

Esport Team Twente Lounge Reservation System

Design Project: Design Report



Design Project Group 18

Ivan Trendafilov (s2309858)

Viktor Tonchev (s2332000)

Boris Belchev (s2388952)

Pavel Hristov (s2315424)

Irvine Verio (s2273454)

Abstract

This report explains the design and process behind “Esport Team Twente Lounge” which is an online reservation system that has been developed for the Esports Lounge Twente (EL). It is a collaborative effort between Esports Team Twente (University student-team), Blueshell (University of Twente Esports organisation) and the Student Union. The main goal of this project is to enable students to use the newly built lounge services. This system is essential since the lounge has just been finished and does not have any means for the students to be able to use the lounge services. In order to realise this, a web application system has been made such that students could use the lounge services by reserving the facilities and/or devices in the lounge. Besides, the administrators are also able to manage the students’ reservations in the system.

Introduction	6
Domain Analysis	8
Introduction	8
Glossary	8
General knowledge about the domain	8
Users, operators and concerned parties	8
The environment	9
Tasks and procedures performed	9
Similarities between other UT software	9
Stakeholders	11
Onion model stakeholder	11
Direct stakeholders	12
Students	12
Developers	12
EL staff	12
Client	12
Indirect stakeholders	12
Student union	12
Esports Team Twente	12
Blueshell	12
University of Twente	13
Requirement Specification	14
Agile Project Management Approaches for Requirement Specification	14
Requirement Formulation	14
Requirement Prioritisation	15
Stakeholder Requirements and System Requirements	15
System Proposal	16
Meetings with client	16
Requirements proposal	16
Mockups proposal	16
Preliminary Design	17
System increment	22
Presentation proposal	23
Requirement Analysis	24
Stakeholder requirements	24
System requirements	26
Functional requirements	26

Non-functional requirements	28
Risk analysis	29
R1: Connecting the new system with the external login system	30
R2: Preserving correct dates and times	31
R3: Time schedule exceeded	32
R4: Insufficient/missing quality of the new system	32
System Design	34
System description	34
Design choices	36
Functionalities	38
System Development	40
Programming languages and frameworks	40
System description	41
Login system	42
Testing	42
Test Plan	43
Approach	43
Unit testing	43
Integration testing	43
Usability testing	43
Features to be tested	44
Item pass/fail criteria	45
Risks and contingencies	45
Schedule	45
Approvals	45
Test results	46
Unit testing	46
Integration testing	46
Usability testing	46
Use cases user side	47
Use cases administrator side	47
Feedback user side	47
Feedback administrator side	47
Future Work	49
Support of the system	49
Integration with DMS system	49
Evaluation	50
Planning	50
Sprint 1 - Requirement analysis / Design (9th - 18th February 2022)	50

Sprint 2 - Design / development (28th February - 11th March 2022)	50
Sprint 3 - Development (14th - 25th March 2022)	51
Sprint 4 - Development (28th March - 8th April 2022)	51
Sprint 5 - Closure (11th - 20th April 2022)	51
Responsibilities	51
Team evaluation	52
Deliverables	52
Functional requirements	52
Non-functional requirements	53
Conclusion	54
References	56
Appendix	57
Mockups	58
Usability testing records	62
Interaction scenario	62
User side	62
Admin side	62
Setup	63
Procedure	63
Questions	63
Final Product	64
Client interface	64
Administrator Interface	66

Introduction

History of video games can be traced back to 1947 when the first known electronic video game was introduced - Cathode-ray tube amusement device (Cohen, *Cathode-Ray Tube Amusement Device: The World's first video game?* 2019). Then came the 1970s and 1980s, when arcade video games became extremely popular in the United States and most other parts of the world. With the advancement of computer technology and graphics, video games evolved into more than just a form of entertainment for friends and family. All of this culminated in the first gaming competition (Good, *Today is the 40th anniversary of the world's first known video gaming tournament* 2013), held in 1972 by Stanford University, with the winner receiving a yearly subscription to the Rolling Stones. For many fans, it became a way to make money by streaming their gameplay or simply compete for awards with other fans. The first esports event, the Red Annihilation tournament (Baker, *Meet Dennis 'thresh' fong, the original pro gamer* 2018), held in 1997, marks the pinnacle of the video game industry's development.

Electronic sports, which abbreviated as esports or e-sports. It denotes a new and modern subdomain of the sporting world. It takes the form of competitive gaming with professional aspects such as wages, substantial awards ranging from a few thousands to a couple of millions, training and development facilities, and, as with any sport, coaches and players who make this their full-time job. On July 25, 2001, Russia became the first country to recognize it as a sport ("cybersport"), followed by China in 2003 and the rest of the world in the following years. The most notable esports event was the Free Fire World Series 2021 Singapore, which was watched by 5.41 million people.

Esports in Twente has been booming since the Esports Team Twente was founded two years ago, on the 3rd of March 2020. Esports Team Twente was founded as the newest official University of Twente student team and were off to a great start with exposure on social media, the radio and an article in Tubantia. However, due to the Pandemic that happened shortly afterwards, the University of Twente was closed to follow the government regulations. Although this slowed the momentum down considerably, Esports Team Twente has truly established itself as an upcoming esports organisation in the middle of the Dutch gaming and esports network due to the quick adaptation of the situation (Esports Team Twente, *Year One in Review* 2021).

Esports Team Twente is the 7th official student team of the University of Twente. The organisation has taken up the challenge to bring gaming to the next level by combining research and engineering of multiple disciplines with the aim of improving esports performance. Esports Team Twente participates in various tournaments and competitions, performs unprecedented research on the many factors that influence performance and functions as a showcase to the rest of the world of the esports University of Twente. The organisation consists of a team of students from in and around Enschede (not just limited to the University of Twente) that are working on Esports research, competition and awareness. The organisation works on various challenging projects related to Esports, with the aim to achieve the primary goal: "Taking esports to the next level!"

Esports Lounge Twente (EL) is a collaborative effort between Esports Team Twente (University student-team), Blueshell (University of Twente Esports organisation) and the Student Union. The lounge will be built in the Bastille and will be UT's very own esports facility. After realisation, the lounge will include the following:

- Facilities for esports competition, tournaments and training.
- A community hub for students of the University of Twente and union card holders.
- Professional broadcasting equipment for streaming and recording.
- Multifunctional spaces for different types of events.

Students can reserve the lounge facilities when the lounge is open. Esports Lounge Twente seeks to build its own reservation system to create, manage and view reservations for the lounge.

Domain Analysis

This section analyses the domain of the specified system, which explains the process by which information used in developing software systems is identified. The main objective of domain analysis is to document all key information such that the development could be carried out with ease and making sure that reusability and future development is plausible since the context in which the software system is built are thoroughly explained (Prieto-Díaz, *Domain analysis* 1990).

Introduction

The system is brought into the reservation domain, and it concerns the process of reserving a device or a facility in the newly constructed Esport Lounge, as well as the option of reserving the entire lounge for events. It should also assist administrators in changing the schedule and device and facility availability, as well as accepting or denying requests and bookings. The initial plan of how the lounge will work was to make reservations by phone or in person. Also this would have required a lot of paperwork for bookkeeping, following schedules and analysing statistics. Since the mentioned processes are error-prone and require a training period for the admins, the idea of investing in a reservation system came up.

Glossary

The terms device, facility, and request are used in this domain. Devices can include a PC/Laptop/console system (e.g., Playstation/Xbox/Nintendo)/Vr-set and many more depending on what types of devices the lounge introduces. Broadcasting or casting space is included in the definition of a facility. This word can be extended in the future to incorporate newly defined facilities. In this realm, request refers to reserving a portion (room) or the entire lounge for an event.

General knowledge about the domain

Students can participate in a variety of events organised by Esports Team Twente, Blueshell, and the Student Union. They also offer a number of services to students. Blueshell conducts gaming events for the students' amusement, while Esports Team Twente researches and develops gaming talents on a professional level. As a result, the system should accommodate not only casual gamers, but also large-scale events and other activities.

Users, operators and concerned parties

The system is developed mainly for the needs of common students and the needs of Esport Team Twente, Blueshell and Student union. The client of this project is mainly Esports Team Twente but it also represents the interests of Blueshell and Student union. The system will be operated by staff employed by Esports Team Twente. The staff would consist of students from

the University of Twente. They are the most prominent stakeholder when it comes to processing reservations and requests.

Students from the University of Twente will use this system. When it comes to the reservation process, they are the most visible stakeholders. Other parties, such as the University of Twente, are primarily concerned with the system's ethical use and whether the system is available to all parties (students and the three above-mentioned organisations). More about the stakeholders and their interests and use of the system will be discussed below in the "Stakeholder" section.

The environment

The client provided no specifications for the software or tools that will be used to develop the system, therefore it had to be started from scratch. The lounge itself is still under construction, therefore we could not observe the processes. The only environment specific requirement requested by the client was to use the University of Twente API for the login process. We chose to use technologies that were familiar to the developer team. Heruko, a private cloud provider, hosts the web application. We used Spring Boot with Gradle and a PostgreSQL database to store the reservations and other useful information, such as the state of the interactive map, during the application's development. For the backend we decided to use Java as the programming language.

Tasks and procedures performed

There are currently no existing procedures, so we had to design the system's business processes in collaboration with the client.

The administrator's point of view is critical. The admin staff will be able to view and filter all reservations after logging in. They can also change the map/schedule and layout of the application. It all depends on what they do after logging in. When they view the bookings, they have the option to approve, decline, or cancel the reservation, depending on the type.

Another point of view is that of a student. The student will be able to login and view his or her bookings, make a booking (reservation), or check notifications. If a student wishes to make a reservation, he or she may do so for PC(s), broadcasting space, practice booth, or the entire lounge. Furthermore, these reservations can be cancelled if there are reasonable reasons to do so.

Similarities between other UT software

As the project of the gaming lounge is fairly new and still in development there is no existing system that can be used by the staff of the lounge. So we had to develop the whole booking system from scratch, without any previous stepping stones such as designs, databases, or software. As a result of that, there are parallels with other booking/reservation systems used at the university, as it would make the system a bit more user friendly and easier to navigate and operate. The reservation system for study spaces on campus is an example of such a system.

Because the client is from Esports Team Twente and the system was created primarily for their benefit, the layout is similar to that of their website and other deployed software.

Stakeholders

Onion model stakeholder

This section presents the different stakeholders that are present in the system such that requirements are captured from all present stakeholders points of view (Alexander, *A Better Fit - Characterising the Stakeholders* 2004).

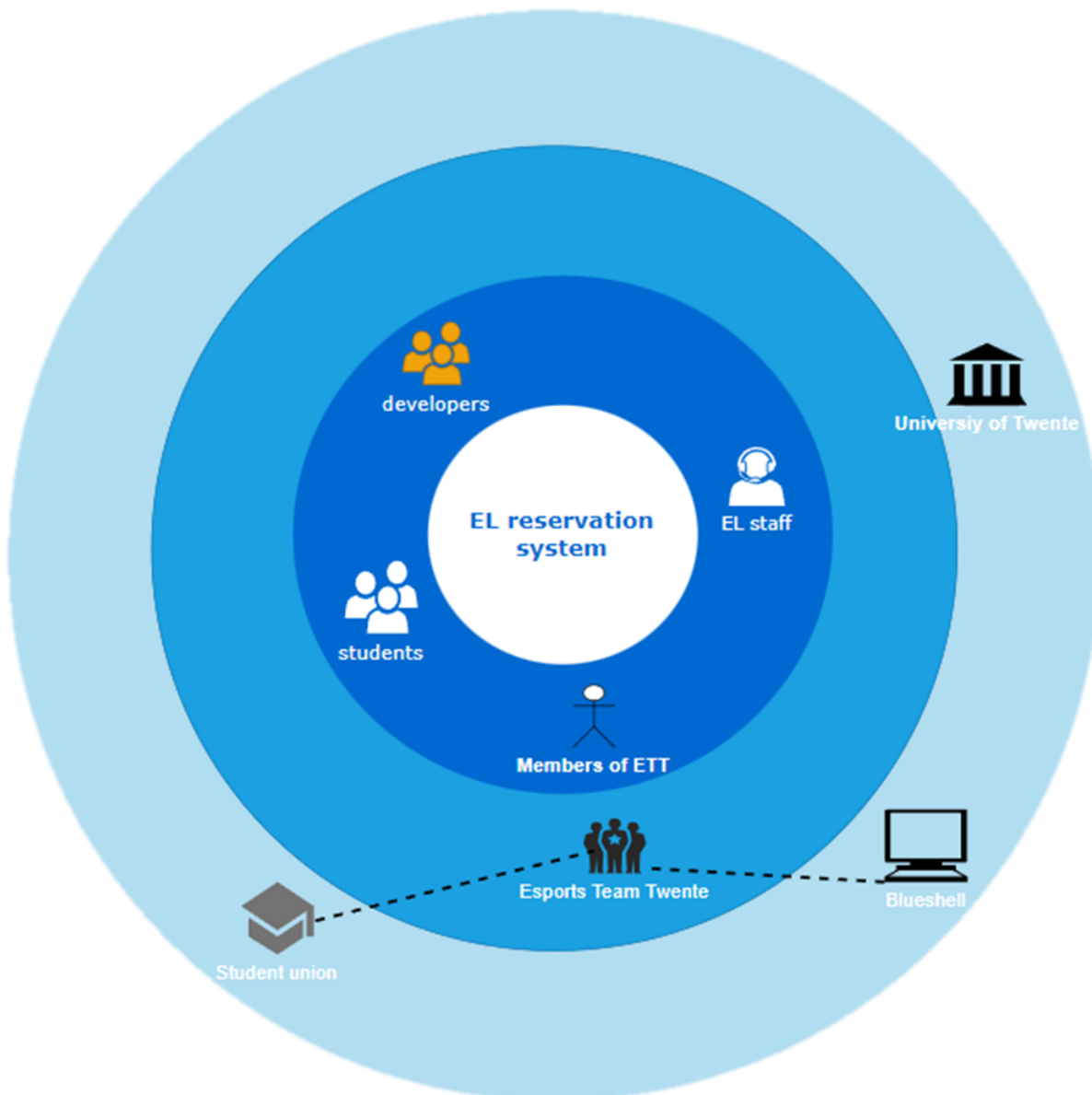


Figure 1: Onion model stakeholder (Alexander, *A Better Fit - Characterising the Stakeholders* 2004).

Direct stakeholders

Students

Those are the end users of the systems and as such they are in the closest to the product circle of the onion model. They are going to use the system to make reservations and requests.

Developers

Those are the providers of the reservation system and as such they are closest to the product circle of the onion model. They are going to provide a functional and robust reservation system as an end product of their work.

EL staff

These are the governance of the system. As such they are going to take care of the business processes such as cancelling reservations, accepting requests, adjusting opening times and adding/removing devices and facilities.

Client

This is the initiator of the project. It aggregates three organisations (Student union, Esports Team Twente and Blueshell) and their common purpose - to obtain a robust reservation system for the lounge. The client provides input (ideas and wishes) on how the system should work and look.

Indirect stakeholders

Student union

A University of Twente student organisation that empowers the University of Twente students in regard to university policies. It also organises events and provides services for University of Twente students. Students can participate in the union with a subscription and with that they get access to what Student Union is offering as services and facilities. With those cards students can access the lounge as well.

Esports Team Twente

Esports Team Twente is the 7th official student team of the University of Twente. They participate in esports competitions all around the world, develop and promote esports in the region of Twente. It combines research and engineering techniques from multiple disciplines to improve esports performance.

Blueshell

This is the student esports and gaming association of the region of Twente. They organise offline and online events and provide opportunities for competitive gaming.

University of Twente

University of Twente is a technical university located in Enschede, Netherlands. In this project the university is an indirect stakeholder and in that category is the one that is the farthest away from the product. It is concerned mainly with the login process to the system because the product is going to use university credentials and also this is the organisation that houses and finances all of the organisations mentioned above.

Requirement Specification

This section describes the requirement specification that the system needs to have. The requirement specification is heavily based on Agile methodology (Dingsøyr et al., *A decade of agile methodologies: Towards Explaining Agile Software Development* 2012), the iterative loops of development also occur in the requirement specification. Hence, these requirements need to be clearly formulated and prioritised when specifying them, in which several methods are used in order to achieve this.

Agile Project Management Approaches for Requirement Specification

The project management methodology that is going to be used is the Agile methodology (Dingsøyr et al., *A decade of agile methodologies: Towards Explaining Agile Software Development* 2012). Thus, the requirements for the system are defined in iterative loops, which ensures that the system has all the functionalities that the users need and the clients want. The specific framework that is going to be used is Scrum framework (Adi, *Scrum method implementation in a software development project management* 2015). We chose this method as the Esport Team Twente Lounge Reservation System is at its core a brand new idea and system with a lot of changes and experimentation for the final system. The scrum method allows for the scope to be flexible in the development process and thorough involvement of the stakeholders to achieve a system that fits the identity of the system.

As specified, the project methodology will work with iterations, also known as sprints, which is iterated biweekly. Every iteration, a new section of the system will be completed, and there will be a weekly meeting with the Esports Team Twente to discuss the progress and direction of the development. Moreover, there will also be a meeting with the University of Twente supervisor (Dr. Vadim Zaytsev) every two weeks to discuss the progress, remarks, or queries. The development team will hold daily scrum meetings to forego miscommunication and will produce a sprint retrospective each iteration to review performance.

Requirement Formulation

The requirement formulation will be based on the SMART method requirements (Mannion & Keepence, *Smart requirements* 1995), this method guides the formulation of the requirements such that they are specific, measurable, attainable, reasonable, and traceable. This results in a simple and straightforward method that ensures requirements are clearly identified and formulated, independent of assumption and ambiguity. In formulating requirements, this method is crucial such that the functionalities that are going to be developed and tested correlates to what the users' needs and clients' wishes.

Requirement Prioritisation

From the requirements that have been defined according to the SMART method, these requirements are going to be prioritised based on the importance of the respective functionality and the urgency for different stakeholders. The requirement prioritisation will be based on the MoSCoW prioritisation method (Craddock, *The DSDM Agile Project Framework* 2014). The method prioritises requirements based on the importance of the requirement for the system, from a “Must”, “Should”, “Could”, and “Won’t”. With these criterias, the requirements could be prioritised accordingly.

Stakeholder Requirements and System Requirements

As mentioned in a previous sections “Domain Analysis” and “Stakeholders”, there are several parties (stakeholders) that want to get something out of the system. Thus, several requirements are going to be formulated according to the needs of these stakeholders. These requirements are called the “stakeholder requirements”. Based on these requirements, “system requirements” are identified to fulfil a number of functionalities that need to be present in the system according to the stakeholders needs. These requirements could be found in the section “Requirement Analysis”

System Proposal

This section explains the system proposal in which it explains the conceptualization and realisation of the project to the client in several meetings that we conducted throughout the project. The proposal consists of requirements proposal, mockups proposal, system increment, a usability testing proposal and a final presentation of the system to the client. Each of the proposals happened in a sequential timeframe in which it was also presented to the client.

Meetings with client

The client was questioned during the initial meeting to assess the project's scope and requirements. This was done to ensure that the parties' interests were adequately represented. These discussions took the form of a Q&A session, in which we first gave a general overview of the current concept before asking specific questions. There were also some open-ended questions and opportunities for the interviewee to ask their own questions or express their own ideas.

The meetings were held every week in order to ensure continuous feedback and involvement from the client such that the system has all the functionalities that the users need and the clients want. This is also due to the fact that we are following the Agile methodology, which was explained in more detail in the previous section "Requirement Specification".

Requirements proposal

As specified, in the initial meeting, the client was asked numerous questions to determine the requirements that he had for the system. The client already had some requirements specified for the system to be implemented. Thus, the meeting was held in order to dig deeper into these ideas of the requirements and if there is anything else that the client wished to be implemented in the system. Moreover, the client is also open to new ideas, in which some of the new ideas are presented to him. These requirements served as a guidance in the development phase. It is also important to mention that since we are following the Agile methodology, some new requirements may emerge in each project iteration and some existing requirements may alter as the client's needs may alter in the next phase. More about the requirements and the risks that are connected to them will be discussed in the next sections "Requirements Analysis" and "Risk analysis".

Mockups proposal

After the requirements are defined and proposed, these were used to visualise the initial concept. The mockups were proposed and presented to the client such that the client could get the look and feel of the system and be able to identify what he likes and dislikes such that it creates a feedback loop. It also helps the client in setting and/or altering the requirements and ideas for the system.

The mockups that were proposed helped the client to provide more detailed information about their requirements and ideas, for example, where the sections of the reservation needs to be and what each page should contain. The client also mentioned that the design of the system needs to adhere to the style guide of the organisation itself, Esports Team Twente.

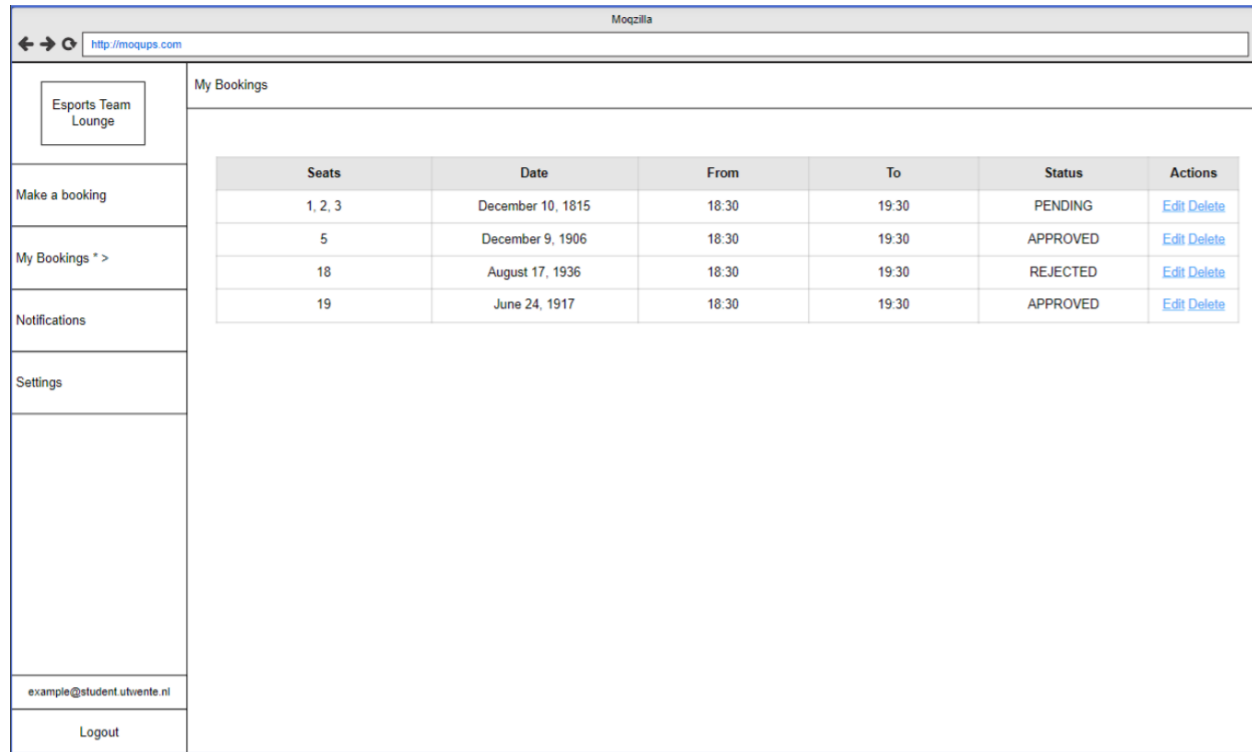


Figure 2: Example mockup of the main page in the client screen, showing all of the bookings he/she made.

More of the mockups that were proposed can be found in the “Appendix” section under the “Mockups” subsection.

Preliminary Design

Following the mockups, a number of system modelling were made such that the developer team and the clients could see a preliminary design of the actual system that is going to be implemented later on. In addition, it also helps the developer team in the conceptualization of what is going to be built and foresee any future problems that might come along with it.

In the figures below, a number of system modelling that have been designed before developing the actual system can be seen. This is done in order to adhere to the SCM (Software Configuration Management) processes. It is important since Software Engineering (SE) would be based not only in development, but also in other issues, such as architecture, building, evolution and so on (Estublier, *Software configuration management* 2000). UML diagrams help in a way that it captures the early phase of Software Engineering (SE), namely in system

analysis and system design. Moreover, UML diagrams are able to help in visualising differences between documents in the early phases with an actual image and not only with lines of text, for example, in a file representing a class diagram, each class might be represented by a few lines of text (Ohst et al., *Differences between versions of UML diagrams* 2003).

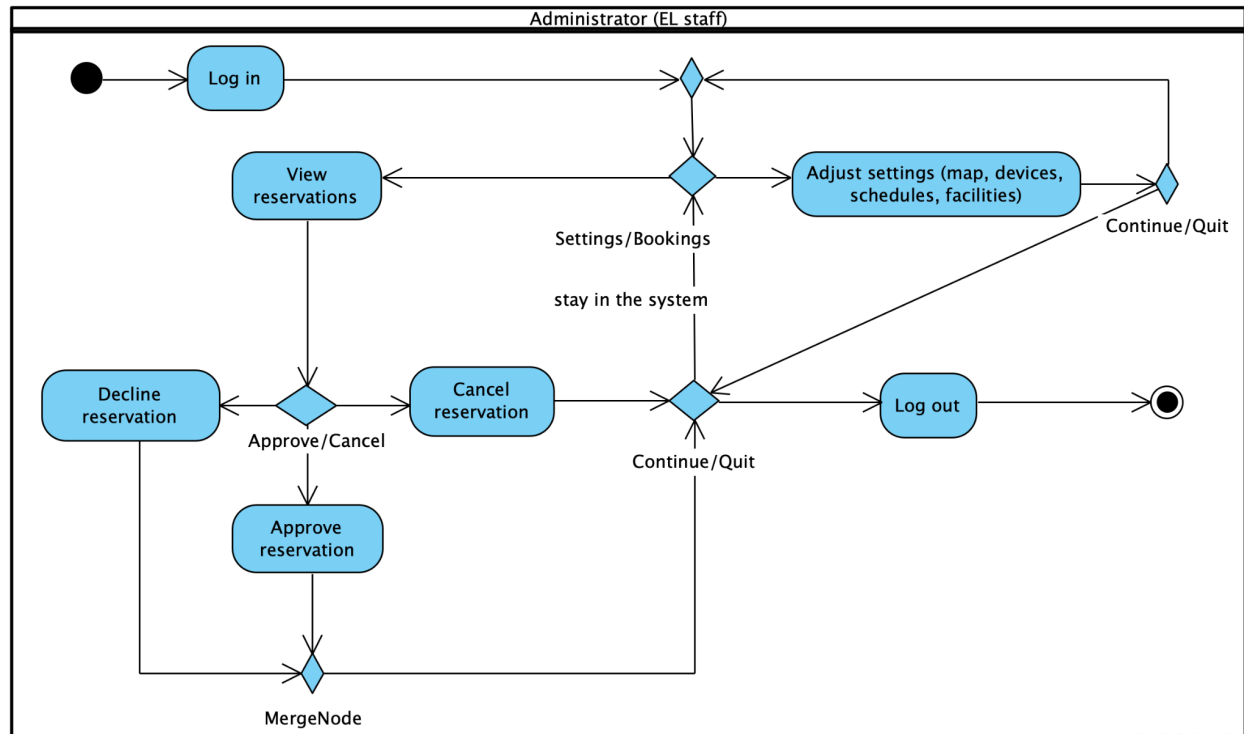


Figure 3: The activity diagram of the system showing the possible administrative actions.

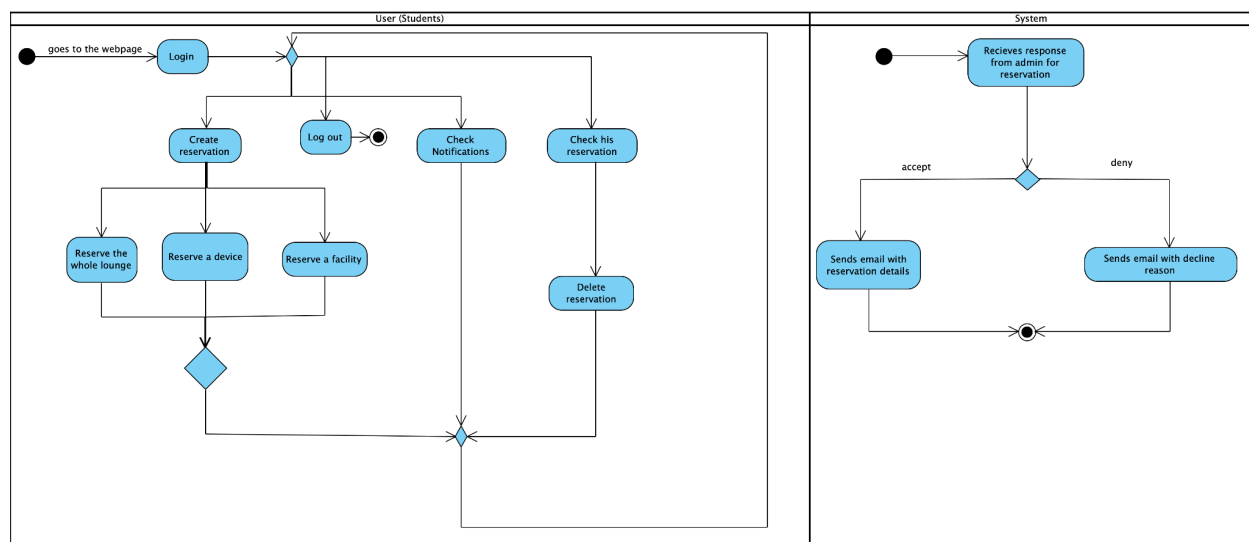


Figure 4: The activity diagram of the system showing the client perspective and the responsive actions taken by the system.

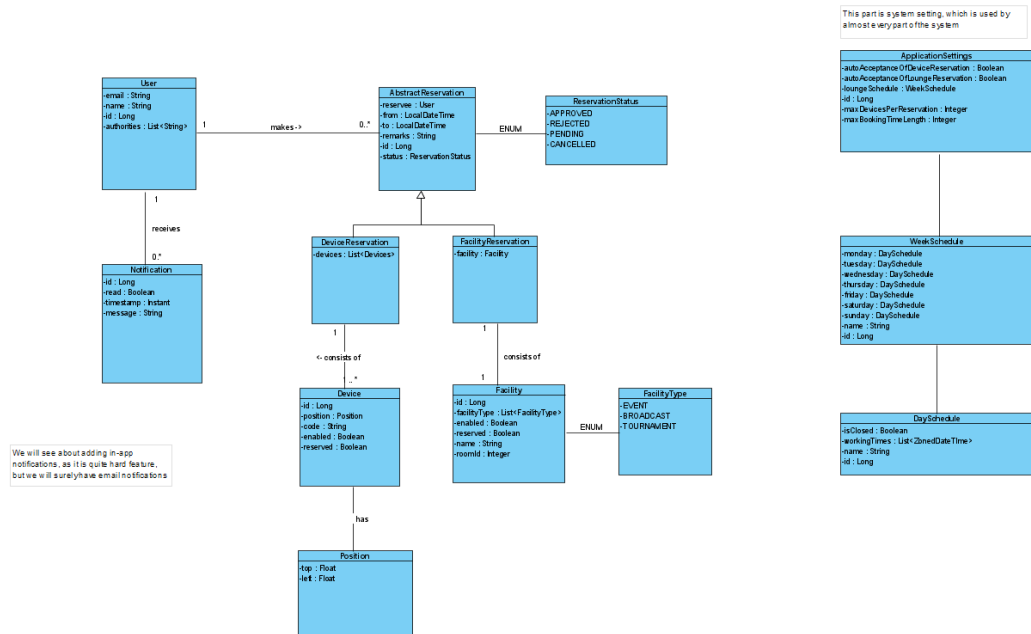


Figure 5: The class diagram of the system showing the application structure.

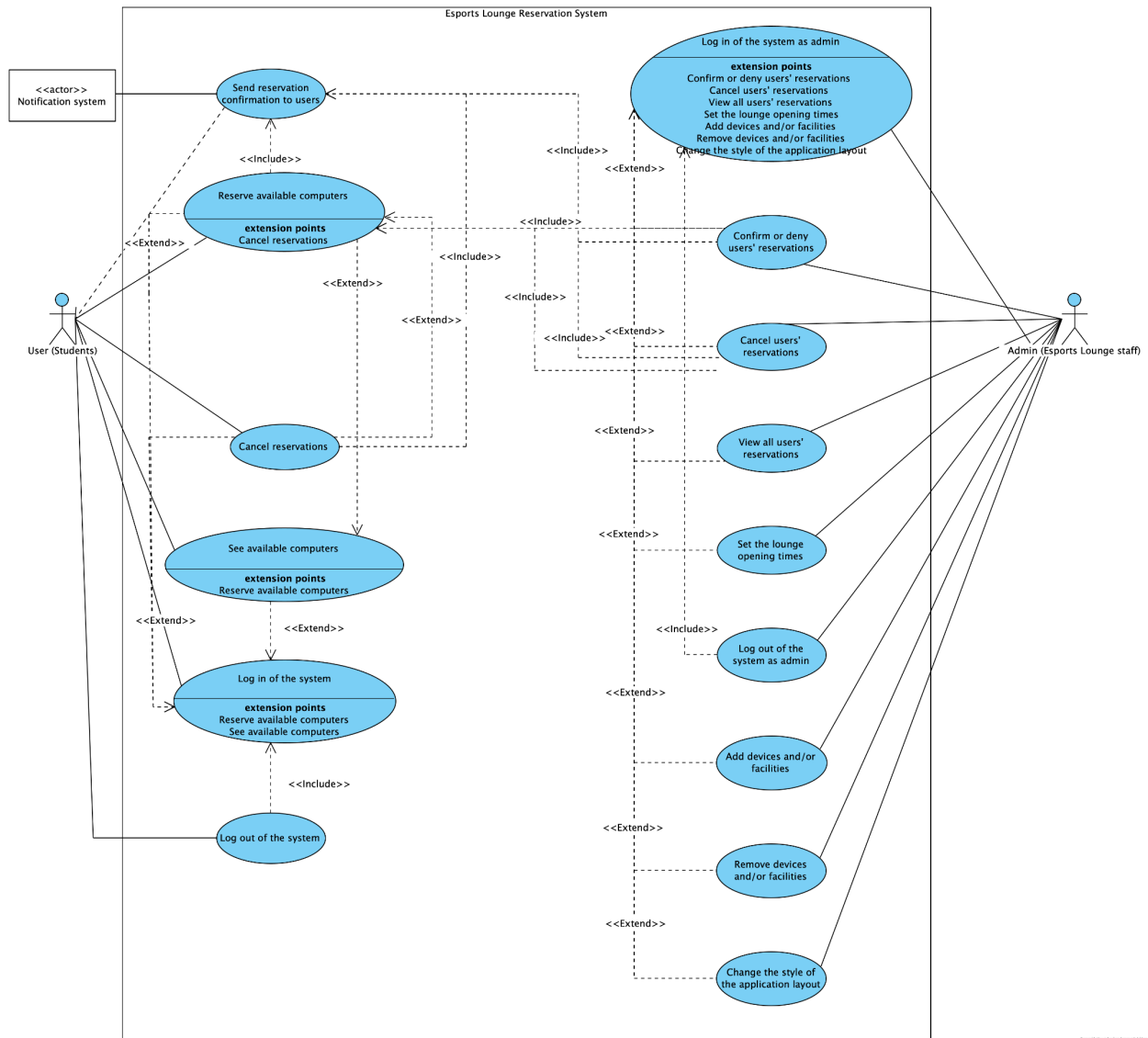


Figure 6: The use case diagram of the system showing an initial version of the requirements.

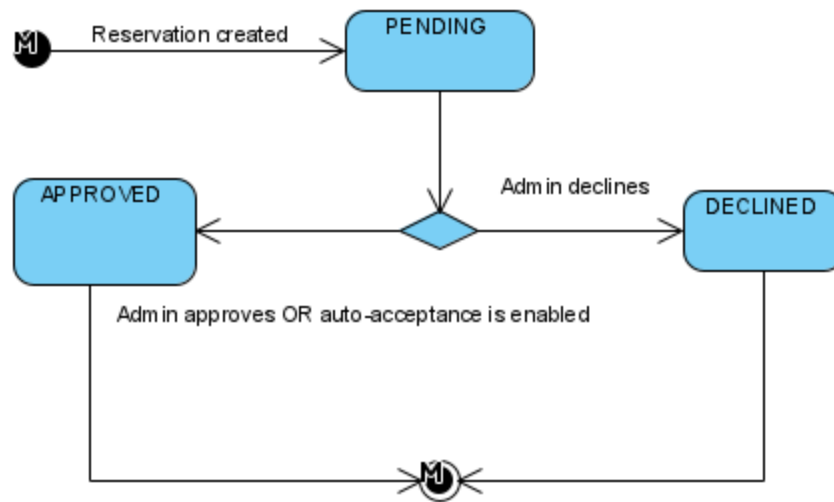


Figure 7: The state machine diagram of the system showing possible states of reservation.

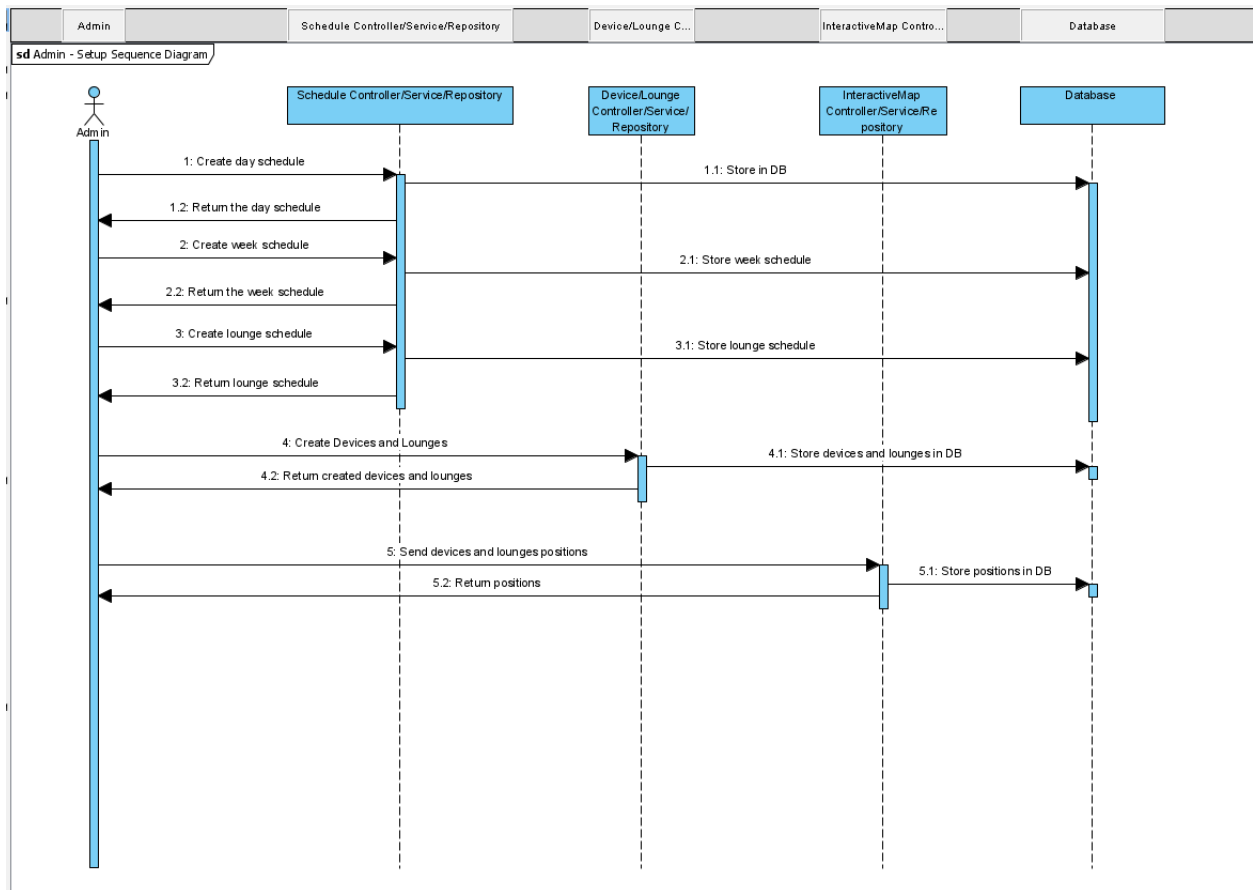


Figure 8: The sequence diagram of the system showing the flow of the application when setting up the schedules.

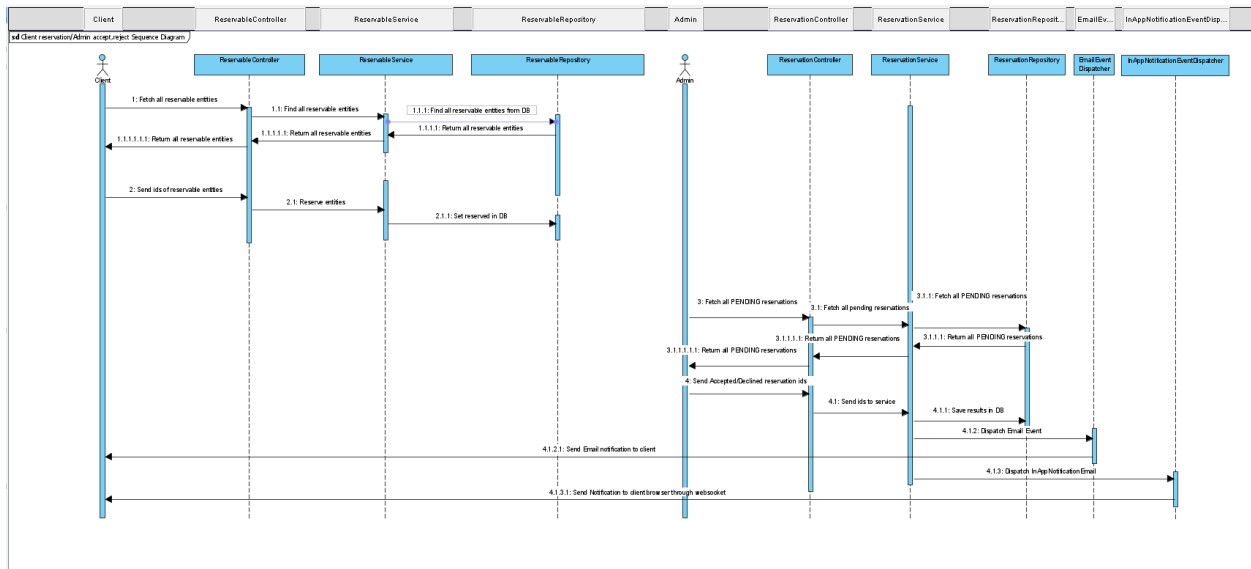


Figure 9: The sequence diagram of the system showing the flow of the application when making a reservation.

System increment

At the end of every iteration of the project, the system is incremented according to the planning that has been defined. In each of the system increments, the client had the chance to use the system itself by either a live demonstration of the system or explore on the client's own via the system that has been hosted in a website. This gave the client a chance to experience the system on his own and to confirm if it fit the needs. This also enables the client to communicate any suggestions for enhancements which will be then integrated into the requirements.

The system increment was very beneficial because it allowed the client to explore through the system, giving them a sense of how the final product will look and feel like. This allowed the client to provide more detailed comments if something was confusing or was not as what the client expected. The result of the process is that the client was in general pleased with each of the system increments and provided some feedback and input, such as clarifying the requirements and adding new requirements. For the final system, the client added a comment that the system was identical to his ideas and that the final result was as he had expected it to be.

The system increment planning for each iteration can be found in the "Evaluation" section under the "Planning" subsection.

Presentation proposal

The system was presented to the client at the end of the project. This presentation demonstrated to the client the system's major purpose, which is the ability to reserve the newly constructed Esports Lounge's resources. It also displayed the resulting application to Esport Lounge Staff personnel, who will also be using the system as administrators.

Requirement Analysis

This section describes the analysis of the requirements from the stakeholders. The analysis of the different requirements for the system contains “user requirements” and “system requirements”. The criterias of analysis for the requirements are based on the system’s capability in completing the correct functionalities, the system’s correctness in executing functionalities, the system’s availability, or the system’s speed rate in executing functionalities.

Stakeholder requirements

1. As a student, I want to log in and out of the system.
A student must be able to log in and out to use the system using their student credentials via University of Twente login portal system in order to reserve devices and/or facilities, such as computers, broadcasting room, or the entire lounge.
2. As a student, I want to see available devices and/or facilities.
A student must be able to see the available devices and/or facilities, such as computers, broadcasting room, or the entire lounge, that they can reserve at a given date and time.
3. As a student, I want to reserve available devices and/or facilities.
The primary objective for students in using this system is to be able to reserve the available devices and/or facilities, such as computers, broadcasting room, or the entire lounge. When this requirement could not be achieved, then the system loses its core functionality.
4. As a student, I want to cancel the reservation.
A student must be able to cancel the reservation that they have made, such as cancelling their confirmed reservations. The deletion could only be done when the student has been confirmed of their reservation, which means that their initial reservation needs to be accepted first by the administrator before being able to cancel the reservation.
5. As a student, I want to be informed of my reservation via a confirmation message or email.
Whenever a student has made a reservation and/or changes in the reservation, such as making new reservations or cancelling the reservations, the student must be informed of their reservation with a confirmation email to their student account or to their phone numbers.
6. As an Esports Lounge staff, I want to log in and out of the system as an administrator
An Esports Lounge staff must be able to log in and out to use the system as an administrator using the employee credentials in order to manage the students’ devices and/or facilities reservations, such as computers, broadcasting room, or the entire lounge.

7. As an Esports Lounge staff, I want to confirm devices and/or facilities reservation requests.
An Esports Lounge staff must be able to confirm the devices and/or facilities reservation requests, such as computers, broadcasting room, or the entire lounge, that were made by the students from the reservation system. This also holds every time the student makes any changes to the reservation.
8. As an Esports Lounge staff, I want to deny devices and/or facilities reservation requests.
An Esports Lounge staff must be able to deny the devices and/or facilities reservation requests, such as computers, broadcasting room, or the entire lounge, that were made by the students from the reservation system. This also holds every time the student makes any changes to the reservation.
9. As an Esports Lounge staff, I want to be able to have the option to cancel a student's reservation.
An Esports Lounge staff must be able to have the option to cancel the devices and/or facilities reservation that the student has made in case an unexpected event arises. There could be an unexpected event, such as match events or staff unavailability that happens after the reservation has been confirmed by the administrator.
10. As an Esports Lounge staff, I want to be able to view all made reservations in an easily readable overview/timetable.
An Esports Lounge staff must be able to view the reservations that students have made in an overview in the form of a timetable. This overview will give a better understanding of the lounge state on the reserved and available computers or rooms for the staff.
11. As an Esports Lounge staff, I want to be able to have the possibility to set the times that the lounge is open and reservations should be possible.
An Esports Lounge staff must be able to set the opening time and status of possible reservations. The opening time of the lounge would give an indication of when students could reserve the devices and/or facilities, such as computers, broadcasting room, or the entire lounge. Moreover, the staff must be able to indicate if devices and/or facilities are not available to be reserved because of availability reasons, such as the computer is being repaired or the room is being renovated.
12. As an Esports Lounge staff, I want to be able to add devices and/or facilities.
An Esports Lounge staff must be able to add devices and/or facilities in the system such that these devices and/or facilities could be reserved by the students and appear in the students' reservation system interface. These devices and/or facilities could be, for example, computers, private room spaces, or broadcasting spaces.
13. As an Esports Lounge staff, I want to be able to remove devices and/or facilities.

An Esports Lounge staff must be able to remove devices and/or facilities in the system such that these devices and/or facilities could not be reserved by the students and does not appear in the students' reservation system interface. These devices and/or facilities could be, for example, computers, private room spaces, or broadcasting spaces.

14. As an Esports Lounge staff, I want to be able to change the style of the application layout

An Esports Lounge staff could change the style of the application layout in the system, such as fonts, colours, and logos, to be matched with the style of different teams or events.

15. As an Esports Lounge staff, I want to be able to provide a limit of the booking time and maximum number of devices being reserved per booking.

An Esports Lounge staff must be able to provide the up-mentioned limits, leaving the responsibility for validation to the application.

System requirements

Functional requirements

1. The reservation system must be able to let students log in and out with their UT account or DMS-account (Unioncard).

The mechanism which is used is the University of Twente account or DMS account login. This is easy to understand and the users do not have to remember another username and password.

2. The reservation system must be able to reserve devices and/or facilities.

The system must be able to reserve devices and/or facilities for the specific students that made the reservation. The reservation of the devices and/or facilities could be a single computer for personal use, a block of computers for their practice space, broadcasting spaces, or the entire lounge.

3. The reservation system must be able to let students cancel their reservation.

The system must be able to let students cancel the reservation, such as cancelling their confirmed reservations. The deletion could only be done when the student has been confirmed of their reservation, which means that their initial reservation needs to be accepted first by the administrator before being able to cancel the reservation.

4. The reservation system must be able to inform students about the reservation confirmation by messages or email.

The system must automatically send a confirmation message to the student about their reservation via messages or email whenever the student has made a reservation and/or

changes in the reservation, such as making new reservations or cancelling the reservations.

5. The reservation system must be able to let the Esports Lounge staff a way to log in and out as admin.

The system must be able to have an interface for administrators to be able to log in and out of the system as an administrator using the staff credentials in order to manage the students' devices and/or facilities reservations.

6. The reservation system must be able to give the Esports Lounge staff the option to cancel a reservation.

The system must have an option to cancel students' devices and/or facilities reservation. The system must also ask the reasoning of the reservation cancellation, for example, match events or staff unavailability that happens after the reservation has been confirmed by the administrator. The student should then get a confirmation message when the cancellation occurred.

7. The reservation system must be able to let the Esports Lounge staff easily view all made reservations in an easily readable overview/timetable.

The system must have an overview of all the reservations that the students have made in the form of a timetable such that the Esports Lounge staff could see the lounge state of the reserved and available computers or rooms.

8. The reservation system must be able to let the Esports Lounge staff confirm devices and/or facilities requests.

The system must send a confirmation to the administrator when students made a reservation on devices and/or facilities, such as computers, broadcasting room, or the entire lounge. The system should then have an option for the reservations to be confirmed by the administrator.

9. The reservation system must be able to let the Esports Lounge staff deny devices and/or facilities requests.

The system must send a confirmation to the administrator when students made a reservation on devices and/or facilities, such as computers, broadcasting room, or the entire lounge. The system should then have an option for the reservations to be denied by the administrator.

10. The reservation system must be able to let the Esports Lounge staff the possibility to set the times that the lounge is open and reservations should be possible.

The system must be able to have a schedule of when the lounge is open or closed, and in these open time periods, students could reserve the devices and/or facilities, such as computers, broadcasting room, or the entire lounge. Furthermore, the system must have a state of all computers and/or rooms if they are available or not available to be reserved

because of availability reasons, such as the computer is being repaired or the room is being renovated.

11. The reservation system must be able to add devices and/or facilities for students to reserve.

The system must have an option to add devices and/or facilities and in which these devices and/or facilities could be reserved by the students and appear in the students' reservation system interface given that these are available and have not been reserved by other students.

12. The reservation system must be able to remove devices and/or facilities for students to reserve.

The system must have an option to remove devices and/or facilities and in which these devices and facilities could not be reserved by the students and does not appear in the students' reservation system interface. They must have types, which in case of devices must be dynamically creatable with variable names and images.

13. The reservation system should be able register full facilities reservations within the DMS system of the University through an API connection.

The system should be able to automatically register the full room reservations in the DMS system of the University of Twente using the University of Twente API connection whenever there are fully made facilities reservations in the system.

14. The reservation system should have an interactive map with all the devices and facilities on it representing the real lounge.

The system should be able to show an interactive map that shows all the available devices and facilities computers on it based on the current floor plan of the lounge.

15. The reservation system could change the style of the application layout.

The system style, such as fonts, colours, and logos, could be changed to be matched with the style of different teams or events by the Esports Lounge staff.

Non-functional requirements

16. The system should handle the users' and/or administrators' login within 30 seconds

The system login should take a maximum of 30 seconds from the moment the user and/or administrator clicks the login button. The accuracy has to be very high, such that if someone logs in and gets to see another profile, the system made a huge mistake at the security level, namely privacy breach.

17. The system should use the Esport Lounge Twente style guide for the frontend.

The system style, such as fonts, colours, and logos, should use the same style of the Esports Lounge Twente style.

18. The system should send confirmation email/message for a reservation within 30 seconds.
The system confirmation email/message should take a maximum of 30 seconds from the moment the user makes a reservation. The accuracy has to be very high, such that if someone gets an incorrect confirmation email/message, the system made a huge mistake at the security level, namely privacy breach.
19. The system should be available 24/7 except in case of maintenance.
The system should be usable at any time by users and/or administrators such that they can manage reservations at any time. In case of maintenance, the system should inform the users and/or administrators at least 5 days before the maintenance.
20. The system should be able to handle at least 5000 requests concurrently, either by users and/or administrators, at the same time without degradation of performance
The system should be able to handle at least 5000 requests at the same time without any performance issues from the users and/or administrators side.
21. In case of a breakdown, the system should roll back all of partially made reservations.
The system should not save any reservations that have not been made and/or confirmed by the users if there is a sudden shutdown or breakdown, for example, when the server fails or when users' local machine disconnects.

Risk analysis

This section explains the conducted risk analysis and assessment associated with this project. This analysis is performed in order to increase the success rate of the project itself, as we can plan and mitigate high-risk upfront. We identified each risk based on its category, owner, likelihood, impact, and what action should be taken. The scale of likelihood indicates the probability of the risk occurring. Whereas the impact indicates how severe the risk is towards the organisation. The likelihood will be measured with a scale from 1 (lowest) to 5 (highest), and the impact will range between 5 levels as well, from insignificant, minor, moderate, major, and catastrophic, respectively. The risk owner column classifies who is responsible for controlling and monitoring the specified risk. While the action column specifies how we are going to deal with the risk. The last column is for the risk rating, which is the combination of likelihood and impact. The risk rating will be ranging from 1 (lowest) until 25 (highest).

Needless to say, all the risks mentioned below are really dynamic, and we consider it essential to monitor and adjust accordingly throughout the entire project process.

ID	Description	Category	Owner	Likelihood	Impact	Risk Rating	Action
R1	Fail to connect with the external login system	Technical	Developer	4	Major	16	Mitigate

R2	Fail to preserve correct dates and times	Technical	Developer	3	Major	12	Mitigate
R3	Time schedule exceeded	Project Execution	Project Manager	3	Major	12	Mitigate
R4	Insufficient/missing quality of the new system	Technical	Developer	2	Major	8	Plan

R1: Connecting the new system with the external login system

Description

The most critical issue is having our software system connected with the University of Twente API to be able to use their login system. Forwarding the credentials must be done in a fully encrypted manner, also securing the piece of the system responsible for this task, so no external actions can affect it.

Category

We classify this risk under the technical category since the integration for the system suggests mainly on the organisational and technical side. The reason for this is because we might not find a way to complete the task because we are not able to contact the external people responsible for the login system.

Owner

The owner of this risk is the developer team of the project. The developer team are the ones responsible for integrating the Universities' login system with the new reservation system. The integration needs to be completed by the developer team.

Likelihood

We classified this potential risk occurrence on a scale of 4 out of 5. The reasoning behind this is due to the fact that this project is quite a new implementation of the system, where we will be also using an external login system. A lot of parties are also going to be involved in this project which makes it even more difficult to organise the technical side.

Impact

Moreover, the impact of this risk is quite major. If this risk is to happen, then the new reservation system would not be available to be used by the users, which are the students of the University of Twente.

Risk Rating

This risk has a rating of 16 out of 25, which is quite high. Therefore, we are planning to mitigate this risk by the action explained below.

Action

The mitigation strategy that we will use to prevent this risk from becoming reality is to do partner research. The network of contacts that the client Esport Team Twente has, should make it easier to find developers which already know about the documentation of the external login system that is connected to the University of Twente login system. They could potentially be interested in helping with the implementation of the new reservation system. Furthermore, if we are not able to connect the new system with the external login system, we will add our own login system for the new system that does not relate to the external login system.

R2: Preserving correct dates and times

Description

The issue is regarding preserving correct dates and times, as this is one of the main elements in our application. This can become a serious problem, interrupting the work of our clients if not being handled properly. Since every browser uses the device's local timezone, we will convert them to UTC according to ISO_8601 standard and convert to the client's local timezone. This action will make us the controllers of dates and times, therefore the risk will be minimised.

Category

We classify this risk under the technical category since preserving correct dates and times for the system suggests mainly on the technical side.

Owner

The owner of this risk is the developer team of the project. The developer team are the ones responsible for preserving correct dates and times for the new reservation system. The feature needs to be completed by the developer team.

Likelihood

We classified this potential risk occurrence on a scale of 3 out of 5. The reasoning behind this is due to the fact that this project is quite a new implementation of the system, but fortunately, we have documentation and good practice in accordance of ISO_8601 standard.

Impact

The impact of this risk is quite major. If this risk is to happen, then the users might not be reserving the correct dates and times which will lead to chaos for the administrator, which are the Esports Lounge staff.

Risk Rating

This risk has a rating of 12 out of 25. This is a moderate one, however, we still think that we need to mitigate this risk considering the impact it might have on. This mitigation can be achieved by means below.

Action

Since the consequences of this risk are quite severe, we have to mitigate this risk. The mitigation strategy that will be used to prevent this risk from becoming reality is to do best practice research. The available documentation ISO_8601 standard already covers the basics

on how the issue could be mitigated. Therefore, these practices can be studied and applied in the new reservation system.

R3: Time schedule exceeded

Description

This risk is about the expected duration of the project execution, occurring if the project takes longer than it should be. This also tightly knit with the above risks, as if the above risks occurred, most likely this risk will also occur to fix other issues. Moreover, this is tied to the fourth risk where we would have insufficient/missing quality of the new system if the time schedule has exceeded the deadline.

Category

We classified this risk under the project execution category since it is related to how effective we can plan and execute the project within the given timeframe.

Owner

The owner that suits best to take this risk is still the project manager, as the project manager will also be the one making the schedule for the project.

Likelihood

We considered the probability that this risk happens is at level 3 out of 5. We are aware that this project will involve a lot of different main stakeholders in its process. Therefore, there will be highly reliant situations on each other, which will lead to unforeseen schedule delay.

Impact

We put a major impact on this risk since the impact of delay for the current organisation without the new system is significant.

Risk Rating

This risk has a rating of 12 out of 25, the highest risk that is mentioned in the risk analysis section. This risk will be the priority when addressing the project. We are planning to mitigate this risk by the action mentioned below.

Action

We plan to mitigate this risk by making a realistic activity estimation on each milestone and deliverable. Making the project scope clear is also required. Unclear project scope might lead to different products delivered than the client expects. In consequence, a rework on the product should be done which will take more time, causing a delay. A daily meeting with the stakeholders for each milestone achieved is also desirable.

R4: Insufficient/missing quality of the new system

Description

This risk covered the case of incomplete qualities of the new system. This risk also accounts for the functional and non-functional qualities of the new system. For example, functional quality could be something like incomplete features, and nonfunctional quality would be security for the new system.

Category

We classify this risk under the technical category since the quality of the system suggests mainly the technical side.

Owner

The owner of this risk is the developer team of the project. They are the ones designing and executing most of the project functionalities. Insufficient or missing quality should be discussed with the development team.

Likelihood

The probability of this risk to occur is 2 out of 5. This risk is not likely to occur since we are working with the agile development methodology, which requires a certain milestone and deliverables to be presented every week. This will make sure all the qualities meet the client's expectations. A good communication process between the developer and the stakeholders will also lower the chance of this risk happening.

Impact

However, for the impact, we would categorise this as a major one. If the final deliverable has some quality issues or missing functionalities, the system might fail at any point and cannot be used by the organisation. This would greatly affect the organisation's performance.

Risk Rating

This risk has a rating of 8 out of 25. This risk is considered a moderate one and will be in a lower priority compared to others. The action that we are going to take regarding this is to do a planning, which is explained further below.

Action

The action that we are going to take regarding this risk is the plan of action. The developer team must ensure that all project deliverables are achieved in each sprint. A contingency plan should be made available beforehand for each sprint and deliverables. If any certain factors make it impossible to accomplish a certain product deliverable, the developer must let the project manager know and run the backup plan accordingly.

System Design

This section describes the design choices that we have chosen for the visual representation of the application in the form of the user interface. Furthermore, the system's functionalities are also elaborated and explained in detail.

System description

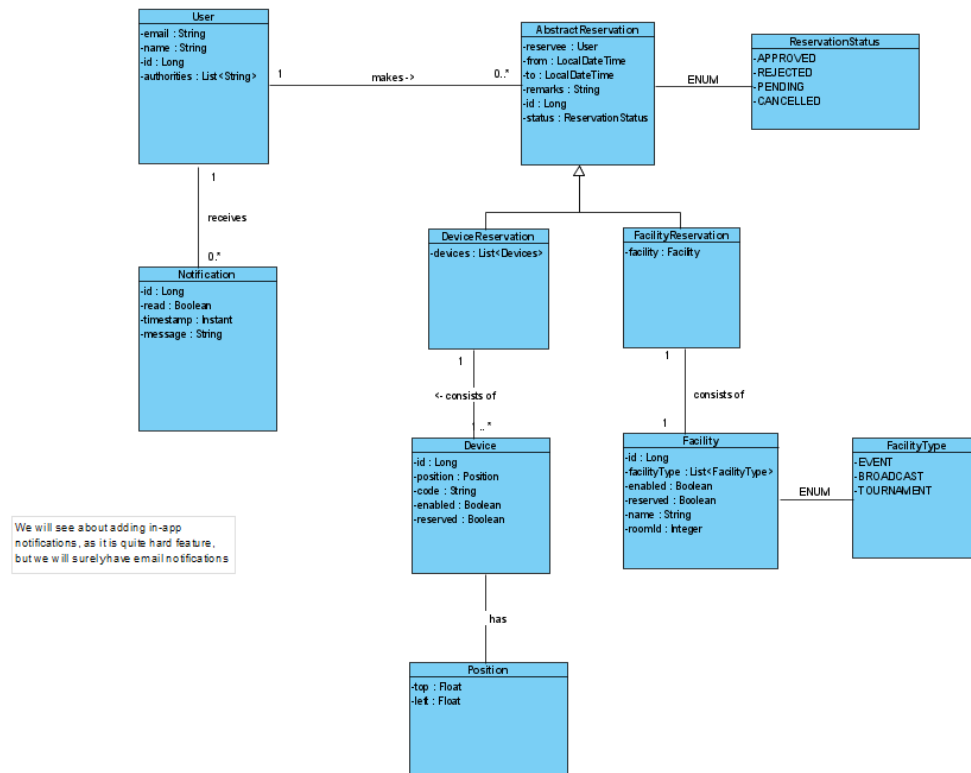


Figure 10 : The class diagram of the system showing the application structure.

In this figure there are nine classes. They represent the core of the system and the objects that are going to be created and used in order for the system to function. Class "User" is the class that represents as the name suggests the user of the system. A user has an email, id and authorities. The last one is used to differentiate between the common users and administrators. Unique id is used in order to connect the reservations to the user. Email is needed in order to send confirmations after a reservation has been made. All of this information is being sent by the SSO login point from University of Twente. User class makes reservations and receives notifications. The notification class has message, id, timestamp and read boolean value. Id represents the notification and differentiates it from other such notifications. Read is used to check if a notification has been read by the user. Timestamp represents the date and time at

which the notification is created. AbstractReservation is the parent class of Device and Facility reservation classes. It makes use of the enum class ReservationStatus. This class contains the shared information between its child classes. It is characterised by reservee, from, to, remarks, id and status. “Reservee” is a User class object. “From” is the date and time from which the reservation starts and “To” the date and time at which ends. Remarks are for in case the reserve has any comments about his/her reservation. Status indicates if the reservation is accepted, rejected, pending or cancelled. DeviceReservation and FacilityReservation inherit all the elements in AbstractReservation as child classes but they are also characterised by lists of devices and a facility respectively. And those are objects from the classes Device and Facility. Device is represented by position, code, enabled, the boolean reserved and the device type, which could be console, laptop or desktop. Position shows where in the lounge the device is located and code the number given by the lounge staff to recognize the device. Facility class contains id, facility type, enabled, reserved and name. The facility type takes the values event, broadcast and tournament from the enum class FacilityType.

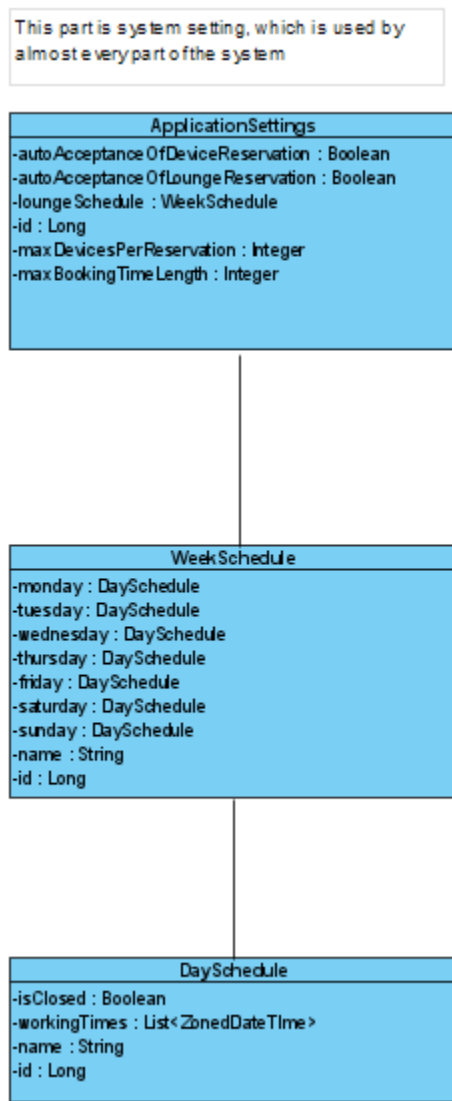


Figure 11: The application structure of the configuration part.

A second part of the system is represented on this class diagram. It concerns the setting of the application. `ApplicationSettings` is the main class here characterised by auto acceptance of lounge and device reservation, schedule of the lounge and id. Auto acceptance gives the possibility to control if a reservation is going to be accepted by the application or it needs an administrator to do so. Lounge schedule contains an object of the class `week schedule`. We can have multiple schedules created and choose one of them for the time being and change it to another existing one if needed. The `WeeklySchedule` contains seven objects of the class `DaySchedule` and a name of the weekly schedule (e.g. “Easter week”, “King’s day week”). The `DaySchedule` class has “isClosed”, working times and name. “isClosed” indicates if the lounge is closed during that day. Working times show at which times during that day the lounge is open. Name indicates the day of the week (e.g. “Monday”, “Tuesday”, etc.).

Design choices

Some of the more interesting design choices that we made are:

- Alerts
We decided to add toasts to most of the actions that are done in the application, because we wanted to have a signal that would tell our users that the actions that they perform have either been successful or not. In almost all cases, we use toast in the corner of the window and the reason for this is because we did not want to include additional clicks to close a notification/alert for proceeding with your work in the application.
- Mailing and confirmation
Mailing and confirmation was added as we wanted our users to know that their bookings have been accepted/declined/cancelled. It was a wise design choice to do the mailing in an asynchronous way, since it is a really expensive process, which can hurt the performance. This is why we made use of the Spring Events, which can execute jobs in an async mode.
- Navigation bar
The navigation bar in our application is on the left. There was a discussion about where to put the navigation bar. It was decided that it will be on the left because of another booking system that the University of Twente provides for project room allocations. In order to keep both systems in consistency with each other, we made this choice.
- Clickable units
Adding hover effects to various buttons was a decision made to make the user experience better. We decided to add the colour change and tooltip displaying the task that button performs when hovering on it as we think that it is more intuitive for the user to identify which parts of the application they can interact with and what they do. We have also added hover effects that display the name and type of the device as well as

from which times that device or facility is reserved so that the users can have additional information on the device.

- Roles

One of the most significant design choices that we made in regards to security was having three types of users - Client, Admin, Superior Admin. The superior admin has full control over the system - adding and deleting admins while the regular admin can only perform part of the actions. Having a hierarchy in the admin roles removes the possibility of administrators to abuse the system.

- Filters, sorting and pagination

We have added a highly configurable option for displaying reservations, as with growth of the dataset, it will be harder to navigate through the bookings. This is why we added pagination for improving performance by fetching a limited number of entries, sorting for having the reservations in certain order and filtering for finding the reservations you are interested in. We have also added a search bar from which the admins can search for various types of reservations: on name, device, remarks.

- General settings

Aim of every system should be to make everything highly configurable and easy to operate with. This is why we discussed with our client and added a setting for almost every part of the application, which can be controlled by such parameters. This will make the services of the system easily changeable in the future providing better alternatives for the administrators.

- Device types

We made it possible to make use of all the existing bootstrap icons, when creating a new device type. This way it is possible to make the representation of every type look different for the users. This was done, because in the future new devices will be bought for the lounge and by creating this abstraction the administrators can easily create new types without having to release a new version of the system.

- Scheduling

Our client demanded that we make a flexible working schedule for the lounge and this is why we made every part of a calendar to be creatable, editable and deletable. Firstly there are day schedules, which represent a working time representation of a single day. By using the day schedules, you can create week schedules, when you pick a schedule for every day of the week and in the end you can put an active week schedule for the lounge. This way you can create as many days and weeks as you want and easily interchange the working time of the lounge. Since it could happen that you want to disable a specific date, such as when there is a tournament, we also made the day schedule exceptions. This is a day schedule, which will overwrite the working time for a specific date. This was all talked through with our client and will satisfy all their needs and edge cases that may occur.

Functionalities

In this section we are going to explain the core functionalities of the system with explanation on how to use them.

- Login with University of Twente credentials
In order to enter the system, the user must be registered in the University of Twente database, because the reservation system is integrated with their SSO login system.
- Create devices and pick their position in the lounge
This is one of the most crucial features in order for the reservation system to function well. In order to make a reservation, there should be an available device, which you can book. For this to be possible, we made in the administrator panel an option in the settings in which you can create a device. The device can have a unique identifier in the lounge, a device type and if we want to disable it in case of technical problems with that device. Beforehand, you must have created a device type, as we know how fast technology changes, we decided to make that flexible. The device type contains a name and an icon, which will represent it in the map. After the creation, you will be able to see all the devices of the lounge in a table overview. The next step is to go to the interactive map tab and position them according to their physical location in the lounge. This will be the map shown, when making a booking.
- Create facilities and pick their position in the lounge
The second booking type we support is facility. The scenario is the same as in the devices, with the difference that the facility types are broadcast, competition or training and you pick a whole room, which is suitable for this type in the lounge. Our interactive map will be displayed when creating the facility, allowing the admin to pick an available room to serve as the specified type. This will give the possibility for users to book a facility in the booking screen.
- Create flexible working time schedules
There is a functionality of making working time schedules or more specifically from what until what time reservations are possible. For this to work, the first step is to go to the day schedules tab in the settings and create one. It is possible to pick from one to several ranges of time throughout the day or mark the day as day off. After creating a day schedule, the admin can go to the week schedules tab and structure one, using the previously created day schedules. For every day of the week, there could be different schedules. There is the freedom of creating as many day schedules and week schedules as the admin wants. The last step is to go to the general settings and set the current active week schedules, which the lounge will follow. This will restrict the booking time slots to the specified active schedule for the respective booking day. There is also a possibility to add a day schedule exception, which will override the day schedule for a certain date. This gives us the ability to create exceptions in the schedule, without interrupting the future weeks.

- **Add administrators**
The flexibility of the administrator's privileges should not be neglected and since the lounge is formed from student organisations we added the possibility to delete and add users, who will have admin roles. This could be easily done only as a global admin, which can be assigned by the developers or maintainers of the website. In order to give users such roles, you must go to the admins panel in the settings and either delete or add a new admin by specifying the name and email address.
- **General settings**
In the general settings tab, there are configurations for the admins to choose from. It is possible to give the system rights to automatically accept reservations, to restrict the maximum booking time length, the maximum number of devices and as mentioned before the active week schedule.
- **Create reservation with acceptance/rejection**
When the administrator part is all set up, all the users can now make bookings. To do so the user has to pick from the make a booking panel what type of reservation he wants to make. It could be a device or facility. Then the booking screen appears in which it must be given the date and from, to the time slot for which the reservation should be made. After this step is finished the interactive map will be populated with all the facilities/devices. If an entity is busy for the whole time slot it will be in red, if it is partly-busy (not occupying the whole time slot), it will be in yellow and if it is free it will be black. You can select an entity by clicking on it. The last box is for entering any comments and remarks. The last step is reviewing the reservation details, which will appear in a modal and you can either agree or cancel. If the auto acceptance is not enabled, the admin must accept or reject the bookings made. This happens in the pending bookings panel. A booking is valid only after it is put in an accepted state.
- **Check your bookings**
In the my bookings tab, you can get an table overview of all the bookings made by you. The table is filterable and sortable, so the user can set it up to his preferences. It displays all kinds of useful information such as time, status and location of the booking.
- **Check all bookings**
Every admin is able to see all the bookings made in a filterable and sortable overview table with all the information about them.
- **Receive email notifications upon cancellation/acceptance/rejection**
Upon changing the state of a booking, the reserver will receive email notification with the changes and details of his reservation.

System Development

This section describes the technical choices and characteristics of the system. The description of the system implementation is justified here along with the implemented design choices and functionalities.

Programming languages and frameworks

- **Database**
For database software we chose PostgreSQL, for the simple reason that all of the team members had experience with it. Other than that, pgSQL has an advantage over other choices as it supports transactions, which helps us to provide data atomicity, consistency, isolation and durability, all kinds of indices for different use cases, full-text-search and really useful types like 'jsonb' and its functions. All the features integrated in this DB software helps us to provide consistent solutions to a wide range of problems. For schema-management we use Flyway migrations. This is a tool that allows us to store our database creation statements in version control software as well as provide incremental builds, do checksum checks to prevent faulty changes and description of what each migration is for.
- **Back-end**
For backend language, we picked Java because of its ecosystem and enormously powerful library tools, enabling us to easily build business logic. For the framework we chose Spring Boot, since it provides a production-ready environment, which is highly configurable and gives a lot of flexibility for design choices. We stick to the layered architecture Controller-Service-Data, separating the concerns at every layer. The controller handles the HTTP requests passed from the dispatcher servlet, the service handles all the complex business logic and the data layer is responsible for querying and saving data to the database. Along with that we use Spring JPA and Hibernate to act as middleware between our application and database and it removes the weight of the shoulders of the developers to write manually SQL queries for every task. Of course, this comes with consequences of performance issues, but we strived to stick with all best practices and provide efficient solutions to all problems. For securing our application, we chose Spring Security because its integration with the rest of the Spring ecosystem is extremely easy and it fit perfectly with the OAuth2 integration we had to implement, leaving us to specify only a few configurations to make it fit our requirements.
- **Front-end**
For the user interface, we picked the simplest option and used plain HTML, CSS and JQuery, as they satisfy our needs and all the team members were experienced with them. For styling we used Bootstrap as it contains a useful grid system and wide range of different styled components ready for use.

System description

- Database

In the database, we use all the default settings, so we will explain a bit on what types and indices we used and why. For all the primary keys we use the 'bigint' type, because it provides much larger range than 'int' and as we could not predict the scale of which our application will grow, it was safer to use this type. For id generation, we've created a sequence to take care of that, because hibernate supports it as and it enables us to do batch updates and inserts. The fundamental difference is that this strategy does not require you to have the previously added entity's id, but it directly gives you a range of available integers, which the database can use. For all date related fields, we use 'timestamp' as this is the required type from Hibernate to correctly map between the Java types and pgSQL types. For all lists we store, we use 'jsonb', since pgSQL provides really fast indexes to search for elements through the jsonb column and we needed that to validate the authorities of the user sending a request.

- Back-end

As soon as a request hits one of the controllers, we are being passed a data transfer object, which is serialised from a JSON object. The reason we use DTOs is simple: we are using ORM framework, so if we use our entities to send and receive data, this will expose our internal structure of the database, which can have security consequences. Another reason is that DTOs provide us with the ability to restrict what the user can see in different situations, such as not having authority over some fields or just there are unnecessary fields, which would only increase the network latency. The next step is to either convert the DTO to an entity or directly pass the DTO to the service layer, which is responsible for business logic. This is where we start to use transactions, in order to prevent all types of non-ACID actions. The default level is used, which is read committed. This disables seeing changes, which are not yet committed by another transaction. This suffices our use case, as we never modify more than one entity at once. All of our business logic starts with validation of the passed object. We check if all the fields are correctly filled, if there are no overlaps for instance with another booking, etc. After all the validations are successfully passed, we modify or add the passed entity. In case there is an error with the validation, we throw an unchecked exception, for the transaction to roll-back. In many places our application will grow in size and this is why we inserted pagination for the table related pages. Luckily Spring JPA has a built-in Page class, which helps us easily to get a set of entities with an offset and page size and satisfy our needs. Working with a large dataset would also require adding nice sorting and filtering in order to easily browse through that set. For sorting, we again use the built-in Sort class, which modifies the query to give us the entities in the ordered list. For filtering, we made use of the so-called Criteria Builder. This is a Java class, which helps you to dynamically build a SQL query using Java. Using it we could easily add a check for a certain field and return the correct result with minimised effort. The last interesting part of the system is how we handle mail sending. Since that is a really expensive action, we couldn't just send it in our business logic, as this will slow down the responsiveness

of our application. This is why we made use of the Spring Event handling and Java Mail API, which gave us the ability to create an event class to handle the receiver information and content and event listener to send the email using Java Mail API, while all of this is handled in asynchronous mode. This means that this is executed by another thread, which is not associated with handling the request sent by the user. The rest of the system is following common Java conventions and practices.

- Front-end

In the front-end we strived to separate all the pages, each having a JavaScript, CSS and HTML file, which implies a lot of repetitive code, but that is one of the disadvantages of not using a modern framework and using reusable components. Still, the speed with which our pages render, fetch data and display it is maximised. Every page initially starts with rendering the navigation bar and sending all the necessary AJAX requests to the backend for the required data. If there are fields for inputting information, they are always validated before being sent, giving a simple error sign to signal the user that there is something wrong. All the errors in our application are being shown by a utility function, which can add a toast on the screen for a successful or failed task. For tasks, which are not supplied by html, such as date and time picker and more user friendly select fields we use libraries only from JQuery-UI, as they easily fit with the rest of the JQuery environment. The most intriguing part of the front-end is the interactive map. We drew it using the HTML canvas, which gives the possibility to do all the figures you can think of and Path2D to simplify the creation of the lines. We create it in an utility function, which helps us to render the map at different pages easily.

Login system

For integration of the Microsoft Single-Sign-on point, which the University of Twente uses, we needed a bit of information from them. In order to connect, your system must be registered in LISA's security registers, which generates unique client ID, secret and metadata. Upon entering the reservation system, Spring security will check if the requested endpoint requires authentication and if so it will search for a session token, in order to give the principal details to the thread which will serve the request. If such a session or token does not exist, it will forward you to the Microsoft login page. Most of the magic happens behind the scenes. The configuration needed from our side was to specify which endpoints require authentication and set login and logout handlers. The login handler is responsible for redirecting the user to the correct page if he is admin or not and the logout handler is responsible for clearing the session associated with that user and its session-token cookie. Everything else is being handled by the default OAuth2 configuration of Spring security.

Testing

This section describes the test plan and the test results that have been defined and executed. The test plan defines and explains the approach in testing different functionalities and features along with the testing pass criteria. Moreover, the risks and contingencies and the schedule in which the testing is executed is also explained.

After executing the test plan, the test results of the unit tests, integration tests, results of the different approach in testing different functionalities and features are also defined. Based on the test results, some of the system designs have been adapted and bugs were identified and able to be solved.

Test Plan

Approach

We will approach testing from three different perspectives - unit testing, integration testing and usability testing. We will strive for maximum possible coverage of the code. Unit and integration testing will cover mainly the backend of the system and usability testing will focus on the front-end of that same system. We cannot test external features such as login which is dependent on how well the university API is working but we are under the assumption that it will work robustly.

Unit testing

Jupiter Junit tests in conjunction with Spring framework will be used. They will cover the separate functionalities and their edge cases in the backend. We will strive for maximum coverage of all lines of code of the critical features.

Integration testing

Integration testing will be used to cover the gaps of the unit tests in the backend. The reason is that even though unit tests might show promising results on how features work separately, using them in conjunction with one another might still show some unexpected behaviour. Our approach will be a top-down approach using the system as a black box to which we give input and check for the expected result. Different test cases with steps will be created to verify that all the critical business processes are covered. We will be using Postman, as we have a small amount of endpoints.

Usability testing

This will be used to cover the frontend of the system and the intuitiveness of the design. As developers we cannot do that because we created the system and as such we will have biased opinions about the design. Because of the nature of the project (reservation system) we will try to draft people from different time zones to test the robustness of the system in such cases. We

will have a mixed approach using remote usability testing and the expert review heuristics of Nielsen (Nielsen & Molich, *Heuristic evaluation of User Interfaces* 1990).

Visibility of system status - The system should always keep users informed about current state and actions through appropriate visual cues and feedback within reasonable time.

Error prevention - Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

Aesthetic and minimalist design - Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.

Consistency and standards - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

We are going to draft potential users of the system (University of Twente students) to test the student perspective of the system. From the administrator perspective, we are going to draft potential employees that are going to operate the system in the future and we are going to do that in conjunction with our client. The test will consist of a series of tasks defined by use scenarios that needs to be completed by the users and administrators. They are then also going to be asked to speak what comes into their mind such that more feedback could be received (Sharp et al., *Interaction design: Beyond human-computer interaction* 2009).

A pass for these tests is considered the user satisfaction with the design of the system according to the heuristics above. This means that the test passes when the users can complete these tests without any problem.

Features to be tested

In this section we are going to list the features that need to be tested. The features are the most important ones because without them the system will not be what the client and the users would expect. Their level of risk is marked with 1-5 scores (1-lowest risk, 5-highest risk).

Features	Level of risk
Reserve a device/facility	5
Cancel reservation as a student	5
Deny device/facility reservation requests	5
Add devices and/or facilities	4
Remove devices and/or facilities	4

View all the reservation	3
Set times for reservation (when the lounge is opened)	5
View all made reservations	3
Inform the students about a confirmation/change of their reservation via email	4
Log into the system	5
Log out of the system	5
Change the style of the application layout	2

Item pass/fail criteria

Items with a 4-5 score must be completed, and their tests must pass without error. While items with scores 1-3 are allowed to have minor issues as long as they do not obstruct critical business processes and functionalities required for the system to be considered "complete."

Risks and contingencies

We have two main risks in this project. One is due to the nature of the methodology of development using Agile (Abrahamsson, *Agile Software Development Methods: Review and Analysis* 2002) and another due to external factors such as logging in using University of Twente credentials.

The former one creates risk whenever new requirements are introduced in the project which can endanger the scope and the time it would take for the project to be finished fully. If too many new requirements might overwhelm the developers team. This would automatically mean that testing will increase. This risk can be mitigated by good time management and with the immediate and consistent testing during the sprints.

The latter one is dependent on external factors such as how smooth the process of acquiring access to the University of Twente login APIs. In the worst case scenario the development team will not be able to acquire access and the system cannot be tested (integration and usability testing) with the login. To mitigate this we will work in conjunction with the client to acquire access to those APIs.

Schedule

Because we use the agile methodology with sprints in this project requirements might change during development. During each sprint a requirement can be altered, added or removed which means respectively to alter, add and remove tests. At the end of each sprint an integration test

will be conducted for the features that were implemented until then. The most important and full integration tests will be conducted in Sprint 4 of the development phase when the product is expected to be fully functional. Usability testing will be conducted during Sprint 5 which is the final sprint so that as developers we have time to polish the design flaws that are identified.

Approvals

Unit testing is a concern of the development team and because of that the approval of a test coverage of a feature will be given by the whole group. Integration tests will also be approved by the developers in conjunction with the client. When it comes to usability testing which is concerned with the end-users the approval will come from the end-users themselves (depending on how satisfied are they with their experience with the system)

Test results

Unit testing

The unit testing was limited only to services, which do not include more than one entity, which resulted in a feasible work, as the most complicated parts were an aggregate of services. All of the Create, Read, Update and Delete were tested, resulting in a working system. Since that was not a large amount of work, during the unit testing, the quality of queries were also inspected which was generated by Hibernate and managed to update several bad relationships which resulted in an increased performance of the system in the end.

Integration testing

During the integration testing, a bug was found. The bug is interfering with making a reservation when having a more complicated day schedule. The check of whether a booking's time slot fits into the schedule was producing wrong results, which thankfully was easy to fix due to the nature of the system architecture. The rest of the integration tests, such as positioning of reservable entities and integration of reservations with settings and schedules went as expected, without introducing any additional work.

Usability testing

The usability tests have provided a really useful feedback in which it helped to discover flaws, bugs, and system design changes from the system. The details on the methodologies that were used to conduct the tests and the features that need to be tested could be seen in the "Testing" section under the "Test plan" subsection. These tests are used to improve the Nielsen's Heuristics that were mentioned in the subsection. The participants were asked to do some number of tasks on the system in order to inspect the system when it is used realistically. The use cases are referred to after this sentence and the interaction scenario is available in the "Appendix" section under the "Interaction scenario" subsection.

Use cases user side

1. Log into the system with the student's credentials
2. See available devices/facilities
3. Reserve available devices/facilities
4. Cancel reservations
5. Get confirmation of the reservation
6. Log out of the system

Use cases administrator side

1. Log into the system with the student's credentials
2. Confirm or deny users' reservations
3. Cancel users' reservations
4. View all users' reservations
5. Add devices and/or facilities
6. Remove devices and/or facilities
7. Set the lounge opening times
8. Change the style of the application layout
9. Log out of the system

The usability testing provided a lot of useful feedback on the system usage. The usability issues that were identified with the tests are as follow:

Feedback user side

1. Tabs button colours is inconsistent when hovered over
Solution: *The tabs button colour is changed to be consistent with the other tabs when hovered over.*
2. Devices needs to be labelled
Solution: *A tooltip with device information is added, when you hover over a device.*
3. Remove rooms from the map, which will not be used.
Solution: *Hide rooms from the map, while preserving the ability to return them back.*

Feedback administrator side

1. The button to make new devices/facilities/schedule is not easily recognized and found.
Solution: *The button is made to be bigger and have more description to be easily recognized and found.*
2. Scheduling the opening time of the lounge is easy to do but needs to be explained.
Solution: *A manual is added in the administrator setting page on how to schedule the lounge opening time. Since there may exist confusion in other modules as of the application, a detailed explanation was added for most of the crucial features.*

3. Undesired changes in all future weeks, if there is an event happening and we change the schedule.

Solution: *The day schedule exception was introduced. It allows you to specify a schedule only for a specific day, without interrupting the normal working hours of the lounge for future weeks.*

4. Devices needs to have flexible types for future changes (PC/Laptop/Consoles)

Solution: *The device types module was created. It allows flexibility on creation of different types and their respective icons. Examples are PC, Laptop, VR set, Playstation, Xbox, etc.*

Future Work

This section explains the prospective development for the current system after the project is done since it is limited to the defined and explained requirements and scope in the previous section. The system will already start to be used, but further features of the system could be added after this project. Examples of the aforementioned features are going to be discussed more here.

Support of the system

The system would be instantly employed by the client, Esports Team Twente, after the development phase was completed. As a result, it is necessary to provide the client with the assistance they require in order for the system to function properly. Due to the first time deployment of the system into production phase, which would be on a server maintained by the client, Esports Team Twente, the system would be extremely unreliable. Because the project's system is a new tool, it is safe to anticipate that Esports Team Twente staff will have a hard time in its maintenance. As a result, it was decided to support the program with members of the development team, who would have sufficient knowledge to fix any issues that may arise during deployment.

Integration with DMS system

Once the system has been deployed in the production environment, which would be on a server maintained by the client, Esports Team Twente. The system's behaviour could be inspected more such that it could be pushed into a stable release environment. When the system has been working stable in the server, then the plan is to integrate the system with the University of Twente DMS system. This integration will enable the system to be used by the Unioncard holders. Thus, the Unioncard holders will be able to use the system by logging into the system via their DMS account login.

Evaluation

This section explains the general evaluation of the project, which includes, the initial planning, the work responsibilities, team evaluation, initial planned deliverables, and the overall conclusion of the project's progression from the start until the finish.

Planning

The planning of the project was delivered in time as the initial planning that we had defined from the beginning of the project. As defined in the beginning of the project in the Project Proposal document, the project adheres to the Agile methodology in the form of Scrum. The project methodology ensures that we have an iteration every two weeks of the project in which we have to finish the tasks that we set up. Furthermore, the progress of the project and the tasks are defined in our project board, which is in the same environment as our code repository, GitLab. The planning is respected during the project which ensured an equally distributed workload between each iteration, while also ensuring high quality of the deliverables.

The Scrum framework is used to plan, design, build and test the application. Our project plan is divided into five sprints. Each sprint would last for two weeks, except for the first sprint until the fifth sprint which would only last for a week and a half due to the kick-in and the easter holiday in University scheduling respectively. Moreover, there is a one week gap from the first and the second sprint as this is due to the spring break holiday in the University scheduling. The details of the plan are as follows.

Sprint 1 - Requirement analysis / Design (9th - 18th February 2022)

- Setup contact with the Product owner.
- Setup contact with the supervisor.
- Finish the project proposal.
- Start with the Requirement Analysis and Specification.
- Contact the University of Twente system manager.
- Set up a GitLab repository.
- Start design of UML diagrams.
- Finish mockup.
- Prepare for the peer review meeting, talking about the project's proposal and planning.

Sprint 2 - Design / development (28th February - 11th March 2022)

- Implement the first increment of the system, where the system must have a ready and functional layout of the user interface, connected with a REST API server, as well as configurable device and facility pages in which an admin can create, edit and delete entities.
- Extend UML diagrams with the idea of the system final version.
- Start the first version of the design report.
- Start and complete the test plan for the system.

- Complete the Requirements Analysis and Specification.
- Prepare for the peer review meeting, talking about the project's requirements specification and test plan.

Sprint 3 - Development (14th - 25th March 2022)

- Implement the second increment of the system, where the system must have an interactive map of the architectural plan of the lounge, in which the admin can manually set up devices and facilities and have all the settings needed for making a reservation to be functional and properly working.
- Complete first version of the design report.
- Continue with the development and focusing on adding new features.
- Continuously testing the system and eliminating all the system errors.
- Prepare for the peer review meeting, talking about the project's first version of the design report and the system.

Sprint 4 - Development (28th March - 8th April 2022)

- Implement the third increment of the system, where the system must have a reservation making tab, in which a user can make a reservation for a specific seat and/or facility in the ETT lounge. In this system increment, we must have a login integration with canvas and make it possible for University of Twente students to use their credentials.
- Start with the poster.
- Start with the presentation slides.
- Continuously testing the system and eliminating all the system errors.
- Continuously working for the design report.

Sprint 5 - Closure (11th - 20th April 2022)

- Final product demonstration, presentation, and poster
- Finalise the testing for the system, eliminating all errors and bugs.
- Finalise the Design report.
- Finalise the design document, providing the system manual and documentation.
- Complete presentation slides.
- Complete the poster.

Responsibilities

The project was well-divided into various tasks because the team was made up of people with diverse interests and skills. The task assignments were primarily assigned to team members based on their interests since we believe that if we work on what we like, we will have really good results. Furthermore, because of the wide range of interests and experience, we delegate different responsibilities as follow:

- Boris Belchev
Front-end developer, Requirement specification, Poster design and presentation slides.

- Irvine Verio
Project manager, Communication manager with the client, Requirement analysis, Front-end developer, Poster design and presentation slides.
- Ivan Trendafilov
Database designer, Back-end developer, System development, Head of testing.
- Pavel Hristov
Database designer, Head of back-end developer, System development, System testing.
- Viktor Tonchev
Head of system development, System testing, Back-end developer, User's manual and documentation editor.

It's worth noting that the project deliverables were written as a collaborative effort including all members of the team. The group has also been in charge of personal communication with the client, Esport Team Twente, and the University of Twente supervisor (Dr. Vadim Zaytsev).

Team evaluation

The team has been working fully online since the beginning of the project. The only time that we met physically was at the start and the end of the project. However, this does not affect the working chemistry of the team itself. We work mostly individually or in a group of two if they have similar tasks since this makes it easier to schedule meetings and to work together. Besides, we always meet every week in order to discuss the progress and plan on the work for the next weeks. The responsibilities that were mentioned previously were identified naturally. Moreover, all of the team members have done a fair share of the whole project work. All in all, the team working chemistry was really good and there were not really any problems or arguments in working together as a team.

Deliverables

The project deliverables that were set in the beginning of the project are also reflected here whether the deliverable is satisfied or not. Following are the system requirements along with the progress of the deliverable if it was achieved successfully or not.

Functional requirements

1. ✓ The reservation system must be able to let students log in and out with their UT account or DMS-account (Unioncard).
2. ✓ The reservation system must be able to reserve devices and/or facilities.
3. ✓ The reservation system must be able to let students cancel their reservation.
4. ✓ The reservation system must be able to inform students about confirmation messages or email.
5. ✓ The reservation system must be able to let the Esports Lounge staff a way to log in and out as admin.

6. ✓ The reservation system must be able to give the Esports Lounge staff the option to cancel a reservation.
7. ✓ The reservation system must be able to let the Esports Lounge staff easily view all made reservations in an easily readable overview/timetable.
8. ✓ The reservation system must be able to let the Esports Lounge staff confirm devices and/or facilities requests.
9. ✓ The reservation system must be able to let the Esports Lounge staff deny devices and/or facilities requests.
10. ✓ The reservation system must be able to let the Esports Lounge staff the possibility to set the times that the lounge is open and reservations should be possible.
11. ✓ The reservation system must be able to add devices and/or facilities for students to reserve.
12. ✓ The reservation system must be able to remove devices and/or facilities for students to reserve.
13. X The reservation system should be able register full facilities reservations within the DMS system of the University through an API connection.
14. ✓ The reservation system should have an interactive map with all the devices and facilities on it representing the real lounge.
15. X The reservation system could change the style of the application layout.

Non-functional requirements

16. ✓ The system should handle the users' and/or administrators' login within 30 seconds
17. ✓ The system should use the Esport Lounge Twente style guide for the frontend.
18. ✓ The system should send confirmation email/message for a reservation within 30 seconds.
19. ✓ The system should be available 24/7 except in case of maintenance.
20. ✓ The system should be able to handle at least 5000 requests concurrently, either by users and/or administrators, at the same time without degradation of performance
21. ✓ In case of a breakdown, the system should roll back all of the partially made reservations.

Some of the deliverables that were not able to be delivered, such as deliverables number 13 and 15, were caused by several reasons. We were not able to deliver number 13 because our client had envisioned a different way for the deliverables. There were many changes that happened during the development. In the beginning, he wanted this deliverable to be realised, but later on the client wanted an entity from the University of Twente that is different from the DMS system, in which would integrate the reservation system in it. In the end, the client told us that the entity did not want to invest money in the integration of a new project. Thus, the client mentioned that there are different plans for the future on how they would integrate the reservation system to the DMS system in which it is out of scope of the project.

Meanwhile, deliverable 15 was not implemented mainly because of time bound issues and its importance. From the beginning, it was clearly stated that this deliverable would be implemented only if there is enough time in the project and that if it was important in the future.

However, there was a different solution to this deliverable. It was agreed upon with the developer team and the client, that the team would provide documentation in the code from where the ETT staff could change the application layout.

In addition, one of the final product figures is also showcased in the figure below such that it could be seen what the product looks like. The other figures could be seen in the “Appendix” section under the “Final Products” subsection.

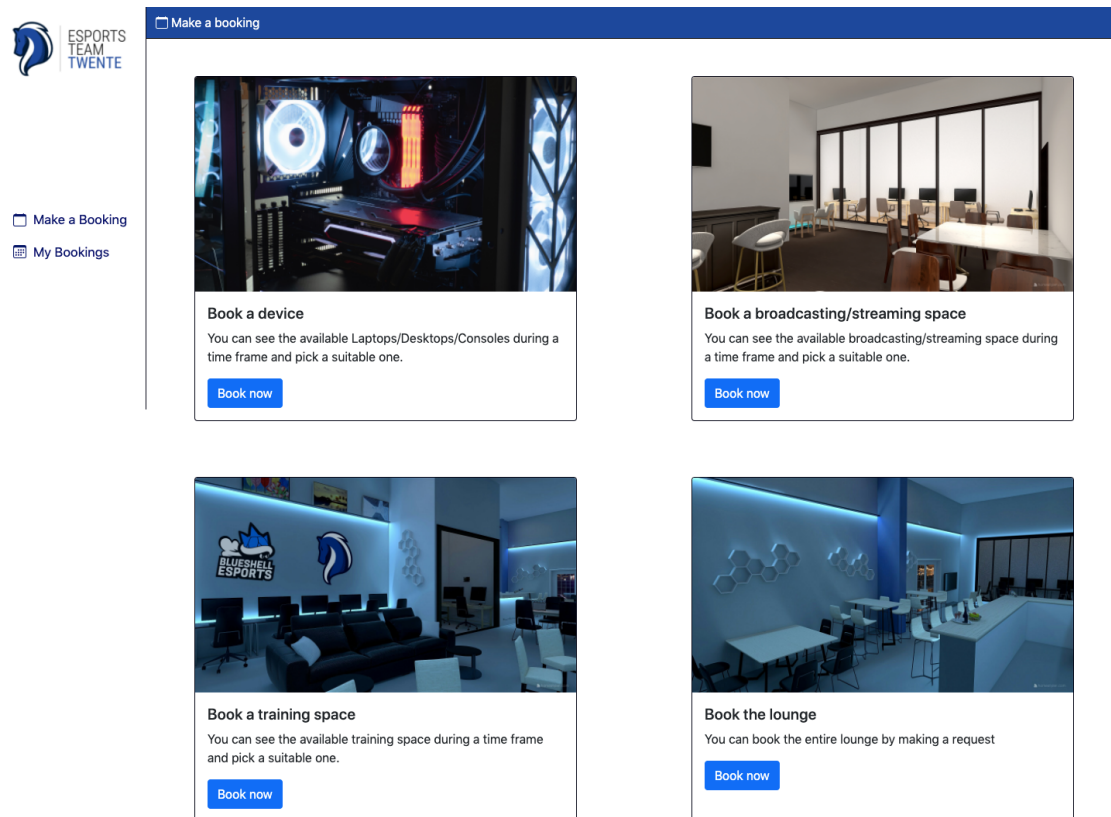


Figure 12: User interface home page to make bookings for devices and/or facilities.

Conclusion

Due to the project methodology, this ensured that the deliverables were completed and delivered on time. The team has been working fully online since the beginning of the project. However, this does not affect the working chemistry of the team itself as we always meet every week in order to discuss the progress and plan on the work for the next weeks.

The aim of the project was to deliver a system that prioritises on its usability, scalability, and the ease of adding new entities to the system with regard to the newly built Esports Team Twente Lounge. Moreover, the aim of the project was also to deliver a completely working and error-free system which can be hosted after the project has ended since the future plans is to deploy the system on servers hosted at the Esports Team Twente servers and might even be integrated into DMS system (as mentioned in the previous section “Future Work”).

As stated previously, the project resulted in a working system that meets the requirements and expectations of the client. After the system has been delivered, it will be taken into operation by the organisation, Esports Team Twente, fully into deployment.

To summarise, there were two general goals of this project. The first was to provide a working system that met the requirements and expectations of our client and the second goal was to obtain a better understanding of the development process and all of its iterations. The first goal was achieved without any problems and the final result of the product can be seen in the “Appendix” section under the subsection of “Final Product”. Because of the duration of the development, this project also provided insight into the challenges that may arise in terms of project work and also teamwork. In addition to that, the developer team has gained a lot of new knowledge in communication with clients and conducting interviews for feedback and improvement of the final product. Lastly, we have gained a lot of new knowledge in software architecture and development, measuring time frames for implementation, and finishing the project with a satisfied client. Therefore, the second goal has also been achieved. This means that we achieved the goals of this project in the end.

References

- Abrahamsson, P. (2002). *Agile Software Development Methods: Review and Analysis*. Technical Research Centre of Finland.
- Adi, P. (2015). Scrum method implementation in a software development project management. *International Journal of Advanced Computer Science and Applications*, 6(9). <https://doi.org/10.14569/ijacsa.2015.060927>
- Alexander, Ian. (2004). *A Better Fit - Characterising the Stakeholders*. 215-223.
- Baker, C. (2018, June 25). *Meet Dennis 'thresh' fong, the original pro gamer*. Rolling Stone. Retrieved March 22, 2022, from <https://www.rollingstone.com/culture/culture-news/meet-dennis-thresh-fong-the-original-pro-gamer-103208/>
- Cohen, D. S. (2019, March 14). *Cathode-Ray Tube Amusement Device: The World's first video game?* Lifewire. Retrieved March 22, 2022, from <https://www.lifewire.com/cathode-ray-tube-amusement-device-729579>
- Craddock, A. (2014). *The Dsdm Agile Project Framework*. DSDM Consortium.
- Dingsøyr, T., Nerur, S., Balijepally, V. G., & Moe, N. B. (2012). A decade of agile methodologies: Towards Explaining Agile Software Development. *Journal of Systems and Software*, 85(6), 1213–1221. <https://doi.org/10.1016/j.jss.2012.02.033>
- Esports Team Twente. (2021). *Year One in Review*. Esports Team Twente. Retrieved April 19, 2022, from <https://esportsteamtwente.nl/year-one-in-review/>
- Estublier, J. (2000). Software configuration management. *Proceedings of the Conference on The Future of Software Engineering - ICSE '00*, 279–289. <https://doi.org/10.1145/336512.336576>
- Flowers, J. G. (2008). Improving the capstone project experience. *Proceedings of the 46th Annual Southeast Regional Conference on XX - ACM-SE 46*, 237–242. <https://doi.org/10.1145/1593105.1593167>
- Good, O. (2013, June 19). *Today is the 40th anniversary of the world's first known video gaming tournament*. Kotaku. Retrieved March 22, 2022, from <https://kotaku.com/today-is-the-40th-anniversary-of-the-worlds-first-known-5953371>
- "IEEE Standard for Software Test Documentation," in IEEE Std 829-1998 , vol., no., pp.1-64, 16 Dec. 1998, doi: 10.1109/IEEESTD.1998.88820.
- Mannion, M., & Keepence, B. (1995). Smart requirements. *ACM SIGSOFT Software Engineering Notes*, 20(2), 42–47. <https://doi.org/10.1145/224155.224157>

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of User Interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Empowering People - CHI '90*, 249–256. <https://doi.org/10.1145/97243.97281>

Ohst, D., Welle, M., & Kelter, U. (2003). Differences between versions of UML diagrams. *Proceedings of the 9th European Software Engineering Conference Held Jointly with 10th ACM SIGSOFT International Symposium on Foundations of Software Engineering - ESEC/FSE '03*, 227–236. <https://doi.org/10.1145/940071.940102>

Prieto-Díaz, R. (1990). Domain analysis. *ACM SIGSOFT Software Engineering Notes*, 15(2), 47–54. <https://doi.org/10.1145/382296.382703>

Sharp, H., Rogers, Y., & Preece, J. (2009). *Interaction design: Beyond human-computer interaction*. John Wiley.

Appendix

Mockups

This section presents the system mockups, specifically the user interface of the system from the user's and administrator's perspectives, in which we presented to the client. These mockups define visually the user interface's characteristics and behaviour and the mockups description define verbally the user interface's description. The user interface mockup and description provide a visual context for understanding the system's other requirements (Flowers, *Improving the capstone project experience* 2008).

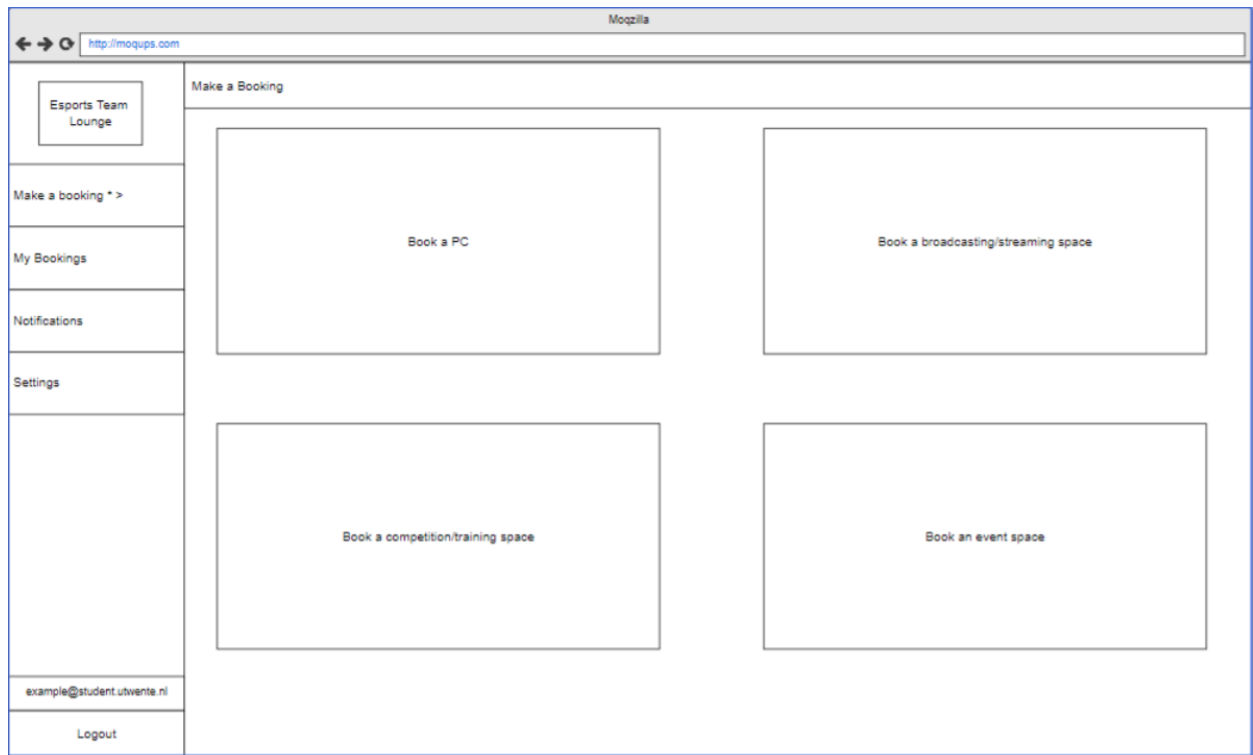


Figure 13: Mockup of the main page in the client view.

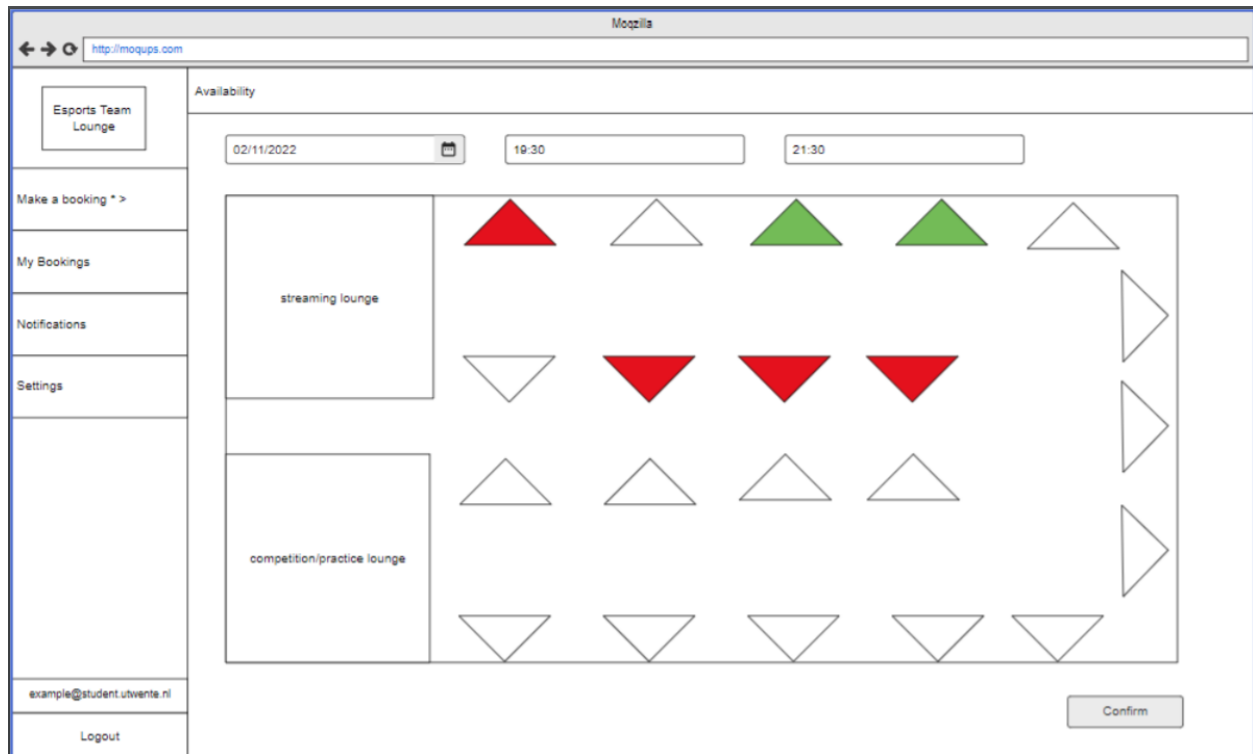


Figure 14: Mockup of the interactive map of the lounge, when making a reservation.

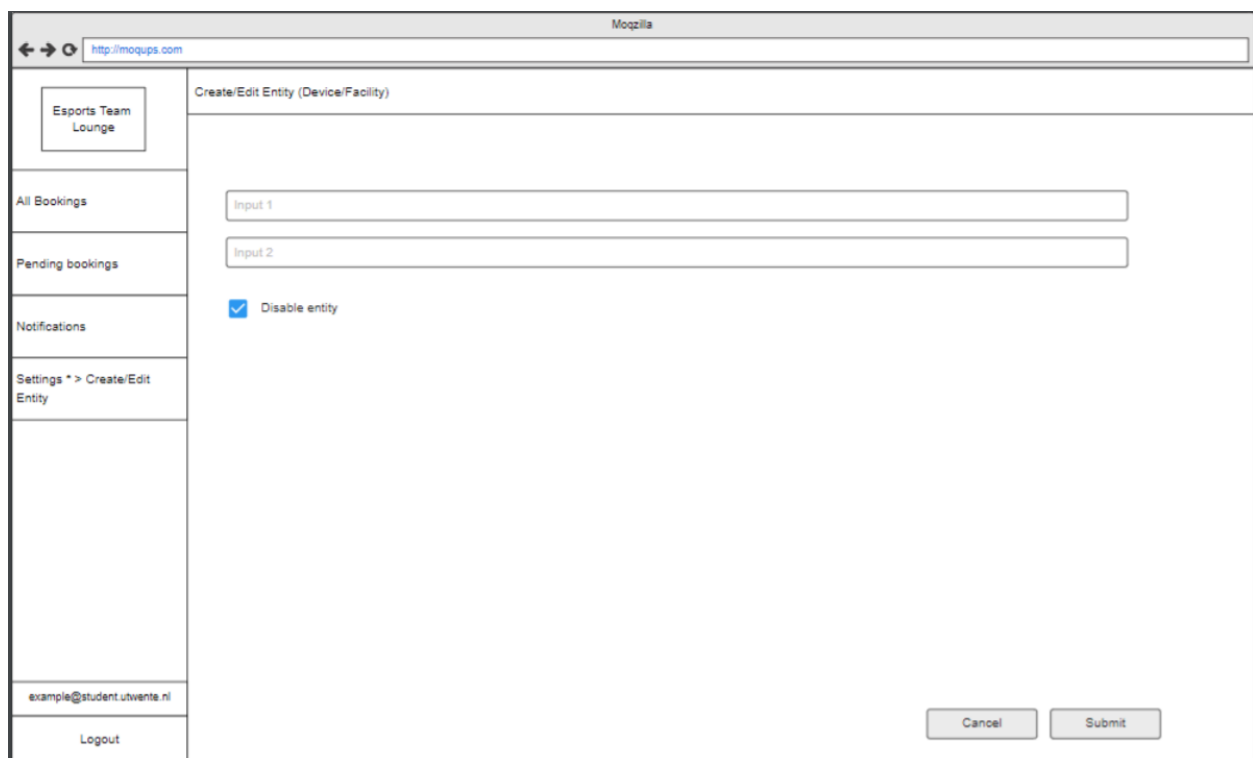


Figure 15: Mockup of device/facility creation in the admin view.

Esports Team Lounge

All Bookings

Pending bookings

Notifications

Settings * > Create/Edit Schedule

example@student.utwente.nl

Logout

Create/Edit Schedule

Name

02/13/2022

02/13/2022

Pick a weekschedule

Cancel

Submit

Figure 16: Mockup of the schedule creation in the admin view.

Esports Team Lounge

All Bookings

Pending bookings

Notifications

Settings * > Create/Edit Week Schedule

example@student.utwente.nl

Logout

Create/Edit Week schedule

Week schedule name

Monday: Select day schedule

Tuesday: Select day schedule

Wednesday: Select day schedule

Thursday: Select day schedule

Friday: Select day schedule

Saturday: Select day schedule

Sunday: Select day schedule

Cancel

Submit

Figure 17: Mockup of week schedule creation in the admin view.

The mockup shows a web browser window with the URL `http://moqups.com`. The sidebar on the left contains the following links: "Esports Team Lounge", "All Bookings", "Pending bookings", "Notifications", "Settings * > Create/Edit Day Schedule", "example@student.utwente.nl", and "Logout". The main content area is titled "Create/Edit Day Schedule" and contains the following form elements:

- A text input field for "Name".
- A checked checkbox labeled "Day off".
- A "From" time dropdown menu.
- A downward arrow labeled "Opened" indicating a transition.
- A "To" time dropdown menu.
- A downward arrow labeled "Closed" indicating a transition.
- A second "From" time dropdown menu.
- A downward arrow labeled "Opened" indicating a transition.
- A second "To" time dropdown menu.
- A "+" button to add more slots.
- "Cancel" and "Submit" buttons at the bottom right.

Figure 18: Mockup of day schedule creation in the admin view.

The mockup shows a web browser window with the URL `http://moqups.com`. The sidebar on the left contains the following links: "Esports Team Lounge", "All Bookings * >", "Pending bookings", "Notifications", "Settings", "example@student.utwente.nl", and "Logout". The main content area is titled "Device Bookings" and contains a table with the following data:

Reservoir	Facility	Date	From	To	Status
Pavel	Broadcast area	December 10, 1815	18:30	19:30	Pending
George	Event area	December 9, 1906	18:30	19:30	APPROVED
John	Competition area	August 17, 1936	18:30	19:30	REJECTED
Yordan	Competition area	June 24, 1917	18:30	19:30	REJECTED

Figure 19: Mockup of the table, displaying all bookings in the lounge.

Usability testing records

Interaction scenario

User side

beep A message has been received from your friend that he wants to hangout with you to go to the Esports Team Twente Lounge in Bastille. He asked if you could reserve the device first before he did so as he is away from his computer. To do this, you must first log in to the system with your account [Log in to the system as a user]. The reservation for the device is to take place on April 4 of this year, it starts at 11:00 and will end at 12:00 [See available devices] [Reserve available devices] [Get confirmation of the reservation].

After getting the confirmation of your reservation, you check your calendar and realise that you have a meeting at the specified time and would like to cancel the reservation. Then, you go to the reservation system and cancel the reservation. [Cancel reservations]

After discussing with your friend, you have decided to reserve a facility from 12:00 until 13:00 [See available facilities] [Reserve available facilities] [Get confirmation of the reservation].

The device is reserved and you can now log out of the system [Log out of the system as a user].

Admin side

The day has just started and you went to your shift at the Esports Team Twente lounge. You log in to the system as an administrator [Log in to the system as an administrator] to check for reservations for the lounge [View all users' reservations]. You noticed that a user has reserved an available device. Then, you want to confirm the reservation [Confirm users' reservation].

Then, another user wants to reserve a device that is currently not available. You realise this and would like to deny the reservation [Deny users' reservation].

After that, there are no more reservation requests. You then go to look around the lounge and see that there is a computer, device number 1, that cannot be used anymore. So you want to replace this with another computer, device number 10, in the same spot. You then do the replacement and update it in the system [Remove device] [Add device].

Continuing to walk around the lounge, you notice that there is a room, facility number 3, that has a water leakage which would not be able to be used. You want to replace the room with another room that could be used by the users, facility number 8. Thus, you update this in the system [Remove facility] [Add facility].

You go back to your seat and you notice that another user is trying to reserve an available facility. Then, you want to confirm the reservation [Confirm users' reservation]. Then, the same user wants to reserve another facility that is currently not available. You realise this and would like to deny the reservation [Deny users' reservation].

You just got a message from a colleague that the whole lounge is going to be reserved for tomorrow. Therefore, you want to cancel the users' reservations because it is going to be used the whole day tomorrow [Cancel users' reservations]

Finally, when your shift for the day almost ends, you want to change the lounge opening for next week since there will be a Blushell tournament event next week. You want to set the lounge opening only from 16:00 - 22:00 [Set the lounge opening times].

Your tasks are done for the day and you can now log out of the system [Log out of the system as an administrator].

Setup

The participants were asked to join a Google Meet video chat in which they are asked to access the test website environment. The other members are also observing through Google Meet and writing down the answers.

Procedure

1. Users will be asked to join the meeting via the Google Meet video chat platform.
2. A short introduction on the purpose of the system given by the experiment leader.
3. Users will be asked to go to the website test environment that is given through the video chat message and follow the interaction scenario specified above.
4. After the user has done the series of tasks from the interaction scenario, a number of questions (could be seen below) will be asked to the user.
5. The answers are going to be noted down and the result is going to be present as well below.

Questions

1. What was your first impression of the system?
2. What do you think about the layout?
3. What do you think about the colour scheme?
4. Do you think the User Interface works intuitively?
5. Do you know the meaning of the colours on the interactive map?
6. Do you think it's better to directly submit a changed setting or have a confirm button to submit all the changed settings at once?
7. Any current features you would say are unnecessary?
8. Any features you would add?
9. Anything else you would change to the application?

10. Are there any tips for the future system?

Final Product

Client interface

ESPORTS TEAM TWENTE

My Bookings

Search

From To Status
Open this select menu

Device bookings **Facility bookings**

From	To	Devices	Remarks	Status	Actions
09:45 14/04/2022	10:45 14/04/2022	Vr-set1		REJECTED	
10:30 14/04/2022	13:00 14/04/2022	VR		REJECTED	
09:45 14/04/2022	12:45 14/04/2022	Switch	Cannot wait	CANCELLED	
17:30 14/04/2022	18:30 14/04/2022	PlayStation		APPROVED	
16:00 14/04/2022	18:30 14/04/2022	Desktop		APPROVED	
10:00 21/04/2022	11:00 21/04/2022	Xbox	test	CANCELLED	
09:45 22/04/2022	12:45 22/04/2022	Laptop1	test	CANCELLED	
13:30 21/04/2022	13:30 21/04/2022	Desktop		CANCELLED	
10:00 21/04/2022	11:00 21/04/2022	Laptop1		PENDING	
10:15 15/04/2022	13:00 15/04/2022	Laptop1		PENDING	

< 1 2 >

Figure 20: Overview of a user bookings page.

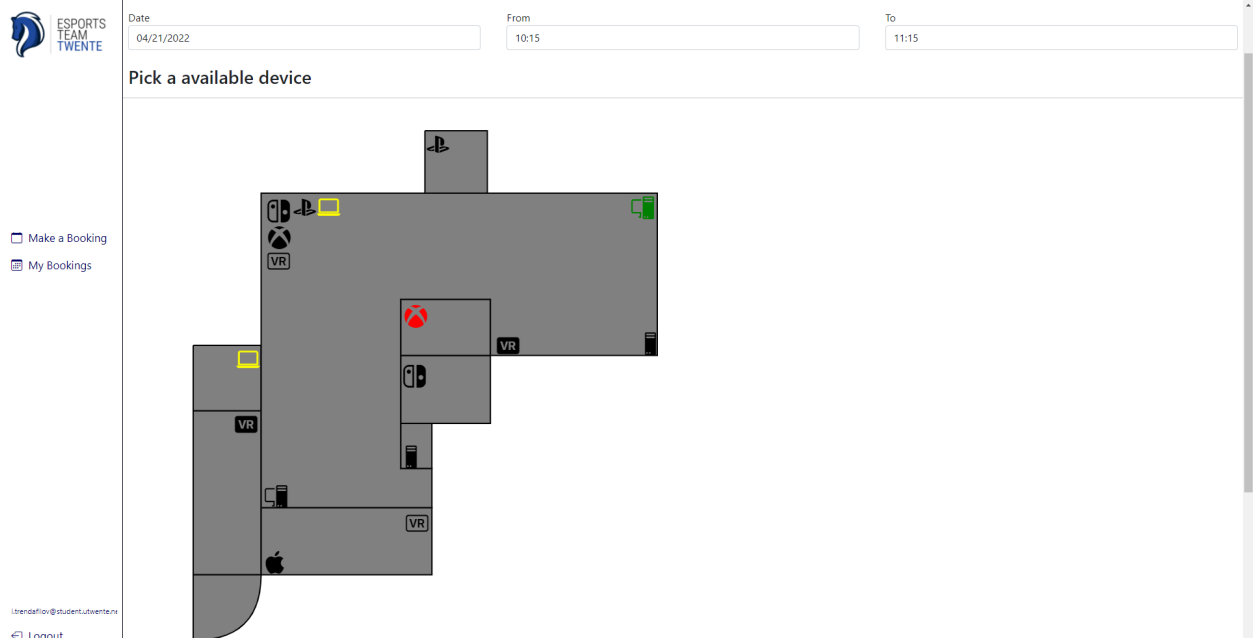


Figure 21: Device booking page for users to select what device they want to book for which date and for how long.

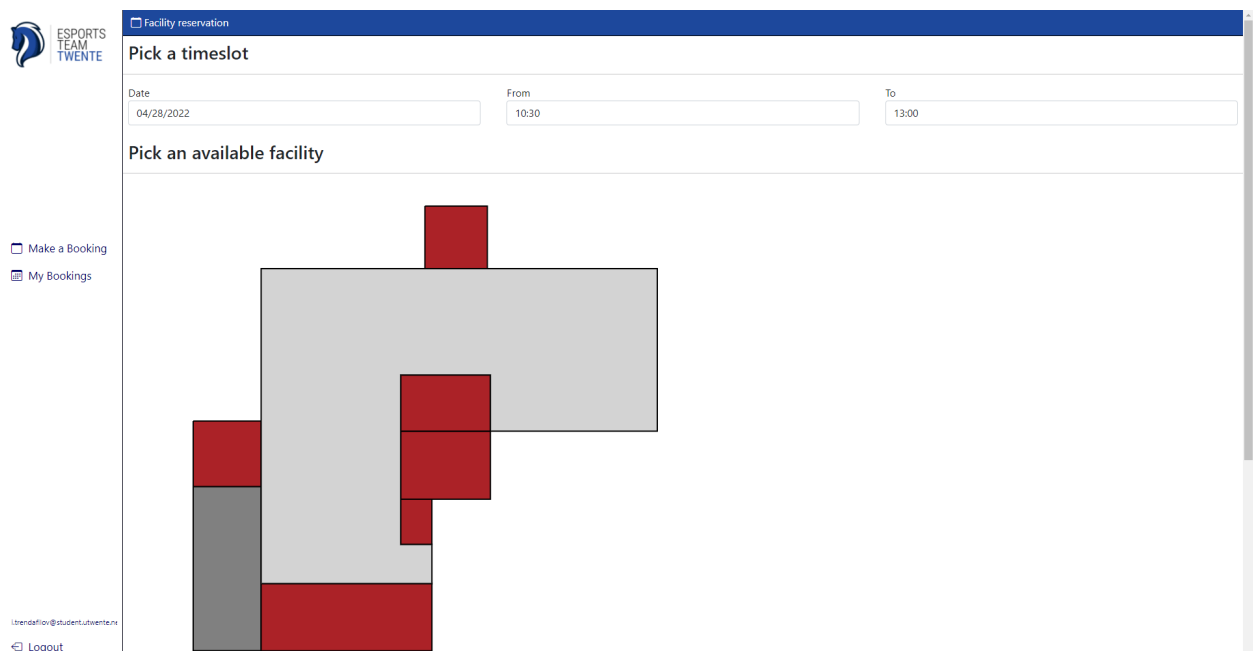
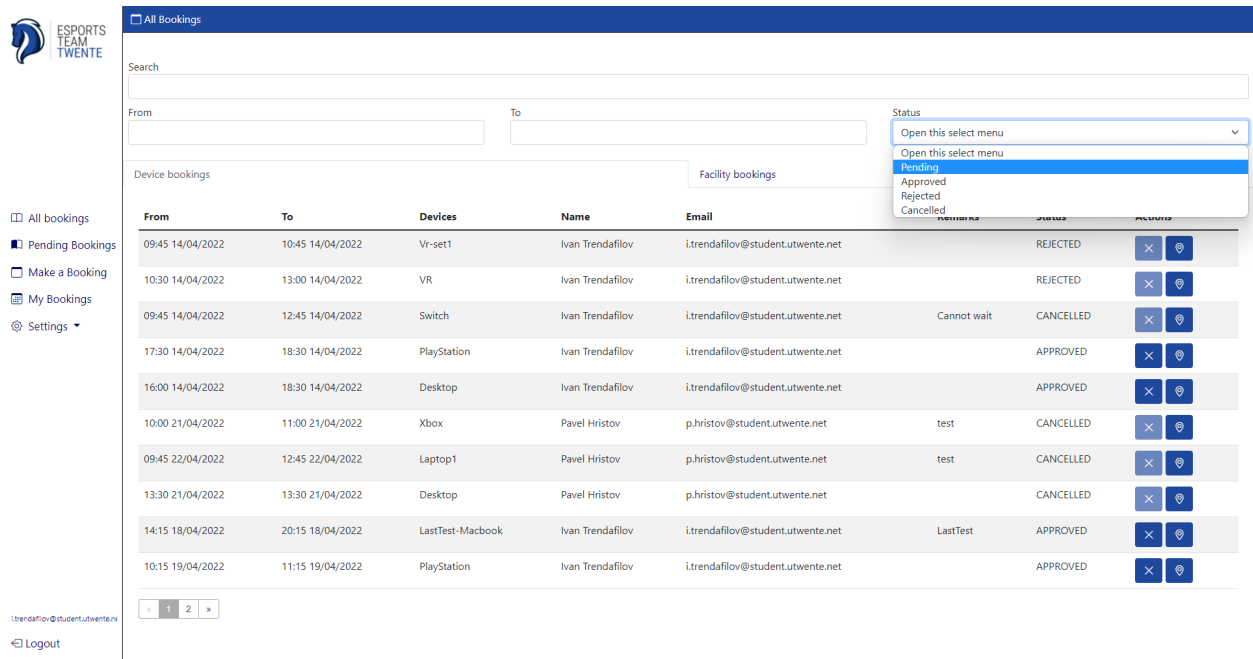


Figure 22: Facility booking page for users to select what facility they want to book for which date and for how long.

Administrator Interface



All Bookings

Search

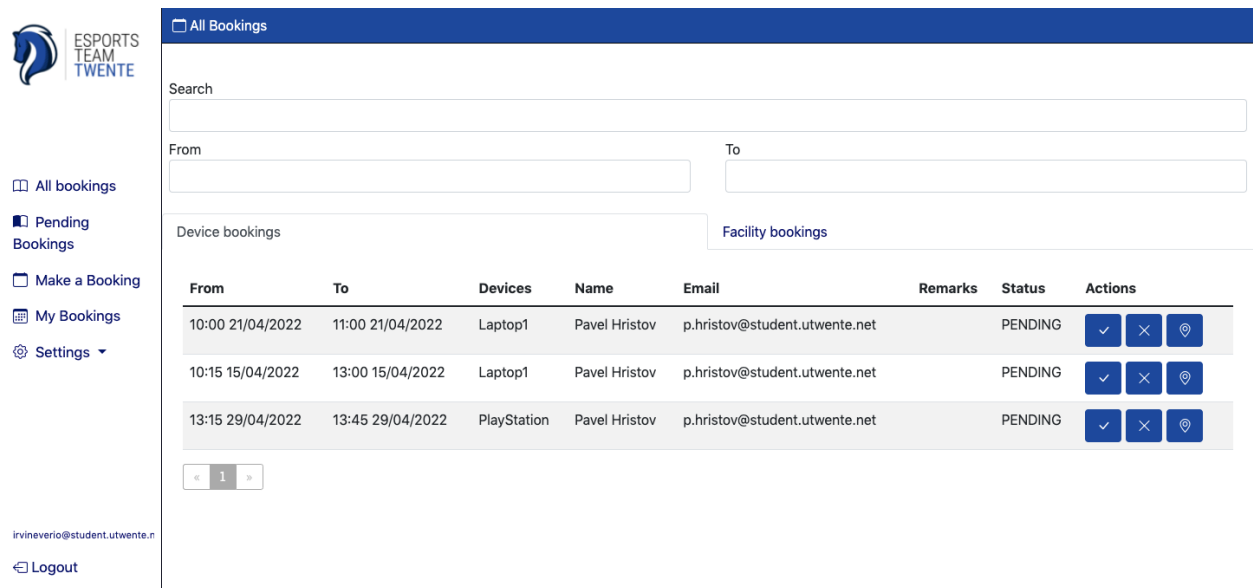
From: To: Status:

Device bookings ☐ Facility bookings ☐

From	To	Devices	Name	Email	Remarks	Status	Actions
09:45 14/04/2022	10:45 14/04/2022	Vr-set1	Ivan Trendaflov	i.trendaflov@student.utwente.net		REJECTED	<input type="button" value="X"/> <input type="button" value="🔔"/>
10:30 14/04/2022	13:00 14/04/2022	VR	Ivan Trendaflov	i.trendaflov@student.utwente.net		REJECTED	<input type="button" value="X"/> <input type="button" value="🔔"/>
09:45 14/04/2022	12:45 14/04/2022	Switch	Ivan Trendaflov	i.trendaflov@student.utwente.net	Cannot wait	CANCELLED	<input type="button" value="X"/> <input type="button" value="🔔"/>
17:30 14/04/2022	18:30 14/04/2022	PlayStation	Ivan Trendaflov	i.trendaflov@student.utwente.net		APPROVED	<input type="button" value="X"/> <input type="button" value="🔔"/>
16:00 14/04/2022	18:30 14/04/2022	Desktop	Ivan Trendaflov	i.trendaflov@student.utwente.net		APPROVED	<input type="button" value="X"/> <input type="button" value="🔔"/>
10:00 21/04/2022	11:00 21/04/2022	Xbox	Pavel Hristov	p.hristov@student.utwente.net	test	CANCELLED	<input type="button" value="X"/> <input type="button" value="🔔"/>
09:45 22/04/2022	12:45 22/04/2022	Laptop1	Pavel Hristov	p.hristov@student.utwente.net	test	CANCELLED	<input type="button" value="X"/> <input type="button" value="🔔"/>
13:30 21/04/2022	13:30 21/04/2022	Desktop	Pavel Hristov	p.hristov@student.utwente.net		CANCELLED	<input type="button" value="X"/> <input type="button" value="🔔"/>
14:15 18/04/2022	20:15 18/04/2022	LastTest-Macbook	Ivan Trendaflov	i.trendaflov@student.utwente.net	LastTest	APPROVED	<input type="button" value="X"/> <input type="button" value="🔔"/>
10:15 19/04/2022	11:15 19/04/2022	PlayStation	Ivan Trendaflov	i.trendaflov@student.utwente.net		APPROVED	<input type="button" value="X"/> <input type="button" value="🔔"/>

i.trendaflov@student.utwente.net

Figure 23: Administrator interface home page to see all available bookings and filters for either device booking, facility booking, status of the booking, or time period as well as a search to find specific bookings.



All Bookings

Search


From: To:

Device bookings ☐ Facility bookings ☐

From	To	Devices	Name	Email	Remarks	Status	Actions
10:00 21/04/2022	11:00 21/04/2022	Laptop1	Pavel Hristov	p.hristov@student.utwente.net		PENDING	<input type="button" value="✓"/> <input type="button" value="X"/> <input type="button" value="🔔"/>
10:15 15/04/2022	13:00 15/04/2022	Laptop1	Pavel Hristov	p.hristov@student.utwente.net		PENDING	<input type="button" value="✓"/> <input type="button" value="X"/> <input type="button" value="🔔"/>
13:15 29/04/2022	13:45 29/04/2022	PlayStation	Pavel Hristov	p.hristov@student.utwente.net		PENDING	<input type="button" value="✓"/> <input type="button" value="X"/> <input type="button" value="🔔"/>

irvineverio@student.utwente.net

Figure 24: The pending bookings page similar to the all bookings page but shows only the bookings that need to be accepted/denied.



ESPORTS TEAM TWENTE

All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

irvineverio@student.utwente.nl

Logout

Settings

☐ Auto acceptance of device reservations

☐ Auto acceptance of facility reservations

Week schedule

Week schedule2

Max booking time length


180

Max number of devices per reservation

1

Save

Figure 25: General settings page of the lounge reservation system.



ESPORTS TEAM TWENTE

All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

ibrendafiov@student.utwente.nl


Logout

Week schedules

+ Create new week schedule

Name	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	
LastTest	LastTest	Normal day schedule	LastTest	Normal day schedule	LastTest	Day off	Day off	<input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 26: Week schedule overview page where you can create/edit/delete week schedules.



All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

Create Week schedule

Name

Weekschedule name

Monday

Normal day schedule

Tuesday

Normal day schedule

Wednesday

Normal day schedule

Thursday

Normal day schedule

Friday

Normal day schedule

Saturday


Normal day schedule

Sunday

Normal day schedule

Create

Figure 27: Create week schedule page.



All bookings

Pending Bookings

Make a Booking

My Bookings


Settings

Day schedules

Create new day schedule

Name	Working time	Day off	Actions
Normal day schedule	09:45 - 13:45/14:45 - 18:30	×	Edit Delete
Day off		✓	Edit Delete
LastTest	10:30 - 12:30/14:00 - 22:30	×	Edit Delete

Figure 28: Day schedule overview where you can create/edit/delete day schedules.



All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

Create Day schedule

Name

Name

From

Selected time

To

Selected time

Add more times

☐ Day off

Create

Figure 29: Create day schedule page.

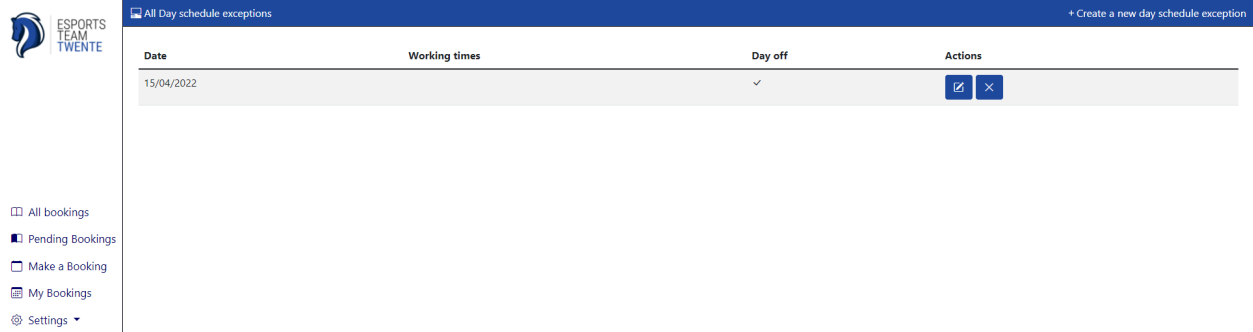


Figure 30: Day schedule exception page where you can create/edit/delete day schedule exceptions.

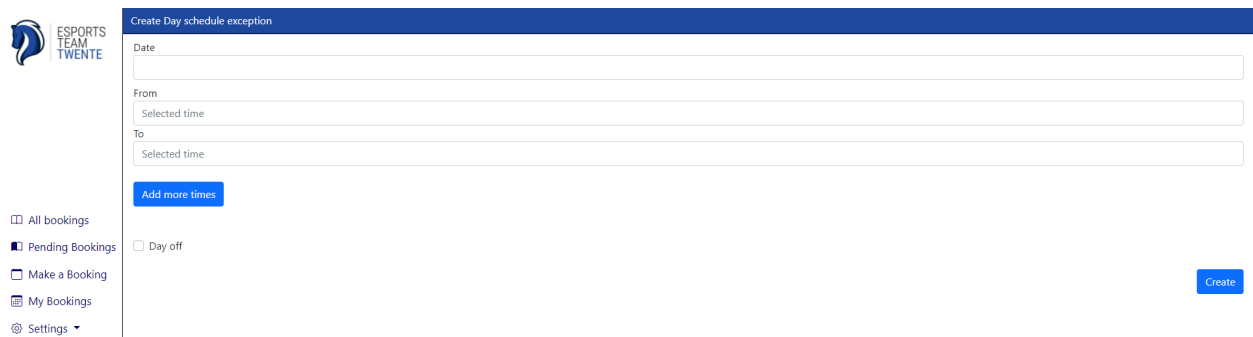


Figure 31: Day schedule exception creation page.

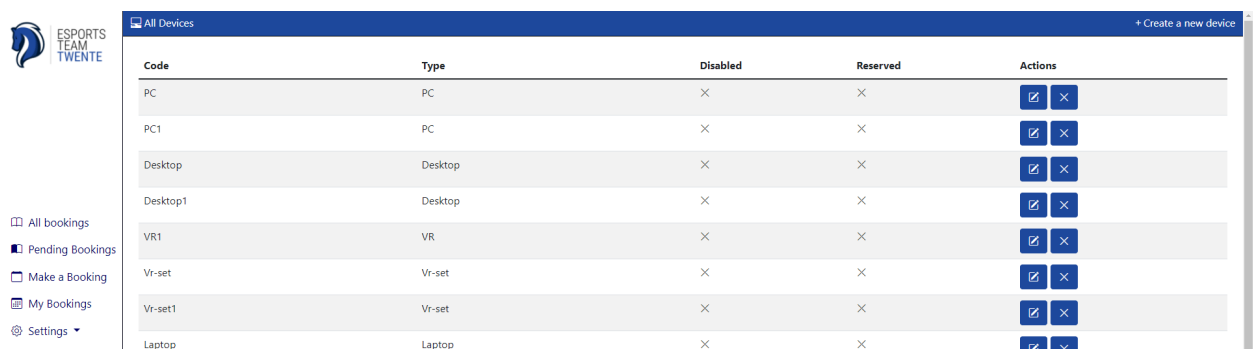


Figure 32: All device overview page where you can create/edit/delete devices.

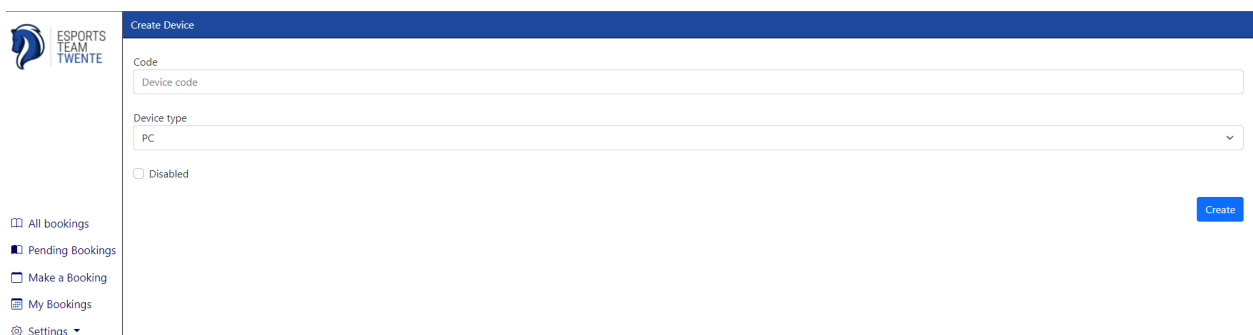



Figure 33: Create device page.



All bookings

Pending Bookings

Make a Booking

My Bookings

Settings ▾

All Devices Types

+ Create a new device type

Name	Image	Actions
PC	bi bi-pc	<div><div></div><div></div></div>
Desktop	bi bi-pc-display	<div><div></div><div></div></div>
VR	bi bi-badge-vr-fill	<div><div></div><div></div></div>
Vr-set	bi bi-badge-vr	<div><div></div><div></div></div>
Laptop	bi bi-laptop	<div><div></div><div></div></div>
PlayStation	bi bi-playstation	<div><div></div><div></div></div>
Xbox	bi bi-xbox	<div><div></div><div></div></div>
Switch	bi bi-nintendo-switch	<div><div></div><div></div></div>
LastTest-Macbook	bi bi-apple	<div><div></div><div></div></div>

Figure 34: Overview of all device types page where you can create/edit/delete device types.

<ul style="list-style-type: none"> All bookings Pending Bookings Make a Booking My Bookings Settings 	<div>Create Device Type</div> <div>Name</div> <div><input type="text"/></div> <div>Image class</div> <div><input type="text"/></div> <div>Create</div>
---	--

Figure 35: Create device type page.

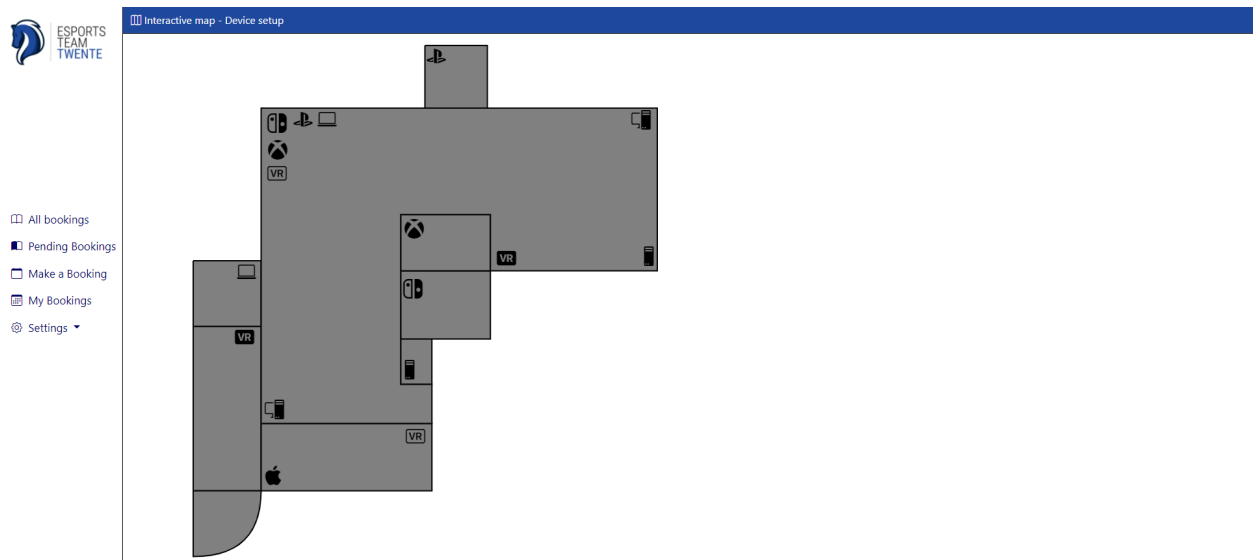


Figure 36: Interactive map page where you can put all the devices wherever you want.

All Facilities

Create a new facility

Name	Facility types	Room ID	Disabled	Reserved	Actions
Broadcast	BROADCAST	2	×	×	
Both	BROADCAST, TOURNAMENT	10	×	×	
LastTest	TOURNAMENT	12	×	×	

All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

Figure 37: Overview of all facilities page where you can create/edit/delete facilities.

Create Facility

Name

Facility name

Facility types

Disabled

trendafilov@student.utwente.nl

Figure 38: Create facility page.

All Admins

Create new admin

Name	Email	Actions
Pavel Hristov	p.hristov@student.utwente.net	
Ivan Trendafilov	i.trendafilov@student.utwente.net	
Jose	j.a.pratdesabalopez@student.utwente.net	
Viktor Tonchev	v.y.tonchev@student.utwente.net	
Boris Belchev	b.belchev@student.utwente.net	
Irvine Verio	irvineverio@student.utwente.net	

All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

Figure 39: Overview of all admins page where you can create/edit/delete admins.

Create Admin

Name

Email

Create

All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

Figure 40: Create admin page where you create admins.

Manual

Administrator Manual

All bookings

Pending Bookings

Make a Booking

My Bookings

Settings

Change opening time schedule

1. Login to the website
2. Go to the "Settings" menu
3. Go to the "Day schedules" sub menu
4. Add a new desired opening time for a day, from the button "Create new day schedule"
5. After that, go to the "Week schedules" sub menu
6. Click on "Create new week schedule" and fill in all the fields
7. Then, go to the "General settings" sub menu
8. Finally, change the "Lounge schedule" according to the desired week schedule
9. If you want to change the day schedule only for a specific date, you have to go to the Day schedule exception sub menu and create it for that date. This will change it only for the date you specified

Add devices

1. Login to the website
2. Go to the "Settings" menu
3. Go to "Device types"
4. Create a new device type from the button "Create new device type", on "Image class" you have to go to <https://icons.getbootstrap.com/> and filter on what you are searching for
5. For example you want to add PC icon you search for PC in the filter click on the icon, and under "Icon Font" you copy only the name example for PC is class="bi bi-pc" you copy only bi bi-pc without the brackets and paste it into the "Image class"
6. Go to "Devices" and click "Create new device", then add name and select the device type
7. Go to "Interactive map", put the device where you want and save

Create a facility

1. Login to the website
2. Go to the "Settings" menu
3. Go to "Facility"
4. Go to "Create new facility button", and create a facility

1trendatlov@student.utwente.nl

Figure 41: Manual page for admins with instructions on how to use different parts of the system.