates it. He creates a circle around a certain part of his trigger finger and explains what the problem is. The nurse recognizes that it could be a finger fracture and writes the text besides the drawn circle of the patient. Afterwards, the nurse invites a general doctor to the call, to create an anamnesis.

Now, the doctor sees what the patients suffer is within seconds. The doctor quickly realizes that a hairline crack appeared in the finger and the patient just should protect his hand for a few weeks. He can reannotate the X-ray image and edit the annotations of the others to show the patient exactly where the hairline crack is.

1.5 Explaining terms: Annotation and Real-Time

There are some main terms which should be clarified to prevent confusions. An annotation can be any kind of information and will be defined to fulfill the domain of this thesis. The term "Real-Time" also should be defined to fulfill the objectives of this thesis.

1.5.1 Annotation

An annotation is any kind of note which adds additional content to existing content [Webs 16]. This could be a graphical object which can be added to existing graphical content, e.g. a line or text. This is used for highlighting or labeling objects on graphical content [Russ 08]. The main purpose of an annotation is to add additional information to graphical content [Sand 16].

Graphical content could be anything that is displayed on a webpage, as e.g. images, PDF files or a video. In simple terms, it means that additional information, as text, lines or shapes can be added to graphical content or certain section can be highlighted.

Generally, an annotation is every kind of element which adds additional information to existing information. In this thesis, an annotation will only be the addition of graphical elements, text, or highlights to existing graphical content. Commenting source code, word files, audio, etc. will be ignored within this thesis.

1.5.2 Real-Time synchronization

"Real-Time" means that a client received actions within a predefined timespan Δ [Scho 06] [Dude 16]. Synchronization means that the whole dataset is always equal on all clients [Kope 87] [Piko 03].

Regarding this thesis all annotations should be synchronized within all clients who have a file opened in Real-Time. That means, that after a passed time Δ all annotations should be the equal on all devices. The Δ should be depending from processing time and the time it takes to transmit data between two clients.

Within this thesis, a protocol will be derived which enables the real-time synchronization of annotations between all clients which currently have a certain file opened.

1.6 State of the Art

Real-time annotation in web based tools already became widespread. Several different aspects of real time annotation are relevant within this thesis. The relation of them is displayed in figure 1.1. In the following part, the state of the art will be described with regard to related work and common technologies in the field of real time and annotations.

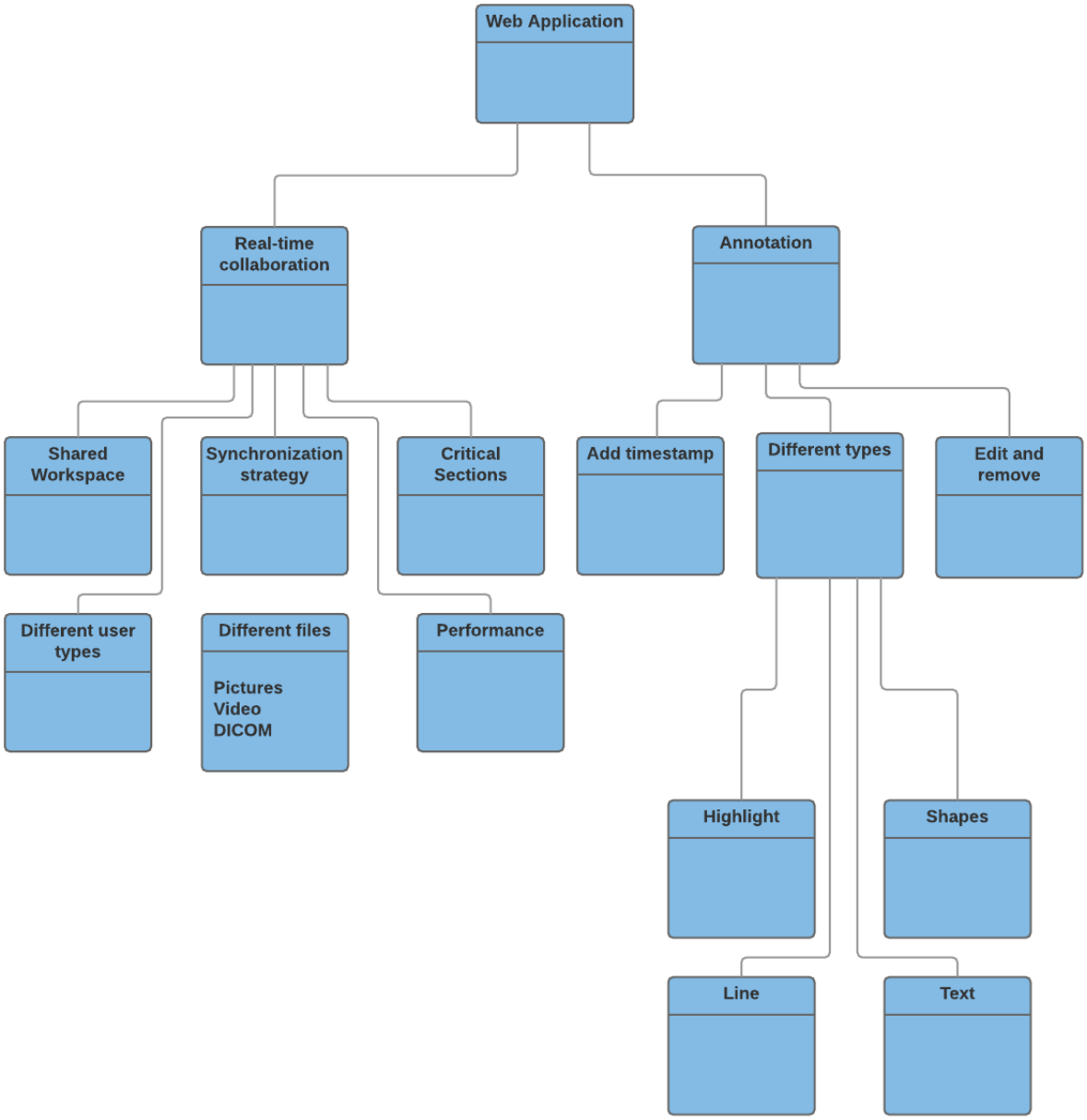


Figure 1.1: Aspects that will be investigated

1.6.1 Annotations

Adding annotations to different contents within the browsers environment already became widespread. Some of the hugest internet platforms, as e.g. YouTube and Snapchat allow annotating complex media as e.g. Videos [YouT 16] [Snap 16]. The following papers show the current research of adding annotations to different media. Afterwards, the similarities, differences and applicability to this thesis will be analyzed.

1.6.1.1 Add, edit and remove different annotation types

The most popular platforms who offer the annotation with different types are YouTube and Snapchat. YouTube provides the possibility of adding speech bubbles, spotlights, notes, titles and labels to a video. [YouT 16] Snapchat provides the functionality of adding text and several shapes and icons to a picture or a video. [Snap 16].

Derntl et. al. proposed a framework for real-time meta modelling in their paper "Echtzeitmetamodellierung im Web-Browser". The model can be collaboratively created within the web browsers environment. The framework is based on modern technologies, as HTML5, JavaScript and CSS3 and concepts, as e.g. the synchronization strategies, can be similar to this thesis [Mich 14].

1.6.1.2 Add timestamp

In the paper "Correlative multi-label video annotation" Qi et. al. describe a framework for the automatic annotation of video. For this, the timestamp of the video is relevant and has to be added to each annotation. As annotations also have to be included to videos in dependency of the video timestamp, this procedure is highly relevant for this thesis [Qi].

Zhai et. al. released the paper "eSports: collaborative and synchronous video annotation system in grid computing environment" which explains the real time collaboration on videos. They propose a framework for educational purposes which lets the user annotate videos and video streams in real time. They try to synchronize everything, including the exact timestamp of the video which is currently playing in order to support tele-coaching and tele-education.

As the annotation tool of this thesis also will be used to annotate videos in real time, the findings of both papers helps to support the strategy which will be used within this paper in order to synchronize the annotation of videos.

1.6.2 Real-time collaboration

There are also several tools available who already allow the synchronization of collaborative work in real-time.

1.6.2.1 Shared Workspace

The most famous tools are the four Google tools which provide typical office software synchronized in real time: Google Docs, Google Sheets, Google Slides and Google Forms [Goog 16]. The functionalities cover almost everything which is commonly used in Microsoft Office [Offi 16c], Open Office [Offi 16a] or Libre Office [Offi 16b].

Another example is Microsoft OneDrive and Microsoft SharePoint which allows the user to collaborate in Microsoft Office products, as Word and Excel, in real time [Supp 16].

Software as Google Docs and Microsoft Office is only designed to fulfill the requirements of office work. They are not designed to support doctors during teleconsultation. Furthermore, the tool cannot be used as a library inside of an existing software and can be connected to a custom database. Annotating more complex files, as videos or medical images is also not possible. However, the synchronization techniques and strategy of locking annotations which are currently annotated are very supportive for real-time annotations in telemedical consultation.

The patent “Real-time collaboration center” by Mark A. Kelley and Michael S. Wengrovitz describes a platform for realizing a call center between an end-user and an employee of the call center. Besides the call, the platform provides instant messaging, desktop- and media-sharing. Data has to be kept synchronized in real-time between the user and the employee [Kell 07].

Within this thesis, also different data streams are used at the same time: Video, audio and the synchronization of annotations. Very similar to Kelley et. al. different data has to be synchronized in real-time. The patent of Kelley et. al. is not concerned with annotations and there is no relation to medical consultation. Only general consultation of call centers gets investigated.

1.6.2.2 Synchronization strategy

The book “Synchronization in real-time systems: a priority inheritance approach” by Ragunathan Rajkuma and his paper “Priority inheritance protocols: an approach to real-time synchronization” introduces an approach of synchronizing data between different clients. Furthermore, he introduces a way to deal with different real time system which

patricianly share the same data. For this, critical sections have to be locked so that only one person at the time can write on it. The access rights for critical sections are based on a priority which gets assigned to the different clients who could operate on a critical section [Rajk 12] [Sha 02].

The approach could be interesting for this thesis. Different user types, as patients, nurses and doctors exists within the scope of this thesis. Applying different rights of editing critical sections can be considered as synchronization strategy, as well.

In the paper "A Web Services Framework for Collaboration and Audio/Videoconferencing", Fox, Wu, Uyar, Bulut present an alternative framework for WebRTC to manage collaborative work and audio-/ videoconferences [Fox 15].
Unfortunately, this work does not provide a web browser technology. Furthermore, the paper is older than 14 years and the framework does not fulfill prevailing standards. Also, WebRTC might not be the best protocol for synchronizing data.
The paper is interesting as a comparison to modern frameworks and to analyze how the main problems in the past were managed, e.g. the problems of bandwidth-lacks in connection with slow internet connections. This problem was an important issue 14 years ago and is now up-to-date again concerning mobile devices.

1.6.2.3 Critical sections

Different people might annotate the same image at exactly the same time. In these situations, data consistency always has to be guaranteed. If two operations on the same section are sent to the database, no one should lose any data while synchronizing with the other clients. For this reason, critical sections have to be locked.

The article "Transactional memory: architectural support for lock-free data structures" by Maurice Herlihy and J. Eliot B. Moss shows that locking critical paths was already highly investigated before annotating content became widespread. He explains that a section has to locked if one thread is currently writing data to this section. The writing access will be denied to other threads while the resource is locked. He especially focusses on the case what happens if a user thread disappears while the section is locked by him. [Herl 93]
During the architecture design of Unix operating systems the problem of mutual exclusion and the access control of critical resources already delivered standard solutions. Solutions as "Semaphores" are commonly used to solve these problems in operating systems and allow the user to define how often a resource can be accessed at the same time {Leung90 [Teva 87] [Maek 85].

The problem of mutual exclusion occurs in the same manner if annotations should be stored in a database. The explained concepts can also be applied for synchronizing notations in only one database within many clients.

1.6.2.4 Different user types

Tiros Chen and Richard Lai, Andy Chen applied the patent "System and method for setting user-right, and recording medium". They propose a system for creating different account groups. Every account can be assigned to one account group. Based on the group the users have the right to access different files or sections of a software. The group is based on the organization the user is working for and their job [Chen 05].

In the paper "End-user controlled group formation and access rights management in a shared workspace system", Haake, et. al. are concerned with forming flexible access right groups for shared workspaces. A user can grant access to others, form groups or grant special rights [Haak 04].

Both systems are similar to the access system which is used within the environment in which the annotation tool is designed. While editing annotations, the user should only be able to add annotations if he is part of a group who can access the file. If the user is a doctor, he should be able to delete annotations which were made by a nurse, but not vice versa. This behavior can be realized very easy using different access groups.

1.6.2.5 Different file types

Carl Vondrick and Deva Ramanan released the paper "Video annotation and tracking with active learning" and proposed a framework which just annotates certain frames of a video track. The purpose of the paper is to minimize performance problems [Vond 11].

Vondrick, Patterson and Ramanan released an article called "Efficiently Scaling up Crowdsourced Video Annotation". The article is based on their paper from 2011 [Vond 11]. In this article they provide a framework for annotating videos while dealing with huge file sizes, complexity and costs. They use their framework together with experts who label certain parts of videos as only experts can do it [Vond 13].

Within this thesis the annotation tool is also used in order to annotate different content with experts. Unlike to the proposed paper of Vondrick et. al. this thesis has only medical staff as experts. Also, the content which will be annotated can be everything and not only videos. Hence, within this thesis the functional requirements and the domain has to be designed in order to fulfill the expert's expectations in order to support their consultation as good as possible.

1.6.2.6 Performance

Holzer introduces in his thesis "Konzeption und Realisierung eines Frameworks für verteiltes Rechnen in Webbrowsern mittels WebRTC" a framework for the distribution of performance intensive algorithms within different browsers using WebRTC [Holz 14].
In general, performance intensive processes can appear during the deployment of a telemedical consultation tool. Bearing in mind performance optimization, it is mandatory to verify the possibility of parallelization for complex algorithms. The framework developed by Holzer may provide a way to distribute such operations within many clients.

In the paper "Live mobile collaboration for video production: design, guidelines, and requirements", Marco de Sá, David A. Shamma, Elizabeth F. Churchill are concerned with design-rules which must be maintained during collaborative work and mobile video production. They introduce a system that allows users of mobile end devices to follow and switch between multiple streams concurrently on any mobile device. Also the users can cooperate with other users making use of that system [S 13].
This paper focuses on performant real time behavior and the construction of guidelines to reach to wanted real time behavior. Also, these guidelines could be important for the development of a consultation tool in the field of telemedicine.

Rodríguez, Cerviño, Trajkovska and Salvachúa are introducing the paper "Advanced Videoconferencing based on WebRTC". The authors are engaged with the lacks which occur in video conferences using WebRTC [Prez 12].
Handling multiple video and data streams for video consultation and at least one consultation tool as an annotation tool has to deal with lacks as well, especially on mobile devices, with slow internet connections. The paper gives an approach how to deal with lacks and presented techniques which could be adapted to handle multiple video streams.

Sanghoon Sull, Hyeokman Kim, Yeon-Seok Seong, Michael Rostoker, Jung Kim published the patent "Techniques for navigating multiple video streams" reflecting techniques to navigate between various incoming video streams. In addition, they give little previews as thumbnails which will be generated automatically [Sull 06].
The idea of generating previews for other streams could be interesting as well in the field of telemedicine since the handling of multiple incoming streams (e.g. webcam, audio, at least one other stream by a consultation tool) has to be handled somehow.

1.6.3 Conclusion

On the one hand, several research exists about the way of annotating content. On the other hand, a lot of research regarding real-time synchronization was conducted during the last years.

Even if some papers are already concerned with the synchronization of annotations in real-time, there is no one which considers the relation to medical consultation during video calls. Also, most of the solutions are not web based and have to be applied for web usage if they are used.

In order to ensure the maximum support of teleconsultation, experts in the field have to be interviewed in order to determine what kind of requirements they have. The process of annotating different content has to be optimized for a maximum support of medical teleconsultation. The goal of this thesis will be to figure out what are companies and medical experts requirements on annotating content in teleconsultation, how they value it and how an annotation tool can be designed in order to fulfill the medical expert's expectations.

Chapter 2

Scenarios

In the following chapter different scenarios of using the annotation tool will be described. The scenarios represent typical situations for which the annotation tool should be used and will have the focus on the following aspects:

- Different roles of the people who participate during the annotation process
- The process itself, describing what the user does in the following scenario
- The used technical environment
- The technical architecture, describing the data flow of the annotation process

2.1 Subprocess of a video call

The annotations are supposed to be conducted during a video consultation. For this, a video call between a patient and an expert have to be established. The process of establishing a video call is already given in the software for which the annotation tool will be designed [Tele 16].

However, the process of establishing a video call will be explained in the following from a technical perspective to prevent confusions.

1. User logs in to a client system
   This could be an app for IOS, Android or the Web. A login works very easy by entering user credentials or signing in with Facebook as on all common platforms.
2. Starting a video call
   For starting a video call, the user has to press one button. Afterwards, the process

of establishing a video call automatically starts. While the user is waiting, a post request gets sent to the server.

3. Receiving post request
   After the server receives a new video call post, he creates a new entry in the database. Meta data about the call as creator, receiver, date etc. gets stored. Parallel, the server sends a push notification via WebSocket to the client who gets called. The client is a random employee of the medical staff who is online at the moment.

4. Accept video call
   After the client receives a notification from the WebSocket, the user on the called side has the possibility to accept the video call. After he accepts the video call by pressing a button, a put operation gets sent to the server. The video call gets updated in the database.

5. Establishing a call
   All clients are registered at a Twilio which is a company who provides an API for establishing video calls. After the client accepts the video call, he sends a second push notification to the Twilio API. Twilio now establishes a peer-to-peer connection between both clients which is based on the WebRTC protocol [Twil 16]. Now, both clients are in a video call and can easily chat about the patients' medical problems.

This process always gets executed in the beginning and will be assumed as a standard process. This process always starts before an annotation session starts. In the following scenarios, the process will just be described as "establish video call", as it is not part of the main annotating process and just a perquisite.

2.2 Scenario 1

The first scenario was extracted from a typical user flow and from expectations that doctors have to the system. The roles and the processes are based on common processes of medical teleconsultation at TeleClinic. Also, the described process is influenced by the wishes of a doctor to share and annotate an image.

2.2.1 Roles

- A patient who plans to talk to a doctor
  Usually, a patient has a suffer he wants to discuss with a doctor. He signs up to the system, explores it and finally decides to start a video call.

- A doctor who video calls the patient
  Most of the doctors are not online the whole time. If a patient created an appointment with the doctor, he logs in just before the appointment starts. Then, he can video call the patient or vice versa.

2.2.2 Process

In order to talk to a doctor about a certain disease, a user took a simple JPEG image of his skin rash, using his smart phone. Now he logs in to the medical portal of TeleClinic, using the Chrome browser. He doesn't want to call a doctor so far because he doesn't want to waste the doctor's time. First, he wants to provide collected information to his skin rash. In order to do so, he uploads the picture of his skin rash and opens it online with the annotation tool, which is provided on the page he just logged in to.

Now, he annotates the uploaded image, using tools such as a pen, a highlighter, and a text field, which are provided by the annotation tool. He circles some regions where he feels some itchiness and highlights a section where the itching is very strong. Using meaningful annotations, the medical personnel understands much faster, what the patients suffer is and where to look at.

When the patient finished the annotations, he feels comfortable to have a video call with a doctor. The patient created an appointment with the doctor. He also provides access to the file he just annotated to the doctor. A few minutes before the appointment starts, the doctor opens the file and can already see the annotations. At the time of the appointment the doctor calls the patient back.

Now, the doctor and the patient collaboratively discuss the annotated image and add even more annotations to it.

After the video call the patients downloads the picture including its annotations to show it to his general practitioner and discuss the consultation results with him.

Figure 2.1 visualizes the process and the interactions between the different roles.

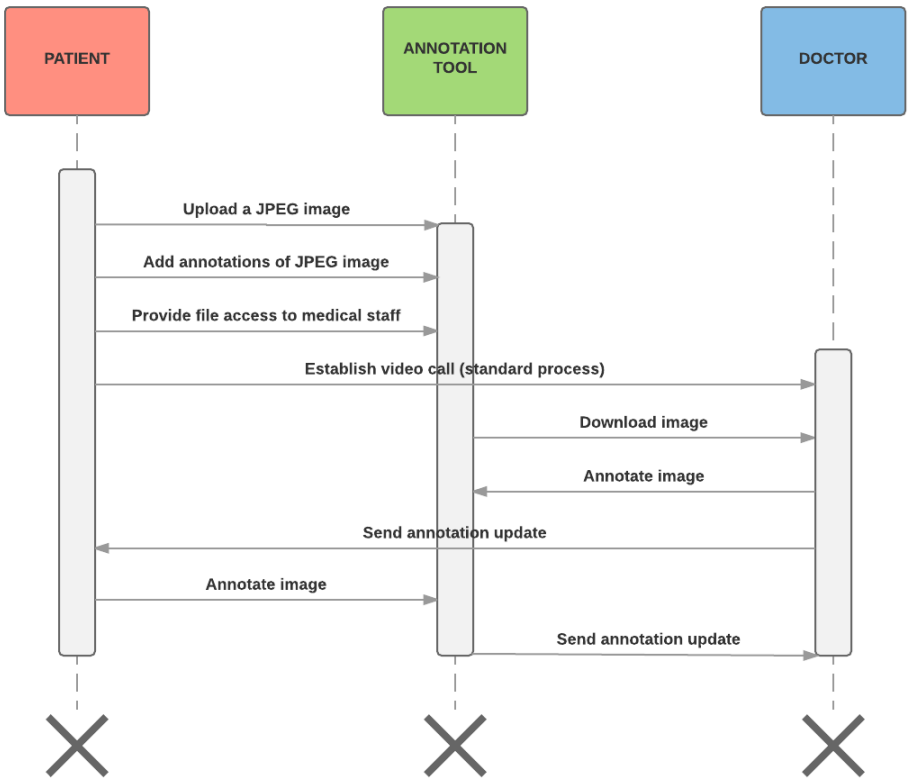


Figure 2.1: Scenario 1 – Sequence Diagram

2.2.3 Technology

The user signs in using a Firefox browser. He uses a MacBook Pro from 2012 and Mountain Lion as Operating System.

The doctor uses a Lenovo YOGA 500 Notebook, Intel Pentium 3805U, 1,9GHz, 4GB RAM and a 128 GB hard drive. As an operating system he has Windows 10 Enterprise Edition and he uses Google Chrome Browser, version 51.0.0.

2.2.4 Architecture

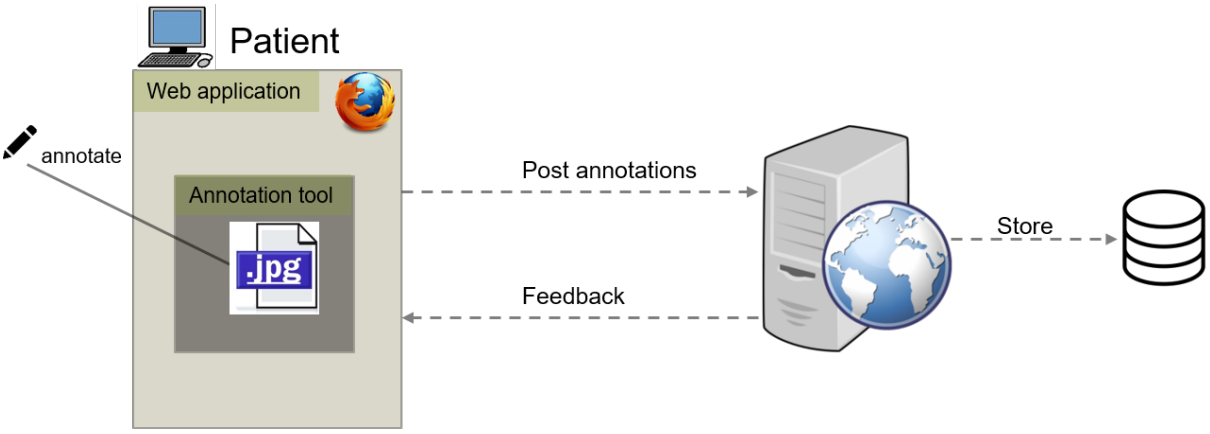


Figure 2.2: Scenario 1 - Previous annotation of a file

Figure 2.2 shows the architecture of annotating a file and storing the annotations on a server. The user starts to annotate the file by himself, so no communication between two clients is needed.

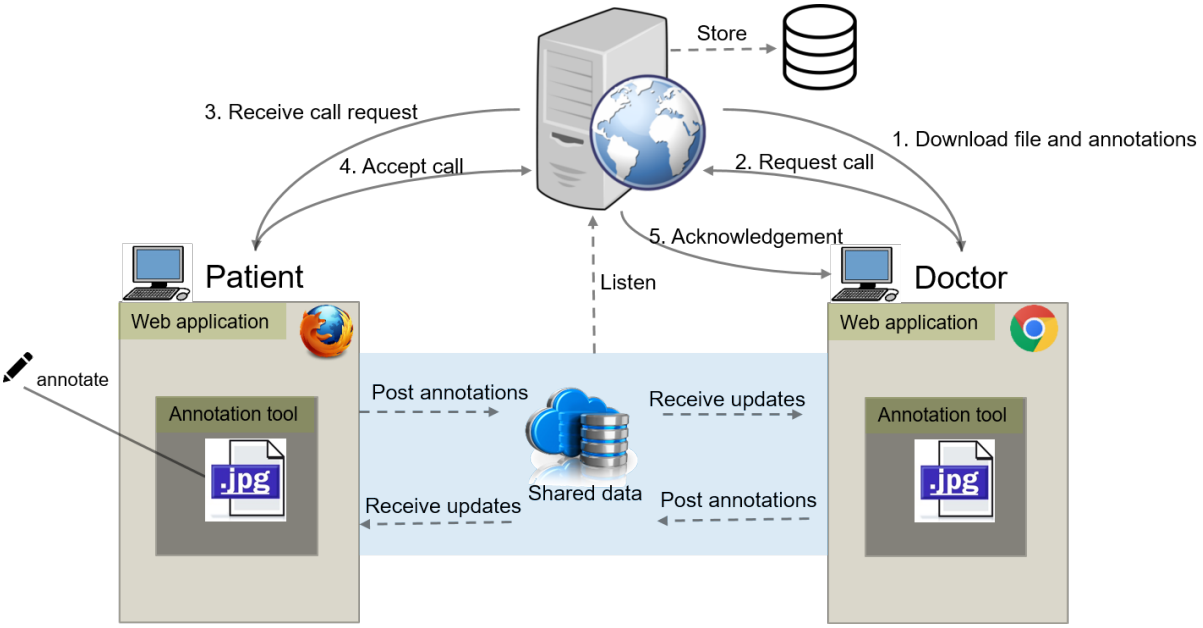


Figure 2.3: Scenario 1 – Initial handshake and data transmission during the annotation process

Figure 2.3 shows the handshake between doctor and patient in case, the doctor calls the patient. In the beginning, the doctor has to download the file and the annotations from the

server, the patient stored his annotations in. After the doctor had read the annotations, he starts to call the patient.

In the beginning of the call, both parties can see the same image and annotations which have been loaded from the server during the initial handshake. Now, both parties start to add annotations to the file at the same time. All necessary data of the doctor's annotations will be transmitted to patient and displayed on his image and vice versa. This is happening in parallel to the existing connection of the video call.

2.2.5 Requirements

Based on the scenario, the following requirements can be derived. In section 3.2 the requirements of all scenarios will specified in detail according to Kazman et. al. [Kazm 12].

- Run in a browser
- Synchronize the view
- Manage file access to others
- Draw a line on a file
- Store author information

2.3 Scenario 2

In the second scenario a process regarding three different users is described. All users have different roles. During certain times, many people call. In order to distribute the workload, all incoming calls are received by some medical personnel. In case, the medical personnel cannot help they will forward the calling user to a doctor.

2.3.1 Roles

- A patient who calls TeleClinic via video
  A user logs in to the system without having an appointment. He decides to start a video call anyways.
- A nurse, who conducts a first interview
  The nurse is part of the medical staff and accepts all incoming calls in the beginning. If she is not able to provide a satisfying consultation she forwards the call to a doctor.

- A doctor who conducts a medical consultation
  In this case, a doctor is online anyway and receives an incoming call. He accepts the call and automatically enters a web meeting with the patient and the nurse.

2.3.2 Process

A patient just logs in to the medical portal of TeleClinic and starts a video call. He gets connected to a nurse immediately and she asks the patient to upload a document with information about his broken hand. After the upload succeeded, both participants of the conversation can see the uploaded PDF document in the annotation tool.

While the patient explains his suffer, both parties highlight important sections in the PDF document. The nurse also adds a little text note for the doctor to the file. Fortunately, a specialized doctor is online and the nurse invites him to the video call.

Due to the annotated document the nurse can provide a brief explanation to the doctor and inform him about the patients suffer. Now, three different parties can discuss the annotated document and also collaboratively add more annotations in order to find a solution.

In order to prevent confusions, every participator of the annotation session has another unique color. So, all participators can clearly differentiate to whom each annotation belongs to.

The doctor figures out, that the nurse added a note which is not correct from a medical perspective. He quickly removes the wrong annotations to avoid confusions.

After a while, the nurse is not needed anymore and decides to leave the video call. The patient and the doctor can see the status update of the nurse immediately and can see that the nurse is not participating anymore.

The doctor diagnoses a complex break and suggests that the patient should go to a radiologist clinic. He should get scanned in an MRT scan. Later, the patient uploads a mp4 video of his MRT scan and shares it with the doctor. Now, both parties collaboratively annotate certain regions of the video which is synchronized in real time. The doctor explains the problem to the patient and explains him a strategy to recover soon. Figure 2.4 visualizes the process and the interactions between the different roles.

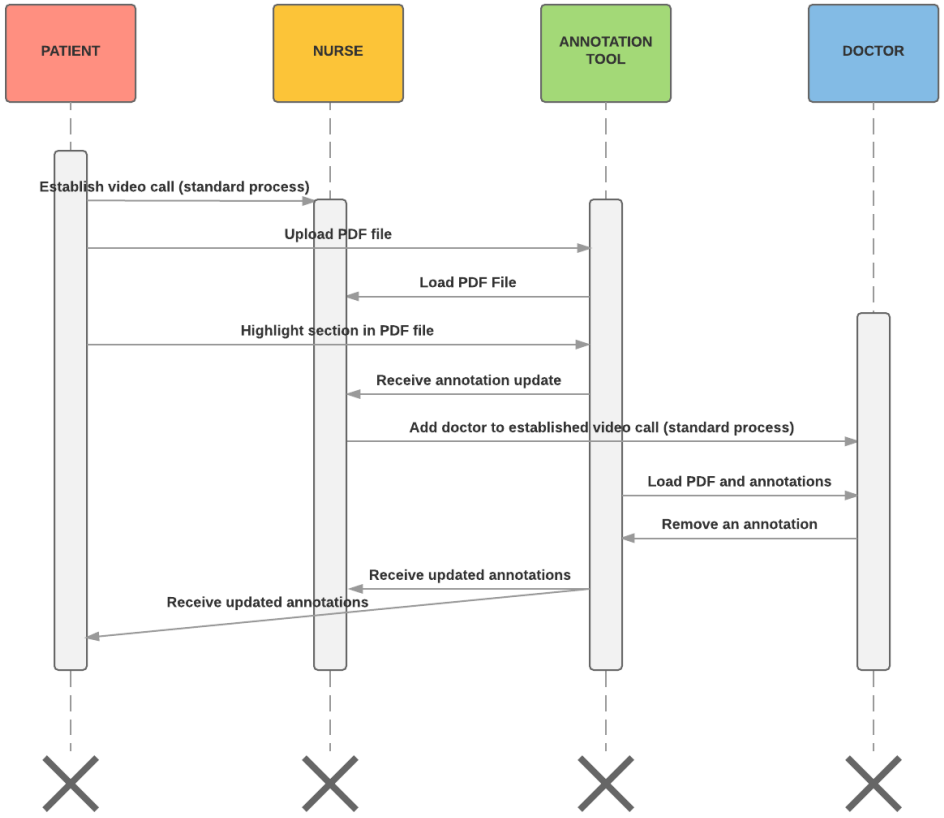


Figure 2.4: Scenario 2 – Sequence Diagram

2.3.3 Technology

The user signs in using an Opera browser. He uses a Laptop from 2010 and Windows 7 as Operating System.

The nurse and the doctor both use a Lenovo YOGA 500 Notebook, Intel Pentium 3805U, 1,9GHz, 4GB RAM and a 128 GB hard drive. As an operating they have Windows 10 Enterprise Edition and a Google Chrome Browser, version 51.0.0.

2.3.4 Architecture

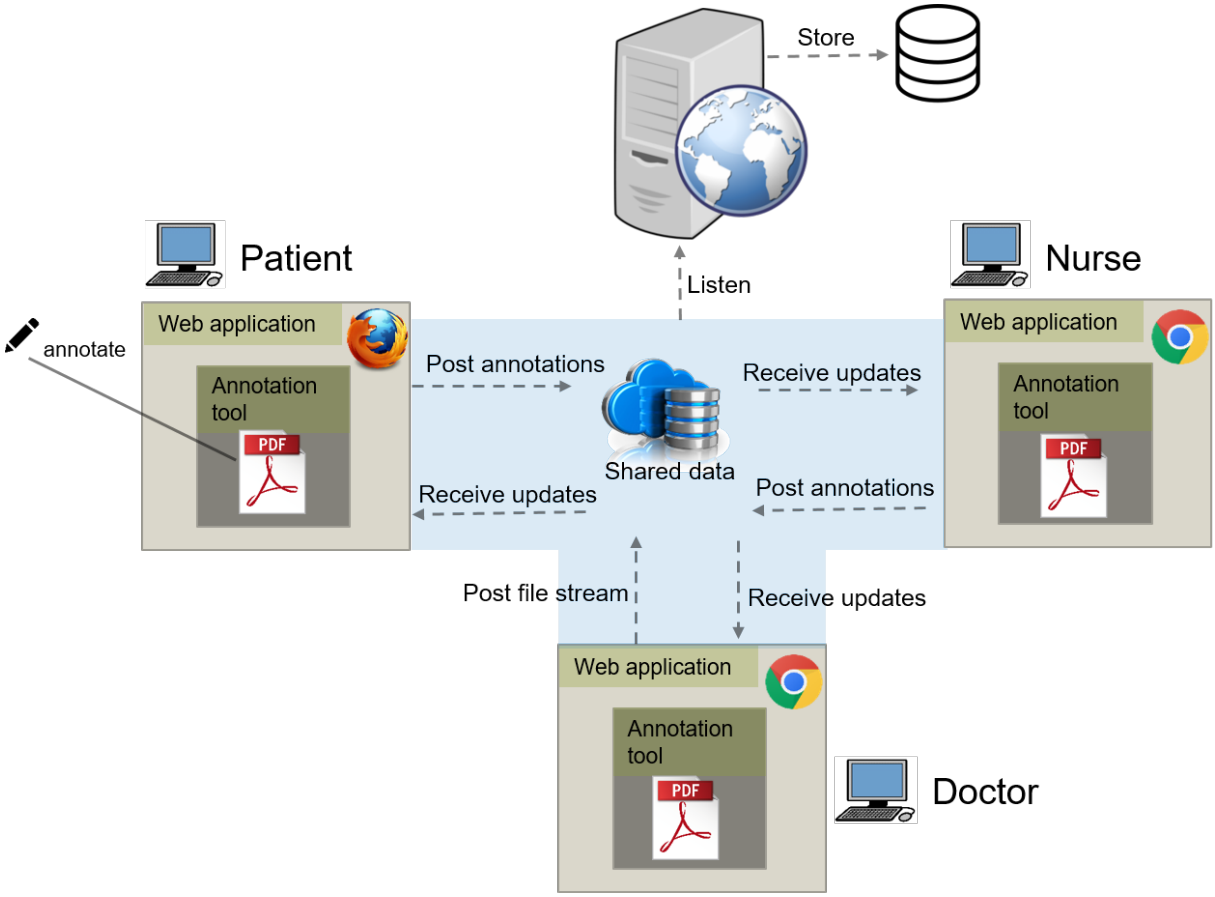


Figure 2.5: Scenario 2 – Annotation of a PDF document with 3 different parties

Figure 2.5 shows the architecture of annotating a PDF file with three different parties. The procedure of an initial handshake was skipped because it was explained in the first scenario already. In the beginning, only the nurse and the patient are annotating the file. Later, also the doctor participates the annotation session and can add annotations.

A few days later the patient signs in again and downloads the annotated file to share it with another doctor.

2.3.5 Requirements

Based on the scenario, the following requirements can be derived. In section 3.2 the requirements of all scenarios will specified in detail according to Kazman et. al. [Kazm 12].

- Highlight a part of a file
- Add text field to a file
- Support different file types
- Download annotated file

2.4 Scenario 3

In this scenario a highly specialized doctor tries to explain an X-ray image to a patient. This scenario originates from expectations of several doctors (e.g. radiologists) who use special medical software and the wish of many customers to spend more focus on files such as X-ray and MRT images.

2.4.1 Roles

- A radiologist, explaining an X-ray image to a patient
  This doctor is an expert. He uses special software to view X-ray images and delivers highly specialized information to the patient. Annotations can support the explanations a lot, in his opinion.
- A patient who wants to review his X-ray image
  This patient visited a radiological clinic a few weeks ago. He is very busy and wants to discuss the results in a video call with an expert.

2.4.2 Process

An X-ray image of a patient was made in a radiological clinic. The patient has a very strong back pain. In order to review the results, the doctor and the patient are having a video call. An X-ray image is not a simple JPEG file. It consists out of many shapes and can have huge sizes. It is usually stored as a DICOM file, which can only be opened with special software.

The doctor has an DICOM image viewer which can be embedded in a Web browser. The annotation tool now embeds the X-ray image which is stored on the encrypted hard drive of the doctor. The patient does not have an X-ray image viewer and the images size is too large to upload it anyways. Hence, the doctor wants to share the frame of his screen where the X-ray image is embedded. He wants to provide the possibility of adding annotations

to shares screen section as well. The doctor starts to highlight some important sections and also circles a hairline crack he wants to show to the patient.

In an DICOM image viewer, many perspectives of an X-ray or MRT image can be viewed. Because of strategic reasons, the doctor doesn't want to transmit the view sometimes. Some pictures of the X-ray might confuse the patient. If the doctor opens certain views of the image, he wants to press a button to turn the screen black and to disable the view for the patient for a while.

In addition, after a while the bandwidth becomes very low for a minute. Therefore, the patient is not able to receive any updates of the annotations for a while. After a minute the connection becomes stable again and the patient receives the updates he didn't receive before.

Figure 2.6 visualizes the process and the interactions between the different roles.

2.4.3 Technology

The user signs in using a Vivaldi 1.2 browser. He uses a Laptop from 2010 and Windows 7 as Operating System.

The doctor uses a Lenovo YOGA 500 Notebook, Intel Pentium 3805U, 1,9GHz, 4GB RAM and a 128 GB hard drive. As an operating system he has Windows 10 Enterprise Edition and he uses Google Chrome Browser, version 51.0.0.

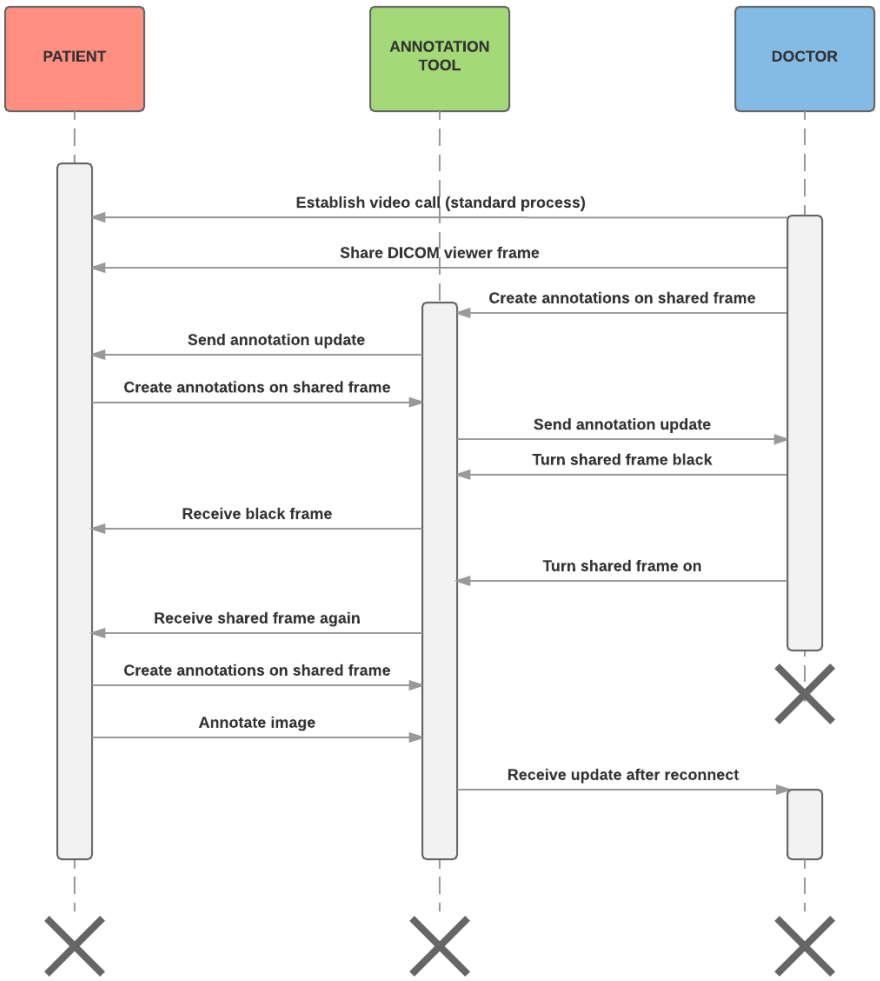


Figure 2.6: Scenario 3 – Sequence Diagram

2.4.4 Architecture

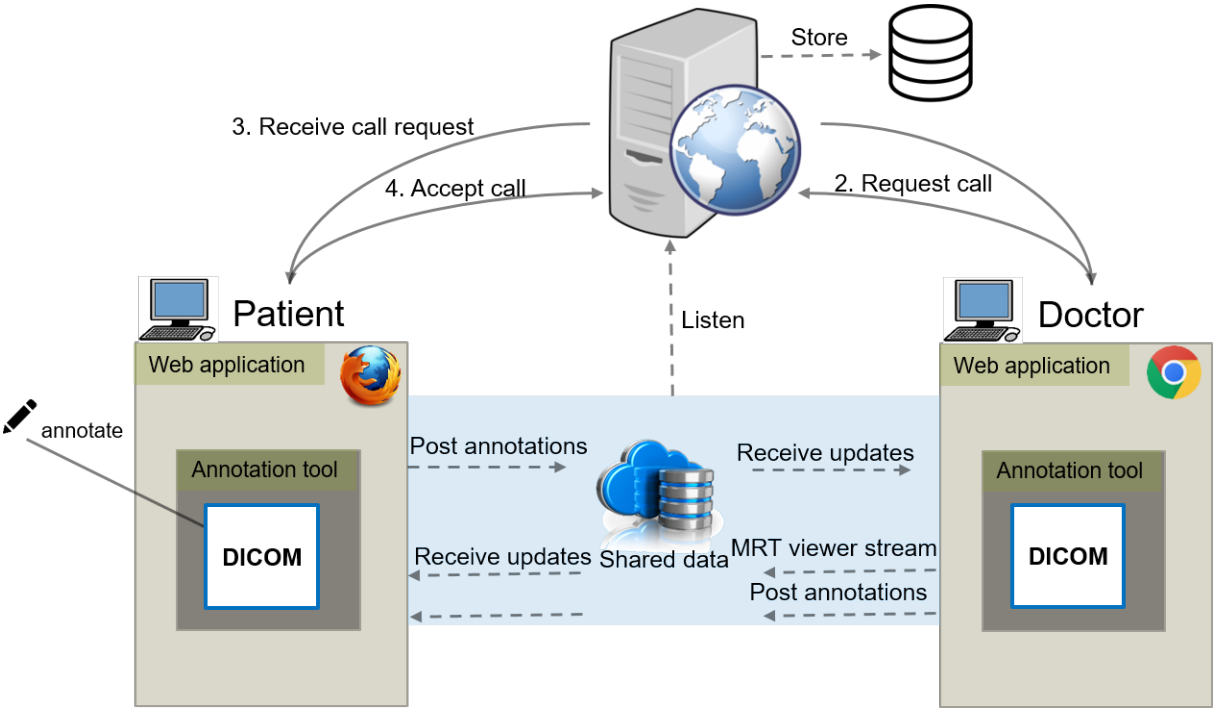


Figure 2.7: Scenario 3 – Annotation of an MRT image which is not stored on the server

Figure 2.7 shows the annotation process between a doctor and a patient. In order to do share a section of his screen, the shared content gets transmitted to the patient. Two streams, one for the annotations and one for the screen video will be transmitted at the same time.

2.4.5 Requirements

Based on the scenario, the following requirements can be derived. In section 3.2 the requirements of all scenarios will specified in detail according to Kazman et. al. [Kazm 12].

- Delete objects which have been added to the annotated area
- Turn annotation screen off for a while
- Support files embedded via plugin

Chapter 3

Requirements

In chapter 2 different scenarios of using the annotation tool were described. Each scenario delivered different requirements, which are mandatory or optional for designing an annotation tool within the given environment. In the following the determined requirements will be summarized.

Brügge suggests, to split requirements into functional and non-functional requirements which is everything that is not a feature of the tool [Brue 10]. As non-functional requirements are not so clear defined, Kazman et. al. suggests to split the requirements into functional requirements and Quality Attributes. A Quality Attribute is everything which is not a new feature and adds measurable quality [Kazm 12].

3.1 Functional Requirements

1. **Annotation area**
   The app should contain an area which can be annotated with different tools. The tools will be explained in the upcoming requirements. The area should be able to contain different file types, e.g. a picture or a pdf file.
2. **Synchronize the view with all participants in real time**
   The app should be a collaborate app. Therefore, as soon as somebody joins an annotation session, all activities should be visible by all participants of the annotation session. All changes should be broadcasted and appear on the screen of all participants as fast as possible. The only limiting factor should be the Internet connection of particular users.
3. **Draw a line on a file**
   The user has the possibility to select different tools. After the pencil tool is selected

and a user clicks on a certain location of the file. While the mouse is pressed, the user draws a line on the screen. All other participants should see the line immediately.

4. **Highlight a part of a file**
   After the highlighter tool is selected and a user clicks on a certain location of the file. Now he can draw a yellow transparent line to highlight sections in a file. The highlighted are appears on the screen of all participants of the annotation session.

5. **Add text field to a file**
   After the text tool is selected and a user clicks on a certain location of the file, a text field will be open. Now, he is able to add some text to a certain location of a file.

6. **Add shapes as a circle or a rectangle to a file**
   This might support the user in pointing to important sections. If a user wants to highlight a part of an image without changing the color (as a highlighter does) he can circle the section.

7. **Delete objects which have been added to the annotated area**
   The user can delete objects which have been added to the annotated area. He should only be able to delete the annotation items if he has the special rights of deleting it. A user has the right to delete an annotation element if he created it or if he has special rights, like a doctor.

8. **Turn annotation screen off for a while**
   The person, who hosts the current annotation stream or a specialist should always be able to mute his current transmission stream. There are two reasons:

   - **Privacy** The user should always be able to protect his privacy. In case, he does not want to share certain images, he can simply turn the annotation screen black for a while by clicking a button.
   - **Doctor information** There are some information, e.g. in X-Ray images which are only relevant for a doctor. The information can confuse the patient. Also It could be possible, that a doctor shares a part of his screen with the patient which contains wrong information. Hence, the doctor should also be able to mute his screen in case there are certain information he doesn't want to share with a patient. Hospitals also almost never share all their files with the patient [§ 10 Abs. 2 BOÄK].

9. **See other collaborators**
   All users should be able to see who participates in the current annotation session. Furthermore, all users should be able to see which user created which elements.

10. **Support different image file types, as JPG, JPEG, PNG**

The upload of different file types like images (JPG, JPEG, PNG) should be possible. Users should share the files with medical staff and vice versa.

11. **Support the annotation of PDF files**
    The annotation of PDF files should also be enabled. People should annotate the files just like a regular picture, but also scroll through the different pages.

12. **Support the annotation of DICOM files**
    The DICOM (Digital Imaging and Communications in Medicine) format is used in medicine to store complex images which contain different layers, shapes or contrasts, as e.g. MRT images or X-ray images [Mild 02]. People who upload medical data might also upload DICOM images. Hence, annotating DICOM images can support the consultation process as well.

13. **Support common video formats**
    This case might only occur rarely but can still be important. If people upload a video of a certain behavior, for example a reflex, an injury or even a longer procedure, like an operation, experts could annotate the video in real time. They can also go back or pause the video to annotate it while other participants see the results in real time.

14. **Support files embedded via plugin**
    Besides just sharing files, it should be possible to share screen sections using a video stream. If e.g. a doctor uses browser plugin for viewing X-Ray images, it should be able to transmit a video stream of that screen part to all participants. They should annotate the video, as a regular image. Everybody should see the same screen with the same annotations.

15. **Store author information**
    During the creation of different annotation elements, it should be clear who created an element. Therefore, the author information should be stored in the annotation element.

16. **Download annotated file**
    It should be possible to download the image, including its annotations. After a button gets clicked, the image should download as regular, but it contains all the added annotations. They should be rendered in the image.

17. **Replay the annotations**
    Annotations can quickly become very confusing. Therefore, it is useful to replay the annotations. After clicking a button, all annotations should be added in the same way they have been added before.

18. **Consistent color scheme**

Each user should obtain his own color which is consistent all over the annotation tool. Hence, each participant can easily see which user caused which annotation.

19. **Display participant status**
    During the collaboration, the status of all participants should be displayed. I.e. that each user can always see, if a participant is offline, online or actively participating.

3.2 Quality Attributes

In the following, all non-functional requirements will be specified using Quality Attributes according to Bass; Clements; Kazman, 2012 [Kazm 12]. Each Quality Attribute will be described in six parts:

- **Source of stimulus:** "This is some entity (a human, a computer system, or any other actuator) that generated the stimulus."
- **Stimulus:** "The stimulus is a condition that requires a response when it arrives at a system."
- **Environment:** "The stimulus occurs under certain conditions. The system may be in an overload condition or in normal operation, or some other relevant state. For many systems, "normal" operation can refer to one of a number of modes. For these kinds of systems, the environment should specify in which mode the system is executing."
- **Artifact:** "Some artifact is stimulated. This may be a collection of systems, the whole system, or some piece or pieces of it."
- **Response:** "The response is the activity undertaken as the result of the arrival of the stimulus."
- **Response measure:** "When the response occurs, it should be measurable in some fashion so that the requirement can be tested."

This six points have been cited from [Kazm 12]. In the following, all determined requirements will be explained in detail, using the structure by Kazman et. al.

Additionally, all Quality Attributes will be explained regarding its necessity for security, scalability and reusability.

The Quality Attributes are given by technical environment and security policies and are not directly derived by the scenarios.

1. Modularity / Reusability in all connected systems.

- **Source of stimulus:** System
- **Stimulus:** Should be included as a library in different systems
- **Environment:** During the implementation process.
- **Artifact:** Frontend, Backend
- **Response:** Easy installation
- **Response measure:** After bootstrapping the package it should be easy to include directly usable
- The annotation tool should not just be used inside of the doctor's workspace. It should also be used by other clients, as the web portal which is used by patients.

2. Run within the browser environment
   - **Source of stimulus:** Human
   - **Stimulus:** All users who open the app in a browser should be able to use the annotation tool
   - **Environment:** During the annotation process.
   - **Artifact:** Frontend
   - **Response:** People just see the tool included in a web application
   - **Response measure:** After logging in people can use the software without using extra software
   - The doctor's software and the portal which is used by the doctor are web based. Hence, the tool which will be inserted should also be usable within the browsers environment.
3. Deal with bad connection quality
   - **Source of stimulus:** System
   - **Stimulus:** If low throughput, high latency, high package lost or high variation occurs
   - **Environment:** Staying in the annotation session with a bad internet connection
   - **Artifact:** Frontend, Backend
   - **Response:** People just see the tool included in a web application

- **Response measure:** After logging in people can use the software without using extra software
- If the connection turns bad, e.g. through time lags, interruptions or very low speed, the tool should not crash. As far, as the connection is established again, the missed data should be received.

4. Optimized data structure
   - **Source of stimulus:** System
   - **Stimulus:** Saving coordinates of annotations in database and broadcast it to other participants should happen using a well selected data structure
   - **Environment:** Saving and transmitting the annotations
   - **Artifact:** Frontend, Backend, Database
   - **Response:** Data size of the annotations should be as small as possible and as easy to parse as possible
   - **Response measure:** Read and write 10000 of tuples within seconds, ready for horizontal scalability, web-ready (e.g. JSON, CSV based)
   - Data should be easy to read and as small as possible. This supports fast transmission rates of data and easy database operations.
5. Different screen sizes and devices
   - **Source of stimulus:** System / Browser
   - **Stimulus:** The screen size of different devices is different; mostly all devices should be supported
   - **Environment:** Opening and using the annotation tool
   - **Artifact:** Frontend, Backend, Database
   - **Response:** The tool should work in all browsers which support WebRTC (Required for video calls) and CSS3
   - **Response measure:** No graphical or functional disruption on all common used devices
   - The tool gets used on several different screens and should support a wide set of common resolutions.
6. Robustness
   - **Source of stimulus:** System

- **Stimulus:** Error handling - Give no possibility to cause a bug and make the software intuitive to use
- **Environment:** Using the annotation tool
- **Artifact:** Frontend, Backend
- **Response:** Catch all possible bugs and deal with them
- **Response measure:** User can do nothing to cause a bug
- The system should be able to deal with bad circumstances, i.e. bad connection, low performance, wrong usage or bugs.

7. Scalability
    - **Source of stimulus:** Human
    - **Stimulus:** Many users use the system at the same time
    - **Environment:** Using the annotation tool
    - **Artifact:** Backend
    - **Response:** Don't let the user wait too long for a response
    - **Response measure:** Use as less operations as possible
    - The tool should be used by several users at the same time without crashing. I.e. the databases, the backend and the frontend can deal with several operations at the same time.
8. Don't store data on hard drive
    - **Source of stimulus:** Human
    - **Stimulus:** Data should be never downloaded to create annotations
    - **Environment:** During an annotation session
    - **Artifact:** Frontend
    - **Response:** Store data within a browser
    - **Response measure:** Data is only available in the browser scope
    - Files should not be downloaded to be annotated and just be editable within the browser environment. This helps to access files from everywhere with a web browser. Also, users don't have to care about secure storing of data on a harddrive.
9. Encrypted data transfer

- **Source of stimulus:** System
- **Stimulus:** If the data gets transmitted
- **Environment:** During an annotation session
- **Artifact:** Frontend, Backend
- **Response:** Data should only be transmitted encrypted
- **Response measure:** No third parties should be able to decrypt the transmitted data
- Transmitted data should always be encrypted. This is important to ensure data security and prevent the access of third parties to the data.

10. End-to-end connection
    - **Source of stimulus:** System
    - **Stimulus:** If the data gets transmitted
    - **Environment:** During an annotation session
    - **Artifact:** Frontend
    - **Response:** Data should be transmitted using an end-to-end connection without any server in the middle
    - **Response measure:** A direct channel between all parties is established
    - This requirement was a demand by the TeleClinic personnel. Possible methods of transmitting data will be discussed in section 4.3.4.2.
11. Frontend in TypeScript / Angular 2
    - **Source of stimulus:** System
    - **Stimulus:** Whole frontend is written in Angular 2
    - **Environment:** -
    - **Artifact:** Frontend
    - **Response:** The tool should be written in Angular 2
    - **Response measure:** The tool can be included as an Angular 2 library
    - The whole frontend is written in TypeScript using the Angular 2 framework. This tools should be used as a library within the Angular 2 app. Hence, writing the app in TypeScript will make the implementation of the tool much easier.

12. Backend in Python
    - **Source of stimulus:** System
    - **Stimulus:** Whole backend is written in Phython / Django REST framework
    - **Environment:** -
    - **Artifact:** Backend
    - **Response:** The backend of the tool should be written in Python
    - **Response measure:** The tool can be included as a Python library
    - The whole backend is written in Python. The annotation tool should be usable within the given environment. Setting up a second server would be a lot of extra implementation work which should be prevented.
13. Low effort of implementation
    - **Source of stimulus:** Programming concept
    - **Stimulus:** Whenever a new concept has to be implemented
    - **Environment:** During the implementation process
    - **Artifact:** Frontend / Backend / Database
    - **Response:** Totally working solution
    - **Response measure:** Implementation time should be as small as possible
    - Implementing the tool should be kept easy and prevent unnecessary work. A concept of a good structure for creating the app is mandatory.
14. High Efficiency
    - **Source of stimulus:** Programming concept
    - **Stimulus:** Whenever a new concept has to be implemented
    - **Environment:** Everywhere, where data gets processed
    - **Artifact:** Frontend / Backend / Database
    - **Response:** Correct calculation
    - **Response measure:** Calculation time should be as fast as possible
    - The tool should also work of older systems that don't have much performance. The programming concept should be chosen in order to work efficient and save resources.

15. Separation of concerns
    - **Source of stimulus:** Programming concept
    - **Stimulus:** Whenever a new concept has to be implemented
    - **Environment:** Components, Services, Backend, Database
    - **Artifact:** Frontend / Backend / Database
    - **Response:** Clear justification
    - **Response measure:** It should be possible for every function why to justify why the function is part of the component.
    - Every component should have its clear responsibility. This reduces the programming effort, because the programmer knows directly where to search for a function. Also, the architecture becomes more understandable.

3.3 Prioritizing Requirements

In the previous section many requirements were derived from different scenarios. Obviously, not all requirements have the same importance. There are two different measures to evaluate the importance of the different requirements. First, it has to be determined which requirement occurs how many times in the scenarios which were given by the stakeholders. Just the occurrence of requirements doesn't deliver enough information about the importance of functional requirements to real users.
Hence, a survey with 22 experts was conducted as a second step with the goal of evaluating the importance of functional requirements. All experts have at least three months of experience in medical teleconsultation and a medical education. In order to do so, several doctors of the TeleClinic network were asked to rate for the importance of the functional requirements using a Likert scale.

The Likert scale is a tool which is frequently used in surveys. Users can easily assign their subjective attitude to different topics by assigning points [Like 32]. Assigning a low number of points means, that people strongly disagree. If people assign a lot of points to one statement, they have a strong agreement to the statement. The advantage of using a Likert scale is that every user can understand it easily. Furthermore, the Likert scale is a numeric value which allows the conductor of the survey to use statistical measurements as the arithmetic mean or the median [Like 61]. A section of the used Likert scale is displayed in figure 3.1.

For each requirement the doctors had the possibility to assign between 1 and 5 points, where 1 means "unnecessary" and 5 means "very important". Using a Likert scale provides

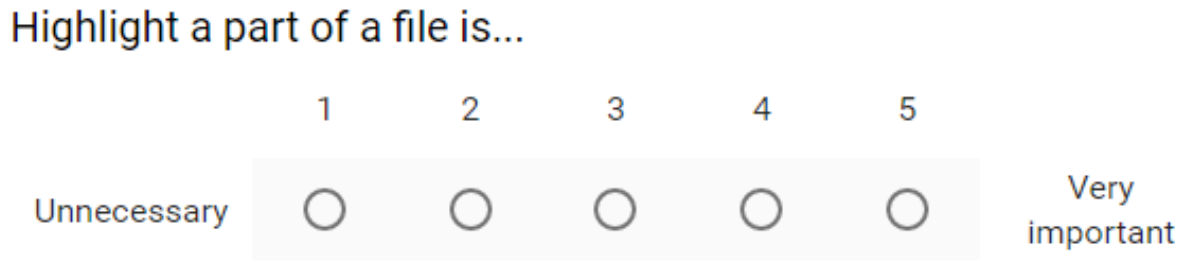


Figure 3.1: Section of the conducted survey

an easy way of assigning a rating to requirements and calculate a score to determine the importance of each requirement. Every doctors understands immediately what he rates for and can express whether he agrees or disagrees with a requirement of the tool.

Two surveys were conducted for this thesis. The first survey contains a Likert scale between 1 (unnecessary) and 5 (very important) for each functional requirement. The doctors can rate how important a functional requirement is. The whole survey, including all questions and all results, is attached to the appendix of this thesis.

3.3.1 Occurrence of functional requirements in scenarios

The following section will display all requirements according to their appearance in the different scenarios.

Table 3.1: Necessity of requirements in different scenarios

No.	Requirement	Scenario 1	Scenario 2	Scenario 3
1	The tool should have an annotation area	X	X	X
2	Synchronize the view with all participants in real time	X	X	X
3	Draw a line on a file		X	X
4	Highlight a part of a file	X		
5	Add text field to a file	X		
6	Add shapes as a circle or a rectangle to a file			X
7	Delete objects which have been added to the annotated area		X	
8	Turn annotation screen off for a while			X
9	See other collaborators	X	X	X
10	Support different image file types, as JPG, JPEG, PNG	X		
11	Support the annotation of PDF files	X		
12	Support the annotation of DICOM files			X
13	Support common videos formats			X
14	Support files embedded via plugin			X
15	Store author information	X	X	X
16	Download annotated file	X		
17	Replay the annotations			X
18	Consistent color scheme		X	X
19	Display Participant status		X	

3.3.2 Functional requirements importance based on a survey

As mentioned before, the importance of the functional requirements was also evaluated using a survey. **22 different experts** of TeleClinic who will use the annotation tool in the future were asked. All participants of the survey were supposed to be experts. I.e. that all participators need to have an **apprenticeship in a medical field** and at least **three months of experience** in using the consultation software for which the annotation tool is written. Furthermore, all participators have work experience with annotating images, as e.g. the annotation of X-ray images. As a target group, general practitioners, radiologists, nurses and medical assistants (in German: “Medizinische Fachangestellte”) were asked.

All potential users were supposed to rate for the importance of each functional requirement using a Likert scale between 1 and 5. 1 means that the requirement is unnecessary

(German: überflüssig) and 5 mean that the requirement is extremely important (German: extrem wichtig).

In the following part, all original questions, its English translations and the answers made by the medical staff will be displayed. Afterwards, the characteristics will be discussed and analyzed for applicability to the tool. As all the participators are Germans, the survey was conducted in German. All German original questions which were asked to the participants are printed after each requirement.

Google Forms was used to create and conduct the survey. The following images of this section were generated by Google Forms based on the survey which was conducted during this Master's Thesis.

1 The tool should have an annotation area

This requirement is the main functionality of the annotation tool. All other requirements are depending on this main functionality. The need of annotating files is the main topic of this master thesis and the importance was already mentioned in the previous sections. The supportive functionality and applicability has been already explained in the motivation of this thesis and in the scenarios from which the requirements were derived. For this reason, people were not asked about the importance of this requirement.

2 Synchronize the view with all participants in real time

Original German question for the participants: "Jeder, der die Datei geöffnet hat, sollte alle Annotationen in Echtzeit sehen."

Figure 3.2 shows that 81.8 % of all participators rated this requirement with 4 or 5. It is obvious that this requirement seems to be very important for most of the medical staff. Many people upload a file during a call and do not annotate it before. Also, callers want to see an annotation directly when they talk to medical staff and not afterwards. The real time synchronization of annotations improves the quality of the annotations a lot which is also shown in the results of the questions.

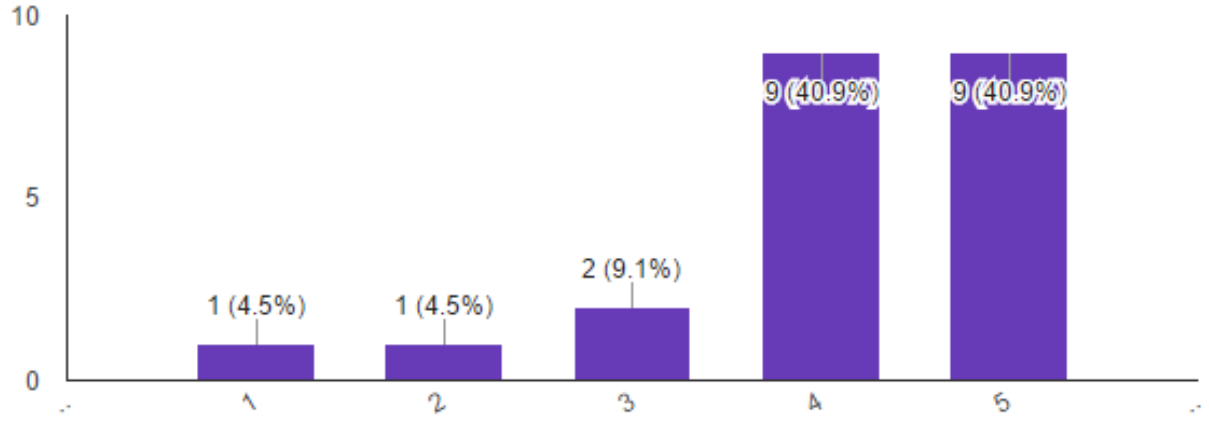


Figure 3.2: Survey – Results of requirement 2: Synchronize the view with all participants in real time

3 Draw a line on a file

Original German question for the participators: "Eine Linie auf dem Bild malen ist ... (1=Überflüssig / 5=Extrem wichtig)"

Figure 3.3 shows the importance of adding a line to a shared file. The mean score of 3,82 shows that adding a line is quite important. Compared to the mean importance of other annotation functions drawing a line is in the middle. Ten people out of 22 assigned four points as importance and six people out of 22 assigned even an importance of five points. Only a few people assigned a low importance for adding a line which means that the function still has a high importance for the participants, in general.

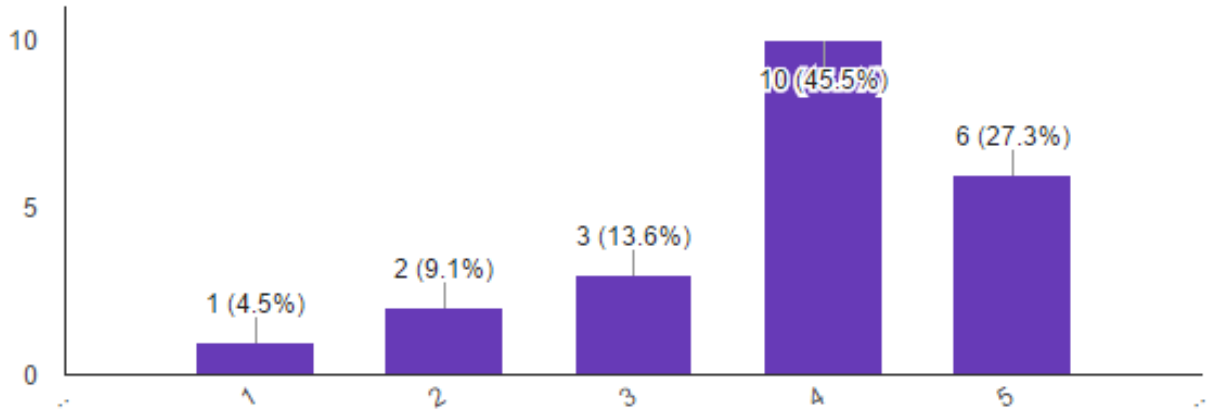


Figure 3.3: Survey – Results of requirement 3: Draw a line on a file

4 Highlight a part of a file

Original German question for the participators: “Einen Bereich hervorheben (wie mit einem Textmarker) ist ... (1=Überflüssig / 5=Extrem wichtig)“

With a mean importance of 4,32 the highlighter has the highest importance of all annotation elements. Figure 3.4 shows that 20 out of 22 people assigned four or five points as importance. During the build process of the annotation tool it should be taken into account that this functionality is rated as extremely important for the participators.

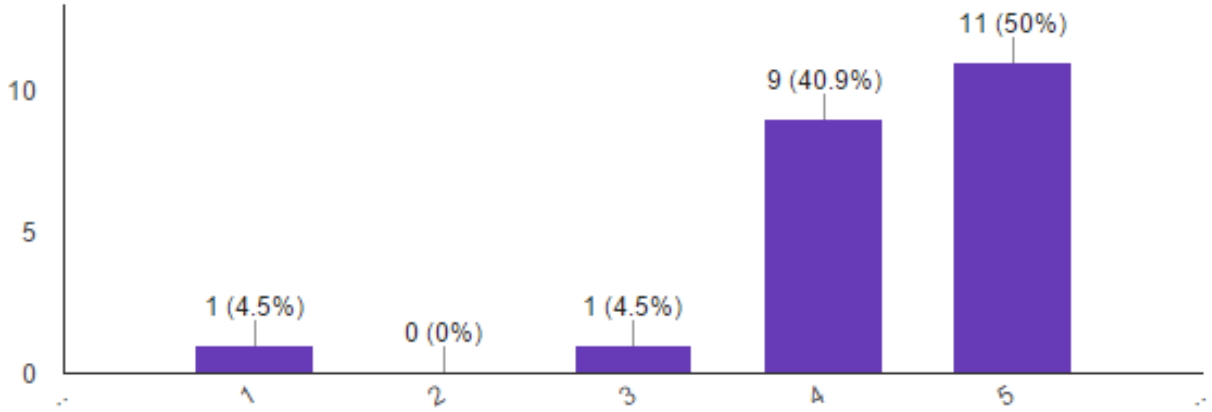


Figure 3.4: Survey – Results of requirement 4: Highlight a part of a file

5 Add text field to a file

Original German question for the participators: “Dem geteilten Bildschirm (z.B. dem Röntgenbild) eine Textbox hinzufügen ist ... (1=Überflüssig / 5=Extrem wichtig)“

The necessity of adding text to a file is very unimportant if only the mean importance of 3,23 is investigated. Figure 3.5 shows that only one person assigned 5 as importance, but the peak is still at 4. Hence, nobody seems to think that this requirement is extremely important but it seems still to be very important many people. Also, only one person selected an importance of 1, which means that adding a text is neither extremely important nor unnecessary for the people. It might useful in many situations even if some other functions are more important.

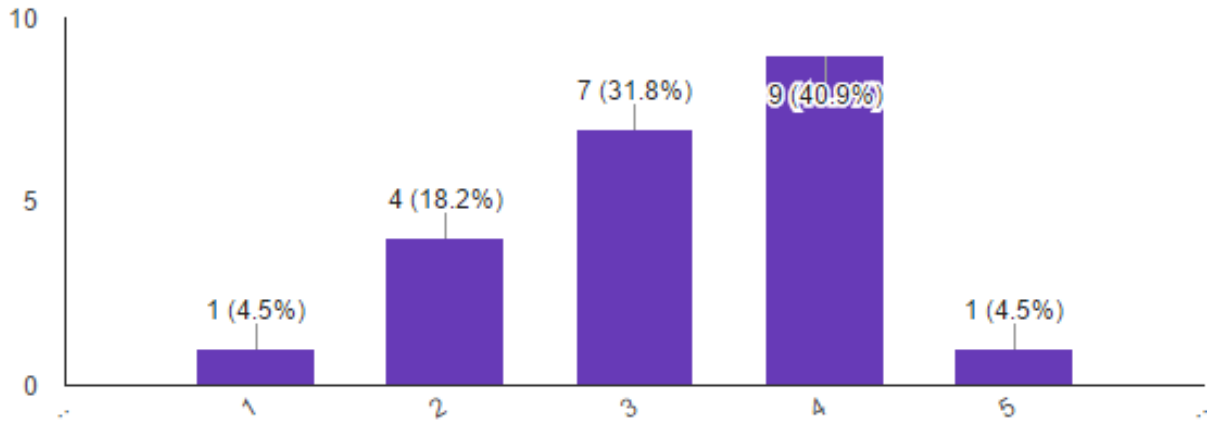


Figure 3.5: Survey – Results of requirement 5: Add text field to a file

6 Add shapes as a circle or a rectangle to a file

Original German question for the participators: "Dem Bild Elemente (wie einen Kreis oder ein Viereck) hinzufügen ist ... (1=Überflüssig / 5=Extrem wichtig)"

The participants were quite undecided about the importance of this requirement. This is visible in the bar chart of figure 3.6. Except for 1 point, all bars have almost the same height. The importance adding shapes, as a circle or a square is important for some people. As a lot of people assigned an importance between 2 and 4, they might not understand why this option would add any additional value for annotating a file.

Another interpretation would be some participants see rare possibilities of adding shapes to a file only. As only one person out of 22 assigned 1 point (which means unnecessary) it seems like the participants of the survey do not think that the option of adding shapes is unnecessary, in general.

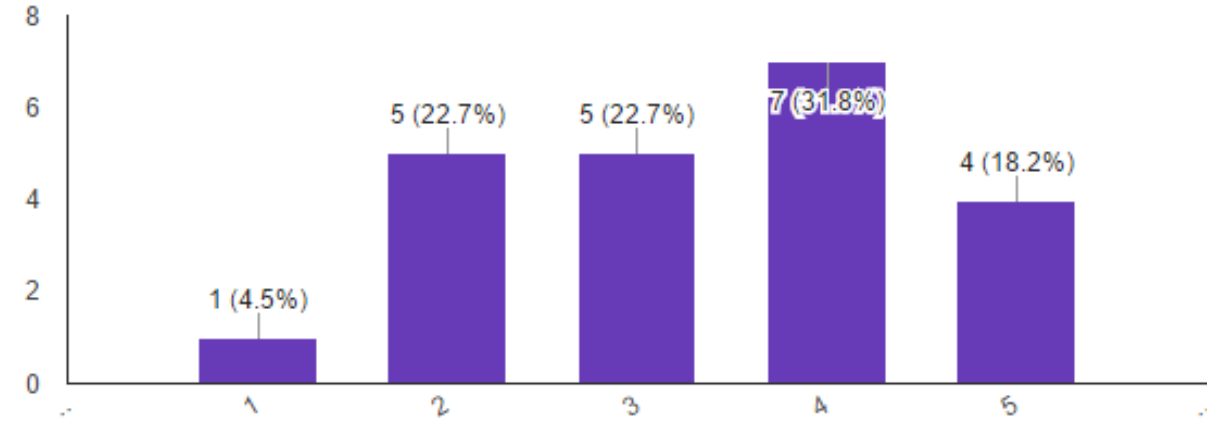


Figure 3.6: Survey – Results of requirement 6: Add shapes as a circle or a rectangle to a file

7 Delete objects which have been added to the annotated area

Original German question for the participators: “Annotationen löschen ist ... (1=Überflüssig / 5=Extrem wichtig)“

As figure 3.7 shows, the participants have a clear opinion on this feature. Apart from some outliers who assigned only between one and three points as importance, 18 out of 22 participants rated deleting annotations with four or five points. 50% of all participant even assigned five points for deleting annotations.

If participants cause an error or do something which is technically wrong it is important to delete these errors to not confuse other participants as the patient.

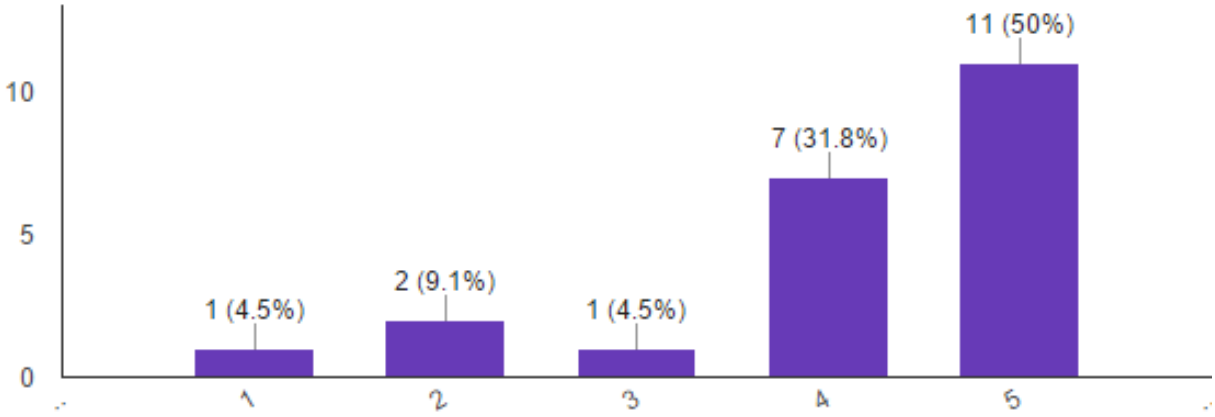


Figure 3.7: Survey – Results of requirement 7: Delete objects which have been added to the annotated area

8 Turn annotation screen off for a while

Original German question for the participators: “Das Ausschalten des geteilten Bildschirms ist ... (1=Überflüssig / 5=Extrem wichtig)“

Figure 3.8 shows the results for turning off the screen. The result equals a normal distribution with $\mu = 3$. Neither turning off the screen seems to be extremely important nor unnecessary. An interpretation for the medium importance could be that people do not understand why this feature is important and when they could use it. I might also be possible that there are only rare possibilities of applying this feature. In the scenario, the reason for turning off the screen was that some sections of shared files, e.g. a part of a video or an MRT image might confuse the patient.

One expert also wrote a textual feedback regarding this requirement:

> “*Ich würde den Button neben ‘Share Screen‘ nicht verstehen (Replay)*“

The participant says that he would not understand this function at all. During the implementation some extra explanation should added, for example using tooltips.

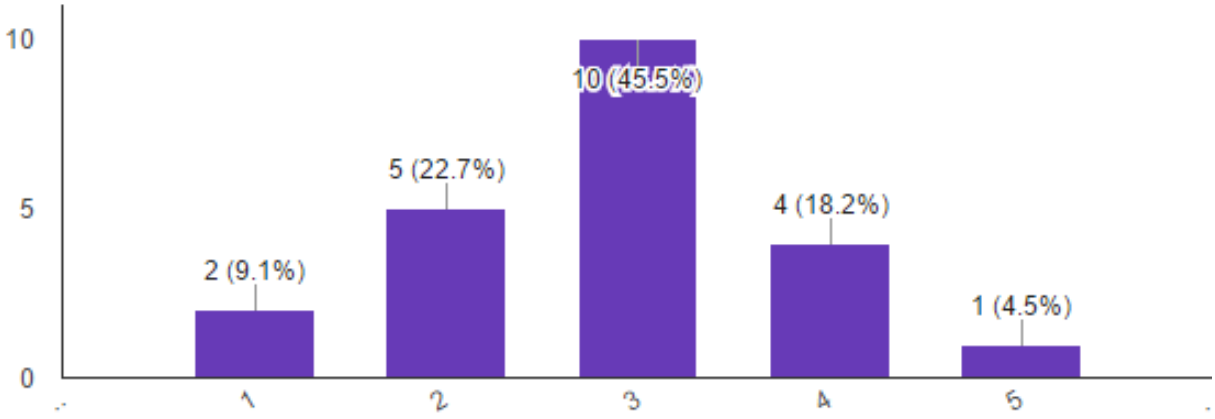


Figure 3.8: Survey – Results of requirement 8: Turn annotation screen off for a while

9 See other collaborators

Original German question for the participators: "Andere Teilnehmer sehen, ist ... (1=Überflüssig / 5=Extrem wichtig)'

Figure 3.9 shows that only two persons out of 22 think that it is not important to see the status of other participants. 3, 4 and 5 points all have many points assigned. The peak is at three points and easily decreases as the points get higher. For most of the participants the feature seems to be very important and to be supportive but not necessary in all situations.

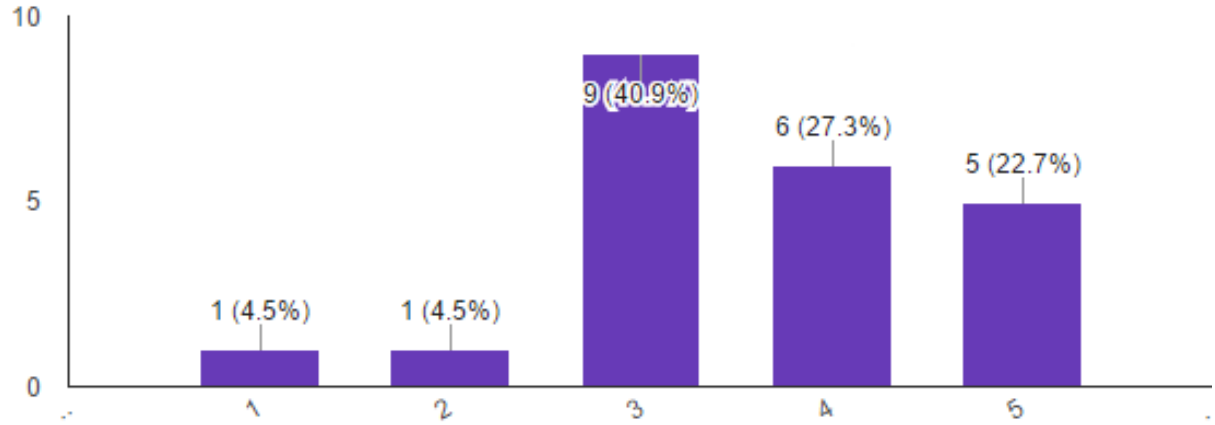


Figure 3.9: Survey – Results of requirement 9: See other collaborators

10 Support different image file types, as JPG, JPEG, PNG

Original German question for the participators: “Die Annotation von JPG, JPEG und PNG Dateien ist ... (1=Überflüssig / 5=Extrem wichtig)‘

The annotation of images is probably the function which will be used most frequently. Figure 3.10 shows that the participants of the survey also think so. Only a few people assigned a low rating to this requirement. More than 3/4 of the people assigned four or five points as importance for supporting typical image formats as JPG, JPEG and PNG.

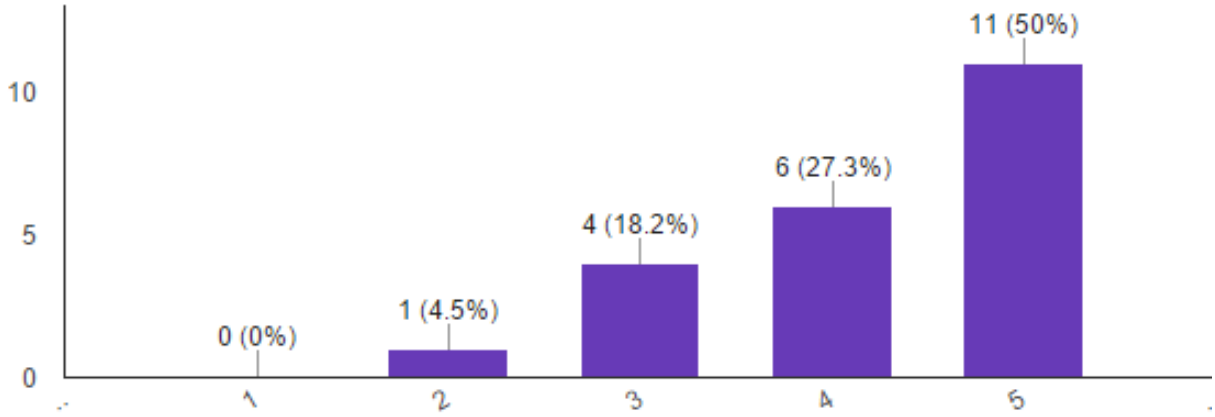


Figure 3.10: Survey – Results of requirement 10: Support different image file types, as JPG, JPEG, PNG

11 Support the annotation of PDF files

Original German question for the participators: “Die Annotation von PDF Dateien ist.. (1=Überflüssig / 5=Extrem wichtig)‘

The bar chart in figure 3.11 shows that the requirement for annotating PDF files has a few more outliers who only assigned between one and three points. However, there is a very high peak at four points and five people even assigned 5 points as important. As many people already upload PDF files to the existing system, it might be important to annotate these files in the future as well.

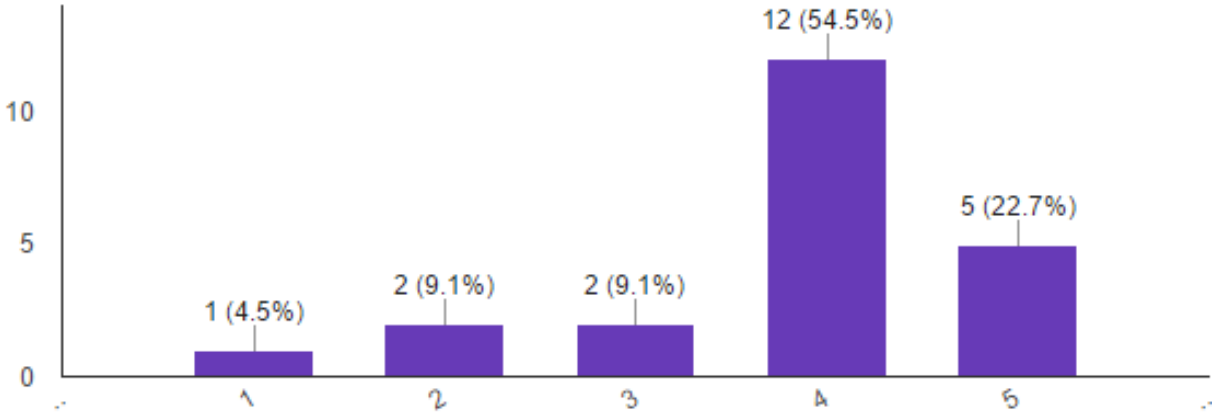


Figure 3.11: Survey – Results of requirement 11: Support the annotation of PDF files

12 Support the annotation of DICOM files

Original German question for the participators: "Die Annotation von DICOM Dateien ist... (1=Überflüssig / 5=Extrem wichtig)'

The results in figure 3.12 show there is a very high peak at three points. Only a few participants assigned four or five points. DICOM images are usually taken by doctors only. The images consist out of many different layers. A special viewer has to be included in order to scroll through the different layers. Not all the medical staff of the target group which was asked is experienced with DICOM images. Nevertheless, there is still an interest of some participants. With consideration of people who do not have any experience with DICOM images, the demand of DICOM annotation support is high.

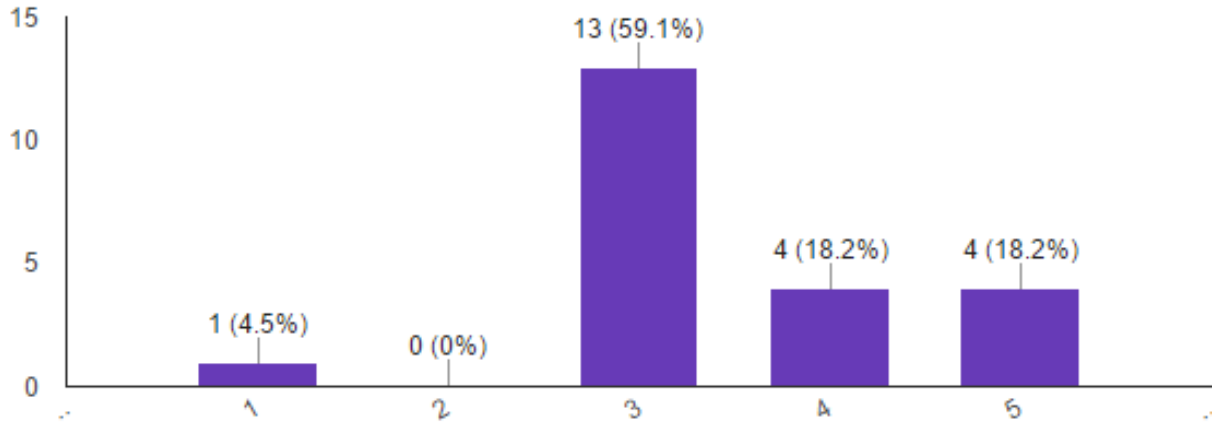


Figure 3.12: Survey – Results of requirement 12: Support the annotation of DICOM files

13 Support common videos formats

As figure 3.13 shows the participants are very undecided in the support of video annotation. It is also interesting to see that there is only one person who thinks that this feature is extremely important. Many people might not understand the benefits of annotating videos in real time. The reason for this is that uploading a video is not the most common case. As there are also 5 people who assigned four points as importance, some participants still think that this functionality could be important in some cases.

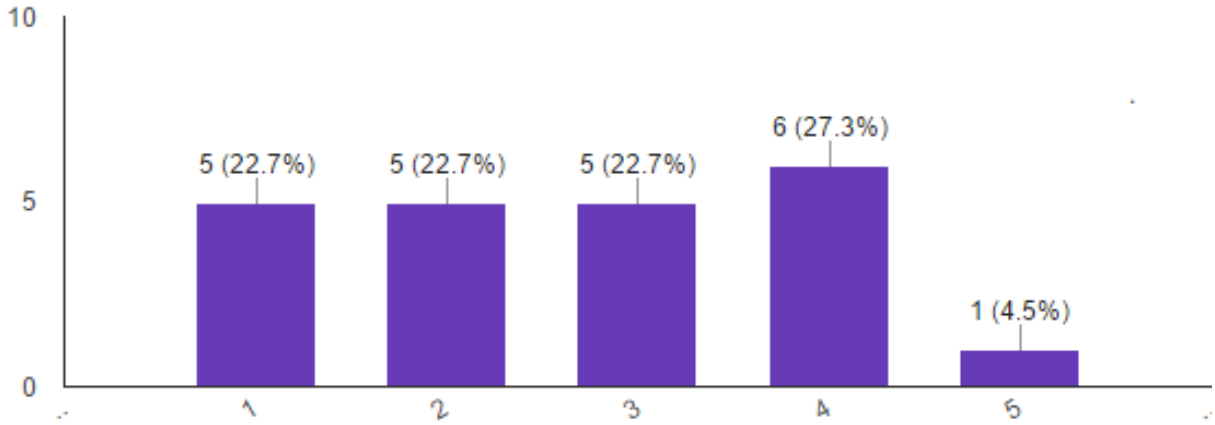


Figure 3.13: Survey – Results of requirement 13: Support common videos formats

14 Support files embedded via plugin

Original German question for the participators: "Das Einbetten eines externen Programms, wie z.B. OsiriX oder Sidexis, und das Teilen per Video Stream (Beispielsweise das Übertragen von Röntgenbildern) ist... (1=Überflüssig / 5=Extrem wichtig)"

People were quite undecided about the importance of including and sharing external software, as figure 3.14 shows. However, nobody of the participants think that including external software is absolutely unimportant. As all other bars almost have the same height, it seems that some people think it could be important and many people are not sure if it will be extremely important for them.

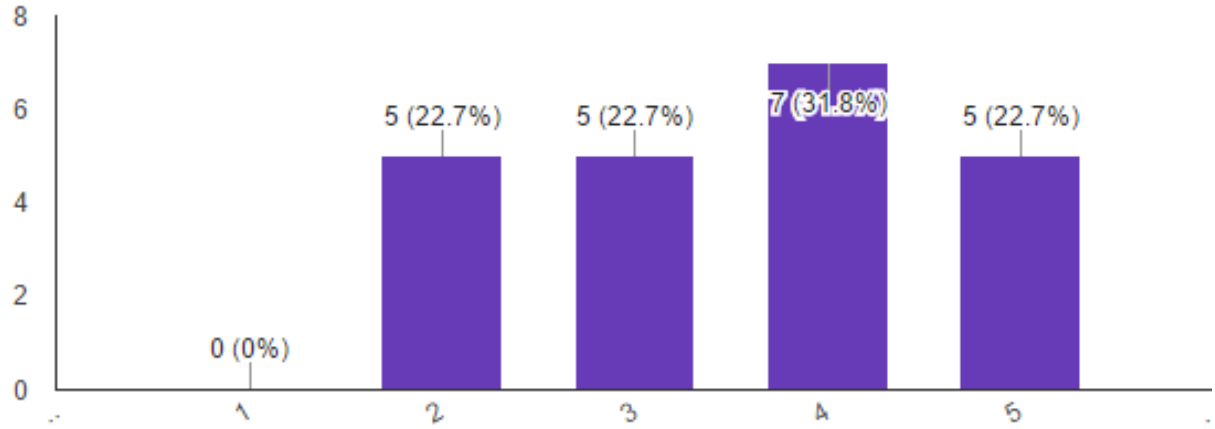


Figure 3.14: Survey – Results of requirement 14: Support files embedded via plugin

15 Store author information

Original German question for the participators: "Den Autor (Ersteller) einer Annotation zu sehen, ist ... (1=Überflüssig / 5=Extrem wichtig)"

The result in figure 3.15 looks like a normal distribution with $\mu = 4$. This could mean that there definitely is a need for displaying the author of annotations. However, only five out of 22 participants think that displaying the author is extremely important.

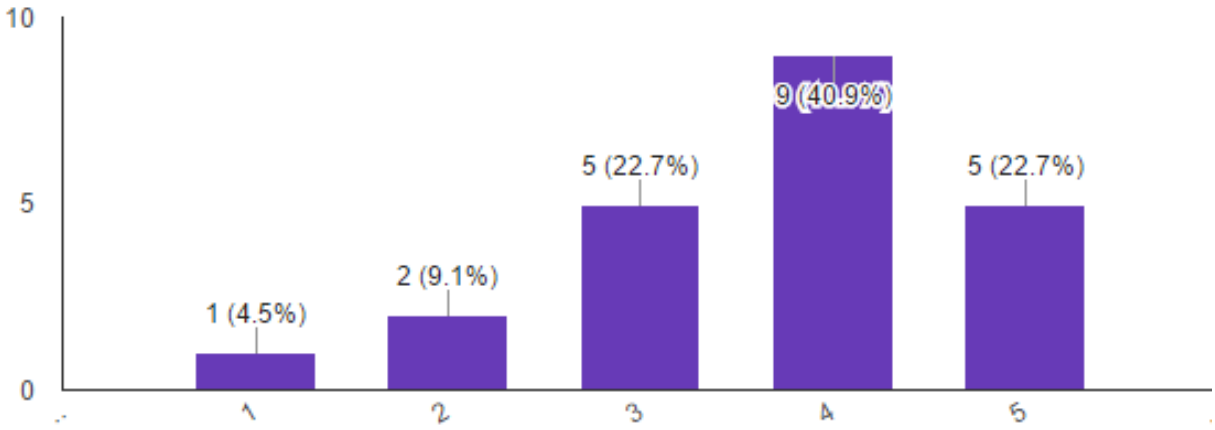


Figure 3.15: Survey – Results of requirement 15: Store author information

16 Download annotated file

Original German question for the participators: "Die annotierte Datei downloaden ist ... (1=Überflüssig / 5=Extrem wichtig)"

Figure 3.16 shows that downloading the annotated file is important at least for 15 out of 22 people very or extremely important. As seven participants only assigned between one

and three points there are some people who would probably almost never download an annotated file.

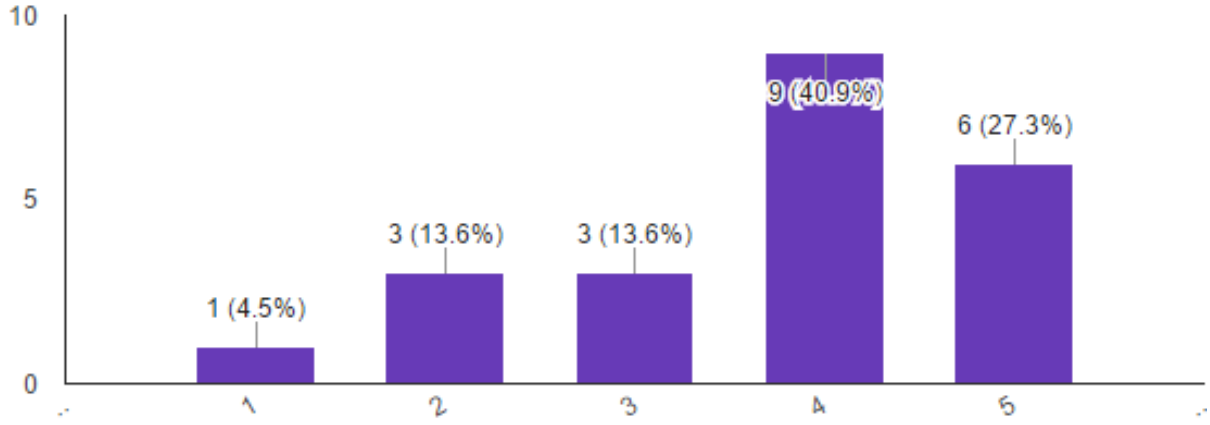


Figure 3.16: Survey – Results of requirement 16: Download annotated file

17 Replay the annotations

Original German question for the participators: "Den Annotierungsprozess erneut (wie ein Video) abspielen ist ... (1=Überflüssig / 5=Extrem wichtig)"

The bar chart in figure 3.17 shows that the participants were very undecided about the importance of replaying annotations. Considering the bars of four and five points it seems like there is a demand for replaying the video. However, there are also several people who are not interested in this feature.

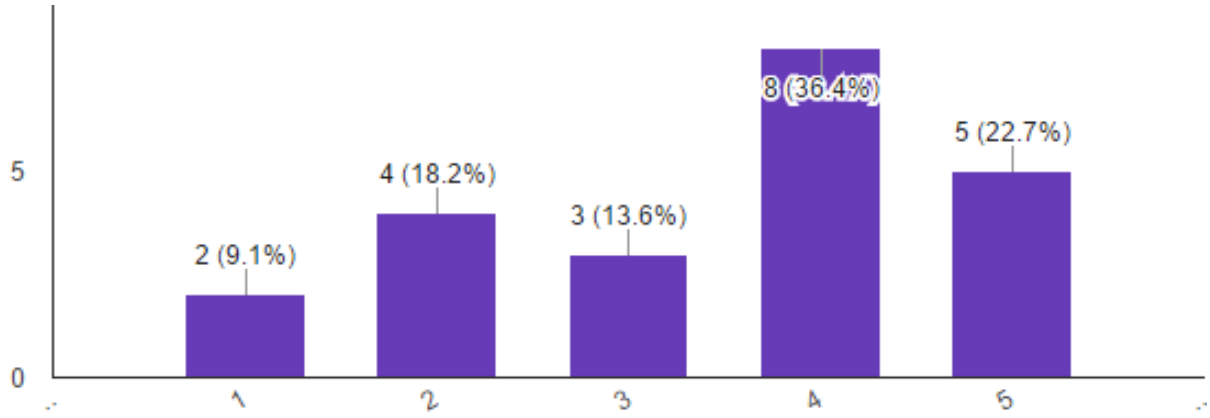


Figure 3.17: Survey – Results of requirement 17: Replay the annotations

18 Consistent color scheme

Original German question for the participators: "Jeder User sollte eine eigene Farbe haben. Das hilft, klar zu sehen wer was geschrieben hat. ... (1=Überflüssig / 5=Extrem wichtig)"

The diagram in figure 3.18 shows that a consistent color scheme seems to be very important to the people. This feature will help to reduce confusion and helps to see clearly who is responsible for each annotation.

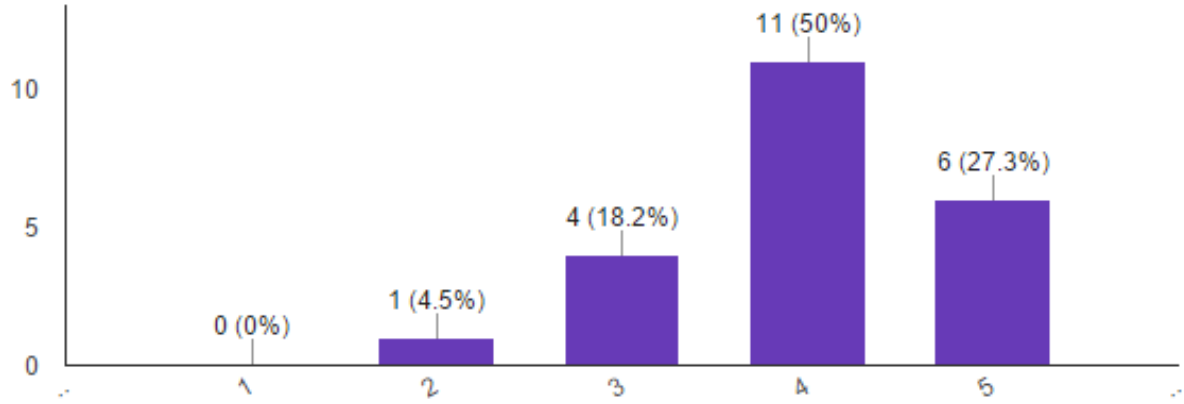


Figure 3.18: Survey – Results of requirement 18: Consistent color scheme

19 Display Participant status

Original German question for the participators: "Den Status anderer Teilnehmer sehen (online, offline, aktiv) ist ... (1=Überflüssig / 5=Extrem wichtig)"

Figure 3.19 shows that the participants were quite undecided about the importance of displaying the state of each user. For some participants the feature would be very important, for others just a little bit. Only one outlier rated this feature as 'unnecessary' which means that there would be a need of this feature.

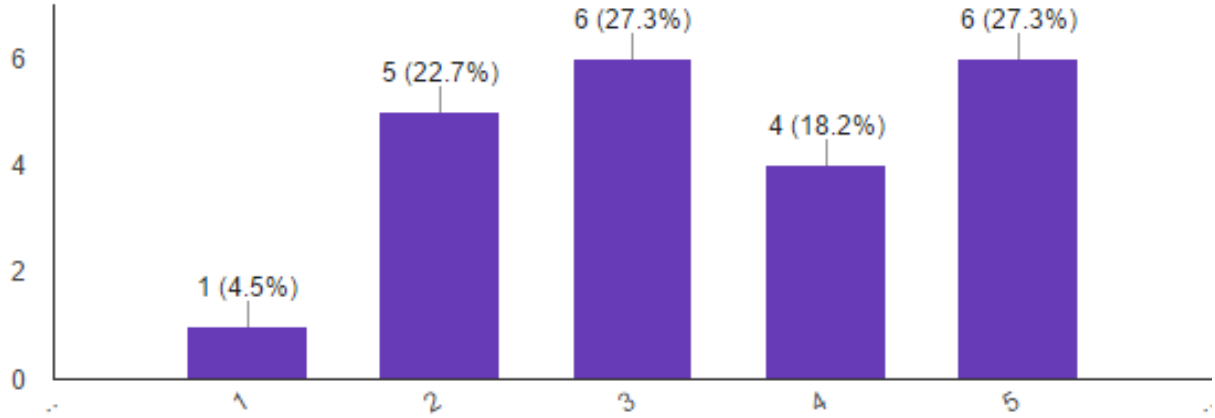


Figure 3.19: Survey – Results of requirement 19: Display Participant status

3.3.3 Conclusion

The survey gave some great insights about the importance of the requirements to real experts. Obviously, the real time synchronization is an indispensable feature for almost all participants of the survey. 18 of the participators assigned four or five points as importance for real time synchronization.

It is very interesting to see highlighting a part of a file seems to be the most important functionality for the participants of the survey. Later, this tool should be placed at a point where the users can see the tool very easy.

Chapter 4

Implementation

In the following Chapter, the implementation of the annotation tool will be discussed. In order to satisfy the requirements which were mentioned before, mock-ups will be created. Furthermore, a well thought through data architecture and a fitting technology stack will be determined which is based on the mock-ups and all determined requirements.

The following chapter will refer many times to the requirements which were determined in the last section. The requirements will be linked by using FR + its number for functional requirements and QA + its number for Quality Attributes. E.g. FR1 means, that the sentence refers to functional requirement 1.

4.1 Mock-up

In the last section different functional requirements based on scenarios were determined. In order to satisfy all the functional requirements, a mock-up has to be created. The mock-up will be used as a base to derive a well-fitting architecture and to support the final implementation process.

Before creating a mock-up, some basic decisions about the design have to be determined. It is fundamental to know the objectives of the mock-up design.

4.1.1 Design objectives

In this part, the main objectives and the main design principles of the annotation tool mock-up will be identified. I.e. it has to be discussed if the mock-up should minimize clicks, maximize overview or maximize possibilities of action. Furthermore, it has to be

determined what are the main elements of the annotation tool. Afterwards, fitting colors and graphical elements to display the collaboration process have to be discussed as well.

In order to **visualize the collaboration** there are several important points to keep in mind.
First, the used colors of the mock-up have to be consistent. I.e. that the colors provide the possibility to derive who created an annotation (FR18). In order to do so, each participant of the collaboration session should have a consistent color, which should be used by him only. Every element, a user creates obtains his unique color. The color should not be used anywhere else. Hence, participants of the annotation session can easily see who created an annotation element.
Second, users need to see the current state of each participant who joins the annotation session. I.e. it is important to see who is offline, who is online and who is actively participating.

Furthermore, the importance of all requirements was rated using a survey. The importance of all these requirements should also be taken into account.

4.1.2 Structure and Design

Basically, the functional requirements consist of the following categories:

- **Tools**, which create additional content over the shared content.
- **Additional tools**, as the replay button or the possibility to turn the screen off.
- **Participants**, who join the current annotation session.
- **A download function**, which enables the user to download a current screenshot of the shared area either with annotations or without.

Consequently, there is an area which should be annotated and four additional categories of functionalities. In order to achieve a separation of concerns, it makes sense to separate all four categories in the app by its locations.

The tools are the only provided functions, which can edit the shared section. Software like Word, Excel, Paint, Adobe Reader have its tools to edit a section placed over the editable area. Thus, users are already used to search tools to edit a section above the editable area, starting from the left side. Furthermore, the functionalities to replay the annotations or to turn the screen off are not used so often, but they are still tools. The additional tools will also be placed on top of the annotated area, but are placed on the right side. This is to separate the additional tools from the main tools and not to place special emphasis on the additional tools.

Furthermore, the other functionalities as the participants or the download button are only used sometimes. It is not important to foreground them so much. Hence, they will be placed at the bottom of the webapp. In order to separate concerns, the participants will be placed on the left side and the download button will be placed on the right side.

Also, the use of a **minimalistic or a minimalistic design** has to be discussed. The different features of the annotation tool which are given by the functional requirements are not so excessive.
On the other side, just placing all the different functions around the annotation area might confuse users. Especially, if there is no need of using all tools with the same frequency. Hence, doctors were asked in a survey what are the major tools during an annotation session.
Furthermore, two different mock-ups will be created. One with all functionalities and another minimalistic one. The minimalistic mock-up contains all functionalities which were stated in the requirements as well. Only the major tools are displayed.

4.1.3 Importance of requirements

The first part of the survey shows that people mainly understand most of the requirements. Some requirements, as annotating a video or turning off the annotation screen might be not so important or even confusing. Anyway, some functionality might be used more frequently than other.

Storing the author's information of each annotation seems to be very important for most of the experts who participated in the survey (FR15). But displaying the name of the author over every annotation he did could confuse the participant, as well. FR 18 says that there should be a consistent color for every user. Maybe a consistent color would not be enough for every user to identify the user. The fitting solution here would be to display the author's information only if the annotation element is hovered. Therefore, the name of the author wouldn't destroy the clean overview of the annotations.

The survey also showed, that some functionalities would not be understandable. In order to give the users a better understanding of the different functionalities all selectors should have a logo which is easy to understand. Furthermore, tooltips should occur over each tool while hovering it.

4.1.4 Minimal and maximal mock-up design

Doctors were asked in a survey about the importance of several functional requirements of the annotation tool. Based on this decision, two different mock-ups were created. The

first mock-up shows all tools and functionalities which were stated in the functional requirements. The second mock-up only shows the functionalities which were important for the doctors. The other functions are included as well, but are not directly accessible.

The mock-up 1 is included in Figure 4.1. In order to refer all the different requirements, the numbers of each requirement are written besides each functionality, as described in the requirements chapter.

Besides the requirements which have been pictured in the map there are some functional requirements left. Even if they are functional, it is not possible to display them in a mock-up because of they are not graphical or not directly a part of the annotation tool.

- **RQ2 - Synchronize the view with all participants in real time**
  All participants should see all changes, caused by every user in real time.
- **RQ15 - Consistent color scheme**
  Every user should obtain one unique color. This color is used to identify all action which were caused by one particular user. All fonts, lines, and highlights which were created by a user have the same color.

Other requirements are linked to the mock-up, but might be not self-explaining.

- **RQ6 - Delete objects which have been added to the annotated area**
  By selecting the eraser every annotation which was created by the current user can be deleted by themselves. A little symbol for deleting the annotation pops up and can be clicked by the user.
- **RQ11 - Support files embedded via plugin**
  A plugin is used similar to the annotation of images and PDF files. Annotations can just be added like to a regular image.
- **RQ12 - Store author information**
  The author ID of every annotation gets stored. If an annotation gets hovered, the name of the author gets displayed.

Figure 4.2 shows a minimalistic version of the mock-up. Only the functions which are used most frequently, as e.g. the pencil. All the other functions are hidden so that they don't bother the user. If other tools are needed the toolbar can easily be extended. Hence, all tool can be found if they are needed without bothering the user too much.

4.1.5 Evaluation

In order to evaluate the two different mock-ups, 22 experts were asked using a survey. An expert has a medical education and at least three months of experience in using the

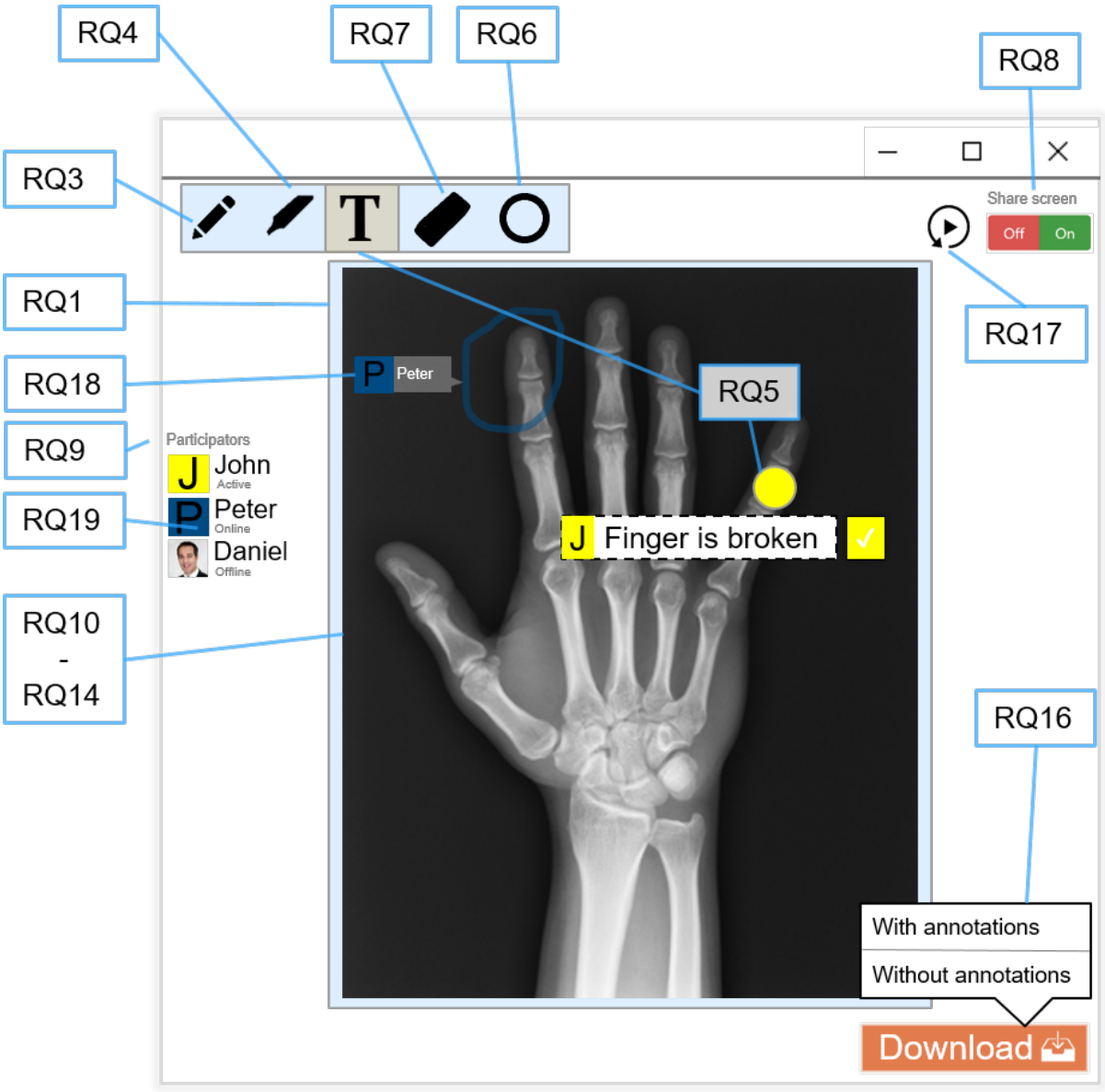


Figure 4.1: Annotation Tool Mock-up 1 – Based on requirements

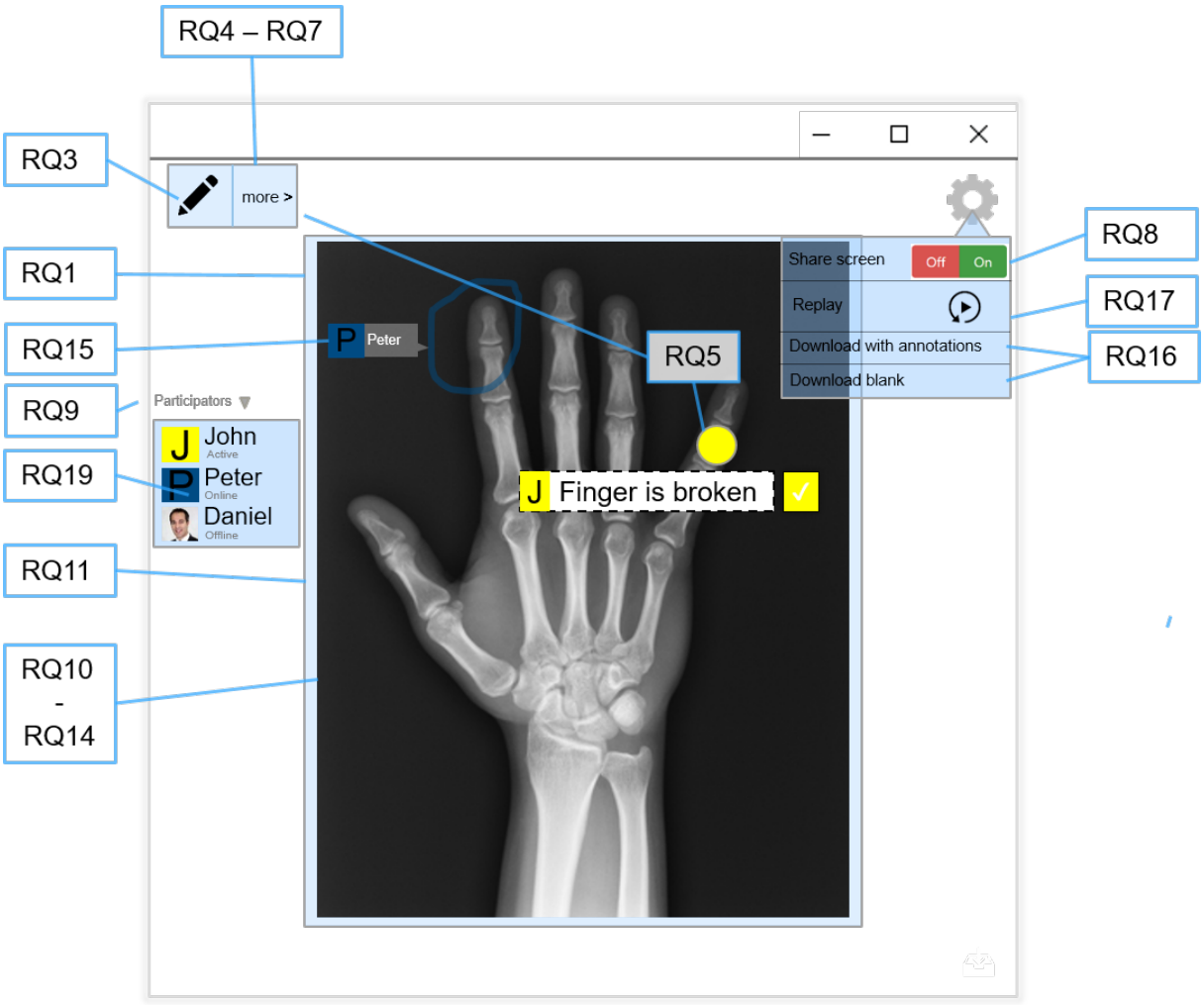


Figure 4.2: Annotation Tool Mock-up 2 – Minimalistic version with just showing the most important functions directly

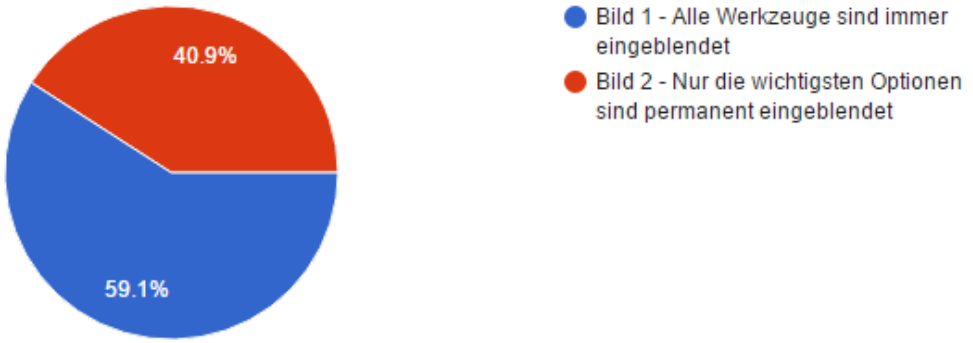


Figure 4.3: Results of the mock-up evaluation – Subjective preference

TeleClinic video consultation software. The whole survey including all its questions is attached to the appendix in chapter 7.

In the beginning, the participants of the survey were asked which mock-up they prefer in a subjective manner. All experts could choose between the mock-ups in figure 4.1 and figure 4.2. The results are shown in the pie chart in figure 4.3. It turned out that 59.1 % of the asked participants prefer the maximized mock-up in figure 4.1. Even if there is a majority of people who like to see all tools, there are nine participants who might be confused if all options are displayed the whole time. One participant wrote the following quote:

> *Ich finde die Option mit den wichtigeren Funktionalitäten besser verständlich bzw. leichter zu überblicken*
>
> *I think that the option which shows only the important functionality is more understandable and more straightforward, respectively.*

This quote shows, some people might be confused if there too many options are displayed in the tool. However, the majority of the participants would not be confused using a tool similar to the mock-up in figure 4.1.

Afterwards, all participants were asked explicitly if the users prefer having many options or only the tools which are most frequently used. The results are printed in figure 4.4. The results of the question show that there are only some outliers who assigned two or three points. As in the subjective perception of both mock-ups, the majority of the participants would also prefer the tool with many options if they could vote. Only 6 people would definitely prefer an annotation tool which displays only the important options. Too many options might confuse them.

The next question was even more concrete. The participants were explicitly asked which options they always want to see. They could decide between all options (blue), all annotation tools (red), only tools who are currently needed (yellow) or no options at

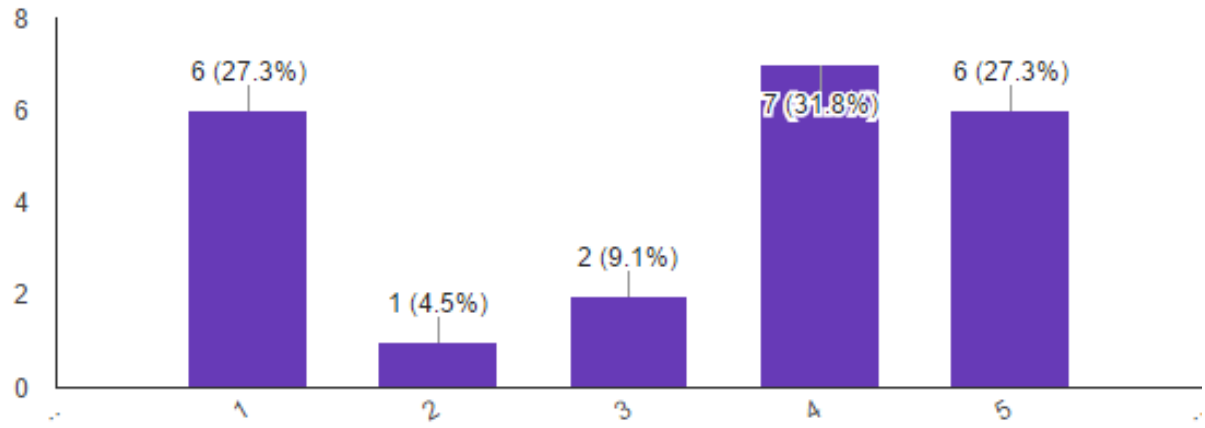


Figure 4.4: Results of the mock-up evaluation – How many options should the tool have?

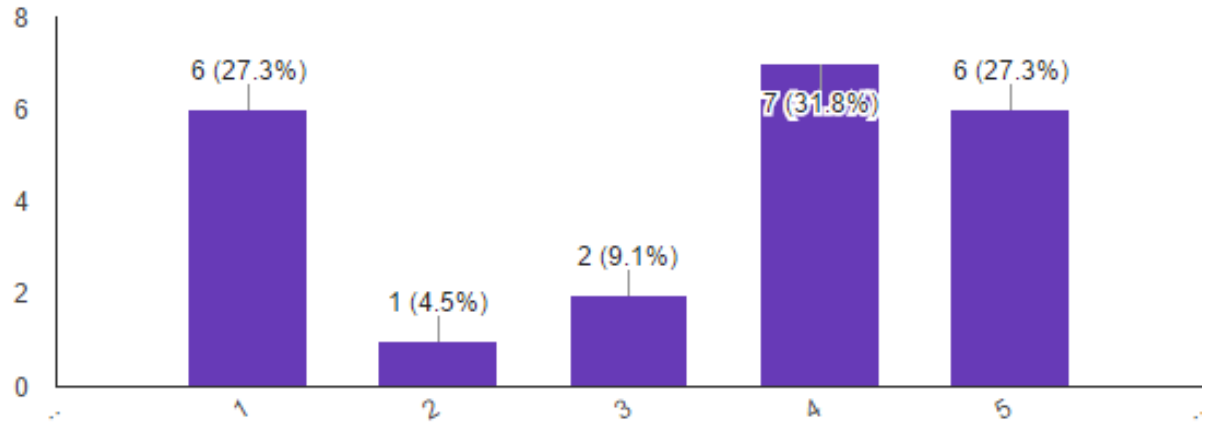


Figure 4.5: Results of the mock-up evaluation – Which option should be displayed?

all without opening an extra menu (green). The results in figure 4.5 most of the people definitely want to see all options or at least all annotation tools. After they know, what kind of options will be displayed there are only a few participants who would still feel uncomfortable having all the options always included.

In the text feedback the eleven participants explicitly wrote that they would understand the tool and think that it would be a sense making supplement. Only three people have reservations regarding the understanding.

Furthermore, an expert wrote to following feedback regarding the function to add shapes:

> *Statt Kreisen würde ich kleine Pfeile verwenden, die auf einen Punkt zeigen. So wird es in gängigen Programmen gehandhabt!*
> *Instead of circles I would use small arrows who point to a section. This is how it is solved in common software.*

The arrows might be more accurate in pointing to a certain section. Another reason for using arrows instead of circles would be that experts are more used to it. Hence, the medical stuff doesn't have to change their behavior so much.

Hence the design of the first mock-up in figure 4.1 will be chosen. As a shape, a little arrow will be used instead of a shape. In chapter 5 the applicability and understanding of the tool will tested with real users.

4.2 Architecture

In this section it will be analyzed, which data is needed in order to display annotations. An optimal way of representing the annotation data in models will be derived. The objective of the architecture should be to keep the data which gets generated as small as possible. Furthermore, a way has to be determined to broadcast the data to all other participants with a minimized time lack.

4.2.1 Annotation data model

In order to store a minimum of data, it is important to know which data is necessary to store in which annotation element. Regarding to the determined requirements, the annotation tool should be able to add the following elements to a shared area:

- a usual black line
- a highlighted area
- a text field
- an arrow

In the following the minimum needed data will be determined. Based on this data, a UML class diagram will be created which is displayed in figure 4.7. Trivial functions, as getters and setters, are not printed in the UML diagram. **AnnotationElement** – The class is an abstract class for creating different annotation elements. All variables, which are part of all annotation elements will be stored within this class. First of all, it is important to determine which data should be stored once for every annotation element. To identify each element in the database, we need an ID. Furthermore, we need to know who created an element. Hence, we need to store the creator of the annotation element using the ID of the author.

If an element will be deleted, it should not be visible anymore. It might be possible, that users might restore some elements which were deleted. Hence, they just should obtain

Boolean flag.

Every annotation belongs to an existing file. In order to assign the annotations to the file, the ID of the annotated file has to be stored in the object, as well. This will help to assign the right annotations to the file.

Also, all elements store attributes as a color, opacity and a width or size. These parameters can also be retrieved by the annotation type and the creator. Storing this information in the object will make the parsing process of the annotations much easier and reduces the complexity.

The type parameter supports the parsing process of the object. If an object gets loaded from the database, the right subtype of the AnnotationElement can be generated very easy.

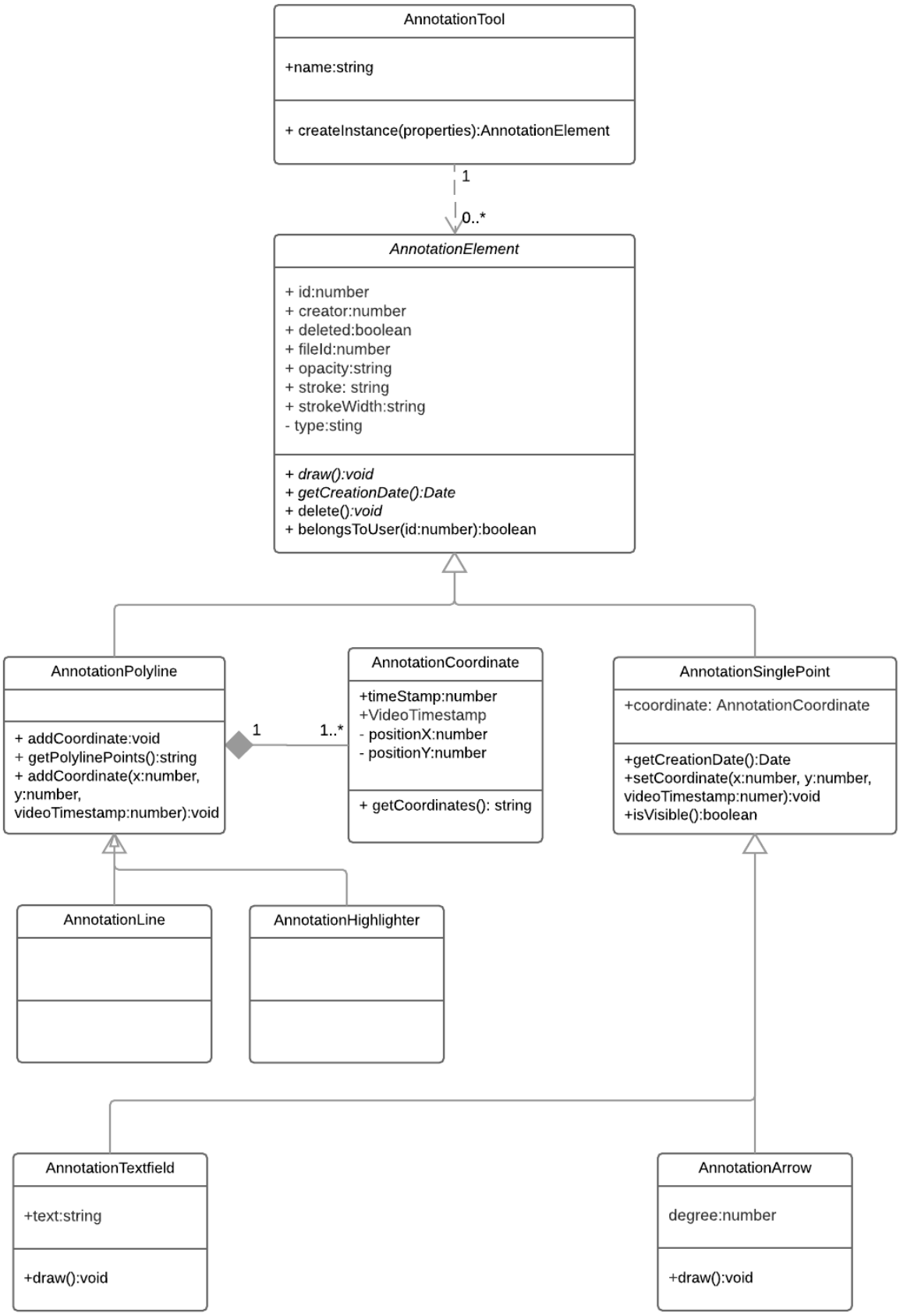
AnnotationPolyline and AnnotationSinglePoint - All annotation elements need to have coordinates. All lines need to have many coordinates, because each line is just a list of coordinates. Text fields and circles only need one coordinate. Hence, it makes sense to create two subclasses – one for elements with many coordinates (called AnnotationPolyline) and another class for elements with only one coordinate (called AnnotationSinglePoint).

In the frontend, both classes should provide a function to get their coordinates as a string. This is needed to generate an SVG graphic which will plot the elements over the annotated file. New coordinates can be added with the function `addCoordinate(x:number, y:number, ?videoTimeStamp:number):void`. Besides the x- and y-coordinates every coordinate has the possibility to have obtain the timestamp of a video. If a video is annotated, this function might be useful in order to rewind the video including its coordinates.

AnnotationLine and AnnotationHighlighter – Both classes are basically lines. The main difference is that the highlighter is thicker and has some opacity. Both Elements could also be stored within one class without storing redundant data. Using two classes helps to get a better understanding of the different annotation elements and to separate them.

AnnotationTextfield – This class stores text as additional data. It heritages from AnnotationSinglePoint because it has only one coordinate.

AnnotationArrow – This element also has one coordinate. An arrow is a shape which can be placed everywhere on the image. The main difference to other elements is that an arrow has to point to a certain section. In order to do so, also a degree of the arrow has been stored.


AnnotationTool
+name:string
+ createInstance(properties):AnnotationElement
1
0..*
AnnotationElement
+ id:number
+ creator:number
+ deleted:boolean
+ fileId:number
+ opacity:string
+ stroke: string
+ strokeWidth:string
- type:sting
+ draw():void
+ getCreationDate():Date
+ delete():void
+ belongsToUser(id:number):boolean
AnnotationPolyline
+ addCoordinate:void
+ getPolylinePoints():string
+ addCoordinate(x:number, y:number, videoTimestamp:number):void
1
1..*
AnnotationCoordinate
+timeStamp:number
+VideoTimestamp
- positionX:number
- positionY:number
+ getCoordinates(): string
AnnotationSinglePoint
+coordinate: AnnotationCoordinate
+getCreationDate():Date
+setCoordinate(x:number, y:number, videoTimestamp:numer):void
+isVisible():boolean
AnnotationLine
AnnotationHighlighter
AnnotationTextfield
+text:string
+draw():void
AnnotationArrow
degree:number
+draw():void

Later, the code should be provided as an Angular 2 library. Users should have the possibility to import certain classes. In order to prevent the user from having different classes with a similar name, all classes in the annotation tool will receive the prefix "Annotation", e.g. AnnotationPolyline.

4.2.2 Backend, Service and Component interaction

The data of the file, its meta data and its annotations will be fetched and also stored in a backend. In the following part the interaction of the data and the role of every component will be explained. The architecture is derived in order to fulfill the Quality Attributes which were determined in chapter 3. Especially the following requirements are important in order to create a powerful and reliable architecture:

- **QA1 - Modularity / Reusability in all connected systems.** All parts of the system should be reusable. Several data, as e.g. the files which will be annotated is used displayed in many different components all over the software. E.g. pictures will be visible in the user's profile, in an upload directory and as thumbnails before loading the file in order to annotate it.

  Furthermore, services, components and endpoints might be also used in other systems. For instance, the annotation component itself should be used in different applications. The user who calls a doctor uses another software than the doctor for receiving the call. The reason for this is that the doctor has different tasks to accomplish than the user who calls the doctor.

  Therefore, all parts of the system should be as modular as possible. It should be easy to use services and components in different parts of the project or even in another project with high cohesion and low coupling.

- **QA6 – Robustness** In order to ensure robustness, every component must ensure high cohesion. This means, that every bug that occurs can be tracked and it is easy to determine the responsible component and function very fast. Also, the functions should be very small and should be named properly. In the case of a bug the function call stack in the error log should be understandable very fast. It should be obvious to see which small function is responsible for the bug so that it can be fixed very quick.

- **QA7 – Scalability** The architecture should be chosen in order to be able with a huge data input. The whole application should be able to deal with one million active users in a few years. Hence, it is important to focus on scalability from the beginning even if the app is a prototypic implementation.

- **QA13 - Low effort of implementation** Problems should be solved without wasting too much time. Also, problems should be solved as simple as possible. A detailed architecture will reduce the complexity, less bugs will occur and time will be saved.
- **QA15 - Separation of concerns** Every part of the system should have a clear concern. This helps the programmer to save a lot of time finding functionalities or debugging software. Also, the whole architecture becomes more understandable for everyone.

In order to achieve an optimized architecture, different concerns have to be determined. Based on these concerns, different components will be derived. The following concerns are necessary:

1. **Retrieve and store data**
   (a) Store files and file meta data

   An RESTful endpoint for uploading and retrieving files including their meta data is already existing and is given by the company's backend. Regular files and a JSON object with meta data can be uploaded and retrieved as usual.

   (b) Store annotations and annotation meta data

   Additionally, the annotations of the different files have to be stored. There is no endpoint or anything similar given, so far. In order to find a perfect strategy for storing the data and synchronize it in real time the best fitting method for building an endpoint will be determined in section 4.3.4. The end point might be not a typical rest endpoint, because it has to be synchronized in real time with other participators.

2. **Parse, manage and synchronize data**
   (a) Manage file and meta data

   The frontend application just needs a service which can load files, store them and update them. Other components should just be able to subscribe data from this service and trigger the download of new data. The services are mainly necessary for two reasons: A separation of concerns and reusability. Other components also need information of the current file or upload a new one.

   (b) Manage annotations and annotation meta data

   After building a backend for synchronizing the data, there has to be an **AnnotationService** which is responsible for the creation and synchronization

of the annotations for a file. Components who plot the annotations or render it into a file can subscribe parts of this service and receive or trigger updates.

Hence, two services have to be created: The **FileService** and the **AnnotationService**. Both services will be responsible for fetching, caching, updating and synchronizing data. So, only one instance of the data exists all over the project. If the data gets changed, all components who subscribe the data from the service will receive an update immediately.

3. **Display data and read new data input**

   In genral, the components are used in order to display the data which has been loaded from a service. If a data update gets triggered in the component.

   (a) Display the file itself

   Many different file types should be supported by the annotation tool. For this concern a component is needed which just pulls the current file from the service. Hence, including files with the HTML5 `<object>` is not enough. E.g. DICOM files have to be loaded with a special player. Furthermore, videos have to emit some meta data about the video as the current timestamp. The current timestamp of the video will be added to each annotation in order to replay the annotation later.

   (b) Plot the annotations and listen for new ones

   Except for the file, also the annotations have to be plotted. The annotations should be editable, removable and redrawable. Hence, the annotations should be not "really" rendered into the file and be overlaid the file. Only, if the file gets downloaded the annotations should be rendered into the file.
   So, the annotations are a completely different concern than displaying the file. The annotations should overlay the file. Furthermore, new annotations should be displayed, they should be editable and removable and new annotations should be added by the user. As this is all one concern, a second component should be built for this concern.

   (c) Wrap the annotated file and supply additional functions

   The AnnotationComponent might be reused in other situations and can just wrap any HTML content and annotate it. The component is built as a custom HTML element and can wrap all HTML content which shall be annotated: `<ng2-annotations><some-content></ng2-annotations>`. With respect to the reusability of the AnnotationComponent the component for displaying the file should not be directly included in the AnnotationComponent. A wrapper component can solve this problem by including both components. In a very

simplified way the wrapper component connects both components as displayed in the following HTML:

```
<ng2-annotations>
  <dynamic-source [src]="selectedFile.fileURL"></dynamic-source>
</ng2-annotations>
```

Furthermore, some meta information of the file as the title or the description has to be edited, as well. Also, participators of the annotation session should be displayed within this component.

The interaction of the backend, the services and the components is displayed in figure 4.7. The services push and pull its data from the backend. Components subscribe the latest data from the services. The FileviewerComponent wraps the DynamicSourceComponent and the AnnotationComponent, which plot the file and the annotations. All updates get immediately sent to the services which update the data in the backend and send updates to all components who subscribe it.

The logic of the services and components (which include a controller and a view), is kept similar to the Model/View/Controller pattern. This pattern almost became a semi standard in modern software architecture because it ensures a clear separation of concerns and modularity [Leff 01].

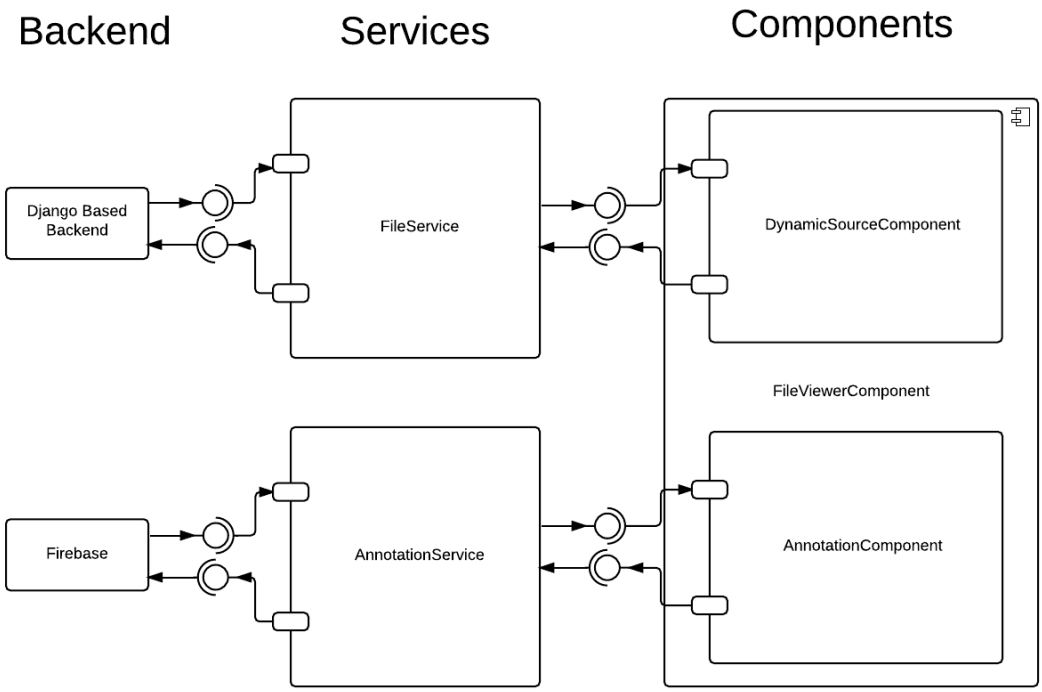


Figure 4.7: Component diagram – Interaction between Backend, Services and Components

4.3 Technology

4.3.1 Frontend technologies

The annotation tool has to be included in the given environment of TeleClinic. The web-frontend for the doctors and for the users is written in Angular 2 and TypeScript. The annotation tool should be provided as a library which can be easily included in the current project. Hence, the library should be written in TypeScript or JavaScript in order to prevent convert errors. TypeScript can be converted to JavaScript very easy and provides a lot of benefits compared to JavaScript. It provides all functionalities of ES6, as the use of classes, and also the possibility of using datatypes. Using classes and datatypes provides the benefit of writing very clear code, minimization of redundant data and a clear separation of concerns [Bier 14].

Regarding the annotation data model which has been derived at the previous section, the use of classes and types helps to achieve a minimum redundancy.

4.3.2 Backend technologies

The backend of all TeleClinic applications is written in Python, using the Django REST Framework. Django is also class oriented and is able to map the same data model which is used in the frontend to the database. The backend already provides an endpoint to upload and download different files including its meta information.

For the annotation a new endpoint or a similar method for storing and synchronizing the data has to be selected. A regular endpoint would not fit with all Quality Attributes. Hence, the best fitting method will be determined in section 4.3.4.

4.3.3 Database

An annotation line can quickly produce hundreds of coordinates which increase while the line is drawn. For each pixel that changes a coordinate will be added. Representing the coordinates as rows means that hundreds of rows have to be inserted into a database in very short time for every coordinate which will be drawn. Also, there are a lot of relations between coordinates, the annotation element itself and the instance of the annotation element (e.g. a line or a highlight). Hence, and in order to handle huge volumes of data it will be more performant to store the data in a NoSQL database [Catt 11].

4.3.4 Data transmission

4.3.4.1 Requirements

One critical part of the implementation is the transmission of shared data. There are several Quality Attributes which have to be fulfilled by the data transmission method. The quality requirements which are given by the environment are also depended on the transmission protocol:

- **QA2** - Run within the browser environment
- **QA3** - Deal with bad connection quality
- **QA6** - Robustness
- **QA7** - Scalability
- **QA9** - Encrypted data transfer
- **QA10** - End-to-end connection
- **QA13** - Low effort of implementation

- **QA14** - High Efficiency

In the following subsections different standard of transmitting the date and operating on shared data in real time will be discussed in matters of fulfilling the above requirements.

4.3.4.2 Possible methods

Regarding to the Quality Attributes which were stated above, a possible method for transmitting and operating on shared data in real time must be determined.

- **HTTP polling**
  HTTP polling is a regular GET request which continuously gets repeated. If the GET request gets repeated every seconds, the whole data which is pulled gets refreshed every time. However, HTTP polling causes a lot of unnecessary HTTP request. That's redundant work for the server and often there is no need for refreshing the data.
  The work of Victoria Pimentel and Bradford G. Nickerson shows that Websockets are solving these two problems and can be even faster than HTTP polling [Pime 12]. Since the use of HTML5, HTTP polling is considered to be deprecated [Wang 13]. However, the advantage of HTTP polling was the support old browsers which don't support HTML5. That's the main reason why HTTP polling is still used sometimes [Lore 11].

- **WebSockets**
  WebSocket is a protocol for sending data that has been changed to clients using TCP [Meln 11].
  Whenever a session starts, all existing data gets downloaded from the server using a regular GET request. Afterwards, all data that changes gets posted to the server using a regular POST or PUT request. Whenever the server receives new data the backend stores the data in the database. At the same time, the server starts to send an update during a WebSocket to all participating clients. Only the delta gets transmitted, i.e. the data that changed compared to the previous emitted data [Meln 11] [Pime 12].
  The papers of WebSockets: Spezifikation / Implementierung and Communicating and Displaying Real-Time Data with WebSocket show that the implementation effort is very low. Setting up a WebSocket connection which listens to incoming strings can be realized with less than ten lines of the code. The disadvantage of this method is that only the delta of the data gets transmitted. I.e. that all clients have to build the data stack themselves over time. This is a lot of implementation work and prone to data lose or other errors.

- **Firebase**

Firebase is a mobile platform that delivers a lot of additional features for the development process. The **shared database** of Firebase extends the idea of operating on shared data even more. A shared database gets established on which all participating users collaborate in real time [Fire 16b]. This makes operating on data very easy and is very fast, as well.
Firebase is using latest security standard and has included its own authentication system. The security system provides a very simple language for defining access rules and is also scalable, fast, concise and simple. Hence, assigning rights to shared data is very simple and absolutely dynamic [Fire 16c].

Furthermore, Google provides a backup of the database and an easy way to synchronize the database with the backend [Fire 16a].

- **WebRTC**
  WebRTC is a collection of communication protocols for exchanging data via peer-to-peer. In comparison to WebSockets there is no backend required. All data gets directly transmitted from browser to browser [John 12]. The server can still access the transmitted data as a listener, using WebSockets or XMPP protocol. So, data can still be processed by the backend and saved to a database. The problem here is that one client has to be responsible for transmitting the data to the server. As the client might change over time, other clients have to be selected to operate as a listener for the server. This is no problem in a logical manner, but a lot of extra implementation effort. Also this approach might be prone to errors.
  Furthermore, WebRTC is designed to transmit videos as a shared screen [Prez 12]. This encourages the fulfillment of the requirement to embed external plugins and stream them as a video.
  The WebRTC is fully end-to-end encrypted by default using Datagram Transport Layer Security (DTLS) [John 12].

4.3.4.3 Summary and final decision

In conclusion, all four methods could solve the problem of collaboratively creating annotations. But no method satisfies all Quality Attributes which were stated before. The following table shows how the certain methods satisfy the stated requirements:

No.	Requirement	HTTP polling	WebSocket	Firebase	WebRTC
QA2	Run within the browser environment	✓	✓	✓	✓
QA3	Deal with bad connection quality	✓	✓	✓	✓
QA6	Robustness	✓	X	✓	X
QA7	Scalability	X	✓	✓	✓
QA9	Encrypted data transfer	X	X	✓	✓
QA10	End-to-end connection	X	X	X	✓
QA13	Low effort of implementation	✓	X	✓	X
QA14	High Efficiency	X	✓	✓	✓

The major problem is the complexity and security of some methods. Fixing these problems is possible but takes a lot of time. Firebase provides all of this by default. The only thing that is not provided is an end-to-end connection. This requirement was only given by the scenarios which were derived from the company's expectation. The main reason for the demand of an end-to-end connection was because it sounds safe. However, Firebase ensures a secured connection without the need of an end-to-end connection. An end-to-end connection wouldn't add additional security, as it is still prone to e.g. man-in-the-middle attacks. An encrypted connection is still ensured by HTTPS, even if there is a Database in the middle [Duru 13].

4.4 Actual implementation

In this section the actual implementation of the annotation tool will be described. In order to describe the concepts which were used no code will be displayed. Instead of displaying code, sections of the code and used patterns will be described using UML diagrams.

4.4.1 Frontend implementation

As the whole frontend of the business application, the frontend of annotation tool will be implemented in TypeScript, using the Angular 2 Framework.

TypeScript is an extension of JavaScript ES6 and allows developers to implement the exact structure of a class diagram [Bier 14]. Basically, the whole data model which was derived in the class diagram of figure 4.7 can be implemented using TypeScript classes. The primitive datatypes are provided by TypeScript and functions can be implemented very similar to JavaScript.

Furthermore, Angular 2 uses the concept of components and services. A service is responsible for fetching data from the backend, store it and update it. Components fetch

the data from the services and plot the data in HTML files. Furthermore, objects can be created within a component, but will never post data to the server. They can easily use the classes, print the annotation elements in the view and listen for actions on the view to intermediate between the classes and the view.

The following implementation concepts will be realized according to the component diagram in figure 4.7.

4.4.1.1 Services

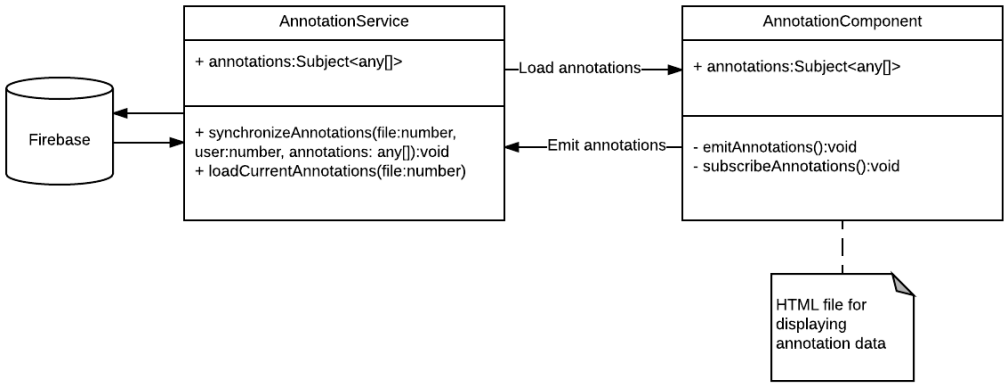


Figure 4.8: Interaction of Backend, FileService and FileViewerComponent

The **FileService** in figure 4.8 is responsible for communicating with the backend regarding all actions on the file itself. The actions are.

- Perform POST, GET and PUT operations on a file.
- Add and change meta information of the file as creator, title and description.
- Create a download link.
  The file will be downloaded from a secure storage. For downloading the file, a token has to be appended to the URL which will be changed every 15 minutes.

Only one file can be opened at the moment. Hence, the service is using a certain type called Subject. A Subject is a variable which uses the observer pattern. Every time, a new active file gets loaded, it gets assigned to the Subject. Components can subscribe the Subject from the service and subscribe its updates. So, every time a new active file gets loaded in the service, all components receive an update and can update their view.

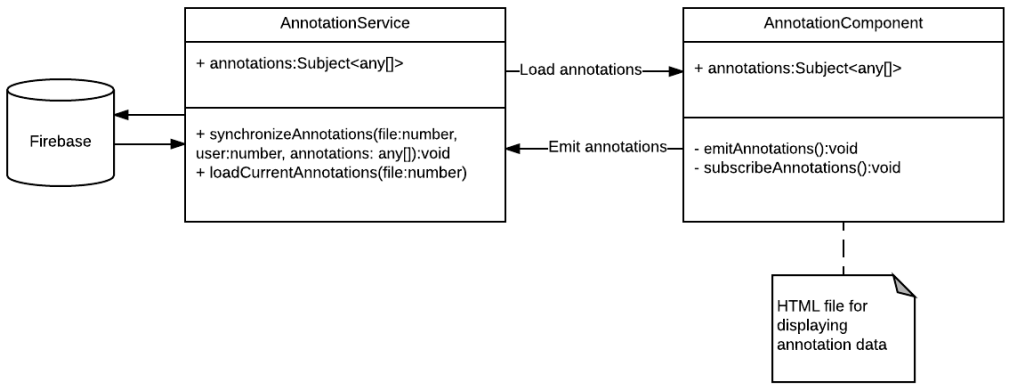


Figure 4.9: Interaction of Firebase, AnnotationService and AnnotationComponent

The **AnnotationService** is responsible for storing the annotations of a certain file into the database, fetching annotations of a file and synchronize them in real time with the database. If a new file gets loaded, the annotations of this file can be loaded from the service. In this case, just one function should be called with the file ID as parameter. The annotations which are currently loaded will be emitted to a Subject, as well. Components which display the annotations will just subscribe the variable and automatically receives the new annotations if the Subjects gets updated.

The service is also responsible for synchronizing the annotations of the current file with the database. The relevant functions and interaction with the database is displayed in figure 4.9. As a database for storing the annotations, Firebase will be used according to the results of section 4.3.4. Firebase offers an Angular 2 library which makes the synchronization of the data very easy. Every time, the annotations will be sent to the annotation service, all annotations get passed to a method of the service as a JSON array. The JSON array will be synchronized with the database automatically by the library. After the annotations get sent to the service, a function of the library gets called which send the annotations to the database. Furthermore, the library provides a Subject, which is subscribed by the service. Every time, the database receives an update, the Subject emits the current dataset. All components who subscribe the annotations from the service will receive an update about all changes in the database immediately.

4.4.1.2 DynamicSourceComponent

This component is used to display different kind of files by just using a custom Angular 2 directive. The directive is used analogically to HTML5 tags as `<object>`, `<video>` or

<img>.

```
<dynamic-source [src]="video.mp4"></dynamic-source>
```

The benefit of the custom component is to include additional file types as DICOM images, PDFs and videos and to emit meta information about the current file. Meta information could be the current time of a video which is currently played or the scroll offset of a PDF file which is currently displayed.

Also, the component enables to communicate with custom video control elements as a play/pause button and a timeline. The source will be overlaid by an SVG image which is responsible for displaying the annotations over the displayed file. As the native HTML video player embeds the controls inside the video, the controls cannot be used because only the overlaid SVG graphic which displays the annotations would be clicked [Anth 12] [Lubb 11].

4.4.1.3 AnnotationComponent

The AnnotationComponent will be used to display annotations over a certain input and to listen for new annotations. As the tool should be reusable and not be dependent from other components and services, the component should be usable independently. In order to synchronize with services, the meta information of the DynamicSourceComponent and in order to display author and participant information, the component should have several optional input parameters. The use of the component should be as easy as possible and just be usable as a custom directive.

All attributes of this directive are optional. They get assigned to variables. If there is no input, a default value will be assumed.

Basically, the component has an HTML file which contains an empty SVG image only. Since 2001 SVG is an official recommendation of the W3C for displaying vector graphics. Furthermore, it supports drawing lines and different shapes by just creating and XML element of the type and use the coordinates as attribute. Since SVG became a standard of HTML5, it can easily be styled within Angular 2 applications using CSS [Dahi 01] [Poma 15].

The AnnotationComponent receives a custom HTML input which could be anything of content. All annotations will be added to the SVG graphic which is just an overlay of the annotated content. All annotation elements which will be loaded as input parameters create SVG elements, as a line with coordinates, color and thickness, a text field or a form.

On the other side, there is an event listener on the whole component. Every time, the

mouse gets clicked over the SVG image, an annotation will be added according to the selected tool. The following sections will explain how the creation of annotations works in detail.

As explained, the annotation component should just be a directive with optional variable inputs. It looks like the following:

```
<ng2-annotations
  (annotationElements)="synchronizeAnnotations($event)"
  [synchronizingUpdates]="synchronizingUpdates"
  [fileId]="selectedFile.id"
  [annotationSet]="_annotationsService.currentAnnotationSet"
  [myAccount]="myAccount"
  [currentVideoTime]="source.currentVideoTime | async">

  <!-- Content that should be annotated -->

</ng2-annotations>
```

All attributes within the round brackets are output events which will call a function outside of the component. All attributes with the cornered brackets are input variables. They are used to pipe information into the annotation component.

The attribute **annotationElements** calls the function `synchronizeAnnotations(annotations:any[])` every time an annotation update happens on the annotation component. It emits an JSON array of all annotation elements. The called function pipes the annotation elements to the AnnotationService, which is responsible for saving the data in the shared database. All clients of the shared database immediately receive an update.

The attribute **synchronizingUpdates** is a regular Boolean variable. It gets piped into the component. The variable is subscribed by the AnnotationService and is always true while the service is synchronizing with the database. The AnnotationComponent displays that the annotations are currently synchronizing if the variable if true.

The attribute **fileId** is just the id number of the file that is currently annotated. If the attribute is filled, every annotation within the JSON array gets an additional field called 'fileId'. This field is used in order to connect the annotated file and the annotations. If the annotations of a file should be loaded again, later, the service can easily search for the annotations by the file ID.

The attribute **annotationSet** is a JSON array of all existing annotations. Every time, annotations get updated by the AnnotationService, an update gets sent to the component through this attribute by just updating the variable. The AnnotationComponent redraws

the whole SVG image immediately.

The attribute **myAccount** is an JSON object which contains the id of a user and his user name. This information will be added to every annotation he creates. So, his name can be displayed over an annotation if is hovered. Furthermore, the id is used in order to identify if an annotation belongs to this account. Only if the annotation belongs to the account, he obtains the right to delete the annotation. This is also validated in the backend. As JavaScript runs on the client side and is not compiled, it can be always manipulated. Hence, critical parts always have to be validated on server side, as well [Trip 11] [Yu 07].

The attribute **currentVideoTime** pipes the current video time into the annotation component if a video is played, as a source. The timestamp of a playing video can be added to the annotations; thus the annotation can be replayed according to the video in real time.

4.4.1.4 Adding text

Adding a text is very straight forward. If the "text tool" is selected and the left button of the mouse is clicked in the SVG image, a function gets triggered. The function just gets the current coordinates of the mouse on the SVG image. These variables are provided by the Angular 2 framework. Now, a new element AnnotationTextfield will be created and pushed to the array of all annotations. Now, Angular 2 afterwards, all text fields which are in the array of the annotation elements gets added to the DOM (HTML file). Angular 2 provides functionality to do this with just one line of code for all text input fields.

Additionally, the coordinates get added to the AnnotationTextfield. With CSS3 the HTML text field gets moved to the position of the coordinates.

4.4.1.5 Drawing lines

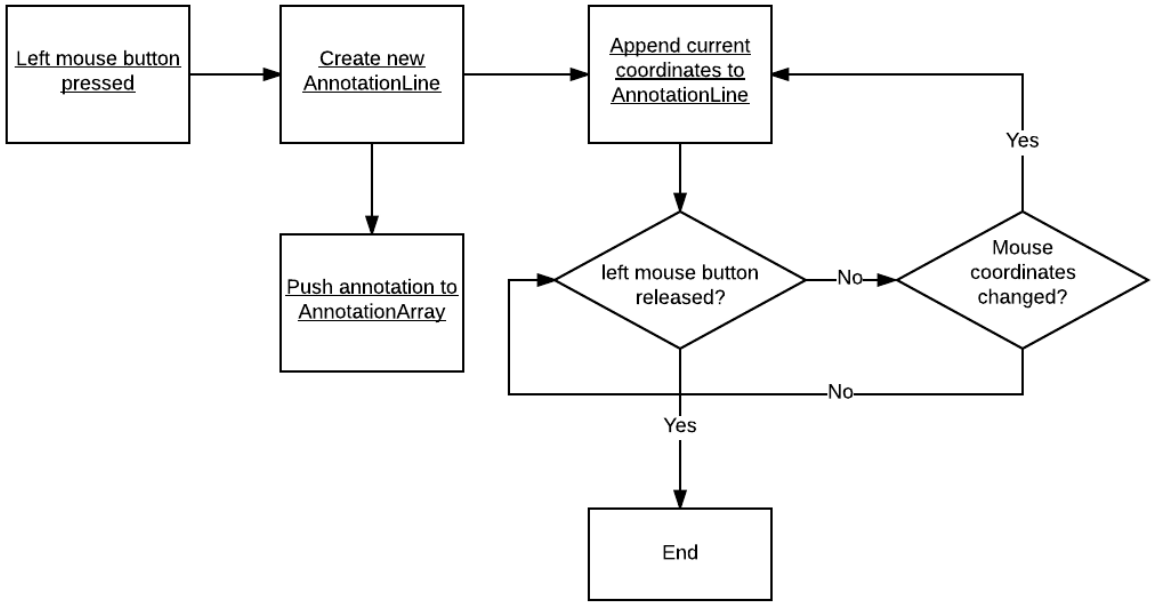


Figure 4.10: Draw a line on a file – How it works

As showed in figure 4.10, the annotation component is listening for mouse events. Every time, the highlighter or the pencil is selected, the coordinates of the mouse on the content gets recorded. If the left mouse button gets pressed down, a new AnnotationLine element get created. Until the left mouse button is released, new coordinate object will be created and appended to the AnnotationLine, every time a coordinate of the mouse changes. The line on the SVG will redraw the line automatically after every coordinate which is drawn.

4.4.1.6 Adding an arrow

An arrow should not just occur in an image; it should also point to a certain direction. Common software as Photoshop and PowerPoint places an arrow on the screen after clicking on a certain position. If the left mouse button is still pressed, the user can pull the mouse in another direction to change the angle of the arrow. As this behavior is already intuitive to many users, it will be used here also.

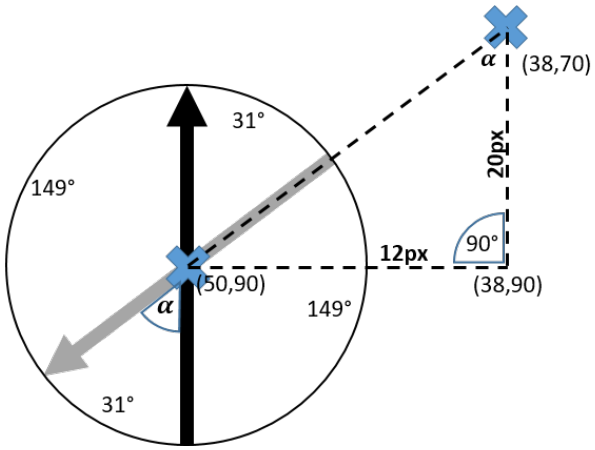


Figure 4.11: Changing an arrows angle based on mouse impact

Hence, the arrow has to turn a certain amount of degrees after the mouse position changes. The example in figure 4.11 shows how to calculate this angle with an example. The arrow should always point in opposite direction of the delta the mouse changed.

Imagine, the mouse clicks on the coordinates (50, 90) on an image. The arrow gets placed on the left blue mark in figure 4.11. Now, the user is changing the mouse's position while the left mouse button is still pressed to the position (38,70). The arrow should now turn in the opposite direction of the mouse change, which is $180° + \alpha$ according to [Koch 02], [Weig 13].

It is important to keep in mind that the coordinates of a digital coordinate system as in figure 4.11 start top left instead of bottom left. Also, the model is not true to scale.

[Koch 02], [Weig 13] also show that alpha can be calculated dividing the opposite leg through the adjacent side. The opposite leg is the adverse edge of alpha and the adjacent side is between the right angle and the opposite leg. The opposite leg can be simply derived from the delta change of the mouse's x axis. Hence, opposite leg $= \Delta x = 12px$. Analog to this the adjacent side can be calculated using the delta of the y axis: adjacent side $= \Delta y = 20px$.

$$Tan(\alpha) = \frac{oppositeleg}{adjacentside} = \frac{\Delta x}{\Delta y} = \frac{12px}{20px}$$

According to [Rapp 01] and [Ried 13] that can be converted to the following result

$$\alpha = arctan(\frac{\Delta x}{\Delta y}) = arctan(\frac{12px}{20px}) \approx 31°$$

Hence, the delta of the mouse position changed and the new angle of the arrow $180°+31° = 211°$. Now, the image of the arrow can easily be rotated 211 degrees using CSS3 {Hogan11 [Frai 12].

The angle has to be calculated again every time, the delta of the mouse position changes.

If the coordinates of the mouse have a negative delta, the angle α will be calculated similar, but has to be subtracted from the 180 degrees. Except for that, the formulas are equal and will not be displayed here.

4.4.1.7 Show participants and their activity

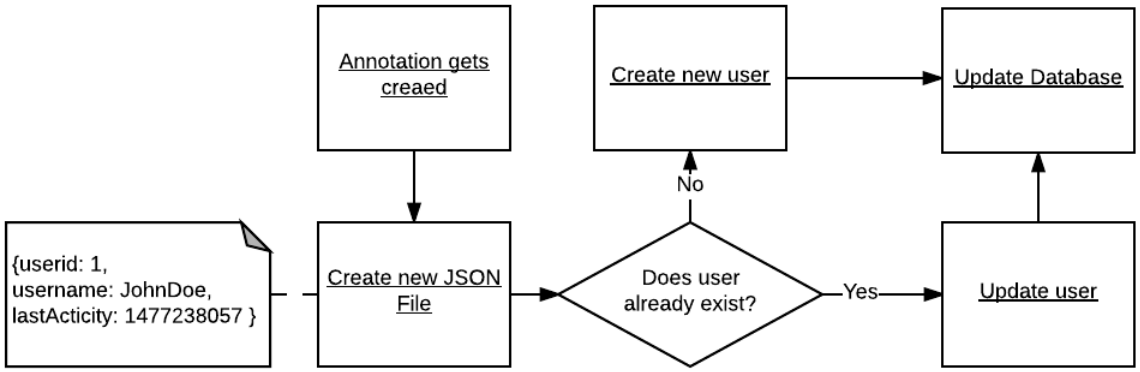


Figure 4.12: Updating the users activity status

Updating the participants of an annotation session or an annotated file is straight forward and displayed in figure 4.12. Every time, a user adds an annotation to a file, an JSON object for the user will be generated. The object contains all relevant information of the user, as his displayed name, user ID and the current timestamp. If the user already exists, the object will be updated in the database. Otherwise it will be added.

The username will be used to display the participator on besides the annotated area with all other participator.

The user ID will be used to assign a user to the annotations he made. As explained before, the name of the author will be displayed over each annotation, if it is hovered by the mouse.

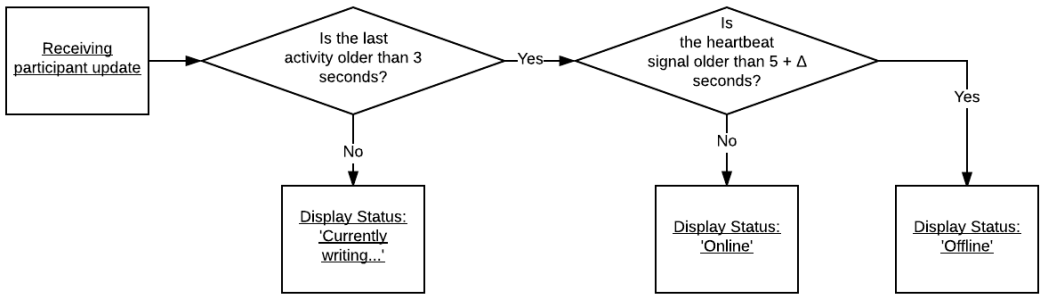


Figure 4.13: Displaying the users activity status

Figure 4.13 shows how to display the users status. The timestamp is used to see the time of the last annotation which was added by the user. As every change of the participators gets emitted to all users who are connected to the database, the participant will immediately see if the timestamp of the last activity updates. If the activity updates, the other users know that the user is currently creating an annotation. Hence, the displayed status of the user changes to "writing" for three seconds. The three seconds are chosen arbitrary in order to display that an activity is going on for a short time period.

Secondly, it has to be checked if the user is still online. In order to do this, he has to emit a "heartbeat" at regular intervals. If the last heartbeat is older than the current time + the heartbeat plus a delta, he will be displayed as "offline". The delta is used in order to prevent delays which occur because of low transmission rated of the internet.

4.4.1.8 Annotating a video

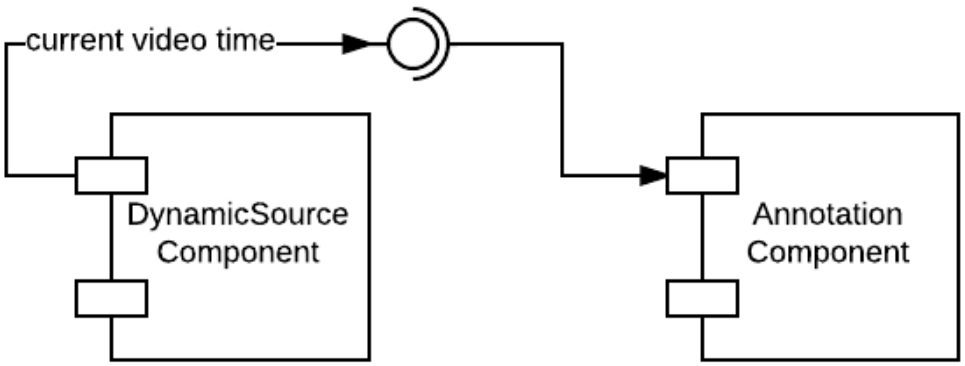


Figure 4.14: Sending video meta data to the annotation component

A video can be annotated in real time and rewinded with its annotation thanks to the meta data of the DynamicSourceComponent. The interaction with the AnnotationComponent is pictured in figure 4.14. The DynamicSourceComponent allows the user to include a video. Also, the component provides the possibility to emit the current time of video, if a video is included as source.

This time gets pipes into the AnnotationComponent. Every time, the user creates a new annotation on a file, the current play time of the video gets assigned to the annotation element or its single coordinates, if the annotation is a line. If the current video time is smaller than the video time which was assigned to an annotation, the annotation will be hidden. If a user replays the video, all annotations will just be visible in the video, if the timestamp is high enough.

Therefore, it looks like the annotations are a native element of the video. If many people watch the same video and one watcher is delayed, the annotation will also just occur when the watcher reached the timestamp where the annotation was added.

4.4.1.9 Annotating a PDF file

Annotating an PDF file is almost like editing an image. The main difference is, that an included PDF file often consists out of many pages and is scrollable. Hence, a new field

should be added to all annotations: The **page** of a current file. The current page should always be saved with an annotation if the file has more than one page. Later, it should only be displayed if the current page is selected. If the page is scrolled, the offset has to be determined. JavaScript provides a function by default to determine the scroll offset in pixels. This offset just has to be added to the annotations if the page gets scrolled. Hence, the annotations are always on the same position, even if the page changes or gets scrolled.

4.4.1.10 Downloading the annotated content

As Quality Attribute 2 and 8 are saying, the data should usually not leave the browser or be stored on the user's hard drive. Hence, rendering the annotations which are just overlaid in the files is not the main purpose of this thesis.

However, rendering the annotations into an image can be solved by the HTML canvas element very easy. The canvas element builds an area on web pages that allows JavaScript to draw inside of this element. Besides simple lines, it is also allowed to copy whole HTML sections, as the image and its overlaid annotations, inside of this canvas element. A canvas element can be downloaded as a regular image in all common formats, as PNG, GIF and JPEG [Baul 11], [Fult 13]. A screenshot of videos, PDF files and DICOM images can be produced with this technology, as well.

However, rendering annotations into different file types is not the purpose of this thesis. It will not support the video consultation process of patients. Hence, it will not be considered within this thesis.

4.4.2 Backend implementation

Setting up the backend is very straight forward. The endpoints for uploading and downloading files including its meta information is already given by the company's backend. I.e. that the FileService can call the endpoint to retrieve a file or post data to an existing file.

The only new part of the backend is method to store all data regarding the annotations. In section 4.3.4 different methods were compared. It turned out the synchronized database of Firebase will fulfill most of the Quality Attributes which were mentioned in Chapter 3. Setting up a shared database in Firebase is just about setting up an account and installing the Angular 2 library to the project. Afterwards, the library sends data a key and a value to the firebase by just calling one function. The database can be subscribed in the service, so that every operation on the node gets updated automatically in the frontend. There is no need of setting up a database structure or implementing anything in the backend. Everything is kept very simple.

Furthermore, the language is based on pure JavaScript and JSON. The language allows users to define very simple who has the right of accessing a certain node. For every file, a node will be created. Only people, who already have writing access guaranteed by the regular backend will get writing access. Basically just one Boolean expression gets checked in the Firebase.

4.4.3 Other challenges

4.4.3.1 Deal with conflicts

During real time editing of shared data, more than one person could edit an object. This could quickly lead to errors and is one of the main challenges during real time editing of data [Beiz 98] [Bly 93]. Usually, the problems are getting solved by locking critical sections. First of all, a user needs to have the right to edit a section, in general. A section turns critical if somebody else is editing the section which gets determined using a handshake between clients who could edit the section. [Hu 11] [Brun 13].

In this annotation tool, every participant has the possibility to add annotations. Annotations are lines, highlights, text and arrows which can overlay. There might be situations, where many forms are overlaid. For instance, one user could highlight a region to show where his suffer is. A doctor wants to specify this region or wants to point even more exactly to a point in this region. Hence, the responsibility of messing the picture up by overlaying annotations is given to the users on purpose.

Deleting: Annotations can usually only be deleted by the user who created them. If annotations are made by a nurse, the doctor should also have the possibility to delete the annotation according to requirement 7. However, it does not matter who of the users delete an annotation. If a nurse and a doctor delete the annotation at the same time, the 'isDeleted' flag in the annotation object gets updated two times to true. This is not a problem and the annotation will disappear on the screens of all participators in the same way.

Editing annotations should always be done by one person at the time. Strategies to prevent conflicts while collaboratively operating on data are Locks and Semaphores. The main difference is that a lock blocks the access to a resource. A semaphore allows a predefined number of operators before blocking the access to a resource [Leun 90] [Teva 87]. In this case, only one person at time should be allowed to operate on an annotation. This problem occurred many times during the engineering of multi process platforms and is solved by the concept of "Mutual exclusion" [Maek 85] [Rica 81]. After one party is asking for access of a resource, it gets checked if the resource is locked. If It is not locked, the resource is accessible and gets locked by the current client until he is

finished with his operation on the resource.

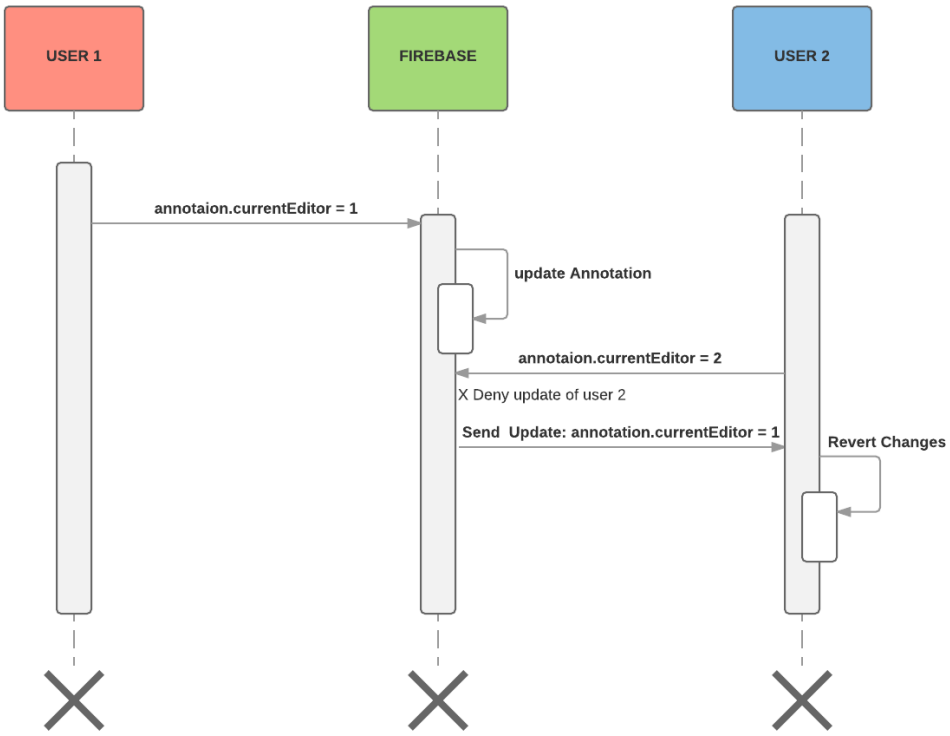


Figure 4.15: Two users are changing the annotation at the same time

Figure 4.15 shows, how conflicts are prevented if two users edit an annotation at the same time. For changing annotations, every annotation element receives an additional field called ‘currentEditor‘. This field is supposed to have the value `null` if there is no current editor. If a person starts to edit the field, he updates the field ‘currentEditor’ in the object of the annotation he starts to annotate. The update gets send to the service immediately. Other participants have no possibility to edit the object while the field ‘currentEditor‘ is something else than `null`.

If two persons start editing an annotation at the same time, the first user who sends an update to the database will be elected. The update message of the other user gets blocked. This happens within milliseconds. Therefore, the user should not even recognize this. If Firebase receives both updates, the update of the second user will be ignored. This can

be determined by checking if the user who sends the update has the same ID as the field ‘currentEditor‘.

4.4.3.2 Deal with timeouts

Thanks to the architecture of the tool, timeouts can not cause any problem. The annotations are synchronized using Firebase. If there is a timeout, the Firebase plugin which is used within the Angular 2 app will be disconnected from the database. Updates are usually pushed to the Angular 2 app using a WebSocket.

After reconnecting, Firebase automatically pulls the whole annotation data. If the user who was disconnected added additional annotations while he was not connected, it will automatically be synchronized by Firebase. If there are conflicts, they will be solved as regular as explained in section 4.4.3.1.

4.4.3.3 Support files embedded via plugin

In scenario 3 of chapter 2 it was suggested to include external DICOM files in the tool, using a plugin, e.g. from a software which is installed on the doctor's computer.

From this, requirement 14. (Support files embedded via plugin) was derived. In the rating of the requirements it turned out that people were quite undecided if this functionality is important. 10 people out of 22 assigned only 2 or 3 points out of 5 as importance. However, some people might still need this functionality and rated it with an importance of 5. The purpose of this functionality was to support the usage of DICOM files, as well. Also, the upload of huge files should be prevented.

This function is only necessary for medical staff who works at the company. All the computers of the medical staff are connected to high-speed internet with at least 50 Mbit/s. This means, that even the upload of huge files would never cause a problem and can be done very fast. DICOM viewer also exist as web viewers [Mont 13] [Kase 13]. Some of them can be simply installed as a JavaScript plugin without the need of any framework [Ivma 16]. Hence, viewing DICOM files is possible without problems. Even if there is no shared screen which gets embedded via plugin, the quality of the doctor's work won't suffer.

Chapter 5

Evaluation

In this chapter the implemented annotation tool will be evaluated. The annotation tool will be checked for a field application, functional requirements and Quality Attributes fulfillment.

The application of the software will be tested with predefined tasks in a real application. Medical staff will be tested during a real video consultation. Their reactions, problems and possible improvements will be noted.

Furthermore, the aim was to fulfill all requirements and Quality Attributes as good as possible with keeping the feedback of the expert survey in mind. Especially, if certain requirements were rated with a high importance during the requirement evaluation, it is important to fulfill these requirements as good as possible. The reason for requirements which were rated with a low score will be investigated as well.

5.1 Field test

5.1.1 Setting and test scenarios

In this test, medical staff who is already using the consultation software will use the new implemented annotation tool. The scenarios will be kept like the scenarios which were mentioned in chapter 3. It will be tested if the medical staff can apply the different functionalities and solve the problems which were given in the use cases. Five different experts were tested in a real video consultation. Different files were uploaded and the medical staff was asked to annotate different regions of different files. Also, they should name some typical cases, like a hair break in a faked X-ray image.

5.1.2 Positive results

All the functions work quite similar. After understanding the first function almost everything worked perfectly fine. All the participants switched between the tools and were able to understand them by just looking at the icon and reading the tooltip.

The medical staff was pretty impressed by seeing who is currently annotating. It was very supportive to see who is already offline and who is still in the session. After a while, there were too many annotations in some cases. After explaining the purpose of the "replay" button the users made use of this function and found it supportive.

5.1.3 Negative results

After first opening the annotation tool almost nobody directly knew how to use it. Some explanation was necessary before users could use the tool. After a one-minute explanation of the main features almost everybody was able to use the annotation tool. Only the deleting function was hard to understand in the beginning. Most of the people tries to use the deletion tool as an eraser. The purpose of the tool was to hover the whole annotation element and delete it completely. The medical staffed expressed the conviction that an eraser might be more intuitive to use. It could also delete small parts of the line which would be more supportive.

Unfortunately, nobody used the function to turn the screen black for a while. As the feedback in the survey already indicated, the feature might only be important in a few cases.

5.1.4 Conclusion

Even, if there were some issues in understanding a few special functions, the main purpose of the tool was clear very fast. Everybody understood all annotating functions after a one-minute introduction. Only some functions which already were rated with a low importance during the survey were not directly understood.

5.2 Functional Requirements

All functional requirements were successfully implemented. The features are usable during every video consultation. Experts were asked about the importance of each functional requirement during a survey.

5.2.1 Importance of functional requirements

In a survey, different experts were asked about their rating of the importance of different requirements. It turned out that the main requirements as "real time synchronization", "highlighting sections" and "drawing lines on a file" are the most important functions.

During the field test this ranking was proofed. All participants understood these main functionalities almost immediately and considered them as very useful.

5.2.2 Understanding and application of functional requirements

The final test showed that the results were very similar to the results of the requirement evaluation in the survey. Some requirements were rated very different from the participants. It was assumed that this could be because the participants of the survey don't understand some features.

The theory that some features were not explained good enough was proofed by the field test. After an explanation the features were clear and understandable for everybody. The feature "turn off screen" was never used by the participants who were tested which corresponds with the feedback of the survey, as well.

5.3 Quality Attributes

In the following part, all Quality Attributes will be investigated. It will be tested if the annotation tool fulfilled all of them.

1. Modularity / Reusability in all connected systems.
   All parts of the tool were written in Angular 2 components. The concerns are separated as much as possible. All components can be used as regular HTML directives with input and output parameters. All inputs and outputs are optional. Every HTML content can just be wrapped by this Angular 2 component and can be annotated. Input and output parameters offer the possibility the receive annotation updates, assign users as authors to an annotation or assign the file id or a timestamp of a video to the annotation.

```
<ng2-annotations
  (annotationElements)="synchronizeAnnotations($event)"
  [synchronizingUpdates]="synchronizingUpdates"
  [fileId]="selectedFile.id"
  [annotationSet]="_annotationsService.currentAnnotationSet"
  [myAccount]="myAccount"
```

```
  [currentVideoTime]="source.currentVideoTime | async">

  <dynamic-source #source
                [src]="selectedFile.fileURL"
                [filetype]="selectedFile.mime_type"></dynamic-source>
</ng2-annotations>
```

2. Run within the browser environment
   The whole app runs completely within the browser. The app will only be loaded to the browsers cache and deleted after a certain time.

3. Deal with bad connection quality
   Angular Firebase takes case of this problem. Even if there is no internet for a while, the data gets updated immediately after the Angular app reconnects to the internet.

4. Optimized data structure
   The tool was engineered very top-down. Classes were created in order to make the whole architecture clear, to reduce complexity and to reduce redundancy. TypeScript allows the user to implement the classes exactly as mentioned in the UML class diagram.

   The data of each annotation which will be created gets converted to a JSON object when storing it in a database. A JSON object has no overhead and can be reused and parsed within the application out of the box. Hence, an optimized data structure is guaranteed.

5. Different screen sizes and devices
   The annotation component is just a directive which can be used as a regular HTML directive. The only difference is that there is a toolbar which will be displayed above the annotated element. Hence, it can be used on every screen like a regular HTML component within every Angular 2 application.

6. Robustness
   During all tests the app didn't crash. As the structure and the component based architecture is very clear, a bug can be found very fast.

7. Scalability
   Hence the app in running in the user's browser, all operations are calculated on the client side. So, it doesn't matter on how many different devices the app runs.

   The backend is provided by firebase with no self-written server between. Firebase ensures to deal with every request rate and can also deal with DDos attacks. Hence, the app is absolutely scalable and can run on a huge number of devices at the same time.

8. Don't Store data on hard drive
   The whole app and all files get cached by the browser. This is necessary and cannot be bypassed. However, all actions are running within the browsers environment and the cache gets cleared after a very short time. No direct download of any files is necessary.

9. Encrypted data transfer
   All data exchange is based on HTTPS which delivers an encrypted connection. All annotations are stored in an encrypted NoSQL Database which is provided by Google Firebase. Firebase ensures highest security and latest encryption algorithms.

10. End-to-end connection
    As data gets stored in a shared database, the data is not stored within the browser of the clients only. Hence, there is no end-to-end connection. However, an end-to-end connection would not ensure absolute security and would still be prone to Man-In-The-Middle attacks, as before.

11. Frontend in TypeScript / Angular2
    The frontend is completely written as an Angular 2 library. Hence, this Quality Attribute is fulfilled.

12. Backend in Python
    Only the backend for retrieving a file is written in Python and was already provided by the company. This requirement was just given to reduce complexity and to fit better into the existing system. It turned out that there was no need of adding the annotation backend to the existing one. Using the shared database of firebase made many things less complex, more Quality Attributes were fulfilled and security is still guaranteed.

    Even if the shared database would be hacked, there would no relation to the registered users. Only an ID of the users is stored to each annotation. This can only be assigned to registered users if somebody would have access to the company's database.

13. Low effort of implementation
    Thanks to Firebase the implementation of the backend was straight forward. The implementation of the Frontend-Application was based on requirements and a clear structured class diagram. Only the concept was complex. Thanks to TypeScript the implementation of the single classes and services was very straight forward, as well.

14. High Efficiency
    The app works very fast. All annotations occur on the screen of other participants within less than one second using a regular DSL internet connection. Using an Intel i3 processor the usage never became bigger than 10% which means that the app

should run on al devices which came out within the last years.

15. Separation of concerns
    During the implementation chapter everything was focused on separating concerns. First, the whole project was divided in backend, services and components. Afterwards, all components were devided in classes which all have a clear concern and structure.

Chapter 6

Conclusion and future work

In this chapter the results of this Master's Thesis will be summarized. All research questions will be shortly summarized again, including its validation. Afterwards, this a conclusion of whole thesis and possible future work will be explained.

6.1 Conclusion

In section, results of the asked research questions which were asked within this thesis will be summarized.

6.1.1 RQ1: What is the state of the art for real-time collaborative web based annotation tools?

It turned out that adding annotations to documents is already realized within many different scopes of application. Also, the synchronization of data between different clients over the internet and operating on critical section using locks was already investigated a lot. Several solutions exist.

Even annotating files which are more complex, as e.g. videos and dealing with huge file sizes was solved by different frameworks, as e.g. Vondrick et. al. [Vond 13].

However, there is no solution which takes the requirements of medical staff into account with the purpose of teleconsultation via video. Hence, the scope of annotations and teleconsultation had to be investigated.

6.1.2 RQ2: What are the requirements for web based component design based on the given scenarios in order to create real-time annotations over the internet?

Different scenarios were given by the company which describe an expected behavior and use cases of the annotation tool. From these scenarios 19 functional requirements and 15 Quality Attributes were derived. The functional requirements were evaluated by 22 experts who have a medical education and at least three months of experience in the field of telemedical consultation.

It turned out that the real-time synchronization, highlighting of sections and the support of common file types seems to be very important for the experts. The experts were undecided about the importance of displaying the participant's status, replay the annotations and embed external software. Hence, these requirements are better explained in the actual implementation using self-explaining symbols and tooltips. As the importance of these requirements was not rated so high, they were placed more decently than very important functions in the tool.

6.1.3 RQ3: How can the tool be designed to be understandable, and fulfill all functional and non-functional requirements?

Keeping the results of the expert survey in mind, the rating of the requirements and the textual feedback was used to design two different mock-ups.

The first mock-ups only shows the functionalities which are most important and will be most commonly used, probably. As the expert rating of the requirements showed, some requirements might lead to confusion. All functionalities which could lead to confusion, were hided.

As there are not so many functions, the second mock-up shows all the different functionalities. Functions which could lead to confusion were not hided, but placed at positions where they look more decent. Hence, the confusion should still be minified.

In the second survey, people were asked about their understanding of both mock-ups. It turned out, that the mock-up which shows all the functions was preferred by the experts. Furthermore, almost everyone would understand the tool which was designed as a mock-up.

Furthermore, different technologies of synchronizing annotations and a way to find an optimized data structure were investigated in order to fulfill the Quality Attributes. Almost every requirement was fulfilled using the shared database concept of firebase and a class-based data architecture which was determined using an UML class diagram.

Afterwards, all functional requirements were solved logically in order to fulfill all functional requirements. Control flow diagrams were derived in order to find a way which is reduced in its complexity and easy to understand in order to solve all functional requirements.

6.1.4 RQ4: How far does the implemented prototype satisfy the requirements derived in RQ2 und how is an additional benefit guaranteed against other typical solutions?

After implementation, experts were invited to use the tool in a real environment. The scenarios from which the requirements were derived were performed. In the beginning, the experts had some problems of understanding how the tool works. After a short explanation of one minute, every expert was able to solve all tasks they were asked to. Every function was used and the experts mentioned, that this tool can be very supportive for the consultation process.

Additional benefit is guaranteed because the tool is included in the regular consultation tool, the doctors use. This reduces complexity and multi-tasking. As everything is embedded within one software, the programmers do exactly know where data is stored and getting processed. The data is not used by any third party software. This is important because the responsibility of securing medical data always has to be by the company and not by any third parties.

Typical solutions would run outside of the system and would store data on another server. Hence, security cannot be ensured by the company itself, anymore. Also, an external annotating system would not be connectable with existing files which were uploaded, the existing user accounts and the permission system which controls the user's right of accessing certain files.

6.2 Future work

Based on the experience which was derived from the survey and an in-field test several research questions came up.

The infield test shows that there are still some users who don't understand all functions of the annotation tool. After learning, almost every user was able to understand all functions. But the learning time was still quite long. Hence, an important task would be to figure out how to reduce the learning time of new users who use the tool for the first time.

Furthermore, the tool could be also used I other areas then telemedical consultation. Collaborative annotation of files is needed in many different discussions and might also

support other fields. It is important to keep in mind that the requirements were given and validated by medical experts. If this project gets applied to other fields, the requirements and their importance might change.

Another wish of some participators was to enable a version control system of annotations. It should be able to annotate a file many times and scroll to different versions. Also, it should be possible to restore an older version again. As in common version control systems, comparing different versions would also support the user in getting more insights about his annotations.

Rendering different into different data types was not a direct objective of this thesis, as the annotated files should never leave the browser. For images, the rendering of annotations was already explained. Rendering overlaid annotations in file types which are more complex, as DICOM images, Videos or PDF files is an issue which still should be solved in the future work.

Chapter 7

Appendix

7.1 Surveys

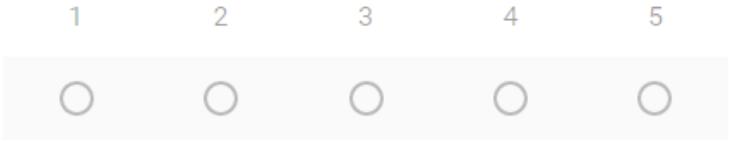
To validate the requirements and to determine which are the major requirements, several doctors were asked using a survey. Google Forms was used as a framework to create and conduct the survey.

In a second survey the doctors were asked about the functional design of an annotation tool.

The survey questions and he detailed results which have been used and interpreted in this thesis are displayed here. In order to give all the doctors, the opportunity of understanding the questions, the original questions are in German. An English translation is always written below.

7.1.1 Part 1 – Requirements

Jeder, der die Datei geöffnet hat, sollte alle Annotationen in Echtzeit sehen.
Anyone who has a file opened, should see all annotations in real time.



Eine Linie auf dem Bild malen ist ...
Drawing a line om an image is...

1 2 3 4 5

Überflüssig  Extrem wichtig

Einen Bereich hervorheben (wie mit einem Textmarker) ist ...
Highlight a part of a file (like with a highlighter) is...

1 2 3 4 5

Überflüssig  Extrem wichtig

Dem geteilten Bildschirm (z.B. dem Röntgenbild) eine Textbox hinzufügen ist ...
Adding a textbox to the shared screen is...

1 2 3 4 5

Überflüssig  Extrem wichtig

Dem Bild Elemente (wie einen Kreis oder ein Viereck) hinzufügen ist ...
Adding elements like a circle or a rectangle to the shared section is...

1 2 3 4 5

Überflüssig  Extrem wichtig

Annotationen löschen ist ...
Deleting annotations is ...

1 2 3 4 5

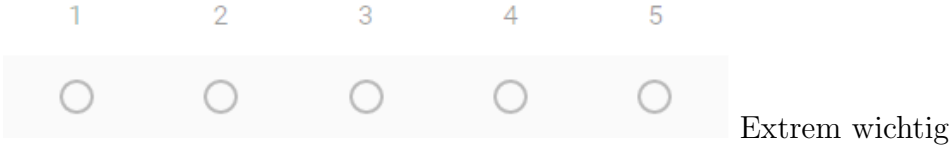
Überflüssig  Extrem wichtig

Das Ausschalten des geteilten Bildschirms ist...
Turning off the shared section for a moment is ...

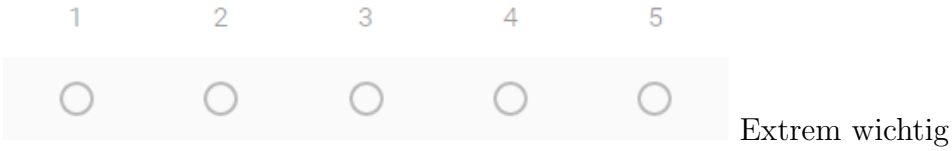
1 2 3 4 5

Überflüssig  Extrem wichtig

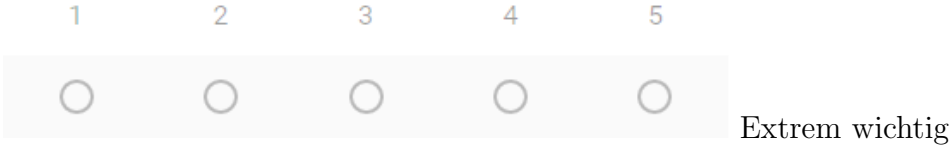
Zu jeder Zeit entscheiden, wer Zugriff auf Dateien hat ist ...
Deciding who has access to a file is...



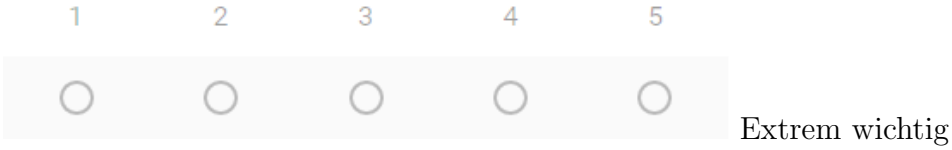
Unterstützung von verschiedenen Formaten (z.B. Röntgenbilder, Fotos, Videos) ist ...
Supporting a huge set of filetypes (like X-ray images, usual images, videos), is...



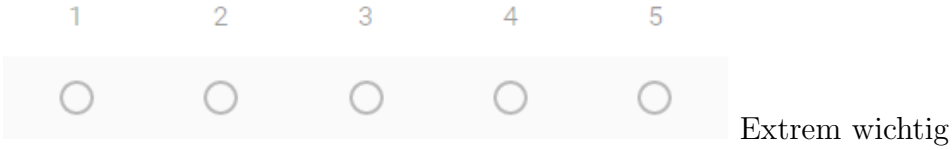
Andere Teilnehmer sehen, ist ...
See other participators oft he annotation session is ...



Die Annotation von JPG, JPEG und PNG Dateien ist...
Annotating images a JPG, JPEG and PNG is...



Die Annotation von PDF Dateien ist...
Annotating PDF files is ...



Die Annotation von DICOM Dateien ist...
Annotating DICOM files is ...

1 2 3 4 5

Überflüssig  Extrem wichtig

Das Einbetten eines externen Programms, wie z.B. OsiriX, und das Teilen per Video Stream (Beispielsweise das Übertragen von Röntgenbildern) ist...
Embedding external software, es e.g. OsiriX, and share the screen via video to other participators (E.g. for transmitting X-ray images) is ...

1 2 3 4 5

Überflüssig  Extrem wichtig

Den Autor (Ersteller) einer Annotation zu sehen, ist ...
Seeing the author of each annotation is ...

1 2 3 4 5

Überflüssig  Extrem wichtig

Die annotierte Datei downloaden ist ...
Downloading the annotated file is ...

1 2 3 4 5

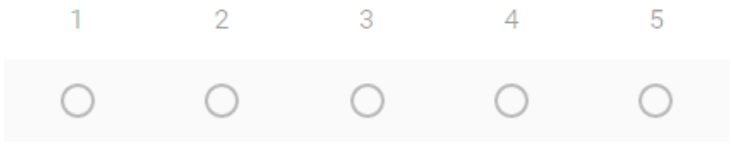
Überflüssig  Extrem wichtig

Den Annotierungsprozess erneut (wie ein Video) abspielen ist ...
Replay the annotating process like a video is ...

1 2 3 4 5

Überflüssig  Extrem wichtig

Den Status anderer Teilnehmer sehen (online, offline, aktiv) ist ...
Seeing the current state of other participators, e.g. online, offline or active, is ...



Überflüssig Extrem wichtig

7.1.2 Part 2 – Design

In the second part of the survey, the design expectations of future users are evaluated. To do so, two different mockups were created. One which included all functional requirements in and another one, which shows a minimalistic view to prevent confusions. There is no functionality missing in the minimalistic mockup. Other functions will just pop up, if the user opens a menu. Only the main functions are always displayed.

Users had to evaluate which mockup they prefer and why. The original questions were asked in German. An English translation is printed below:

Welches Version bevorzugen Sie rein subjektiv?

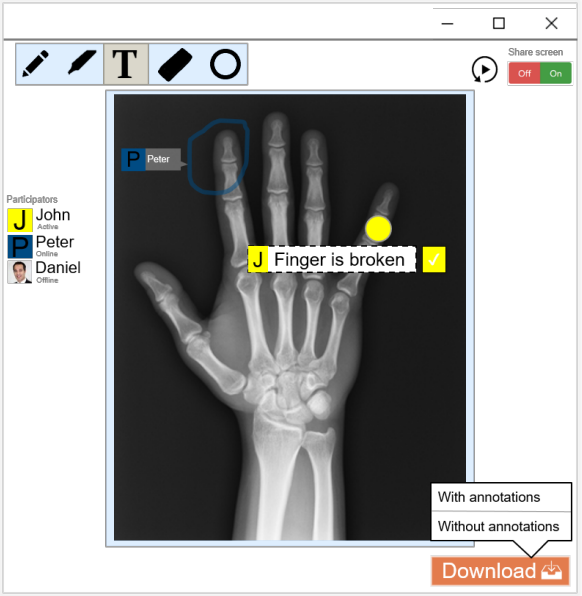


Bild 1

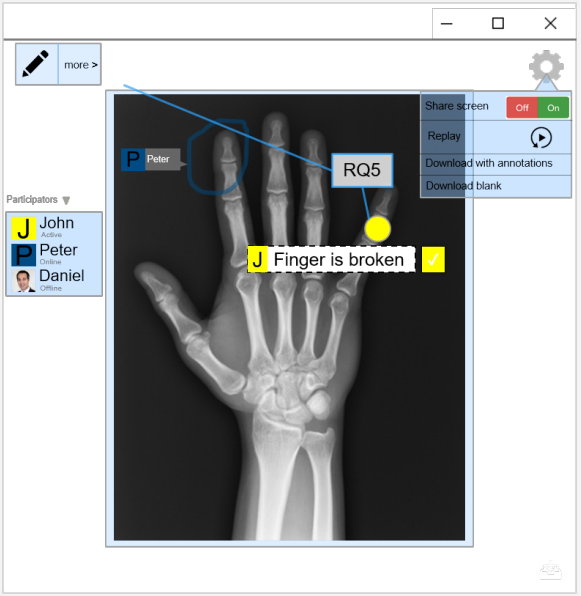
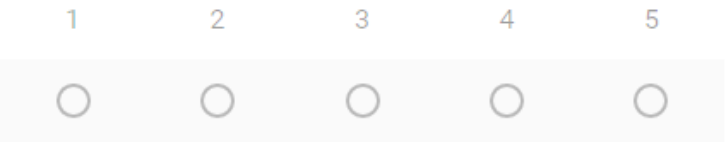


Bild 2

Das Tool sollte...
The tool should ...

Nur die wichtigsten Funktionen direkt anzeigen



Möglichst alle Funktionen gleichzeitig anzeigen

Verschiedene Optionen (z.B. Werkzeuge) sollten jeder Zeit sichtbar sein. Different options, as e.g. tools, should be visible all the time.

○ Ja, alle Optionen sollten immer sichtbar sein.
Yes, all options should always be visible.
○ Nur die Werkzeuge (Stift, Textmarker, ...) sollten immer sichtbar sein.
Only tools, e.g. the pen or highlighter, should always be visible.
○ Optionen sollten nur angezeigt werden, wenn ich Sie wirklich benötige (z. B. wenn man mit der Maus über das Feld fährt).
Options should only be visible if I need them. E.g. when I hover it with the mouse.
○ Optionen sollten nur zu sehen sein, wenn ich ein extra Menü öffne (z.B. ein Dropdown Menü).
Options should only be visible if I open an extra menu, e.g. a dropdown menu.

7.1.3 Text responses

Furthermore, the participants of the survey were asked about additional feedback. They were asked about their understanding of the tool from the mockups and about their ideas, wished, suggestions and feedback.
All the responses are not edited in their orthography or grammar. The original responses are printed under the textboxes. Under each response an English translation is written in gray letters.

Würden Sie das Tool mit allen Funktionalitäten verstehen? Was verstehen Sie nicht? Was würde Sie anderes machen?
Would you understand the tool including all its functionalities? What do you not understand? What would you do different?

- Ja, ich würde es verstehen.
  Yes, i would understand it.
- Ich würde den Button neben S̈hare Screenïicht verstehen (Replay)
  I wouldn't understand the button next to share screen (Replay).
- gut zu verstehen
  Good to understand.
- würde Tool verstehen
  I would understand the tool.
- verstanden
  Understood.
- Denke, dass man sich da ähnlich wie bei OsiriX relativ schnell rein finden würde
  I think that it would be as easy as OsiriX to manage yourself quickly.
- Ich verstehe das Tool! Scheint eine sinnvolle Ergänzung zu sein!
  I understand the tool! It seems to be a sense making supplement.
- ja / - / -
  Yes / - / -
- Ich finde die Option mit den wichtigeren Funktionalitäten besser verständlich bzw. leichter zu überblicken
  I think that the option which shows only the important functionality is more understandable and more straightforward, respectively.
- Das Tool ist sehr übersichtlich aufgebaut und die Icons der Optionen sind selbsterklärend
  The tool is build very clear and the icons of the options are self-explanatory.
- Was ich noch nicht ganz verstehe ist, wie das ganze in den Beratungsprogess miteinfließt. Ist das eine zusätzliche option zu einem normalen Videotelfonat? Quasi wie auf Skype, wo man dann seinen Bildschirm teilen kann... Bezüglich des Tools ist alles gut verständlich und es ist klar, was damit gemacht werden soll.
  I don't understand how this supports the consultation process. Is this an additional

option for a usual videocall? Is it the same as the screen sharing function in Skype? Regarding the tool everything is clear and understandable.

- Ja
  Yes.

Was für Wünsche, Anregungen oder Feedback haben Sie zu dem Annotations Tool bis jetzt?
What wishes, suggestions or feedback do you have regarding the annotation tool so far?

- Besser keine gefüllten Punkte, die Die Röntgenaufnahme verdecken. Besser wäre, nur Hinweispfeile und ggf. Textmarkerkreise.
  Better don't fill the circles. They could hide important sections of the X-ray image. It would be better to add arrows only or highlighter circles.
- finde ich sehr übersichtlich
  For me it is clear.
- Super Idee! So kann man einem Patienten leichter verständlich machen was man sieht.
  Great idea! You can explain much easier what you see to the patient.
- Es sollte simpel gehalten werden
  It should be kept simple.
- Statt Kreisen würde ich kleine Pfeile verwenden, die auf einen Punkt zeigen. So wird es in gängigen Programmen gehandhabt!
  Instead of circles i would use small arrows who point to a section. This is how it is solved in common software.
- Ich möchte gerne ein Bild mehrfach v̈on Anfang anb̈earbeiten. Später möchte ich zwischen verschiedenen Versionen hin und her schalten können.
  I want to edit an image from the beginning. Later I want to switch between different versions.
- Nie Annotationen auf das zu untersuchende Organ! Auf dem Bildbeispiel sind wesentliche Teile überdeckt! Wenn Annotationen, dann müssen sie ausblendbar sein!

Never add annotations to the examined organ. In the example image significant parts are overlaid! If you add annotations, there must be a way to hide them.

- gute Idee!
  Good idea!
- Eventuell eine temporäre Annotation wäre hilfreich. Damit der Patient ggf. dem Doktor etwas zeigen kann. ZB. was dieser Punkt auf den Röntgenbild ist oder so.
  A temporary annotation might be supportive, perhaps. Thereby, people could show something to the doctor if necessary. E.g. explaining a point on the X-ray image or something else.
- Kann man eine History sehen?
  Can I see a History?

7.2 List of Abbreviations

API	Application Program Interface
CSS	Cascading Style Sheets
CSV	Comma-separated values
DICOM	Digital Imaging and Communications in Medicine
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
PDF	Portable Document Format
JPEG	ISO/IEC 10918-1 Standard for compressing images Developed by Joint Photographic Experts Group
JSON	JavaScript Object Notation
MVC	Model View Controller
NoSQL	Non-Structured Query Language.
PNG	Portable Network Graphics
QA	Quality Attribute
REST	Representational State Transfer
RTC	Real-Time Communication
SQL	Structured Query Language
WebRTC	Web Real-Time Communication

List of Figures

Bibliography

[Anth 12] G. Anthes. *HTML5 leads a web revolution.* Magazine Communications of the ACM, 2012. http://dl.acm.org/citation.cfm?id=2209256.

[Baul 11] D. Baulig. *High Performance ECMAScript und HTML5 Canvas.* Fachhochschule Frankfurt am Main, 2011. http://www.danielbaulig.de/wp-content/uploads/2011/04/Bachelor-Thesis.pdf.

[Beiz 98] M. M. Beizer, D. Berg, R. Scullard, P. R. Simha, and M. A. Solomon. *Method of resolving data conflicts in a shared data environment.* US Patent Office US 6240414 B1, 1998. https://www.google.com/patents/US6240414.

[Bier 14] B. Bierman, M. Abadi, and M. Torgersen. *Understanding TypeScript.* Volume 8586 of the series Lecture Notes in Computer Science pp 257-281, 2014. http://link.springer.com/chapter/10.1007/978-3-662-44202-9$_1$1.

[Bly 93] S. A. Bly, J. D. Hodges, B. T. L. Michael D. Kupfer and, M. L. Tallan, S. B. Tom, and Weniger. *Updating local copy of shared data in a collaborative system.* US Patent Office US 5220657 A, 1993. https://www.google.com/patents/US5220657.

[Brue 10] B. Bruegge, J. Helming, M. Koegel, F. Schneider, M. Haeger, C. Kaminski, and B. Berenbach. *Towards a unified requirements modeling language.* IEEE, 9 2010. https://scholar.google.com/citations?view$_op = view_citationhl = en$
$user = abfwxtkAAAAJcitation_for_view = abfwxtkAAAAJ : u5HHmVD_uO8C.$

[Brun 13] Y. Brun, R. Holmes, M. D. Ernst, and D. Notkin. *Early Detection of Collaboration Conflicts and Risks.* IEEE Transactions on Software Engineering (Volume: 39, Issue: 10, Oct. 2013), 2013. http://ieeexplore.ieee.org/document/6520859/.

[Bund 15] Bundesnetzargentur. "Anzahl der Breitbandanschlüsse im Festnetz in Deutschland von 2001 bis 2014 nach Anschlusstechnologie (in Millionen)". 2015. http://de.statista.com/statistik/daten/studie/3174/umfrage
/entwicklung-der-breitbandanschluesse-nach-anschlussart-seit-2001/.

[Bund 16] Bundesnetzargentur. "Anteil der Personen, die zumindest selten Videos im Internet abrufen in Deutschland in den Jahren 2006 bis 2015". 2016.

http://de.statista.com/statistik/daten/studie/163165/umfrage
/abruf-von-videodateien-im-internet-seit-dem-jahr-2006/.

[Catt 11] R. Cattell. *Scalable SQL and NoSQL data stores.* Newsletter, ACM SIGMOD Record, Volume 39 Issue 4, Pages 12-27, 2011. http://dl.acm.org/citation.cfm?id=1978919.

[Chen 05] T. Chen, R. Lai, and A. Chen. *System and method for setting user-right, and recording medium.* US Patent 20050144060 A1, 2005. https://www.google.com/patents/US20050144060.

[Coff 02] K. G. Coffman and A. M. Odlyzko. *Internet Growth: Is There a "Moore's Law" for Data Traffic?* Springer US, 2002. http://dx.doi.org/10.1007/978-1-4615-0005-6_3.

[Dahi 01] T. Dahinden, A. Neumann, and A. M. Winter. *Webmapping mit SVG.* Institut für Kartographie, ETH Zürich, 2001. http://carto.net/papers/svg/articles/paper$_k arlsruhe_d ahinden_n eumann_w inter_2$001.*pdf*.

[Desh 00] J. L. Deshayes and H. J. Philippe. "Internet use for telemedecine: fetal medicine applications". *Journal de radiologie*, Vol. 81, No. 4, pp. 441–444, 2000.

[Dude 16] Duden. *Echtzeit, die.* 2016. http://www.duden.de/rechtschreibung/Echtzeit.

[Duru 13] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. *Analysis of the HTTPS certificate ecosystem.* IMC '13 Proceedings of the 2013 conference on Internet measurement conference, 2013. http://dl.acm.org/citation.cfm?id=2504755.

[Fire 16a] Firebase. *Firebase Docs.* Google, 2016. https://www.firebase.com/docs.

[Fire 16b] Firebase. *Firebase Official Homepage.* Google, 2016. https://firebase.google.com.

[Fire 16c] Firebase. *Firebase Security Docs.* Google, 2016. https://www.firebase.com/docs/security/quickstart.html.

[Fox 15] Fox, Wu, Uyar, and Bulut. "A Web Services Framework for Collaboration and Audio/Videoconferencing". 2015.

[Frai 12] B. Frain. *Responsive Web Design with HTML5 and CSS3.* Packt Publishing Ltd., 2012. https://books.google.de/books?hl=delr=id=
fcfarreMQ9sCoi=fndpg=PT12dq=css3ots=ZsbfWlXVPqsig=gsIQHVU6f3MyQjctxl0D-r8eL5cv=onepageq

[Fult 13] S. Fulton and J. Fulton. *HTML5 Canvas.* Fachhochschule Frankfurt am Main, 2013. https://books.google.de/books?hl=delr=
id=zLUyKvtdCQwCoi=fndpg=PR2dq=canvas+w3cots=Haxp-SSO1ksig=NHOa8wgzETL9-PKNJ729sTo

[Goog 16] Google. *Google Docs: Kostenlose Dokumente online erstellen und bearbeiten.* 2016. https://www.google.de/intl/de/docs/about/.

[Gupt 15] N. Gupta. *Vertical social software for remote collaboration over video.* Master's thesis, Technische Universität München, 2015.

[Haak 04] J. M. Haake, A. Haake, T. Schümmer, M. Bourimi, and B. Landgraf. *End-user controlled group formation and access rights management in a shared workspace system.* Proceeding CSCW '04 Proceedings of the 2004 ACM conference on Computer supported cooperative work, Pages 554-563, 2004. http://dl.acm.org/citation.cfm?id=1031702.

[Herl 93] M. Herlihy and J. E. B. Moss. *Transactional memory: architectural support for lock-free data structures.* ISCA '93 Proceedings of the 20th annual international symposium on computer architecture, Pages 289-300, 1993. http://dl.acm.org/citation.cfm?id=165164.

[Hoga 11] B. P. Hogan. *HTML5 CSS3.* O'Reilly Verlag, 2011. https://books.google.de/books?hl=delr=id=
xqmgd5jmWXMCoi=fndpg=PR1dq=css3ots=s-UzJHXMDasig=bD4EkyGYO2FSbyNYxQbcoqUum3Qv=

[Holz 14] M. Holzer. *Konzeption und Realisierung eines Frameworks für verteiltes Rechnen in Webbrowsern mittels WebRTC.* Master's thesis, 2014.

[Hu 11] H. Hu, G.-J. Ahn, and J. Jorgensen. *Detecting and resolving privacy conflicts for collaborative data sharing in online social networks.* ACSAC '11 Proceedings of the 27th Annual Computer Security Applications Conference, 2011. http://dl.acm.org/citation.cfm?id=2076747.

[Ivma 16] Ivmartel. *DWV (DICOM Web Viewer).* 2016. https://ivmartel.github.io/dwv/.

[John 12] A. B. Johnston and D. C. Burnett. *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web.* Digital Codex LLC, 2012.

[Kase 13] M. Kaserer. *DICOM Web Viewer.* Bachelor's Thesis at TU Wien, 2013. https://www.cg.tuwien.ac.at/research/publications/2013/kaserer-2013-webdicom/kaserer-2013-webdicom-

[Kazm 12] L. B. P. C. R. Kazman. *Software Architecture in Practice, Third Edition.* Addison-Wesley Professional, 2012. http://proquest.tech.safaribooksonline.de.eaccess.ub.tum.de/book
/software-engineering-and-development/9780132942799/.

[Kell 07] M. A. Kelley and M. S. Wengrovitz. *Real-time collaboration center.* 2007. https://www.google.com/patents/US8737596.

[Koch 02] H. Koch. *Einführung in die Mathematik.* Part of the series Springer-Lehrbuch pp 289-303, 2002. http://link.springer.com/chapter/10.1007/978-3-662-06857-1_9.

[Kope 87] H. Kopetz and W. Ochsenreiter. *Clock Synchronization in Distributed Real-Time Systems.* IEEE Transactions on Computers, Volume: C-36 Issue: 8, 1987. http://ieeexplore.ieee.org/document/5009516/.

[Leff 01] A. Leff. *Web-application development using the Model/View/Controller design*

pattern. IBM Thomas J. Watson Res. Center, Hawthorne, NY, USA, 2001. http://ieeexplore.ieee.org/document/950428/authors?ctx=authors.

[Leun 90] W. L. Leung, R. A. Kelley, and L. F. McDermott. *Data processing system and memory controller for lock semaphore operations*. US patent US5293491 A, 1990. https://www.google.com/patents/US5293491.

[Like 32] R. Likert. *A technique for the measurement of attitudes.* Archives of Psychology, Vol 22 140, 1932, 55., 1932. http://psycnet.apa.org/psycinfo/1933-01885-001.

[Like 61] R. Likert. *New patterns of management.* New York, NY, US: McGraw-Hill New patterns of management.(1961). ix 279 pp., 1961. http://psycnet.apa.org/index.cfm?fa=search.displayRecorduid=1962-05581-000.

[Lore 11] S. Loreto, P. Saint-Andre, S. Salsano, and G. Wilkins. *Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*. Internet Engineering Task Force (IETF), 2011. http://www.rfc-editor.org/info/rfc6202.

[Lubb 11] P. Lubbers, B. Albers, and F. Salim. *Pro HTML5 Programming*. Apress, 2011. http://link.springer.com/book/10.1007/978-1-4302-3865-2.

[Maek 85] M. Maekawa. *An algorithm for mutual exclusion in decentralized systems*. ACM Transactions on Computer Systems Volume 3 Issue 2, May 1985 Pages 145-159, 1985. http://dl.acm.org/citation.cfm?id=214445.

[Meln 11] A. Melniko. *The WebSocket Protocol*. Google, Inc., 2011. https://tools.ietf.org/html/rfc6455.

[Mich 14] P. N. R. K. M. J. Michael Derntl, Stephan Erdtmann. *Echtzeitmetamodellierung im Web-Browser*. RWTH Aachen, Chair of Informatics 5, 2014. https://pdfs.semanticscholar.org/0829/cada43a4b57612366885763ebb555578a6f0.pdf.

[Mild 02] P. Mildenberger, M. Eichelberg, and E. Martin. *Introduction to the DICOM standard*. Srpinger, The European Society of Radiology, 4 2002. http://link.springer.com/article/10.1007/s003300101100.

[Mont 13] E. J. M. Monteiro, C. Costa, and J. L. Oliveira. *A DICOM viewer based on web technology*. e-Health Networking, Applications Services (Healthcom), 2013 IEEE 15th International Conference on, 2013. http://ieeexplore.ieee.org/document/6720660/.

[Offi 16a] A. O. Office. *Die freie Büro-Software*. 2016. https://www.openoffice.org/de/.

[Offi 16b] L. Office. *Home — LibreOffice - Free Office Suite - Fun Project - Fantastic People*. 2016. https://libreoffice.org/.

[Offi 16c] M. Office. *Bürosoftware-Anwendungen für Produktivität*. 2016. https://products.office.com/de-de/home.

[Piko 03] A. Pikovsky, M. Rosenblum, and J. Kurths. *Synchronization: a universal concept in nonlinear sciences*. Cambridge Nonlinear Scrience Series 12, 2003. https://books.google.de/books?hl=delr=id=FuIv845q3QUCoi=fndpg=PP1 dq=synchronizationots=RL0yDnAiSdsig=kibcZLOf4V$_U$k9$xwUdtMYpWDYKcv$ = $onepageq = synchronizationf = false$.

[Pime 12] V. Pimentel and B. G. Nickerson. *Communicating and Displaying Real-Time Data with WebSocket*. IEEE Internet Computing, Volume: 16 Issue: 4, 2012. http://ieeexplore.ieee.org/document/6197172/authors.

[Poma 15] G. Pomaska. *Grundkurs Web-Programmierung: Interaktion, Grafik und Dynamik*. 2015. https://books.google.de/books?hl=delr=id=HNSGBwAAQBAJoi= fndpg=PR5dq=SVG+w3cots=n0qgzvs2Oysig=Lrq7nXJL3ZgblMtUUBjZ$_M$h2y48v = $onepageq = SVG$

[Prez 12] P. R. Pérez, J. C. Arriba, I. Trajkovska, and J. S. Rodríguez. "Advanced Videoconferencing based on WebRTC". 2012. https://www.mysciencework.com/publication/show/ede01555efb27df60f99f5c255ac746c.

[Qi] G.-J. Qi, X.-S. Hua, Y. Rui, T. Mei, and H.-J. Zhang. "Correlative multi-label video annotation". http://dl.acm.org/citation.cfm?id=1291245.

[Rajk 12] R. Rajkuma. *Synchronization in real-time systems: a priority inheritance approach*. 2012. https://books.google.de/books?hl=delr=id=j$_Y$$ICAAAQBAJoi = fndpg = PP10dq = real + time + synchronizationots = sqMyEPcGA1sig = LMTMWXPXpLeYWoh1hMIcPsB3wG8v = onepageq = real$

[Rapp 01] H. Rapp. *Mathematik für die Fachschule Technik*. Mathematik für die Fachschule Technik, 2001. http://link.springer.com/chapter/10.1007/978-3-322-91971-7$_3$.

[Rica 81] G. Ricart and A. K. Agrawala. *An optimal algorithm for mutual exclusion in computer networks*. Communications of the ACM Volume 24 Issue 1, Jan. 1981 Pages 9-17, 1981. http://dl.acm.org/citation.cfm?id=358537.

[Ried 13] L. Riedl, D. Rost, and E. Schörner. *Brückenkurs für Studierende des Lehramts an Grund-, Haupt- oder Realschulen der Ludwig-Maximilians-Universität München*. Part of the series Konzepte und Studien zur Hochschuldidaktik und Lehrerbildung Mathematik pp 55-65, 2013. http://link.springer.com/chapter/10.1007/978-3-658-03065-0$_5$.

[Russ 08] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. *LabelMe: A Database and Web-Based Tool for Image Annotation.* International Journal of Computer Vision, 2008. http://link.springer.com/article/10.1007/s11263-007-0090-8.

[S 13] M. de Sá, D. A. Shamma, and E. F. Churchill. "Live mobile collaboration for video production: design, guidelines, and requirements". 2013.

[Sand 16] R. Sanderson, P. Ciccarese, and B. Young. *Web Annotation Data Model*. W3C Candidate Recommendation 06 September 2016, 2016. https://www.w3.org/TR/annotation-model/.

[Scho 06] P. Scholz. *Softwareentwicklung Eingebetteter Systeme*. Physica-Verlag, 02.05.2006 - 232 Seiten, 2006. https://books.google.de/books?id=xyCGFox6XCUCpg=PA39dq=echtzeit+begriffhl=desa=Xei=PWvMUbGGHKiN7AbV2oHoCAved=0CGMQ6AEwCQv=onepageq=echt

[Sha 02] L. Sha, R. Rajkumar, and J. Lehoczky. *Priority inheritance protocols: an approach to real-time synchronization*. 2002. http://ieeexplore.ieee.org/document/57058/.

[Snap 16] Snapchat. *Snapchat*. 2016. https://www.snapchat.com/.

[Sull 06] S. Sull, H. Kim, Y.-S. Seong, M. Rostoker, and J. Kim. "Techniques for navigating multiple video streams". 2006.

[Supp 16] M. O. Suppport. "Collaborate on Word documents with real-time co-authoring". 2016.

[Tele 16] TeleClinic. *Die TeleClinic App - Alle Funktionen mit einem Klick*. TeleClinic GmbH, 2016. https://www.teleclinic.com/funktionen/.

[Teva 87] A. Tevanian. *MACH threads and the UNIX kernel: the battle for control*. Carnegie Mellon University, 1987. http://repository.cmu.edu/cgi/viewcontent.cgi?article=2728context=compsci.

[Tool 16] L. Tools. *Medical Web Viewer Framework SDK Technology*. LEAD Technologies, Inc., 2016. https://www.leadtools.com/sdk/medical/web-viewer-framework.

[Trip 11] O. Tripp and O. Weisman. *Hybrid analysis for JavaScript security assessment*. IBM Rational Software, 2011. http://www.lcis.com.tw/paper$_s$*tore/paper*$_s$*tore/paper* − 20158492511187.*pdf*.

[Twil 16] Twilio. *WEBRTC - Embed IP communications into a web interface*. Twilio, 2016. https://www.twilio.com/webrtc.

[Ullr 12] B. Ullrich. *WebSockets: Spezifikation / Implementierung*. TUM, 2012. https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2012-04-1.pdfpage=61.

[Vond 11] C. Vondrick and D. Ramanan. *Video annotation and tracking with active learning*. Advances in Neural Information Processing Systems 24 (NIPS 2011), 2011. http://web.mit.edu/vondrick/vatic/videoalearn.pdf.

[Vond 13] C. Vondrick, D. Patterson, and D. Ramanan. *Efficiently Scaling up Crowdsourced Video Annotation*. International Journal of Computer Vision January 2013, Volume 101, Issue 1, pp 184–204, 2013. http://link.springer.com/article/10.1007/s11263-012-0564-1.

[Wang 13] V. Wang, F. Salim, and P. Moskovits. *The Definitive Guide to HTML5 WebSocket*. Springer, 2013. http://link.springer.com/book/10.1007/978-1-4302-4741-8.

[Webs 16] M. Webster. *Simple Definition of annotation.* 2016. http://www.merriam-webster.com/dictionary/annotation.

[Weig 13] H.-G. Weigand, A. Filler, R. Hölzl, S. Kuntze, M. Ludwig, J. Roth, B. Schmidt-Thieme, and G. Wittmann. *Didaktik der Geometrie für die Sekundarstufe I.* Part of the series Mathematik Primarstufe und Sekundarstufe I + II pp 238-262, 2013. http://link.springer.com/chapter/10.1007/978-3-642-37968-$0_1$1.

[YouT 16] YouTube. *Anmerkungen erstellen und bearbeiten.* YouTube Support, 2016. https://support.google.com/youtube/answer/92710?hl=de.

[Yu 07] D. Yu, A. Chander, N. Islam, and I. Serikov. *JavaScript instrumentation for browser security.* POPL '07 Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 2007. http://dl.acm.org/citation.cfm?id=1190252.

[Zhai] G. Zhai, G. Fox, M. Pierce, W. Wu, and H. Bulut. "eSports: collaborative and synchronous video annotation system in grid computing environment". http://ieeexplore.ieee.org/document/1565818/.