Gottfried Wilhelm
Leibniz Universität Hannover
Faculty of Electrical Engineering and Computer Science
Institute of Practical Computer Science
Software Engineering

Web Application with Django-Framework for Video Analysis

Bachelor Thesis

in Computer Science

by

Penske Williano

First Examiner: Prof. Dr. Kurt Schneider
Second Examiner: Dr. Jil Klünder
Supervisor: M. Sc. Jianwei Shi

Hannover, 9 March 2021

Zusammenfassung

Videos sind wichtig für die Ermittlung von Softwareanforderungen. Die meisten Softwareentwickler sind jedoch keine Videoprofis. Daher zielt diese Arbeit darauf ab, eine Webanwendung zu entwickeln, die sie dabei unterstützt, die Qualität ihrer Videos zu verbessern. Diese sollte insbesondere in der Lage sein, die technischen Merkmale von echten Vision Videos zu analysieren und umsetzbares Feedback zu geben, um diese Merkmale zu verbessern. In diesem Zusammenhang wird ein Vision Video als eine Art Video definiert, das eine Vision oder Teile davon für alle Beteiligten darstellt.

Um das Ziel zu erreichen, wurde mit Hilfe des Django-Frameworks eine Webanwendung entwickelt. Die Anwendung analysiert die Eigenschaften des gesamten Videos sowie die einzelnen Szenen des Videos. Dabei erhält der Benutzer empfohlene Werte, Bewertungen und Rückmeldungen in Bezug auf jedes Merkmal. Dieses wird auch von einer Visualisierung und Erklärung begleitet, um die Benutzerfreundlichkeit zu verbessern.

Um die Nutzbarkeit der Webanwendung zu untersuchen, wurden Online-Interviews mit 17 Teilnehmern durchgeführt, darunter Informatikstudenten, Forscher und Softwareprofis. Im Rahmen des Bewertungsprozesses wurden die Teilnehmer gebeten, verschiedene Aufgaben zu erledigen, Fragen zu beantworten und subjektive Bewertungen zu relevanten Aspekten abzugeben. Die Ergebnisse deuten auf ein hohes Maß an Intuitivität der Webanwendung und verständliche Rückmeldung hin.

Abstract

Web Application with Django-Framework for Video Analysis

Video is an important tool for requirements elicitation. However, most software engineers are not video professionals. This thesis aims to develop a web application that could assist people who are not video professionals to improve their videos' quality. Specifically, the web application should be able to analyze the technical characteristics of real-life vision videos and give actionable feedback to improve these characteristics. In this context, a vision video is defined as a type of video which represents a vision or parts of it for all parties involved.

To achieve this goal, a web application was developed with the help of the Django framework. The application analyzes the characteristics of the whole video, as well as each shot that composes the video. Recommended values, ratings, and feedback are given to each of them. Each characteristic is also accompanied by visualization and explanation in order to improve usability.

In order to investigate the usability of the web application, a survey of 17 participants, including computer science students, researchers, and software professionals, was conducted. As part of the evaluation process, the participants were requested to do several tasks, answer diverse questions, and give subjective ratings on relevant aspects. The results suggest a high degree of intuitiveness of the web application and good comprehensibility of the feedback.

Contents

Chapter 1

Introduction

In order to develop software which satisfies the needs of a client, an accurate understanding of their requirements is crucial. According to Nuseibeh and Easterbrook [25], requirements engineering is a methodical procedure to discover the purpose of a software system, identify stakeholders and their demands, and document these in a form tractable for communication, analysis, and subsequent implementation.

Video is an important form of documentation in requirements engineering, specifically during the requirement elicitation process. According to Karras et al. [20], a vision video is a type of video depicting a vision or its parts to achieve shared comprehension among all parties involved. It serves as an alternative or extension to written documentation, which allows an easier way for clients to understand how the developers envision the product.

1.1 Motivation

A research study done by Karras and Schneider [18] shows that, despite 35 years of research in utilizing videos for requirements engineering, videos are not yet an established documentation practice. According to a survey of 64 software professionals worldwide by Karras [17], the most mentioned weakness of vision videos is the effort required for planning, producing, watching and processing the videos. Furthermore, Karras et al. [20] concluded that missing guidance is a crucial reason of why videos are not yet established in requirements engineering.

One possibility of rectifying these issues is by hiring video professionals to produce the videos. However, there are some controversies that are associated with it. According to Brill et al. [5], outsourcing video production to professionals will likely result in improved video quality, due to the fact that it takes into account enhanced visual impact, which improves the persuasiveness of the video. On the other hand, hiring video professionals does not necessarily improve the effectiveness of the video, increases the production cost, and might result in creating irrelevant video scenes, since the videos that can be created for elicitation meetings are based on rather abstract requirements [8, 18].

1.2 Goal Definition

Software developers might want to produce vision videos on their own. However, there needs to be a guideline for software developers to produce their own vision videos, since they are most likely amateurs in video production. Karras and Schneider [19] have developed a ready-to-use guideline for producing vision videos, based on theoretical knowledge and practical experience that have been gained in recent years by utilizing vision videos in requirements engineering. However, an easy-to-use tool which analyzes the technical characteristics of vision videos and specializes in extracting these features is not yet available. Such a tool could assist software developers in producing vision videos, by giving quick feedback on specific characteristics of the video, which decreases the time it takes to detect technical deficiencies and improve the video.

The purpose of this thesis is to develop a web application which assists users, especially software developers, to be able to automatically analyze the technical details of their vision videos and provide them feedback for improvement. This was achieved by integrating various video analysis tools such as FFmpeg to analyze the video technical details, as well as ffprobe for shot detection. The details of how these analysis tools are used will be elaborated in the implementation chapter.

Other than the analysis functionality, the web application should also be intuitive to use and utilize excellent user experience (UX) practices. Furthermore, an effective way to visualize the extracted data and compare it to the recommended values is essential. Each of the analyzed characteristics should have an easily comprehensible rating. These allow the users to get quick feedback on the features that are lacking and quickly allow them to get a brief overview of the video quality.

Moreover, an easy way for users to store their videos and past analysis is required. This necessitates the usage of a user authentication and registration system. A video upload and video playback functions are also necessary, in order for the users to be able to analyze and store their videos. Each video should have a data model which lists all their relevant characteristics, that are relevant to the video analysis. These characteristics should be saved in the database after the analysis has been executed, so that a user only needs to analyze a particular video once. This allows faster performance of the web application and allows the users to easily compare past analysis.

In order to effectively integrate these features into the web application, a scalable and powerful server-side web framework is needed. Django was chosen as the web framework to fulfill this role. It provides many built-in functions such as user authentication and template system, which allows a rapid development of the web application. Due to the fact that it is widely used in the industry, it has a well defined standard and allows easier modification or extension of the web application, in case features need to be added or updated in the future.

By providing such an automated online service, it is hoped that developers will spend less time in doing analysis of the technical characteristics of their videos. As a result, developers have more time to focus on the actual content of the vision videos.

1.3 Structure of the Thesis

The content of the thesis is structured as follows. The first chapter serves as an introduction to the topic of the thesis by providing the motivation and proposed solution, as well as the structure of the thesis. Chapter 2 provides an overview of the fundamental concepts and tools that were utilized to achieve the goals of the thesis. In addition, information about other scientific papers that are related to the topic at hand are discussed in this chapter. In chapter 3, the requirements that are essential to be taken into account during the development process are elaborated. Furthermore, the design process that has been undertaken to create a GUI prototype of the project, as well as the stakeholders of the software, are described. Chapter 4 describes the implementation of the back end and front end of the web application, as well as the concepts used. In chapter 5, the evaluation process and results are described in order to ensure that the web application fulfills the goals of the project. The threats to validity and discussions of the video analyzer are also described in this chapter. Chapter 6 concludes the thesis and provides a summary of the whole thesis, as well as providing a concise outlook for future improvements and extensions of the video analyzer.

Chapter 2

Fundamentals

In this chapter the fundamentals which are required to understand the vision video analyzer are explained. Section 2.1 explains the process of requirements engineering in general, with a focus on requirements elicitation. Vision videos and their usage are elaborated in section 2.2. Section 2.3 outlines the general idea of the video analysis required in order to assist the users in improving their videos' technical quality and explains the characteristics that could be analyzed by the web application. Meanwhile, section 2.4 gives a brief description of the tools and libraries that are used for the analysis. Section 2.5 explains the role of Debian 10 as the back end operating system of the server. The next section describes the Django framework and its usage, as well as its scope in the web application. Lastly, scientific publications that are related to the vision video analyzer are discussed in the final section of this chapter.

2.1 Requirements Engineering

In order to understand what a vision video is, it is mandatory to first understand requirements engineering. According to Nuseibeh and Easterbrook [25], requirements engineering is a methodical procedure to discover the purpose of a software system, identify stakeholders and their demands, and document these in a form tractable for communication, analysis, and subsequent implementation. The reference model in Fig 2.1 by Börger et al. [4] outlines the components of requirements engineering.

The aspect of the reference model which is relevant to vision video is the requirements analysis phase. It refers to the first phase of requirements engineering before the approval of the project specification and the start of the development phase. It consists of 5 stages, namely: *Elicitation*, *Interpretation*, *Negotiation*, *Documentation*, and *Validation/Verification*. These stages are elaborated in more detail below, with descriptions provided by Schneider [32]:

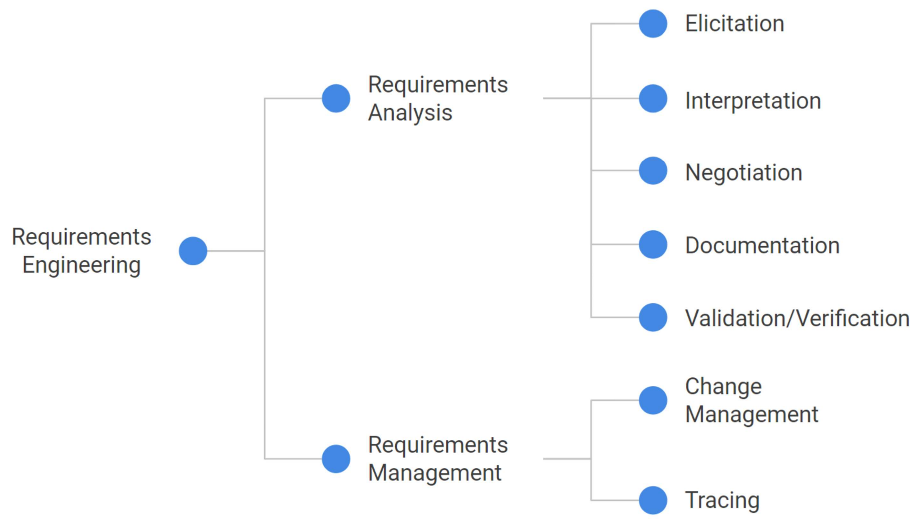


Figure 2.1: Reference model of requirements engineering [4]

- **Elicitation**

  Elicitation is a stage of gathering requirements from the client. The first stage of the process is the identification of stakeholders, which refers to individuals, groups, or organizations that are impacted by the project. After that, the environment that is related to the project must be surveyed. These can be old systems, interfaces, or related documentations. Furthermore, interviews with the stakeholders must be conducted in order to obtain their requirements. Lastly, these rough requirements are refined into more understandable requirements that can be analyzed in the interpretation stage.

- **Interpretation**

  In this stage, the requirements that were gathered in the elicitation stage are interpreted in order to check for what is essential, as well as if the meaning of the requirements is correct. The requirements are then structured and concretized into more formal requirements.

- **Negotiation**

  Afterwards, dependencies and conflicting requirements are identified. Conflicts and inconsistencies with the clients' requirements should also be handled in this stage, with the goal of finding a compromise which satisfies all parties that are involved.

- **Documentation**

After the requirements and other necessary information are gathered, they should be formally documented in this stage in order to get a fixed project specification. However, the requirements may still change, provided that these changes are documented afterwards.

- **Validation/Verification**

  This is the last stage of the requirements analysis process. Validation refers to the process of checking if the content of the produced specification matches with what the client wishes. On the other hand, verification is a formal process which checks if the specification matches with the previously documented requirements.

2.2 Vision Video

Vision video is defined by Karras et al. [20] as:

> "A video that represents a vision or parts of it for achieving shared understanding among all parties involved by disclosing, discussing, and aligning their mental models of the future system."

They are used during the elicitation stage of requirements analysis to demonstrate how the developers conceive the product would function and if the client expects the same. Creighton et al. [8] proposed that the lack of a clear vision is one of the crucial issues in requirements engineering, which is one of the main reasons why vision videos could serve as a supplement to written documentation.

2.3 Video Analysis

In this thesis, the scope of the video analysis will be on analyzing the technical characteristics of the video that can be extracted by utilizing the web application. In Fig 2.2, Karras et al. [20] have developed a model of a hierarchical decomposition of video quality. This model helps in decomposing the concept of video quality into individual aspects, which can be examined to determine the quality of the whole video.

The focus of this thesis will be on the representation dimension. It includes the factors from the video stimuli, such as image quality, sound quality, and video length. The characteristics outlined below are the chosen characteristics of a vision video, which has an impact on the representation dimension of the video quality.

- **Resolution**

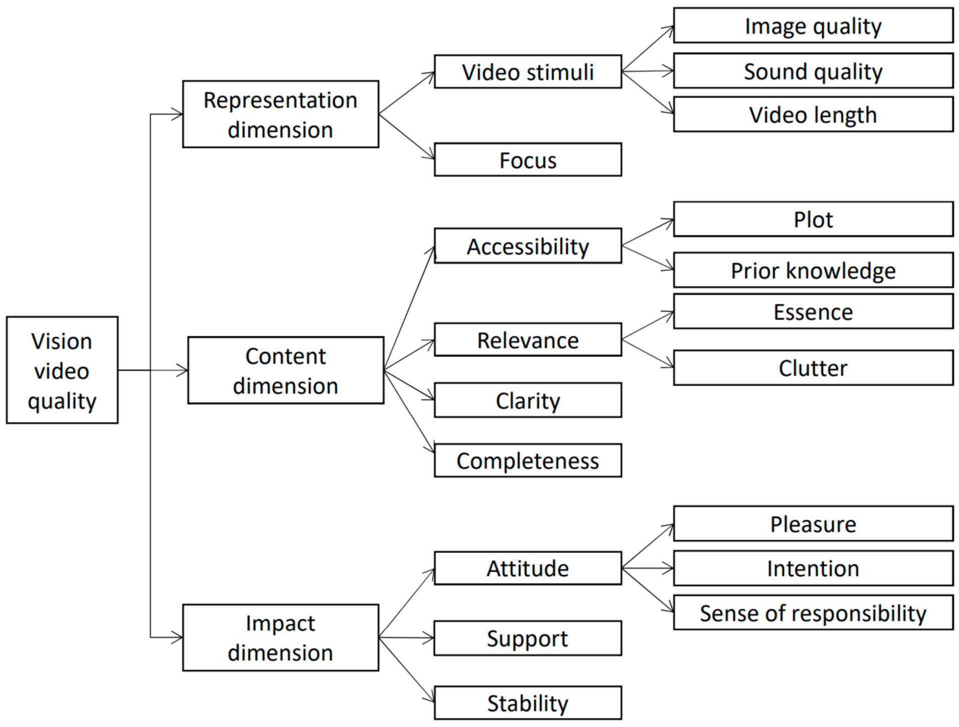


Figure 2.2: Hierarchical decomposition of vision video quality [20]

Resolution refers to the number of distinct pixels that can be displayed in a screen. It is measured in a *width* x *height* format and uses *pixels* as a unit. A higher resolution increases the image quality aspect of the video.

- **Frame rate**

  Frame rate is the number of individual frames that are displayed per second. It is measured in frames per second (FPS). Although a higher frame rate does not imply a higher image quality, it increases the overall quality of the video by reducing the duration between frames, which is related to the video stimuli aspect of the quality model.

- **Bit rate**

  Bit rate refers to the number of bits that are transmitted per unit of time. According to Google, LLC [13], video bit rate is typically measured in Mbps and audio bit rate in kbps. A higher bit rate increases the image and sound quality of a video. However, a higher bit rate requires more bandwidth.

- **Bit depth**

Bit depth is defined as the number of bits that make up a single pixel. According to Winken et al. [37], videos that are encoded in H.264 video compression standard, which are most videos that are produced, have a bit depth of 8-bit. Increasing bit depth to 10-bit increases the range of color that can be perceived in each pixel. Therefore, the choice of the video bit depth has a direct impact on the video image quality.

- **Sample rate**

  Sample rate refers to the number of samples per second taken from continuous audio signal converted to discrete time signal. According to Shannon [33], a sampling rate of more than twice the maximum frequency of the signal to be reproduced is required to reconstruct the original audio sample without the distortion known as *aliasing*. Therefore, a sampling rate higher than 40kHz is recommended, which is twice the limit of human audible frequency range. Maintaining the sample rate above the limit helps in ensuring the sound quality of the video.

- **Video length**

  The length of the video is important to take into consideration. According to Karras and Schneider [19], the length of the video should be short enough (up to 5 minutes) to ensure that only important details are shown to the audience.

2.4 Tools and Libraries

In order to extract the characteristics that are outlined above, several external tools and libraries are utilized by the web application. These tools provide command-line interface (CLI) access, which allows them to be executed by the Python subprocess module in the back end. These are:

- **FFmpeg**

  FFmpeg[1] is an open-source framework which consists of various tools for handling media files, such as video and audio. It provides some functionalities that will be required by the video analyzer, such as extracting a thumbnail from a video. Other than that, FFmpeg also offers ffprobe as a part of the framework, which is the main tool that is used for extracting video characteristics.

- **ffprobe**

[1]https://www.ffmpeg.org/

ffprobe[2] is a tool for extracting media characteristics. It is a part of the FFmpeg framework and is the main tool that is used for extracting the video characteristics.

- **MediaInfo**

  MediaInfo[3] is another open-source, cross-platform tool for extracting media metadata. It is used as a complement to ffprobe to extract video metadata.

- **OpenCV**

  OpenCV[4] is an open-source computer vision library. It is used to extract background color, determine contrast, and other usages that require computer vision.

- **PySceneDetect**

  PySceneDetect[5] is a Python tool which assists with shot segmentation and video splitting. It is the main tool that was used for shots analysis.

2.5 Debian 10

Debian 10[6] is a free and open-source Linux distribution, which serves as the operating system for the back end server of the web application. According to Amor et al. [1], Debian is not only the largest Linux distribution, but also one of the most stable ones. Since it is based on a Linux kernel, it allows for a simpler CLI execution of the tools that are required for the video analyzer.

2.6 Django Framework

Django[7] is a free and open-source Python web framework that is used for programming the web application. It handles both the front end and the back end, due to the fact that it is a full stack framework. It comes fully loaded with an object-relational mapper (ORM) to describe the database layout in terms of Python code, administrative interface, as well as a template system to design the site layout. By default, it provides integration with SQLite database. According to the SQLite website [34], the database is limited to the size of 281 terabytes, which should be enough for video analysis usage at the current scale. Furthermore, Django provides built-in security features

[2]https://ffmpeg.org/ffprobe.html
[3]https://mediaarea.net/en/MediaInfo
[4]https://opencv.org/
[5]https://pyscenedetect.readthedocs.io/en/latest/
[6]https://www.debian.org/releases/buster/
[7]https://www.djangoproject.com/

such as protection against cross site scripting (XSS), cross site request forgery (CSRF), and SQL injection.

Django uses a form of MVC pattern called MTV, which stands for *model*, *view*, and *template*. The *view* in traditional MVC pattern refers to the *template* in Django, since the template defines how the user perceives the data. Meanwhile, *controller* in traditional MVC refers to the *view* in Django, since the application logic which relays data to the template resides in the *views.py* in Django. Similar to traditional MVC, the *model* in Django refers to the database of the web application. The details of how this is accomplished will be elaborated in more details in the implementation chapter.

2.7 Related Work

The research on analyzing the quality of vision videos is not a new topic. In the past few years, there has been much scientific literature that focuses on this area of research. This section describes the various attempts by other researchers to analyze the quality of vision videos.

- **Quality Model for Vision Videos**

  Karras et al. [20] identified 15 quality characteristics that make up the overall quality of vision videos. These characteristics are combined to form a quality model for vision videos. The model is represented in two different ways: a hierarchical decomposition of vision video quality and a mapping of these characteristics to video production and use process. Furthermore, an experiment with 139 students shows that 6 characteristics (*video length*, *focus*, *prior knowledge*, *clarity*, *pleasure*, and *stability*) are significantly correlated with the probability that a vision video is rated as 'good' by the observer. As an extension to this paper, Karras and Schneider [19] developed an experienced based, ready-to-use guideline on the production of vision videos. The selections of technical characteristics that are analyzed by the vision video analyzer are based on this guideline.

- **Automating Guidelines for Video Production**

  Happe [15] has developed a prototype of a video recorder named *ViViRecorder*, which assists users in creating high quality vision videos. Instead of using the guidelines from Karras and Schneider [19], video production guidelines from sources such as MIT and Massive Open Online Course (MOOC) researchers were used [14, 23, 24]. By utilizing hardware sensors, the program is capable of detecting light level, noise, camera orientation and camera stability. If the user surpasses a certain limit, *ViViRecorder* provides recommendations on adjusting the related environmental variables. This is similar to how the vision video analyzer provides recommendations to users based on the technical aspects of the video. However, our tool differs by measuring other factors instead such as bit rate, bit depth and video length, by using the guidelines by Karras and Schneider [19].

- **Automation of Subjective Video Quality Measurements**

  Joskowicz et al. [16] presented the design, development, and deployment of a software which automates subjective video quality measurements. It allows researchers and developers to quickly analyze users' subjective rating of a video. The system has been used in 25 sessions with 85 subjects with excellent results according to the researchers. However, contrary to the vision video analyzer, the software is not specialized in analyzing vision videos. A similar tool, known as *Feedback Recorder*, which specializes in subjectively assessing vision videos, has also been developed by Sankaranarayanan [31]. These tools might be helpful in complementing the vision video analyzer, since it does not assess the subjective qualities of vision videos.

- **Impact of Video Quality on User Engagement**

  Dobrian et al. [9] investigated how video quality affects user engagement. They measured quality metrics such as average bit rate, buffering ratio, and rendering quality to check how these metrics affect user engagement. The degree of impact of each metric was also compared, as well as whether the critical quality metric differs across content genres. A relevant implication of their findings to this thesis is how they emphasize the importance of some quality characteristics, such as the average bit rate, on how engaged users are on the video. Therefore, they may also guide technical solutions on trade-offs between quality metrics, such as whether to focus on bit rate or rendering quality, to maximize efficiency given the limited resources of the developers. A key difference from this thesis is how their analysis focuses on live or streaming video content, which may not be very relevant for the production of vision videos.

Chapter 3

Requirements

In this chapter the requirements which were taken into consideration during the development of the application are explained. Section 3.1 explains the design process of creating the web application, followed by the description of the stakeholders in section 3.2. Furthermore, the functional and non-functional requirements are elaborated in section 3.3 and 3.4 respectively.

3.1 Design Process

The design process that was used to develop the web application is a mixture of both agile and traditional software development methodologies. The process began with the analysis of the features to be developed. After taking the deadline into account, a weekly iteration plan to encourage small releases was established. The whole plan was divided into two major iterations, namely the first and second iteration. The first iteration prioritized the development of the core functionalities of the web application. Meanwhile, the second iteration focused on debugging, polishing, and the addition of optional features.

Before the development phase started, a prototype of the web application was designed using Figma. Figma[1] is a web-based design tool that is used to create software prototypes. After the design was approved, the development cycle began. The prototype only served as a guideline and not as the final design of the project, in order to keep the design agile and reactive to change.

During the development phase, the Kanban agile method was adopted by utilizing Trello[2] as a Kanban board. Each week, the features to be developed were broken down into tasks that were listed in Kanban story cards. During the development of each feature, integration tests were executed by inspecting the behaviour of the feature in the whole web application, in order to maintain continuous integration. At the end of each week, a meeting with

[1]https://www.figma.com/
[2]https://www.trello.com/

the supervisor was held, to discuss the work done during the week, problems encountered, as well as the plan for the next week.

In order to maintain code consistency, Google's Python coding standard was used. This was implemented by using YAPF[3], a Python formatter by Google Open Source. By utilizing Git Hooks, YAPF is executed after each commit to ensure that the code is always properly formatted.

The development process followed the Extreme Programming (XP) agile methodology by utilizing aspects such as coding standard, small releases, and continuous integration.

3.2 Stakeholders

The target audience of the video analyzer is software developers, who wish to develop vision videos by themselves. Since it can be assumed that most software developers are not video professionals, the application should be accessible to users without expert video knowledge. Moreover, it is important that the application does not hamper the developers' productivity. Therefore, the UX and the user interface (UI) of the application should enable rapid and clear feedback, as well as being easy to use. Since vision videos might include confidential information about a company's or organization's project, the application should be sufficiently secure from malicious attacks.

Another target audience is computer science students. The application might assist them in the production of vision videos during software projects. In order to encourage usage, the application should be simple to install, deploy, and utilize.

3.3 Functional Requirements

- **[R1]** *The application should be developed as a web application.*

  Building the application as a web application enables the possibility of actualizing the vision video analyzer as a remotely hosted web service.

- **[R2]** *Django should be used as the web framework.*

  Django enables quicker development and it is widely used in the industry, which allows for more related documentation and modules.

- **[R3]** *Debian 10 should be the back end operating system.*

  It offers a lightweight, stable, Linux-based operating system to execute all the required modules.

[3]https://opensource.google/projects/yapf

- **[R4]** *The application should have a user authentication system.*

  User authentication provides security and enables personalized video analysis data.

- **[R5]** *Videos should be uploadable and playable.*

  It allows users to upload video for analysis and easily review each video.

- **[R6]** *Personalized analysis data should be stored in the database.*

  After analyzing once, it should no longer be necessary to analyze the video again. This reduces the time taken to obtain video analysis data.

- **[R7]** *Resolution, frame rate, bit rate, bit depth, sample rate, and video length of the whole video should be analyzed.*

  These characteristics are relevant in evaluating the quality of the whole video.

- **[R8]** *Shot length, background color, and contrast of each shot of the video should be analyzed.*

  These characteristics are relevant in evaluating the quality of each shot of the video.

- **[R9]** *Each characteristic should be rated individually.*

  Individual ratings allow users to receive feedback and improve specific characteristics.

- **[R10]** *Each rating should include visualization and improvement suggestions.*

  Characteristic visualization improves the user interface and allows users to observe certain characteristics easily. Suggested improvements provide quick feedback on how to improve these characteristics.

- **[R11]** *Each characteristic should have an explanation of its relevance to the video quality.*

  Not all users understand the relevance or definition of each characteristic. Therefore, an explanation for each characteristic is essential.

3.4 Non-functional Requirements

- **[NR1]** *The application should be simple and intuitive to use.*

  Simplicity and intuitiveness reduce the learning curve of the application.

- **[NR2]** *The feedback of each characteristic should be clear and actionable.*

  Clear and actionable feedback aids users in discovering and improving video defects.

- **[NR3]** *The application should be easy to install and deploy.*

  Easy deployment and installation improve UX, which might increase user adoption rate.

- **[NR4]** *The application should be extendable.*

  Extensibility is an important feature, since it is likely that additional characteristics, which may impact video quality, need to be added.

- **[NR5]** *The application should be secure from malicious attacks.*

  Security is important, due to the fact that vision videos might contain confidential information.

- **[NR6]** *The application should be as performant as possible.*

  Performance is important in creating a smooth UX and enabling faster analysis feedback reception.

- **[NR7]** *The database should support sufficiently large data size for videos and shots.*

  Users might need to store multiple variations of videos, each with multiple shots. This necessitates a database which supports sufficiently large data size.

- **[NR8]** *The application should be robust and easy to maintain.*

  The application should function well, even if there are OS or module updates.

Chapter 4

Implementation

In this chapter, the implementation of the web application is elaborated. Section 4.1 explains the concepts of how video analysis, shots analysis, as well as the feedback system work in the web application. Section 4.2 describes the detailed implementation of the web application's back end. Meanwhile, section 4.3 focuses on the implementation of the front end.

4.1 Concepts

This section explains the terms used in the context of the implementation of the web application.

4.1.1 Definitions

- **[D1]** *Video Analysis*

  Video analysis refers to the analysis of only the video characteristics that can be derived from the video without splitting the video into different shots.

- **[D2]** *Shots Analysis*

  Shots analysis refers to the characteristics of the video after the video has been split. The algorithm to determine the shot segmentation is also included in shots analysis.

- **[D2]** *Feedback System*

  The feedback system refers to both the characteristic rating and feedback. It is used to tell the users if the value of each characteristic is within the optimal range or not.

4.2 Backend

This section describes the implementation of the backend.

4.2.1 Database and Models

A database is necessary in order to store user data, videos, and other relevant information. Django provides official support for five databases: PostgreSQL, MariaDB, MySQL, Oracle, and SQLite. The web application uses SQLite[1] as the database. According to the SQLite website [34], the database supports a maximal size of 281 Terabytes, which was deemed sufficient in order to store vision videos. Furthermore, it was also easy to implement due to the fact that Django uses SQLite as a default database configuration setting.

According to the Django documentation [11], a model is a single, definitive source of information about relevant data. It contains the behavior of the data, as well as attributes that represent the data. Each model is a subclass of `django.db.models.Model`. The fields of the database are represented by the attributes of a model. In the web application, two models were used: User and Video.

The Django admin site allows administrators to manually edit the content of the database. It was enabled by default when the project was started. By utilizing the interface, administrators are capable of editing the usernames, changing the passwords, deleting the videos, and many more capabilities.

The User model stores information that is related to the user. It is an implementation of `django.contrib.auth` User model. The fields that are used are: `username`, `email`, `password`, `is_staff`, and `is_superuser`. The last two fields are reserved for administrator accounts, which allows them to access the Django admin interface. It also has an attribute `is_authenticated`. This attribute is globally accessible by the whole application to check whether the user has logged in successfully or not. Certain parts of the website, such as video listing, are disabled when the `is_authenticated` field is set to false. An instance of the User model is generated whenever the user successfully registers their account. Currently, deletion of the model is only possible through the admin interface, since a feature for users to delete their own account is not implemented yet.

The Video model stores information that is related to the uploaded video. The whole video characteristics, rating and recommendations are stored in the model with corresponding field types. The characteristics, video, name, upload date, and uploader fields are populated right after a video has been uploaded by the user. The fields other than the characteristics and feedback are populated by directly instantiating the video object with their respective parameters (`name`, `video`, `uploader`, `thumbnail`). After the instantiation, a

[1]https://www.sqlite.org/

wrapper function called `analyze_video()` is called in order to bind the values of the extracted characteristics from the video analyzer into their respective fields of the Video model. The details of `analyze_video()` will be elaborated in the video analyzer section.

4.2.2 Video Analyzer

The characteristics of the video are extracted by using separate functions. These functions are called by the `analyze_video()` wrapper function, which is called after a video has been uploaded. This section lists the characteristics of the whole video that are analyzed.

Resolution

Resolution is defined as the number of distinct pixels in each dimension that can be displayed by the video, in the form of *width x height*. In order to analyze the resolution of the video, ffprobe was used. This algorithm shows how it is implemented in the web app:

```
# Use ffprobe to get the video resolution
def get_resolution(video):
    video_input_path = f'{videos}/{video}'
    resolution = subprocess.run([
        'ffprobe', '-v', 'error', '-select_streams', 'v:0', '-
                                        show_entries',
        'stream=width,height', '-of', 'csv=s=x:p=0 ',
                                        video_input_path
    ],
                                capture_output=True,
                                text=True,
                                input="Y")
    return resolution.stdout
```

A Python subprocess allows the application to spawn a process, which is ffprobe in this case. It connects the input and output pipes. Using `-v error` sets the logging level and allows all errors arising from the process to be displayed. Next, `-select_streams v:0` was used to select only the streams specified, which is the video stream with index 0 in this case. `-show_entries stream=width,height` was used in order to retrieve only the width and height of the video, since those are the only data points required to calculate the resolution of the video. `-of csv=s=x:p=0` sets the output printing format and specified that x should be used as a separator between the width and height, in order to retrieve *width x height* as the output format.

The output of `get_resolution()` is directly bound with the resolution field of the Video model when the wrapper `analyze_video()` is called after a video has been uploaded. Due to the fact that ffprobe very quickly retrieves the resolution value, it was deemed unnecessary to use a distributed task queue system like Celery[2], unlike during the shot analysis.

Frame rate

Frame rate is defined as the number of frames per second of the video. Similar to the resolution, ffprobe was used to extract the frame rate from the video. The code is very similar to the resolution, whereby a Python subprocess was utilized to spawn an ffprobe process. The only difference is that `-show_entries stream=r_frame_rate` was used in order to extract the frame rate. The output of the process is a string a/b, where a is the number of frames and b is the number of seconds required for the video to display a.

Due to the fact the measurement of the frame rate in the web app is FPS or frames per second, `split()` was used to separate the output. After that, these strings are converted into integers and then divided. NumPy[3] was used in order to get a rounded output of the division. Just like the resolution, the output is directly bound to the Video model.

Bit rate

Bit rate is defined as the number of bits that are processed by the video per second. The web app currently focuses on the video bit rate instead of the audio bit rate, since audio quality is already covered by the sample rate and audio is not the focus of the video analysis. Unlike frame rate and resolution, MediaInfo was used to extract the bit rate. The algorithm below shows how the bit rate extraction is done:

```python
# Get video bit rate using MediaInfo and format it to mbps
def get_bit_rate(video):
    video_input_path = f'{videos}/{video}'
    media_info = MediaInfo.parse(video_input_path)
    bit_rate = media_info.video_tracks[0].bit_rate
    return str(np.round(np.divide(int(bit_rate), 1000000), 1))
```

Unlike earlier, a wrapper of MediaInfo from PyPi[4] was used as a Python module in order to ease the implementation. Since the output of the analysis function is in bits, it needs to be converted to Megabits by using NumPy. The output of `get_bit_rate()` is directly bound with the video model.

Bit depth

Bit depth is defined as the number of bits used to indicate the color of a single pixel of the video. Similar to bit rate extraction, MediaInfo was used

[2]https://docs.celeryproject.org/en/stable/
[3]https://numpy.org/
[4]https://pypi.org/project/MediaInfo/

to extract the bit depth. Since the output of the function is already in bits, there was no need to convert it to any other values using NumPy.

Sample rate

Sample rate is defined as the number of times a sound is sampled per second. It is measured in kilohertz. Similar to resolution and frame rate, ffprobe was used in order to extract it. The only difference is in the stream and entry selection. Audio stream was used instead of video. Since the output is in hertz, NumPy was used to convert it to kilohertz.

Video length

Similar to the sample rate, video length was extracted by using ffprobe. The duration is already a specific entry of ffprobe, so only `-show_entries format=duration` was needed in order to point to the video duration. The output is rounded into integers, since there seems to be little value in showing the users how many miliseconds exactly a video lasts. For the purpose of vision video analysis, video duration in seconds was deemed to be sufficiently accurate.

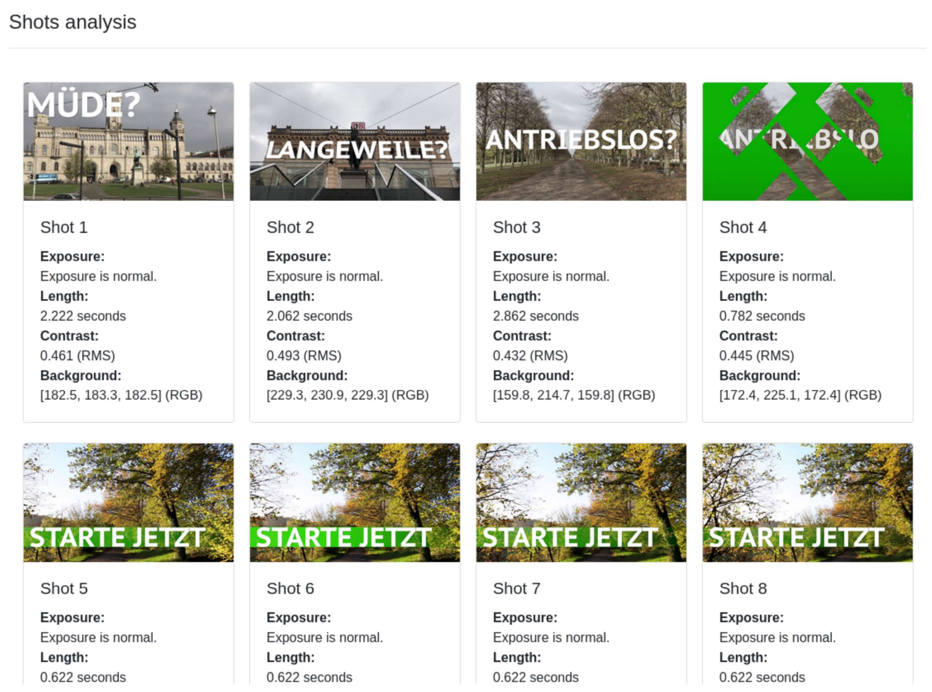
4.2.3 Shots Analyzer

Shot segmentation

In order to analyze the shots that make up the whole video, it is necessary split the video into individual scenes. PySceneDetect was utilized to achieve this. There are two scene detection algorithms that are provided by PySceneDetect: *content-aware detector* and *threshold detector*.

The *content-aware detector* works by finding the area where the difference of the content between two subsequent frames exceeds a certain threshold. The content itself is based on the color and intensity of the frame and it is known as *content value*. Meanwhile, the *threshold detector* works by comparing the threshold with the frame's brightness. Scene changes are triggered whenever the brightness value exceeds the predefined threshold. The value is calculated by computing the average RGB values for each pixel in the frame. According to Castellano [7], *threshold detector* is how most traditional scene detection algorithms function.

The *content-aware detector* was determined to be the more effective detection algorithm for the shot analyzer. This is because it can detect minor, abrupt changes, which the *threshold detector* cannot do well. In order for the detection algorithm to work, a set threshold must be provided. PySceneDetect provides a default threshold, which is 30. However, the default value may be oversensitive to changes in scenes. This may lead to false positives in scene detection. Fig 4.1 shows an example of oversensitive scene detection. Notice how shot 4 is separated into multiple shots, even though all of them belong to the same shot.



Shots analysis

Shot 1
Exposure:
Exposure is normal.
Length:
2.222 seconds
Contrast:
0.461 (RMS)
Background:
[182.5, 183.3, 182.5] (RGB)

Shot 2
Exposure:
Exposure is normal.
Length:
2.062 seconds
Contrast:
0.493 (RMS)
Background:
[229.3, 230.9, 229.3] (RGB)

Shot 3
Exposure:
Exposure is normal.
Length:
2.862 seconds
Contrast:
0.432 (RMS)
Background:
[159.8, 214.7, 159.8] (RGB)

Shot 4
Exposure:
Exposure is normal.
Length:
0.782 seconds
Contrast:
0.445 (RMS)
Background:
[172.4, 225.1, 172.4] (RGB)

Shot 5
Exposure:
Exposure is normal.
Length:
0.622 seconds

Shot 6
Exposure:
Exposure is normal.
Length:
0.622 seconds

Shot 7
Exposure:
Exposure is normal.
Length:
0.622 seconds

Shot 8
Exposure:
Exposure is normal.
Length:
0.622 seconds

Figure 4.1: An example of oversensitive scene detection

In order to remedy this issue, a custom algorithm that automatically determines the threshold was developed. The algorithm starts with the premise that a threshold of a particular frame is an actual and possible threshold, if there are no frames around it with similar change in threshold value. Algorithm 1 shows the pseudocode of the algorithm. The custom algorithm parses the generated CSV file from PySceneDetect statistics output. The content values and the frame numbers are then extracted from the CSV file. The difference between the content values of neighboring frames is calculated, as well as the absolute difference in the content values. The former is known as *content value difference* and the latter as *absolute content value difference*.

A threshold is considered a possible threshold if its *content value difference* is above the default threshold of PySceneDetect (which is 30). Therefore, only each possible threshold is added to the CSV in order to save time taken to analyze the possible thresholds. A sliding window approach was used to ensure that only one threshold exists within a given window. The size of the window is determined by the frame rate of the video for one second of playback time. For example, the size of the window for a 60 fps video is 60 frames. After each possible threshold in the video is marked,

Algorithm 1: A custom algorithm for finding the threshold for scene segmentation

```
1  function getThreshold ($video$);
   Input   : Name of the video $video$
   Output: Integer representing the new threshold value $threshold$
   /* Initialize the algorithm                                              */
2  Set the paths for the generated threshold CSV and storage of threshold box
     plots using $video$;
3  $change\_vals := []$ ;          // An array for a tuple of ($frame\_number$,
     $change\_val$), where $change\_val$ is change in content values
4  $borders := []$ ;                        // An array for possible thresholds
5  $threshold := 30$ ;                      // Default PySceneDetect threshold
6  $window\_size :=$ frames per second of $video$;
7  Generate a CSV file containing $frame\_number$, $content\_val$, $change\_val$,
     $abs\_change\_val$;
   /* Parsing data from the generated CSV                                   */
8  foreach $row \in CSV$ do
       /* Initial pruning to optimize time taken to check possible
          borders                                                           */
9      if $row.change\_val > threshold$ then
10         Append the tuple ($row.frame\_number$, $row.change\_val$) to $borders$;
11     Append ($row.frame\_number$, $row.change\_val$) to $change\_vals$;
   /* Container for pruned borders to avoid directly mutating the
      borders list                                                          */
12 $pruned\_borders := borders$;
   /* Prune possible borders by using the sliding window technique         */
13 foreach $border \in borders$ do
       /* Define bounding region for each window                            */
14     if $border.frame\_number - window\_size < 1$ then
15         $region\_start := 0$;
16     else
17         $region\_start := border.frame\_number$ - $window\_size$;
18     if $border.frame\_number + window\_size > change\_vals.length - 1$ then
19         $region\_end := change\_vals.length$ - 1;
20     else
21         $region\_end := border.frame\_number + window\_size$;
22     $region :=$ Region of $change\_vals$ between $region\_start$ and $region\_end$;
23     $region\_vals :=$ Array of all $change\_val$ in the defined region.;
           /* Generate boxplot for each region to see if each possible
              border is outside of normal distribution                      */
24     Generate boxplot with $region\_vals$ distribution;
25     if $border.change\_val \leq$ upper whisker of boxplot then
26         Remove $border$ from $pruned\_borders$;
   /* Obtain minimum threshold from the list of possible thresholds        */
27 Prune each $border$ in $pruned\_borders$ with content value less than initial
     threshold.;
28 return $min(pruned\_borders)$
```

the sliding window method was used to see if multiple possible thresholds exist within a single window. If that is the case, then a box plot will be used to determine if the possible thresholds are within normal distribution or below the upper whisker of the box plot. If they are within the normal distribution, then these thresholds are pruned, since it means that there may be many content changes occurring, but only on a particular scene. After this sliding window pruning is done, the default PySceneDetect threshold value is replaced by the threshold with the minimum *content value* out of the possible thresholds.

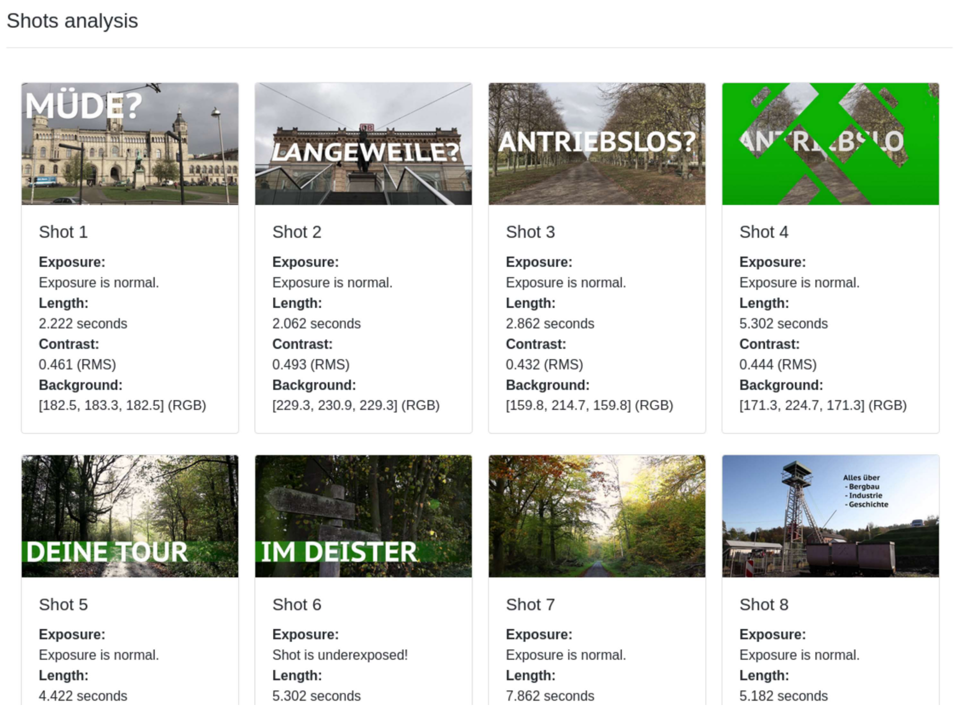
One important thing to note is that each border with less *content value* than the initial threshold is pruned. This is due to the fact that the next frame after a scene change occurs has a low *content value* (since it is similar to the previous frame, where scene change occurs). This sudden decrease in content value means that the *absolute content value difference* of that frame will be high. If this is not done, then the oversensitivity will normally be even higher than when we use PySceneDetect's default threshold. Another possible pruning which was done in the previous version of the algorithm is pruning each odd-indexed entry of the possible thresholds, since those entries have high *absolute content value difference* but very low *content value* because of the reason described above. However, an error was discovered after testing the algorithm on videos with some corruption in the encoding. FFmpeg skips those corrupted frames and an even-indexed entry with low *content value* appears in the possible thresholds, which renders the algorithm useless.

After applying the algorithm, false positives are removed from the scene list. Fig 4.2 shows the scene lists after the application of the custom algorithm. Notice how shot number 4 is consolidated into one shot, instead of being separated into multiple shots.

Exposure

Exposure is the measurement of the overall brightness or darkness of a particular shot. Overexposure is a state of being too bright, while underexposure is a state of being too dark. Histogram analysis was used in order to analyze the exposure of a shot. Fig 4.3, Fig 4.4, and Fig 4.5 by Rockwell [30] show the histogram of images that are underexposed, normal, and overexposed respectively. As can be seen from the figures, the exposure of the image can be detected by the distribution of pixel color.

The main idea behind the exposure analysis is to analyze the histograms of the screenshots of each shot. The screenshots were generated by `save-images -n 3` option of the `scenedetect` used in `get_shots()`, which was called before the exposure is analyzed. As can be seen from the option, three screenshots were generated for each shot. In order to reduce the time it takes for shot analysis and due to the fact that exposure rarely changes



Shots analysis

Shot 1
Exposure:
Exposure is normal.
Length:
2.222 seconds
Contrast:
0.461 (RMS)
Background:
[182.5, 183.3, 182.5] (RGB)

Shot 2
Exposure:
Exposure is normal.
Length:
2.062 seconds
Contrast:
0.493 (RMS)
Background:
[229.3, 230.9, 229.3] (RGB)

Shot 3
Exposure:
Exposure is normal.
Length:
2.862 seconds
Contrast:
0.432 (RMS)
Background:
[159.8, 214.7, 159.8] (RGB)

Shot 4
Exposure:
Exposure is normal.
Length:
5.302 seconds
Contrast:
0.444 (RMS)
Background:
[171.3, 224.7, 171.3] (RGB)

Shot 5
Exposure:
Exposure is normal.
Length:
4.422 seconds

Shot 6
Exposure:
Shot is underexposed!
Length:
5.302 seconds

Shot 7
Exposure:
Exposure is normal.
Length:
7.862 seconds

Shot 8
Exposure:
Exposure is normal.
Length:
5.182 seconds

Figure 4.2: Scene list after applying the custom algorithm

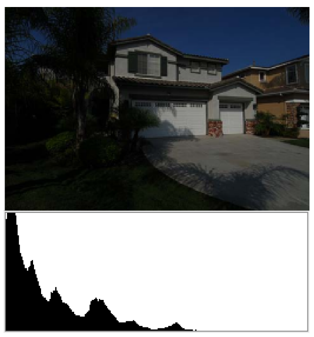


Figure 4.3: Underexposed image

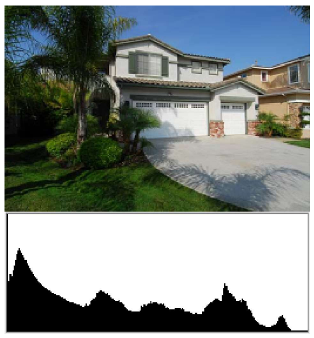


Figure 4.4: Normal image

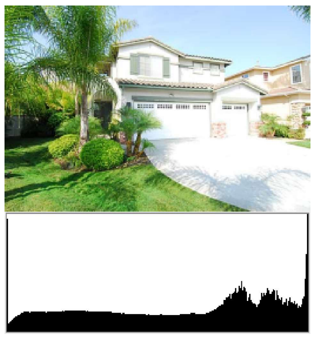


Figure 4.5: Overexposed image

during a continuous shot, three screenshots were determined to be sufficient in order to analyze the exposure of the shot.

HSV stands for hue, saturation, and value. It is an alternate representation of RGB color. OpenCV was used in order to read and convert each screenshot into HSV color space. The reason behind choosing HSV over RGB is because the grayscale value of a particular image can simply be calculated by extracting the V value of the HSV. It is also possible to use OpenCV's `BGR2GRAY` conversion if RGB was used instead, but using HSV is more intuitive because the value in V is by definition the brightness of a pixel.

After the conversion of the screenshot to grayscale, OpenCV's `calcHist()` was used in order to calculate the number of pixels of each color intensity, from a range of 0 (darkest) to 255 (brightest). After that, two variables which calculate the sum of the pixels within a particular threshold were defined, namely `dark_pixels` and `bright_pixels`. `dark_pixels` stores pixels with a value up to 50, meanwhile `bright_pixels` stores values from 200 to 255. Those thresholds were obtained by splitting the possible pixel values of 0 to 255 into five regions and taking the first and last region as underexposure and overexposure thresholds respectively. This separation of possible brightness values into five regions was inspired by Gibson [12]. The screenshot is considered underexposed if the sum of `dark_pixels` is more than half of the total pixels. Meanwhile, the screenshot is considered overexposed if the sum of `bright_pixels` is more than half of the total pixels. If neither of these conditions are fulfilled, it means that the screenshot has a normal exposure.

Shot length

The length of each shot can be calculated by using ffprobe on each of the shots that were split by the `get_shots()`. The implementation is similar to calculating the length of the whole video, namely by adding `-show_entries format=duration` as an entry to access the duration of each shot. These shots are then stored in a Python list and will be used to generate a JSON file containing the characteristics that are related to each shot.

Contrast

Contrast in the context of the vision video analyzer is known as the measurement of the difference in luminance or color which makes the pixels in a particular shot distinguishable from one another. There are multiple measurement techniques for contrast, some of the most commonly known are Michelson and RMS contrast. According to Kukkonen et al. [21], RMS contrast is better than Michelson contrast for complex or aperiodic luminance distributions, which is the case for vision videos. This is the reason why RMS contrast was chosen as the contrast measurement for the video analyzer, due to the fact that real world scenes are usually more complex in lighting distribution than artificial scenes. According to Peli [28], RMS contrast is defined as:

$$rms = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{1/2}, \tag{4.1}$$

where x_i is normalized gray-level value such that $0 \leq x_i \leq 1$ and $\bar{x}$ is the mean normalized grey level:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{4.2}$$

Here is a snippet of the Python code used to implement the RMS equation above:

```python
shots_output_path = f'{shots}/{video}/screenshots/'
contrasts = []
for filename in sorted(os.listdir(shots_output_path)):
    img = cv2.imread(os.path.join(shots_output_path, filename))
    if img is not None:
        # Use RMS contrast for contrast measurement
        img_grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        cv2.normalize(img_grey,
                      img_grey,
                      alpha=0,
                      beta=1,
                      norm_type=cv2.NORM_MINMAX)
        contrast = img_grey.std(ddof=1)
        contrasts.append(str(contrast))
```

As can be seen from the code above, the implementation of RMS contrast measurement is done with the help of OpenCV. Similar to exposure measurement, the contrast analysis is done on each set of three screenshots of all scenes in the video. Firstly, OpenCV is used to read the image file of the path provided. Null error handling was added to ensure that the file was actually an image file and not the statistics file of the screenshots that remains as an output of ffprobe. If the image is not null, then it is converted into `COLOR_BGR2GRAY`, which is defined as an RGB/BGR to grayscale color space by OpenCV. Afterwards, the `normalize()` method was used. It is a built-in operation of OpenCV which is used to normalize the norm or value range of the array. The first two parameters are source and destination paths of the file to be normalized. Since we want to overwrite the grayscale image with its normalized version, the path for source and destination is the same. The parameters `alpha` and `beta` are the lower and upper range boundaries of range normalization. These are set to 0 and 1 respectively to obey the normalized gray-level value range of the RMS contrast definition. Lastly, `norm_type` determines the normalization type of the function. `NORM_MINMAX` is an enumerated type which determines the minimum value of the destination as `alpha` and its maximum value as `beta`. After normalizing the grayscale image, the standard deviation of the image is calculated by using NumPy's `std` function. The optional parameter `ddof` stands for degree of freedom. The divisor used in calculation is N - `ddof`, where N is the number of values in the array and 0 is the default value of `ddof`. Therefore, the `ddof` needs to be manually set to 1 in order to follow the defintion of RMS contrast. Lastly, the mean contrast of each set of three screenshots is calculated in order to determine the contrast value of each shot that is displayed to the user. This is done simply by using NumPy to calculate and round the mean value to 3 decimal places.

Background color

The background color is the color that is mostly present in the background of each particular shot. In order to extract the background color, a method for background color extraction is needed. One possible way to do this is by extracting the most common colors that are present in a particular shot. Under the assumption that the majority of the pixels in a particular shot represents the background and that the background color consists of a gradient of colors with similar values, it is possible to calculate a mean RGB value which represents the background color. Parker [26] has done a similar approach to extract the background color of an application by using OpenCV and Python. The algorithm used to extract the background color from each shot is based on his algorithm. It works by setting a list of counters for the possible RGB values in the shot. OpenCV is then used to read the image and detect the RGB value of each pixel. If the RGB value of the pixel was already detected, the detected value in the counter will be incremented. If not, then a new entry for the aforementioned RGB value will be appended to the list and its count set to 1.

Afterwards, a sample of the 5 most common RGB values (RGB value counters with the highest count) are taken and their mean value calculated. This differs from Parker's implementation, which uses 10 samples. The reason for this change is that after testing with various vision videos, it was discovered that artificial elements (especially texts) that were added during the video editing process have a huge impact on the mean RGB value. Due to the fact that these artificial elements tend to have a single or limited RGB gradient, it led to a high number of pixels with that RGB color. This leads to false positives in the calculation of the most common RGB values, since texts are not part of the background. Setting the sample size to 5 reduces the probability of this happening, whilst also taking into account the gradient color of the background of real life vision videos. The value of 5 was obtained from a heuristic analysis of the samples of vision videos provided by the SE institute of the Leibniz University of Hannover. As a future work, the sample size still needs to be verified by testing it with more vision videos to obtain the optimal value.

A special case of the background color detection occurs when the percentage of a particular RGB value is over 80% of the whole shot. This usually occurs during credits or openings, where the background is fully black or white. In this case, the approach of calculating the most common RGB values and calculating their mean value will not work well, since the background will almost certainly have the color of the value which covers over 80% of the shot. If we take into account the other values, it will only spoil the resulting background color. An example of this is a credit scene with black background color and white credit texts. If we use the previous algorithm, it will output gray as the mean background color, since the mean between black and white is gray. In order to remedy this, a special case which only returns the color which covers over 80% of the shot was added. Similar to the previous algorithm, the value of 80% was determined from a heuristic analysis of the samples of vision videos. More research is needed in order to obtain the optimal value.

Lastly, the average background of each set of three screenshots of a particular shot is calculated in order to determine the background color value that is displayed to the user. This approach is similar to the RMS contrast extraction.

4.2.4 Feedback System

The feedback system consists of the recommended value, the rating, and feedback or recommendation on how to improve each characteristic of the vision video. Various scientific publications were utilized in order to determine the recommended values. In case no relevant literature could be found for a specific characteristic, values derived from analyzing the vision videos of the software engineering institute of the Leibniz University of Hannover were used instead. Each characteristic feedback is also accompanied by a visualization in order to make it easier for the user to understand it and visualize where the current value lies, in relation to the optimal value region. This sub section explores the detailed explanation of the feedback for each characteristic analyzed by the web application.

Resolution

According to Karras and Schneider [19], a vision video should use at least 720p or optimally at least 1080p resolution. The reasoning behind this is that a higher resolution increases the image quality and therefore improves the perceived video stimuli quality by the audience of the vision video. The guideline does not provide any upper limit on the resolution of vision videos. Furthermore, according to Donelan [10], even at very high resolution such as 8K, the human visual system can still pick up on finer gradations between colors, with less banding and tighter dynamic range mapping, so the viewer sees a more realistic, lifelike image. Therefore, an upper limit of resolution for vision video is still unknown and requires further research. By following this guideline, a rating of "Okay" is given to videos with resolution between 720p and 1080p, while a rating of "Great!" is given to videos with at least 1080p resolution. Videos which have a resolution below 720p are rated as

"Bad" by the rating system. In addition to the rating, recommendations to improve the characteristic to its optimal range are also provided. Here are the recommendations depending on the region where the video resolution value lies:

- **1080p and above** *Great video resolution!*
- **Between 720p and 1080p** *Your resolution is okay. However, you can still improve it by increasing the resolution to 1080p.*
- **Below 720p** *Your resolution is too low. Please increase it to at least 720p.*

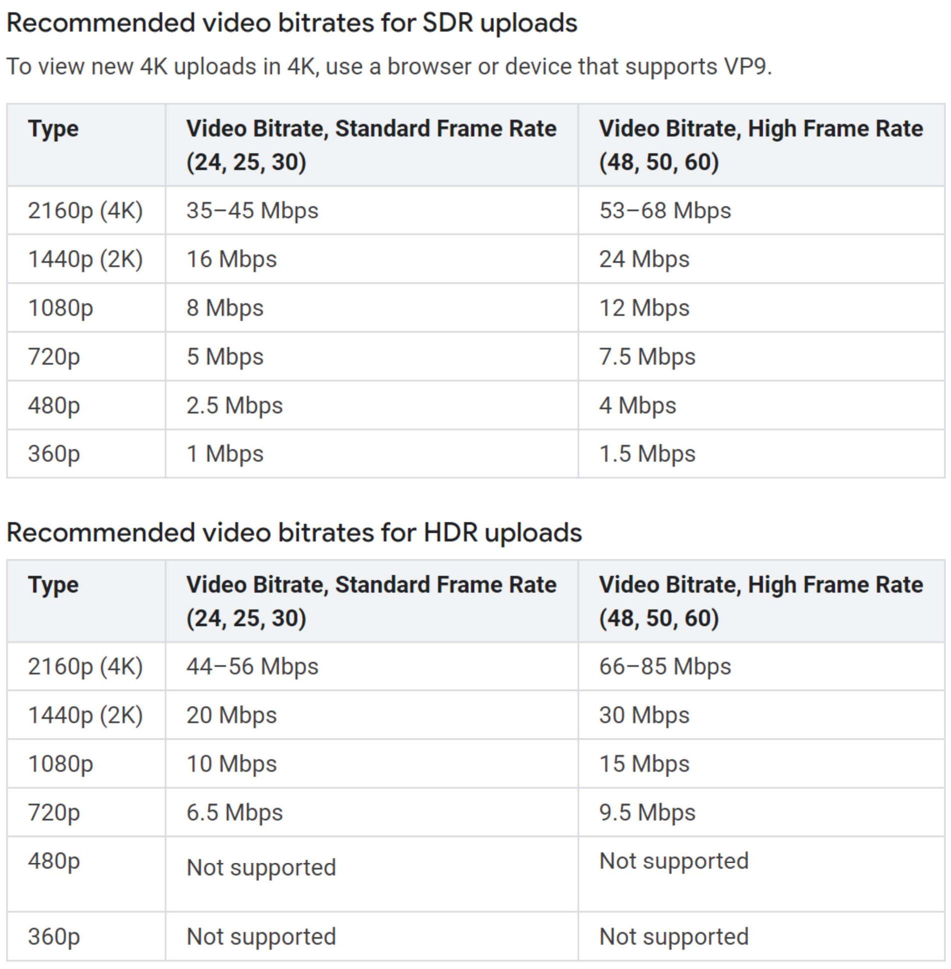
Frame rate

According to Kurniawan and Hara [22], most video content uses the standard 24 fps in order to achieve a classic cinematic look. Utilizing 30 fps is also useful in order to show clear movement. Furthermore, 60 fps also increases the smoothness of action in the video. However, frame rates above that, such as 120 fps, are only necessary for slow motion videos and videos with very fast actions such as sports games. Weech et al. [36] have also discovered that there is a diminishing return for perceived visual display quality improvement after 60 fps. After taking these findings into consideration, the optimal range of frame rate for vision video purposes was determined to be between 24 and 60 fps inclusive. By following this guideline, a rating of "Great!" is given to the region between 24 and 60 fps. A rating of "Bad" is given to frame rate below 24 and above 60 fps. Here are the recommendations depending on the region where the video frame rate value lies:

- **Between 24 and 30 fps** *Your frame rate is great! You can increase it to 60 fps if you wish to capture fast moving footage.*
- **Between 30 and 60 fps** *Great frame rate! You can lower it to 24 fps if you want a more cinematic video look.*
- **Below 24 fps** *Your frame rate is low. Please increase it to at least 24 fps by changing the settings in your camera.*
- **Above 60 fps** *Your frame rate might be too high. Please lower it to at least 60 fps since such a high frame rate is unlikely needed for vision videos and has a diminishing return.*

Bit rate

The optimal bit rate value is dependent on the frame rate and the resolution of the video. According to Google, LLC [13], the recommended video bit rates are shown in Fig. 4.6.



Recommended video bitrates for SDR uploads

To view new 4K uploads in 4K, use a browser or device that supports VP9.

Type	Video Bitrate, Standard Frame Rate (24, 25, 30)	Video Bitrate, High Frame Rate (48, 50, 60)
2160p (4K)	35–45 Mbps	53–68 Mbps
1440p (2K)	16 Mbps	24 Mbps
1080p	8 Mbps	12 Mbps
720p	5 Mbps	7.5 Mbps
480p	2.5 Mbps	4 Mbps
360p	1 Mbps	1.5 Mbps

Recommended video bitrates for HDR uploads

Type	Video Bitrate, Standard Frame Rate (24, 25, 30)	Video Bitrate, High Frame Rate (48, 50, 60)
2160p (4K)	44–56 Mbps	66–85 Mbps
1440p (2K)	20 Mbps	30 Mbps
1080p	10 Mbps	15 Mbps
720p	6.5 Mbps	9.5 Mbps
480p	Not supported	Not supported
360p	Not supported	Not supported

Figure 4.6: Recommended bit rates by Google, LLC [13]

However, video bit rates are usually not exactly equal to the values provided by the recommendation. In order to handle this, an algorithm which determines the optimal region based on the recommendation was written. The algorithm determines if the frame rate of the video is high (above 30 fps) or low. After that, it retrieves the bit rate of the video and checks if it lies in the optimal region. Unfortunately, Google has only determined the optimal range for 4K videos. In order to remedy this, the optimal region is defined by the region between twice the recommended value as the maximum boundary and half of the recommended value as the minimum boundary, in order to set the maximum and minimum value of the optimal threshold. This region is determined by the observation that the optimal bit rate value of a specific resolution and frame rate tends to be about twice the previous value. This recommendation needs to be supported by more experimental data. However, it still serves as a decent heuristic, since it is unlikely that a particular video bit rate is optimal if it is less than half of the recommended value. Furthermore, bit rates that are higher than twice the recommended bit rate are most likely too high.

By following this guideline, a rating of "Great" is given to the region of the optimal bit rates, a rating of "Okay" is given if it is above it and a rating of "Bad" is given to the region below it. Here are the recommendations depending on the region where the video bit rate value lies:

- **Within the optimal region** *Great bit rate!*
- **Below the optimal region** *Your bit rate is too low. Please increase it to at least* ***minimum_ acceptable_ bit_ rate*** *Mbps.*
- **Above the optimal region** *Your bit rate might be too high. Please reduce it to at least* ***maximum_ acceptable_ bit_ rate*** *Mbps.*

Bit depth

According to Azimi et al. [2], a bit depth of 8-bit was deemed sufficient by the range of brightness supported by Standard Dynamic Range (SDR) technology. Nowadays, there is a trend for modern devices such as newer smartphones and laptops to support High Dynamic Range (HDR) technology. According to Azimi et al. [2], 10 to 12 bit depth is recommended by the BT.2100 standard for videos to be displayed in HDR. BT.2100 [6] is a technical standard from the International Telecommunication Union for the production or distribution of HDR content. By following this guideline, a rating of "Great!" is given to the region between 10 and 12 bits. A rating of "Okay" is given to to the region with 8 or 9 bits. A rating of "Bad" is given to bit depths below 8 bits. Lastly, a rating of "Unknown" is given to bit depths above 12 bits, since there is currently a lack of scientific literature nor any standard which regulates bit depths above 12 bits. Here are the recommendations depending on the region where the video bit depth value lies in:

- **Between 10 and 12 bits** *Great video bit depth! Your current bit depth supports HDR video technology.*
- **With 8 or 9 bits** *Your bit depth is good enough. However, consider increasing bit/color depth to 10 or 12 bits to increase the range of possible displayed colors and enable the usage of HDR video technology.*
- **Below 8 bits** *Your bit depth is too low. Try to increase it to at least 8 bits, in order to display a wider range of possible colors.*
- **Above 12 bits** *Your bit depth is very high. You may want to lower it to 10 or 12 bits to save data and bandwidth. It is unknown if increasing bit depth above 12 bits increases the perceptible quality of vision videos.*

Sample rate

According to Pras and Guastavino [29], 44.1 kHz is the standard sample rate for commercial release. Furthermore, they discovered that expert listeners are able to perceive the difference between 44.1 kHz and 88.2 kHz sample rates. Although they have observed these audible disparities, these differences are very subtle and difficult to detect. By drawing from this conclusion, a rating of "Great!" is given to the region between 44.1 kHz and 88.2 kHz. A rating of "Okay" is given to sample rates above 88.2 kHz, since sampling at such a high rate might result in unnecessary video size increase, but with negligible gains in perceptible audio improvement. Lastly, a rating of "Bad" is given to sample rates below 44.1 kHz, since it is below the standard sample rate and might result in aliasing. Here are the recommendations depending on the region where the video sample rate value lies:

- **Between 44.1 and 88.2 kHz** *Great audio sample rate!*
- **Above 88.2 kHz** *Your sample rate is too high. Try lowering it to at most 88.2 kHz in order to reduce the video size. Sampling at a higher rate might result in negligible gains in perceptible audio improvement.*
- **Below 44.1 kHz** *Your sample rate is too low. Try increasing it to at least 44.1 kHz to prevent sound distortion.*

Video length

According to Karras and Schneider [19], vision videos should be kept short (no more than 5 minutes). This is in order to ensure that only the relevant details are shown to the audience. However, there is no recommendation on the minimum length of vision videos nor any research that investigates it. Therefore, no minimum value is set for the optimal vision video length. By drawing from this conclusion, a rating of "Great!" is given to vision videos that last for less than 5 minutes. Lastly, a rating of "Bad" is given to vision videos that are longer than 5 minutes. Here are the recommendations depending on the region where the whole video length lies:

- **Less than 5 minutes** *The length of your video is already short enough to only show the important details to the audience.*
- **More than 5 minutes** *Keep the final video short to ensure that you only show the important details to the audience.*

Exposure

The implementation of the exposure rating system is simply based on the output of `get_exposure()` function. If the shot is overexposed or underexposed, then a "Bad" rating is given. Otherwise, the shot has a "Great" rating. Here are the recommendations depending on the exposure of the shot:

- **Underexposed** *Try increasing the brightness of the shot by using brighter lights and using lighter colored backgrounds.*
- **Overexposed** *Try reducing the brightness of the shot by avoiding strong lights and using darker colored backgrounds.*
- **Normal** *Your exposure is already great!*

Shot length

According to Karras and Schneider [19], the length of a shot should be between 15 and 30 seconds. Following this guideline, all shots that are within this range are rated as "Great", otherwise they are rated as "Bad". Here are the recommendations depending on the region where the shot length lies:

- **Under 15 seconds** *Try increasing the length of your shot to at least 15 seconds to enable audience to understand the information presented.*
- **Over 30 seconds** *Try reducing the length of your shot to at least 30 seconds. Avoid too long shots that the audience cannot understand.*
- **Between 15 and 30 seconds** *The length of your shot is great, since it is between 15 and 30 seconds.*

Contrast

Unfortunately, no relevant scientific literature could be found which defines the optimal RMS contrast range for videos. By analyzing the shots of the videos that were provided by the Leibniz University of Hannover, contrast values between 0.3 and 0.8 were determined to be of sufficient quality. More research is needed in order to obtain the optimal value range. All shots, that possess a contrast within that range are rated as "Great", otherwise they are rated as "Too low" or "Too high" depending on where they are. Here are the recommendations depending on the region where the contrast lies:

- **Below 0.3** *Try to increase your RMS contrast to at least 0.3. You can do this by either changing your settings or by choosing video backgrounds with colors which contrast with the subject.*
- **Over 0.8** *Try to decrease your RMS contrast to at least 0.8. You can do this by either changing your settings or by choosing video backgrounds with colors which contrast with the subject.*
- **Between 03 and 0.8** *The contrast of your shot is great!*

Background color

According to Karras and Schneider [19], black and strong background colors (red, yellow, or bright green) should be avoided. Unfortunately, the exact RGB threshold for these colors were not clearly defined. By analyzing the shots of the videos that were provided by the Leibniz University of Hannover, the following RGB range were determined: Between (150, 0, 0) and (255, 150, 150) for red, between (200, 200, 0) and (255, 255, 150) for yellow, between (0, 150, 0) and (150, 255, 150) for green, and between (0, 0, 0) and (20, 20, 20) for black. Further research is needed in order to clearly define these boundaries. Here are the recommendations depending on the region where the background color lies:

- **Red** *Please avoid using red as a background color to not distract the audience and to not modify apparent color of the subject.*
- **Bright green** *Please avoid using bright green as a background color to not distract the audience and to not modify apparent color of the subject.*
- **Yellow** *Please avoid using yellow as a background color to not distract the audience and to not modify apparent color of the subject.*
- **Black** *Please avoid using black as a background color to not distract the audience and to not modify apparent color of the subject.*
- **Others** *Your background color is normal and is neither strong red, yellow, bright green, nor black.*

4.3 Frontend

This section describes the implementation of the frontend.

4.3.1 Design System

The web application was designed by using Figma, which is an online tool for web and application prototyping. It was used because it is free, intuitive, and has various component libraries. One of these component libraries that was utilized to provide the GUI elements of the website is the Bootstrap 4+ UI Kit[5]. According to the Bootstrap[6] website, it is the world's most popular front end open source toolkit, which features a responsive grid system, prebuilt components and Javascript plugins. It serves as a CSS framework, which means that it will be unnecessary to write custom CSS in order to design the web application. Although using a CSS framework

[5]https://www.figma.com/community/file/832800692655327277
[6]https://getbootstrap.com/

tends to limit the design customizability of a web app, the benefits of rapid prototyping and pre-built components outweigh the downsides, since the design is not the main focus of the web app.

Django provides a built-in template system which can dynamically generate HTML pages. The web responses from `views.py` are rendered by this template system.

4.3.2 Page Layout

This subsection explains the page layout and the functionalities of the frontend. The screenshots of each page are included in the appendix of the thesis.

Home page

Fig. B.1 displays the home page of the web application if the user is logged in. The user could upload a video to be analyzed from this page.

Registration page

Fig. B.2 displays a registration form that the users can use to register an account on the web app. The design was implemented by using `django-crispy-forms`, which automatically fills in and renders the required fields defined in the `forms.py` of the registration module root directory. There is also a password validation feature, which ensures that the user chooses a strong password. A password is considered strong if it is valid according to the default settings of Django's `AUTH_PASSWORD_VALIDATORS`. The default validator states that the password should not be too similar to other personal information, it should contain at least 8 characters, it cannot be entirely numeric, and it cannot be from Django's list of 20000 most commonly used passwords. The details of how this is handled are elaborated in the documentation of Django password validator. Other fields such as email validation and password matching are automatically handled by Django.

Log in page

Fig. B.3 displays a log in form that the users can access their account from. If the user does not have an account, they can click on the create one here to be redirected to the registration page.

Videos page

Fig. B.4 displays the list of videos that the user has uploaded is shown in the form of cards. The user can click on the card image or the analyze button to go to the video page. There is also a function to delete a the video beside the analyze button.

Deletion page

Fig. B.7 shows a confirmation page after the user clicks on the delete button. It provides an additional layer of security from accidental click of the delete button.

Video page

Fig. B.5 shows the whole video analysis page. The user can play the selected video by using the HTML5 video player provided. On the right side, there is a summary providing the values of each characteristic that are analyzed for the whole video. The user can click on the hint button beside each characteristic value to find the definition of those characteristics. Below the summary, there is a button which analyzes the shots of the video, as well as a confirmation notice below the button in case the shots has been successfully analyzed. After the shots have been analyzed, the button turns green and goes from "Analyze shots" to "Shots analysis."

Below the video name and upload date, there is a section for a detailed analysis of each characteristic. Fig. B.6 shows a sample of the detailed analysis for resolution and frame rate. For each characteristic, there is the current video value, recommended value, rating, feedback, and visualization. The only exception to this is the video length. There is no visualization for it since the minimum recommended length is unknown.

The visualization of the boxplot is done with the help of Matplotlib[7]. It is a comprehensive Python library for generating visualizations. In the boxplot, the optimal threshold which is defined by the range of optimal values is drawn as a box. The red vertical line along the box plot signifies the current video value. The range of the data shows the range of acceptable, but not necessarily optimal values.

Shots analysis page

Fig. B.8 shows the shots analysis page. It is only accessible after the shots have been analyzed. Each individual shot that has been analyzed and segmented are shown as cards. The rating of the exposure, length, and contrast are shown on each card. If a particular characteristic is not within acceptable range, the font color of the rating changes to red. Each characteristic is also accompanied with explanation beside it that is accessible when the user clicks on the hint icon.

[7] https://matplotlib.org/

If the user clicks on the view details button, a popup showing the detailed analysis of each shot is shown. Fig. B.9 shows a sample of the exposure and shot length detailed analysis. The exposure is shown as a grayscale histogram, and the background color is shown as as a box filled with its respective color, as shown in Fig. B.10.

Chapter 5

Evaluation

This chapter elaborates the evaluation process in order to determine if the developed web application fulfills the goals of the thesis. Firstly, the goals of the thesis are defined by using the Goal-Question-Metric system, as well as the hypotheses for the experiment. After that, the planning and the design of the evaluation are explained. Next, the results are presented, along with the conclusions that can be drawn from them. Lastly, threats to the validity of the evaluation and discussions regarding the video analyzer are elaborated in the last sections.

5.1 Concepts

5.1.1 Preparation with Goal-Question-Metric (GQM)

In order to systematically plan the experiment, the Goal-Question-Metric (GQM) system was used to define the goals. According to Wohlin et al. [38], the GQM method is based on the premise that an organization needs to fulfill these steps in order to measure in a purposeful way:

1. Specify the goals for itself and the project,
2. Trace those goals to the data that is intended to define those goals operationally, and
3. Provide a framework for interpreting the data with respect to the stated goals.

The resulting measurement model which serves as a framework to assist those three steps is the GQM model. According to Basili et al. [3], the model has three levels in its hierarchical structure, which are defined as follows:

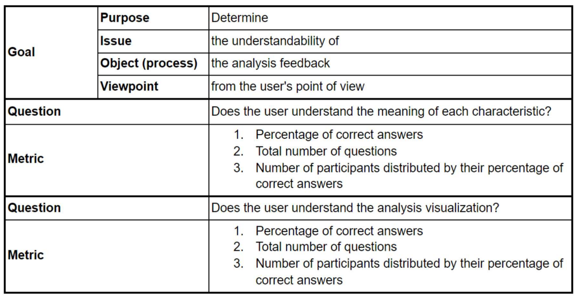
1. *Conceptual level* (**Goal**). A goal is determined for an object, for various reasons, concerning several quality models, from various perspectives,

relative to a particular context. Objects of measurement are products, processes, and resources.

2. *Operational level* (**Question**). A set of questions is utilized to characterize how evaluating a particular goal will be conducted based on some characterizing model. Questions attempt to characterize the object of measurement concerning a chosen quality issue and to determine its quality from the chosen perspective.

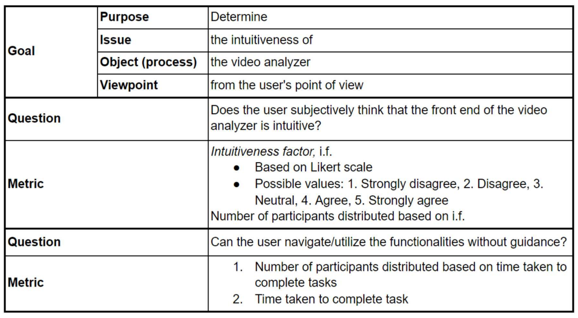
3. *Quantitative level* (**Metric**). A set of subjective or objective data is bound with each question in order to answer it in a quantitative approach.

By following the GQM model guidelines listed above, here are the goals of the evaluation:



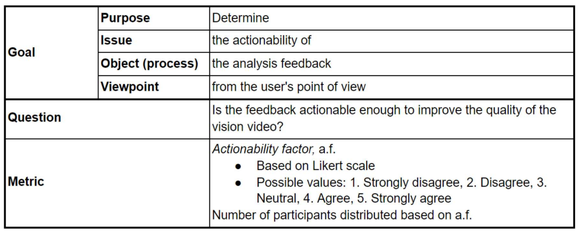
Goal	**Purpose**	Determine
	Issue	the understandability of
	Object (process)	the analysis feedback
	Viewpoint	from the user's point of view
Question		Does the user understand the meaning of each characteristic?
Metric		1. Percentage of correct answers 2. Total number of questions 3. Number of participants distributed by their percentage of correct answers
Question		Does the user understand the analysis visualization?
Metric		1. Percentage of correct answers 2. Total number of questions 3. Number of participants distributed by their percentage of correct answers

Figure 5.1: G1. Determine the understandibility of the analysis feedback



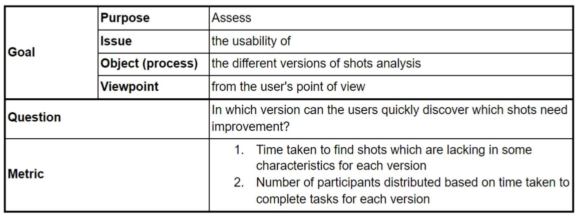
Goal	**Purpose**	Determine
	Issue	the intuitiveness of
	Object (process)	the video analyzer
	Viewpoint	from the user's point of view
Question		Does the user subjectively think that the front end of the video analyzer is intuitive?
Metric		*Intuitiveness factor,* i.f. ● Based on Likert scale ● Possible values: 1. Strongly disagree, 2. Disagree, 3. Neutral, 4. Agree, 5. Strongly agree Number of participants distributed based on i.f.
Question		Can the user navigate/utilize the functionalities without guidance?
Metric		1. Number of participants distributed based on time taken to complete tasks 2. Time taken to complete task

Figure 5.2: G2. Determine the intuitiveness of the video analyzer



Goal	**Purpose**	Determine
	Issue	the actionability of
	Object (process)	the analysis feedback
	Viewpoint	from the user's point of view
Question		Is the feedback actionable enough to improve the quality of the vision video?
Metric		*Actionability factor,* a.f. ● Based on Likert scale ● Possible values: 1. Strongly disagree, 2. Disagree, 3. Neutral, 4. Agree, 5. Strongly agree Number of participants distributed based on a.f.

Figure 5.3: G3. Determine the actionability of the analysis feedback



Goal	**Purpose**	Assess
	Issue	the usability of
	Object (process)	the different versions of shots analysis
	Viewpoint	from the user's point of view
Question		In which version can the users quickly discover which shots need improvement?
Metric		1. Time taken to find shots which are lacking in some characteristics for each version 2. Number of participants distributed based on time taken to complete tasks for each version

Figure 5.4: G4. Assess the usability the different versions of shots analysis

5.1.2 Hypotheses

The following hypotheses have been derived from the goals of the evaluation. For each hypothesis, the **IF** condition refers to the independent variable and **THEN** refers to the dependent variable. Each null hypothesis is represented with the notation of $Hi_0, i \in \{1, 2, 3, 4, 5, 6\}$. The following null hypotheses were made:

$H1_0$ (derived from Fig. 5.1) **IF** *less than half of the participants have a score of at least 60% on T2 in A.1,* **THEN** *the comprehensibility of each video characteristic meaning remains undecided.*

$H2_0$ (derived from Fig. 5.1) **IF** *less than half of the participants have a score of at least 60% on T3 in A.1,* **THEN** *the comprehensibility of the visualizations remains undecided.*

$H3_0$ (derived from Fig. 5.2) **IF** *the mean intuitiveness factor selected by all participants is less than 4,* **THEN** *the video analyzer is claimed to not be intuitive.*

$H4_0$ (derived from Fig. 5.2) **IF** *the mean task completion time is more than 10 seconds,* **THEN** *the navigability of the video analyzer without guidance remains undecided.*

$H5_0$ (derived from Fig. 5.3) **IF** *the mean actionability factor selected by all participants is less than 4,* **THEN** *the feedback is claimed to not be actionable enough.*

$H6_0$ (derived from Fig. 5.4) **IF** *the the mean time taken to accomplish the tasks listed in T6 for version B is equal to version A,* **THEN** *then the usability of both versions remains undecided.*

There are several reasons behind these. According to Tekian and Norcini [35], there is no "gold" standard for setting a passing score, but 60% is a typical, albeit imperfect, institutional passing score. This applies to both $H1_0$ and $H2_0$. Furthermore, the score of 4 represents "Agree" in the Likert scale used for the experiment, which is why it was selected as the minimum mean score for both $H3_0$ and $H5_0$. According to Pearson and Pearson [27], the user's ability to focus on website content diminishes after 10 seconds. This explains the reasoning behind $H4_0$. Lastly, mean task completion time is used to test the usability of both versions to gather empirical evidence in determining which version is more usable in $H6_0$.

It is assumed that for each null hypothesis, there exists an alternative hypothesis $Hi_1, i \in \{1, 2, 3, 4, 5, 6\}$, such that it is the opposite of its respective null hypothesis.

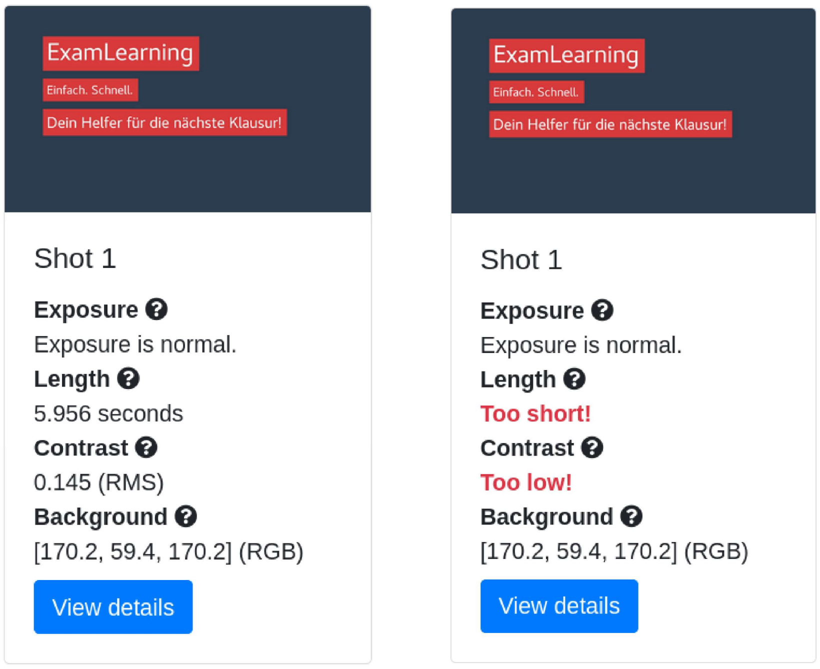
5.2 Planning and Design

5.2.1 Test subjects

A total of 17 participants took part in the survey. Among the participants, 2 are professional software engineers, 3 are PhD students, and 12 are bachelor students. Among the bachelor students, two also work as software test automation engineers in addition to their studies. All of the students studied computer science at the Leibniz University of Hannover. Furthermore, it could be observed from A.2 that 14 out of 15 students have passed the course *Software-Technik* (German for Software Engineering), 12 out of 15 have passed *Software-Projekt* (German for Software Project), and 5 out of 15 have passed *Software-Qualität* (German for Software Quality). Among the professionals, one is a back end software engineer and another one is a front end web developer. Both have 2 years of work experience in their respective fields. Meanwhile, the two bachelor students have 6 months and 1 year of work experience as software test automation engineers. Overall, it could be observed from the demographics that the participants have at least a basic understanding of programming and software engineering, which indicates that they are valid test subjects for the survey.

5.2.2 Experiment setup

Due to the fact that this experiment was done during the COVID-19 global pandemic, the experiment was conducted fully remotely as an online video interview. The web application was hosted on a server that was provisioned



ExamLearning

Einfach. Schnell.

Dein Helfer für die nächste Klausur!

Shot 1

Exposure

Exposure is normal.

Length

5.956 seconds

Contrast

0.145 (RMS)

Background

[170.2, 59.4, 170.2] (RGB)

View details

(a) Version A

ExamLearning

Einfach. Schnell.

Dein Helfer für die nächste Klausur!

Shot 1

Exposure

Exposure is normal.

Length

Too short!

Contrast

Too low!

Background

[170.2, 59.4, 170.2] (RGB)

View details

(b) Version B

Figure 5.5: Difference between version A and B

by the Leibniz University of Hannover. A test account was made that was then used by the participants. Four videos were pre-uploaded that were provided by the Software Engineering institute of the Leibniz University of Hannover. Three of these were pre-analyzed in order to save time. The experiment was conducted with the BigBlueButton [1] web conferencing platform.

Each participant was given a consent form to sign before the experiment. Before each session, the version of the web application that the participant will be exposed to was determined. Fig. 5.5 displays the only difference between the two versions, which is in how the shot characteristics are displayed on the cards. A voice recorder was set up before the start of each interview to analyze the results.

5.2.3 Experiment procedure

After greeting the participant, the idea of vision video, the concept of the web application, and the process of the interview were explained. Afterwards, the participant was asked about their background and consent. Next, the participant was given a username, password, as well as the link to the website

[1] https://bigbluebutton.org/

to log in.

The experiment was structured with a human-centered approach in mind. The participant was first requested to go to the video analysis page of the first video. Next, the participant was asked about the meaning of bit rate and bit depth. The participant was hinted that the answer could be found in the website, but not how to find the answer. Afterwards, the participant was requested to look at the visualization of the bit rate and was asked where the current bit rate value is in the box plot, if the bit rate is within acceptable range, and tell the optimal bit rate range from the visualization. Next, the participant was asked if they know what a shot is. A brief explanation of what constitutes a shot was provided to the participant in case they do not know. The participant was then told to find which shots are too long in the video.

After accomplishing the previous task, the participant was requested to navigate to and analyze the shots of the second video. In order to save time, the participant was asked to move on to the next video while the shots were being analyzed. The participant was then questioned about the meaning of sample rate and frame rate. Afterwards, the participant was asked questions regarding the frame rate visualization. The questions asked were identical to the bit rate questions. Afterwards, the participant was requested to delete the current video.

The participant was then told to go to the fourth video. After that, the participant was asked about the meaning of contrast and how it is measured in the context of the vision video analyzer. Next, the participant was requested to point out each shot that had a low contrast. After identifying the shots, the participant was requested to explain why those shots have low contrast from the visualization. Next, the participant was asked to go back to the second video and say what the meaning of exposure is. The participant was then asked to point out which shots are too bright. Finally, the participant was asked to describe what the visualization of the exposure would look like, if it were too dark instead of too bright, in order to check their understanding of the exposure visualization.

Afterwards, the participant was questioned if the feedback was understandable enough for them to improve the video, if they were to develop the vision video in real life. The participant was also asked if they subjectively think if the web application is intuitive to use. Both of these feedback were rated by using the Likert scale. Lastly, any personal feedback about the web application was asked and thus the session concludes. Each session typically lasts around 30 minutes.

5.3 Results and Analysis

The results are structured based on the goals that were defined by the GQM model. The complete result data set can be seen in A.3. This section merely attempts to recognize, describe, and explain the patterns that were apparent in the survey results and utilize these analyses to prove or disprove the hypotheses defined in the previous section.

5.3.1 Feedback understandibility

Video characteristics

Fig. 5.6 presents the number of answers each participant managed to answer correctly. The maximum number of correct answers is 6. The questions that were asked are all related to the meaning of video characteristics. The full list of the questions can be seen in T2 of A.1. Since there is no difference between version A and B of the web application for this part of the experiment, both versions were aggregated into the same statistics. Overall, the majority of the participants managed to answer all questions correctly, with two bachelor students answering one question incorrectly each. In general, therefore, it seems that the null hypothesis $H1_0$ is disproved, since more than half of the participants have a score of at least 50%.

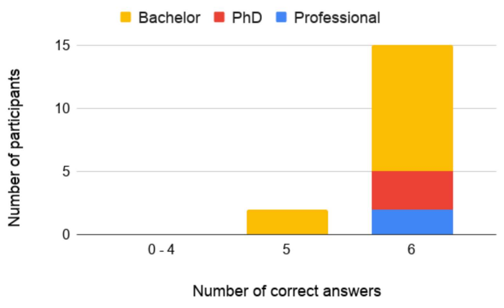


Figure 5.6: Meaning of video characteristics results

Visualizations

Fig. 5.7 shows the number of answers about the visualization of each characteristic that each participant answered correctly. Overall, all participants managed to answer at least 7 out of 8 questions correctly. The question that stumped 8 participants was: "What would the visualization of the exposure look like, if the shot were too dark instead of too bright?". From this, it can be assumed that either the visualization of the exposure is not clear enough or there needs to be more explanation for interpreting exposure visualization. The visualization of the exposure can be seen in Fig. B.9. These findings suggest that $H2_0$ is disproved for characteristic visualizations other than exposure.

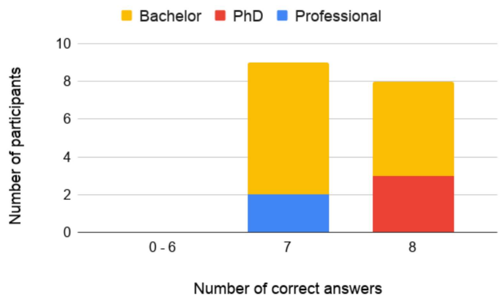


Figure 5.7: Visualization of video characteristics results

5.3.2 Intuitiveness of the web application

In order to test the intuitiveness of the web application, both subjective assessment and time taken to accomplish certain tasks were taken into account.

Subjective assessment

The results of the subjective assessment of the intuitiveness of both versions are shown in Fig. 5.8 and Fig. 5.9. The mean intuitiveness factors for version A and version B are 4.1 and 4.6 respectively. Closer inspection of Fig. 5.8 shows that two participants rated the intuitiveness of the video analyzer below 4. A possible explanation for this might be that the difficulty of quickly recognizing which shots need improvement in the shots analysis leads to a lower overall perception on the intuitiveness of the application. This might be because the only difference in the two versions is in how the shots characteristics are displayed, as can be seen in Fig. 5.5. Overall, these results indicate that the null hypothesis $H3_0$ is disproved, since the mean intuitiveness factor of both versions are above 4. This observation may support the alternative hypothesis $H3_1$.

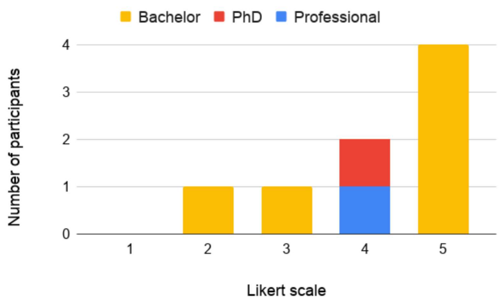


Figure 5.8: Intuitiveness factor (Version A)

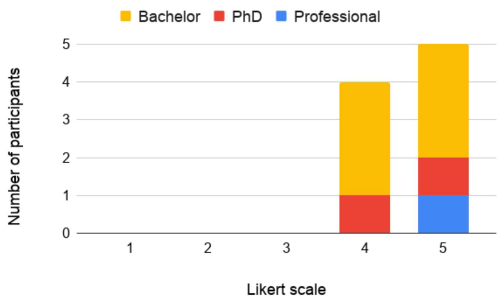


Figure 5.9: Intuitiveness factor (Version B)

Time taken to accomplish certain tasks

Other than the subjective assessment using Likert scale, it is also important to measure the time it takes to accomplish certain tasks and identify if there are any road blocks in the navigation process. Figure. 5.10 shows the mean task completion time for professionals, PhD, and Bachelor students interviewed. The list of tasks can be seen in T1 of A.1. These results seem to suggest that the professionals were the fastest in completing the tasks. However, the sample size of 2 might be too small to draw a conclusion from. On average, all groups managed to do all tasks, except task e, in under 10 seconds. Task e refers to the task where the user needs to delete the video. This discrepancy might be due to the fact that the delete button could only be found on the page which lists all videos. In order to delete the video, the user needs to navigate through multiple pages in order to access the button. A possible way to remedy this is to add a delete button on the video and/or shots analysis page, which was a recommendation from two participants. These findings seem to negate the null hypothesis $H4_0$, except for the deletion of the video.

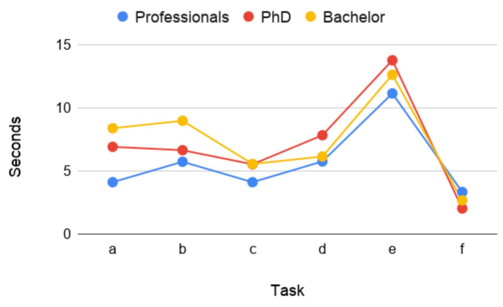


Figure 5.10: Mean task completion time

5.3.3 Actionability of feedback

The graphs that are shown in Fig. 5.11 and Fig. 5.12 highlight the difference between the actionability factors of all participants of the survey. As can clearly be seen from the graphs, both version A and version B of the web application have a mean actionable factor of 4.6 and 4.4 respectively. Due to the fact that the feedback provided in both versions is identical, the slight difference of actionability factor between the two versions might be caused by a small sample size of 17 participants. Hence the null hypothesis $H5_0$ is disproved, since the mean actionability factors for both versions are above 4. It can thus be suggested that the alternative hypothesis $H5_1$ is accepted.

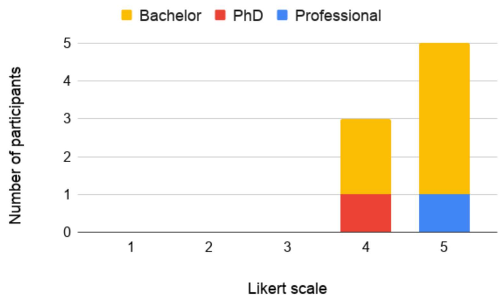


Figure 5.11: Actionability factor (Version A)

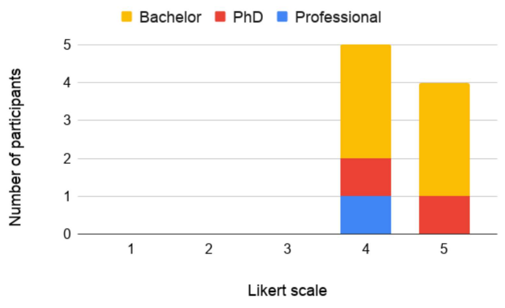


Figure 5.12: Actionability factor (Version B)

5.3.4 Usability between the two versions

Fig. 5.13 highlights the difference between the time it takes to accomplish the tasks listed in T6 of A.1 for version A and version B of the website. The tasks listed are all related to the shots analysis. What can clearly be seen in the graph is the difference between the two versions on task a and b, which is to find which shots are too long and which shots have low contrast respectively. The completion time is similar for task c, which is most likely due to the fact that the exposure serves as a control variable (identical for both versions). These findings suggest that displaying the rating instead of the value on the shots card has a huge impact on the task completion time. Overall, it can be inferred that the null hypothesis $H6_0$ is disproved from the findings discussed above.

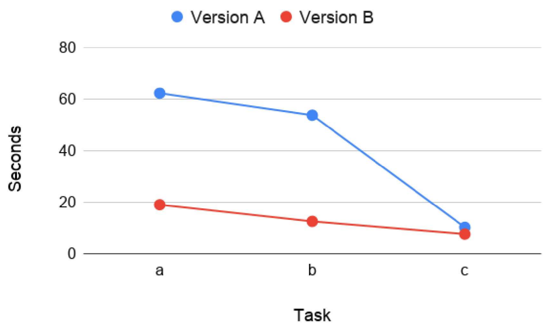


Figure 5.13: Comparison between version A and B task completion time

5.3.5 User feedback

At the end of each interview session, each participant gave feedback on how to improve the web application. Section A.4 on the appendix shows a list of the most common feedback that were given by the participants.

5.4 Threats to Validity

It is important to consider the threats that might influence the results of the evaluation. According to Wohlin et al. [38], there are four types of threats to validity. *Conclusion validity* refers to the relation between treatment and outcome of the evaluation. A small sample size is a possible threat of this type, since it leads to an issue of low statistical power. Seventeen participants with diverse, but relevant background (software professionals, PhD, and Bachelor students) were chosen in order to reduce the impact of small sample size.

Internal validity refers to matters that might influence the independent variable without the experimenter's knowledge. A major threat of this type is the inherent video knowledge of the participants. There were no questions during the survey which asked the participant about their background in video editing. In order to remedy this, a possible inference could be drawn from the results in how the participants answered the meaning of subject-specific terms such as bit depth. From observation, none of the subjects seemed to have answered those questions without utilizing the available hints, which might indicate that they are not video professionals. However, this must be proven with further testing.

Another type to consider is *construct validity.* It refers to the generalisation of the results from the evaluation to the theory behind the experiment. A possible threat to this is the exclusion of several factors, such as uploading the participants' own vision videos. It was not possible to distribute vision videos to participants because of copyright issues and the participants cannot be expected to possess their own vision videos. Since analyzing their own videos is a major part of the video analysis process, the lack of its inclusion might be a possible threat to validity of the conducted experiment. In order to reduce the threat, the participants were exposed to 4 different vision videos to provide more variation.

The last type to consider is *external validity.* It refers to the generalisation of the experiment to external environments other than the original experiment setting. A major threat of this type is that the experiment was conducted online and not onsite. In order to remedy this, videos were pre-uploaded to reduce impact of the participant's internet upload speed and live screen sharing was utilized in order to clearly track the behavior of the participant during the experiment.

5.5 Discussion

This sections discusses the problems encountered during the development of the video analyzer, as well as its current limitations.

5.5.1 Role of a feature in perceptible video quality improvement

The video analyzer managed to extract characteristics from a vision video. However, the impact of each of these characteristics on the overall perceptible quality of the vision video is still unknown. More research is needed in order to rank each of these characteristic depending on their impact and importance on the overall quality of the vision video.

5.5.2 Shutter speed

Shutter speed refers to the length of time that a camera shutter is exposed to light. An attempt to extract shutter speed from a video was done. However, it was unsuccessful since shutter speed is not included in .mp4 (and possibly other formats') metadata, which was why neither ffprobe nor mediainfo support shutter speed extraction from video. It is possible to take a reference picture for each scene and extract the shutter speed from the EXIF data of the picture taken. However, it does not seem possible to extract it from the video alone.

5.5.3 Types of vision videos

There are various types of vision videos, but two of the most prominent types are real life vision videos that are recorded by a camera and vision videos that are generated with a computer, for example a recording of a software prototype by using a screen recorder instead of a camera. The issue with the second type of vision video is that characteristics such as exposure and contrast, as well as functionalities such as shot segmentation, are very difficult to determine by using the algorithms that are used for real world vision videos, since the shots are artificial. Ratings that might be accurate for real life vision videos might be not accurate at all for artificial vision videos. In order to remedy this issue, it might be possible to construct a video analyzer with a trained deep learning model to automatically detect the type of vision video. Wu et al. [39] managed to propose a hybrid deep learning framework for video classification. In addition to this, a new rating system as well as shot analysis methods must be utilized. However, this is not within the scope of this thesis and remains as future work. Therefore, the video analyzer currently only supports real life vision videos.

Chapter 6

Conclusion

This chapter provides a summary of the most important topics that were discussed in the thesis. Furthermore, it provides an outlook on how the features of the video analyzer could be improved in related future works.

6.1 Summary

Vision videos can be very helpful for requirements elicitation. However, most developers are not video professionals. In the course of this thesis, a web application which analyzes the characteristics of vision videos was developed.

The main goal of the tool is to assist developers in quickly identifying technical deficiencies in their vision videos and give clear, actionable feedback on how to resolve them. Django was used as the web framework for the video analyzer, as well as various open-source libraries such as ffprobe and OpenCV to analyze the characteristics of the whole video and shots respectively. Matplotlib was also used to display the visualization of each characteristic in order to assist users in understanding the feedback.

By utilizing GQM, the most important goals of the tool were defined and a human-centered survey was designed in order to evaluate the video analyzer. Usability, intuitiveness, and feedback actionability were determined to be the most important goals and an online interview with 17 participants was conducted in order to test these factors.

Overall, the results obtained from the survey indicate that the tool is intuitive enough to use without guidance and that the feedback is actionable and understandable enough to improve the quality vision videos. Furthermore, identifying shots that are deficient in some characteristics became relatively fast by using version B of the video analyzer.

A possible use case for this tool is in assisting computer science students in improving their vision videos for software projects or for software professionals to quickly discover technical deficiencies in their software prototype videos and find ways to alleviate them.

6.2 Outlook

Although an evaluation was done in order to test the usability of the video analyzer, in-depth experiments for scalability and concurrency handling have not been done yet. These factors are important to test if the tool were to be hosted and utilized by multiple users at the same time. Furthermore, some rating recommendations such as contrast and background color were only based on the few (around 10) real-life vision videos that were provided by the Software Engineering institute of Leibniz University of Hannover. More testing with more variations of vision videos is necessary in order to adjust and optimize these rating values. An extension to the tool which detects the content of the video and adjusts the settings of the rating and shot analysis system, perhaps by using a deep learning framework for video classification, will be very useful since it will increase the variety of vision video types that can be analyzed by the video analyzer.

Appendix A

Evaluation Data

The details of the evaluation are included in this section.

A.1 Tasks list

A.1.1 [T1] Task completion time measurement

[a] *Start going to the video page of the Exam learning video.*

[b] *Try to analyze the shots of the Dashboard video.*

[c] *Start going to the video page of the Textomat video.*

[d] *Go to the detailed shot analysis of the first shot of the Textomat video.*

[e] *Try to delete the Textomat video.*

[f] *Start going to the video page of the Metis video.*

A.1.2 [T2] Meaning of analyzed characteristics

[a] *What is the definition of bit rate?*

[b] *What is the definition of bit depth?*

[c] *What is the definition of sample rate?*

[d] *What is the definition of frame rate?*

[e] *How is contrast defined and measured in the video analyzer?*

[f] *What is the definition of exposure?*

A.1.3 [T3] Visualization comprehension

[a] *Can you point where the current bit rate value is in the box plot?*

[b] *Can you tell me if the bit rate is within acceptable boundaries?*

[c] *Can you tell me the optimal range of acceptable bit rates?*

[d] *Can you point where the current frame rate value is in the box plot?*

[e] *Can you tell me if the frame rate is within acceptable boundaries?*

[f] *Can you tell me the optimal range of acceptable frame rates?*

[g] *Can you tell me why those shots have low contrasts from the visualization?*

[h] *What would the visualization of the exposure look like, if the shot is too dark instead of too bright?*

A.1.4 [T4] Actionability subjective assessment

[a] *If you were to develop the video in real life, would the feedback be understandable enough for you to improve the video? You can freely check out the feedback of the videos. Please rate it from a scale of 1 to 5. Where 1 is if you strongly disagree, 3 is neutral and 5 if you strongly agree.*

A.1.5 [T5] Intuitiveness subjective assessment

[a] *Do you think that the web app is intuitive enough to use? Please rate it from 1 to 5 like before.*

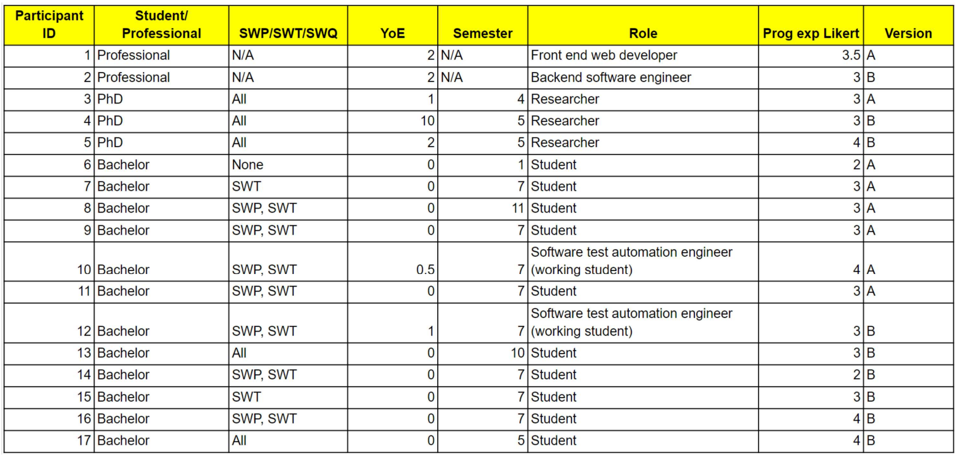
A.1.6 [T6] Task completion time comparison between different versions

[a] *Which shots are too long?*

[b] *Which shots have low contrasts?*

[c] *Which shots are too bright?*

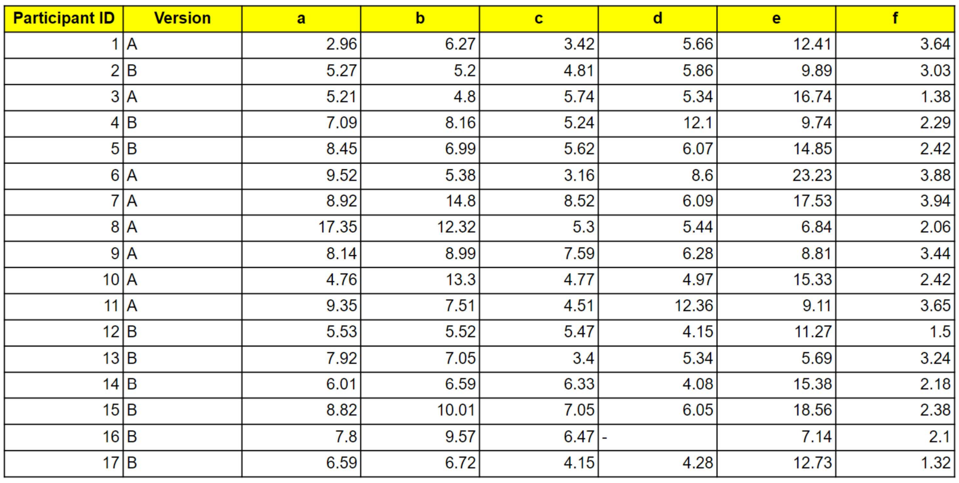
A.2 Participant details



Participant ID	Student/ Professional	SWP/SWT/SWQ	YoE	Semester	Role	Prog exp Likert	Version
1	Professional	N/A	2	N/A	Front end web developer	3.5	A
2	Professional	N/A	2	N/A	Backend software engineer	3	B
3	PhD	All	1	4	Researcher	3	A
4	PhD	All	10	5	Researcher	3	B
5	PhD	All	2	5	Researcher	4	B
6	Bachelor	None	0	1	Student	2	A
7	Bachelor	SWT	0	7	Student	3	A
8	Bachelor	SWP, SWT	0	11	Student	3	A
9	Bachelor	SWP, SWT	0	7	Student	3	A
10	Bachelor	SWP, SWT	0.5	7	Software test automation engineer (working student)	4	A
11	Bachelor	SWP, SWT	0	7	Student	3	A
12	Bachelor	SWP, SWT	1	7	Software test automation engineer (working student)	3	B
13	Bachelor	All	0	10	Student	3	B
14	Bachelor	SWP, SWT	0	7	Student	2	B
15	Bachelor	SWT	0	7	Student	3	B
16	Bachelor	SWP, SWT	0	7	Student	4	B
17	Bachelor	All	0	5	Student	4	B

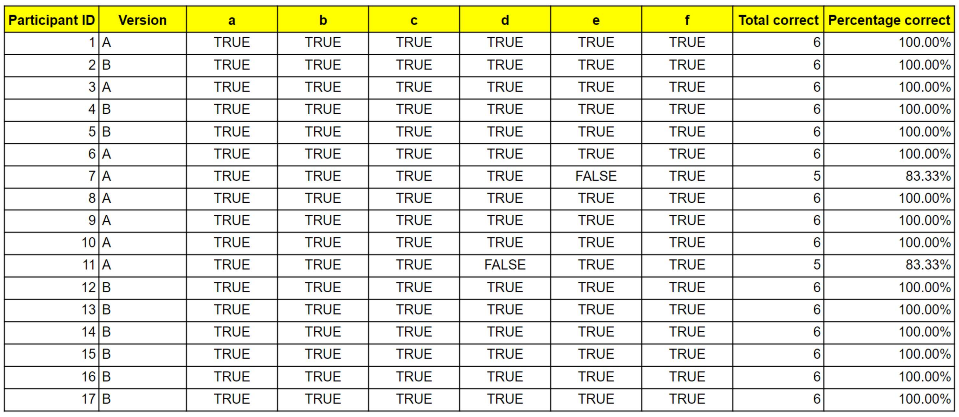
Figure A.1: Participant details

A.3 Detailed evaluation results



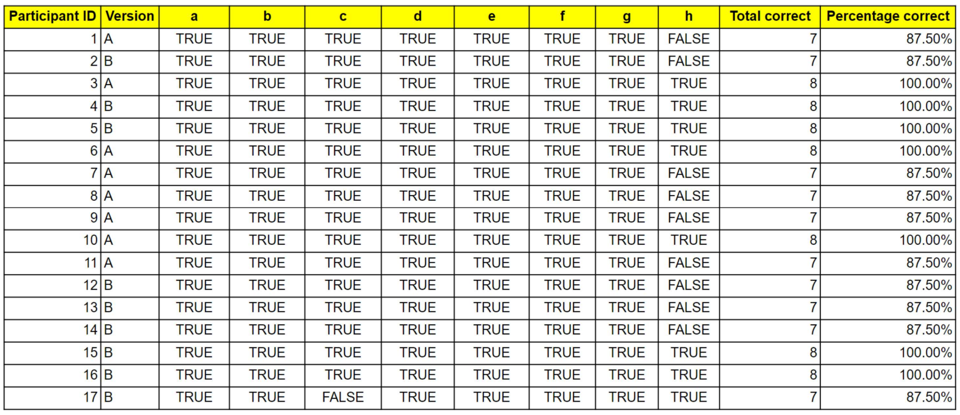
Participant ID	Version	a	b	c	d	e	f
1	A	2.96	6.27	3.42	5.66	12.41	3.64
2	B	5.27	5.2	4.81	5.86	9.89	3.03
3	A	5.21	4.8	5.74	5.34	16.74	1.38
4	B	7.09	8.16	5.24	12.1	9.74	2.29
5	B	8.45	6.99	5.62	6.07	14.85	2.42
6	A	9.52	5.38	3.16	8.6	23.23	3.88
7	A	8.92	14.8	8.52	6.09	17.53	3.94
8	A	17.35	12.32	5.3	5.44	6.84	2.06
9	A	8.14	8.99	7.59	6.28	8.81	3.44
10	A	4.76	13.3	4.77	4.97	15.33	2.42
11	A	9.35	7.51	4.51	12.36	9.11	3.65
12	B	5.53	5.52	5.47	4.15	11.27	1.5
13	B	7.92	7.05	3.4	5.34	5.69	3.24
14	B	6.01	6.59	6.33	4.08	15.38	2.18
15	B	8.82	10.01	7.05	6.05	18.56	2.38
16	B	7.8	9.57	6.47	-	7.14	2.1
17	B	6.59	6.72	4.15	4.28	12.73	1.32

Figure A.2: Task **[T1]**. Unit is in seconds.



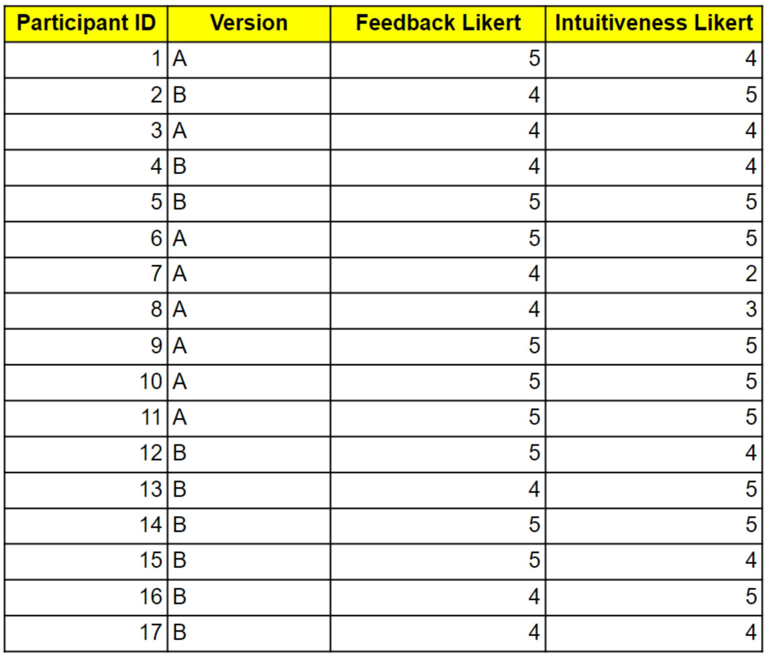
Participant ID	Version	a	b	c	d	e	f	Total correct	Percentage correct
1	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
2	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
3	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
4	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
5	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
6	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
7	A	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	5	83.33%
8	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
9	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
10	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
11	A	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	5	83.33%
12	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
13	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
14	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
15	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
16	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%
17	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	6	100.00%

Figure A.3: Task **[T2]**.



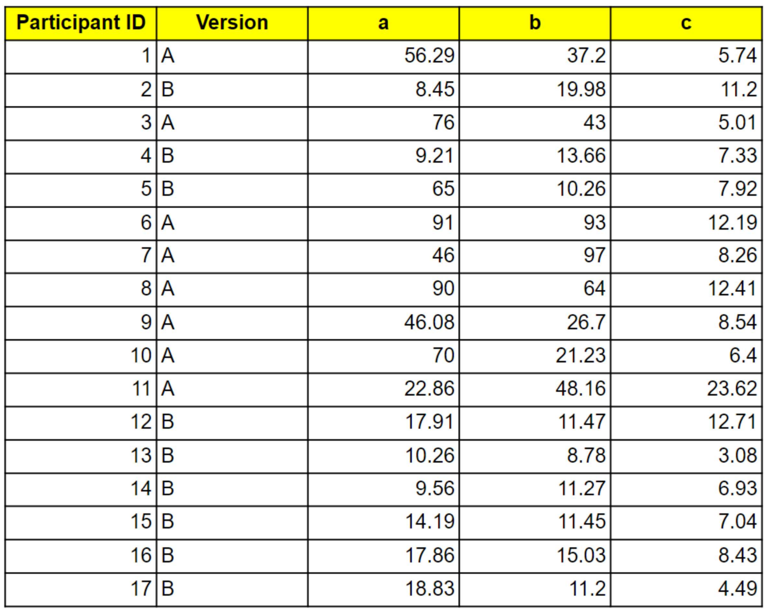
Participant ID	Version	a	b	c	d	e	f	g	h	Total correct	Percentage correct
1	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
2	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
3	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
4	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
5	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
6	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
7	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
8	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
9	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
10	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
11	A	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
12	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
13	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
14	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	7	87.50%
15	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
16	B	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	8	100.00%
17	B	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	7	87.50%

Figure A.4: Task **[T3]**.



Participant ID	Version	Feedback Likert	Intuitiveness Likert
1	A	5	4
2	B	4	5
3	A	4	4
4	B	4	4
5	B	5	5
6	A	5	5
7	A	4	2
8	A	4	3
9	A	5	5
10	A	5	5
11	A	5	5
12	B	5	4
13	B	4	5
14	B	5	5
15	B	5	4
16	B	4	5
17	B	4	4

Figure A.5: Task **[T4]** and **[T5]**.



Participant ID	Version	a	b	c
1	A	56.29	37.2	5.74
2	B	8.45	19.98	11.2
3	A	76	43	5.01
4	B	9.21	13.66	7.33
5	B	65	10.26	7.92
6	A	91	93	12.19
7	A	46	97	8.26
8	A	90	64	12.41
9	A	46.08	26.7	8.54
10	A	70	21.23	6.4
11	A	22.86	48.16	23.62
12	B	17.91	11.47	12.71
13	B	10.26	8.78	3.08
14	B	9.56	11.27	6.93
15	B	14.19	11.45	7.04
16	B	17.86	15.03	8.43
17	B	18.83	11.2	4.49

Figure A.6: Task **[T6]**. Unit is in seconds.

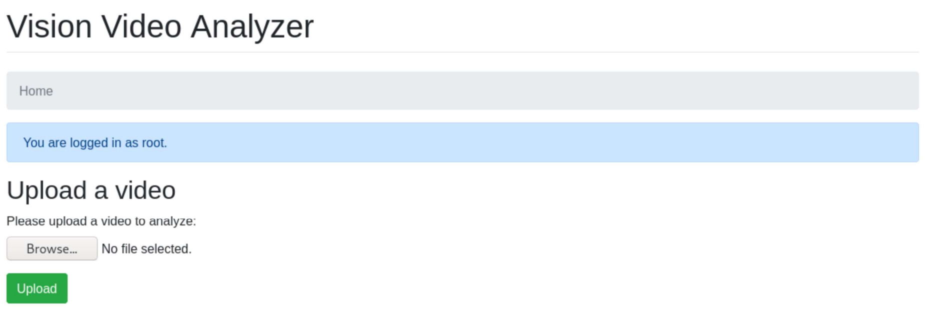
A.4 Common feedback

Count	Feedback
7	Click on card image or video title to open video page
5	Make hints clickable instead of hover only
4	Give more emphasis on the shots analysis button
2	Add delete button to shot and/or video analysis page
2	Add dark mode
2	Improve exposure histogram understandibility

Appendix B

Page Layouts

This section includes the screenshots of pages and popups that are available on the web application.



Vision Video Analyzer

Home

You are logged in as root.

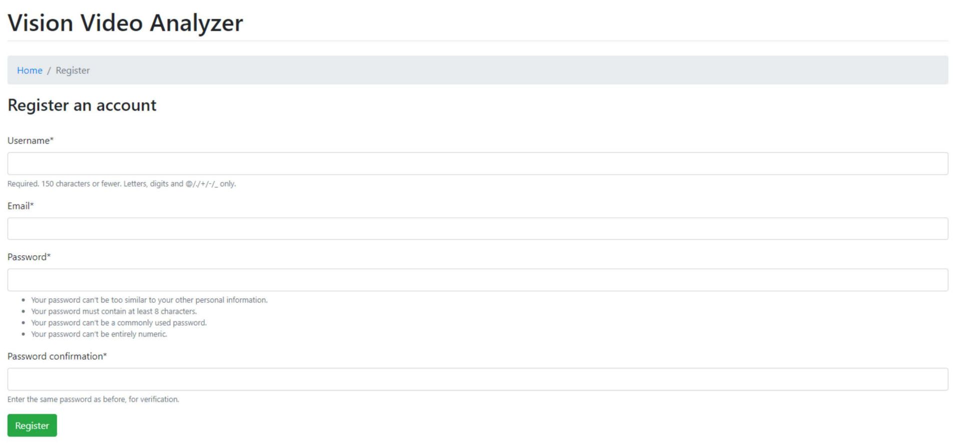
Upload a video

Please upload a video to analyze:

Browse... No file selected.

Upload

Figure B.1: The home page



Vision Video Analyzer

Home / Register

Register an account

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

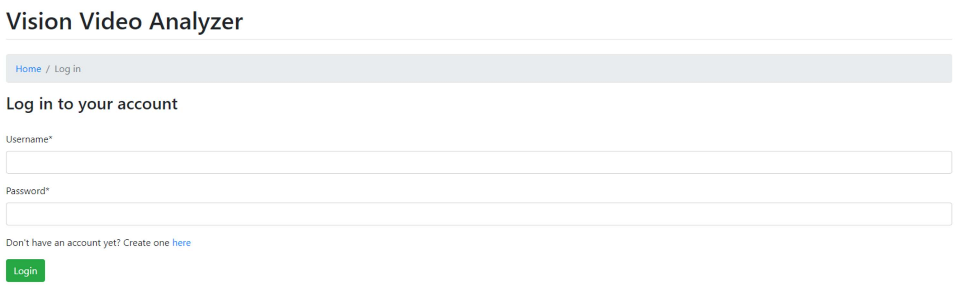
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Register

Figure B.2: The registration page of the web app


Vision Video Analyzer

Home / Log in

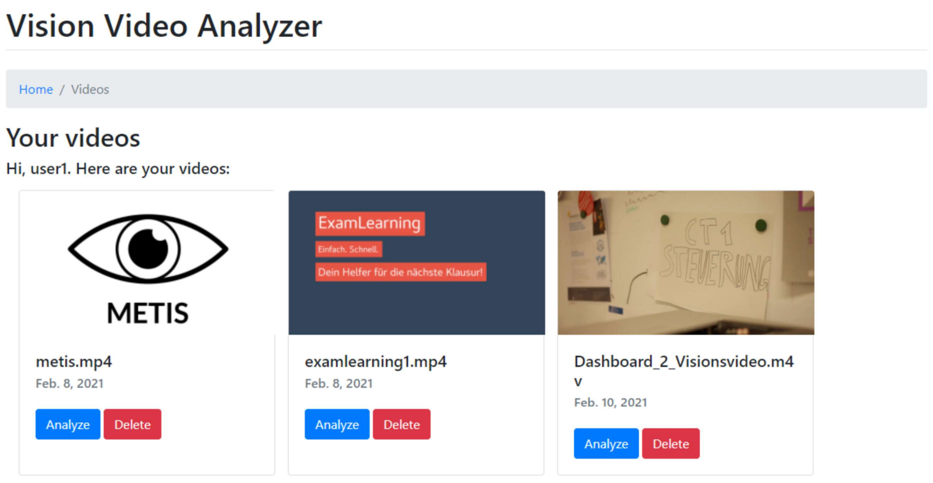
Log in to your account

Username*

Password*

Don't have an account yet? Create one here

Figure B.3: The log in page of the web app


Vision Video Analyzer

Home / Videos

Your videos

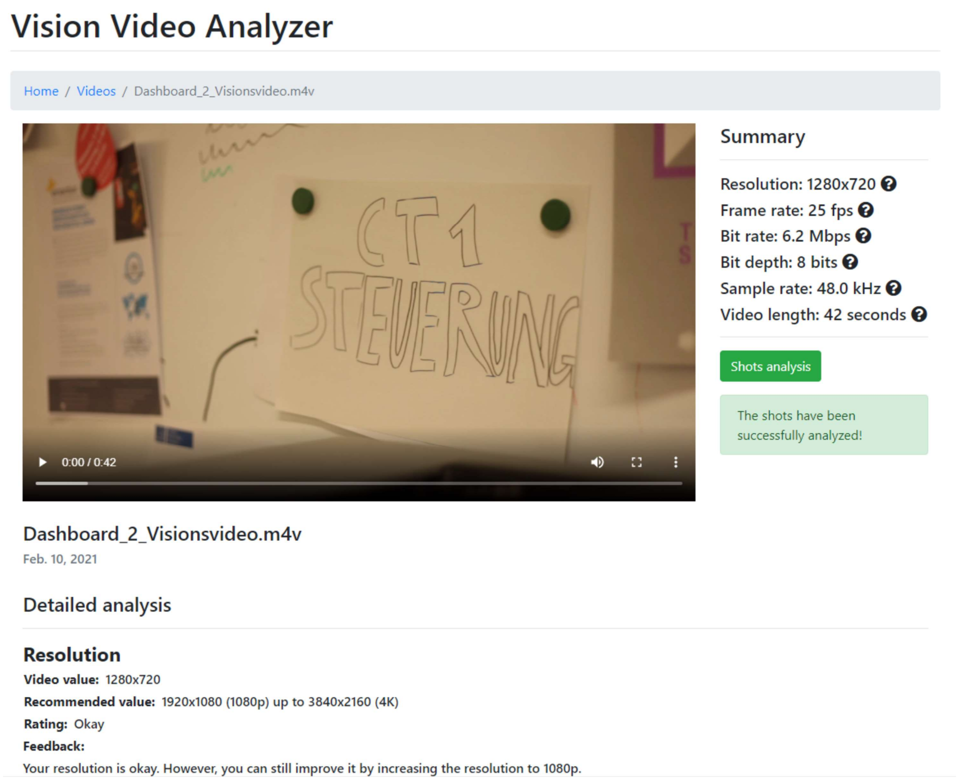
Hi, user1. Here are your videos:

metis.mp4
Feb. 8, 2021

examlearning1.mp4
Feb. 8, 2021

Dashboard_2_Visionsvideo.m4v
Feb. 10, 2021

Figure B.4: The videos page



Vision Video Analyzer

Home / Videos / Dashboard_2_Visionsvideo.m4v

Summary

Resolution: 1280x720

Frame rate: 25 fps

Bit rate: 6.2 Mbps

Bit depth: 8 bits

Sample rate: 48.0 kHz

Video length: 42 seconds

Shots analysis

The shots have been successfully analyzed!

Dashboard_2_Visionsvideo.m4v

Feb. 10, 2021

Detailed analysis

Resolution

Video value: 1280x720

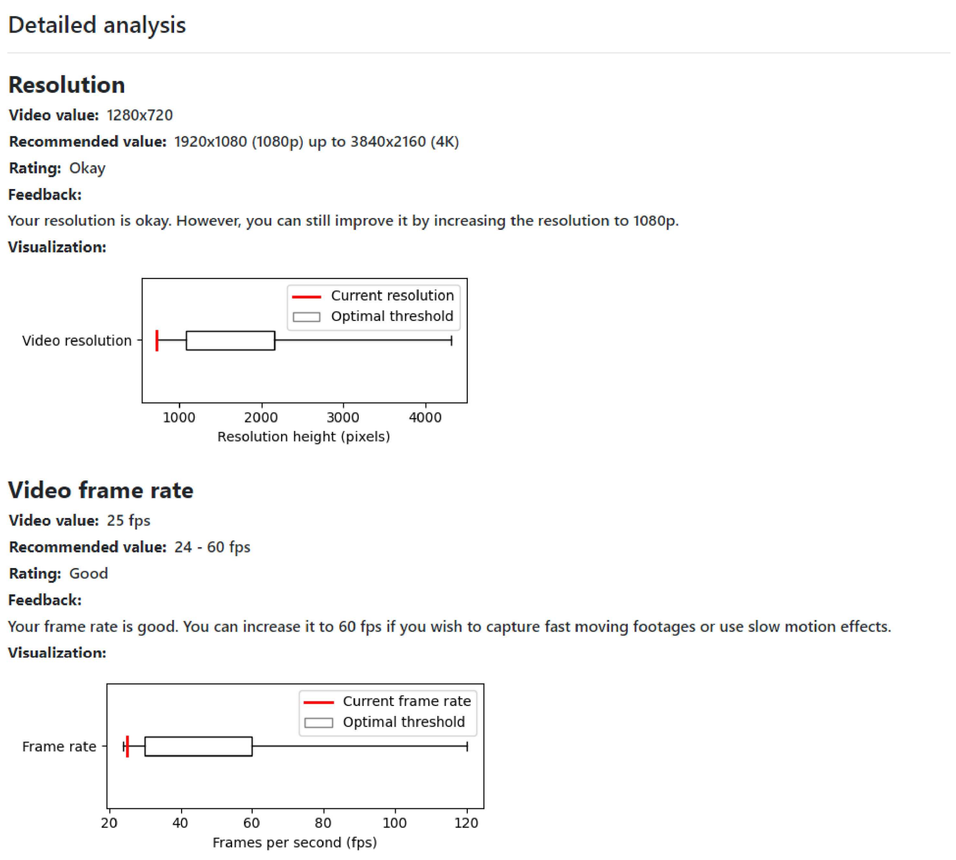
Recommended value: 1920x1080 (1080p) up to 3840x2160 (4K)

Rating: Okay

Feedback:

Your resolution is okay. However, you can still improve it by increasing the resolution to 1080p.

Figure B.5: The video analysis page



Detailed analysis

Resolution

Video value: 1280x720

Recommended value: 1920x1080 (1080p) up to 3840x2160 (4K)

Rating: Okay

Feedback:

Your resolution is okay. However, you can still improve it by increasing the resolution to 1080p.

Visualization:

Video frame rate

Video value: 25 fps

Recommended value: 24 - 60 fps

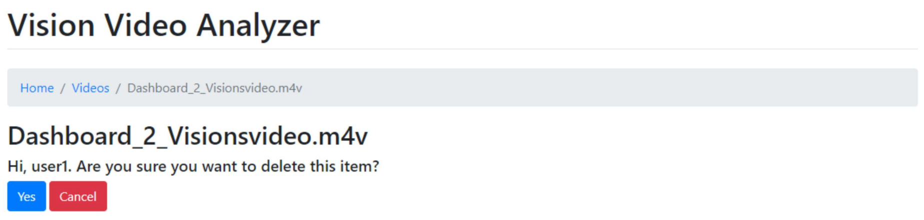
Rating: Good

Feedback:

Your frame rate is good. You can increase it to 60 fps if you wish to capture fast moving footages or use slow motion effects.

Visualization:

Figure B.6: Detailed analysis section of video characteristics



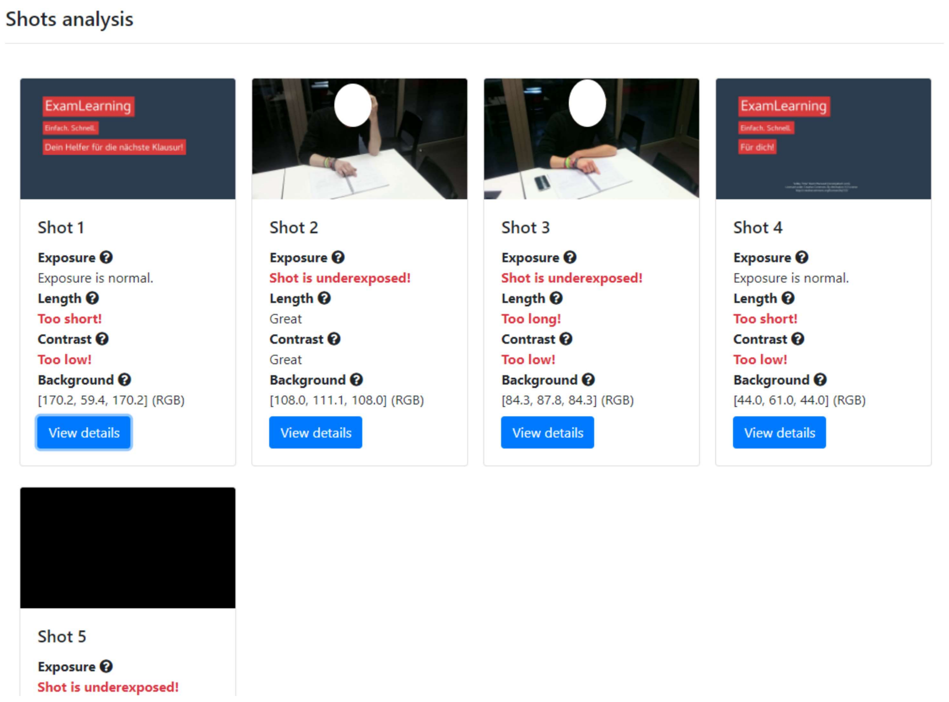
Vision Video Analyzer

Home / Videos / Dashboard_2_Visionsvideo.m4v

Dashboard_2_Visionsvideo.m4v

Hi, user1. Are you sure you want to delete this item?

Figure B.7: The delete confirmation page



Shots analysis

Shot 1
Exposure
Exposure is normal.
Length
Too short!
Contrast
Too low!
Background
[170.2, 59.4, 170.2] (RGB)
View details

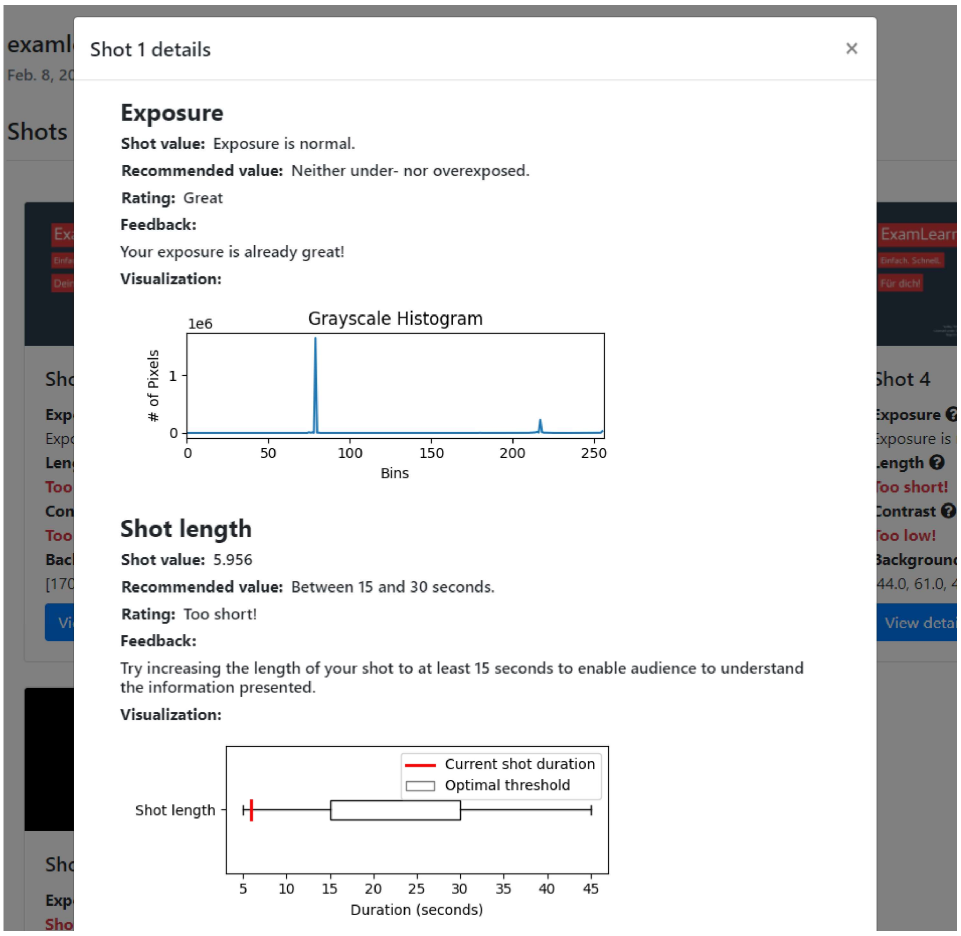
Shot 2
Exposure
Shot is underexposed!
Length
Great
Contrast
Great
Background
[108.0, 111.1, 108.0] (RGB)
View details

Shot 3
Exposure
Shot is underexposed!
Length
Too long!
Contrast
Too low!
Background
[84.3, 87.8, 84.3] (RGB)
View details

Shot 4
Exposure
Exposure is normal.
Length
Too short!
Contrast
Too low!
Background
[44.0, 61.0, 44.0] (RGB)
View details

Shot 5
Exposure
Shot is underexposed!

Figure B.8: The shots analysis page



Shot 1 details ×

Exposure

Shot value: Exposure is normal.

Recommended value: Neither under- nor overexposed.

Rating: Great

Feedback:

Your exposure is already great!

Visualization:

Shot length

Shot value: 5.956

Recommended value: Between 15 and 30 seconds.

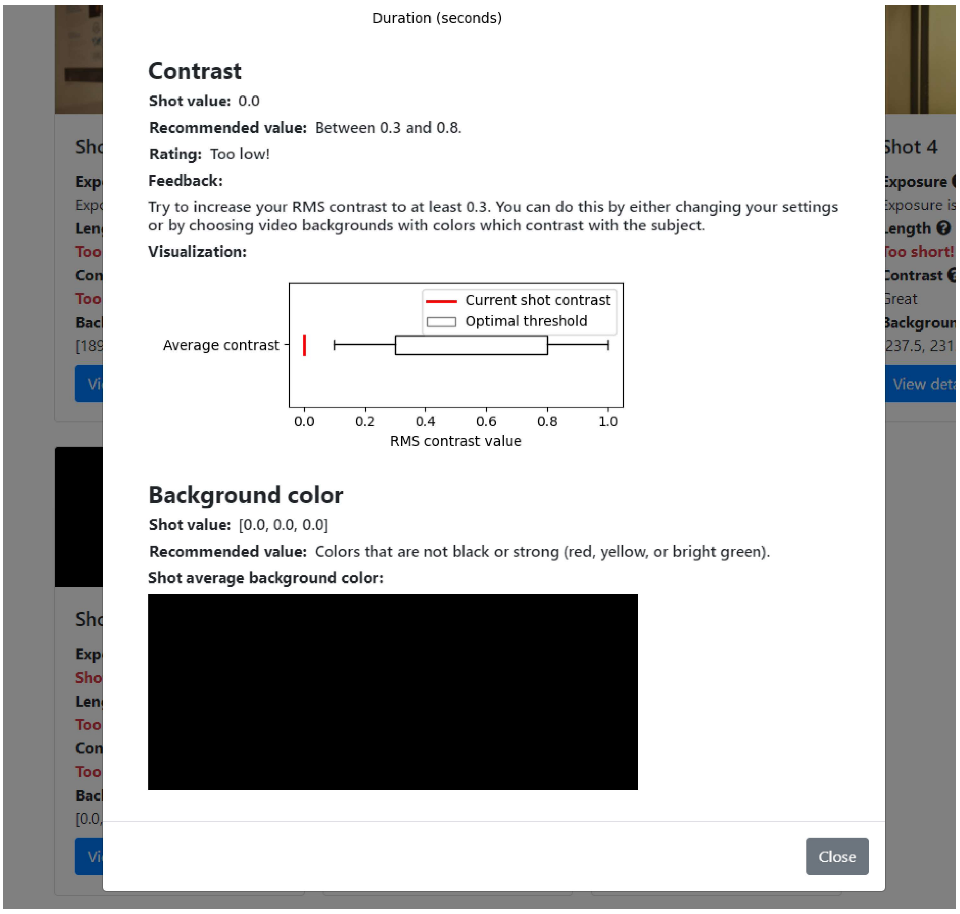
Rating: Too short!

Feedback:

Try increasing the length of your shot to at least 15 seconds to enable audience to understand the information presented.

Visualization:

Figure B.9: Shot details popup



Contrast

Shot value: 0.0

Recommended value: Between 0.3 and 0.8.

Rating: Too low!

Feedback:

Try to increase your RMS contrast to at least 0.3. You can do this by either changing your settings or by choosing video backgrounds with colors which contrast with the subject.

Visualization:

Background color

Shot value: [0.0, 0.0, 0.0]

Recommended value: Colors that are not black or strong (red, yellow, or bright green).

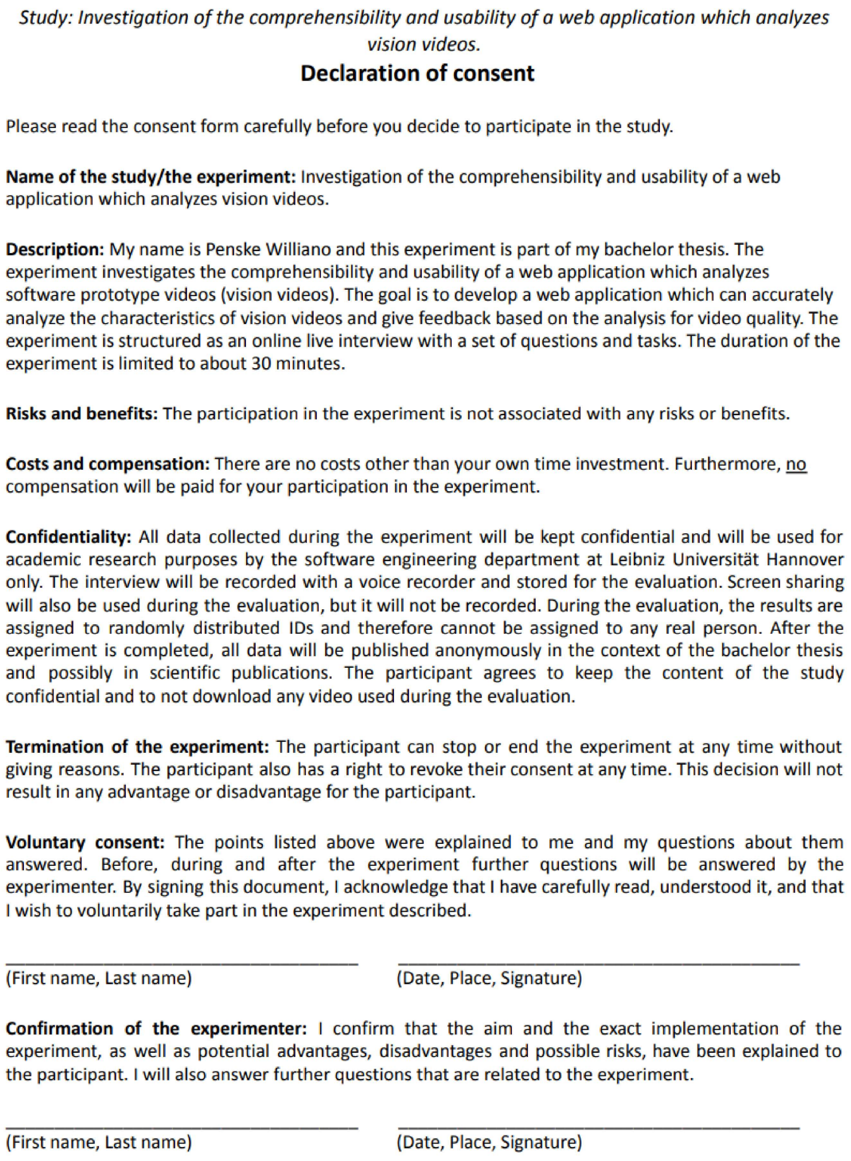
Shot average background color:

Figure B.10: Background color visualization

Appendix C

Relevant Documents

This section includes the consent form template and the detailed experiment protocol.



Study: Investigation of the comprehensibility and usability of a web application which analyzes vision videos.

Declaration of consent

Please read the consent form carefully before you decide to participate in the study.

Name of the study/the experiment: Investigation of the comprehensibility and usability of a web application which analyzes vision videos.

Description: My name is Penske Williano and this experiment is part of my bachelor thesis. The experiment investigates the comprehensibility and usability of a web application which analyzes software prototype videos (vision videos). The goal is to develop a web application which can accurately analyze the characteristics of vision videos and give feedback based on the analysis for video quality. The experiment is structured as an online live interview with a set of questions and tasks. The duration of the experiment is limited to about 30 minutes.

Risks and benefits: The participation in the experiment is not associated with any risks or benefits.

Costs and compensation: There are no costs other than your own time investment. Furthermore, no compensation will be paid for your participation in the experiment.

Confidentiality: All data collected during the experiment will be kept confidential and will be used for academic research purposes by the software engineering department at Leibniz Universität Hannover only. The interview will be recorded with a voice recorder and stored for the evaluation. Screen sharing will also be used during the evaluation, but it will not be recorded. During the evaluation, the results are assigned to randomly distributed IDs and therefore cannot be assigned to any real person. After the experiment is completed, all data will be published anonymously in the context of the bachelor thesis and possibly in scientific publications. The participant agrees to keep the content of the study confidential and to not download any video used during the evaluation.

Termination of the experiment: The participant can stop or end the experiment at any time without giving reasons. The participant also has a right to revoke their consent at any time. This decision will not result in any advantage or disadvantage for the participant.

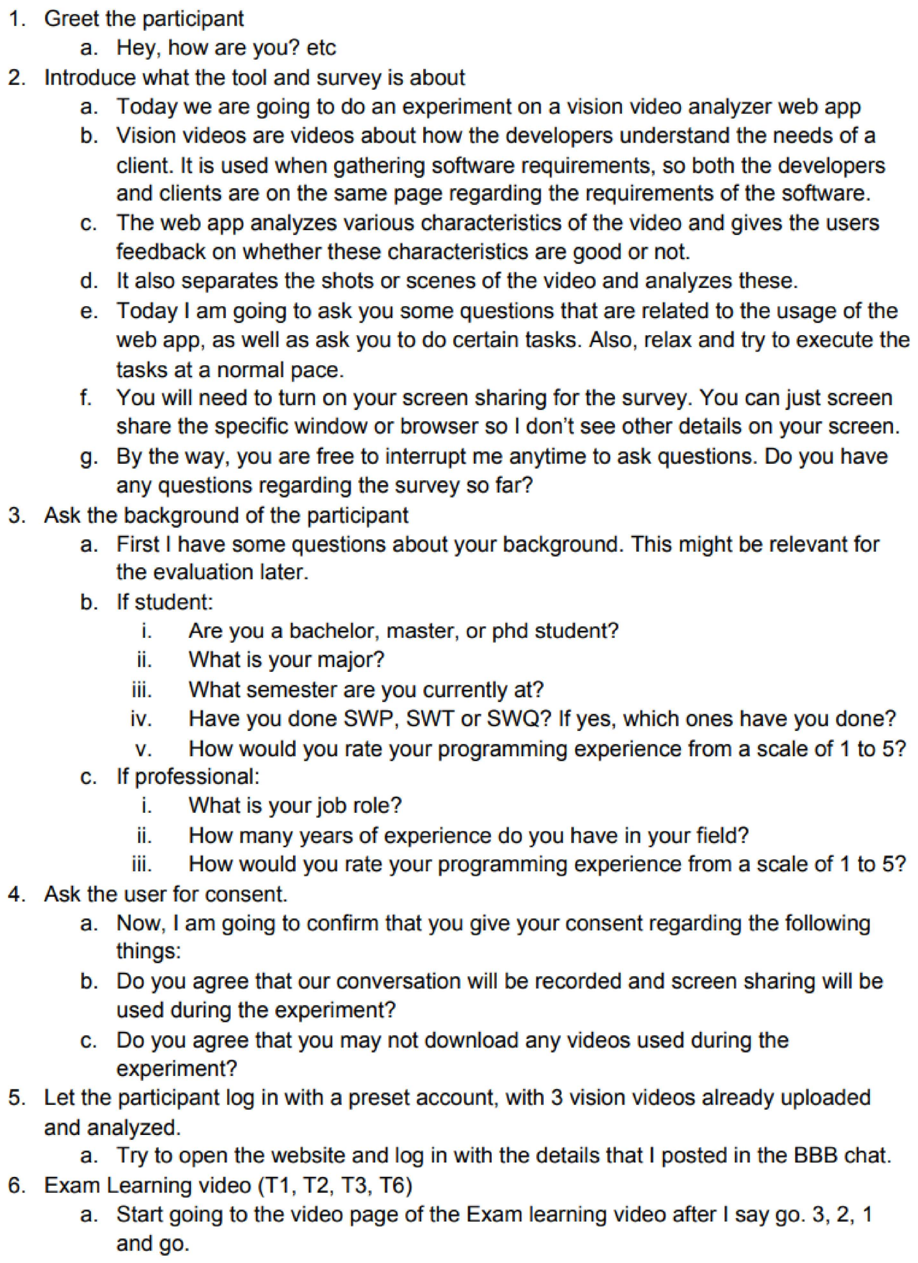
Voluntary consent: The points listed above were explained to me and my questions about them answered. Before, during and after the experiment further questions will be answered by the experimenter. By signing this document, I acknowledge that I have carefully read, understood it, and that I wish to voluntarily take part in the experiment described.

______________________________	______________________________
(First name, Last name)	(Date, Place, Signature)

Confirmation of the experimenter: I confirm that the aim and the exact implementation of the experiment, as well as potential advantages, disadvantages and possible risks, have been explained to the participant. I will also answer further questions that are related to the experiment.

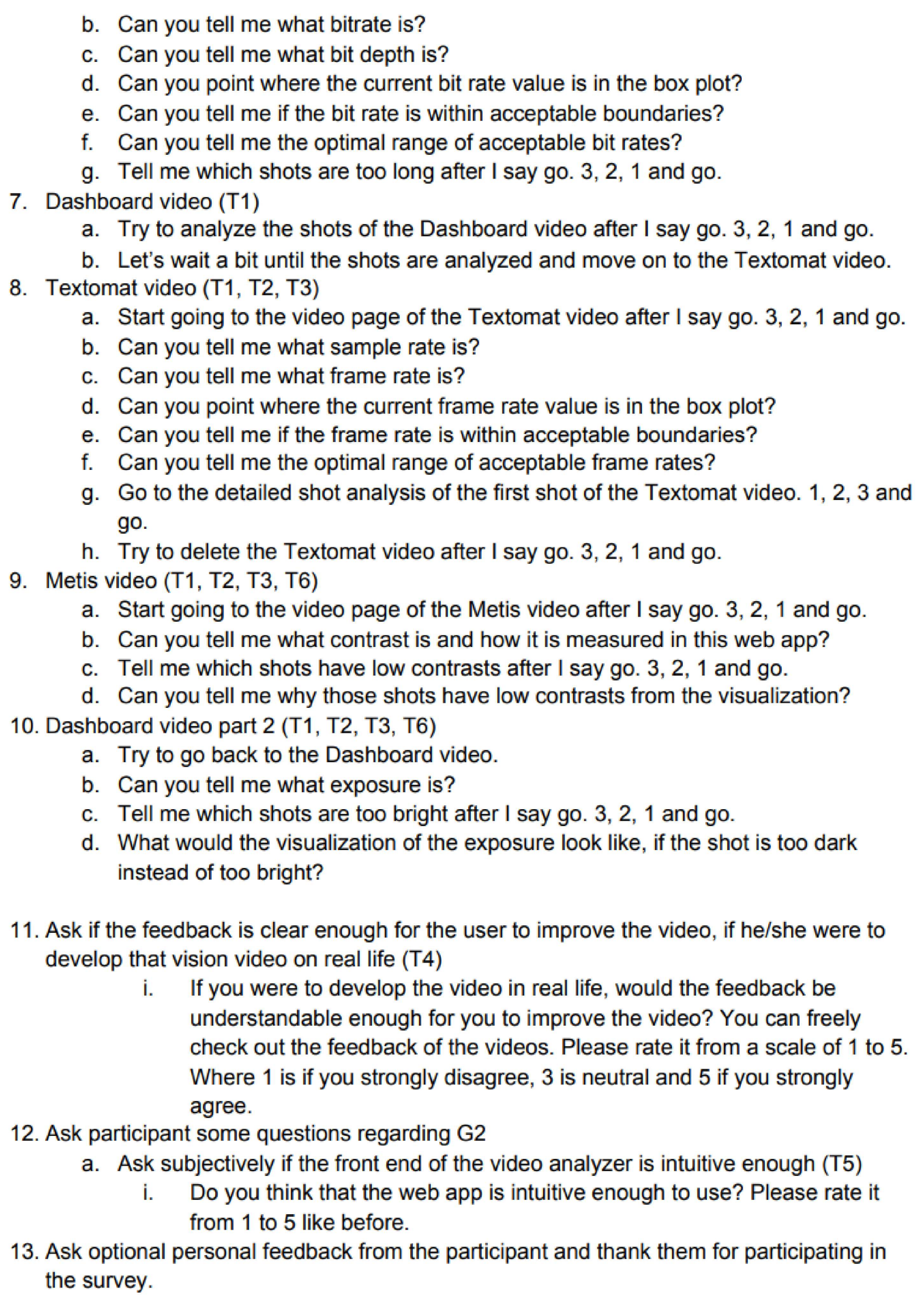
______________________________	______________________________
(First name, Last name)	(Date, Place, Signature)

Figure C.1: The consent form template.



1. Greet the participant
   a. Hey, how are you? etc
2. Introduce what the tool and survey is about
   a. Today we are going to do an experiment on a vision video analyzer web app
   b. Vision videos are videos about how the developers understand the needs of a client. It is used when gathering software requirements, so both the developers and clients are on the same page regarding the requirements of the software.
   c. The web app analyzes various characteristics of the video and gives the users feedback on whether these characteristics are good or not.
   d. It also separates the shots or scenes of the video and analyzes these.
   e. Today I am going to ask you some questions that are related to the usage of the web app, as well as ask you to do certain tasks. Also, relax and try to execute the tasks at a normal pace.
   f. You will need to turn on your screen sharing for the survey. You can just screen share the specific window or browser so I don't see other details on your screen.
   g. By the way, you are free to interrupt me anytime to ask questions. Do you have any questions regarding the survey so far?
3. Ask the background of the participant
   a. First I have some questions about your background. This might be relevant for the evaluation later.
   b. If student:
      i. Are you a bachelor, master, or phd student?
      ii. What is your major?
      iii. What semester are you currently at?
      iv. Have you done SWP, SWT or SWQ? If yes, which ones have you done?
      v. How would you rate your programming experience from a scale of 1 to 5?
   c. If professional:
      i. What is your job role?
      ii. How many years of experience do you have in your field?
      iii. How would you rate your programming experience from a scale of 1 to 5?
4. Ask the user for consent.
   a. Now, I am going to confirm that you give your consent regarding the following things:
   b. Do you agree that our conversation will be recorded and screen sharing will be used during the experiment?
   c. Do you agree that you may not download any videos used during the experiment?
5. Let the participant log in with a preset account, with 3 vision videos already uploaded and analyzed.
   a. Try to open the website and log in with the details that I posted in the BBB chat.
6. Exam Learning video (T1, T2, T3, T6)
   a. Start going to the video page of the Exam learning video after I say go. 3, 2, 1 and go.

Figure C.2: Experiment procedure (page 1).



b. Can you tell me what bitrate is?
c. Can you tell me what bit depth is?
d. Can you point where the current bit rate value is in the box plot?
e. Can you tell me if the bit rate is within acceptable boundaries?
f. Can you tell me the optimal range of acceptable bit rates?
g. Tell me which shots are too long after I say go. 3, 2, 1 and go.

7. Dashboard video (T1)
   a. Try to analyze the shots of the Dashboard video after I say go. 3, 2, 1 and go.
   b. Let's wait a bit until the shots are analyzed and move on to the Textomat video.
8. Textomat video (T1, T2, T3)
   a. Start going to the video page of the Textomat video after I say go. 3, 2, 1 and go.
   b. Can you tell me what sample rate is?
   c. Can you tell me what frame rate is?
   d. Can you point where the current frame rate value is in the box plot?
   e. Can you tell me if the frame rate is within acceptable boundaries?
   f. Can you tell me the optimal range of acceptable frame rates?
   g. Go to the detailed shot analysis of the first shot of the Textomat video. 1, 2, 3 and go.
   h. Try to delete the Textomat video after I say go. 3, 2, 1 and go.
9. Metis video (T1, T2, T3, T6)
   a. Start going to the video page of the Metis video after I say go. 3, 2, 1 and go.
   b. Can you tell me what contrast is and how it is measured in this web app?
   c. Tell me which shots have low contrasts after I say go. 3, 2, 1 and go.
   d. Can you tell me why those shots have low contrasts from the visualization?
10. Dashboard video part 2 (T1, T2, T3, T6)
    a. Try to go back to the Dashboard video.
    b. Can you tell me what exposure is?
    c. Tell me which shots are too bright after I say go. 3, 2, 1 and go.
    d. What would the visualization of the exposure look like, if the shot is too dark instead of too bright?

11. Ask if the feedback is clear enough for the user to improve the video, if he/she were to develop that vision video on real life (T4)
    i. If you were to develop the video in real life, would the feedback be understandable enough for you to improve the video? You can freely check out the feedback of the videos. Please rate it from a scale of 1 to 5. Where 1 is if you strongly disagree, 3 is neutral and 5 if you strongly agree.
12. Ask participant some questions regarding G2
    a. Ask subjectively if the front end of the video analyzer is intuitive enough (T5)
       i. Do you think that the web app is intuitive enough to use? Please rate it from 1 to 5 like before.
13. Ask optional personal feedback from the participant and thank them for participating in the survey.

Figure C.3: Experiment procedure (page 2).

Appendix D

Contents of the CD

A CD is attached to this bachelor thesis, which consists of the following content:

- The bachelor thesis in digital form (.pdf)
- Three versions of the source code:
  - Version A and version B of the web app that were used for the evaluation
  - Most recent version of the web app (as of 09.03.2021)
- Evaluation data:
  - Consent forms
  - Experiment protocol
  - Experiment results as Excel spreadsheet

Bibliography

[1] J. J. Amor, G. Robles, J. M. González-Barahona, and I. Herraiz. From pigs to stripes: A travel through debian. In *Proceedings of the DebConf5 (Debian Annual Developers Meeting)*. Citeseer, 2005.

[2] M. Azimi, R. Boitard, P. Nasiopoulos, and M. T. Pourazad. Visual color difference evaluation of standard color pixel representations for high dynamic range video compression. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1480–1484, 2017.

[3] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532, 1994.

[4] E. Börger, B. Hörger, D. Parnas, and H. Rombach. Requirements capture, documentation, and validation. In *Dagstuhl Seminar, number 99241*. Citeseer, 1999.

[5] O. Brill, K. Schneider, and E. Knauss. Videos vs. use cases: Can videos capture more requirements under time pressure? In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 30–44. Springer, 2010.

[6] BT.2100-2. Image parameter values for high dynamic range television for use in production and international programme exchange. Standard, International Telecommunication Union, Geneva, CH, July 2018.

[7] B. Castellano. Scene detection algorithms - pyscenedetect. `https://pyscenedetect.readthedocs.io/en/latest/reference/detection-methods/#threshold-detector`. (Accessed on 01/25/2021).

[8] O. Creighton, M. Ott, and B. Bruegge. Software cinema-video-based requirements engineering. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 109–118. IEEE, 2006.

[9] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on

user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.

[10] J. Donelan. The state of 8k. *Information Display*, 35(1):17–19, 2019.

[11] D. S. Foundation. Django documentation. `https://docs.djangoproject.com/en/3.1/topics/db/models/`. (Accessed on 02/07/2021).

[12] A. Gibson. *Exposure and Understanding the Histogram.* Peachpit Press, 2011.

[13] Google, LLC. Recommended upload encoding settings. `https://support.google.com/youtube/answer/1722171?hl=en`. (Accessed on 12/01/2020).

[14] P. J. Guo, J. Kim, and R. Rubin. How video production affects student engagement: An empirical study of MOOC videos. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 41–50, 2014.

[15] J. Happe. Automating guidelines for video production in requirements engineering. `https://www.pi.uni-hannover.de/fileadmin/pi/se/Stud-Arbeiten/2018/Happe2018.pdf`, 2018. (Accessed on 12/02/2020).

[16] J. Joskowicz, R. Sotelo, M. Juayek, D. Durán, and J. P. Garella. Automation of subjective video quality measurements. In *Proceedings of the Latin America Networking Conference on LANC 2014*, LANC '14, New York, NY, USA, 2014.

[17] O. Karras. Software professionals' attitudes towards video as a medium in requirements engineering. In *International Conference on Product-Focused Software Process Improvement*, pages 150–158. Springer, 2018.

[18] O. Karras and K. Schneider. Software professionals are not directors: What constitutes a good video? In *2018 1st International Workshop on Learning from other Disciplines for Requirements Engineering (D4RE)*, pages 18–21. IEEE, 2018.

[19] O. Karras and K. Schneider. An interdisciplinary guideline for the production of videos and vision videos by software professionals. *arXiv preprint arXiv:2001.06675*, 2020.

[20] O. Karras, K. Schneider, and S. A. Fricker. Representing software project vision by means of video: A quality model for vision videos. *Journal of Systems and Software*, 162:110479, 2020.

[21] H. Kukkonen, J. Rovamo, K. Tiippana, and R. Näsänen. Michelson contrast, rms contrast and energy of various spatial stimuli at threshold. *Vision research*, 33:1431–6, 08 1993.

[22] M. Kurniawan and H. Hara. A beginner's guide to frame rates in movies. `https://www.adobe.com/creativecloud/video/discover/frame-rate.html`. (Accessed on 02/16/2021).

[23] E. Lackner, M. Kopp, and M. Ebner. How to MOOC? – a pedagogical guideline for practitioners. In *Proceedings of the 10th International Scientific Conference "eLearning and Software for Education"*, pages 1–7. Editura Universitatii Nationale de Aparare "Carol I", 2014.

[24] Massachusetts Institute of Technology. Video production guide. `http://web.mit.edu/techtv/videoprodguide/videoprodguide.pdf`. (Accessed on 12/02/2020).

[25] B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46, 2000.

[26] D. Parker. Background colour detection using opencv and python. `https://medium.com/generalist-dev/background-colour-detection-using-opencv-and-python-22ed8655b243`. (Accessed on 02/15/2021).

[27] J. M. Pearson and A. M. Pearson. An exploratory study into determining the relative importance of key criteria in web usability: A multi-criteria approach. *Journal of Computer Information Systems*, 48(4):115–127, 2008.

[28] E. Peli. Contrast in complex images. *Journal of the Optical Society of America A*, 7(10):2032–2040, 1990.

[29] A. Pras and C. Guastavino. Sampling rate discrimination: 44.1 kHz vs. 88.2 kHz. *Journal of the Audio Engineering Society*, May 2010.

[30] K. Rockwell. How to use histograms. `https://www.kenrockwell.com/tech/histograms.htm`. (Accessed on 02/09/2021).

[31] V. A. Sankaranarayanan. Tool-supported data collection for experiments to subjectively assess vision videos. `https://www.pi.uni-hannover.de/fileadmin/pi/se/Stud-Arbeiten/2019/Arulmani2019.pdf`, 2019. (Accessed on 12/02/2020).

[32] K. Schneider. Lecture 3: Anforderungen wirksam berücksichtigen. In *Grundlagen der Software-Technik*. Leibniz Universität Hannover, 2018.

[33] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[34] SQLite. Appropriate uses for SQLite. `https://www.sqlite.org/whentouse.html`. (Accessed on 12/02/2020).

[35] A. Tekian and J. Norcini. Overcome the 60% passing score and improve the quality of assessment. *GMS Zeitschrift für Medizinische Ausbildung*, 32(4), 2015.

[36] S. Weech, S. Kenny, C. M. Calderon, and M. Barnett-Cowan. Limits of subjective and objective vection for ultra-high frame rate visual displays. *Displays*, 64:101961, 2020.

[37] M. Winken, D. Marpe, H. Schwarz, and T. Wiegand. Bit-depth scalable video coding. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I – 5–I – 8, 2007.

[38] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[39] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 461–470, 2015.

List of Figures

Acknowledgements

I wish to express my sincerest gratitude to the following persons who have contributed much to making this thesis possible.

M. Sc. Jianwei Shi, thesis supervisor, for always giving his valuable suggestions and encouragement to improve the thesis.

Prof. Dr. rer. nat. Kurt Schneider and Dr. rer. nat. Jil Ann-Christin Klünder, thesis examiners, for their valuable critiques and grading of the thesis.

M. Sc. Melanie Busch, for her valuable advice on the organisation and improvement of the thesis

The participants who took part in the survey, for their time, willingness to take part, and valuable advice feedback.

The groups who made the vision videos for their software projects, for their invaluable vision video samples.

My friends, family, and girlfriend for supporting me throughout this journey.