Ola Henrik Otterlei Navelsaker
Sveinung Olsen
Andreas Haug Sellesbakk

Barco - Projector Calculator for Residential Use

Bachelor's thesis in Computer Science
Supervisor: Giorgio Trumpy
Co-supervisor: Kiran Raja
May 2023

Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

BARCO

Ola Henrik Otterlei Navelsaker
Sveinung Olsen
Andreas Haug Sellesbakk

Barco - Projector Calculator for Residential Use

Bachelor's thesis in Computer Science
Supervisor: Giorgio Trumpy
Co-supervisor: Kiran Raja
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

The project aims to create a user-friendly, easily expandable tool that customers can use to receive product recommendations based on screen dimensions and screen brightness for residential purposes. The product is targeted for internal and external use for Barco and its customers to help them choose the best solution for their specific setup. This thesis develops a solution according to the needs of Barco with considerations to modular, expandable and robust backend driven solution that is both easy to maintain and extend the functionality in the future. To accomplish this, the project has considered various aspects of backend development and user interface development. The project also involves tool development to enable data migration from historic files (e.g., an Excel spreadsheet) into a database. The proposed solution offers a user interface, an administrator tool to manipulate the database, and a tool to migrate product information from Excel into a database. The application has been tested from both a software development perspective and a usability standpoint.

Sammendrag

Prosjektet har som mål å utvikle et brukervennlig, lett utvidbart verktøy som kunder kan bruke for å motta produkt anbefalinger basert på skjermdimensjoner og skjermlysstyrke for boligformål. Produktet er målrettet for intern og ekstern bruk for Barco og deres kunder for å hjelpe dem med å velge den beste løsningen for deres spesifikke oppsett. Denne oppgaven utvikler en løsning i henhold til Barcos behov med hensyn til modulær, utvidbar og robust backend-drevet løsning som både er enkel å vedlikeholde og utvide funksjonaliteten i fremtiden. For å oppnå dette har prosjektet vurdert ulike aspekter ved utvikling av backend og utvikling av brukergrensesnitt. Prosjektet involverer også verktøy utvikling for å muliggjøre data migrering fra historiske filer (f.eks. et Excel-regneark) til en database. Den foreslåtte løsningen tilbyr et brukergrensesnitt, et administrator verktøy for å manipulere databasen, og et verktøy for å migrere produktinformasjon fra Excel til en database. Applikasjonen har blitt testet både fra et programvareutvikling perspektiv og et brukervennlighet synspunkt.

PREFACE

We would first like to thank our main supervisor Giorgio Trumpy for the guidance and help he has provided. We would also like to thank Kiran Raja, who has been our secondary supervisor and have provided the group with guidance and technical support.

Lastly, the group would like to thank our task provider, Barco, representatives Thomas Andersen, Andy Jones, and Victor Steen. They gave us the opportunity to work on this exciting project and have been available to answer any question at a short notice throughout the process.

Acronyms

API Application Programming Interface. 11, 12, 25, 31, 35, 37, 48, 51, 54, 58

DOM Document Object Model. 9, 10

HTML HyperText Markup Language. 8, 9

JSON JavaScript Object Notation. 25, 30, 32, 35, 48

JWT JSON Web Token. 30, 36, 37

LINQ Language-Integrated Query. 24

NTNU Norwegian University of Science and Technology. 55

PWA Progressive Web Application. 11

QA Quality Assurance. 17

REST Representational State Transfer. 51, 54

SQL Structured Query Language. viii, 1, 12–14, 24, 25, 42, 48

URL Uniform Resource Locator. 32, 51, 53

Table of Contents

List of Figures

List of Tables

1 Introduction

In the domain of projectors, Barco designs and manufactures a wide range of digital projectors for different industries and applications. Their projectors are known for their brightness, color accuracy, and image quality. They offer projectors for various environments such as conference rooms, education, live events, and large venues such as movie theaters and concert halls. They also offer projectors with specialized features such as 3D projection and ultra-short throw projection. Barco's projectors are equipped with advanced image processing technology and support multiple connectivity options, making them suitable for a wide range of applications. They support the use of their projector by providing a range of services and solutions, such as maintenance services and control systems. Barco has a wide range of projectors and lenses that need to be selected based on many parameters. These parameters include, but are not limited to: room dimensions, screen dimensions, lens shift, throw ratio, focus range, output resolution, and contrast ratio. This requires complicated calculations to get the best result based on the specific scenario.

1.1 Target Audience

The project has a specific target audience in mind, consisting of two distinct groups. The first group is Barco's own employees. These employees can use our application to increase efficiency when working, by having several tools available in one application. The second group are Barco's customers that want to explore the home cinema market and easily calculate what projectors and lenses would work best for their specific setup.

1.2 Delimitations

This project was given by Barco. Their project requirements were well-defined in the project description sent to us, and because of this, very few delimitations were set by us. Barco did not expect us to implement all the features and calculators they wanted for the full application in the time frame of this project. Therefore, it was our intention to implement the different components of the project according to the requirement priority section, which is described in Section 2.1.6.

1.3 Constraints

Constraints are important to keep in mind both during planning, and during execution. Constraints aid in shaping the problem area, and helps by providing a general overview of the problem area, and therefore it is important to have a good grasp of the project's constraints.

1.3.1 Time Constraints

This thesis project started in early January, and ended in May, this means we had about five months to develop our application. The thesis project has a clearly defined delivery date of 22.05.2023, which marks the deadline for its submission.

1.3.2 Software Constraints

For the code, Barco requested that we use C # and Blazor for our application, and Microsoft SQL Server for the database. The reason for this was that they wanted us to use a language and database they are familiar with, so that they are able to modify or continue developing and improving the application.

Blazor supports modern browsers such as Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, but old versions of these browsers, or different browsers may not be supported.

1.4 Goals

For a project of this size, it is important to set certain goals that we aim to achieve. The project goals are a good tool to provide direction and an overview on what we need to accomplish, and effect goals are useful to evaluate the success of the project.

1.4.1 Project Goals

- Reduce installation challenges for both Barco engineers and customers.
- Create better workflow for the technical support by helping customers choose projector and lens combination based on defined parameters.
- Combine functionality from other solutions into one application to improve usability and workflow.
- Design a software architecture that is scalable and flexible, allowing for future development and improvement of the projector calculator's capabilities.
- Implement a user-friendly graphical interface for the calculator that is easy to navigate and intuitive to use.
- Test the software thoroughly to identify and resolve any bugs or issues, ensuring that it is reliable and robust.

1.4.2 Effect Goals

- Improve convenience for adding new products and highlighting products nearing their end of life (EOL).
- Gain experience in developing a full-stack application.
- Enhance our critical thinking and problem-solving skills by addressing technical challenges and resolving software issues.
- Build a strong foundation for future software development projects by developing a deep understanding of software design and architecture, programming best practices, and testing and debugging techniques.

1.4.3 Sustainability Goals

By making it easier to choose the best product for the situation, the cases of customers purchasing the wrong product is reduced. This leads to less shipping, as customers would have a lower likelihood of wanting to return the product. Satisfaction among customers would increase since they are now more likely to purchase the product that best suits their environment and receiving better performance from the projectors on average, which contributes to positive growth for the company.

- Reduce cases of buying the wrong product
- Create a modular software that allows for reuse of code
- Provide well-documented code and follow best practices to support future development

1.5 Contribution

We have developed a fully working application, and a foundation which supports further development. The application serves to aid customers of Barco in choosing the correct projector and lens combination for their specific situation. The development of the application is expected to continue after this thesis, and this has been facilitated and planned for throughout the thesis by following best practices for coding and providing documentation and comments to explain the thought process. Barco has been involved during the development by suggesting functionality to implement, providing sample data for our database, and giving feedback on the aesthetics of the application. This application will continue to be developed by Barco, as not all the functionalities they would want for their customers are implemented.

A database has been created to store information about all the projectors and lenses, and the compatibility between them. Barco had previously been storing this information in a large Excel sheet, which is hard to pull data from. This database is fully functional and could be used as is for other projects at a later stage.

1.6 Thesis Structure

This thesis will be divided into several main sections. Each of the main sections also have several subsections, where we will go in depth on the specific topics related to the main section's subject matter. These subsections allow for a more detailed exploration and analysis of the concepts and ideas introduced in the main section, providing a comprehensive understanding of the subject as a whole.

- Chapter 2 - Requirements, Design and Development process, will explain the requirements laid out for the project, the design choices that were made, and how the application was developed.
- Chapter 3 - Implementation and Deployment , will explain the specific implementation of different features, and they were chosen. How the application was deployed will also be discussed here.
- Chapter 4 - Testing And User Feedback, will explain all the types of testing we performed, and why we used them. This includes both code testing, and user testing.
- Chapter 5 - Discussion, will discuss the choices we made, and what we could have done differently.
- Chapter 6 - Conclusion, will discuss what we think went well in the project, what goals we reached, and our overall satisfaction level after completing this project.

2 Requirements, Design and Development process

The Requirements, Design and Development process chapter aims to provide a comprehensive overview of the requirements of the project, the design choices that were made, and the development process itself.

2.1 Project Requirements

The project goal is to create a secure, robust and scalable application where it is easy to implement new tools and calculators.

These calculators aim to facilitate installation challenges, for both Barco engineers and for Barco's customers. Examples of these challenges could be: what is the best projector for this installation, how much light will there be on the screen, where can the projector be placed, will it be able to fill the screen, what lens should be used etc.

The project consists of several parts that work together to make the application usable, as shown in Figure 1. These parts are the database, the admin tool, and the projector calculator. Even though we can divide them into different parts, it does not mean they can work independently. The calculator is fully dependent on the database to do its calculations, and the admin tool is used to manage the database.

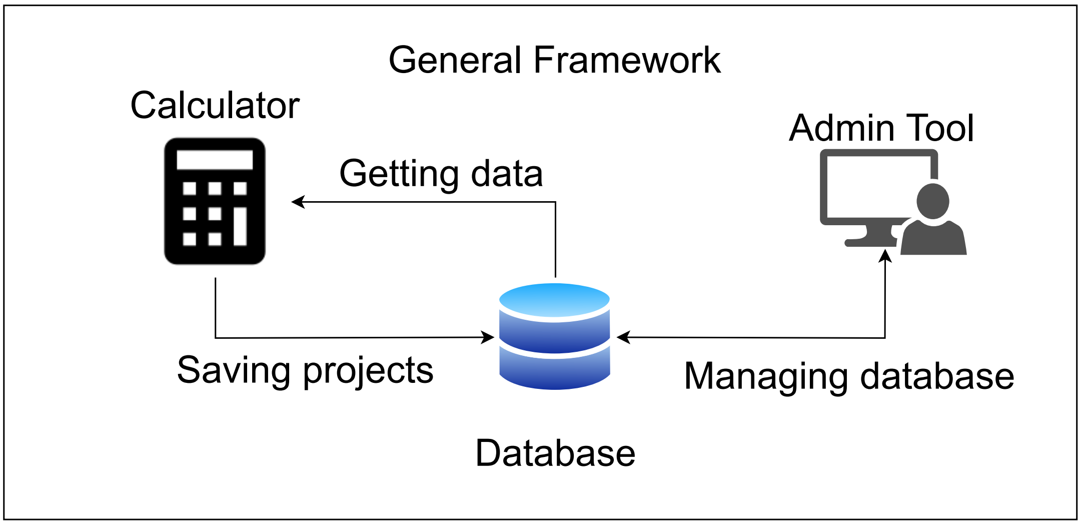


Figure 1: Illustration of how the different parts of the project interact.

2.1.1 General Framework

The application should have an eye pleasing graphical user interface that follows the style of Barco's other application. It should also give the user the ability to register a new user and login, and this should be done securely. The application should also be made in a way that makes it easy to expand and add more calculators at a later point.

2.1.2 Database

The database is used to hold information about all the different projectors and lenses Barco provides. This information consists of many parameters such as dimensions, output lumen, minimum and maximum throw ratio, and more. These parameters will later be used in the calculators to select the best projector and lens combination for the specific setup. The database will also be used to store users, their login credentials and their permissions.

2.1.3 Admin Tool

The admin tool is a feature of the main application that is only accessible to users with the admin role. It allows administrators to add, remove, and edit projectors and lenses in the database. Additionally, the tool can be used to manage user credentials and permissions.

2.1.4 Projector Calculators

The application should include several calculators, and the user should have the opportunity to save and export their projects. The separate calculators are as follows:

2.1.4.1 Find Correct Projector

This calculator should be able to find the best projector and lens combination based on the distance to the screen, screen size, and target luminance.

2.1.4.2 Simple Project Calculator

For this calculator the user will input the projector model, one of the screens dimensions, screen aspect ratio, throw distance, and screen type or customized screen with gain. If you are using a mirror and a port window, include that in the input as well. The calculator will provide the necessary details such as the remaining screen dimensions, recommended lens, maximum shift available, f-number, alternative lens options (if any), and luminance values in either nits, foot-Lambert, or both. (Luminance is the measure of how much light there is, and nits and foot-Lambert are some commonly used units.) The old version of this calculator, made in Excel, is illustrated in Figure 2.

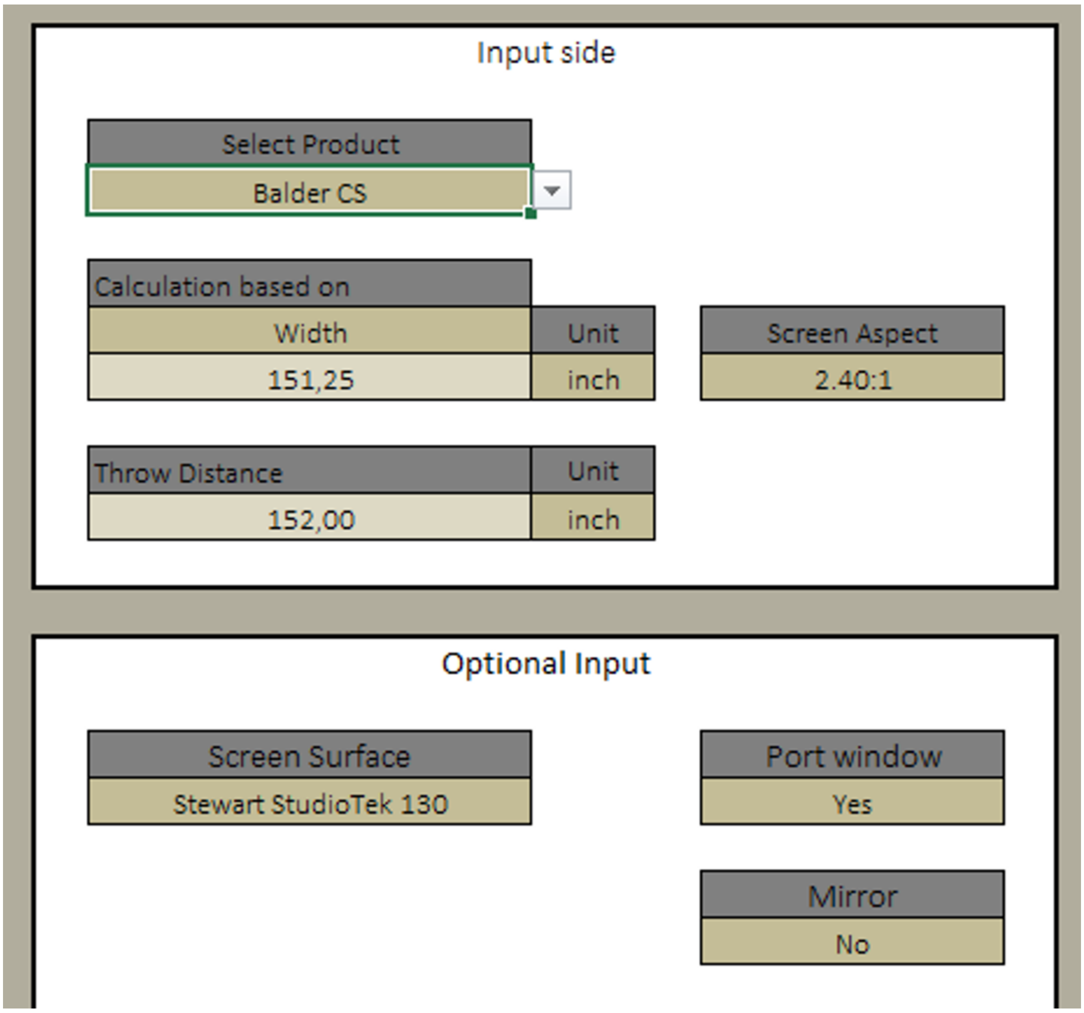
2.1.4.3 Lens Shift Calculator

For this calculator the user will input the projector model, screen dimensions, screen aspect ratio, throw distance and the distance from the center of the lens to the center of the screen. Then the calculator will return if the specific installation can be used, and how much lens-shift will be available. The old version, made in Excel, is illustrated in Figure 3.

2.1.5 Privacy Aspects of Personal Data

The General Data Protection Regulation, also known as GDPR, is a legal framework that sets guidelines for the collection, storage, and processing of personal data for individuals within the European Union (EU) .The regulation was implemented in May 2018 to provide individuals with greater control over their personal data and to ensure that organizations handling personal data are transparent, accountable, and secure. Under the GDPR, individuals have the right to access, correct, and erase their personal data, as well as the right to be informed about how their data is being used. Organizations must also obtain explicit consent from individuals before collecting their data, and must notify them in the event of a data breach. Failure to comply with the GDPR can result in significant fines and reputational damage for organizations [2].

As part of our commitment to GDPR compliance, we have incorporated various measures, such as email encryption, to enhance data protection, which is further discussed in Section 3.1.7.3.



Input side

Select Product
Balder CS

Calculation based on	
Width	Unit
151,25	inch

Screen Aspect
2.40:1

Throw Distance	Unit
152,00	inch

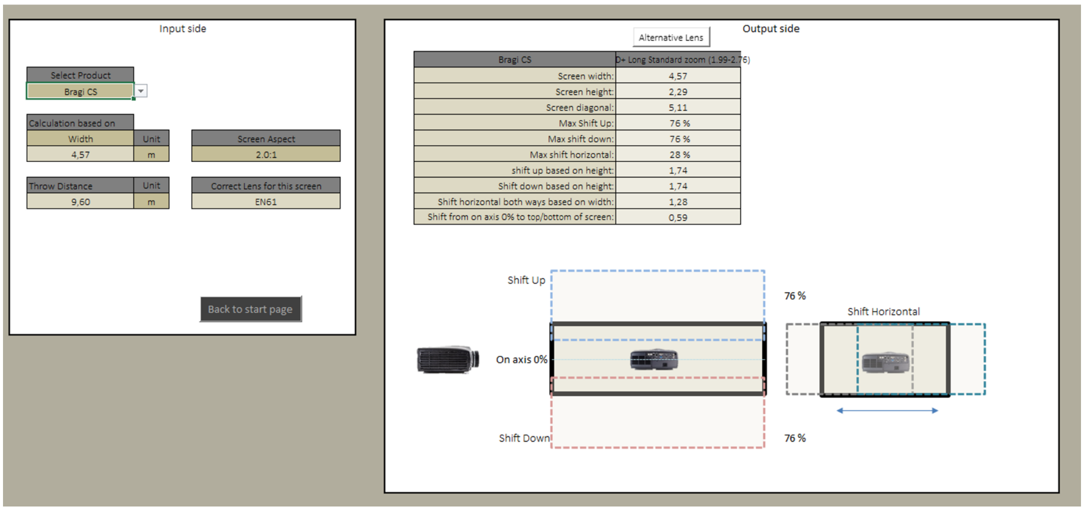
Optional Input

Screen Surface
Stewart StudioTek 130

Port window
Yes

Mirror
No

Figure 2: Barco Excel sheet simple project calculator.



Input side

Select Product
Bragi CS

Calculation based on	
Width	Unit
4,57	m

Screen Aspect
2.0:1

Throw Distance	Unit
9,60	m

Correct Lens for this screen
EN61

Output side

Bragi CS	D+ Long Standard zoom (1.99-2.76)
Screen width:	4,57
Screen height:	2,29
Screen diagonal:	5,11
Max Shift Up:	76 %
Max shift down:	76 %
Max shift horizontal:	28 %
shift up based on height:	1,74
Shift down based on height:	1,74
Shift horizontal both ways based on width:	1,28
Shift from on axis 0% to top/bottom of screen:	0,59

Figure 3: Barco Excel sheet lens shift calculator.

Additionally, we have enabled users to exercise their rights under GDPR article 17, commonly referred to as the *right to be forgotten*, by allowing them to delete their accounts and associated data.

As GDPR is a complicated matter, and it is the intention of Barco to continue development on this project, they should review, improve and expand the GDPR considerations that are taken before releasing this product to customers. This is due to the fact that we lack the necessary legal expertise to fully comprehend the complexities of GDPR regulations.

2.1.6 Requirement Priority

The different requirements for this project are not equal in priority, and we have decided to split them into three levels of importance, high, medium, and low. High indicates that this requirement is a critical component, and is essential to implement. Medium indicates that it is desirable to implement, but it is not critical. Low indicates that it is intended to be implemented at some point, but this can be a part of future works.

- **High Priority**: The most critical components of this project are the framework itself, the database, and the admin tool. These components are critical as all other components needs these features to work as intended, and therefore we choose to put these in high Priority.
- **Medium Priority**: In the medium priority group, we decided to put "Find Correct Projector" and "Simple Project Calculator". These calculators are desirable to be implemented, but no other features depend on them to work, and therefore their priority is set at medium.
- **Low Priority**: In the medium priority group we decided to put "Lens shift calculator", "Advanced offset calculator" and "Advanced project calculator". No other features rely on these calculators to work, and they can easily be implemented at a later point, and therefore their priority is set at low. We also set Privacy Aspects of Personal Data as low priority, even though this is a very important matter. The reason for this is that this product will not be directly released, but rather developed further before releasing, and therefore Barco can implement this as we do not have the legal expertise needed to fully understand the legal complications of the GDPR regulations and all its articles.

2.2 Technical Design

The technical design chapter is a critical component of any software development project, as it provides a high-level overview of the chosen solution. This chapter aims to outline the system architecture, which includes elements such as frameworks, databases, and hosting models. In this chapter, readers will gain insight into how the system was designed, as well as the reasoning behind the selection of architecture. Understanding the technical design is crucial to ensuring that the software solution meets the project's requirements and objectives. By explaining the big picture of the system, the technical design chapter serves as a foundation for the remaining chapters of the project. See Figure 1 for an overview of the technical design.

2.2.1 Blazor

Blazor is a framework developed by Microsoft, and is used to develop full-stack applications using HTML, CSS and C#, instead of JavaScript. The replacing of JavaScript with C# grants the opportunity to use the .NET framework, which offers a wide range of functionalities and capabilities. The .NET framework is also widely used, and very well documented, both by its official sources, and by its vast community. Blazor also uses Razor templates, which makes it easy to quickly create UI elements. This in combination with easy development and good performance makes Blazor a great choice for developing a web-application [3].

2.2.2 Hosting Models

Firstly, Blazor offers three different hosting models, called Blazor Server, Blazor WebAssembly and Blazor Hybrid. Choosing between these three hosting models was the first real decision that had to be made about the architecture of the application. To better understand the choice that was made, we first need to look into and understand what each hosting model has to offer. Table 1 contains an overview of the main differences between the models. The data is gathered from Microsoft [1].

Table 1: Shows the hosting models' features and support difference. Data gathered from Microsoft [1].

Feature	Server	WebAssembly	Hybrid
Complete .NET API compatibility	✓	✗	✓
Direct access to server and network resources	✓	✗	✗
Small payload size with fast initial load time	✓	✗	✗
Near native execution speed	✓	✓	✓
App code secure and private on the server	✓	✗	✗
Run apps offline once downloaded	✗	✓	✓
Static site hosting	✗	✓	✗
Offloads processing to clients	✗	✓	✓
Full access to native client capabilities	✗	✗	✓
Web-based deployment	✓	✓	✗

2.2.2.1 Blazor Server

With Blazor Server applications, the components run on the server instead of the client browser. Any UI updates or event handling are handled over a SignalR connection by using WebSockets protocol [4]. SignalR is an ASP.NET open-source library which simplifies adding a new real-time connection between the server and the client's browser [5]. While WebSockects is a communication protocol used to enable two-way communication from a client running untrusted code in a controlled environment to a remote host to communicate from that code [6].

UI updates and events are handled by the server rather than in the browser directly. When a request, event or UI update are sent to the server, Blazor renders the output of the components to an abstract object. Blazor then proceeds to compare the newest render with the previous render in order to calculate the difference. The difference found is then applied to the DOM and shown to the user [1]. Document Object Model (DOM) is used to connect web pages to scripts or programming languages by representing the structure of a document in memory. Normally, the DOM is used to make HTML elements reachable for JavaScript to update, add or delete elements on a web page [7].

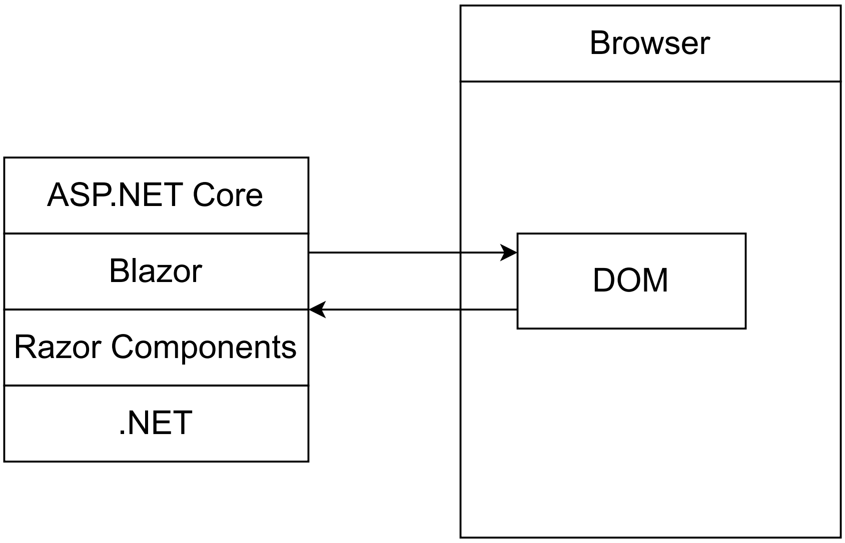


Figure 4: Blazor Server hosting model.

The components are run on the server rather than the user's browser as illustrated in Figure 4. Events that happen in the browser, such as editing a table or clicking a button, are sent to the server over a real-time connection to be handled. The response from the server will be the process just mentioned before, where Blazor will find the difference in the newest render and the previous one and apply the difference to the DOM. This heavily reduces the download size and the application loads faster. However, the big trade-off comes with network trafficking. Due to how execution is kept on the server side, every user interaction involves network traffic to and from the server itself. This creates problems for bigger applications and increases the response time of the user interface, as the server must be able to handle multiple client connections at the same time. This also eliminates any offline support, meaning without an internet connection, the app is unreadable for users [4].

2.2.2.2 Blazor WebAssembly

Firstly, WebAssembly is a low-level, assembly-like, language that can run in modern browser achieving near native performance due to the compact binary format. WebAssembly is used as a compilation target for higher-level language, like C# [8].

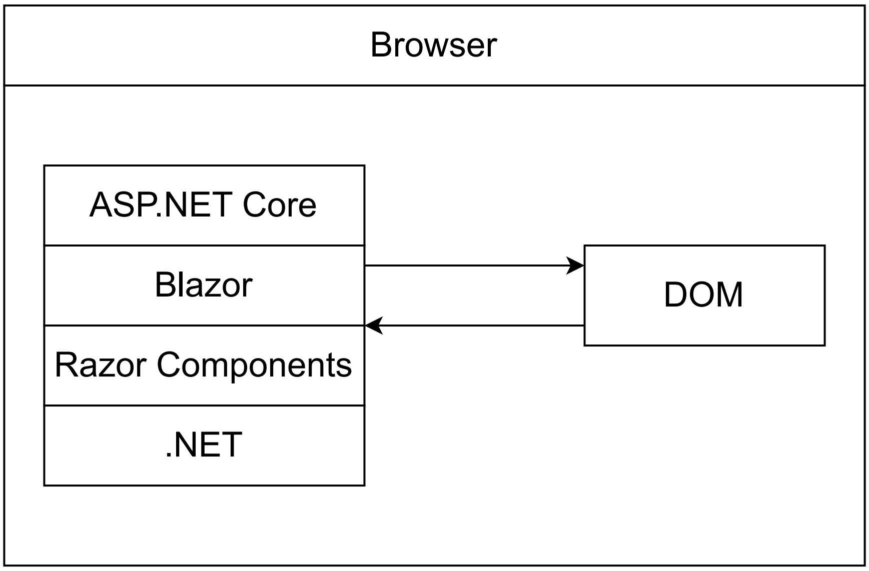


Figure 5: Blazor WebAssembly hosting model.

Within the Blazor WebAssembly model, there are two different hosting options. One runs the app client-side in the browser on a WebAssembly-based .NET runtime. This means that the app itself, any dependencies and the needed runtime is downloaded to the browser. Any UI updates and events are, unlike the Server model, handled within the browser, see Figure 5. This reduces the workload on the server hosting the application as well as better utilizes the user's resources. However, there are some drawbacks from this model, such as browser capabilities may restrict functionality and that a user with older hardware may experience performance issues. This model may be problematic for larger applications since the application, along with any dependencies, have to be downloaded by the user before it can run. This results in large download sizes and potentially long load times. When an application is hosted without any ASP.NET app as backend, the application is known as a standalone Blazor WebAssembly [4].

The other hosting options for a Blazor WebAssembly app is called core hosted. Instead of doing almost all execution on the server or in the client's browser, this option tries to get the best of both worlds. In this model, the project is split into three different parts. These parts being client, server and shared. The client solution works like normal Blazor WebAssembly, meaning every page or component in this part is rendered on the client side. While in the server, the solution is code which will only run on the server itself and send any result to the client. This is especially useful for protecting certain parts from the user, such as the hashing algorithm used to hash passwords stored in the database and making all requests to the database be from the server. The last part is named shared and is a part of the server solution. Shared allows the application to share code between the client and the server, thus ending up being a combination of the two hosting models [4].

Another notable feature of Blazor WebAssembly is the support for progressive web applications. This allows users to download the app to their own computer and to run it in its own window using modern browser APIs. The APIs can be used to enable some capabilities of a native app, such as working without network connection. To enable progressive web applications, one simply has to enable it during the creation of the project, like with the core hosted project setting [4].

2.2.2.3 Blazor Hybrid

Lastly, Blazor offers a hybrid model which is used to create native apps. In this model, components run directly in the native app, not in WebAssembly, along with any .NET code. Native apps are apps installed on a device and not accessed through the browser. To use this model, one must build the app with a .NET native app framework. Microsoft gives the following examples: .NET MAUI, WPF, and Windows Forms. Since this model is built around native support, they may offer functionality which is not supported in the most popular browser. However, one of the biggest flaws with this model is that separate native client apps must be built, deployed, and maintained for each target platform [4].

2.2.2.4 Choosing a Model

The Hybrid model was the first model we discarded when we were choosing a hosting model. This was mostly due to the fact that we only need to support the major browser and at least for now have no intention to support mobile users. Also, this model as mentioned needs separate clients for each target platform. During one of our meetings with Barco they also mentioned that they wanted to support offline use. This was not a part of the requirements, but a feature they wished to implement, nonetheless it dictates which model we are able to use. Offline support was mostly meant for Barco employees to be able to work while traveling. Due to this, we realized that the Server model could not be use since this model always requires network connection to be able to run.

After ruling out the hybrid model, only two options were left, a core hosted or standalone WebAssembly. We quickly decided that a core hosted model would be better. The benefits of a core hosted application, at least from the group's perspective, were far greater than with a standalone application. The main argument was that we are better able to control the database access with a core hosted model. With a standalone model, the user themselves would make a request to the database, but with a core hosted model, every request to the database goes to the server first. This way we will have greater control over the database. By having a core hosted application, we are able to better separative certain functionalities from the user's browser, such as our hashing and authentication functions. These are typical functions which are often seen as best practice to have on the server side. The primary justification for putting these features on the client side is that if you do not trust the server with your data, and the client should be the one handling it. In our case, we will have full control over the server side and will trust it with such data.

Barco wanted the option of offline support, thus we chose to activate the Progressive Web Application (PWA) when we constructed the basic layer of our applications. PWA uses modern browser APIs and capabilities to behave as a desktop application. It is called a Progressive Web Application (PWA) because a user will first use the application in the browser and later decide to download it to their operating system. Blazor WebAssembly can use any browser API, including the PWA APIs which is required for the following capabilities [9]:

- Working offline and loading instantly, independent of network speed.

- Running in its own app window, not just a browser window.
- Being launched from the host's operating system start menu, dock, or home screen.
- Receiving push notifications from a backend server, even while the user is not using the app.
- Automatically updating in the background.

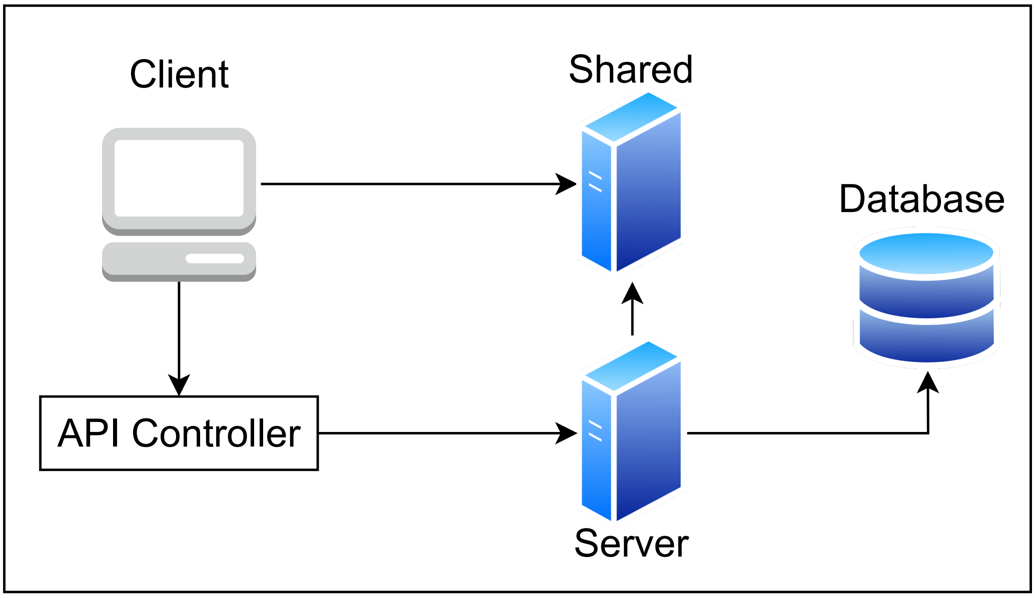


Figure 6: Illustration of the chosen hosting model and design.

Figure 6 illustrates how the chosen model will affect the technical design. The shared object in the figure are resources that both the server and the client can use, this is where most of the model classes will be stored as they will be needed by both parties. The figure shows how the server will be able to do direct request to the database, while the clients will have to send their request to the API controller. The controller is a part of the server, thus any request from the clients to the database will go through the server.

2.2.3 Database

A database is an important component of any software application where you intend to store data. In some projects where you store data you can avoid using a database, but you still have to implement another solution to handle the data, and often a database is the simpler option. When using a database, selecting the appropriate database type and database management system is essential to ensure efficient and effective operation of the application. It is therefore important to carefully consider what options that best suit the specific application you are implementing a database for.

2.2.3.1 SQL or NoSQL

When choosing between SQL and NoSQL, there are several factors that need to be considered, and one of the most important things to consider is what you are storing. The type of data you want to store in your database largely dictates what type of database is best suited for your specific needs. SQL databases are table based, which means there needs to be predefined tables where you input your data. This makes SQL suited for storing data that is highly structured, and has little to no variation in what fields you need.

On the other hand, NoSQL is document based, this makes the database very flexible and capable of storing unstructured data. This makes NoSQL suited for situations where you do not know exactly how the data you are storing is structured, or when the structure varies between elements [10].

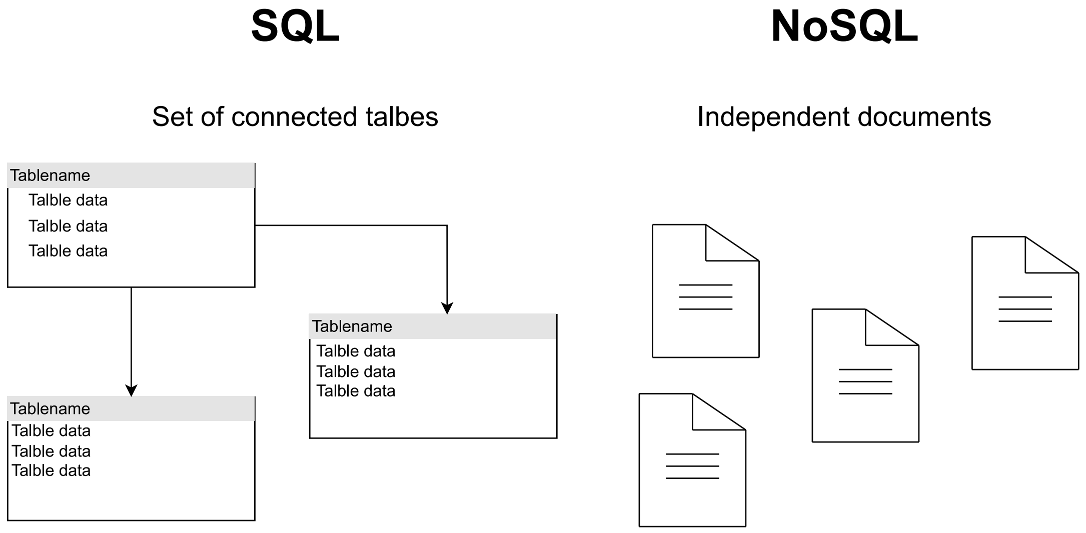


Figure 7: Illustration of SQL vs. NoSQL.

In Figure 7 you can see that SQL is illustrated as several connected tables, while NoSQL is illustrated as several not connected files.

The data we are storing in our database consists of information about projectors, lenses, and the users of the application. All the projectors, and lenses have the same fields for every unit, and we will be storing the same information about all the users. This means the data we are storing is highly structured data. Because the data we are storing is highly structured, a SQL database is a good choice for our project.

2.2.3.2 SQL Server Management Studio

SQL Server Management Studio (SSMS) is a tool created by Microsoft, and it is used to create and manage SQL databases. SSMS was developed by Microsoft, and therefore has good integration with other Microsoft products. It is also a very popular tool with a big online community, which makes it easy to get support or help solve problems. In addition to this, it is also the tool Barco use for their other projects that require databases. These factors make SSMS a good choice for us in this project.

2.3 Development Process

For the software development model, the group decided on using the scrum method, which is illustrated in Figure 8. This method is based on defining a set of tasks or features for a sprint. The duration of a sprint can vary between projects, and it can even vary within a project, but our general plan was to have each sprint last two weeks. We chose to set this duration so that we could set smaller sub-goals for our development, but still have enough time for each sprint so that we would get significant progress between each sprint. This method is used to better adapt to unforeseen problems or changes in requirement and allows for incremental development of the software. Another reason the group decided on this method is because after each sprint we would have an opportunity to get the status of the overall progress. From previous experiences, the group found this method to be the easiest to work with, since we would be able to determine what features or tasks that would be needed to be done during the next sprint [11].

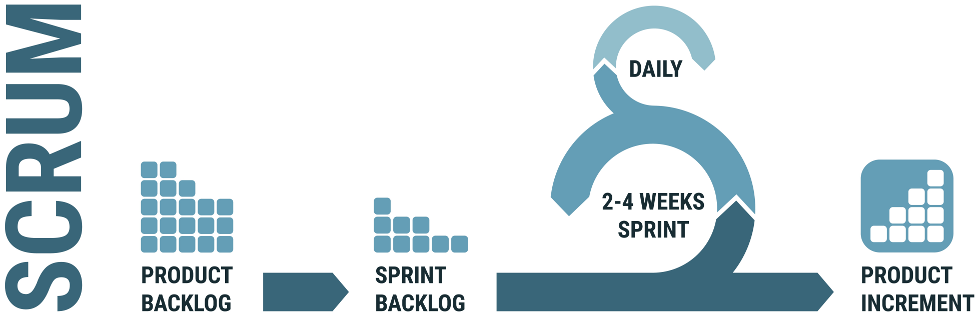


Figure 8: Illustration of the scrum process.

In total, we ended up with five sprints in the period 27.01.2023 to 17.04.2023. These sprints varied in length, with the longest being three weeks, and the shortest one week. After our last sprint, our group decided to not use sprints for the remaining period. This decision is further discussed in Section 5.5.

2.3.1 Database Development

Database development is a crucial aspect of software development that involves designing, creating, and managing databases to efficiently store and organize data. When using a SQL-database, it is also important to separate the data into several tables to keep your data organized. The following sections will explain the responsibilities and choices that affected the database.

2.3.1.1 Projector/Lens Database

This database holds all the information about the projectors, lenses and the compatibility between them. Previously, Barco had used an Excel spreadsheet to hold all this information. To be able to populate the database, we created a tool that could read the data from the spreadsheet and insert the data into the database. For the first iteration all the fields in the database were set to be of datatype nvarchar(255) which means all the attributes are data of type text with a max length of 255 characters. This was done due to the dataset not being consistent, and not to slow down the development of new features which needed data from the database. Later in the process, the tool was updated to use more concrete data types for the different fields, see Section 3.1.1.

The first tables created were Projector, Lens and ProjectorLens. The Projector and Lens tables were used to hold information about the specified product, while the ProjectorLens table is used to pair the projectors with the lenses that fit with the projector. This table contains three fields. One ID field, which is the primary key, meaning a unique identifier of the projector-lens pair. The other two fields are foreign keys which connects a record in the ProjectorLens table to one entry in the projector table and one entry in the lens table, thus making a projector-lens pair.

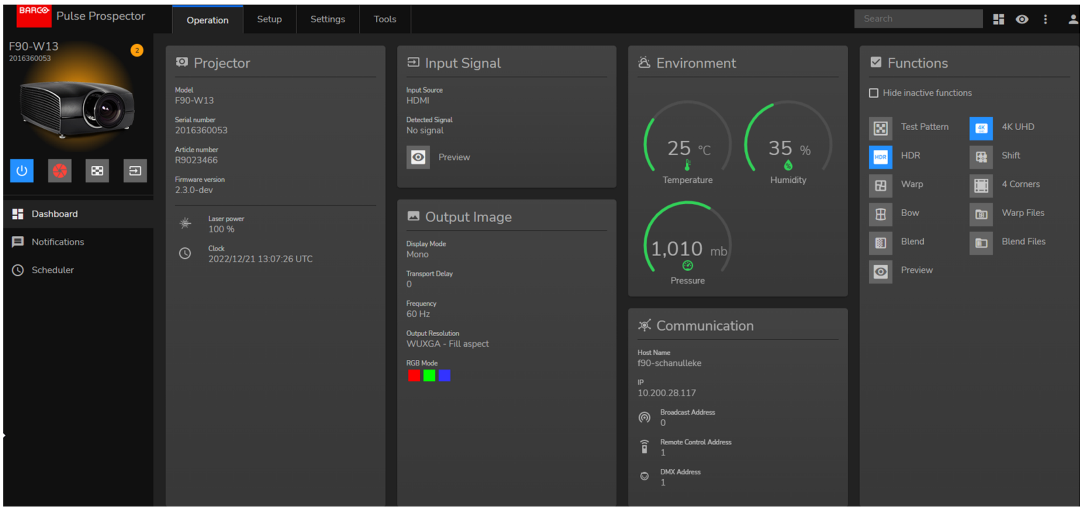
2.3.1.2 User Database

This database holds all needed information about users. This database was highly impacted more by the several iterations it went through, more so than the projector/lens database. This was due to changes in how different users' attributes were stored, such as the implementation of encryption on email addresses, which influenced how this attribute would be stored. This database design was also influenced by the need of other non-user related information, which was needed server side, such as a table for pre-approved domains.

All the attributes which needs to be stored as a string have been set to nvarchar(150). This means that all the fields, such as first name, last name and company information, can be stored as a normal text strings with a maximum of 150 characters. Another possible solution could be to make these attributes to have nvarchar(max). The valued passed into the nvarchar defines the string in size in byte-pairs, where max is a value of 4000. By using the max, a maximum storage size of $2^{31} - 1$ characters (2 GB) would be used [12]. This solution would allow for longer text to be stored, but would also increase the potential for abusive use cases by the users. With the mentioned solution, users would be able to store names of unrealistic length, which could be used to abuse the server. This could lead to the server using an unnecessary large amount of storage space to hold the user's information. Because of these drawbacks, the first solution was was deemed preferable.

2.3.2 Front-end Development

Barco wanted this program to follow the style and color scheme of their other software, so they provided us with a screenshot of a different program then have, that is used to control their projectors, and said they wanted something that looked similar.



Model
F90-W13

Serial number
2016360053

Article number
R9023466

Firmware version
2.3.0-dev

Laser power
100 %

Clock
2022/12/21 13:07:26 UTC

Input Source
HDMI

Detected Signal
No signal

Display Mode
Mono

Transport Delay
0

Frequency
60 Hz

Output Resolution
WUXGA - Fill aspect

RGB Mode
🟥🟩🟦

Host Name
f90-schanulleke

IP
10.200.28.117

Broadcast Address
0

Remote Control Address
1

DMX Address
1

☐ Hide inactive functions

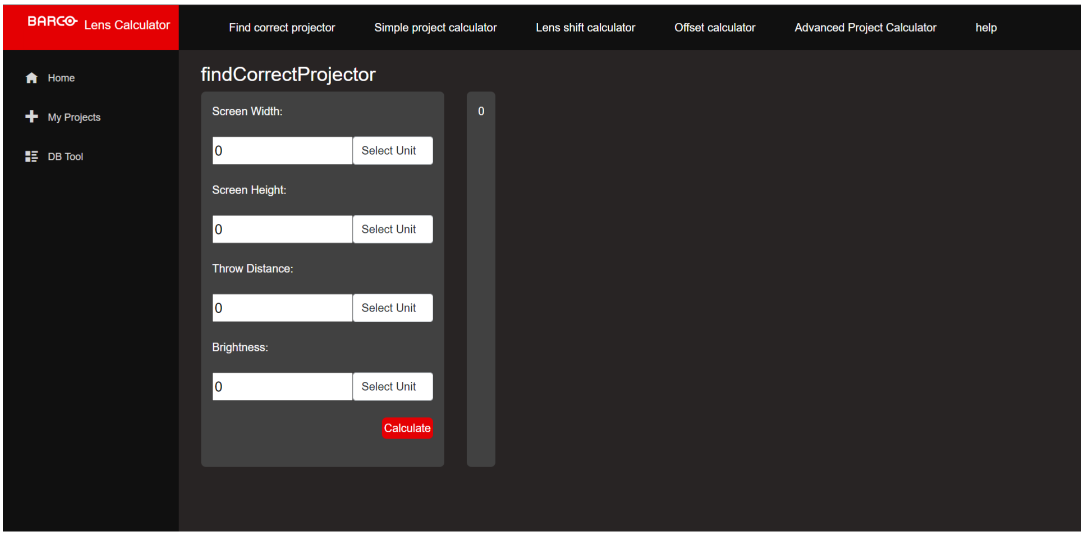
Figure 9: Barco example project.

To make sure the colors we were using were the same as in the other program, we used an online image color picker that gives you the hex code for the colors in the image. After finding the correct colors, we styled the different elements on the page. When the general style of the page was set, we started working on the different calculators. This consisted of creating a place to input the different values and units. This ended up taking several iterations, and many changes to get a layout that was both functional, and visually pleasing. Even though the different calculators have different fields, the way they are implemented are the same for all of them, therefore we could reuse a lot after the first one was made.

Figure 10 is a snapshot of an iteration of the "find correct projector" calculator. In this iteration of the calculator, the input section is finished, but there is not a section to display the output in an understandable way. This iteration also does not have any code to calculate the result.

Figure 11 is a later iteration of the "find correct projector" calculator, and now comes with a section to display the output of the calculation in a readable and easy to understand way. This iteration also has an early implementation of the code that does the calculation.

Later on, we decided it would be a good idea to give a visual representation of the throw ratio the lenses could be used for. To do this, we added a bar to show the range, and an indicator on the



findCorrectProjector

Screen Width: 0 Select Unit

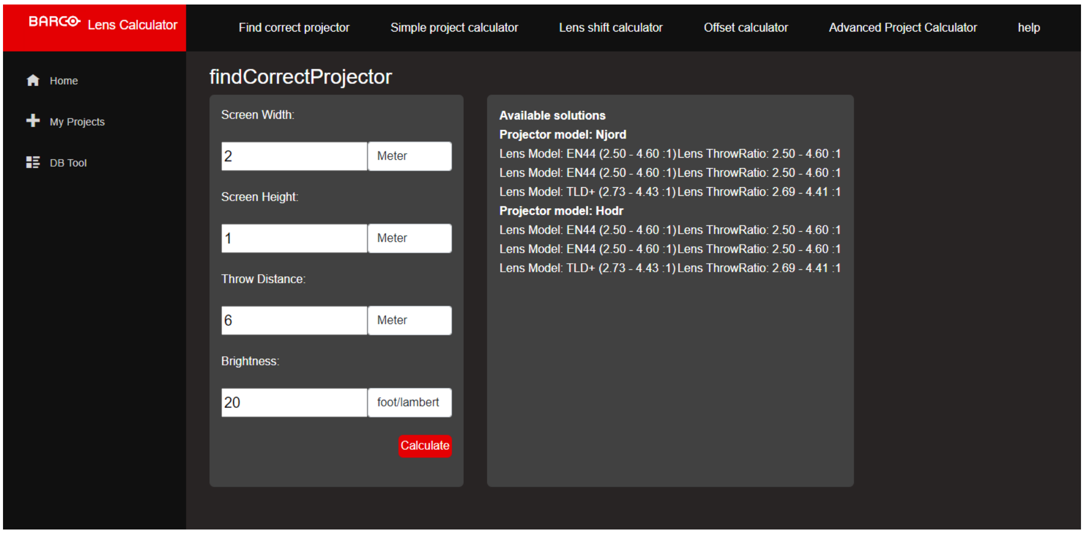
Screen Height: 0 Select Unit

Throw Distance: 0 Select Unit

Brightness: 0 Select Unit

0

Figure 10: Iteration of calculator front-end.



findCorrectProjector

Screen Width: 2 Meter

Screen Height: 1 Meter

Throw Distance: 6 Meter

Brightness: 20 foot/lambert

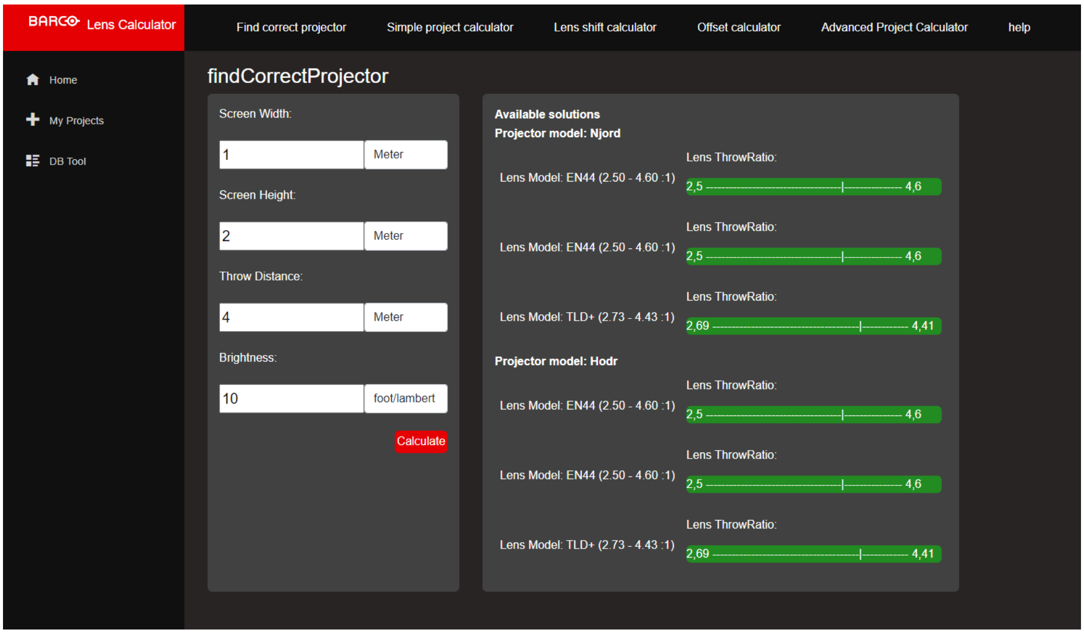
```
Available solutions
Projector model: Njord
Lens Model: EN44 (2.50 - 4.60 :1)Lens ThrowRatio: 2.50 - 4.60 :1
Lens Model: EN44 (2.50 - 4.60 :1)Lens ThrowRatio: 2.50 - 4.60 :1
Lens Model: TLD+ (2.73 - 4.43 :1)Lens ThrowRatio: 2.69 - 4.41 :1
Projector model: Hodr
Lens Model: EN44 (2.50 - 4.60 :1)Lens ThrowRatio: 2.50 - 4.60 :1
Lens Model: EN44 (2.50 - 4.60 :1)Lens ThrowRatio: 2.50 - 4.60 :1
Lens Model: TLD+ (2.73 - 4.43 :1)Lens ThrowRatio: 2.69 - 4.41 :1
```

Figure 11: More developed iteration of calculator.

bar to show where within the range the given throw ratio lies.
The simple project calculator is very similar to find the correct projector, because of this, we started with the same design with a few additional fields, and some minor adjustments to sizes. It also has an additional output that shows all the screen dimensions and the projector output. After this was implemented, the calculator looked like Figure 13.

After the new inputs and outputs were made, we implement saving and loading projects into the simple project calculator. To do this we added a new save button next to the calculate button, and a new window with a drop-down menu of your projects, and a load button. we also added a delete button here, so that you can delete your old projects. We also did a few other minor tweaks, all of which can be seen in Figure 14.

For the admin page we needed icons for buttons that redirect the user to the corresponding admin tool, and to avoid having to deal with copyright, we decided to create them on our own. To create these icons, we used Photoshop. The icons can be seen in Figure 15.



findCorrectProjector

Screen Width: 1 Meter

Screen Height: 2 Meter

Throw Distance: 4 Meter

Brightness: 10 foot/lambert

Available solutions

Projector model: Njord

Lens Model: EN44 (2.50 - 4.60 :1) Lens ThrowRatio: 2,5 —|— 4,6

Lens Model: EN44 (2.50 - 4.60 :1) Lens ThrowRatio: 2,5 —|— 4,6

Lens Model: TLD+ (2.73 - 4.43 :1) Lens ThrowRatio: 2,69 —|— 4,41

Projector model: Hodr

Lens Model: EN44 (2.50 - 4.60 :1) Lens ThrowRatio: 2,5 —|— 4,6

Lens Model: EN44 (2.50 - 4.60 :1) Lens ThrowRatio: 2,5 —|— 4,6

Lens Model: TLD+ (2.73 - 4.43 :1) Lens ThrowRatio: 2,69 —|— 4,41

Figure 12: Visual representation of throw ratio.

2.3.3 Quality Assurance

Quality Assurance (QA) is a process that ensures that a product or service meets specific quality standards before it is released to the market. This is done to reduce the number of errors and issues by catching them early in the development process. When working with, QA it is important to be critical, and non-biased. It is therefore a good idea to follow a set of concepts and guidelines to help with this. These concepts and guidelines will be discussed further below.

2.3.3.1 Version Control

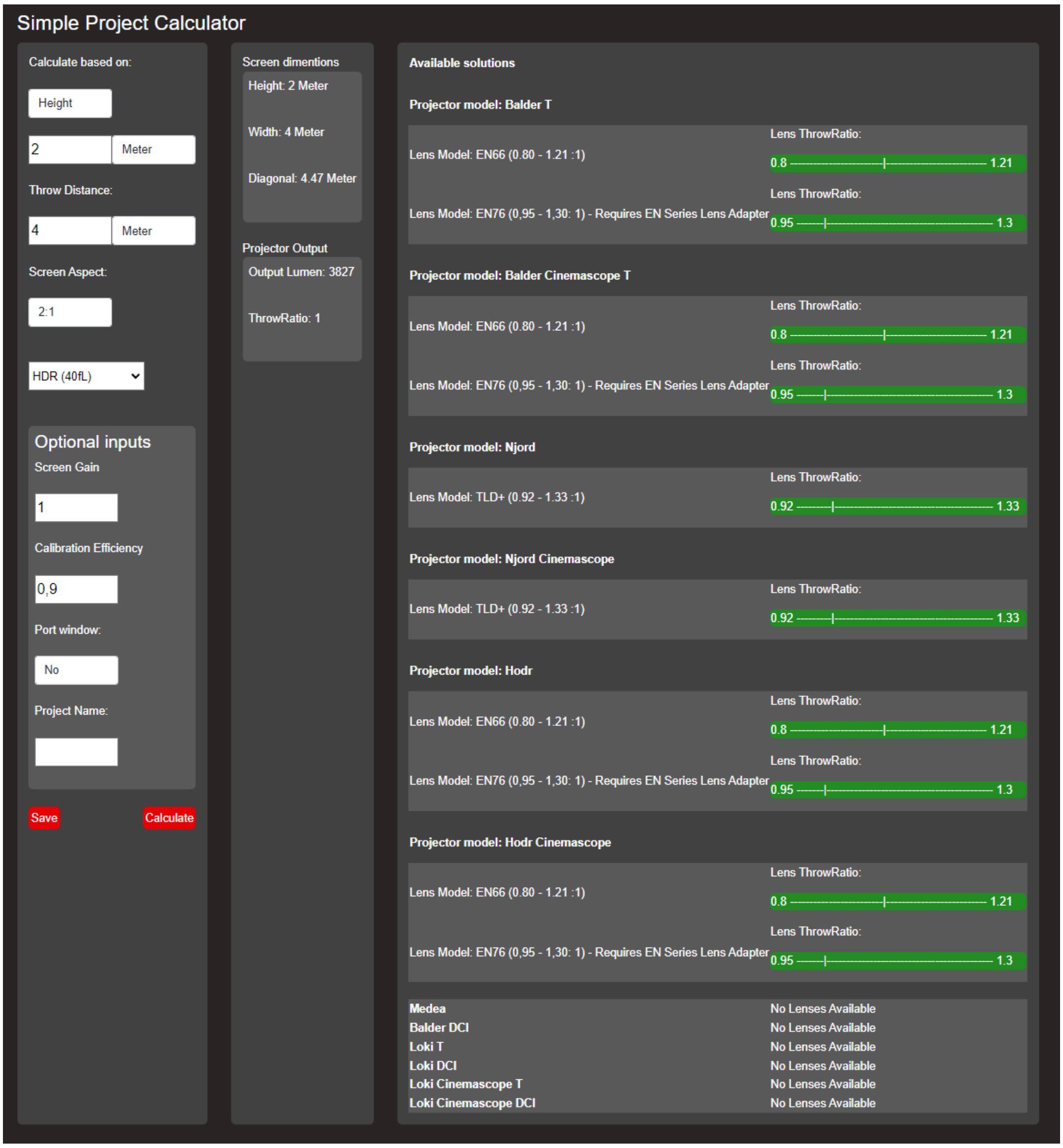
Version control, also known as source control, is an important tool to use when working on large projects. Version control is used to keep track of changes to files, so that at any point you can go back to a previous version. This also means that you have a backup of all the files, so if your local version of the files gets corrupted or lost, you can get them from your version control system.

The simplest implementation of version control is just manually taking a backup of files. This is very inefficient, and if your project involves several developers it would be impossible to keep track of where the different files are, what version should be used, and so on. For this very reason, people have implemented better version control systems. one of, if not the most used version control system is called “Git”.

Git is a version control system that can also be used as a distributed version control system, which means that all the projects files and their change history are stored in a repository, and that repository is mirrored on every developer’s own computer. This makes it very easy to collaborate on projects, and know that you have all the changes the other developers made.

Git also has more advanced features like branching, which are essentially different versions of the same codebase that can be worked on independently of one another. They allow developers to work on different features or bug fixes simultaneously without affecting the main codebase.

To use git with other developers, you need to have a remote repository in addition to your own local repository. There are several services that implement this remote repository, the most well known are GitHub [13] and GitLab [14]. GitHub and GitLab have some key differences, but they essentially do the same job, and therefore the choice often comes down to personal preference.



Simple Project Calculator

Calculate based on: Height

2 Meter

Throw Distance: 4 Meter

Screen Aspect: 2:1

HDR (40fL)

Optional inputs

Screen Gain: 1

Calibration Efficiency: 0,9

Port window: No

Project Name:

Screen dimentions

Height: 2 Meter

Width: 4 Meter

Diagonal: 4.47 Meter

Projector Output

Output Lumen: 3827

ThrowRatio: 1

Available solutions

Projector model: Balder T

Lens Model: EN66 (0.80 - 1.21 :1) — Lens ThrowRatio: 0.8 – 1.21

Lens Model: EN76 (0,95 - 1,30: 1) - Requires EN Series Lens Adapter — Lens ThrowRatio: 0.95 – 1.3

Projector model: Balder Cinemascope T

Lens Model: EN66 (0.80 - 1.21 :1) — Lens ThrowRatio: 0.8 – 1.21

Lens Model: EN76 (0,95 - 1,30: 1) - Requires EN Series Lens Adapter — Lens ThrowRatio: 0.95 – 1.3

Projector model: Njord

Lens Model: TLD+ (0.92 - 1.33 :1) — Lens ThrowRatio: 0.92 – 1.33

Projector model: Njord Cinemascope

Lens Model: TLD+ (0.92 - 1.33 :1) — Lens ThrowRatio: 0.92 – 1.33

Projector model: Hodr

Lens Model: EN66 (0.80 - 1.21 :1) — Lens ThrowRatio: 0.8 – 1.21

Lens Model: EN76 (0,95 - 1,30: 1) - Requires EN Series Lens Adapter — Lens ThrowRatio: 0.95 – 1.3

Projector model: Hodr Cinemascope

Lens Model: EN66 (0.80 - 1.21 :1) — Lens ThrowRatio: 0.8 – 1.21

Lens Model: EN76 (0,95 - 1,30: 1) - Requires EN Series Lens Adapter — Lens ThrowRatio: 0.95 – 1.3

Medea	No Lenses Available
Balder DCI	No Lenses Available
Loki T	No Lenses Available
Loki DCI	No Lenses Available
Loki Cinemascope T	No Lenses Available
Loki Cinemascope DCI	No Lenses Available

Figure 13: New combined calculator.

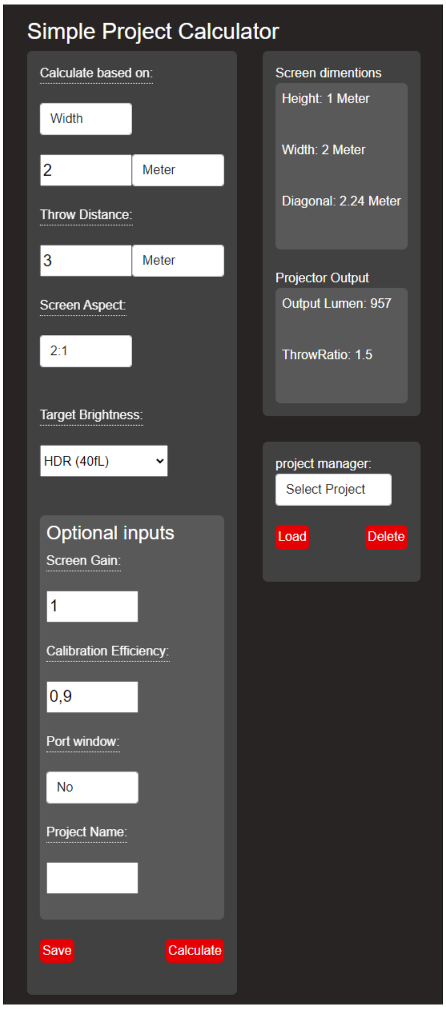
GitHub and GitLab differ primarily in the philosophy that underlies each platform. While GitHub prioritizes infrastructure performance and boasts higher availability, GitLab focuses on providing a centralized, integrated platform for web developers with a variety of features [15]. In our case, we chose to use GitLab, as we have experience with it from previous projects.

2.3.3.2 Coding Best Practices

Coding best practices are a set of rules and guidelines that developers should try to follow when writing code. This is to help ensure good code quality, and to make it easier to read, understand, and maintain. Following these best practices can also help prevent common coding mistakes and improve the efficiency and performance of the code.

Cohesion

”The term cohesion relates to the number and diversity of tasks for which a single unit of an application is responsible” [16, p. 260]. Cohesion is applicable to an entire class or a single method. One



Simple Project Calculator

Calculate based on:

Width

2 Meter

Throw Distance:

3 Meter

Screen Aspect:

2:1

Target Brightness:

HDR (40fL)

Optional inputs

Screen Gain:

1

Calibration Efficiency:

0,9

Port window:

No

Project Name:

Save Calculate

Screen dimentions

Height: 1 Meter

Width: 2 Meter

Diagonal: 2.24 Meter

Projector Output

Output Lumen: 957

ThrowRatio: 1.5

project manager:

Select Project

Load Delete

Figure 14: Project loader.

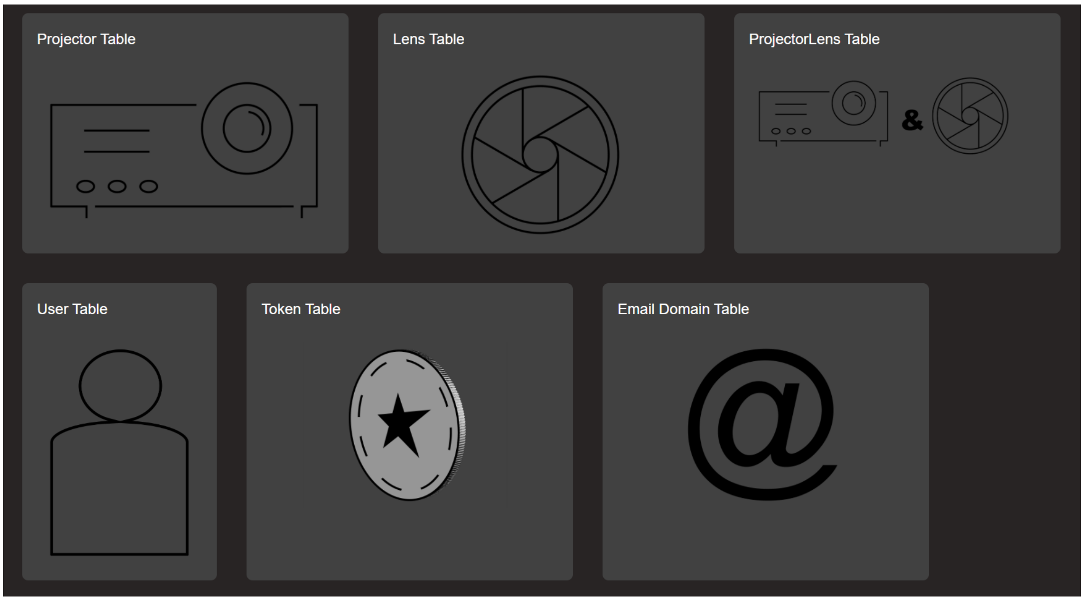


Figure 15: All DBtool icons.

class or method should fulfill a singular purpose. The purpose of cohesion is reuse. If a method or a class is very specialized, and only responsible for one well-defined thing, then it is more likely to be reused at a later stage [16, p. 260].

One should strive to achieve a high degree of cohesion by having specialized classes and methods. In our code, we attempted to achieve a high degree of cohesion by separating classes by their responsibilities and creating methods that only perform a singular task.

Coupling

"The term coupling refers to the interconnectedness of classes. We aim to design our applications as a set of cooperating classes that communicate via well-defined interfaces. The degree of coupling indicates how tightly these classes are connected" [16, p. 259]. We aim for a low degree of coupling, also called loose coupling. The coupling of classes indicates how they behave when changes are made. In tightly coupled classes, a change in a class will necessitate changes in the other classes as well, while loosely coupled classes will remain mostly unaffected. Tightly coupled classes generally require more effort to maintain and are harder to work with [16, p. 259].

Code Duplication

Code duplication is a term for when nearly identical code is present in multiple functions, and is an indicator of bad design. "The problem with code duplication is that any change to one version must also be made to another if we are to avoid inconsistency. This increases the amount of work a maintenance programmer has to do, and it introduces the danger of bugs" [16, p. 261]. When a maintenance programmer changes a piece of code to fix a bug, he assumes that the bug is fixed and could easily overlook the bug that comes from the duplicate code [16, p. 261].

If we notice, while coding, that different functions have lines of code that are identical, we will move the identical parts into its own function and use that function where the duplicates occurred. This way, any changes that are made in the new function are also applied wherever the function is used.

Encapsulation

"The encapsulation guideline suggests that only information about what a class can do should be visible to the outside, not about how it does it" [16, p. 266]. Developers can be sure that any changes in how the class works will not affect other classes as long as the output of the class remains unchanged. We followed this guideline by making every field and function, that is not used or intended to be used by other classes, private. This ensures that no information about how the class stores information and performs its tasks is available to the outside.

3 Implementation and Deployment

The implementation and Deployment chapter aims to transform the requirements and design choices outlined in Chapter 2, into a functional software system. To do this, the following chapter will explain the technical side of the solution, and explain the implementation of the program. It will also explain the way it is deployed.

3.1 Implementation

The implementation section will outline the specific way different technical components of the project were implemented.

3.1.1 Excel to Database Tool

To populate the database with product information, a tool which reads an Excel file was developed. This tool was used to read the raw data from the Excel file provided by the task giver, and then insert the data into the database. The Microsoft library called Microsoft.Office.Interop.Excel makes manipulation of the Excel file possible. To update the database with the read data, Entity Framework was used. This tool uses Entity Framework (EF6) which is not the same as the main web application that uses Entity Framework Core. The two libraries have the same use cases and are almost identical [17]. However, Microsoft recommends using Entity Framework Core for new development [18]. The main reasoning behind using EF6 is that development on this tool was started already in the previous summer, when one of our group members had an internship at Barco. During the initial phase of development during that summer, it was recommended by one of Barco‘s employees that EF6 should be used.

The tool creates two predefined dictionaries that contain the indexes of the different attributes. A dictionary is an object which holds key-value pairs. Each dictionary holds the attribute name as the key, and the value is the index that the attribute value can be found at. By using this solution, any changes to the position of different values in the Excel spreadsheet would need to be reflected in the creation of indexes. The tool uses in total four dictionaries. Two are used for the attributes indexes of lenses and projectors. While the last two are used to enable creation of projector/lens pairs. One of the last two dictionaries is used to hold the projector model as the key and the projector ID as the value. The other dictionary holds the projector model as the key and a list of integers as the value. The list of integers holds all the ID of all the lenses that a projector can use. The last two dictionaries are used to create all the projector/lens pairs, and is done by iterating over each element in the dictionary, which holds the list of lens IDs. This allows the tool to get each key value (projector model) and the value (list of lens IDs). Once the tool has the projector model, it uses this variable to get its own ID from the first dictionary and the ID of all compatible lenses in the second dictionary.

3.1.2 Database

The database consists of two separate databases: A user database which consists of information about the users and their projects, and a database which represent the different projectors, lenses, and their compatibility.

3.1.2.1 User Database

The user database consists of six tables: user table, user email table, tokens table, project saving table, domain table, keys table and shared keys table. The last three tables are independent, and are floating without a connection to any table. The domain table is only used by the server and the admins. It contains information about pre-approved email domains. When a new user is registered, a confirmation email will be sent if the user is from a pre-approved domain, or an admin will have to do it manually. Shared keys hold the shared encryption key used to encrypt and decrypt user emails, while keys table hold all the keys used to create the shared key. User email table holds the encrypted email address of users. Since one user can only have one email address, the connection between the email table and the user table is one-to-one. While the connection between user table and both tokens and project saving table is one-to-many, all of these connections are illustrated in Figure 16. This is because one user can save many projects, while one project can only have one user. The same goes for tokens, one user can have many tokens, while one token can only be connected to one user.

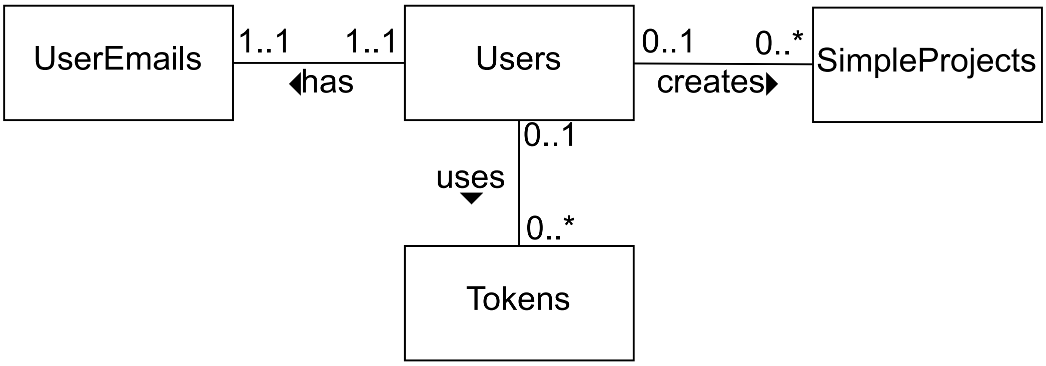


Figure 16: Relationship diagram showing the relationship of tables in user database. A more detailed figure can be seen in the appendix A, Figure 40.

3.1.2.2 ProjectorLens Database

The projector database contains three tables, projector table, lens table, and a projector-lens table. The projector holds information about each of the projectors, while the lens table holds information about each lens. It is important to highlight that one lens model may have multiple entries into the lens table. This is due to changes in the values of different parameters based on which projector they are connected to. For example, one lens model may have different shift value based depending on which projector they are paired with. Shift value is how much the lens can move the image without moving the projector itself. The ProjectorLens table is used to make a projector/lens pair. One projector can have many projector lenses, thus making the connection one-to-many. Due to the variation of values in lenses, one lens can only have one projector lens, thus making the connection one-to-one, as can be seen in Figure 17. Since the last mention connection is one-to-one, we could have dropped the ProjectorLens table entirely and just have a one-to-many connection between projector and lens. We decided against this due to wishes about adding lenses without have connection to a projector. If we had a one-to-many connection, the foreign key reference would either allow null, be set to a dummy value, or being set on creation. By having the ProjectorLens table, we avoid this problem. New lenses can be added with our current solution without a relationship to a projector, and the relationship can be added later at any point in time.

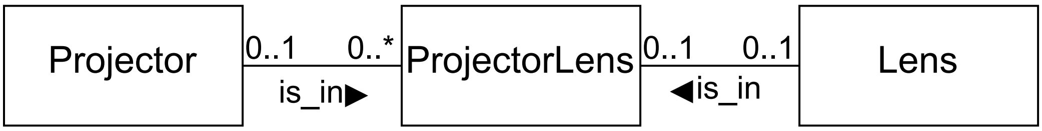
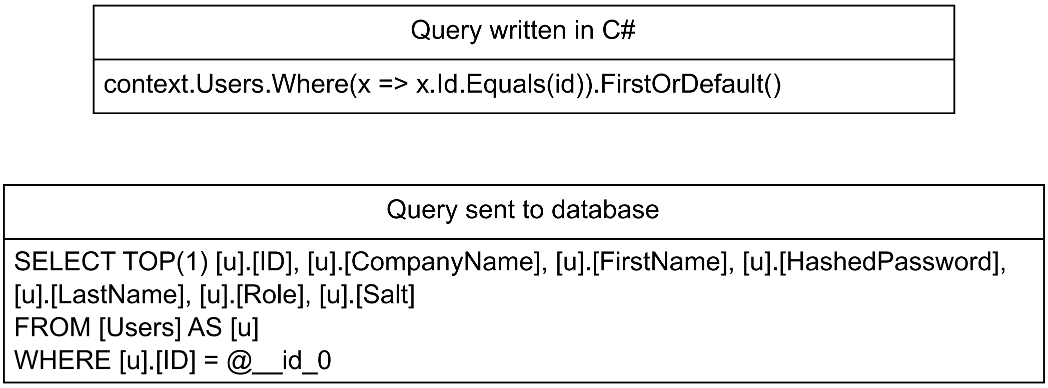


Figure 17: Relationship of tables in projector/lens database. A more detailed figure can be seen in the appendix A, Figure 41.



Query written in C#

```
context.Users.Where(x => x.Id.Equals(id)).FirstOrDefault()
```

Query sent to database

```
SELECT TOP(1) [u].[ID], [u].[CompanyName], [u].[FirstName], [u].[HashedPassword],
[u].[LastName], [u].[Role], [u].[Salt]
FROM [Users] AS [u]
WHERE [u].[ID] = @__id_0
```

Figure 18: Simplified example of how Entity Framework translate a query written in C# into SQL.

3.1.2.3 Database Setup

To host our database, we used a virtual machine, also known as a VM, which has the sole purpose of hosting the database and nothing more. To create a MSSQL server, a Microsoft SQL server, we followed a Microsoft tutorial which goes through each step of the process [19]. Once the SQL server was up and running, we could use SQL Server Management Studio to connect, once a connection was established, we could write SQL queries to create the needed tables and relationship. The next process was to populate the projector/lens database with information from multiple Excel spreadsheets. Instead of manually populating the database, we created a C# console application which reads the Excel spreadsheets and adds all projectors and lenses to their respective tables. We also had to add logic for adding the compatibility between the projectors and lenses.

3.1.3 Admin Tool

The admin tool allows for easier manipulation of the database, which contains information about all the projectors and lenses that Barco sell. The database also contains information about the users of the application and their projects.

To access the database and allow for changes to be made through the code, Microsoft's Entity Framework was used. To enable interaction between the code and database, a DbContext class was implemented. This class works as a layer between the code of the application and the database. The context class is used to create a session with the database and can be used to query and save instances of different entities. The class also contains the name of all the tables in the database and of what object the data consist of. This enables the application to cast the results of queries directly into C# objects [20]. Since the application have two separated databases, two of these classes were needed, one for each database. Entity Framework also has the advantage of enabling SQL query free usage. The framework uses something called Language-Integrated Query (LINQ), that enables developers to write queries in C# against the database context without using any SQL commands [21]. The framework will translate the written C# query into a SQL query which is sent to the database and then casts the result into an object. See Figure 18 for an example. However, the package also allows the developer to write normal SQL queries. This is something one usually does not need, but in some cases the framework translates the code query ineffectively, which tempts developers to write their own queries.

To present the tables from the database, the task provider wanted to use a licensed package from Infragistics called "IgniteUI.Blazor". From this package, an element called "IgbGrid" was used to create the tables. In this element we can define a variety of different properties for the table, such as what columns to display, which can be edited, if the table can be exported to an Excel file, to name a few of the possible properties. The table comes pre-styled by the package. By setting the table to be editable, the application allows the user to double-click on a row and then edit that row directly in the table. To pull information from the edited row, we created a function that would be called when the user finished editing the row called editTable. This function creates a JSON element representing the changes and sends a post request to the API containing the JSON element to make the corresponding changes in the database.

The main page for admin tool contains six buttons, which when clicked will redirect the user to the Projector, Lens, ProjectorLens, Domain, Token, or User page.

3.1.3.1 Projector and Lens Pages

The purpose of these pages is to provide a visual representation of the database and allow the administrator to add, edit and delete entries in the database. For these tables to communicate with the database, we created a web API where certain actions on the tables would make a GET/POST request to the API, which in turn would use the database context to perform the changes in the database. When the page loads, it sends a GET request to the web API. The web API calls a database context function. The context function sends a SQL query to the database and returns the values received from the query, in this example it would be the entire projector/lens table.

3.1.3.2 User Table

The user table is only visible to the administrators of the application. Here, admins can see a visual representation of all the registered users as a table and will be able to see both users from pre-approved domains and user that have not been approved yet. For more information about the registration logic, see Section 3.1.5. Unlike the Projector and Lens tables, the admins are not allowed to create new user directly in this page, all users have to use the normal registration form. But they are allowed to edit the table, which in turn will update the database. The main reason to have this table be editable is to allow admins to change the roles of a user, such as making a user admin or manually approving a new user. The latter is not recommended, because this will skip the email confirmation logic. Other than this, user page has two key features, which is to delete a user and to send a confirmation email to a user from a not pre-approved domain.

3.1.3.3 Token table

The token table contains all stored tokens created by the server to handle confirmation and reset password requests. In some way this table is connected to the user table since each token is created for a specific user. Meaning, each token entry in the table contains a user ID corresponding to a user in the user table. This table however is not editable for the admins, the only functionally admins have over this table is the possibility to remove all tokens which are no longer valid based on their time of creation. For more information about the creation and validation of tokens, see Section 3.1.7.1.

3.1.3.4 Domain table

The domain table contains an overview of all the pre-approved email domains. Admins have the opportunity to delete or add new email domains. For more information about how the email domain is used, see Section 3.1.5.

3.1.4 Calculator Code Implementation

The calculator code can be divided into a few different parts. The code for the calculator itself, code for saving loading and deleting projects, and code for visualizing the throw ratio.

3.1.4.1 Simple Project Calculator Code

The code for the “simple project calculator” works with a few relatively simple functions. The first of these functions use the value given for height, width or diagonal, the corresponding unit, and the aspect ratio to calculate all screen dimensions. An example of this could be that the user inputs a screen height of 2 meters and an aspect ratio of 2:1, and the calculator then calculates the screen width, which would be 4 meters, and the screen diagonal, which would be 4.47 meters.

The next function is used to calculate the throw ratio. By taking the given inputs for throw distance, and the corresponding unit, and the screen width that was inputted or calculated in the previous method. Then we convert everything to meters, because both parameters needs to be of the same unit for the calculations to be correct. Then we take the throw-distance and divide by the screen width, this will give us the throw ratio. After converting the units, the formula for calculating the throw ratio would look like Formula 1.

$$Throwratio = \frac{throwDistance}{screenWidth} \tag{1}$$

The next method is used to calculate the output brightness needed from the projector to hit the target brightness. We take the inputted target brightness and the corresponding unit, and convert it to foot-Lambert. Then we take the target brightness and multiply it by the screen size in square feet. After that, we divide that value by the screen gain multiplied with the calibration efficiency and the port window efficiency. The result is the amount of lumen the projector needs to output to hit the target brightness. The formula for this would look like Formula 2.

$$Outputbrightness \quad = \quad \frac{(TargetBrightness \times screenSquareFeet)}{(gain \times calibrationEfficiency \times portWindowEfficiency)} \tag{2}$$

The last method simply sends a query to the database and gets the correct projectors and lenses based on the values from the two previous functions, and then creates a dictionary, which is an object that holds a key value pair, with the projectors as the key, and a list of the lenses that are compatible with the projector as the value.

As can be seen in Figure 19 most of the functions return a value to the calculate function, instead of calling the next method from the previous. This is done, so the internal workings of each function does not impact the rest of the functions. This is the concept of loose coupling, and is done to make it easier to alter the inner workings of a method, without having to rewrite all the other functions.

3.1.4.2 Project Saving, Deleting and Loading

The simple project calculator also has the ability to save, load, and delete projects. These functions each have a simple method to do their respective task.

The project saving method starts by checking if the project has a name, if not the user gets an alert telling them they need a project name. When the user has given a project name, we check if they already have a project with this name. If they do, they get a confirmation window which asks the user if they want to overwrite the project with the same name. If they answer yes, the old project is deleted and the new project is saved. When the project is saved, the user gets an alert to tell them that the project was successfully saved. When a project is saved, all the numbers and units are saved, but in addition to that we also store the user-id with every project. This is done, so we know which user created each project, and it is later used in the project loading.

The project deletion method first needs the user to select a project from a drop-down list. Then

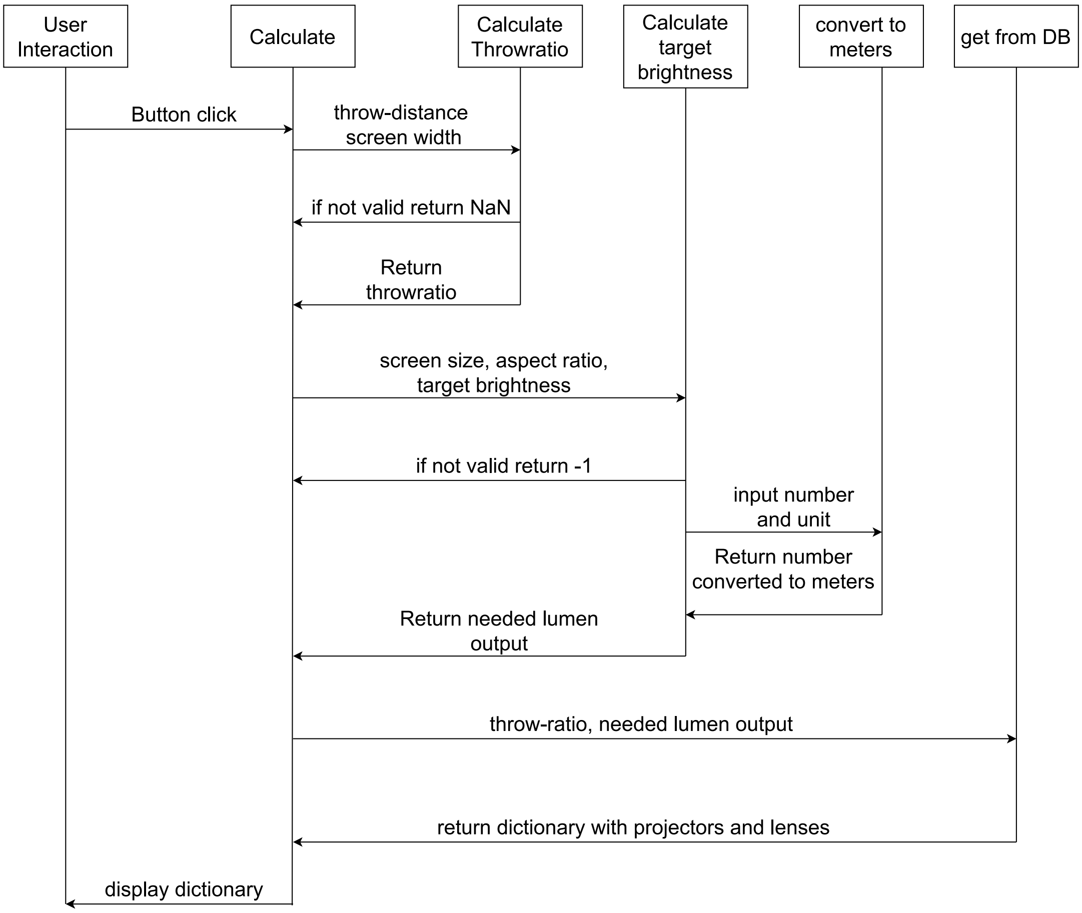

User Interaction
Calculate
Calculate Throwratio
Calculate target brightness
convert to meters
get from DB
Button click
throw-distance screen width
if not valid return NaN
Return throwratio
screen size, aspect ratio, target brightness
if not valid return -1
input number and unit
Return number converted to meters
Return needed lumen output
throw-ratio, needed lumen output
return dictionary with projectors and lenses
display dictionary

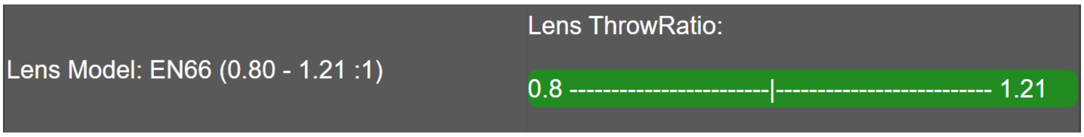
Figure 19: Sequence diagram illustrating how the code for the calculator works.

when they click on the delete button they get a confirmation window, and if they confirm the project is permanently removed.

When opening the simple project calculator a list of all the users projects is collected from the database, to do this we compare the user-id field saved in the database with the user-id of the current logged-in user. Then the user can choose a project from a drop-down list, and press the load button. When the user clicks the load button and a project is selected, all the input fields are set, and the calculate method is run, so the user can see the result.

3.1.4.3 Throw-ratio Visualization

The way we visualize the throw ratio was initially intended as a temporary proof of concept, but in the end we did not end up replacing it. Before this application will be deployed to customers, the way this information is visualized should be improved. The way this looks can be seen in Figure 20.



Lens Model: EN66 (0.80 - 1.21 :1)

Lens ThrowRatio:

0.8 -----------------------|------------------------- 1.21

Figure 20: Visual representation of throw-ratio.

To visualize the throw ratio, we first have to change the range to be between 0 and z instead of x to y. To do this we take the max throw-ratio value and subtract the min throw-ratio value, in our code we called this variable “normThrowDB”, as in normalized throw-ratio from the database. Then we take the throw-ratio calculated from the userinput and subtract the min throw-ratio to normalize that value as well, we called this variable “normThrowIn”, as in normalized throw-ratio from input. Next, we check if the calculated throw-ratio is bigger than the min throw-ratio of the lens, as the next step would not work if it is outside the range. Then a variable called “normThrowRatio” is created. This variable is normThrowIn divided by normThrowDB and then multiplied by 100. This tells us how far along the throw range the value is, in the form of a percentage. Then we divide normThrowRatio by an integer value to scale it down, in this case that value is 2. the value we get from dividing normThrowRatio by the scale is then used in a stringbuilder to append that amount of hyphen’s. then a vertical line is appended, and finally we append 100 minus the normThrowRatio divided by the scale. This function has several more steps in between the ones explained before, these steps are for validating the result to make sure the result is valid. These validation steps are removed in code snippet 1, to make it more readable, and easier to understand.

```
private string throwRatioVis(LensPair lens)
  {
      //makes the throwratio between 0-y instead of x-y
      double normThrowDB = lens.ThrowRatioMax - lens.ThrowRatioMin;
      double normThrowIn = throwRatio - lens.ThrowRatioMin;

      //how many percent along the range the throw ratio value is
      int normThrowRatio = (int)((normThrowIn / normThrowDB) * 100);


      // used for making string shorter
      int scale = 2;

      stringBuilder.Append('-', normThrowRatio / scale);
      stringBuilder.Append('|');
      stringBuilder.Append('-', 100 / scale - normThrowRatio / scale);

      return stringBuilder.ToString();
  }
```

Listing 1: Code snippet of simplified throwRatioVis function

3.1.4.4 Find Correct Projector Code

The calculations for "Find correct projector" is mostly the same as for "Simple project calculator". The main difference is that here you have to manually input the screen height and width instead of using the aspect ratio to calculate the dimensions. This calculator also does not include the step where we divide by the screen gain multiplied with the calibration efficiency and the port window efficiency.

3.1.5 Registration

All the user inputs are checked in the page itself. Here we check that both email address fields are equal and by using regex we make sure that the password given by the user is at least eight characters long, contains at least one number, one uppercase, one lowercase and one special character. This is to make sure that the user has a somewhat strong password. If the user input passes all the checks, a POST request is sent to the server with all the user information in the body. Next the controller for the registration endpoint will check if the given email address already is in the database, if that is the case the controller will return a status code of 400, also known as a bad request. If the email address is available for use, it will create a new user based on the body from the POST request. During the creation operation, the server will check if the given email is from a pre-approved domain. If this is true, a token will be generated and a confirmation email will be sent with a link to the confirmation page with the token in the link. If the newly created user is not from a pre-approved domain, an admin has to manually go into the user table in order to send this user a confirmation email. To better get an overview of the process once the request is received by the server, see Figure 21.

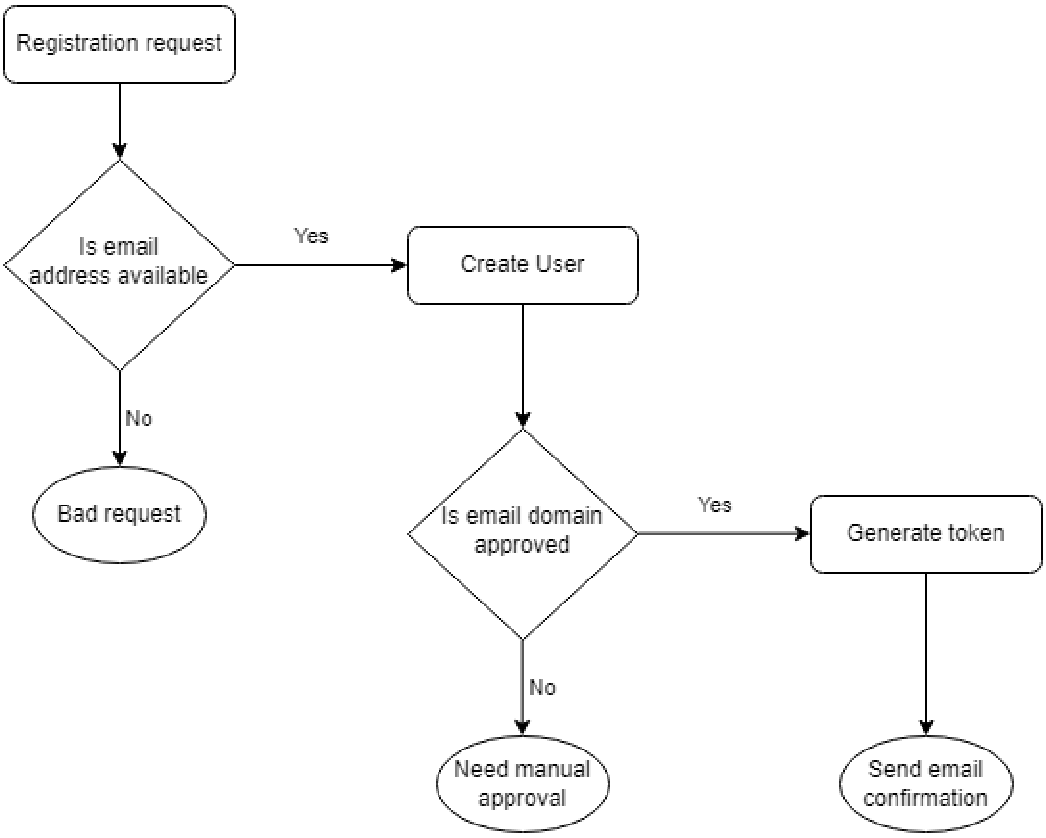


Figure 21: Activity diagram of when a registration request is received by the server.

3.1.6 Login

The user must enter their email, password, and click the login button. This will send a POST request to the server with the email and password in the body. Once this is received by the controller, it will try to generate a JSON Web Token (JWT), for more information about the creation and usage of JWT see Section 3.1.7.5. In the token generation function, user credentials must first be validated. First the server checks if a user with the given email exist in the database, if no user with the given email is found or the user have not been approved yet an empty user session will be returned resulting in a failed login attempt. Next comes checking the password, since we do not store the user's password in plain text, we must use the salt from the user object together with the given password and pass it through our hashing function. Then we will be able to compare the newly generated hashed password with the hashed password in the database. If these are equal, the server will generate the JWT and return a user session object which consist of five different attributes. These being the email, role, token, expires in, and expiry time stamp.

Once the user session object is received by the user's browser from the server, the application will call a function in our custom authentication state provider to update the authentication state of the user. This function creates a new claims principal with a claim's identity, containing a list of our claims, more on this in Section 3.1.7.5. This list contains claims of type role and name, the role is the role received from the database while the name is set to the identifier value which is the hashed value of the email. Next, the function will update the expiry time stamp by taking the current time and adding the number of seconds the session is valid. This value is also taken from the user session. To save the session information, it will be written to the browser's sessions storage, for more information about storage see Section 3.1.6.1. To avoid storing the user session in plain text, the user session object is first converted to bytes and then base 64 string. This makes the valued stored seem random, but we are still able to read the content later. With the user session information written to the session storage, the user will be redirected to the homepage and is now able to freely use all the pages they can access with their role.

3.1.6.1 Storage

There are two alternative storage options that can be used to keep track of each user's sessions: local storage or session storage. Session storage is in the browser and are unique for each tab and cleared when a session ends. While local storage store information on the local disk of the user and does not expire until the user clears it manually or the application clears it. The difference and the benefits and drawbacks are further discussed in Section 5.2.1. The application currently uses the session storage to read and write the user session information.

Information in session storage is stored as a key and value pair. For user session information, we used "UserSession" as the key and the converted user session object as the value. The user session is as mentioned earlier first serialized to a JSON item. The JSON item is then converted into bytes and then converted into base 64 string. Lastly, the base 64 string is written to the user's session storage. However, since we are writing directly to the user's sessions storage, they will be able to see the stored value if they go into either the developer tool in most browser or tries to write the session storage to the console. In Chrome, this can be done by first inspecting the page after logging in, then opening the console and write "sessionStorage", see Figure 23 for example. This would display the session storage to the user, and they would be able to see that the web application is storing one key-value pair with the key name "UserSession". Next, for the user to be able to get their user session, they will need to write "sessionStorage.getItem("UserSession")" in the console, see Figure 24 for example. This will write the user session value to the console. Do to this being available to the user, one could argue that this is a security flaw. Since the user can read the session storage, they will also be able to write to the storage with their own session information.

However, since the user session object have a timestamp, each user session written to the storage is only valid for a given amount of time. This means that even if the user can see their current user session, they will have very few ways to abuse it. So far, we are only aware of two abuse cases.

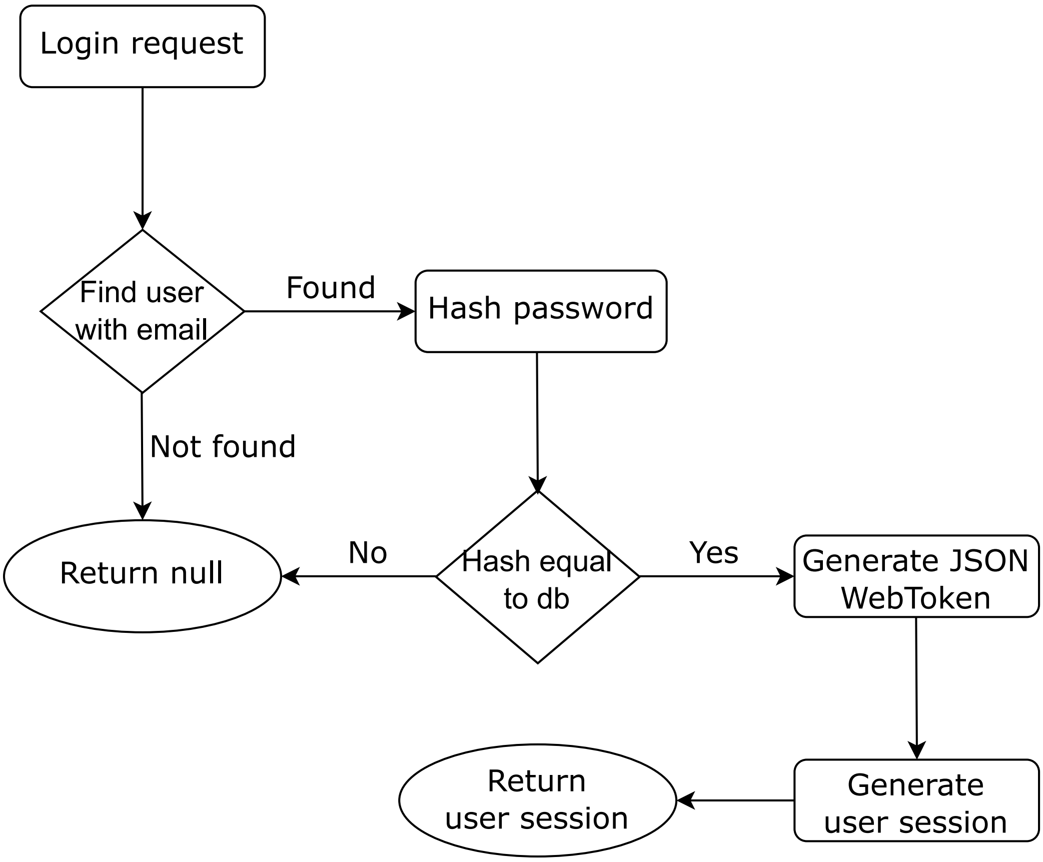
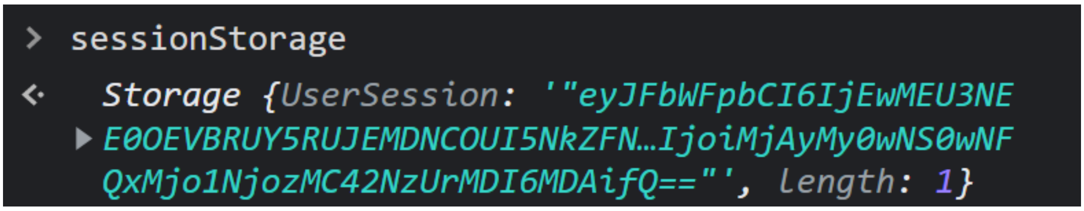


Figure 22: Activity diagram of when a login request is received by the server.



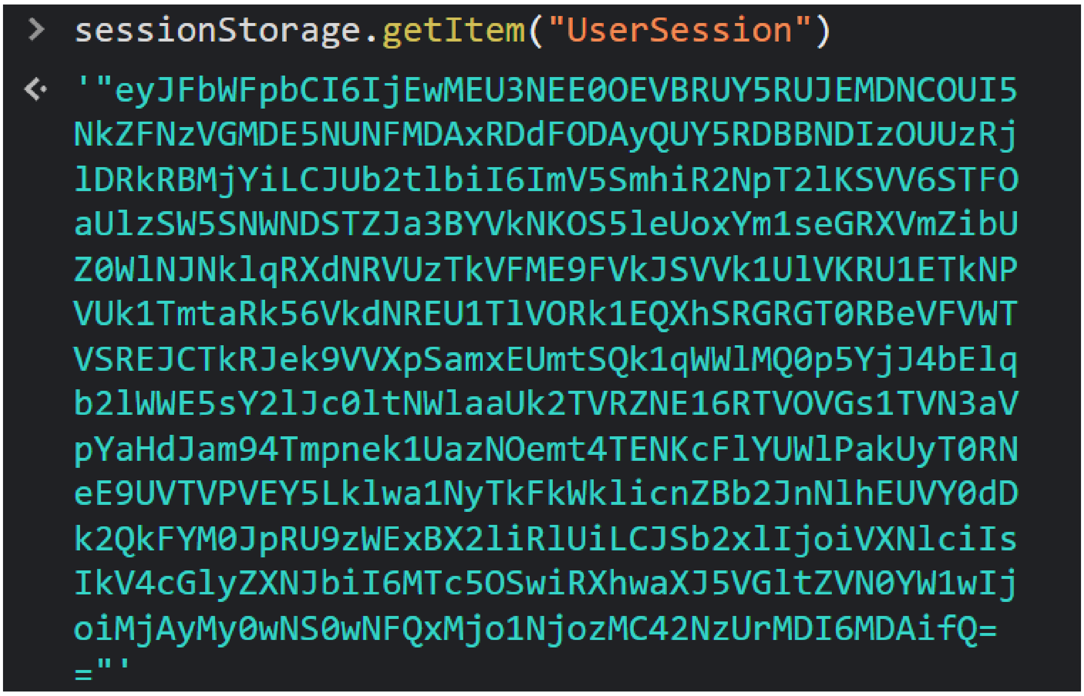
```
> sessionStorage
<·  Storage {UserSession: '"eyJFbWFpbCI6IjEwMEU3NE
  ▶ E0OEVBRUY5RUJEMDNCOUI5NkZFN…IjoiMjAyMy0wNS0wNF
    QxMjo1NjozMC42NzUrMDI6MDAifQ=="', length: 1}
```

Figure 23: Example output in console when checking session storage in browser.

One of which being that a user can log in, copy the value written to the session storage, delete their user. When deleting their user, they will be logged out, however, they will then be able to write the value they copied earlier, thus making them appear to be logged in. This will make them bypass the protection of pages and API endpoints, but since their user is deleted, they will not be able to create or delete projects. The other abuse case is that a valid user can copy their value from the session storage and send it to an unauthorized user, making them to bypass the login requirement. Due to the timestamp each abuse will only work for a predefined amount of time, if the timestamp is no longer valid, neither of these abuses will work.

3.1.7 Security Features

To protect sensitive information and prevent unauthorized access, we have implemented a variety of security features. Security features play a vital role in both safeguarding our web application as well ass protecting user data. By implementing the security feature talked about in this chapter, we have increased the confidentiality, integrity, and availability of both user data and the application itself. This also increases the trust and credibility of our platform. Security features covered in



```
> sessionStorage.getItem("UserSession")
<· '"eyJFbWFpbCI6IjEwMEU3NEE0OEVBRUY5RUJEMDNCOUI5
   NkZFNzVGMDE5NUNFMDAxRDdFODAyQUY5RDBBNDIzOUUzRj
   lDRkRBMjYiLCJUb2tlbiI6ImV5SmhiR2NpT2lKSVV6STFO
   aUlzSW5SNWNDSTZJa3BYVkNKOS5leUoxYm1seGRXVmZibU
   Z0WlNJNklqRXdNRVUzTkVFME9FVkJSVVk1UlVKRU1ETkNP
   VUk1TmtaRk56VkdNREU1TlVORk1EQXhSRGRGT0RBeVFVWT
   VSREJCTkRJek9VVXpSamxEUmtSQk1qWWlMQ0p5YjJ4bElq
   b2lWWE5sY2lJc0ltNWlaaUk2TVRZNE16RTVOVGs1TVN3aV
   pYaHdJam94Tmpnek1UazNOemt4TENKcFlYUWlPakUyT0RN
   eE9UVTVPVEY5Lklwa1NyTkFkWklicnZBb2JnNlhEUVY0dD
   k2QkFYM0JpRU9zWExBX2liRlUiLCJSb2xlIjoiVXNlciIs
   IkV4cGlyZXNJbiI6MTc5OSwiRXhwaXJ5VGltZVN0YW1wIj
   oiMjAyMy0wNS0wNFQxMjo1NjozMC42NzUrMDI6MDAifQ=
   ="'
```

Figure 24: Example output in console from session storage when using the key “UserSession”.

this chapter are Token, Hashing, Encryption, Role based authorization and JSON web token.

3.1.7.1 Token

As previously mentioned, the database has a dedicated token table with the attributes; Id, UserId, TokenValue, and CreatedAt. The ID attribute is the primary key and is a unique int, while UserId is a foreign key, meaning this attribute is linked to the User table. This means that a token is created for a specific user. The TokenValue attribute is of type nvarchar(32), which is a string with max length of 32 characters, and is randomly generated by the server. CreatedAt is a timestamp of when the token was created, for now each token is valid for 30 minutes after is creation, after this the token will be invalid, and the server will be unable to use this token for any operations.

The tokens have two important usages, one of which is in the email confirmation. When a user is created or manually approved by an admin, a token is created and is added to the confirmation URL, which is sent to the email address the user registered with. This is done in order to ensure that the newly registered user actually has access to the given email address. When the user uses the link received in the email, the server will take the token from the URL and check if the given token is valid or not. If the token is valid, the server will change the role of the user from “Not approved” to “User”, allowing the user to log in and use the application.

The other usage of tokens is for resetting the password, this is done in two steps. First, the user will need to enter their email address and send a request to the server from the forgot password page. The server will then check if the database contains any users with the given email address, if so, the server will generate a new token and assign it to the user. The token will then be included in the URL as a link to the reset password page. This is where step two begins. The user now must go to the URL received in the email in order to type in their new password, but first the server will validate the token once it receives a request. If the token is valid, the user will be able to type in their new password and send it to the server, which will change the password stored in the database (See Figure 25).

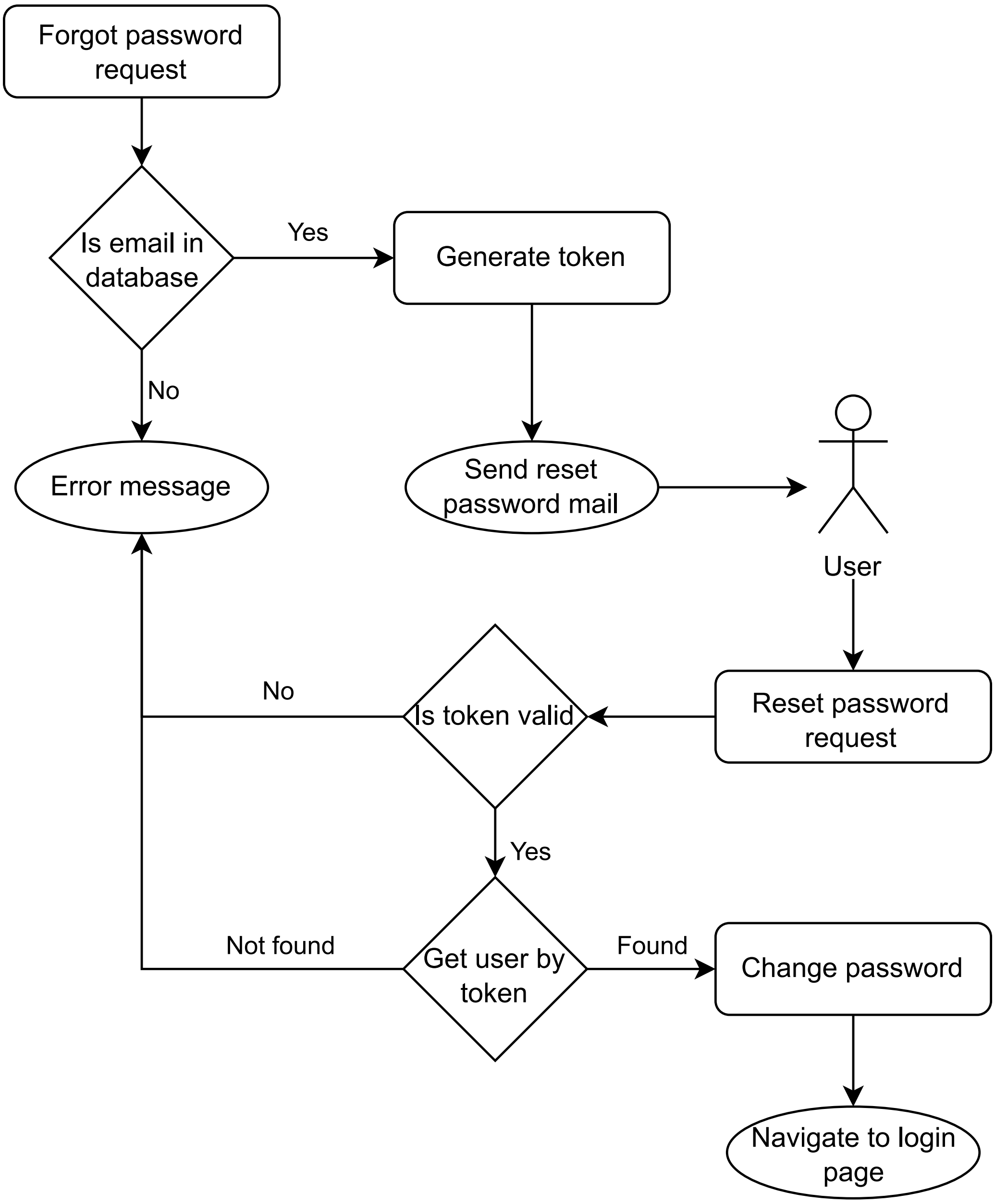
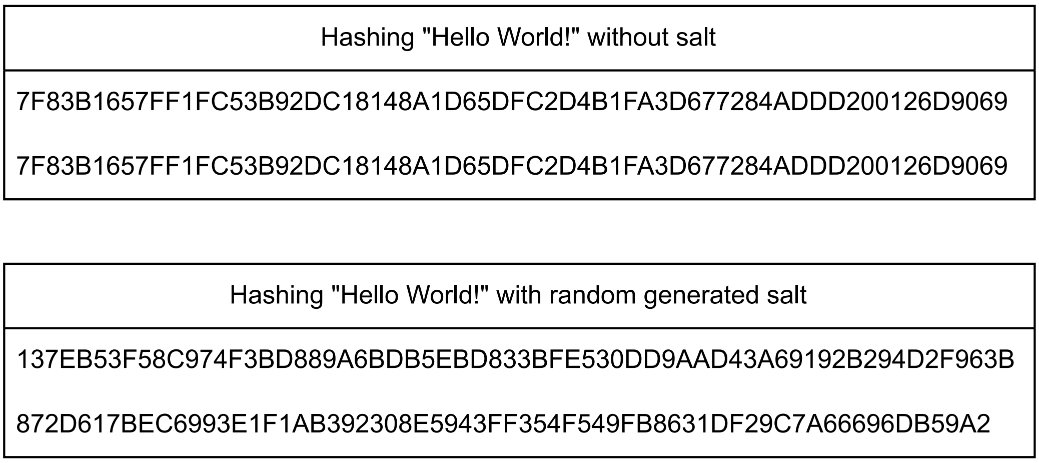


Figure 25: Activity diagram of when a forgot password request is sent to the server.

3.1.7.2 Hashing

In order to store the password of each user safely, we implemented hashing of passwords. Hashing is used to transform a string of variable length to what seems like a random string, usually of a fixed size. However, it is possible to use hashing function which produce output of variable length. In the context of databases, hashing is used to avoid storing information in plaintext. We decided on using the SHA-256 with salt. SHA-256 produces an output of 256 bits or 32 bytes. In our database we are storing the hashed strings as hexadecimal, each byte gives two hexadecimal digits. Meaning, the string we store for password is always of length 64 with the current hashing function. Our implementation of the hashing makes it easy to change which hashing algorithm is used, should this be desired. Changing the hashing algorithm only requires one line of code to be changed. However, if the hashing algorithm is changed, every registered user would need to reset their password to update the hashed password stored in the database. The system admins would also need to change the max length allowed in the database if the new hashing function produces a longer output.

Salt is added to the password before the string is passed through the hashing function. This is done to produce a different output from the hashing function, even if the password string is the same. The salt we have used is 32 randomly generated characters. In Figure 26 you can see two tables illustrating the different hashing output. The top table contains the output from the hashing function when we passed “Hello World!” as the password and the salt as an empty string. As you can see, the two outputs are equal. While in the table below, we passed through the same “Hello World!” string, but this time with a random generated salt for each. This makes the two outputs not equal even though we passed the same password string through the function. By using salt when hashing the password, we add more randomness and complexity to the result. This enables us to have the same password stored as different strings, so anyone that has access to the hashed password values will not be able to abuse their access by comparing different hashed values to guess the password of a different user.



Hashing "Hello World!" without salt
7F83B1657FF1FC53B92DC18148A1D65DFC2D4B1FA3D677284ADDD200126D9069
7F83B1657FF1FC53B92DC18148A1D65DFC2D4B1FA3D677284ADDD200126D9069

Hashing "Hello World!" with random generated salt
137EB53F58C974F3BD889A6BDB5EBD833BFE530DD9AAD43A69192B294D2F963B
872D617BEC6993E1F1AB392308E5943FF354F549FB8631DF29C7A66696DB59A2

Figure 26: Example output from hashing function when passing the same string with and without salt.

We have also implemented the use of the hashing function in the user email table. We hash the email of the user to avoid storing it in plaintext. This is done without using salt, because we want to use the hashed email as a unique identifier. By not using salt, we guarantee that the same input will result in the same output. With salt, the same input would result in a different output, allowing the user to create multiple users with the same email. The reason for wanting a unique identifier is so that we can check if the given email address already exists in the database. This logic is used both when sending a login request and when creating a new user.

SHA-256 does not guarantee that the generated output is unique. If two different inputs result in the exact same output when using a hashing algorithm, it is known as a hashing collision. This could be a problem when the result of the hash is used as a unique identifier. Even though this is a known issue, the probability of this happening is so incredibly low that it is a theoretical issue. Common practice is to ignore it as there are no known solutions and is a risk that comes with using SHA-256. In our case we use SHA-256 to hash the email when a user registers, this is done without salt. If another user tries to register, and a hashing collision happens, the second user trying to register will get a message telling them they are unable to register with this information, as the generated hash already exists in the database.

3.1.7.3 Encryption

Our encryption implementation is based on a tutorial by David Tavarez, an open-source advocate who primarily codes in Python, C#, PHP, Java, JavaScript, and C/C++ [22]. The tutorial uses the Portable.BouncyCastle package instead of the standard Microsoft ECDiffieHellmanCng Class. This was necessary to avoid raising CA1416 issues, which occurs when a platform-specific API is used in a different platform context, potentially causing verification problems, or using unsupported APIs [23]. The tutorial primarily focused on key exchange between two parties, but due to time constraints, it was decided to modify it to only generate and store the shared key on the server. As a result, certain functions and logic had to be rewritten. Once the key generation feature was implemented, we created separate functions for encryption and decryption, following Microsoft's example [24].

Our encryption implementation employs the elliptic-curve Diffie-Hellman (ECDH) key exchange algorithm, a cryptographic protocol widely used for secure communication. Within this protocol, two parties engage in the generation or possession of a key pair, comprised of a private and a public key. The utilization of these key pairs enables the establishment of a shared key between the two parties. In our implementation, this shared key serves as the sole mechanism for encrypting and decrypting data through a symmetric-key cipher. It is worth noting that the elliptic-curve Diffie-Hellman algorithm permits the derivation of an additional key from the shared key; however, this particular functionality has not been integrated into our implementation. Symmetric algorithms use the same key to encrypt and decrypt, while asymmetric encryption uses the public key to encrypt and the private key to decrypt.

Due to time constraints, we decided to only implement encryption for the user's email addresses. This was done in order to protect our registered user's information stored in the database and provide a proof of concept for encryption. Ideally, we should also have implemented encryption in the communication between the server and the users. Currently, all information is sent to or from the server in plaintext as JSON, this is seen as bad practice and should be fixed in future development of the software. This is seen as bad practice as it is susceptible to a "man in the middle" attack. Man in the middle is an attack where someone is eavesdropping on the connection between the server and the client. This enables an attacker to see all communication between the parties, by encrypting the transmission this problem would be solved, as the transmission would be unreadable without decrypting.

To implement encryption of email address, we had to create a new table called UserEmail. This table holds only information about the user's email address. The tables have a unique ID attribute of type integer, which acts as the primary key. A foreign key attribute was added, which is the user ID of type integer, which references the ID attribute of a given user in the user table. This was done to connect an entry from the user table to an entry in the user email table. The relationship between these tables is one-to-one, since one user can only have one email address and one email can only have one user. To find a given encrypted email address, we also added an identifier, which is the hashed value of the user email without any salt. Next, we needed to add two attributes both are of type varbinary, meaning these values are stored as a byte array. One of these attributes are called EncryptedEmail which is the email encrypted, and the other is called Iv. Iv stands for initialization vector, which can be seen as the equivalent of salt in hashing. Without the correct initialization vector, we will be unable to decrypt any message.

3.1.7.4 Role Based Authorization

When a user is created they are assigned a role, this could be one of three roles. The first role, that is the default role, is "Not approved". This role is for the users that are not yet approved, and they have no access to any of the pages or any of the API endpoints. The second role is called "Users", this role is for normal users, and they have access to the calculator pages and the needed endpoints for the calculators. The Last role is the "Admin" role. This role has access to all the endpoints and pages, including the DBtool pages.

To implement the role system, we use Microsoft.AspNetCore.Authorization. This authorization system support both role based and policy based authorization. For our application, we only use role based authentication, but it is very simple to implement policies if wanted. We chose to use a role based system because some functionality should only be available to admin users, and the rest of the project should only be available to approved users and admins. When using the Microsoft.AspNetCore.Authorization you are able to make an entire page unavailable for users of the wrong role. It's also possible to require a given role for a part of a given page, for example on our sidebar the DB-tool option is only visible to admins.

3.1.7.5 JSON Web Token

To use the JSON Web Token (JWT), we have to use something called claim principal. One principal can be seen as the user itself and contains many identities. One identity can hold many claims, while one claims is just a key-value pair. To see how these relationships between these objects (see Figure 27). Our implementation of this logic uses only one identity and two claims (see Figure 28). The identity can be seen as the session, while the two claims are of type name and role. Both claims are defined by the Claims Type class, to see a full overview of every defined type see Microsoft documentation about this class [25]. The name is the user's email address after it has been hashed and the role value is either "Not approved", "User" or "Admin" depending on what the user is stored as in the database.

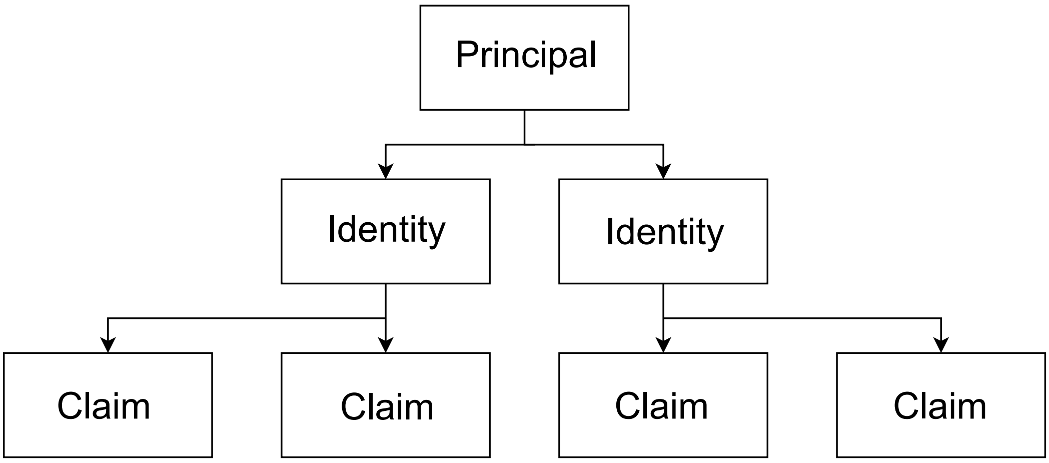


Figure 27: Relationship example between principal, identity, and claim.

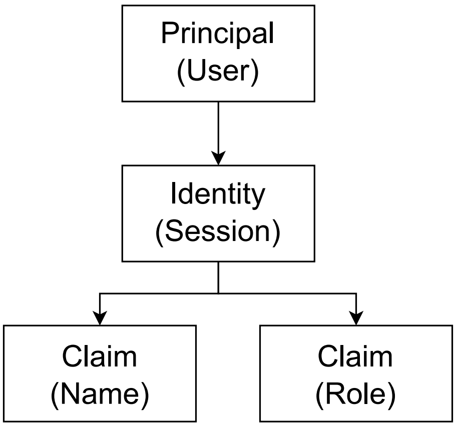


Figure 28: The relationship of our implementation of principal, identity, and claim.

To generate a JWT, a variable for when the token expires is first created. This variable takes the current data time and adds a predefined number of minutes, this is how long the token will be valid for. This value is now set to 30 minutes, but can be easily changed by change one constant in the code. Next a token key variable is created by converting a constant security key of type string which contains 40 predefined random characters into a byte array. This too can be easily changed, and should be changed in production to a longer string to increase the security of the tokens. Then a claim's identity is created. This identity contains a list of two claims, one being the name which is the hashed value of the user's email and the other being the role for the given user. Next the application creates signing credentials which uses the bytes array from the security token and what hashing function to use. Here once again we are using SHA256 function as the signature. Lastly, we need to create a security descriptor which contains the claim identity, when the token expires, and the signing credentials. With the security descriptor, the function can generate the JWT and then write the token to the user session object, which is what the generate token function returns.

There are two main uses for the token. First, it enables the application to check the role of a given user when they make a request to a page or an API endpoint, this is already mentioned in Section 3.1.7.4 about role-based authorization. The other usage of the token is to check if the current session this is valid. Once a request is made to a page, an on initialized functions is called, which first creates a custom authentication state provider object. This object is then used to call a function called get token. The get token function first reads from the session storage to get the user session object. Once the object is read, the next step is to check the timestamp of the given session. If the session is valid the token itself will be returned, if not an empty string will be returned. When the result is returned to the initialized function in the page, the function will check if the token gotten is empty or not. If the token is empty, the page will update the authentication state of the user to be a null value, thus logging the user out. If the token is not empty, the page will load in the needed resources.

3.2 Deployment

In this section, it will be explained the theory behind the application deployment, and how this was used.

3.2.1 Docker and Docker Hub Theory

Docker is a container virtualization technology and can be seen as a lightweight virtual machine. Docker consist of three main parts, Docker file, Docker image, and Docker container. A Docker file is a text document with a set of instructions for creating a Docker image, and contains information about the application and any needed dependencies. Docker image is as mentioned built from the Docker file and holds all needed components such as the application's code, rune time, and libraries. With the image, Docker can create a runnable instance of the image known as Docker container. The container is an isolated and portable environment that only runs the application and any needed dependencies. It is also possible to run multiple containers on a single host, where each container will have their own file system and network. Docker Hub is a registry service used to hold Docker images and is almost identical to the functionality that GitLab offers for coding projects. Images can be pushed and pull, making it easy for developers to access and updates their images [26].

3.2.2 Docker and Docker Hub usage

To host the application we used Docker, for more information about the theory of Docker see Section 3.2.1. Visual Studio has built in support for Docker. By right-clicking on the server solution, Visual Studio have an option called add Docker support. By clicking this option, a fully built Docker file is created. Theoretically, one could just use this docker file to build the needed image for the virtual machine. However, due to using a third-party package which requires a license, some changes were needed. The problem with the third-party package happens when building the image, and it fails to find the correct version for this package. This happens because the package comes from a private feeder rather than the public NuGet package feeder. In the public feeder, only a trial version of the package is available, which does not have the functionality needed. To get the licensed package to be included in the build, the package must first be downloaded. The next step is to add a NuGet configuration file. This file is used to add the local package to the Docker build. Once the downloaded package is included in the build, the build can be published.

To publish the Docker image, we once again relied on a built-in function in Visual Studio. This function can be access by right-clicking the server solution and clicking on publish. This is used to publish the Docker image to a given destination. We used Docker Hub to create a private repository, that only we have access to. By using all this combined, we can do changes in the code and publish the new Docker image to Docker Hub, all without leaving Visual Studio. Once the new version of the image has been pushed, the image can be pulled from Docker Hub to the virtual machine. Once the new image is downloaded to the virtual machine, a container can be started that will run the application with the new changes applied.

4 Testing And User Feedback

Testing is a vital part of software development, and it is essential to test your software before releasing it. Software testing is a huge domain, with many options and variables, and there are many ways to tackle this field. One of the ways this is done, and the way we chose to do it, is to use a stage based system that helps us define the scope of testing at a given point in development. In this stage based system, the early stages had a very limited scope in terms of testing, as the main focus is still on general development. As you move to the later stages testing becomes more and more the main focus and in the end the goal is to have a stable build with all the intended features.

4.1 Testing Methodologies

While we discuss relevant testing strategies in this chapter, we leave out testing strategies like pre-alpha, alpha and beta testing to the task provider (Barco). We assert these testing strategies will be effective from task provider when the product is ready to be launched.

4.1.1 Pre-alpha Stage

The first stage is the pre-alpha stage. In the pre-alpha stage, all the initial development is done, and no formal testing is completed. The goal for the pre-alpha stage is to get started with the project, and create a code-base that can be tested at a later stage.

4.1.2 Alpha Stage

In the Alpha stage, the initial features are implemented, so that some formal testing can be done. It is also quite common to do an early release, called an alpha-release, in this stage. For alpha-testing, it is common to mostly relied on simple forms of testing, such as smoke testing (see Section 4.2.4) and sanity testing (see Section 4.2.3), as the build often is not stable enough to do regression testing for the entire application.

4.1.3 Beta Stage

The next stage is called the Beta stage. This stage usually starts when most, or all, features are implemented, but bugs still exist. If your project is too advanced to be in the alpha-stage, but features are still being implemented, you can do a perpetual-beta. This is where you continually add new features without yet making a final stable build. During the beta-stage, it is also common to do a beta-release. There are two type of beta-releases, open and closed. An open-beta-release is open for everyone to try out, while a closed-beta-release is only for a select few.

In the beta stage, the build is usually stable enough to use more advanced types of testing, such as the ones outlined in Section 4.2.

4.2 Code Testing

An important part of software testing, is testing the code itself. This is done to resolve potential bugs, and to improve the effectiveness of the code. When testing the code, several techniques and approaches can be employed to ensure its quality and functionality, the ones we have used are presented below.

4.2.1 Unit Testing

Unit testing is a software testing technique that focuses on testing individual units or components of software. These units are tested in isolation, to ensure that each separate unit behaves as expected and functions correctly. Unit testing is performed by creating test cases that cover various scenarios and input values. These test cases are designed to target specific functionalities or behaviors of the unit and validate its outputs against expected results [27].

Unit testing was not implemented for our project, this will be discussed in Section 5.3.2.

4.2.2 Regression Testing

Regression testing consists of rerunning tests after code changes to make sure features still works as intended. These code changes can be new implementations, bug fixes or refactoring [28]. After these types of changes, it is very important to verify that everything still works. To do this, several methods which can be found in the section below were used.

4.2.3 Sanity Testing

In the stages where regression testing of the whole application were not possible, or very unpractical, sanity testing was used. Sanity testing is a subdomain of regression testing that you can use when you have a somewhat stable build, or relatively small changes were done since the last iteration. Sanity testing consists of testing only the part you changed since the last build, to see if the bug fix or feature worked as intended. This is done, so you can quickly reject the build if the fix or feature does not work, or accept the build if it works and continue with further regression testing [29].

4.2.4 Smoke Testing

Smoke testing is also a subdomain of regression testing, and is often done in combination with other types of testing. The point of smoke testing is to check the stability of a build. As mentioned, this is often done in combination with other types of testing, as when you are doing other types of testing, your application would have to pass a basic smoke test first. Smoke testing has varying levels of depth, but in its simplest form it consists of running the application, and seeing if it crashes by doing simple tasks. This is very similar to sanity testing, but here the stability of the build is tested, rather than the rationality of the result [29].

4.2.5 Black-box Testing

Black-box testing is a method of evaluating the performance and behavior of software where the tester only has access to the input and output fields. The tester should not have any knowledge about the inner workings of how the software calculates its output [30]. To do this, we tested each other's code, so that we would not have knowledge about the specific implementation.

4.2.6 Random Testing

Random testing consists of testing your code with random inputs, either until you crash the application, find a bug, or until you believe you have found all the errors this method can find. This method of testing is often categorized as a subdomain of black-box testing, as the tester does not need knowledge about the implementation. To perform this type of test, we simply pressed random keys when entering inputs, as this would give us random enough results.

4.2.7 White-box Testing

White-box testing is similar to black-box testing, but in this scenario, the tester has knowledge about the internal structure and implementation. This can make it easier to find bugs, as you often have some idea of what can break [30]. The team member that implemented a piece of code also tested the functionality of that code, and would be a way to perform this type of testing.

4.2.8 Output Testing

In addition to the black- and white-box testing, the outputs of the different methods were tested, to make sure they were the correct and expected values. This was done by logging the value and comparing it to the expected value. There are some cases where different inputs give the same output, and it is therefore not always enough to check the final output to make sure everything works as intended. This type of testing is very similar to unit testing, the main difference is that this is not automated [31].

4.2.9 Stress Testing

Stress testing consists of putting a program under higher demand than you would normally expect, to see how it reacts under high load. The goal of this test is to make sure the application does not crash when under high demand, and it can also help uncover bottlenecks [32]. We decided to not perform this type of testing, as it is highly dependent on the hardware you run the application on, and do not know what type of hardware Barco intend to use to host the server for this application.

4.2.10 Security Testing

Security testing is the process of trying to find flaws or issues with the implementation of security features. This could result in gaining access to protected functionality, such as gaining admin rights or removing data. Because security flaws can have such big implications, it is very important to strictly test your software security before releasing it. Software can never be completely safe, but it can be made in such a way that the risks are minimized. It is also important to mitigate the security flaws, so that if they are abused the damage is limited.

Our security testing consisted of attempting to perform SQL injection, trying to gain access to protected functionality without permission, and trying to abuse features like resetting password. In addition to this, a student with relevant competence, that is not involved in our thesis, was asked to try to find security flaws in our system. He does not have any prior knowledge of the implementation of the security features, and therefore does not have any preconceived notions of how to abuse the system. This student helped us identify a flaw in the implementation of session storage, which could then be fixed.

4.2.11 Compatibility Testing

Compatibility testing consists of testing if the application works on different platforms. In our case, this consists of testing different browsers and operating systems to see if the application still functions as intended. According to statcounter.com [33] the most used desktop operating systems are Windows, macOS and Linux, this is also supported by several other sources. Because windows, macOS, and Linux were by far the most popular, we decided to test our application on these three operating systems. Statcounter.com also lists chrome, safari, Microsoft Edge, and Firefox as the most used browsers, and therefore these were chosen to be tested.

To test on the different platforms, the same types of tests used before were re-used. This testing was not quite as comprehensive as the initial rounds of testing, and mainly relied on smoke-testing and sanity-testing. To perform this testing, a dockerized version of the program was used. The testing was performed by attempting to use the program on the different operating systems and browsers. From these tests, no issues were found on any of the platforms, and there were no significant differences between them.

4.3 User Testing

When developing software, developers get a good grasp of how the application behaves and how it is supposed to work. This makes it hard to notice aspects of the application that are unintuitive as a user. By watching users interact with the application for the first time, the developer can get new insight into the usability of the application. Additionally, the users will provide feedback on what they think should be done to the application.

4.3.1 Usability testing

Usability testing is an important part of testing, which aims to make the application more user-friendly, and easy to use. To do this, you have to gather feedback from test subjects, and then use the feedback to improve your application.

4.3.1.1 Tasks for testing

To perform the usability testing, a list of tasks for the user to perform was created. This was done to get a grasp of what worked, and what did not. These tasks were as follows:

- Create a user
- log in with the admin user (admin@gmail.com, admin)
- Navigate to the user table
  - Approve the newly created user
- navigate to the projector table
  - Edit a projector
  - Add a projector
  - Delete a projector
- Go to the simple project calculator page
  - You have a screen that is 1 meter tall and 2 meters wide, your projector is 4 meters away from the screen, and you want the brightness to be 40 Foot/Lambert. find the best projector and lens combination.
  - Some questions about the visualization perhaps
- Go to the simple project calculator
  - You have a screen that has a 2-meter diagonal and has a 16:9 aspect ratio. your projector is 3 meters away from the screen, and you want the brightness to be 43 Foot/Lambert. your screen only reflects 90 percent of the light that hits it. find the best projector and lens combination.
  - save your project
  - load a project
  - delete a project
  - Some questions about the visualization, perhaps
- reset the password of the user you created.
- login to the user you created
- delete the user you created.

4.3.1.2 Performing Usability Testing

In the Usability-testing, the test subjects were asked to perform the task given above, and then rate the difficulty on a scale of 1-10, with 1/10 being ”very hard”, and 10/10 being ”very easy”. Reviewing these scores help us gain insight into what features were intuitive to use, and what features that were difficult for new users to understand. By using the insights gained from these Usability-tests, we were able to improve the usability of our product, and make it more user-friendly. It was decided to conduct two rounds of testing, with the intention of improving any areas that proved challenging for users, in between the rounds. Each round of testing was conducted with ten distinct users.

During the process of selecting participants for the Usability-tests, a deliberate effort was made to ensure a diverse range of subjects with varying levels of relevant competence and knowledge. This included, but were not limited to Barco employees, engineering students, engineers, and people with little to no technical insights. By selecting subjects of such varying prior knowledge and competence, we insured that our product was easy to use for all types of users, and that little to no prior knowledge was needed.

The averages and medians from round one can be seen in Figure 29.

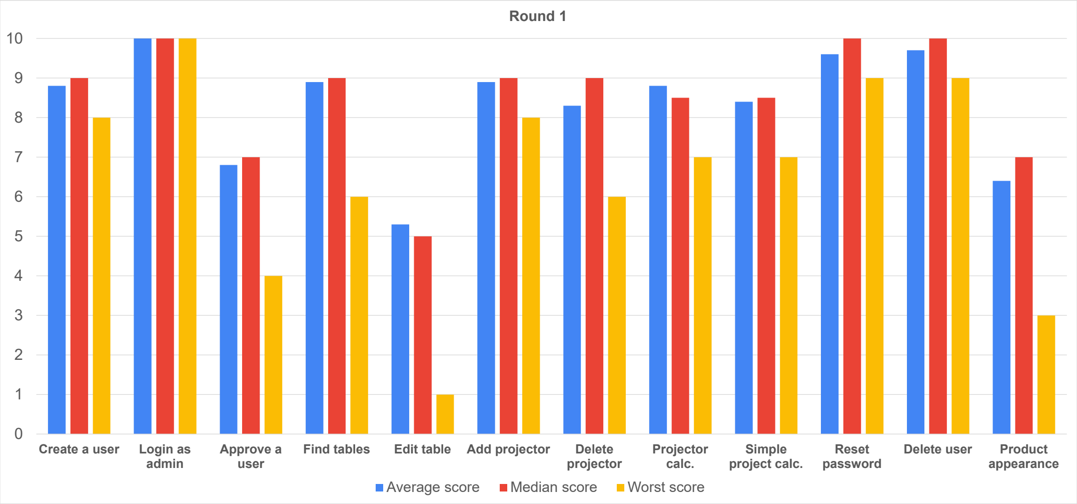


Figure 29: Averages, medians and max values from the first round of user testing.

After the first round of testing it was clear that the way users edit the tables needed to be improved, this question scored a difficulty of 5.3 / 10, which was significantly worse than any other question. Users also found it challenging to approve a user, with a score of 6.8 / 10. The users also gave a lot of verbal feedback on things they felt were missing, but was not covered by the questions we used.

For the second round of testing, a number of issues that users provided verbal feedback on were addressed. The majority of these issues were minor issues such as phrasing or slight visual improvements. In addition to these minor changes, the way a user would edit a table was changed.

The averages and medians from round two can be seen in Figure 30

The change in score from the first round to the second round can be seen in Figure 31

The improvement to the editing feature was clear to see in the scores given, as the average difficulty changed by 1.7, resulting in a new average score of 7/10. The median score also changed by 2.5, indicating that it was not just an outlier that impacted the average in such a major way. During the first round of testing, the scores had a very even distribution, this changed in round two, as most of the scores were the best alternatives. This can be seen in Figure 32.

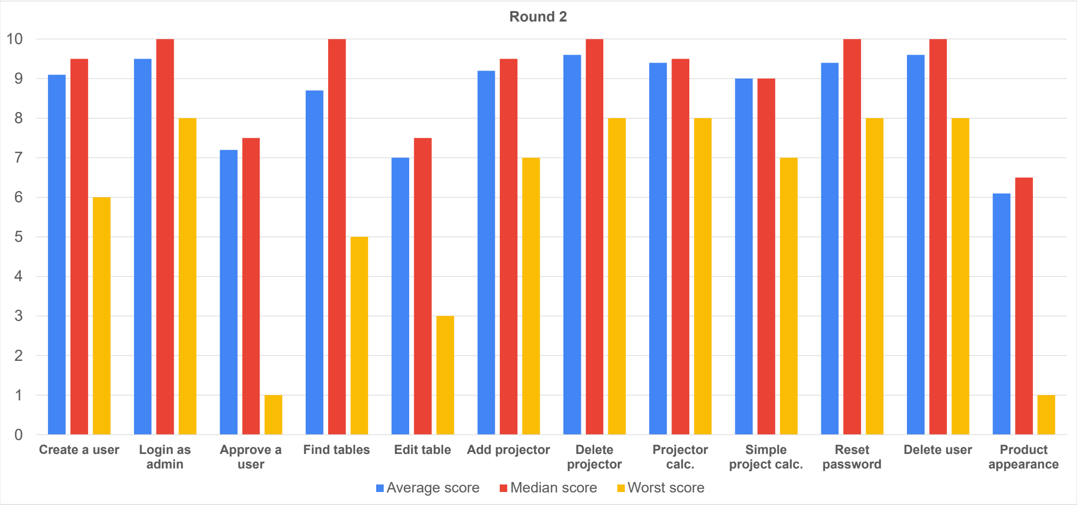


Figure 30: Averages, medians and max values from the second round of user testing.

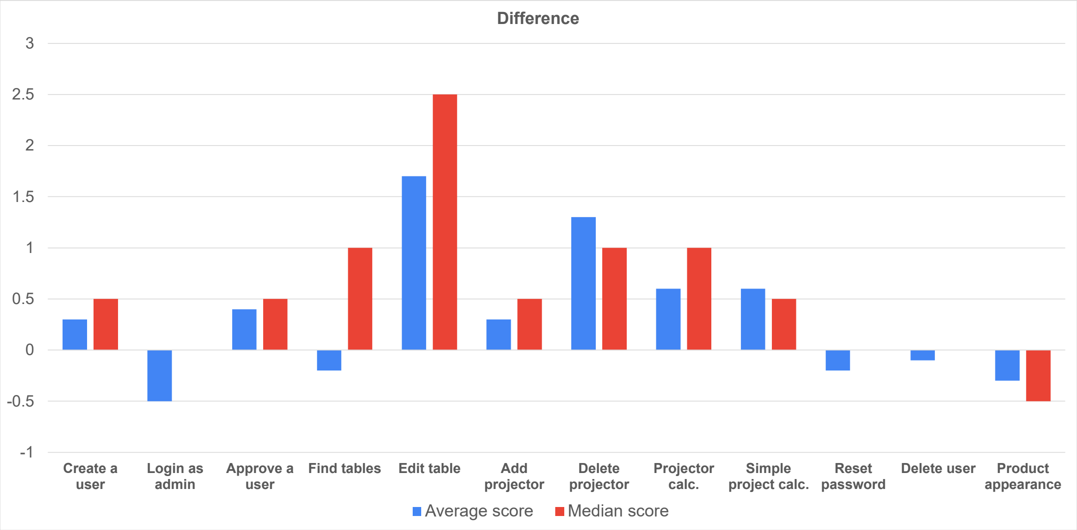


Figure 31: Averages, medians and max values from the second round of user testing.

There was also a significant change in difficulty score for deleting a projector. In the first round users on average scored this an 8.3/10, and in the second round this became a 9.6/10, resulting in a change of 1.3. Even though the average score changed by a significant margin the median only changed by one, this could indicate that the average was heavily impacted by an outlier. When we look at the numbers we can see that two of the initial ten users scored this a 6/10 while all the other users from both rounds scored it between 8-10, we can also see that more users scored it a 10/10 in round two. This can be seen in Figure 33.

After testing was concluded, it was clear that the sample size should have been bigger. With such a small sample size, every single user can have a large impact on the average score, and it is therefore difficult to get accurate and consistent results. During the first round of testing, the users gave a lot of feedback. In the second round the users gave noticeably less verbal feedback, but it is hard to know if this is down to the specific users that were tested with, or if the application was improved in these areas. To improve this issue, the questionnaire used should have been more detailed, but this would also result in increased difficulty in finding users to test with, as this would significantly increase the testing time per user.

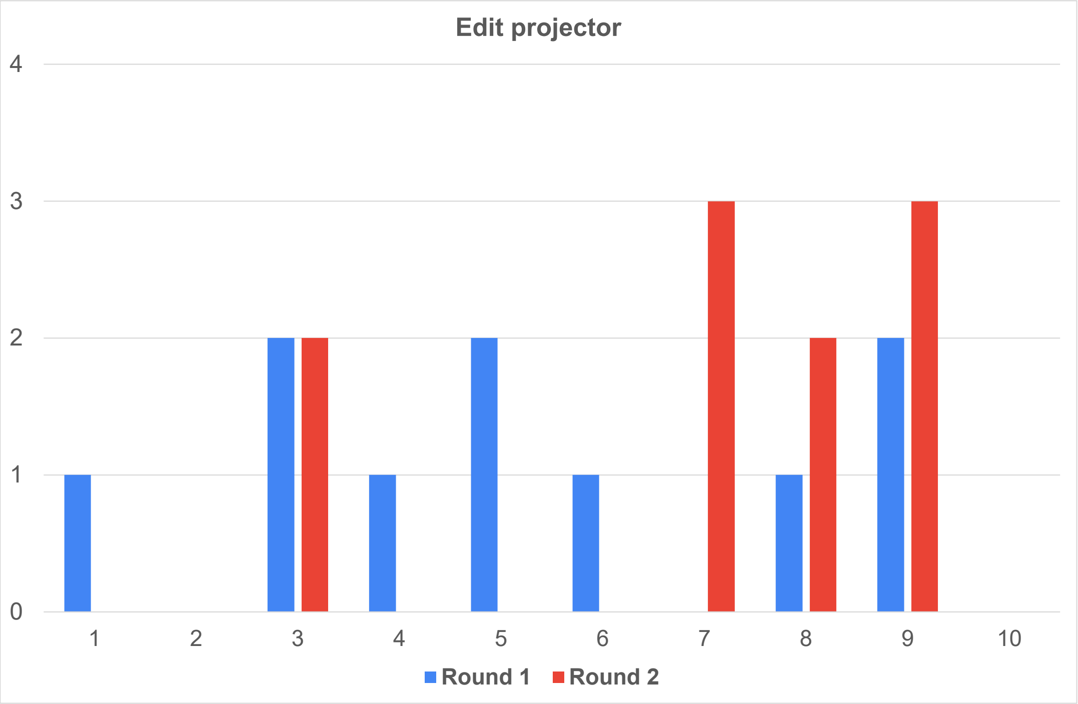


Figure 32: Chart showing the distribution of scores in both rounds of testing.

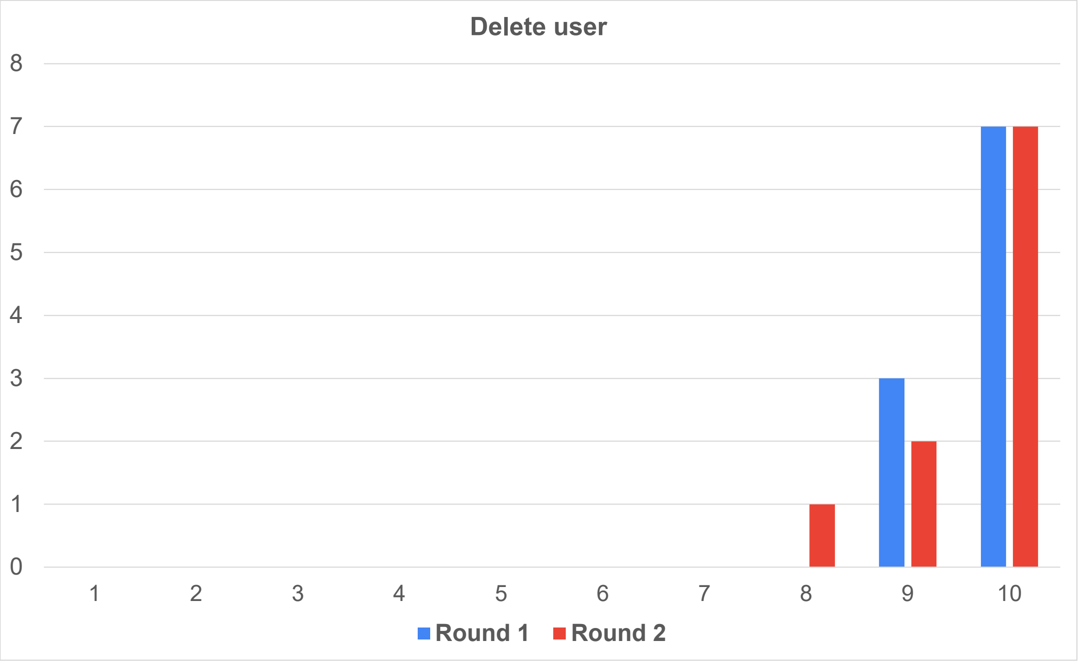


Figure 33: Chart showing the distribution of scores in both rounds of testing.

5 Discussion

After completing a project of this scale, it is important to discuss the choices that were made, and if anything could have been done differently. Not only does this aid in setting a direction for future works, but it also helps the team reflect around the choices that were made, and therefore presents a great learning opportunity.

5.1 Database Design

When it comes to the design of the ProjectorLens database, which holds information about projectors and lenses, the ProjectorLens table might not be needed. This table is currently used to make a projector and lens pair to showcase which lenses can be paired with which projector. But the current implementation creates a new lens entry for each model based on what projector it is used with. This means that the same lens has multiple entries, because some attributes are dependent on what projector it is paired with. This makes the relationship between Lens table and ProjectorLens table a one-to-one connection, since one lens can only be paired with one projector. Given this logic, the ProjectorLens table could be replaced in its entirety and the foreign key connection from the ProjectorLens table could be moved into the Lens table. To move the foreign key connection, a new attribute would be added in the lens table called projectorId, which would represent which projector the given lens can be used with.

On the other hand, the current design creates a better separation of responsibilities, where projector and lens tables hold just information about their respective products, while only the ProjectorLens table takes care of the pairing. Another advantage with the current design is that it allows for creation of new lens entries without having to define what projector the lens belongs to. If the proposed changes were made, the foreign key relationship would either have to allow null, allow the relationship to be defined later or just use a dummy value. For the dummy value solution to work, a projector entry would be needed in the projector table. The reason for this is that we need to connect the projector ID foreign key to an actual ID entry in the projector table.

The proposed changes to the database would decrease the number of tables and could possibly make the querying, especially of the inner-joins queries, more effective. An inner-join in SQL is used to get combined to tables and find records having matching value in both tables, see Figure 34 for illustration. Due to the limited amount of data, this would not necessarily decrease the query time by a significant amount. With the current dataset, the database contains in total 81 projector/lens pairs, making the query effectiveness a relatively small time save. The suggested changes could potentially make it more intuitive what projector can be used with what lens. Which solutions are more intuitive comes down to personal preference and understanding. From the groups' perspective, the current solution is the best of both worlds, where it is relative understandable which projector is paired with which lens, while having good separation of responsibilities.

The entries in the ProjectorLens table currently only displays the ID of projectors and lenses, which makes the table difficult to read and understand. The ProjectorLens table can be seen in Figure 35. An easy improvement to the usability of this table would be to display the model names instead. However, this seemingly small change would require quite a bit of code. We would need to first create a new model class for the entries in the table with the associated JSON properties, then a new API endpoint would need to be created in the DatabaseController class. This endpoint would call another new function in the DataAccessSqlProvider class that would send an SQL query to get the fields associated with the new model, and then return these back to the table.

This issue was not uncovered during the usability testing of the application and was not considered until the later stages of the thesis. The main purpose of the ProjectorLens table was to add the connection between projectors and lenses, which is done through selecting projectors and lenses based on their model names. This means that the usability issue is only relevant when it comes to managing the connections and deleting them. Since this is not the main purpose of the table and the issue was discovered very late in the thesis, fixing the issue was not prioritized.

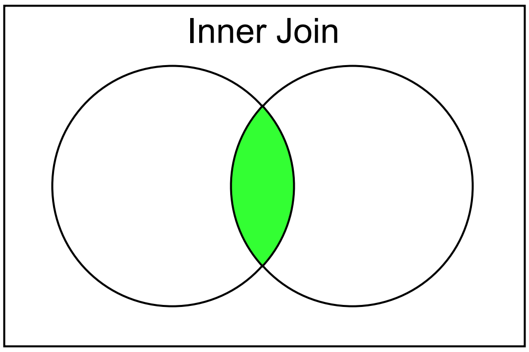
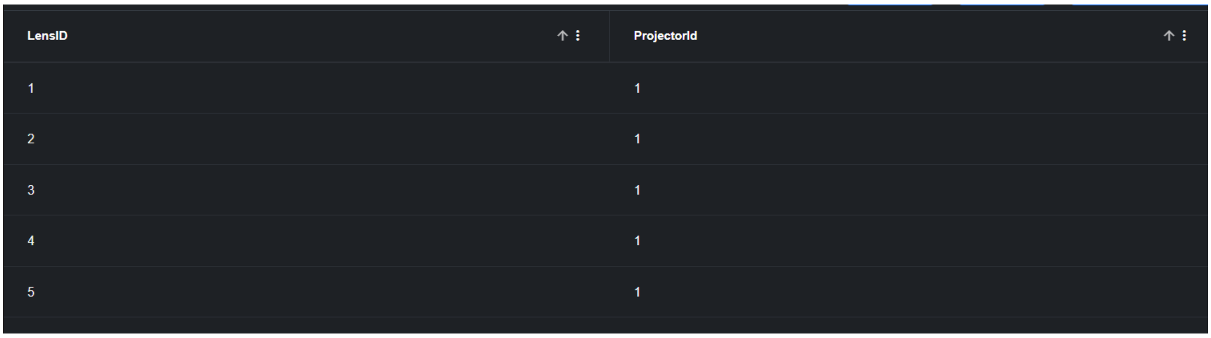


Figure 34: Inner Join, where each circle is a database table and the green is the result where records have matching values in both tables.

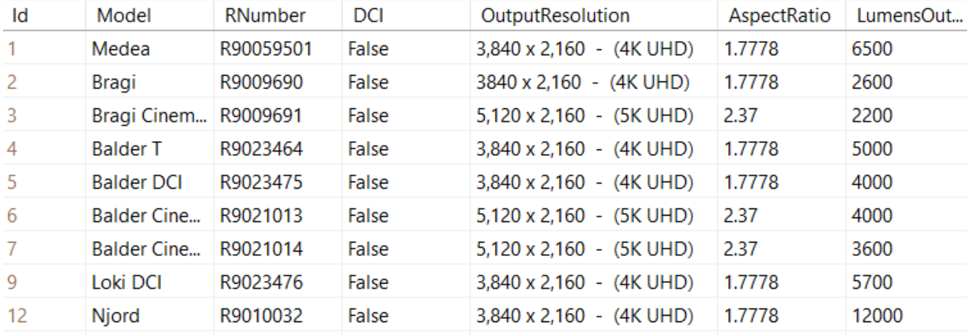


LensID	ProjectorId
1	1
2	1
3	1
4	1
5	1

Figure 35: Snippet of the ProjectorLens table.

In the user database, the focus for improvements is more productive if used on the actual tables itself rather than the connection between the tables. The reasoning for this is that in this database, all the connections are more forced and does not allow for the same number of different solutions to be explored. For example, the group decided to keep the email address attribute in the user table, even though this is a redundant field and is always set as null. The reasoning for keeping this field was that the implementation of encrypted email addresses happened rather late in the development process. Thus, the group encountered some problems when changes were made to the object model used for the user table. This happened because the user model is used for multiple purposes in the application.

The database does not follow the standard normal forms, which are a series of guidelines that ensures the database is efficient, organized, and free of data anomalies [34]. For the table to be in



Id	Model	RNumber	DCI	OutputResolution	AspectRatio	LumensOut...
1	Medea	R90059501	False	3,840 x 2,160 - (4K UHD)	1.7778	6500
2	Bragi	R9009690	False	3840 x 2,160 - (4K UHD)	1.7778	2600
3	Bragi Cinem...	R9009691	False	5,120 x 2,160 - (5K UHD)	2.37	2200
4	Balder T	R9023464	False	3,840 x 2,160 - (4K UHD)	1.7778	5000
5	Balder DCI	R9023475	False	3,840 x 2,160 - (4K UHD)	1.7778	4000
6	Balder Cine...	R9021013	False	5,120 x 2,160 - (5K UHD)	2.37	4000
7	Balder Cine...	R9021014	False	5,120 x 2,160 - (5K UHD)	2.37	3600
9	Loki DCI	R9023476	False	3,840 x 2,160 - (4K UHD)	1.7778	5700
12	Njord	R9010032	False	3,840 x 2,160 - (4K UHD)	1.7778	12000

Figure 36: Snippet of the Projector table.

the first normal form state (1NF), each table cell should contain only one value, and each column name should be unique. The Projector table, can be seen in Figure 36, has a column named OutputResolution, that contains the resolution and the name of the resolution, such as "4K UHD". Therefore, our Projector table is not in the first normal form. We would need to split the resolution and the name to make this column adhere to the 1NF guideline.

The second normal form (2NF) states that each column should only be dependent on the primary key, in this case the Id, and not other fields. In the Projector table, the AspectRatio column is dependent on the OutputResolution column, since the aspect ratio is decided by the resolution. To change the Projector table into 2NF, we would need to create a new table that represents a resolution with aspect ratio and the resolution name, and then add a foreign key into the Projector table.

To convert to third normal form (3NF), no transitive functional dependencies need to be present. A transitive functional dependency is when a non column changes depending on another non-primary key column. This would not be a problem after moving resolution and aspect ratio into a table with resolution, as every column would only be dependent on the primary key.

Lastly, for Boyce-Codd normal form (BCNF), only the primary key can be a candidate key. A candidate key is a column that is unique, meaning that the column could serve as a primary key. In the Projector table, the RNumber column could serve as a primary key since it is unique. BCNF would require that RNumber is changed to be the primary key instead of Id.

The Lens table has similar issues to the Projector table, however in this table the RNumber could not be a primary key as the same lens is stored more than once with slightly different values based off which projector it is paired with. Since the RNumber is unique to the lens, this would result in the same RNumber being stored multiple times. To convert this table to BCNF, we would need to fix the previously mentioned issue of duplicate entries and perform the same steps as in the Projector table.

Many of the database columns were specified by Barco in the project requirements, and the data provided to us were in a similar format from their existing solution. Converting to BCNF would also complicate the queries required to pull data from the database, making it harder for Barco to read and understand our thought process when they eventually continue development. The main reason for converting tables to these normal forms is for efficiency, and since our database store small amounts of data, this efficiency increase will give negligible benefits. Because of this, we decided that converting the database to BCNF was not a priority, but should be a consideration if Barco expands the database and adds many more projectors and lenses.

5.2 Code Design

Code design refers to the process of planning, organizing, and structuring software code in a way that makes it easy to understand, modify, and maintain. Good code design is essential for building software that is reliable and efficient. This section will be discussing some decision that were made about code design and alternative solutions.

5.2.1 Storage

There are two alternative storage options that can be used to store the session information for each user: local storage or session storage. Currently, the application uses the session storage to read and write session information. With the use of session storage, some drawbacks have been discovered, such as the user must log in for each tab they use. Since the session storage is unique for each tab, the user will be counted as a new user whenever they open a new tab. This is a setback which could be avoided with the use of local storage. Local storage is not unique for each tab, thus making it possible to get the session information from multiple tabs and bypassing the login requirement for each tab if the stored session is still valid. However, the local storage is usually not used to store session information. Local storage is downloaded to the user's local disk, meaning they will have access to the files this information would have been written to. Local storage is usually used to store unimportant long-term information such as preferences or settings.

One of the advantages of session storage over local storage is that session storage is automatically cleared whenever a session ends. With local storage, this is a functionality which needs implementation, or the user could do it manually. Local storage has more capacity in terms of storage than session storage, however, this is not a valid reason to change, since the amount of information the application needs to store is small. It is seen as standard practice to use session storage for information which is temporary and will change in the future. This relates to application requirements, since the user session contains a timestamp, the session is only valid for a specific period of time.

5.2.2 API Flaws and Improvements

For the server side of the application, we have implemented a controller-service model. This implementation separates responsibilities. Every controller has the responsibility of receiving the request itself and to take out any passed parameters from the request. The controllers then use the service classes to call the function which does the actual operations, such as adding or finding a user in the database. Passed parameters are taken out from the request in two ways, it is either a part of the URL or in the body. Whenever a GET request is sent, the parameter will always be a part of the URL, because a GET request does not send any content in the body of the request and is primarily used to get data from the server. While a POST request is used to send data to the server. In some rare cases, a combination of these possibilities is used. For example, when the user saves a project, all the needed information about the project itself is stored in an object and added to the body of the request. The server still needs to know which user sent the request to set the correct owner of the project. In the given example, this is done by adding the identifier of the user, which is the user's hashed email address, in the URL.

This can also be seen as one of the drawbacks with using a framework like Entity Framework when the reliance on model classes is so strong. Since the defined object model is used to define both column names and data type, this model cannot be easily changed to hold an extra value. The need for an extra attribute is linked to when the client wants to pass information from their browser to the server to update the database. As mentioned in the chapter above, the save project functionality uses a model to send all needed information about the project in the body of a POST request. However, the model does not contain a field for username, only for user ID. The problem is that the user does not have access to their own ID, and the server still needs to know which user sent a project to set the correct owner. This is a minor issue, as a workaround has been found and implemented, like mentioned in the paragraph above. Inheritance could have been used to extend a current model class to contain some extra attributes, which would also solve the problem. This solution could significantly increase the total number of models in the application if a new model was added every time the application needed to send a model with some extra information, such as a token or identifier.

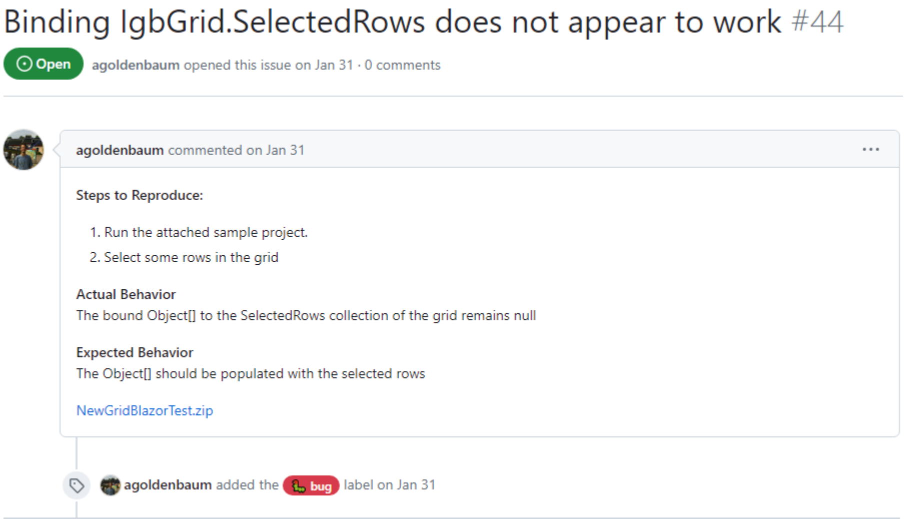
Due to time limitations, and the close connection between the service and controller classes, the focus on REST API principals have been lacking, especially the principal of returning status codes. This makes it harder for other developers to intuitively understand some endpoints. For example, some endpoints are set to return void from the function, which can be seen as bad practice since the controller does not give any feedback about the success of the operation. A few of the current endpoints follows the principal of returning a status code, while others return objects, list of objects or boolean values.

5.2.3 Tables

To create the tables to represent the different entities in the database, an element called IgbGrid, from the library IgniteUI.Blazor, was used. This library was developed by Infragistics. Our group found this library difficult to work with, and it requires a license to use. Barco already had this license, and was the biggest reason for choosing to work with this library.

Initially, our group intended to use a different library, which seemed to work better than the

IgniteUI library, but because Barco already had a lisence for IgniteUI, it was decided that it should be used to facilitate further development. The reason we found the IgniteUI library difficult to work with, is that the customization and manipulation of the data in the table through code is very restricted and mainly done through certain events returned by functions. IgbGrid has a function called “SelectedRows” which should return a list of the selected objects in the table. This function does not work as it is supposed to and just returns null. The issue has been reported to the Infragistics developers, as can be seen in Figure 37. To work around this issue, we had to find the ID of the selected row and then save the selected ID in a string variable. If Barco did not have a license for this library, we would prefer to have used a different solution.


Binding IgbGrid.SelectedRows does not appear to work #44

Open agoldenbaum opened this issue on Jan 31 · 0 comments

agoldenbaum commented on Jan 31

Steps to Reproduce:

1. Run the attached sample project.
2. Select some rows in the grid

Actual Behavior
The bound Object[] to the SelectedRows collection of the grid remains null

Expected Behavior
The Object[] should be populated with the selected rows

NewGridBlazorTest.zip

agoldenbaum added the bug label on Jan 31

Figure 37: Open issue on the Infragistics GitHub page.

5.3 Testing

During our project, we performed several types of testing. These were both aimed at improving the code, and at improving the user experience. Even though the result of our testing was helpful, the process was not flawless. This will be further discussed in the following section.

5.3.1 Usability Testing

We would ideally have used a larger sample size during our usability testing. Most of the statistical issues that were encountered would have been resolved, or at least minimized, by increasing the sample size. The form that was used, should have been more comprehensive to get more reliable and in-depth data. The only reason we decided against this was that it was already difficult to find users to test our application, and this problem would become even more prominent if the time span of the test increased.

5.3.2 Code testing

In our code testing, we decided to not implement unit testing for our project. This decision was made in collaboration with Barco, as both our group and Barco thought it would be better if the time was spent developing the application, rather than making sure the application was completely bug free. Barco’s reasoning for this is that they intend to continue development of the application after our submission, and therefore want as much of the general development already implemented when they take over the development.

5.4 Future Works

This section will go through the group's thoughts on what is missing and what could be improved in terms of functionality and usability, such as missing features or design flaws.

5.4.1 Secure Communication

We would like to highlight the need to implement secure communication between the server and the client. A potential solution could be to use encryption, which is something that was considered at a later stage in the thesis, but ended up not being implemented due to the group prioritizing the development of the database tools and pages. Currently, the application sends information from the server to the client in plain text, this makes it possible for an attacker to perform a man in the middle attack, which is when someone is intercepting and listening to the communication between two parties. With an implementation of encryption, an attacker could still be able to read the communication, but the information would be protected by a layer of encryption. Thus, making the captured communication be nearly impossible to decrypt without the correct encryption key. This would not only increase the security level of the application, but also the confidentiality. We can never be certain that sensitive information is only shown to users who have authorization to see it without the usage of secure communication, which encryption could provide.

An encryption implementation would also require the implementation of key exchange. This would only lead to some changes in the current solution, since the application contains functionality which enables encryption of email addresses. One possibility is to use the existing encryption classes. The classes contain three key features at the moment: the ability to encrypt and decrypt messages and key generation. As mentioned in the implementation chapter about encryption, we are currently using a shared key to encrypt and decrypt. A solution could build on this logic, where each user would need to have a shared key with the server to enable both parties the ability to read and write messages. With such a system, the key management could be a vulnerability, and would need to be looked into. The user's keys could either be stored in a database and sent to the user once a session was started, or the user would have to download their key. If the key were to be downloaded to the user, some key management logic or security measures would be needed to protect the keys from falling into an unauthorized user's hands.

5.4.2 API

For the server side of the application, we have implemented a controller-service model. This implementation separates responsibilities. Each controller has the responsibility of receiving the request itself and to take out any passed parameters from the request. The controllers then use the service classes to call the function which does the actual operations, such as adding or finding a user in the database. Passed parameters are taken out from the request in two ways, it is either a part of the URL or in the body. Whenever a GET request is sent, the parameter will always be a part of the URL, because a GET request does not send any content in the body of the request and is primarily used to get data from the server. While a POST request is used to send data to the server. In some rare cases, a combination of these possibilities is used. For example, when the user saves a project, all the needed information about the project itself is stored in an object and added to the body of the request. The server still needs to know which user sent the request to set the correct owner of the project. In the given example, this is done by adding the identifier of the user, which is the user's hashed email address, in the URL.

This can also be seen as one of the drawbacks with using a framework like Entity framework when the reliance on model classes is so strong. Since the defined object model is used to define both column names and data type, this model cannot be easily changed to hold an extra value. The need for an extra attribute is linked to when the client wants to pass information from their browser to the server to update the database. As mentioned in the chapter above, the save project functionality uses a model to send all needed information about the project in the body of a POST request. However, the model does not contain a field for username, only for user ID. The problem is that the user does have access to their own ID, and the server still needs to know which user sent a project to set the correct owner. This is a minor issue, as a workaround has been found and implemented. A potential fix to this problem is to create new models which could build on existing models. A possible solution would be to increase the usage of inheritance of classes. This would allow us to expend a current model class to contain some extra attributes. This solution would drastically increase the total number of models if a new one was added every time the application needed to send some extra information, such as token or identifier.

Due to the close connection between the service and controller classes and time limitation, the focus on REST API principals have been lacking, especially the principal of return status codes. This makes it harder for other developers to intuitively understand some endpoints. For example, some endpoints are set to return void from the function, which can be seen as bad practice since the controller does not give any feedback about the success of the operation. A few of the current endpoints does follow the principal of returning a status code, while others return objects, list of objects or boolean value. This is something that should be focused on in future development, because status codes are helpful both for developers and users alike. With better status codes, the user will be able to describe any problems more accurately if they encounter any problems with receiving or adding information to the database.

5.4.3 Design

Due to the focus being on backend development and not design, the group feels there is room for improvement when it comes to design. The redesign should focus on improving usability and ease of use. An idea would be to continue or at least do more usability testing to get insight from users before committing to changes. As developers, it can often be hard to understand or predict what users will find intuitively, this is the point of usability testing. Ideally the total number of tests should be drastically increased to do actual statistics analysis which could prove to be a pointer in terms of increasing the usability. The process of doing tests on the design and then redesign or applying changes based on the feedback should over time improve the application, not only in terms of design, but potentially highlight missing features as well.

5.5 Development Process

With minor exceptions, each sprint of the process lasted two weeks. Later in the process, the group did feel that a switch to a different method was preferable. This decision was made on the basis that our daily tasks were highly variable, and the task sizes were small, but the overall number of tasks was large. From this, we concluded that the scrum format would not be productive for this type of work. As an alternative, we chose to switch to a more linear approach, and opted to use the waterfall model for the remaining period of development.

To keep track of the issues each member was working on, GitLab kanban board was used. Here members could create a new issue for each bug, missing feature, or other issues encountered. This allowed the group to work more dynamically and structured. When a member solved their current active issue, they could simply pick a new issue from the open section of the issue board. By using this system, communication about specifics and progress was improved, since every member could see what the other members were working on. In the kanban table we had four main labels, these were “Open” an issue not assigned, “Doing” an active issue, “For review” a resolved issue awaiting approval and “Closed” marking it as approved.

5.6 Comparison to Existing Solution

When compared to Barco’s already existing tool, it is difficult to assess which is better. With more time, the group would have liked to take usability tests on both solutions to get raw data about both tools. The tools they have used this far, lacks most of the backend features we have created, and it is difficult to manage the database, but the calculators themselves are more in depth and they have better graphical visualizations. Even though the existing calculators themselves are better at this point in time, our solution has a better backend codebase, and therefore it is easy to continue development to make it the superior tool.

In retrospect, the group would have started the overall development differently. The group thinks that usability testing should have been done on the existing solution to get data about the advantages and disadvantages of that application. This would have granted insights into the problems with the existing solutions. The new application could have been built on the insights gained from the testing, which could have resulted in a more user-friendly interface. However, making the backend robust and scalable for future development was prioritized over making the interface user-friendly.

5.7 Time Usage

According to NTNU's "for students" page, a course with 20 credits should equate to somewhere between 500 and 600 hours [35]. Our group, which consists of three people, should then have around 1500 to 1800 hours in total. Our hourly total ended up being in this range, and with all team members working a fairly even amount. This can be seen in Figure 38.

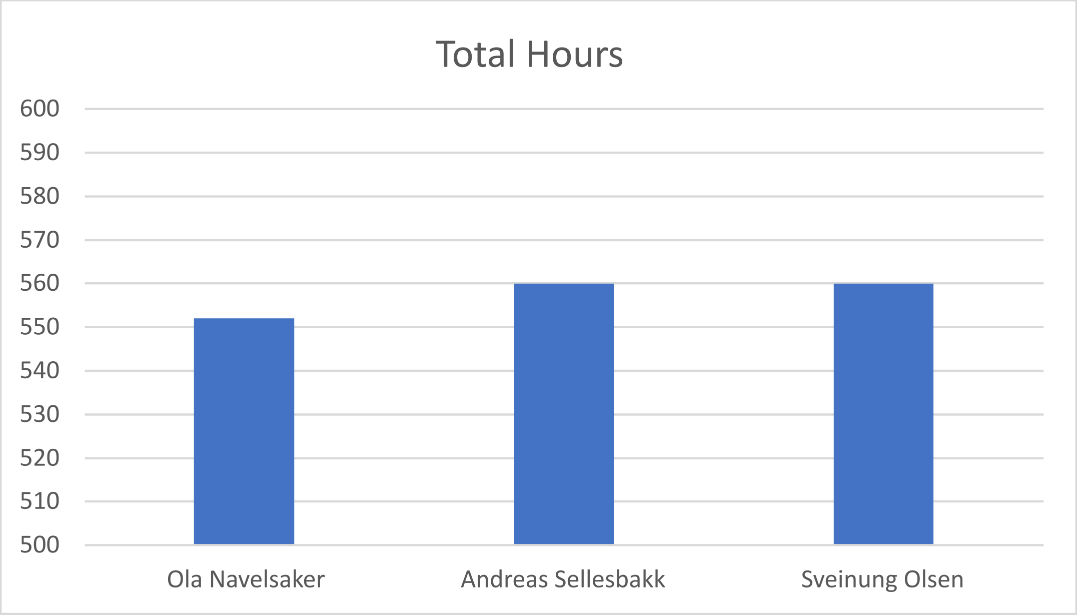


Figure 38: Bar chart showing how much time each group member has used.

The first two months of the semester we had a course running in parallel with the thesis. This meant that we had less time in the beginning to work on our thesis project, and therefore had a lot more hours later in the project. Time distribution per week is illustrated in Figure 39.

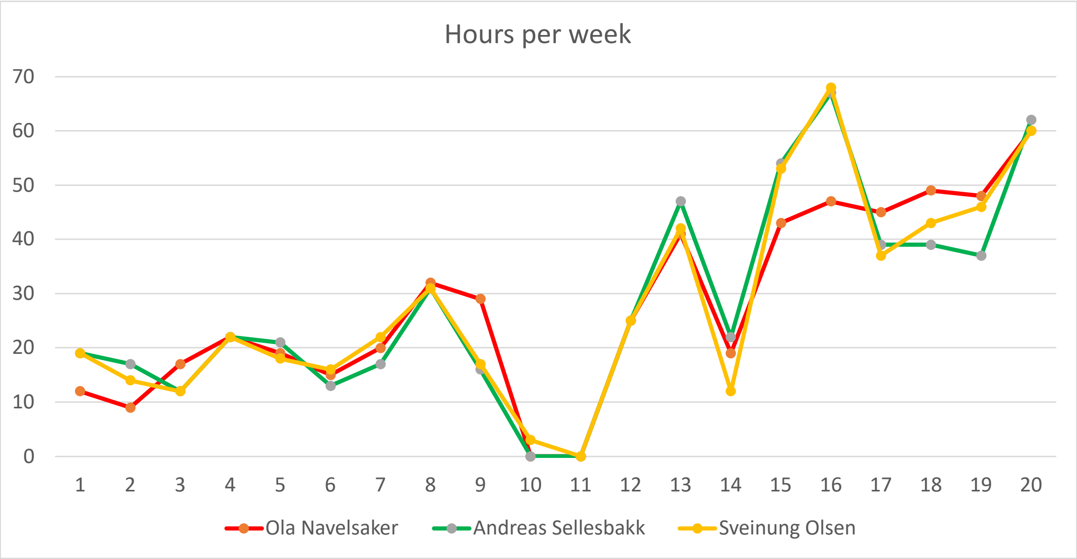


Figure 39: Line chart showing the distribution of hours.

6 Conclusion

In order to achieve a successful project conclusion, a comprehensive review was conducted to address and finalize the diverse components encompassing our project journey. This final phase aims to ensure that project goals are completed, and that we have learned something from this project.

6.1 Process

The group sees the advantages of using the sprint and thinks that the sprint was a good method to use to start the work, but as the issues got smaller over time, an alternative method was better. The group thinks that the change to using waterfall together with a Kanban issue board resulted in better overall progress than we would have had if we kept using sprints for the whole project. The group also thinks that this solution helped with the general overview of progress and made it easier in terms of planning ahead.

In addition, the group believes that incorporating the waterfall methodology with a kanban issue board helped us address the issue of changing requirements. With waterfall, each phase must be completed before moving on to the next, making it difficult to adapt to changing requirements. However, by using a kanban issue board, the team can prioritize and adjust their tasks based on changing requirements as they arise. This enabled the team to be more flexible and responsive to changing needs while still maintaining a structured approach to project management. Overall, the group feels that using waterfall with a kanban issue board provided a better balance between structure and flexibility and ultimately result in better overall progress.

6.2 Product

The group is overall satisfied with the final product. The purpose of the final product is to give the task provider a solid foundation where more calculators can be added as needed in the future. The group is confident that the final product will meet the needs of the task giver and provide a solid foundation for future development. Since the task provider wants to expand the application with more calculators in the future, a robust backend has one of the primary objectives.

During the user testing, some Barco employees gave us feedback. This feedback was very positive, and they seemed very happy with our overall progress. This positive feedback has given us the confidence that the final product meets the requirements, and that it is a solid foundation that Barco can continue to build on. Overall, the group is proud of the final product and the effort put into its development, and we believe that it has the potential to be a valuable tool for the task giver and those that use it.

6.3 Reaching Our Goals

The group think we did a good job working towards, and reaching, our target goals. Even though we did not reach all the goals, we made good progress towards them. We also believe that the work we have done provides a solid foundation that can be further developed to achieve all the goals we aimed for.

For this project, we set a lot of goals that we wanted to reach. The goals we think we did a good job of achieving are the goals that focus on what we can do, instead of what we can do for the customers. The reason we believe this is that the application is simply not ready for customers yet, and therefore we can not reach the goals that directly involve them. Some examples of the goals we think we did a good job achieving are as follows:

- Combine existing solutions into one application to improve usability and workflow.

- Design a software architecture that is scalable and flexible, allowing for future expansion and improvement of the projector calculator's capabilities.
- Test the software thoroughly to identify and resolve any bugs or issues, ensuring that it is reliable and robust.
- Gain experience in developing a full-stack application.
- Enhance our critical thinking and problem-solving skills by addressing technical challenges and resolving software issues.
- Build a strong foundation for future software development projects by developing a deep understanding of software design and architecture, programming best practices, and testing and debugging techniques.

On the other hand, some goals were not achieved, the group thinks that with more time or better initial planning these goals could have been achieved. The reasoning for this is that the final product is not yet ready for actual customers, thus any goals about having an impact on customers is seen as failed by the group. However, the group is still satisfied with the work done and thinks that the end goal for reaching customers is not that farfetched. Due to this, the group do not see this as a failure, but rather points which needs improvement. Here are the goals the group thinks are not fully achieved or does not have satisfactory results:

- Reduce installation challenges for both Barco engineers and customers.
- Create better workflow for the technical support by helping customers choose projector and lens combination based on defined parameters
- Reduce cases of buying the wrong product

6.4 Future Works

Before the application gets put to use, encryption of the communication between users and the server should be implemented to avoid sending sensitive information as plain text. This is mainly to reduce the effectiveness of a man-in-the-middle attack. Additionally, proper status codes from some of the API endpoints and error handling in case of unexpected responses from the database. Since a lot of our focus went into creating the backend for the application, the design is very basic and unrefined and should be improved upon to meet the standards one would expect from modern applications.

References

[1] Microsoft. Blazor app hosting models, 2022. Available: https://learn.microsoft.com/en-us/dotnet/architecture/blazor-for-web-forms-developers/hosting-models. Last accessed 04.04.2023.

[2] EUR-Lex - 02016R0679-20160504 - EN - EUR-Lex, 2023. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02016R0679-20160504. Last accessed 17.04.2023.

[3] Microsoft. ASP.NET Core Blazor, 2023. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0. Last accessed 13.04.2023.

[4] Microsoft. ASP.NET Core Blazor hosting models, 2023. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-6.0. Last accessed 04.04.2023.

[5] Microsoft. Overview of ASP.NET Core SignalR, 2023. Available: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-6.0. Last accessed 04.04.2023.

[6] Alexey Melnikov and Ian Fette. The WebSocket Protocol. RFC 6455, December 2011. Available: https://www.rfc-editor.org/info/rfc6455. Last accessed 06.05.2023.

[7] Mozilla. Document Object Model (DOM), 2023. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. Last accessed 04.04.2023.

[8] Mozilla. WebAssembly, 2023. Available: https://developer.mozilla.org/en-US/docs/WebAssembly. Last accessed 04.04.2023.

[9] Microsoft. ASP.NET Core Blazor Progressive Web Application (PWA), 2023. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/progressive-web-app?view=aspnetcore-6.0&tabs=visual-studio. Last accessed 20.05.2023.

[10] Microsoft. Relational vs. NoSQL data, 2022. Available: https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data#the-cap-theorem. Last accessed 13.04.2023.

[11] scrumguides.org. The 2020 Scrum Guide, 2020. Available: https://scrumguides.org/scrum-guide.html. Last accessed 13.04.2023.

[12] Microsoft. nchar and nvarchar (Transact-SQL), 2022. Available: https://learn.microsoft.com/en-us/sql/t-sql/data-types/nchar-and-nvarchar-transact-sql?view=sql-server-ver15#remarks. Last accessed 15.05.2023.

[13] GitHub, 2023. Available: https://github.com. Last accessed 15.04.2023.

[14] GitLab, 2023. Available: https://gitlab.com. Last accessed 15.04.2023.

[15] GitHub vs GitLab: Difference between GitHub and GitLab, 2022. Available: https://www.upgrad.com/blog/github-vs-gitlab-difference-between-github-and-gitlab/. Last accessed 15.04.2023.

[16] David J. Barnes & Michael Kölling. *Objects First with Java.* Pearson, 2016.

[17] Microsoft. Compare EF Core & EF6, 2022. Available: https://learn.microsoft.com/en-us/ef/efcore-and-ef6/#ef6. Last accessed 15.05.2023.

[18] Microsoft. ASP.NET Core and Entity Framework 6, 2023. Available: https://learn.microsoft.com/en-us/aspnet/core/data/entity-framework-6?view=aspnetcore-6.0. Last accessed 15.05.2023.

[19] Microsoft. Quickstart: Install SQL Server and create a database on Ubuntu, 2023. Available: https://learn.microsoft.com/en-us/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver16. Last accessed 05.05.2023.

[20] Microsoft. DbContext Class, 2023. Available: https://learn.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-6.0. Last accessed 06.05.2023.

[21] Microsoft. LINQ to Entities, 2023. Available: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/linq-to-entities. Last accessed 06.05.2023.

[22] David Tavarez. Implementing Elliptic-curve Diffie–Hellman Key Exchange Algorithm using C# (cross-platform), 2019. Available: https://davidtavarez.github.io/2019/implementing-elliptic-curve-diffie-hellman-c-sharp/. Last accessed 15.05.2023.

[23] Microsoft. CA1416: Validate platform compatibility, 2023. Available: https://learn.microsoft.com/nb-no/dotnet/fundamentals/code-analysis/quality-rules/ca1416. Last accessed 20.05.2023.

[24] Microsoft. ECDiffieHellmanCng Class, 2023. Available: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.ecdiffiehellmancng?view=net-7.0. Last accessed 20.05.2023.

[25] Microsoft. ClaimTypes Class, 2023. Available: https://learn.microsoft.com/en-us/dotnet/api/system.security.claims.claimtypes?view=net-7.0. Last accessed 05.05.2023.

[26] Charles Anderson. Docker [software engineering]. *Ieee Software*, 32(3):102–c3, 2015.

[27] Stf. Unit testing, 2023. Available: https://softwaretestingfundamentals.com/unit-testing/. Last accessed 17.05.2023.

[28] Katalon. What is regression testing? definition, tools, examples. Available: https://katalon.com/resources-center/blog/regression-testing. Last accessed 17.05.2023.

[29] Thomas Hamilton. Sanity testing vs. Smoke Testing – difference between them, 2023. Available: https://www.guru99.com/smoke-sanity-testing.html. Last accessed 17.05.2023.

[30] GeeksforGeeks. Differences between Black Box Testing vs White Box Testing, 2023. Available: https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/. Last accessed 17.05.2023.

[31] GeeksforGeeks. Random Testing in Software Testing, 2021. Available: https://www.geeksforgeeks.org/random-testing-in-software-testing/. Last accessed 17.05.2023.

[32] Wikipedia. Stress testing (software), 2023. Available: https://en.wikipedia.org/wiki/Stress_testing_(software). Last accessed 17.05.2023.

[33] Desktop Operating System Market Share Worldwide, 2023. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide. Last accessed 15.04.2023.

[34] GeeksforGeeks. Normal Forms in DBMS, 2023. Available: https://www.geeksforgeeks.org/normal-forms-in-dbms/. Last accessed 19.05.2023.

[35] NTNU. For deg som er student. Available: https://www.ntnu.no/isl/for-studenter. Last accessed 19.05.2023.

Appendix

A Detailed Database Tables

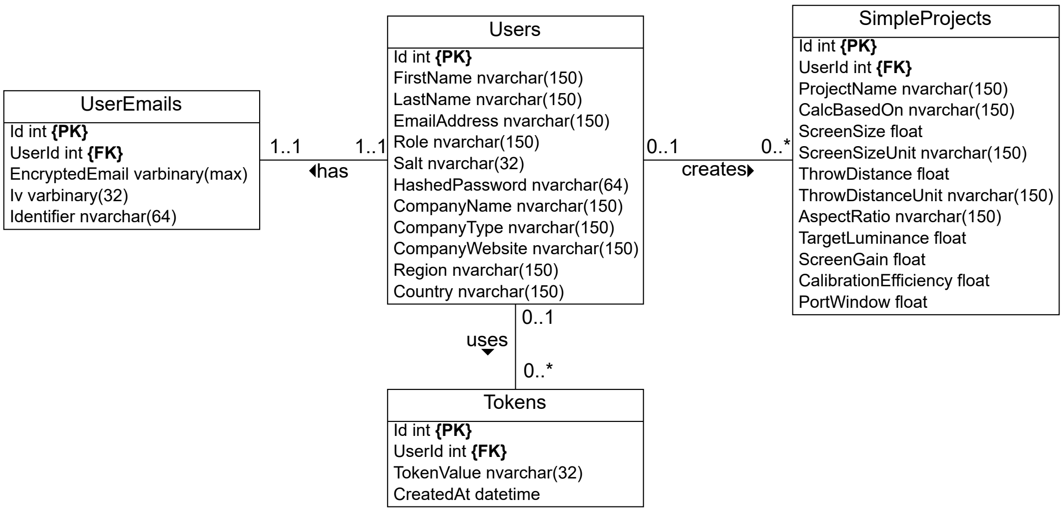


Figure 40: Shows how all tables attribute are defined, primary, foreign keys, and the relationship between tables.

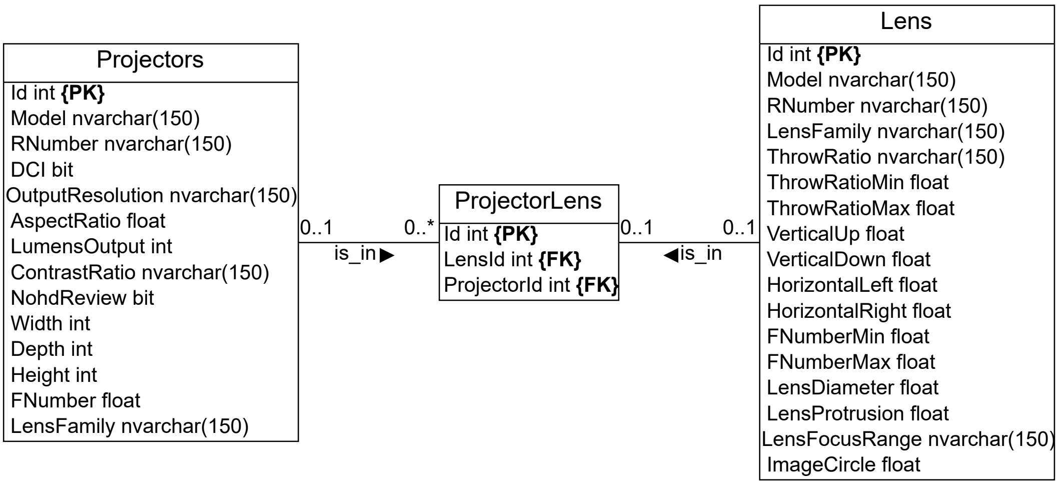
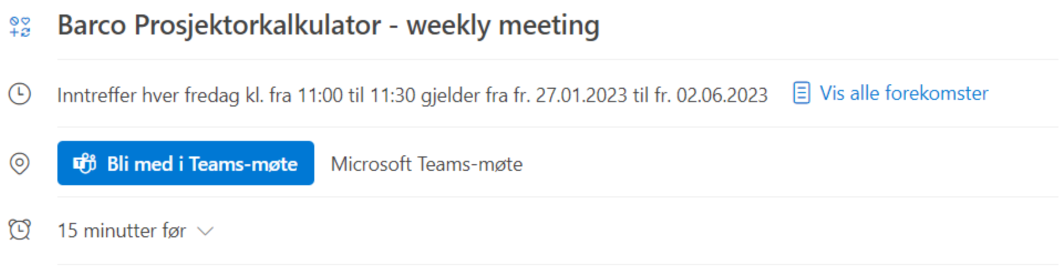


Figure 41: Shows how all tables attribute are defined, primary, foreign keys, and the relationship between tables.

B Meeting invitations



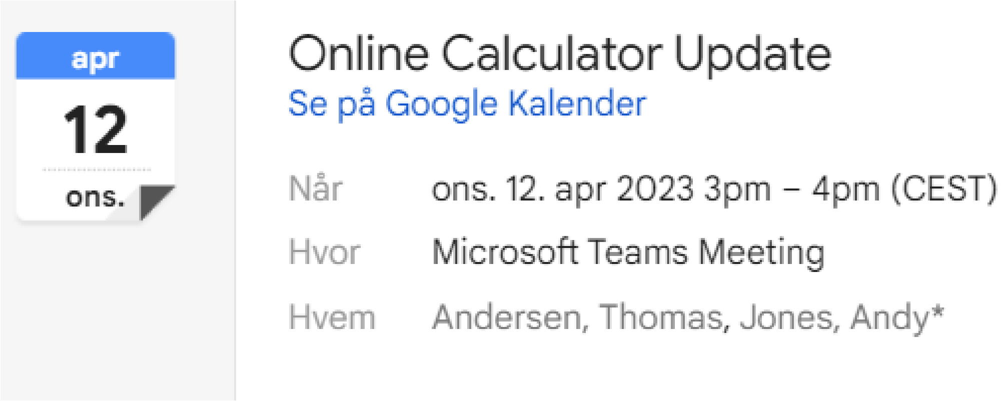
Barco Prosjektorkalkulator - weekly meeting

Inntreffer hver fredag kl. fra 11:00 til 11:30 gjelder fra fr. 27.01.2023 til fr. 02.06.2023

Microsoft Teams-møte

15 minutter før

Figure 42: Screenshot of the weekly meeting invite.



Online Calculator Update

Når ons. 12. apr 2023 3pm – 4pm (CEST)

Hvor Microsoft Teams Meeting

Hvem Andersen, Thomas, Jones, Andy*

Figure 43: Screenshot of meeting invitation from one of the meetings held with Barco employees.

C Meeting Notes

Dato: 27.01.2023

Figuren som viser rollene til hvert medlem, trenger større / mer lettlest tekst.

Legge til en figur av kanban tabellen.

Gjøre om alle kalkulatorene fra en liste til under kapitler.

Legg til bildene / figurer om de forskjellige kalkulatorene (de som vi har fått fra Barco).

Få ferdig "Plan of Action" kapittelet og få lagt inn det nye Gantt diagrammet.

Legge til noe tekst for å forklare risiko analyse tabellen.

Skrive mer eller omskrive «Delimination» kapitelet.

Skrive mer i «Case Description», unngå korte / få linjers paragraf, trenger bedre flyt.

Dato: 03.02.2023

Gjøre om felt typene i test databasen til å holde tekst av lengde 150 fremfor 50, kan heller senere endres til enda mer spesifikke typer senere.

Utvikling av verktøy må prioriteres nå tidlig. Mye av senere utvikling er avhengige av at database har innhold.

Dokumenter prosessen underveis, skriv ned feil eller bugs som blir funnet så disse blir dokumentert. Som f.eks. hvilke projektorer som man ikke finner noen lense til.

Standardiser datasettet så godt det er mulig.

Bruk tid på å skjønne hva datasett er, og hva det inneholder. Kan bli viktig å skjønne datasettet for å kunne hente ut riktig informasjon.

Dato: 17.02.2023

Endre datatypene i user tabellen. Som f.eks. salt og hashed password vil ha en forhåndsbestemt maks lengde.

Endre fra å bruke brukernavn til å bruke e-postadresse når de logger inn.

Få satt opp en database for brukere, design er unnagjort bare implementering og oppsett av database som gjenstår.

Så fort verktøyet og databasen som holder produkter er oppe må den implementeres inn i applikasjonen. Bruker forhånds definerte lister i minnet nå.

En-til-en mellom lens og projector lens, ikke mange-til-mange.

Videre fokusere på database development og begynne arbeid på kontrollerende.

Viktig å dokumentere prosessen og valg som blir gjort underveis. Dette kan brukes senere når oppgaven skal skrives ferdig.

Dato: 24.02.2023

Ekstra notat:

- Lite fremskritt før dette møte pga. emne ved siden av bachelor oppgaven.

Fremfor å bruke «\section» i mainbody filen fordel de forskjellige hoved kapitelene i egne filer.

Prøve å unngå å bruke personlig pronomen, da spesielt tanke på vi, gruppa.

Når det kommer til skriving: prioritere å bli ferdig med innlednings kapitelet.

Få lagt inn alle under kapitelene slik at man lettere kan se hva som mangler og hvordan strukturen på oppgaven blir.

I innlending kan mye tas fra prosjekt planen. Evt. Skrive om / legge til mer om noe skulle mangle.

Når det kommer til kode: prioriter kontroll klassene slik at applikasjonen kan begynne å snakke med databasene.

Fjerne unødvendige / ikke brukte filer fra prosjektet (kode).

Få lagd modell klassene som trengs til å koble applikasjonen til databasene.

Dato: 03.03.2023

Skriving: prioritere å komme i gang med kapital 2 (Requirement) og 3 (Development Process).

Fjerne «Language Constraints» fra innledningen, gir ikke mening å ha den med.

Unngå korte / noen linjers paragraf, disse bør enten slås sammen andre mindre paragraf eller bli omskrevet til at det gir mening at de er et eget paragraf.

Skrive / legge til mer om «Target Audience».

Fjern «Group Background», irrelevant for oppgaven.

Under kapitel 1.5 (Constraints) legge et underkapittel for programvare (software). Bør handle om hva oppdragsgiver ønsker at vi bruker av språk og generelle software constriants.

Utvide / legge til flere mål (golas) både project og effect.

Legge til et under kapitel om bærekraft for kapitel 1.4 (Goals).

Legge til nytt kapitel i innledningen om Contribution.

Så på muligheter for å hoste applikasjonen på en virtuell maskin. Blir da evt. Kun internt på NTNU sitt nettverk.

Dato: 24.03.2023

Ekstra notat:

- 3 uker siden sist møte pga. eksamen i emne som så langt har gått parallelt med oppgaven og klassetur til Sveits.

Under kapitel 2.1.3 (Projector Calculator) legge inn bilder fått fra Barco av hvordan kalkulatorene fungerer. Må også forklares hva som menes, type hvilke inputs trenger kalkulatoren fra brukeren.

Fortsette å legge inn figurer som kan hjelpe leseren med å skjønne hva vi prøver å forklare i teksten.

Skriving: prioritere å bli ferdig med kapitel 2.

Husk å skrive / ta notater underveis. Gjør det lettere å forklare det i teksten senere.

Implementere token system. Token kommer til å bli brukt for epost konfirmasjon og tilbake stilling av passord.

Legge til en spinner eller loading bar når man venter på svar fra serveren etter at brukeren har sendt en request.

Forbedre det visuelle med kalkulatorene. Spesielt hvordan resultatene vises til brukeren. Per nå gnaske uoversiktlig og lite intuitivt.

Dato: 31.03.2023

Legge til figur under kapitel 2.2 (Technical Design) som gir et bilde av det tekniske designet.

Legge til en under kapitel i 2.2 (Technical Design) om hvilken modell som ble valgt og hvorfor.

Legge til figur for forrige punkt.

Legge til en tabell som viser de viktigste forskjellene på modellene.

Trenger intro mellom kapitler, ikke la kapitler komme etter hverandre uten tekst imellom.

Vurdere å fjerne de kalkulatorene det ikke er skrevet noe om under kapitel 2.1.3.

Så fort dette er fikset i kapitel 2, begynne å skrive på kapitel 3.

Viktig at figurer som legges til må bli referert til i teksten og at teksten i figuren er stor nok til at den er leselig også når oppgaven skrives ut.

Fortsett å se på muligheter for å bedre displayet / resultatene fra kalkulatoren, gjøre de mer bruker vennlige.

Bli ferdig med e-post konfirmasjon logikken.

Dato: 14.04.2023

Ekstra notat:

- Ikke noe møte forrige uke pga. påske.

Mangler fortsatt bilde under kapitel 2.2 (Technical Design) hvor man snakker om hvilken modell som ble valgt. Figuren bør vise hvordan modell valg påvirker arkitektur.

Figur 2.5 under kapitel 2.3.3 om SQL vs. NoSQL, er ikke intuitiv. Trenger en bedre figur og bedre forklaring.

Første setning i siste avsnitt i innledning er uviktig og bør fjernes. For innlysende.

Under kapitel 2.2.2 (Blazor Server) mangler kilde, ikke lagt inn enda.

Fortsatt noen kapitler etter hverandre uten tekst imellom, bør fikses.

Bør sjekke hvilket epost domene nye brukere er fra, og utfra det bestemme om de kan motta epost med en gang eller om en administrator må gjøre det manuelt.

Fikse ekstra verktøyet slik at databasen bruker mer spesifikke datatyper og ikke bare tekst.

Fjerne epost fra bruker tabellen og heller ha det som en egen tabell. Implementere kryptering (epost i ikke klartekst).

Dato: 21.04.2023

Legge til en tabell som viser de viktigste forskjellene på modellene. (Teknisk design)

Flytte kapitelet om git til en egen kvalitets seksjon og rette noen små feil.

Camel case på kapitel navn.

Gjøre bildene bedre eller ta flere bilder som viser mindre. Ikke prøv å vis alt i en figur der det ikke er hensiktsmessig.

Fjerne korte og en linjers paragraf.

Legge til intro mellom kapitler der det er naturlig.

Legge til mer om testing og de forskjellige typene tester.

Legge til en bekreftelse knapp på alle steder hvor man sletter noe fra database.

Få opp en nyere versjon på Docker.

Gjøre så brukere kan slette sine egne kontoer og all informasjon som applikasjonen lagrer.

Dato: 28.04.2023

Under kapitel 1.2 (Target Audeience) skrive om og legge til mer, lite informasjon.

Under kapitel 1.3 (Delimitations) siste avsnitt må omformuleres.

Mangler figur under kapitel 2.1 (Project Requirements) ønskelig oppnåelse. Unngå «I», der det trenges bruk heller tredjeperson.

Under 2.1.3 (Simple Project Calculator) legge til forklaring på hva Nits og foot lambert er.

Mangler figur for 2.2 (Technical Design).

2.3.3 (Database) database er strengt tatt ikke nødvendig, skriv om. Siste del av avsnitet trenger også omformulering, ikke tydelig hva som menes.

Under kapitel 4.1.2 (Code Testing) må under kapitelene enten være et faktisk underkapittel eller i det minste fet tekst.

Komme i gang med user test. Trenger en god del for å få statistikk av verdi.

Viktig å forhånds definere hva slags oppgaver de skal testes på og bruk et verktøy for å ta være på resultatene.

Dato: 05.05.2023

Få lagt inn akronymer og definert i «glossary» filen.

Legge inn kildene brukt under kapitel 2.3.3 om «Coding Best Practices»

Legge til database diagram om de nevnte tabellene i kapitel 3.1.1 (Database).

Dobbelt sjekke alle referanser til figurer, tabeller og kapitler. Disse skal starte med stor bokstav, f.eks. see Figure X.

Figur under kapitel 3.1.3 (Calculator Code Implementation) trenger større / mer lesbar tekst. Figuren mangler også figurtekst.

Legge til figurer for 3.1.4 (Registration) og for 3.1.5 (Login).

Vurdere å legge til et kapitel om Docker teori. Må vurderes om er nødvendig eller ikke.

Legge inn resultatene fra user testene enten som tabeller eller som søylediagram. Om det blir for mye så kan noe legges i appendices og refereres til fra teksten.

Prioritere å skrive på kapitel 5 (Discussion) og kapitel 6 (Conclusion) og heller lese over når mer av helheten er på plass.

Viktig at valg blir begrunnet. Også forklar alternative løsninger, positive og negative sider.

I konklusjonen dra frem målene som ble satt.

Dato: 12.05.2023

Forbedre tabellene, er for uoversiktlig. Legge til maks verdiene fått for hvert spørsmål i testen.

Fjerne hva slags form vi har brukt uviktig.

I kapitel 2.1.6 – gjøre det om til en liste med kulepunkter.

I kapitel 2.3 skrive mer om hva som forandra seg og heller ha hvorfor i diskusjon.

Skrive i fortid, retrospektiv skriving.

Skrive mer om "Expandable, Security and Usability Testing".

Restrukturere «Thesis Structure».

Konverter Excel dataen til overleaf tabeller og ikke bilder.

På starten i «Techincal Design» referer tilbake til første figuren vist.

Legge til en figur som viser teknisk design i kapitelet hvor man velger hosting modell.

Legge til bilde av featuren som brukere syns var vanskelig.

Diskusjon skal inneholde hva som kunne blitt gjort annerledes, hva som ble lært, planlegging.

Dato: 19.05.2023

Legge til lett / vanskelig på x-aksen på søylediagrammene for å øke leseligheten.

Legge til hvor inspirasjon for figurene som er laget ble hentet fra.

Legge til hvor dataen som er i tabellen under kapitel 2.2 er fra.

Fiske kildelista, per nå 3 kilder som ikke vises riktig.

Skrive noe om database diagrammene. Forklare hva 0, 1 og * betyr, nevne noe om UML standard hvordan vi unnviker fra standarden når det kommer til primary key og foreign key.

Noen feil med figur 19 under 3.1.4.1, kan også vurdere å forklare den enda bedre.

Kan vurdere å legge til flere figurer som ligner på figur 19.

Flytte den ene delen av Unit testing kapitelet til diskusjon.

Se på andre måter / andre typer diagram som kan vise resultatene fra user testene.

Under Throw-Ration Visualization kapitelet legge til en figur som viser hvordan det ser ut evt. Også hvordan den lages.

D Sprints

Sprint 1:

Start: 27.01.2023

End: 17.02.2023

The first sprint will be three weeks instead of the previously agreed two weeks since at the start of this sprint finalizing the project plan is the main concern.

Tasks to be done:

1. Set up a virtual machine which will host our SQL database.
2. Create entity relational diagram for database. Diagram for the projector and lens tables are priority. Later we will need diagrams for user and projects as well.
3. Create the database itself. Using the information from the diagram.
4. Create tool for populating database with test data. The tool must be able to extract data from an excel sheet and populate the database.
5. Set up the blazor project.
6. Create the basic layout of the application.
7. Create the basic color scheme to fit with Barcos already existing software.
8. Once the basic design is done, work on the simple calculator can begin.
9. Set up a virtual machine that can host the application.
10. Host an example application on the virtual machine reachable within the network.

Task distribution:

Sveinung: Tasks 1-4.

Andreas: Tasks 5-8.

Ola: Tasks 9-10.

Sprint 2:

Start: 17.02.2023

End: 03.03.2023

Tasks to be done:

1. Start looking into using views for the database, rather than the tables itself.
2. Start work on database controller. Creating a tool/feature for getting data from the database to be used in the different calculators.
3. Create the user database where user information such as name and password are stored. Also needs to look into how to store the passwords (using salt and hashing). Need to determine what hashing function to use.
4. Start work on authentication and authorization. Look into role-based authorization but prioritize authentication during this sprint.
5. Continue work on the different calculators. Improve output display and overall functionality.

Task distribution:

Sveinung: 3-4

Andreas: 5

Ola: 1-2

Sprint 3:

Start: 03.03.2023

End: 17.03.2023

Due to an exam and a class trip, the progress during this sprint will be significantly reduced.

Tasks to be done:

1. Implement functions for the database tool.
    a. Implement add projector/lens functions.
    b. Implement delete projector/lens functions.
2. Start work on authorization. Includes tasks such as making login mandatory, limiting the access to pages based on the role gotten when logged in.
3. Continue implementation of calculators. Improve error handling, output display and general usability.
4. Set up a virtual machine to host the application (continuation of task number 9 and 10 from sprint 1).
5. Write chapter 2 in the report.

Task distribution:

Sveinung: 2, 4

Andreas: 3, 5

Ola: 1a, 1b, 5

Sprint 4:

Start: 20.03.2023

End: 03.04.2023

Tasks to be done:

1. Implement functions for the database tool.
    a. Implement add projector/lens functions.
    b. Implement delete projector/lens functions.
    c. Implement update projector/lens functions.
2. Make admin page for users. Admins should be able to see every registered user from the database. Also need to look into a feature which enables the admin to send a confirmation email. Admin should also be able to remove users.
3. Continue implementation of calculators. Continuation of sprint 3 task number 3.
4. Continue writing on the report.

Task distribution:

Sveinung: 2

Andreas: 3-4

Ola: 1

Sprint 5:

Start: 03.04.2023

End: 17.04.2023

Tasks to be done:

1. Make visualization for throw ratio to improve usability.
2. Continue calculator development. Start work on save and load projects. Users should only see their own projects.
3. Continue admin tool development, continuation of the previous sprint number 4 tasks 1 a-c.
4. Start development on lens shift calculator.
5. Refactor token usage and email confirmation. Create a token service and controller.

Task distribution:

Sveinung: 4-5

Andreas: 1-2

Ola: 3

E Project Plan

NTNU
Kunnskap for en bedre verden

Department of Computer Science

IDATG2900 - Bachelor Thesis

Project plan

Authored by:

Ola Henrik Otterlei Navelsaker
Sveinung Olsen
Andreas Haug Sellesbak

Date: 18.01.23

Table of Contents

List of Figures

List of Tables

1 Background

Barco is a technology company with headquarters in Belgium, while we are working with their office in Norway, Fredrikstad. The section in Norway was previously a company known as Projectordesign which Barco bought in 2013. Their main focus is on high-end projectors for 3D, cinema, home entertainment and simulation for planes and trucks. When it comes to projectors they are known mostly for their simulation and home entertainment solutions and have secured contracts with customers such as the American Air Force. Our project is focused mainly on the residential use, where Barco wants to reduce the number of customers who choose the wrong projector-lens combination for their given room.

2 Project Organization

2.1 Responsibilities and roles

Barco has provided us with three different contact persons, with different competence, and can provide us with different input based on the current problem we are facing. These contact persons are Thomas Andersen, Victor Steen and Andy Jones.

Thomas is the product owner, and can support us with general feedback and input about the product.

Victor is the technical manager, and can support us with coding principles and internal standards, database structure and management, and any other technical issues.

Andy is the field engineer, he helps their costumers with calculations and selecting products or help costumers in other ways. This means he will be using the tool that we are going to be creating on a frequent basis, and therefore can come with general input on how it should work, what features to implement, design and usability. He can also help with the mathematical equations used in the different calculations needed.

From NTNU we have a project supervisor, Giorgio Trumpy. Giorgio can help us with questions related to the report, and general guidance.
Within our team we are three members, Sveinung, Andreas and Ola.

Sveinung is the group leader, and will be in charge of the contact between our group and Barco.
Andreas will be in charge of contact between our group and Giorgio.
Ola responsible for calling meetings with the group.

All the members of our group will also serve a role as full-stack developers (Front-end and back-end development).

2.2 Rules and procedures

- Meetings: Ola will be responsible for sending out summons for meetings, this will be done using Discord.
- Notification in case of absence: A valid reason for absence must be presented and notifications must be sent to the other group members as soon as possible, no later than 24 hours before the meeting.
- Presence and commitment: We will work continuously and be productive during meetings. If productivity falls, the group leader will propose a break to restore productivity.
- Monitoring task: The Kanban board in the group's gitlab will provide an overview of the tasks being worked on, future tasks, and the completed tasks ready for review. See figure 2.

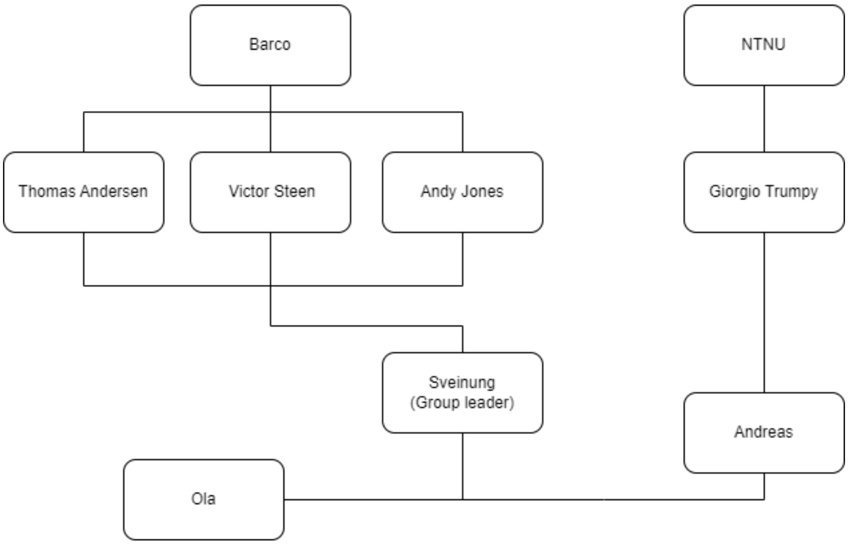
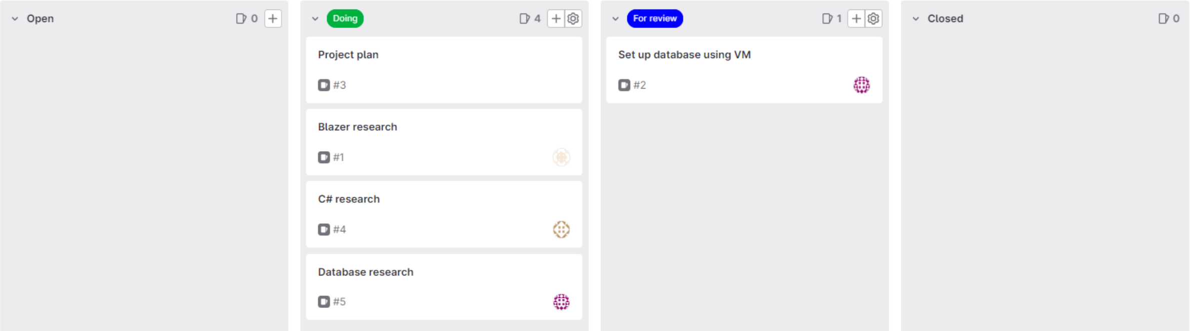


Figure 1: Role diagram

- Distribution of tasks: Will be decided during meetings. We will try to spread tasks evenly and try to give each member tasks they find interesting and want to solve. However, the groups' needs takes priority over individual members' wishes.
- Versioning: Git will be used for versioning, while the repository will be on gitlab.
- Timesheet: After work is done each member is responsible for updating the timesheet which is an excel sheet to keep track of work-hours, shared using teams.
- Breach of contract:
  1. Discussion with just the group member to resolve any issues.
  2. Written warning.
  3. Involving supervisor on the third warning to resolve the conflict.



Project plan #3

Blazer research #1

C# research #4

Database research #5

Set up database using VM #2

Figure 2: Kanban table

3 Case Description

The project is to build an online calculator that helps Barcos' customers pick the right projector and lens combination.

The calculator is divided into three main tasks.

The first part of the project is a database. The database will need to contain at least four different tables. These tables will be used for storing information about users, projectors, lenses and projects.

The project table may be done at a later stage since it is a part of the advanced calculator and will not be prioritized from the start.

The second part of the project is an admin tool. This admin tool is to maintain and update the database, the admin should also be able to manage users.

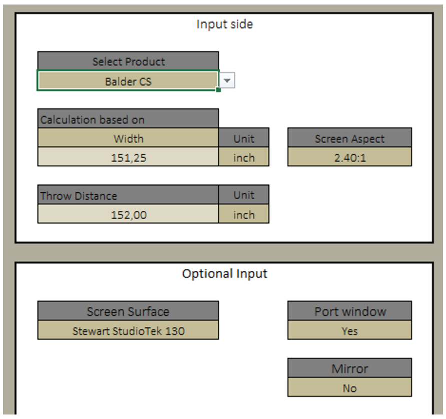
The third part is the actual calculator application. This application should give the user the option to create a user, login, and save their projects. It also includes several separate calculators, which are as follows:

3.1 Find correct projector

Add distance to screen, screen size and target luminance. The tool should be able to find the best projector and lens for the application based on target luminance.

3.2 Simple project calculator

Add Projector, screen dimensions, screen aspect, throw distance, screen type or custom screen with gain and if you use a mirror and a port window. Calculator should return, all screen dimensions, lens to use, maximum shift available, f-number, alternative lens options if available, and all luminance values in either Nits, foot lambert or both.



Input side

Select Product: Balder CS

Calculation based on
Width: 151,25 — Unit: inch
Screen Aspect: 2.40:1

Throw Distance: 152,00 — Unit: inch

Optional Input

Screen Surface: Stewart StudioTek 130
Port window: Yes
Mirror: No

Figure 3: Simple project calculator

3.3 Lens shift calculator

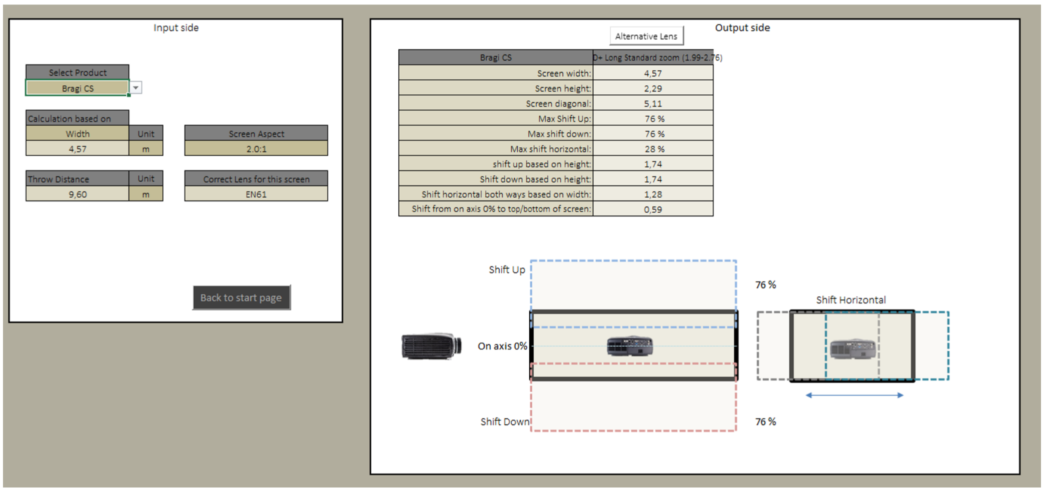
Add projector, lens, screen dimensions, aspect ratio, throw distance, center of lens vs center of screen. Calculator will return if you can run this installation and how much shift you will have available.

3.4 Advanced offset calculator

Tools and formulas to calculate this will be provided by Barco at a later stage.

3.5 Advanced project calculator

This is the advanced version of the calculator, where the goal is to combine all the above calculations into one, and add a few more layers of information. This is to calculate full cinema room projects.



Input side

Select Product
Bragi CS

Calculation based on	
Width	Unit
4,57	m

Screen Aspect
2.0:1

Throw Distance	Unit
9,60	m

Correct Lens for this screen
EN61

Output side

Bragi CS	D+ Long Standard zoom (1.99-2.76)
Screen width:	4,57
Screen height:	2,29
Screen diagonal:	5,11
Max Shift Up:	76 %
Max shift down:	76 %
Max shift horizontal:	28 %
shift up based on height:	1,74
Shift down based on height:	1,74
Shift horizontal both ways based on width:	1,28
Shift from on axis 0% to top/bottom of screen:	0,59

Figure 4: Lens shift calculator

4 Delimitation

In this project we were given the task to create a system for Barco. This system is well defined by Barco in the project description, and therefore very few delimitations will be set by us. The implementation of separate calculators will be prioritized in the order listed in the case description chapter. Barco does not expect us to implement all the listed calculators, but we will implement as many as time allows.

5 Domain

In the domain of projectors, Barco designs and manufactures a wide range of digital projectors for different industries and applications. Their projectors are known for their high brightness, color accuracy, and image quality. They offer projectors for various environments such as conference rooms, education, live events, and large venues such as movie theaters and concert halls. They also offer projectors with specialized features such as 3D projection and ultra-short throw projection. Barco's projectors are equipped with advanced image processing technology and support multiple connectivity options, making them suitable for a wide range of applications. They support the use of their projector by providing a range of services and solutions, such as maintenance services and control systems.

Barco have a wide range of projectors and lenses that need to be selected based on many different parameters. These parameters include, but are not limited to: room dimensions, screen dimensions, lens shift, throw ratio, focus range, output resolution, and contrast ratio. This requires complicated calculations to get the best result based on the specific scenario.

6 Framework

Together with Barco we have decided on the following framework requirements:

The project will be using C# and Blazor.
For the database we will be using Microsoft SQL server management studio (SSMS).

Barco has also recommended that we develop the application using Visual Studio.

7 Planning, Follow Up and Reporting

7.1 Development Model

For the software development model, the group decided on using the scrum method. This method is based on defining a set of tasks or features for a sprint. Each sprint usually has a duration of around 30 days and at the end of a sprint all new features are evaluated. This method is used to better adapt to unforeseen problems or changes in requirement and allows for incremental development of the software. Another reason the group decided on this method is because after each sprint we will have an opportunity to get the status of the overall progress. The group finds this method to be the easiest to work with since we will be able to determine what features or tasks that will be needed to be done during the next sprint.

7.2 How will the group use the method

At the start of each new sprint the group will list all new features or tasks to be included in the sprint and then distribute tasks to each member. We also plan to use the Kanban table in gitlab to keep track of every new issue encountered for later sprints. Each member is supposed to add issues or missing features as they are encountered. By doing this, time will be saved when looking in the sprint backlog for tasks to be completed during a sprint. This assumes that features from the requirement is already added to the Kanban table, if not they will need to be added as the group progresses. When it comes to the duration of each sprint, the group decided that 30 days is too long and decided to use a duration of two weeks instead. However, this may change depending on progress and complexity. It is also possible to work on the same task over multiple sprints if the task is too complex to be completed in one sprint.

8 Project Goals

8.1 Result Goals

- Reduce installation challenges for both Barco engineers and customers.
- Create better workflow for the technical support by helping customers choose projector and lens combination based on defined parameters.
- Combine existing solutions into one application to improve usability and workflow. The current solution is based on various excel sheets, some python scripts and a simple lens calculator which can be found at: https://lenscalculator.barco.com/

8.2 Effect Goals

- Improve convenience for adding new products and highlighting products nearing their end of life (EOL).
- Gain experience about developing a full stack application with `C#` and Blazor.

9 Quality assurance

9.1 Documentation

To keep the project easier to maintain and develop we decided to make some basic rules for documentation, these are the following:

- Each git commit should have a descriptive commit message.
- Issues, and their status should be kept track of with the kanban board.
- All code must be commented.
- Comprehensive README for the project.

9.2 Standards

To ensure a high level of coding standard we will be following best practices. This will also make it easier to document and maintain the product in the future. These coding standards can be found at https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions.

9.3 Testing

To test our code we will be using unit testing. This will be done using an external library called bUnit. Unit testing is a software testing method where each individual part of the code is tested separately. This helps ensure that each part of the code works as intended. Unit testing makes it easier to fix bugs earlier in the development process, and makes it easier to change the implementation of functions without creating new issues or bugs.

9.4 Risk Analysis

9.4.1 Risk Assessments

Likelihood and consequence are on a scale of one (low) to five (high), and the risk is these two numbers multiplied.

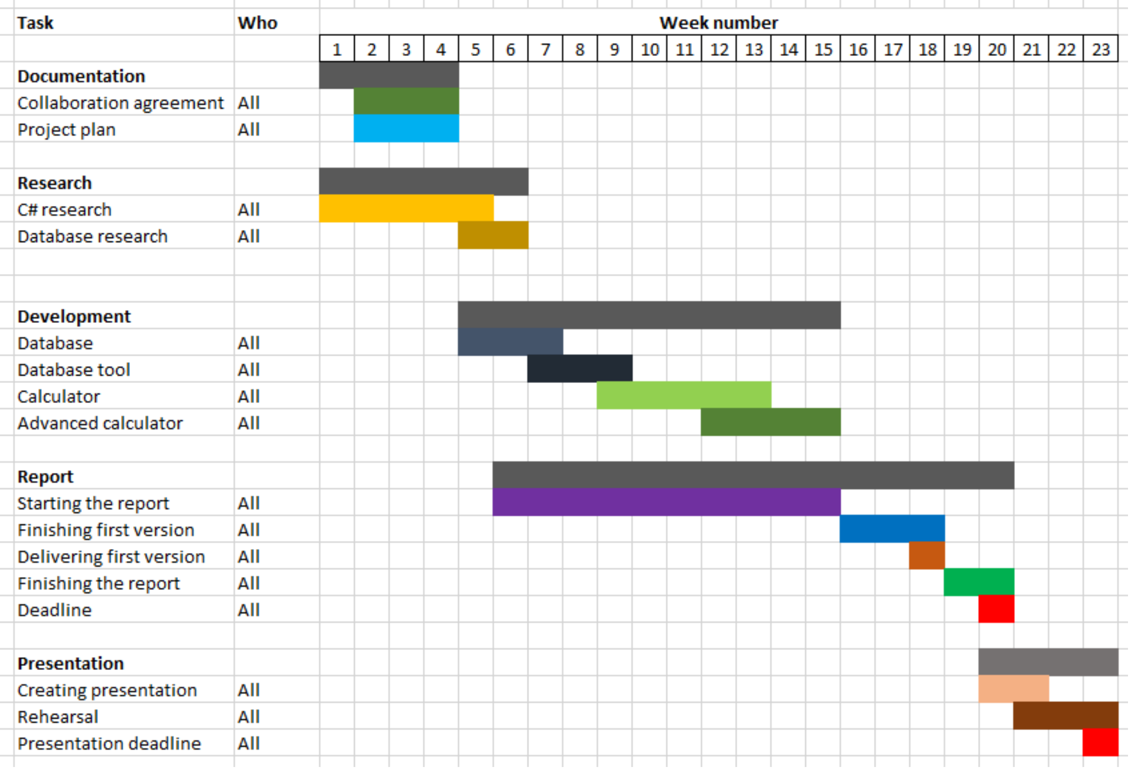
Description	Likelihood	Consequence	Risk	Measures
Internal conflict	1	2	2	Take a break, then discuss it after.
Not meeting deadlines	2	4	8	Hosting weekly meetings where attendance is required to work with the project. If short on time, host more meetings and set internal deadlines.
Unable to implement a functionality	4	3	12	Ask Victor for help.
Unable to implement everything the client wishes	5	1	5	Discuss with the client the importance of said wishes and weigh it against the importance of the work that would take precedence.
Loss of documents or code	1	3	3	Frequent use of Git with version control. After every work session commit the work to the repository.
Group member is away	4	1	4	Every member performs similar tasks and work together so the work can continue even if a member is not present.

Table 1: Risk Assessment

10 Plan Of Action

10.1 Project Plan

To make sure we did not go into this project without a solid idea of how to proceed we made a plan of action. This plan does not need to be followed precisely and we will most likely deviate from the plan slightly, but it is there to provide an overview of the project and an insight into when the different parts of the project should be completed. We decided to create a Gantt diagram, since this is a good way to both make a plan, and illustrate it in a simple way.



Task	Who	Week number																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Documentation		■	■	■	■																			
Collaboration agreement	All		■	■	■																			
Project plan	All		■	■	■																			
Research		■	■	■	■	■	■																	
C# research	All	■	■	■	■	■																		
Database research	All					■	■																	
Development						■	■	■	■	■	■	■	■	■	■	■								
Database	All					■	■	■																
Database tool	All							■	■	■														
Calculator	All									■	■	■	■	■										
Advanced calculator	All												■	■	■	■								
Report							■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			
Starting the report	All						■	■	■	■	■	■	■	■	■	■								
Finishing first version	All																■	■	■					
Delivering first version	All																		■					
Finishing the report	All																			■	■			
Deadline	All																				■			
Presentation																					■	■	■	■
Creating presentation	All																				■	■		
Rehearsal	All																					■	■	■
Presentation deadline	All																							■

Figure 5: Gantt diagram

10.2 Crash Courses

Early in this project our group will go to Barco's office in Fredrikstad to get a crash course on projectors. We chose to do this since it is vital to our project to have a good understanding of how projectors work, what the different parameters actually mean, and how to convey this information to the user.

F Project Agreement

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt:	
Veileder ved NTNU: e-post og tlf.	Giorgio Trumpy Giorgio.trumpy@ntnu.no 61135142
Ekstern virksomhet: Barco Fredrikstad Ekstern virksomhet sin kontaktperson, e-post og tlf.: Thomas Andersen, thomas.andersen@barco.com, 91109011	
Student: Fødselsdato:	Andreas Sellesbakk 12.07.2001
Ev. flere studenter[1]	Sveinung Olsen 13.07.2000
	Ola Navelsaker 06.06.1999

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 11.01.2023
Sluttdato: 22.05.2023

Oppgavens arbeidstittel er:

Barco - Projector calculator for residential use

[1] Dersom flere studenter skriver oppgave i fellesskap, kan alle føres opp her. Rettigheter ligger da i fellesskap mellom studentene. Dersom ekstern virksomhet i stedet ønsker at det skal inngås egen avtale med hver enkelt student, gjøres dette.

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:
Reiser til Barco Fredrikstad.

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[2]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten

[2] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

X	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene: Resultatene fra oppgaven (kalkulatoren, ikke selve oppgave teksten) skal benyttes av Barco ansatte og deres kunder. Det er derfor ønskelig at dette eies av Barco. Barco ønsker også å se selve oppgaven før den sendes inn som ferdig oppgave.

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

X	Oppgaven skal være offentlig

Selve oppgaven (teksten kan offentliggjøres etter at Barco har sett gjennom teksten, men kalkulatoren og dens kode skal ikke offentligjøres.

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder:
Dato:

Veileder ved NTNU:
Dato: 19/5/23

Ekstern virksomhet: Thomas And
Dato: 8/3-23

Student: Andreas Sellesbakk
Dato: 8/3 -23

Ev. flere studenter Sveinung Øen
8/3 -23

8/3 -23 Ola Nordsæther

NTNU

Norwegian University of
Science and Technology

BARCO