

Classroom utilization optimization via mobile application

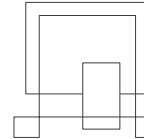
**Automatic registration of students' attendance in lectures to
optimize future classroom utilization**

**Jordi Larzo (313296),
Nils Franke (313446),
Anne Schneider (313444)**

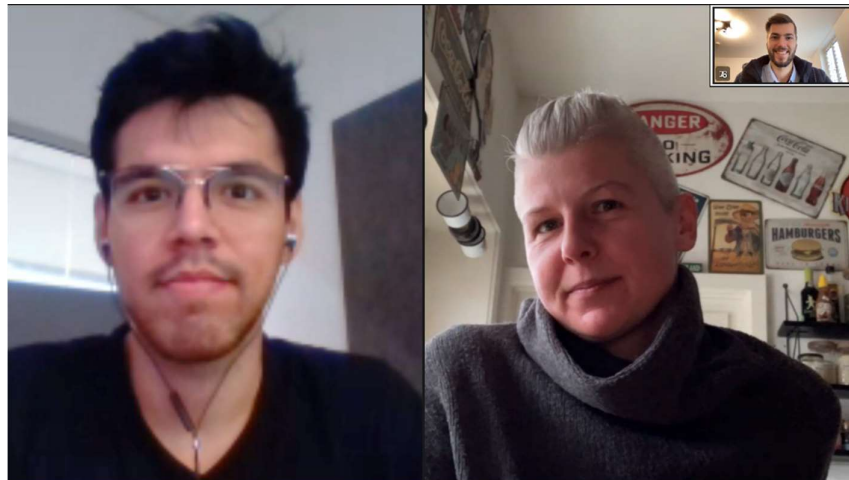
**Supervised by:
Poul Væggemose (POV)
Lene Overgaard Sørensen (LEOS)**

**Characters: 83,859
ENG-FPRPM-A21: International Project within Business
and Communication (GBE)
ENG-FPRPM-A21: International Project within Software
Engineering (ICT)**

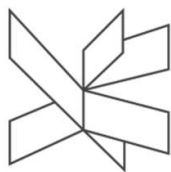
**Semester A2021
2021-12-17**



The project team



The company



VIA University
College

**VIA University College
Campus Horsens**

Banegårdsgade 2
DK - 8700 Horsens
T: +45 87 55 00 20
E: horsens@via.dk

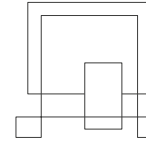
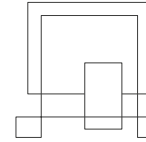
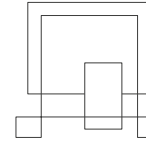


Table of content

List of figures and tables	v
Glossary	vi
Abstract	1
Executive Summary	2
1 Introduction	4
2 Analysis.....	6
2.1 Requirements.....	6
2.2 Functional Requirements	7
2.2.1 Actor Description.....	9
2.2.2 User Stories	10
2.2.2.1 End user student.....	11
2.2.2.2 End user professor	12
2.2.2.3 End user schedule manager	13
2.2.3 Use Case Diagram.....	14
2.2.4 Use Case Descriptions.....	15
2.2.4.1 Use Case 1: End user login	15
2.2.4.2 Use Case 2: Register attendance for lecture.....	15
2.2.4.3 Use Case 3: Check of current attendance	17
2.2.4.4 Use Case 4: Check of past attendances	17
2.2.4.5 Use Case 5: Check average attendance.....	18
2.3 Non-Functional Requirements.....	18
2.4 Test Cases.....	20
2.5 Domain Model.....	21
3 Design.....	23
3.1 Design Tools	23
3.2 Design Principles	24
3.3 User Interface Workflow	29
4 Implementation.....	30
4.1 Dependencies	31
4.2 Class QrScannerActivity	33
4.3 Class CheckRoomFragment	37
5 Software Testing	38

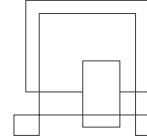


5.1	Testing tools	41
5.1.1	USB-Debugging	42
5.1.2	Debugging.....	42
5.2	Test Case Specifications.....	43
5.2.1	Test scenario 1: login page validation	44
5.2.2	Test scenario 2: end user profile validation	45
5.2.3	Test scenario 3: lecture attendance registry validation	46
5.2.4	Test scenario 4: wireless communication validation	48
5.2.5	Test scenario 5: history option validation.....	49
5.2.6	Test scenario 6: system performance validation.....	51
5.3	Results.....	53
5.3.1	Test results scenario 1	55
5.3.2	Test results scenario 2	55
5.3.3	Test results scenario 3	56
5.3.4	Test results scenario 4	56
5.3.5	Test results scenario 5	57
5.3.6	Test results scenario 6	58
5.4	Incident report.....	58
6	Results and Discussion	59
7	Conclusions.....	61
8	Project future.....	62
8.1	Political aspects	63
8.2	Economical aspects	63
8.3	Social aspects.....	63
8.4	Technological aspects.....	64
8.5	Environmental aspects.....	64
8.6	Legal aspects.....	65
9	Source of information	66
10	Appendices	68
	Appendix A User Interface Workflow Student	68
	Appendix B User Interface Workflow Professor.....	71
	Appendix C User Interface Workflow Schedule Manager	73



List of figures and tables

Table 2.1: functional requirements	8
Table 2.2: non-functional requirements	19
Table 2.3: overview on test cases for system requirements	21
Table 3.1: prototype development platform comparison	23
Table 3.2: coding software comparison	24
Table 4.1: dependencies	32
Table 4.2: class QrScannerActivity	34
Table 4.3: database information handling	35
Table 4.4: class CheckRoomFragment	38
Table 5.1: overview of incidents during testing	58
Figure 2.1: application concept	10
Figure 2.2: use case diagram (UCD)	14
Figure 2.3: domain model of application	22
Figure 3.1: home screen - end user student (prototype)	26
Figure 3.2: current attendance - end user professor (prototype)	27
Figure 3.3: history data selection (prototype)	28
Figure 3.4: date selection - calendar function (prototype)	28
Figure 5.1: V-Model software testing	39
Figure 8.1: PESTEL analysis board	62
Figure A. 1: workflow login (end user student)	68
Figure A. 2: workflow registry to lecture (end user student)	69
Figure A. 3: workflow histroy & latest attendance (end user student)	70
Figure A. 4: workflow attendance rate (end user professor)	71
Figure A. 5: workflow history data (end user professor)	72
Figure A. 6: workflow average room occupation (end user schedule manager)	73
Figure A. 7: workflow history data (end user scheduler)	74



Glossary

Attending student: student who is being present at the lecture

Classroom: the room in which a lecture with several students takes place

Current attendance: screen within app (end user: professor) to check actual number of attending students (live counter)

End user: target group of the app, i.e. person who will actually use the application. In this case defined as student, professor and schedule manager

Home screen: first screen of the app after login into user profile

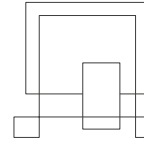
Lecture: an educational unit attended by several students. A lecture is held by one professor only and refers in this case to the regular study program (no project work)

Latest attendance: screen within app (end user: student) to see last registry for a lecture, i.e. the last ID that was scanned via QR code

Login screen: first screen of the app where end users can log into their profile

Registered student: student who is enrolled in the lecture

Schedule manager: employee at VIA university, responsible for scheduling the lectures to the classrooms according to the timetables of all different study programs



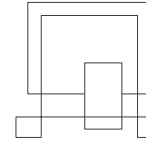
Abstract

With the move into a new facility in August 2021, the VIA University College campus in Horsens was facing the problem of too less classroom resources available for the high number of registered students. The project follows the approach of classroom utilization optimization by collecting data about the real attendance of students in regular lectures via mobile application.

The proof-of-concept evaluated and developed enable students to register their attendance via QR code providing real-time data about lecture attendances. The data collected is analysed (average, deviation) and provided as history data for professors and schedule manager in order to improve lecture quality and resource utilization.

Within a help of several business tool (value proposition canvas, customer journey and MoSCoW prioritisation) the different requirements of the target groups were evaluated and for the most part successfully implemented into the prototype. The development process was executed according to IEEE standards and under consideration of design guidelines in order to provide the market-ready application interface and functionality.

Concluded to the development the analysis of future aspects for a product launch show the high potential of attendance data collection in terms of classroom resources and budget savings as well as the improvement of lecture quality and final student grades.



Executive Summary

Background

In August 2021, the VIA University College moved its campus in Horsens into a new facility offering its student an improvement of the education quality provided with more space to exchange with others and with quiet zone allowing a full focus on their studies. With the increase of small classrooms available for group work the schedule manager is facing the problem of available classrooms suitable for all student registered to the different lectures of all the different programs provided.

Project Opportunity

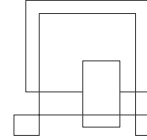
The project team sees an opportunity to optimize the utilization of classroom resources by collecting data about the actual attendance of students in lectures. With a database allowing to represent the past attendance rate of regular lectures, the schedule manager can allocate lectures based on the real demand of seats and professors can review their lecture quality. The students itself can track their own attendance rate and could see a relation to their grades in exams.

Problems to be solved

- Balance of available room resources and the high number of registered students through various study programs
- Lecture allocation based on number of registered students without any insights of how many students are actual attending the lecture
- short-term scheduling and potential waste of budget for unused classroom capacities

Project Outcome

The project successfully created an application concept, that would allow the real-time lecture attendance of students via QR code scan. With an easy understandable and consistent functional GUI design the different requirements of the students, professors and schedule manager were implemented into an application prototype.

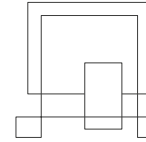


Projects Vision

- allocation of lectures based on attendance rate to save money and capacity in terms of room resources
- facilitation of lecture allocation decision due to availability of real time attendance data, or rather detachment of allocation decision from number of registered students
- improvement of lecture quality based on low attendance rates
- enhanced decision support in terms of study program related budgets allocations
- improvement of final grades based on higher attendance in lectures after review of relation between grades and attendance rates

Recommendation

With the developed proof-of-concept, the project team recommend to expand the application features to ensure not only a high rate of end users (especially students), but also to allow a certain level of data security and integrity. The scanning of the QR code could allow the confirmation of attendance in case students are about to enter the wrong classroom. The profile of the schedule manager could be expanded to see also the lecture attendance rate to give more insights into past data. For students, the main target group, an analysis about their past attendance rates and final grades could be implemented.



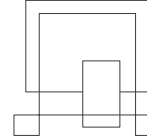
1 Introduction

During the past ten years the number of students enrolling in Danish universities has increased. From 2010 to 2020 Danish universities has experienced a 18% increase of student enrolments with a total of roughly 151,000 registered students in 2020. (Statista, 2021) With the growing number of enrolling students, lectures and classrooms have reached maximum capacity leading into more and more problems of scheduling the lectures at appropriate time slots for students and for professors.

To schedule the lectures to fit into the timetable and into the available classrooms in the best way possible is known as the University Class Scheduling Problem (UCSP). Given its wide use in colleges and universities worldwide, UCSP have been studied for decades in order to use resources effectively and economically. However, most of the studies refer to metaheuristic algorithms and computerized software solutions, which are not applicable for many universities. (Alobaedy & Ku-Mahamud, 2014)

The VIA University College is one of six university colleges in Denmark, offering more than 40 programs for more than 20,000 students, of which 2,300 are international. One of the eight campuses is located in Horsens. (VIA University College, 2021) In August 2021, the Horsens campus moved into a new facility, offering its student more space to learn, to exchange with others, but also to find quiet zones to focus on their studies and therefore, improve the quality of education provided. (VIA University College Horsens, 2021)

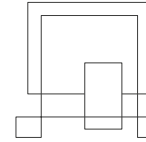
With the move to its new facility and the increase of small classrooms available for group work (suitable for up to six people) the schedule manager of VIA Horsens is facing the problem of available classrooms suitable for all student registered to the different lectures of all the different programs provided. Furthermore, special attention is paid to the students of the first semester as they need more time to get used to the new environment and therefore, are preferred to have the same classroom schedule every week, if possible.



So far, all classrooms are scheduled with the constraints of seats available in the classrooms and the number of registered students for the lectures. Regardless the time-consuming process, many times the current schedules result in weekly changes of classrooms for the same lecture and a low efficiency in classroom capacities due to the number of not-attending students.

Unlike the previous mentioned mathematic approaches to the UCSP, this project follows the idea of improving future classroom utilization by collecting data of real-time classroom occupation based on the number of actual attending students (see Project Description, chapter 2 and chapter 3) via mobile application. A representative history of lecture-attending students does not only provide the opportunities to schedule classrooms more efficiently and economically, it will also help students to improve their academic performance and offers an detailed retro perspective on the lecture quality for professors.

The following report describes the development of the mobile application as a proof-of-concept. Next to the technical details, the report focuses on the economic and social aspects of the application and evaluates the future potential of the concept.



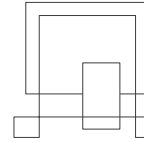
2 Analysis

The analysis phase is the first phase within the software development lifecycle (SDLC). Within this phase precise requirements from the customer (end user) are collected and aligned with a fine-tuned solution to their need. This phase also identifies any potential risk and defines all weak point of the project (see Project Description, chapter 7). The analysis phase is also used to understand and picture the real scope of the project (see Project Description, chapter 3 and chapter 4) as well its anticipated issues and its opportunities.

In the following the collection and analysis of the project related business and system requirements are presented. Next to the classification of those requirements (see chapter 2.2 and chapter 2.3) an overview of what and how the requirements will be tested (see chapter 2.4) is shown. Finally, the key concept of the project is visualized as a domain model (see chapter 2.5).

2.1 Requirements

Clearly defined requirements are the backbone of a successful project and establish a formal agreement between the client (VIA, end users) and the provider (project team). (Altexsoft, 2021) Requirements are the description of functionalities and features of a target system and can be obvious, known and expected, or the opposite of it. In order to achieve the most basic version of the project's outcome (minimal valuable product, MVP), it is essential to define all requirements that needs to be fulfilled from a business, client and developer perspective. With a potentially long list of requirements, it is also crucial to prioritize them. The outcome of this prioritization is an ordering of requirements which need to be considered first in the process during the developing process. (Dabbagh & Lee, 2014)



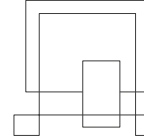
Within this project all requirements are based on the evaluated subproblems (see Project Description, chapter 2.1) and classified into business, stakeholder and solution requirements (in accordance with BABOK¹). Business requirements do not have specific features and state only the business objective to be achieved (see chapter 2.3). The stakeholder or end user requirements are a definition of what is expected from the provided solution and build a bridge between the business and solution requirements. Solution requirements are distinguished into functional (FR) and non-functional requirements (NFR) describing specific characteristics the product or system must have to meet the end users and business needs. (Altexsoft, 2021) While FR describes what the system does, NFR are used to describe the systems properties, also known as quality attributes. All requirements were defined according to the IEEE standard for software requirement specifications. (IEEE Computer Society, 2018)

For the project, all requirements were evaluated by interviews with the schedule manager and supervisor (representative end user “professor”), and by brainstorming within the project team (representative end user “student”). The evaluated requirements were then prioritized based on the MoSCoW method with no differentiation between the requirement classes. In the following, FRs and NFRs will be viewed separately as both require different aspects of presentation and description.

2.2 Functional Requirements

FR are functions or features need to be implemented to enable the end user to accomplish their task in order to fulfill their need. They describe how the system will react to certain inputs, what kind of services the system will or should provide, and also what the system should not do. (Laplante, 2014, p. 12)

¹ *Business Analysis Body of Knowledge, standard for practice of business analysis*

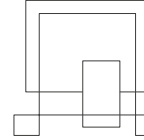


There are several forms of how FR can be presented, for this project a combination of visual models (Use Case Diagram, see chapter 2.2.3) and written text (Use Cases, see chapter 2.2.4) was used. Use cases describe the interaction between the external users and the system and include the elements of actor, system and goals. (Altexsoft, 2021) All elements are used to represent a sequence of events necessary to fulfill the end users task (User Story).

The following table (see Table 2.1) shows an overview of the FRs in order to their assigned priority, beginning with #1 as the most important.

ID	DESCRIPTION	PRIO	ACCEPTANCE CRITERIA	SUB AREA	CATEGORY
1	real-time entrance tracking (time stamp) by QR code scan	must have	accurate time stamp transmission to database	data quality	solution
2	wireless communication to database	must have	no further tool for data transmission necessary	data quality	solution
3	password protection on login	must have	password request for login	data quality	solution
4	real-time notification on database connection error	must have	immediate pop-up on app screen	data quality	solution
5	different end user roles & rights (logins)	must have	different log in options, user profile specifications	end users	stakeholder
6	availability of data history	must have	logbook of past collected data	useability	stakeholder

Table 2.1: functional requirements



2.2.1 Actor Description

In preparation to the use cases (see chapter 2.2.3) the actors need to be defined. The application will be used by three different types of external users. End users will act as primary actors here. The primary actors are: the student, the professor, and the schedule manager.

The student is the main subject who will use the application to login to the lecture's classroom during entrance. The student will use the application to observe his lecture's attendances over time. He is also the key target group for the application as it is the only actor generating data for future usage (database).

The professor is an actor with information access only. The professor will use the application to observe the analytics of the students' attendance of his lectures provided by the database.

The schedule manager is also an actor with information access only, but with no limitation regarding classrooms and lectures. The schedule manager will use the application to observe the real classroom occupation in relation to the scheduled lectures.

Next to the primary actors the system will interact with an external, secondary actor: NetScaler (VIA login system). NetScaler is the system that allows the end users to access their university account and therefore will give access to the application.

The following (see Figure 2.1) gives a high-level overview about the application concept idea.

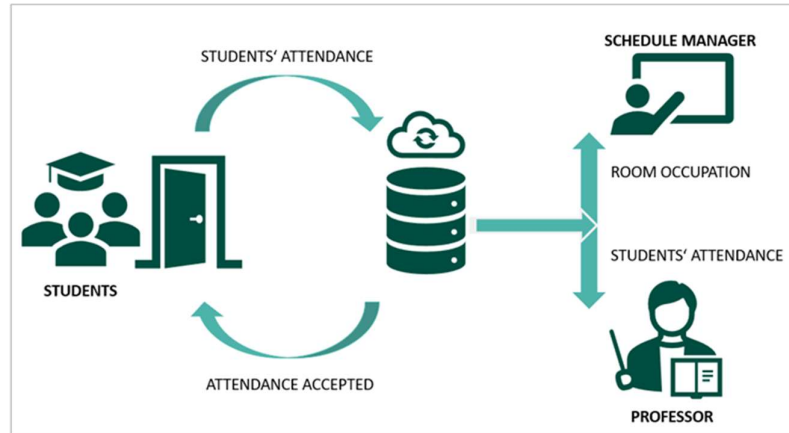
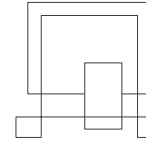
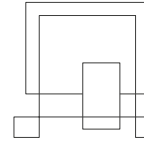


Figure 2.1: application concept

2.2.2 User Stories

The user stories represent a task the end user wants to complete in order to achieve a specific goal. All user stories are described from an end user perspective and are following a specific template of structure: as an [end user] I want to [perform an action] so that I can [achieve a goal]. To ensure the later acceptance of the product by the end users, the user stories also need to include acceptance criteria, which will be related to the test cases (see chapter 2.4).

Considering the systems interaction with the different types of end users (primary actors) user stories are defined.



2.2.2.1 End user student

#1: As a student, I want to login to my account providing my ID and password, so that I can login to the lecture's classroom.

Ensure the user is able to:

- see login screen with Net Scaler Login
- type in VIA ID and password
- access app after (successful) login (displayed home screen)

#2: As a student, I want to register to a lecture, so I can track my lectures' attendance.

Ensure user is able to:

- see only student's profile (user profile settings by Net Scaler)
- click REGISTRY TO LECTURE (button)
- get notifications about mandatory camera access permission (pop up)
- is transferred to camera on mobile phone (reading QR code)
- is transferred back to app
- see the corresponding lecture (QR code) ID on screen

#3: As a student, I want to register to another lecture, so I can track all of my lectures' attendance.

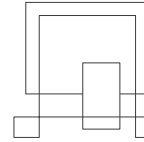
Ensure user is able to:

- see last registered lecture (classroom) "latest attendance"
- is able to create new registry (REGISTRY TO LECTURE)
- is transferred to camera on mobile phone ... (see story "register for lecture")

#4: As a student, I want to check my current attendance, so I can make sure I am registered for the lecture.

Ensure user is able to:

- see last login for lecture ("latest attendance")



#5: As a student, I don't want to allow access to my camera, so that I can protect my privacy.

Ensure user is able to:

- deny (DISAGREE) the mandatory camera access permission during lecture registry
- is transferred back to "home screen"

#6: As a student, I want to check past attendances, so I can check my attendance history for several lectures.

Ensure user is able to:

- see only students' profile (user profile settings by Net Scaler)
- is able to choose HISTORY in the menu
- is able to specify the lecture (dropdown) and the period (from and to, calendar function)
- is transferred to a new screen with the requested data (lecture ID, time stamp, chronological order with latest on top)

2.2.2.2 End user professor

#7: As a professor, I want to login to my account providing my ID and password, so that I can see the students' attendance in my lectures.

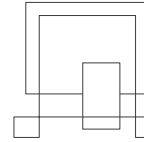
Ensure user is able to:

- see login screen with Net Scaler Login
- type in VIA credentials and password
- access app after (successful) login (displayed home screen)

#8: As a professor, I want to check the attendance of the current lecture, so that I can see how many students are currently attending the lecture.

Ensure the user is able to:

- see only professors' profile (user profile settings by Net Scaler)
 - click CHECK CURRENT ATTENDANCE from the menu
-



- see corresponding lecture on screen
- see also current number of registered users (attending students) and assigned students
- is able to refresh the screen to see the actual number of attending students (refresh option)

#9: As a professor, I want to check the average attendance of my lecture, so that I can see how many students are attending over time.

Ensure the user is able to:

- see only professors' profile (user profile settings by Net Scaler)
- click HISTORY from the menu
- is able to specify the period for the lecture (from and to, calendar function)
- is transferred to a new screen with the requested data (lecture ID, average attendance, deviation)

2.2.2.3 End user schedule manager

#10: As a schedule manager, I want to login to my account providing my ID and password, so that I can get the real-time occupation of lectures' classrooms.

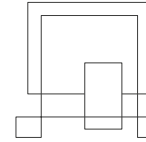
Ensure user is able to:

- see login screen with Net Scaler Login
- type in VIA credentials and password
- access app after (successful) login (displayed home screen)

#11: As a schedule manager, I want to see the history occupation of a specific classroom, so I can get knowledge about the real occupation rate of the classroom over time.

Ensure the user is able to:

- see only schedule manager profile (user profile settings by Net Scaler)
- is able to choose the classroom ID (drop down)
- is able to specify the period (from and to, calendar function)



- is transferred to a new screen with the requested data (lecture ID, average of occupied seats, number of available seats, deviation)

2.2.3 Use Case Diagram

An use case diagram is a summary of the relationships between actors, system and their actions (use cases) and represents a high-level overview. It does not show in which steps or in which order the steps within the use cases are performed.

For the project, the system use case diagram was created (see Figure 2.2) representing the actors (student, professor, schedule manager, NetScaler AAA), the system (“Attending VIA”) and their corresponding use cases.

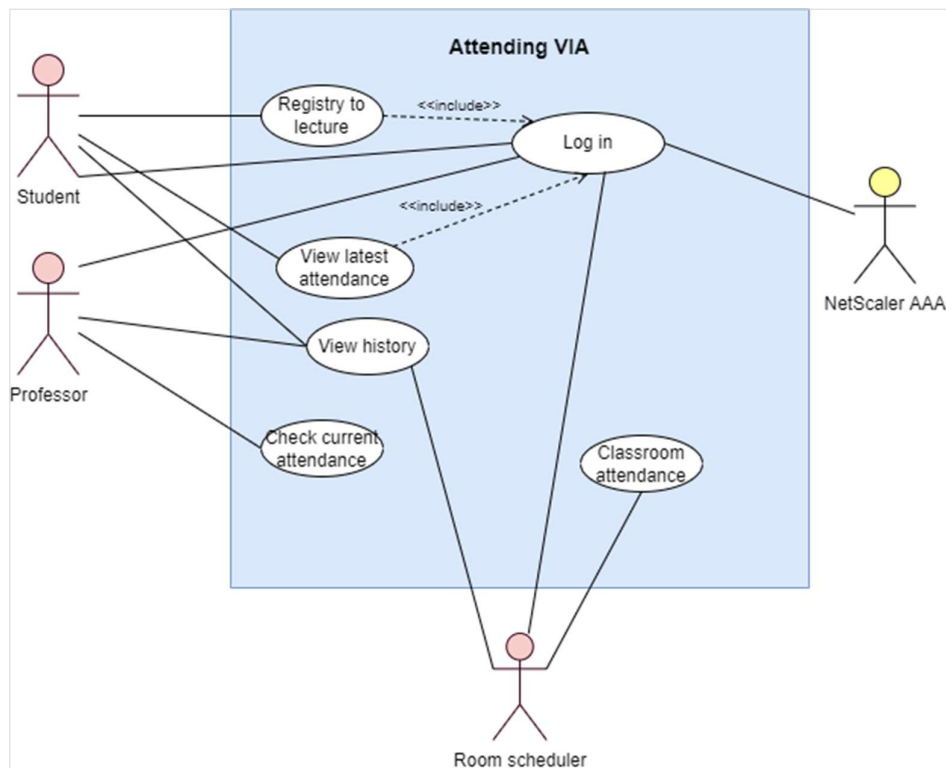
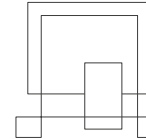


Figure 2.2: use case diagram (UCD)



2.2.4 Use Case Descriptions

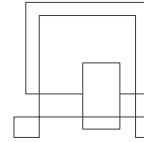
Based on the evaluated user stories (see chapter 2.2.2) different use cases were defined.

2.2.4.1 Use Case 1: End user login

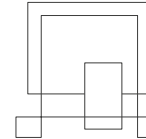
UC 1	
Use Case Title:	End user login (#1, #7, #10)
Actor:	End user, Net Scaler
Precondition:	Internet connection, Net Scaler account
Description:	general login to application every time the user wants to use its functionalities
Primary flow	
<ol style="list-style-type: none"> 1. System displays login screen with login button 2. End user clicks login (button) and systems transfers end user to Net Scaler login screen 3. End user types in login information (VIA ID, password) and confirms (button) 4. System gets approval from Net Scaler and transfers end user back to application home screen with a confirmation of a successful login 5. End users sees application home screen 	
Outcome:	
Success , end user identification and transfer to home screen (of user profile)	
Alternate flow 3.1A	
<ol style="list-style-type: none"> 1. End user types in wrong login information (VIA ID and password do not match) 2. Systems gets no approval from Net Scaler and transfers end user back to application main screen with notification of failed login 3. End users sees application main screen 	
Outcome:	
Failure ; no end user identification by system	

2.2.4.2 Use Case 2: Register attendance for lecture

UC 2	
Use Case Title:	Register attendance for lecture (#2, #3, #5)
Actor:	End user [student], database
Precondition:	Successful login, internet connection
Description:	Confirmation of registered lecture after successful scanning of the QR code (make sure it is the correct lecture)
Primary flow	
<ol style="list-style-type: none"> 1. End user clicks REGISTER (button) 	



<ol style="list-style-type: none"> 2. System transfers end user to camera module 3. End user scans QR code of lecture classroom 4. System translate QR into classroom ID and request corresponding lecture ID from database 5. System transfers end user back to application and displays lecture ID 6. End user confirms the attendance to the displayed lecture (press CHECK) 7. System transmits classroom ID, student ID and time stamp to database and displays the end user the confirmation of successful registry to the lecture (with ID))
Outcome:
Success , transmission of corresponding data to database, track of lecture attendance
Alternate flow 2A
<ol style="list-style-type: none"> 1. End user denies access to the camera module 2. Systems throws error notification of the denied access and transfers end user back to the application main screen
Outcome:
Failure ; system transmits no data, no attendance tracking
Alternate flow 3A
<ol style="list-style-type: none"> 1. End user cannot scan QR code (not readable) 2. System cannot translate QR code into classroom ID and displays error notification of failed registry to end user
Alternate flow 3B
<ol style="list-style-type: none"> 1. End user scans not valid QR code (no code for classroom) 2. System cannot translate QR code into classroom ID and displays error notification of failed registry to end user
Outcome:
Failure ; system transmits no data, no attendance tracking
Alternate flow 6A
<ol style="list-style-type: none"> 1. End user denies attendance to the displayed lecture ID (press "X") 2. Systems do not transmit any data to the database and confirms denial to end user with a notification about the failed registry
Outcome:
Failure ; system transmits no data, no attendance tracking



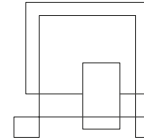
2.2.4.3 Use Case 3: Check of current attendance

UC 3	
Use Case Title:	Check attendances (#4)
Actor:	End user [student], database
Precondition:	Successful login, internet connection
Description:	End user wants to observe current attendance registration
Primary flow	
<ol style="list-style-type: none"> 1. End user select latest attendance (button) 2. System displays screen with latest attendance: lecture ID, classroom ID and date 	
Outcome:	
Success; system displays information about current attendance	

UC 4	
Use Case Title:	Check attendances (#8)
Actor:	End user [professor], database
Precondition:	Successful login, internet connection
Description:	End user wants to observe current attendance registration
Primary flow	
<ol style="list-style-type: none"> 1. End user select check current attendance (menu) 2. System displays screen with current attendance: lecture ID, number of attending students, number of registered students 	
Outcome:	
Success; systems displays information about current attendance	

2.2.4.4 Use Case 4: Check of past attendances

UC 5	
Use Case Title:	Check past attendances (#6)
Actor:	End user [student], database
Precondition:	Successful login, internet connection
Description:	End user wants to observe past lecture attendances
Primary flow	
<ol style="list-style-type: none"> 1. End user select history (button) 2. System displays screen for search function: lecture ID (drop down) and period (calendar function) for the end user 3. End user confirms selection (button: search/confirm) 4. Systems displays lecture ID, classroom ID and date in chronological order 	
Outcome:	
Success; system displays all past attendance data	



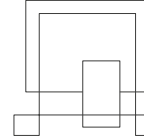
UC 6	
Use Case Title:	Check past attendances (#9)
Actor:	End user [professor], database
Precondition:	Successful login, internet connection
Description:	End user wants to observe past lecture attendances
Primary flow	
<ol style="list-style-type: none"> 1. End user select history (button) 2. System displays screen for search function with lecture ID and period (calendar function) for the end user 3. End user confirms selection (button: search/confirm) 4. Systems displays lecture ID, classroom ID and date in chronological order 	
Outcome:	
Success ; system displays all past attendance data	

2.2.4.5 Use Case 5: Check average attendance

UC 7	
Use Case Title:	Check average attendance of lectures (#11)
Actor:	End user [schedule manager], database
Precondition:	Successful login, internet connection
Description:	End user wants to observe actual lecture attendance
Primary flow	
<ol style="list-style-type: none"> 1. End user select lecture attendance (button) 2. System displays screen for search function: lecture ID (drop-down) and period (calendar function) for the end user to choose 3. End user confirms selection (button: search/confirm) 4. Systems displays lecture ID, average number of attendances, deviation of attendances, number of assigned students 	
Outcome:	
Success ; systems displays requested occupation data	

2.3 Non-Functional Requirements

NFRs distinguish themselves from FR by describing how the system shall do something in contrast to what the system will do. They describe the capabilities and constraints of a system that enhances its functionality and therefore plays a critical role during system development.

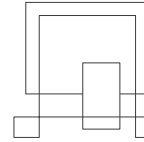


There are hundreds of different NFRs serving as operational qualities and selection criteria for choosing among different methods of design and implementation. (Chung, et al., 2000) The proper definition of NFRs is critical as an over-specification may lead to a too expensive solution, an underachievement could result in a for its intended use inadequate system. (Scaled Agile, 2021) Furthermore, the importance of NFRs vary and needs to be aligned to the system context. (Banger, n.d.)

There are different approaches how to categorizes NFRs, the most common categories are: security, audit, capacity, performance, availability, reliability, recoverability, robustness, integrity, maintainability, usability and documentation. Given the scope of the project and the time frame only the most important NFRs were evaluated and categorized as followed (see Table 2.2). Like the FRs (see chapter 2.2) all NFRs were prioritized by the MoSCoW method, showing the most important NFR as #1.

#	DESCRIPTION	ACCEPTANCE CRITERIA	CATEGORY
1	intuitive understandable GUI	according to GESTALT principles (see chapter 3)	usability
2	delay in data transmission	max 5 seconds	performance
3	simultaneous transmission of data	at least 5 students login at the same time	capacity
4	error notification on failed database connection	max 1 second delay	security
5	accurate calculation of attendance average and deviation	Max deviation of $\pm 1\%$	integrity
6	time to load screens, refresh screens	max 1 second	performance
7	time to export data as csv file	max 5 seconds	performance
8	modular & scalable system	integration of future adaption without changing initial stable solution	robustness
9	mobile application for smartphones	Available for download in application store	availability
10	free of charge for end users	No application charge, free for download	availability

Table 2.2: non-functional requirements



In regard to the business aspects there were FRs (#5) and NFRs (#9, #10) that could be evaluated during the pre-phase of the project (see Project Description, chapter 2). As the project was decided to be a proof-of-concept only and therefore will be developed and tested with simulated data only, the corresponding NFRs will not be considered further in the developing process. However, given the potential of the concept, the business aspects will be reviewed for the projects future (see chapter 8).

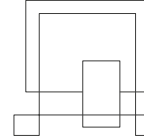
2.4 Test Cases

In order to execute the software testing, test case specifications (TCSs) need to be listed. The TCSs can be based on different test basis's, such as use cases, system or business requirements or functional and/or technical design specifications.

TCSs are a detailed summary of what will be tested, how it will be tested and how often it will be tested. Each TCS is unique and specifies the purpose of the test together with the required inputs and expected results. Like the use cases, TCS provide a step-by-step procedure and the acceptance criteria's for passing or failing the specific test.

For this project, the system and business requirements (see chapter 2.1) were used as test basis to define the different TCSs. The following (see Table 2.3) represent an high-level view on the test procedure and shows each requirement with its corresponding test case. The TCSs (see chapter 5.2), the test results (see chapter 5.3) and the assessment of those results (see chapter 5.4) are described in more detail in the following chapters.

As the requirement for the Graphical User Interface (GUI) to be intuitively understandable (NFR, #1) is difficult to be described in a single test case specification, this project follows the assumption of self-realisation. All end user specific TCSs are executed by a random selection of end users (testers) who will get the TCS instruction only. The fact that no user manual is (or rather should be) necessary to complete the steps, the intuitively comprehensibility of the GUI is assumed as self-evident and will be not considered in the test cases.



ID	DESCRIPTION	TCS
FR #1	real-time entrance tracking (time stamp) by QR code scan	TCS 6, TCS 7, TCS 8
FR #2	wireless communication to database	TCS 9
FR #3	password protection on login	TCS 1, TCS 2
FR #4	real-time notification on database connection error	TCS 10
FR #5	different end user roles & rights (logins)	TCS 3, TCS 4, TCS 5
FR #6	availability of data history	TCS 11, TCS 12, TCS 13
NFR #2	delay in data transmission	TCS 14
NFR #3	simultaneous transmission of data	TCS 17
NFR #4	error notification on failed database connection	TCS 9
NFR #5	Accurate calculation of attendance average and deviation	TCS 16
NFR #6	time to load screens, refresh screens	TCS 15
NFR #7	time to export data as csv file	
NFR #8	modular & scalable system	

Table 2.3: overview on test cases for system requirements

2.5 Domain Model

Domain models are used to describe the problem domain space. As a model of real-world entities, the relationship between them and their problem responsibilities, domain models derived from system-level requirements and provide an effective basis for understanding the problems core. (Scaled Agile, 2021)

Within the project and its addressed problem, the entities of the end users (student, professor and schedule manager), the attended lectures (*here*: courses) and its corresponding classrooms could be identified. In relation to the evaluated project model (see Figure 2.1) the domain model (see Figure 2.3) shows the relationship between the entities from a high-level perspective.

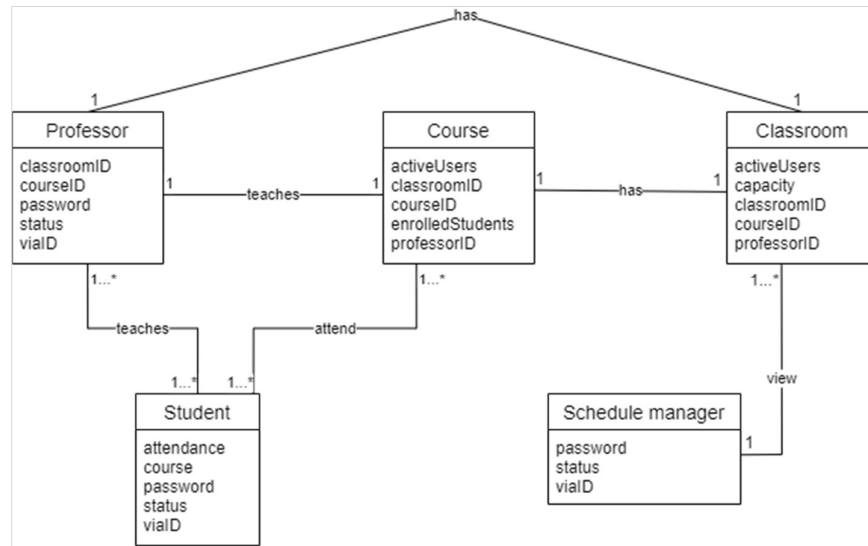
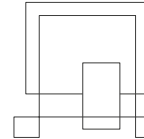
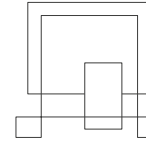


Figure 2.3: domain model of application

Every end user is uniquely identified by its VIA ID, the name and surname and its date of birth. While a professor can teach several students within several courses, every student can be taught by several professors by attending several courses. Every course is taught by one professor and a professor is in charge for one course only. A course is always assigned to one classroom, every classroom has one assigned course. To be able to schedule the different courses for the classrooms available, the schedule manager can view all classrooms, the classrooms it selves are only visible to a single schedule manager.



3 Design

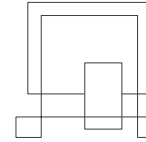
Design is an important part of creating a product, not only from the customer's point of view. Especially if it comes to aspects such as marketing, it is essential to have a design that meets all customer's expectation. In the context of application development, the term design distinguishes from the common sense of "design", focusing on more than the colour of the interface. Within application development it is crucial to create a visual and functional consistency. Next to the creation of appealing design elements like buttons, labels and colours (visual consistency), the application should function similarly through all elements (functional consistency) and should offer an logical workflow to allow the end user to "understand" the application easily (see NFR #1, chapter 2.3)

3.1 Design Tools

In order to develop a design prototype, the project team has found many possibilities and websites with which this is possible (see Table 3.1). However, there was already some experience with the online app design option of *MarvelApp*. On the one hand, it was possible for three people to work together in a team for free, on the other hand, time could be saved due to the experience already gained.

	 proto.io	 MarvelApp	 Adobe Xd
Web-based			
Android app			
Online			
Work as a team			
Price	free for 1 month	free of charge as long as it is only one project	free for 1 month
Experience			
Features	<ul style="list-style-type: none"> • Mobility • Wireframe • Prototyping 	<ul style="list-style-type: none"> • App Design • Wireframe • Prototyping 	<ul style="list-style-type: none"> • Animation • Wireframe • Prototyping • UX

Table 3.1: prototype development platform comparison



For the coding software, the choice of tools was limited to two. Then these two choices were compared with the help of an analysis (see Table 3.2). The Android Studio tool offers some advantages for prototype programming compared to Visual Studio. For this reason, Android Studio was chosen (see also chapter 4).













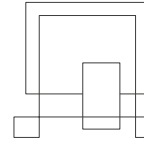
	 Android Studio	 Visual Studio
Quality of Support		
Android app		
Online		
Comes with emulator		
Price	free trial	free trial
Experience		
Pros and Cons	<ul style="list-style-type: none"> + Official from google + Easy to use - Huge memory usage - slow emulator 	<ul style="list-style-type: none"> + Many Plugins + Complete ide and debugger - Only available on Windows - Crashes more often

Table 3.2: coding software comparison

3.2 Design Principles

Design is an important part of the development of an app. In order to be successful on the market, it is important that the app reached a certain number of end users. To ensure this, the design must be both eye-catching and intuitive to use. Only satisfied and reoccurring customers (or this case end users) will allow a penetration of the target market.



For the design of applications, it is important to understand that design goes beyond the simple color-coding of elements. If the target end user is not able to use the application in an appropriate way, the outer design becomes irrelevant and the application will fail on the market. Useability is the key and is defined as a capability to be used in an easy, efficient and satisfying way. An user interface (UI) design brings the concept of interaction and visual design together and connects it with the information architecture. (Usability.gov, n.d.)

The following describes the design aspects from the perspective of the design prototype created with the platform *MarvelApp* (see chapter 3.1). The final design of the application is shown within the description of the application workflow (visual and functional consistency, see Chapter 10 Appendices).

As the app was developed, the usability and design principles were always kept in mind to make the app as user-friendly and simple to use as possible. One of the most important things is to display buttons and generally everything that can be tapped and operated in a certain size. There is no use if there are 100 touch options on a page and the user keeps getting lost. *Microsoft* has a guideline for human interface for this that suggest a minimum touch target size of 34px/9mm. (Uxbert Labs, 2017)

That is why the prototype has a large button in the middle of the home screen with the main function (see Figure 3.1). This button has a different look and function depending on the end user. The size, however, remains the same in any case.

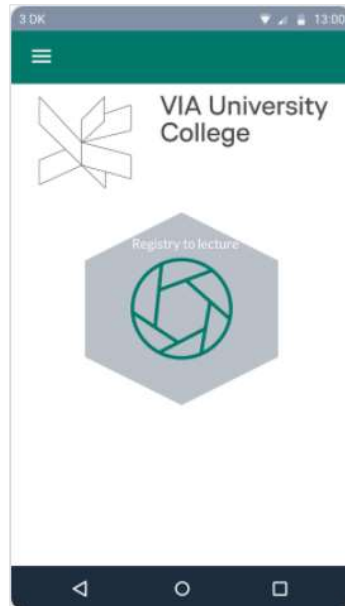
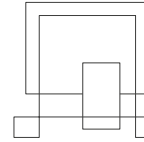


Figure 3.1: home screen - end user student (prototype)

Human attention spans are really short, eight seconds to be exact. This makes it extremely important to grab the users' attention within the first few seconds of interaction with a product. This is exactly why it is important to prioritise the content and then place it accordingly. Therefore, the main function for scanning the QR code is in the middle and the menu is at the top left.

Secondary functions, such as the menu that is displayed in the top left of the prototype with three parallel lines and can be unfolded with a touch, are significantly smaller. Nevertheless, it meets the size recommended by *Microsoft* to be easy to use.

Another good example of the implementation of this design rule is the professor's "current attendance" screen (see Figure 3.2). The most important function here, the current attendance of the students in percent, is exactly in the middle of the screen. Other data that is also interesting, but not as much as the current attendance, can be found below the graph.

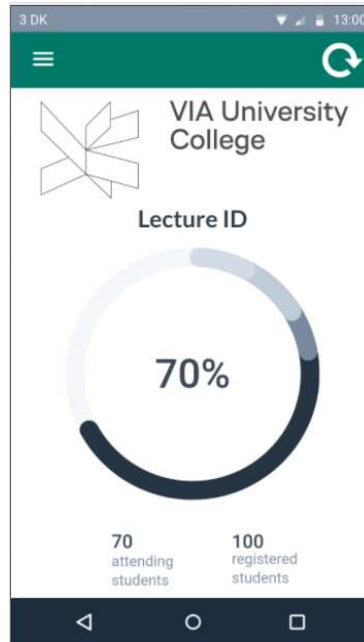
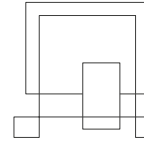


Figure 3.2: current attendance - end user professor (prototype)

This ensures that the end user is always shown the functions that are most important to him or her, prioritised.

Another important point is to keep the data input and time expenditure but also the sources of error low. There are many ways to do this when developing an app. For text input, the use of autocorrect or the preselection of options but also a dropdown menu are just some of these possibilities. Some of these options were used in the development of the app.

Among other things, a dropdown menu with suggested selection options was used for the "Latest Attendance" menu (see Figure 3.3). This helps the user to make a selection quickly and thus save time.

ENG-FPRPM-A21: Room utilization optimization via mobile application

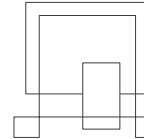


Figure 3.3: history data selection (prototype)

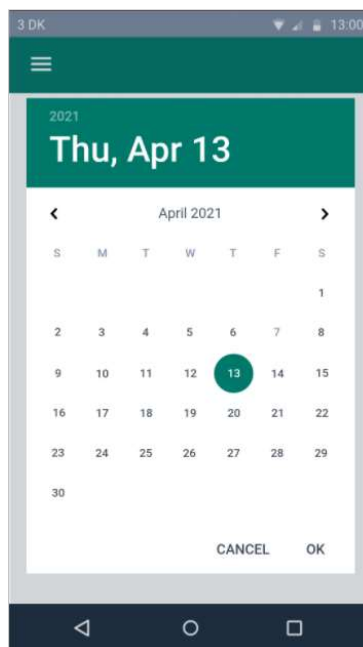
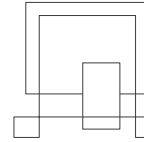


Figure 3.4: date selection - calendar function (prototype)



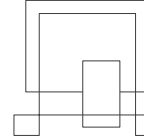
If the user now wants to select the date under the item “period”, he can simply select a date in the calendar instead of entering it manually (see Figure 3.4). This saves time, but also reduces the sources of error when entering the date manually.

3.3 User Interface Workflow

According to the defined use case diagram (see chapter 2.2.3) and use case descriptions (see chapter 2.2.4) application workflows for each type of end user were created. The workflows contain the final design in terms of visibility and functionality.

Every type of end user needs to log into the application first and has its own credentials provided and checked by the external system NetScaler AAA. After the login (see Figure A. 1) the end user student lands on the “registry to lecture” screen immediately. With a press on the button the camera function opens (once allowed) and the student is able to scan the QR code. After the scan, the screen confirms the registry to the lecture (see Figure A. 2). In the menu (sidebar) the student also has the possibility to check his past registrations with a click on “history”. The screen offers to choose the lecture from a drop-down and to pick a start and end date for the requested period by calendar function. Once confirmed, the student sees the history with attended lecture IDs, rooms IDs and the date of attendance (see Figure A. 3).

For the end user professor the login procedure is the same. After the login, the screen immediately shows the current attendance rate of the lecture (graph and text). The rate can be refresh with a click on the button “check real attendance” (see Figure A. 4) Within the menu, the professor also has the possibility to check the history data of the lecture attendance. As only one lecture is available, the professor can only select the dates for the requested period. After confirmation, the past attendance rate is shown as average (graph and text), as well as the deviation (percentage) between the single attendance rates within the selected period (see Figure A. 5).



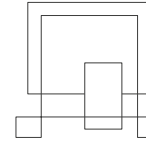
Similar to the previous end user types, the schedule manager also logs into the application with her credentials. After login, the screen shows the classroom ID, which can be selected with a drop-down, its average occupation rate, its available number of seats and the deviation between all past data (see Figure A. 6). Like the student, the schedule manger can review the past of classroom occupation with a selection of the classroom ID from a drop-down and the start and end date of the requested period. The output in the screen is similar to the screen after login (see Figure A. 7).

4 Implementation

For the implementation of the application, the Android Studio IDE (Integrated Development Environment) has been used. This is because VIA University College of Horsens offers a course called Android Development (IT-AND1) which is taught with the Android Studio development tool. This has made it easier to resolve doubts during the implementation of the application either through professors or colleagues from other lectures. In addition, this tool has also been chosen because it offers easy access, configuration and synchronization to a Firebase database which is essential for the application to function properly.

The database used in the project is on the Firebase platform due to the following advantages:

- Easily sync your project data without having to manage connections or write complex sync logic.
- Use a set of cross-platform tools: it is easily integrated for web platforms as well as mobile applications. It is compatible with large platforms, such as IOS, Android, web applications, Unity and C ++.
- Use Google's infrastructure and scale automatically for any type of application, from the smallest to the most powerful.
- Create projects without the need for a server: the tools are included in the SDKs for mobile and web devices, so there is no need to create a server for the project.



To create the application within the Android Studio tool, different tools have been used, such as different activities, fragments, spinners, layouts, etc.

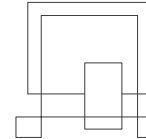
4.1 Dependencies

For the implementation of the application, it has been essential to make use of different external libraries (dependencies in Android) that help and facilitate the implementation of classes, interfaces, methods, databases, etc.

In Android Studio, dependencies allow to include external library or local jar files or other library modules in the Android project. External libraries are pieces of pre-written code that is easy to implement. It really has not to know exactly anything about what is inside that code, just need to know how to use it. The developers who make these libraries are going to tell exactly how to implement it through documentation.

The following dependencies have been included in the project (see Table 4.1):

```
dependencies {
1
    //Navigation drawer
2    implementation androidx.appcompat:appcompat:1.4.0
3    implementation com.google.android.material:material:1.4.0
4
5    //ConstraintLayout
6    implementation
7    androidx.constraintlayout:constraintlayout:2.1.2
8
9    //Lifecycle
10   implementation androidx.lifecycle:lifecycle-livedata-
11   ktx:2.4.0
12   implementation androidx.lifecycle:lifecycle-viewmodel-
13   ktx:2.4.0
14
15   //Navigation
16   implementation androidx.navigation:navigation-
17   fragment:2.3.5
18   implementation androidx.navigation:navigation-ui:2.3.5
19
2021  //Firebase
}
```



```

implementation platform('com.google.firebase:firebase-
bom:29.0.0')
implementation 'com.google.firebase:firebase-database:20.0.2'
implementation 'com.google.firebase:firebase-auth:21.0.1'
implementation 'androidx.legacy:legacy-support-v4:1.0.0'

//JUnit
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'

//QR Code + Camera
implementation 'me.dm7.barcodescanner:zxing:1.9.13'
implementation 'com.karumi:dexter:6.2.3'

//PieChart
implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
}

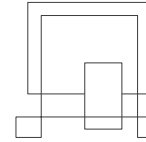
```

Table 4.1: dependencies

It was decided to implement these dependencies since without the use of these, developing the application would have been a more difficult and laborious task. In addition, thanks to the documentation provided by the developers and the large community that have already used them, implementing them has been relatively easy.

The use of these has provided simplicity, agility and speed when developing the application but it has been necessary to read the documentation to understand them correctly and how they work.

However, the most important and at the same time most difficult part of the application has consisted of reading a QR code with the mobile camera which consists of the name of the class and writing the student's attendance in the database. As well as then analyse and treat the acquired data and then show them in graphs for the professors and schedule manager.



The following will analyse the classes responsible for using the camera to scan the QR code and for sending and modifying information to and within the database (see 4.2) as well as the classes for checking the room fragment (see 4.3).

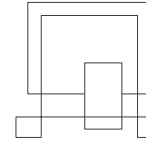
4.2 Class QrScannerActivity

This class is in charge of reading a QR code which will be composed of a String with the name of the classroom. In order to process this data, an interface has been implemented, which is included in the dependencies mentioned above, which allows reading QR and barcodes (see Table 4.2).

```

1  public class QrScannerActivity extends AppCompatActivity
2  implements ZXingScannerView.ResultHandler {
3
4      private ZXingScannerView scannerView;
5      private DatabaseReference databaseReference;
6      private Map<String, String> map1;
7      private Map<String, String> map2;
8      private Map<String, String> map3;
9      private Map<String, String> map4;
10     private Map<String, String> finalmap;
11     private ArrayList<String> arrayList;
12     private String currentStudent;
13     private static final String COURSE = "course";
14     private static final String ROOM = "room";
15     private static final String DATE = "date";
16     private static final String HOUR = "hour";
17     public boolean flag = false;
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         this.currentStudent = CurrentStudent.getCurrentViaID();
23         this.map1 = new HashMap<String, String>();
24         this.map2 = new HashMap<String, String>();
25         this.map3 = new HashMap<String, String>();
26         this.map4 = new HashMap<String, String>();
27         this.finalmap = new HashMap<String, String>();
28         this.arrayList = new ArrayList<String>();
29         this.scannerView = new ZXingScannerView(this);
30         this.databaseReference = FirebaseDatabase.
31             getInstance("https://eng-fprpm-a21-82b48-default-rtdb.europe-
32             west1.firebaseio.com").getReference();

```



```

33         setContentView(scannerView);
34
Dexter.withContext(getApplicationContext()).withPermission(
Manifest.permission.CAMERA).withListener(new PermissionListener()
{
    @Override
    public void
onPermissionGranted(PermissionGrantedResponse
permissionGrantedResponse) {
        scannerView.startCamera();
    }

    @Override
    public void
onPermissionDenied(PermissionDeniedResponse
permissionDeniedResponse) {
        Intent intent = new
Intent(QrScannerActivity.this, LoginActivity.class);
        startActivity(intent);
        Toast toast =
Toast.makeText(QrScannerActivity.this, "Please grant permission
of your camera to use the app", Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    }

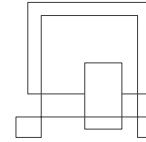
    @Override
    public void
onPermissionRationaleShouldBeShown(PermissionRequest
permissionRequest, PermissionToken permissionToken) {
        permissionToken.continuePermissionRequest();
    }
}).check();
}

```

Table 4.2: class QrScannerActivity

The class can be described as followed:

- First, the variables that will be used later are declared. It is worth highlighting the use of different Maps that will later allow to store the data received from the scanned code as well as the declaration of the object in charge of connecting to the database and the object in charge of activating the device's camera.



- Second, it can see the onCreate method where the variables declared above will be initialized. In addition, the Dexter class is called (implemented in the dependencies) which will check if the user has allowed access to the use of the camera of his device to be able or not to read the QR code.
- Finally the handleResult method will be overwritten, which handles the result obtained and where all the declared variables will be used.

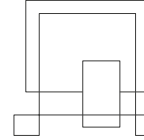
```

1      @Override
2      public void handleResult(Result rawResult) {
3          String nameRoom = rawResult.getText();
4
5          databaseReference.child("classroom").child(nameRoom).addListenerF
6          orSingleValueEvent(new ValueEventListener() {
7              @Override
8              public void onDataChange(@NonNull DataSnapshot
9              snapshot) {
10                 //for(DataSnapshot item: snapshot.getChildren()){
11                 //String classroomID =
12                 item.child("classroomID").getValue().toString();
13                 if (snapshot.exists()) {
14                     flag = true;
15                     SimpleDateFormat dateFormat = new
16                     SimpleDateFormat("dd/MM/yyyy");
17                     Date date = new Date();
18                     String finalDate = dateFormat.format(date);
19                     SimpleDateFormat timeFormat = new
20                     SimpleDateFormat("HH:mm:ss");
21                     Date time = new Date();
22                     String finalHour = timeFormat.format(time);
23                     map1.put(ROOM, nameRoom);
24                     map2.put(DATE, finalDate);
25                     map3.put(HOUR, finalHour);

```

Table 4.3: database information handling

The above (see Table 4.3) show the longest and most complex method in the QrScannerActivity class. Due to its long size, only a fragment is shown. This method receives a String which will be the class code and later the date and time are calculated to add it to a Map. Next, the database will be accessed to see the courses in which the student is registered and one of them will be chosen randomly to add it to their history.



Finally, if it has been successfully registered, a correct message will be displayed next to the name of the course, otherwise an error message will be displayed. Additionally, two functions have been created that access the database and increase the number of students attending in the classroom and in the course.

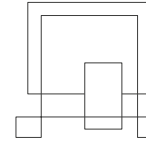
The main reason why this has been the most important challenge when developing the application is because the developer had never before implemented a feature of reading a QR code with the mobile.

Because of this, the fear was to not being able to do it in a functional way. Furthermore, even with the correct implementation of this function, the next challenge has consisted of sending the information received to the database, storing it and later extracting it to manipulate it and display it in the application.

To solve this challenge, use of the dependency that allows reading the camera's QR code was used. This has enabled the smartphone camera to be activated and read the QR code. In addition, to store the data extracted from the camera, a real-time database of Firebase has been used allowing the attendance of the students to be stored.

The most important part has been how to treat the data received in the application. This has been possible thanks to the use of different hash table (HashMap in Java). A hash table is a data structure that implements an associative array abstract data type, a structure that can map keys to values. This data structure has made possible to manage the received data and then send it to the database in the way desired by the development team.

So, the use of the real-time database, hashmaps, and the dependency has made it possible to solve the problem of reading QR codes.



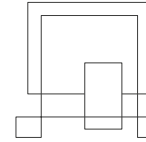
4.3 Class CheckRoomFragment

To display the pie chart in the application, an external dependency called MPAndroidChart has been used, which is responsible for generating different types of graphics and also which allows modifying the implemented graphics (see Table 4.4).

```

1      @Override
2      public View onCreateView(@NonNull LayoutInflater inflater,
3      ViewGroup container, Bundle savedInstanceState) {
4          binding =
5      SmFragmentClassroomAttendanceBinding.inflate(inflater, container, false);
6
7          View root = binding.getRoot();
8          this.pieChart = root.findViewById(R.id.sm_piechart);
9          this.spinner = root.findViewById(R.id.sm_spinnercourse);
10         this.arrayList = new ArrayList<>();
11         this.dataBase = FirebaseDatabase.getInstance("https://eng-
12 fprpm-a21-82b48-default-rtdb.europe-
13 west1.firebaseioapp/").getReference();
14         showDataSpinner(root);
15         binding.smActionShowGraph.setOnClickListener(new
16 View.OnClickListener() {
17     @Override
18     public void onClick(View v) {
19         String item = spinner.getSelectedItem().toString();
20         loadPieChartData(item);
21     }
22 });
23
24     return root;
25 }
26
27     private void loadPieChartData(String item) {
28
29     dataBase.child("classroom").child(item).addListenerForSingleValueEvent(new ValueEventListener() {
30     @Override
31     public void onDataChange(@NonNull DataSnapshot
32 snapshot) {
33         String activeUsers =
34 snapshot.child("activeUsers").getValue(String.class);
35         String enrolledStudents =
36 snapshot.child("capacity").getValue(String.class);
37         float occupiedSeats =
38 Float.parseFloat(activeUsers);
39
40

```



```
41         float availableSeats =  
        Float.parseFloat(enrolledStudents);
```

Table 4.4: class CheckRoomFragment

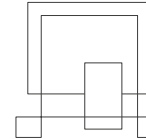
As it can be seen, a function called loadPieChartData has been created which is in charge of generating the pie chart by calling the methods and objects of the dependency installed in the project. This method will create a graph based on the parameter passed to it based on the object (in this case roomID) that the schedule manager has selected. Finally, in order to show the legend of the graph, it has been necessary to create an extra function called setLegend which adds the data to be shown in the pie graph.

In order to implement this, it has been necessary to read the official documentation on GitHub. Implementing the pie chart has not been complicated. The most difficult thing has been to understand how the library worked and what methods had to be modified or created to be able to customize the graphics to the desired design.

5 Software Testing

Software testing is a very important phase in the software development life cycle (SDLC) and is essential to ensure the quality of the developed software. Software testing itself is defined as a process of evaluating the capability of a program or system to determine it meets the required outcome, or rather to execute a system with the intent of finding expected and as-yet undiscovered errors. (Mathur & Malik, 2010)

Early testing is the key and testing activities should start as soon as requirements and design documents are available. The earlier the testing begins, the earlier defects can be found and fixed as it is much cheaper to change an incorrect requirement than to change a coded system which is not working as requested.



For this project, the V-Model was used. The V-Model is a widely used, generic and comprehensive testing model also referred as the Verification and Validation model. It contains the verification phases of the developers life cycles following the sequential process of the waterfall model (left side) as well as the validation phases representing the tester's life cycle (right side). The coding phase joins both sides leading to the V-shape of the model and demonstrating the relationship between the associated phases from both cycles. (Java T Point, n.d.)

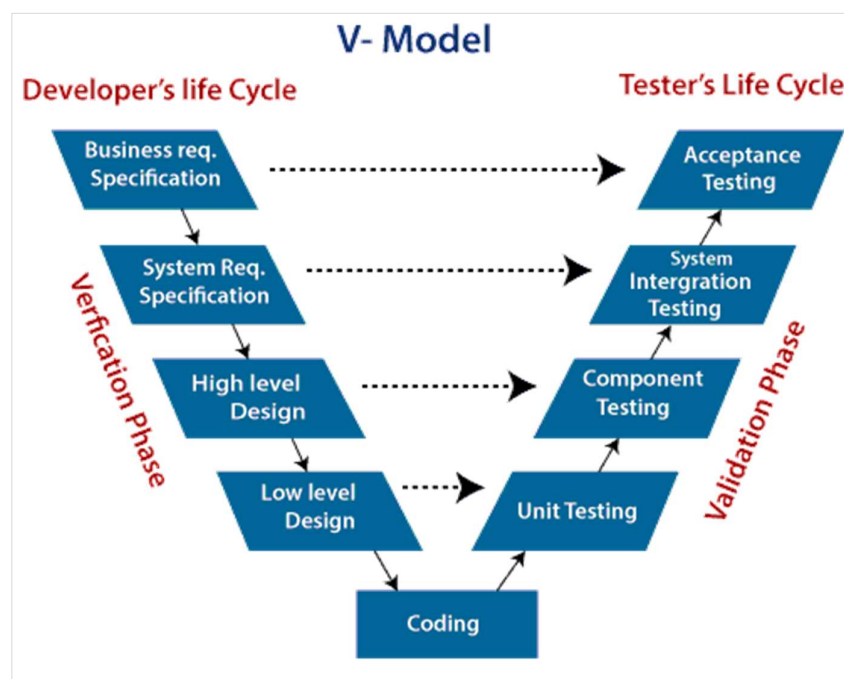
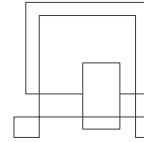


Figure 5.1: V-Model software testing²

The developers cycle is divided into various phases of understanding the users requirements (business requirement specification), perform an analysis for those requirements (system requirement specification), design an initial outline of the software (high level design) followed by the design of advanced details (low level design) and the coding.

² (Java T Point, n.d.)



The tester's cycle follows the IEEE standards³ of software testing categories and is divided into unit testing, component testing, integration testing and (user) acceptance testing. For unit testing, unit testing plans (UTPs) are developed and executed during the low level design phase to eliminate errors at code or unit level. UTPs ensure that the smallest entity of the system functions correctly when isolated from the rest of the code and are performed most of the times with white box testing techniques⁴.

During the high level design phase, integration testing plans (ITPs) are developed and executed using black box testing techniques⁵. ITPs addresses the assembling and integration of the software components being able to coexists and to communicate among themselves.

System testing plans (STPs) are developed during the system design phase and are executed to evaluate the systems compliance with analysed and specified requirements. STPs are composed by the business team and do not require any knowledge about the inner code design or logic.

To ensure the development system meets the end user's specified requirements, acceptance testing plans (ATPs) are developed during the business requirement analysis. ATPs are executed in end user atmosphere to discover potential incompatibilities within this atmosphere and to determine non-functional problems like performance issues.

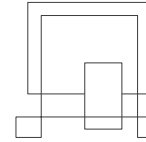
The IEEE standard 829⁶ for software and system test documentation specifies the form of documents for use in the different stages of testing. Given the scope and size of the project, only LTD, LTC, LTP for testing as well as LTR were considered.

³ *IEEE Standard for System and Software Verification and Validation (IEEE Computer Society, 2012)*

⁴ *software testing method carried out by developers and used to test software with knowledge about inner coding structure and/or logic (Software Testing Class, 2014)*

⁵ *software testing method carried out by testers (customers, users) and used to test software without knowing the inner coding structure or logic (Software Testing Class, 2014)*

⁶ 829, Appendix B



While test cases (see chapter 2.4) are executed to verify a particular software feature or functionality, test scenarios are a collective set of test cases giving an high-ideal of what need to be tested. In the following (see chapter 5.2.1 to 5.2.6), all test scenarios developed for this project are listed with their corresponding TCSs. The TCSs only represent the expected results to consider the test case as “passed”. If one of the steps does not provide the expected result in any way, the test is considered as “failed”. The TCSs also classify the integrity level and its consequence level of failures.

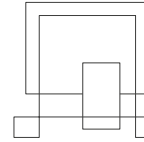
IEEE defines the integrity level as followed:

- **Level 1:** negligible consequences; software must execute correctly or intended function will not be realized, no mitigation required
- **Level 2:** marginal consequences; software must execute correctly or intended function will not be realized causing minor consequences, mitigation possible
- **Level 3:** critical consequences: software must execute correctly or intended function will not be realized causing serious consequences (permanent injury, major system degradation, environmental damage), partial-to-complete mitigation possible
- **Level 4:** catastrophic consequences: software must execute correctly or grave consequences (loss of life, loss of system, environmental damage) will occur, no mitigation possible

5.1 Testing tools

To test the application as it was being developed, the debugging and USB-Debugging tools were used. Debugging allows to go through each line of code, evaluating app's variables, methods and how well your code is working.

It should be noted that the Junit test could not be applied to test the classes and methods due to delays in the application delivery. JUnit is a unit testing framework for Java programming language.



JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

5.1.1 USB-Debugging

USB Debugging allows an Android device to communicate with a computer that's running the Android SDK in order to use advanced operations. This mode allows to root the phone or flash a serial image on the Android device from the computer. The debugging mode is a way for the computer to communicate with the phone via the ADB terminal (Android Debugging Bridge), and to be able to access the internal files of the terminal. With ADB, files and commands can be received from the PC, in such a way that the computer accesses any part of the terminal making it possible to test the application as if it were real.

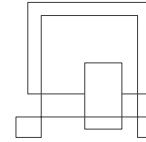
5.1.2 Debugging

Debugging is the process of finding and fixing errors (bugs) or unexpected behaviour in the code. All code has bugs, from incorrect behaviour in the app, to behaviour that excessively consumes memory or network resources, to actual app freezing or crashing.

Bugs can result for many reasons:

- Errors in your design or implementation
- Android framework limitations (or bugs)
- Missing requirements or assumptions for how the app should work
- Device limitations (or bugs)

Use the debugging, testing, and profiling capabilities in Android Studio help to reproduce, find, and resolve all of these problems. Those capabilities include:



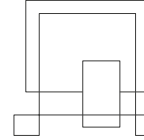
- The Logcat pane for log messages
- The Debugger pane for viewing frames, threads, and variables
- Debug mode for running apps with breakpoints
- Test frameworks such as JUnit or Espresso
- Dalvik Debug Monitor Server (DDMS), to track resource usage

5.2 Test Case Specifications

The project follows the approach of a proof-of-concept only and will be developed and tested with simulated data only. Furthermore, the main idea of the application to track the students' real-time attendance in lectures do not provide any physical risk to the end users or their environment. Therefore, all TCSs will be considered on an maximum integrity level of 2.

Every feature or system function, which only provides the basics of the application (login, QR code scanning, history data or system response times) is defined as integrity level 1. If these functions fail, of course, the application cannot be used for its initial purpose, but there are no consequences for the system or the end users.

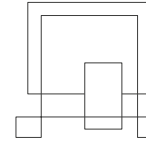
However, functions and features, which provide the main function of the application, to provide actual (or average) data about the attendance level of lectures, are defined as integrity level 2. A malfunction of those features would provide incorrect data and could result in further economic consequences. Same goes with the malfunction of end user profiles and the risk of getting access to restricted data (e.g., student see's professors profile).



5.2.1 Test scenario 1: login page validation

TCS 1		
Test stage:	STP	
Test case:	Enter valid username and password	
Tester:	End user	
Prerequisites:	Access to application	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Open application	Login screen should be visible
2	Enter ID and password	Credentials can be entered
3	Click LOGIN button	End user should be login

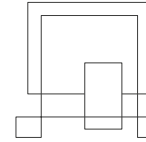
TCS 2		
Test stage:	STP	
Test Case:	Enter invalid username and password	
Tester:	End user	
Prerequisites:	Access to application	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Open application	Login screen should be visible
2	Enter invalid ID or wrong password	Credentials can be entered
3	Click LOGIN button	End user receive error notification about invalid credentials
		End user is transferred back to login screen



5.2.2 Test scenario 2: end user profile validation

TCS 3		
Test stage:	ATP	
Test Case:	Enter student ID	
Tester:	End user student	
Prerequisites:	Access to login screen	
Integrity level:	2; marginal	
#	Step details	Expected result
1	Enter student ID (six figures) with corresponding password	Credentials can be entered
2	Click LOGIN button	End user should see home screen with "registry to lecture" option
3	Access sidebar (menu)	End user should see ID and profile "student"
		End user should see menu with "registry to lecture", "latest attendance" and "history"

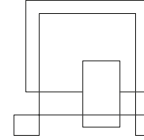
TCS 4		
Test stage:	ATP	
Test Case:	Enter professor credentials	
Tester:	End user professor	
Prerequisites:	Access to login screen	
Integrity level:	2; marginal	
#	Step details	Expected result
1	Enter professor credentials (official name abbreviation) and corresponding password	Credentials can be entered
2	Click LOGIN button	End user should see home screen with attendance data of his lecture
3	Access sidebar (menu)	End user should see credentials and profile "professor"
4		End user should see menu with "check current attendance" and "history"



TCS 5		
Test stage:	ATP	
Test Case:	Enter schedule manager credentials	
Tester:	End user schedule manager	
Prerequisites:	Access to login screen	
Integrity level:	2; marginal	
#	Step details	Expected result
1	Enter schedule manager credentials (official name abbreviation) and corresponding password	Credentials can be entered
2	Click LOGIN button	End user should see home screen with occupation data of classroom
3	Access sidebar (menu)	End user should see credentials and profile "schedule manager"
4		End user should see menu with "classroom attendance" and "history"

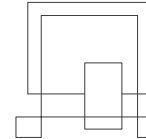
5.2.3 Test scenario 3: lecture attendance registry validation

TCS 6		
Test stage:	STP	
Test Case:	Scanning of valid QR code	
Tester:	End user student	
Prerequisites:	Successful login as student	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Navigate to home screen	"scan qr code" option should be visible
2	Click REGISTRY TO LECTURE button	Camera function should open to scan QR code
3	Scan valid QR code	Scan successful
		End user should see attendance details (lecture ID, classroom ID, date stamp) on screen



TCS 7		
Test stage:	STP	
Test Case:	Scanning of invalid QR code	
Tester:	End user	
Prerequisites:	Successful login as student	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Navigate to home screen	"scan qr code" option should be visible
2	Click SCAN QR CODE button	Camera function should open to scan QR code
3	Scan invalid QR code	End user should see error notification about invalid QR code
		End user should see home screen with scanning option again

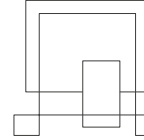
TCS 8		
Test stage:	STP	
Test Case:	Scanning QR code without camera permission granted	
Tester:	End user	
Prerequisites:	Successful login as student	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Navigate to home screen	"scan qr code" option should be visible
2	Click SCAN QR CODE button	Camera function should open to scan QR code
3	End user deny camera permission	End user should see error notification about permission camera not granted
		End user will come back to login page to accept or not the permission of using camera.



5.2.4 Test scenario 4: wireless communication validation

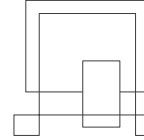
TCS 9		
Test stage:	STP	
Test Case:	Data transmission with wireless connection	
Tester:	End user	
Prerequisites:	Successful login as student, internet access	
Integrity level:	2; marginal	
#	Step details	Expected result
1	Navigate to home screen	"registry to lecture" option should be visible
2	Click SCAN QR CODE button	Camera function should open to scan QR code
3	Scan valid QR code	Scanning successful
4	Navigate to history section	End user should see screen for history selection
5	Select today's date and the lecture from previous scan	Lecture ID should be available in dropdown, period selection should be available
6	Click OK button	End user should see at least previous scan of lecture (full history possible)

TCS 10		
Test stage:	STP	
Test Case:	No data transmission without wireless connection	
Tester:	End user	
Prerequisites:	Successful login as student, disconnected internet access	
Integrity level:	2; marginal	
#	Step details	Expected result
1	Navigate to home screen	"registry to lecture" option should be visible
2	Click SCAN QR CODE button	Camera function should open to scan QR code
3	Scan valid QR code	End user should see error notification about failed database connection (no translation of QR code into lecture ID)
4	Navigate to history section	End user should see screen for history selection
5	Select today's date and the lecture from previous scan	Lecture ID should be available in dropdown, period selection should be available
6	Click OK button	End user should see error notification about failed database connection

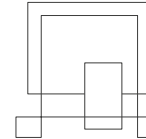


5.2.5 Test scenario 5: history option validation

TCS 11		
Test stage:	ATP	
Test Case:	History data selection as student	
Tester:	End user student	
Prerequisites:	Successful login as student	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Navigate to history section	End user should see screen for history selection
2	Select lecture ID in dropdown	End user should see dropdown menu with previous attended lectures
3	Click into line of start date (FROM)	End user should see calendar function to select start date
4	Select start date	End user should see selected date in line of FROM
5	Click into line of end date (TO)	End user should see calendar function to select end date
6	Select end date	End user should see selected date in line of TO
7	Click SUBMIT button	End user should see selected lecture ID as headline
		End user should see complete history of selected data with lecture IDs, date and time
		End user should see attendance in descending order (latest attendance first)



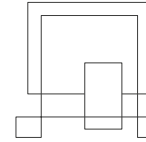
TCS 12		
Test stage:	ATP	
Test Case:	History data selection as professor	
Tester:	End user professor	
Prerequisites:	Successful login as professor	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Navigate to history section	End user should see screen for history selection
2	Click into line of start date (FROM)	End user should calendar function to select start date
3	Select start date	End user should see selected date in line of FROM
4	Click into line of end date (TO)	End user should calendar function to select end date
5	Select end date	End user should see selected date in line of TO
6	Click SUBMIT button	End user should see selected lecture ID as headline
		End user should see average attendance of students (% as text and as percentage bar)
		End user should see average number of attending students below graphic
		End user should see deviation of attendance (%) below graphic



TCS 13		
Test stage:	ATP	
Test Case:	History data selection as schedule manager	
Tester:	End user schedule manager	
Prerequisites:	Successful login as schedule manager	
Integrity level:	2; negligible	
#	Step details	Expected result
1	Navigate to history section	End user should see screen for history selection
2	Select classroom ID in dropdown	End user should see dropdown menu with all available classroom IDs
3	Click into line of start date (FROM)	End user should calendar function to select start date
4	Select start date	End user should see selected date in line of FROM
5	Click into line of end date (TO)	End user should calendar function to select end date
6	Select end date	End user should see selected date in line of TO
7	Click SUBMIT button	End user should see selected classroom ID as headline
		End user should see average occupation (%) of selected classroom (as text and as percentage bar)
		End user should see average number of occupied seats below graphic (text)
		End user should see number of available classroom capacity below graphic (text)
		End user should see deviation of classroom occupation (%) below graphic (text)

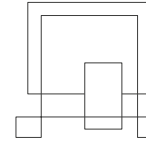
5.2.6 Test scenario 6: system performance validation

TCS 14		
Test stage:	ITP	
Test Case:	Response time after scan of QR code	
Tester:	developer	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Scan QR code	Translation of QR into lecture ID < 2s



TCS 15		
Test stage:	ITP	
Test Case:	Response time after refresh of student counter	
Tester:	developer	
Integrity level:	1; negligible	
#	Step details	Expected result
1	Press refresh button in professor's profile	Update of number of attending students < 1s

TCS 16		
Test stage:	ITP	
Test Case:	Calculation of average	
Tester:	developer	
Integrity level:	2; marginal	
#	Step details	Expected result
1	Set number of assigned students to lecture XY to seven	Database show lecture ID with seven assigned students
2	Simulate five successful attendance registration by five different students (same classroom ID)	Database show lecture ID and time stamp in five different student profiles
3	Login as professor	
4	Navigate to "check current attendance" and select lecture and today's date	
5	Confirm selection	Screen should show average of attending students (71%), seven assigned students and 0% deviation
6	Repeat step 2 with two different dates, one date with three students, other date with two students	Database show lecture ID and three different time stamp in corresponding student profiles
7	Repeat step 3-5	Screen should show average of attending students (48%), ten assigned students and 22% deviation
8	Set capacity of classroom ID to ten	Database should show classroom ID with a capacity of ten (seats)
8	Login as schedule manager	
9	Select corresponding classroom ID and simulated period	Screen should show average of 33% occupation, a capacity of ten and a deviation of 15%



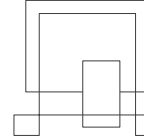
According to the corresponding NFR (#5; see Table , chapter 2.3) the calculation of the average and the deviation within the history report needs to be correct. Within this project, the deviation data will be simulated as the calculation is not planned to be implemented to this point of time.

TCS 17		
Test stage:	ITP	
Test Case:	Simultaneous logins	
Tester:	End user student; developer	
Prerequisites:	Successful login as student	
Integrity level:	2; marginal	
#	Step details	Expected result
	Number of end users testing at the same time: 5	
1	Navigate to home screen	“registry to lecture” option should be visible
2	Click SCAN QR CODE button	Camera function should open to scan QR code
3	Scan valid QR code	Scanning successful
4		End user should see attendance details (lecture ID, classroom ID, date) on screen
5	Developer: check database	Database should show all IDs with lecture ID and date and time stamp

As the database for the application system contains simulated data only, there will be no possibility to test simultaneous logins of end users in real environment. Therefore, this test case (#17) will be excluded from the test execution section (see chapter 5.3).

5.3 Results

The following tables give an overview about the executed tests, the parameters used (if available) for testing and the test results. According to the developed TCSs and their corresponding FR/NFR some tests were executed several time, some only one time (see number of test runs).

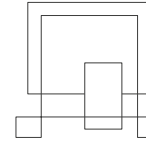


A test run “pass” it, if all steps were completed and the expected results were achieved. In case of failure, the incidents are logged (ascending numbering) and described (see chapter 5.4)

The executed tests show that the main aspects of the application are implemented successfully and allow the different end users to work within their profiles. The basic function like login (see chapter 5.2.1 and chapter 5.2.2) and scanning of the QR code (see chapter 5.2.3) are tested multiple times and always passed for valid parameters. On the other hand, the function failed in terms of invalid parameters (invalid QR code, invalid credentials), as expected.

Some of the requirements could not be realized completely, leaving out some of the expected and requested details within the output (see chapter 5.2.5) for students. However, as the main information about the attended lecture, the ID of the classroom and the date of the registry is logged and presented, the missing headline and time stamp are considered as negligible as this request only minor adjustments in the coding.

As there were a simulated database only, some of the performance requirements could not be tested in a real environment, e.g. simultaneous logins, calculation of average and deviation (see chapter 5.2.6). The integrity of the calculation is a major aspect in the functionality of the application and would be crucial for the realization of it. For the prototype and the proof-of-concept approach in this project, the real-time calculation of average and deviation were neglected.



5.3.1 Test results scenario 1

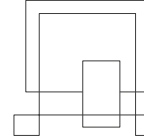
TCS 1		
Description:	Enter valid username and password	
Requirement:	FR #3	
Test run:	Result:	Parameters:
TR-001	pass	ID: 313313, pw: jordilazo
TR-002	pass	ID: 311111, pw: jordi
TR-003	pass	ID: TRO, pw: tars
TR-004	pass	ID: GIBA, pw: queen

TCS 2		
Description:	Enter invalid username and password	
Requirement:	FR #3	
Test run:	Result:	Parameters:
TR-005	pass	ID: 313313, pw: 123
TR-006	pass	ID: 123, pw: tars
TR-007	pass	ID: TRO, pw: 123

5.3.2 Test results scenario 2

TCS 3		
Description:	Enter student ID	
Requirement:	FR #5	
Test run:	Result:	Parameters:
TR-008	pass	ID: 313313, pw: jordilazo
TR-009	pass	ID: 311111, pw: jordi

TCS 4		
Description:	Enter professor credentials	
Requirement:	FR #5	
Test run:	Result:	Parameters:
TR-010	pass	ID: TRO, pw: tars
TR-011	pass	ID: LEOS, pw: tars



TCS 5		
Description:	Enter schedule manager credentials	
Requirement:	FR #5	
Test run:	Result:	Parameters:
TR-012	pass	ID: GIBA, pw: queen
TR-013	pass	ID: GITA, pw: queen2

5.3.3 Test results scenario 3

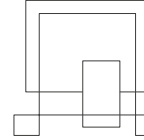
TCS 6		
Description:	Scanning of valid QR code	
Requirement:	FR #1	
Test run:	Result:	Parameters:
TR-014	pass	valid QR code
TR-015	pass	valid QR code

TCS 7		
Description:	Scanning of invalid QR code	
Requirement:	FR #1	
Test run:	Result:	Parameters:
TR-016	pass	invalid QR code
TR-017	pass	invalid QR code

TCS 8		
Description:	Scanning QR code without camera permission granted	
Requirement:	FR #1	
Test run:	Result:	Parameters:
TR-016	pass	camera permission not granted
TR-017	pass	camera permission not granted

5.3.4 Test results scenario 4

TCS 9		
Description:	Data transmission with internet connection	
Requirement:	FR #2	
Test run:	Result:	Parameters:
TR-018	pass	scan QR code, internet connection
TR-019	pass	scan QR code, internet connection



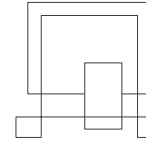
TCS 10		
Description:	No data transmission without internet connection	
Requirement:	FR #4, NFR #4	
Test run:	Result:	Parameters:
TR-020	failed	scan QR code, without wireless connection
TR-021	failed	scan QR code, without wireless connection

5.3.5 Test results scenario 5

TCS 11		
Description:	History data selection as student	
Requirement:	FR #6	
Test run:	Result:	Parameters:
TR-022	failed	ID: 313313, pw: jordilazo
TR-023	failed	ID: 311111, pw: jordi

TCS 12		
Description:	History data selection as professor	
Requirement:	FR #6	
Test run:	Result:	Parameters:
TR-024	pass	ID: TRO, pw: tars
TR-025	pass	ID: LEOS, pw: tars

TCS 13		
Description:	History data selection as schedule manager	
Requirement:	FR #6	
Test run:	Result:	Parameters:
TR-026	pass	ID: GIBA, pw: queen
TR-027	pass	ID: GITA, pw: queen2



5.3.6 Test results scenario 6

TCS 14		
Description:	Response time after scan of QR code	
Requirement:	NFR #2	
Test run:	Result:	Parameters:
TR-028	1,7 sec, pass	ID: 313313, pw: jordilazo
TR-029	1,9 sec, pass	ID: 311111, pw: jordi

TCS 15		
Description:	Response time after refresh of student counter	
Requirement:	NFR #6	
Test run:	Result:	Parameters:
TR-030	1 sec, pass	ID: TRO, pw: tars
TR-031	1 sec, pass	ID: LEOS, pw: tars

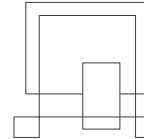
TCS 16		
Description:	Calculation of average	
Requirement:	NFR #5	
Test run:	Result:	Parameters:
TR-032	failed	
TR-033	failed	

5.4 Incident report

The following (see Table 5.1) gives an overview about unexpected issues during the testing.

TEST RUN	INCIDENT	DESCRIPTION
TR-020	ISS-01	Endless scanning of QR code, no error message
TR-021	ISS-02	Endless scanning of QR code, no error message
TR-022	ISS-03	Can't see selected lecture as headline, can't see time
TR-023	ISS-04	Can't see selected lecture as headline, can't see time
TR-032	ISS-05	Simulated data only, average not calculated
TR-033	ISS-06	Simulated data only, average not calculated

Table 5.1: overview of incidents during testing



The first two incidents (ISS-01, ISS-02) describe the missing error notification (step 3) in case there is no internet connection and therefore, no connection to the database. By now, the application simply goes back to the “registry to lecture” option. The next two incidents (ISS-03, ISS-04) refer to missing details in the output of the history data for the end user student. The selected lecture is not shown as headline (step 7) and there is no time stamp, date information only (step 7). The last two incidents (ISS-05, ISS-06) are the result of the database being simulated. The average shown in the output is not calculated, it is a random number.

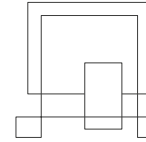
6 Results and Discussion

The following refers to the outcome of the app developing process in regard to the previous defined problem statement (see Project Description, chapter 2) and its corresponding sub-problems (see Project Description, chapter 2.1).

The project follows the approach to optimize the classroom utilization by collecting data about the real-time attendance of students in the lecture. With the developed prototype it is shown, that the real-time registration (data quality) of students to a lecture by scanning a QR code with a mobile application is possible (type of technology). It is also shown that the collected data can be used to review the students attendance (end user profile student), to review the attendance rate of students (end user profile professor) and to get information about the occupation rate of the corresponding classroom (end user profile schedule manager) (end users).

The concept also shows that the application without the integration of the NetScaler AAA identification system, the integration of a connection to ItsLearning and the integration to the scheduling system would not be doable (key partners, external resources).

As the concept depends on the number of students using the application for lecture registry, the development process shows that the visual and functional design is the key to gain a certain level of market share and brand awareness (marketing, usability).



Only if students are inclined to use the application based on an intuitively understandable interface and a coherency within the system functions and features, the application is able to penetrate the market and gain its necessary huge customer base (the more students login, the more precise the database gets).

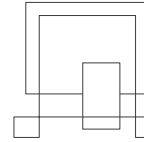
Due to the agility of the project, the expanded analysis phase, and the lack of time available for the coding and testing phase, the project were developed for and with simulated data only.

Furthermore, the idea of real-time connection to external resources like NetScaler AAA or ItsLearning were not feasible, so the database was created with fictitious data. Therefore, any regulation in regard to GDPR were not considered any further (data security).

To launch the application into a real environment the multi-threading model for simultaneous data transmission must be implemented and extended to realize the maximum of students' registrations at the same time (performance).

Also the correct and precise calculation of the average and deviation of the attendance would be crucial for the market chance of a real product. To optimize the end user's usability it could also be possible to allow the student to confirm or deny the registration to a lecture, in case the student is about to enter the wrong classroom. Another point would be the implementation of all lectures a professor can teach, so the history section would allow to choose from a dropdown and keep track of the attendance rate of all lectures. For the schedule manager, it could be essential to export the classroom occupation data, e.g. as a csv-file, to import the collected data directly into the scheduling tool (real product).

Furthermore, during the design phase the project team sat together with the schedule manager (Gitte Bernhard Andersen) to get more insights into the process of scheduling the classrooms and what kind of information the schedule manager would need as output (meeting minutes, see progress report).



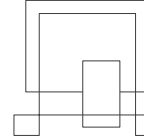
During this meeting it turned out, that the allocation of the lectures to the classrooms is based on the lecture and the number of registered students. The application concept follows the idea of presentation the occupation based on the chosen classroom ID. As the project was too far into the development process to change the requirements, the required adjustment of presenting the occupation based on the lecture ID were not taken into account. However, for a real product in the future, this would be a mandatory change for the presented output.

The project team organized itself and managed its timeline with a Gantt chart and scrum sprint reports in Microsoft Excel. With the expanded analysis phase and refinement of the FRs and NFRs, there were major shifts in the acquired milestone dates, so the phase for coding and testing must be shortened. However, the agreement to a sprint period of seven days and the division of the work in single packages allowed the project team to simultaneously work on the project and follow up on major and minor changes in the development process (time management).

7 Conclusions

The project shows the huge potential of an application concept, that collects data about the real and real-time attendance of students in lectures. With the data the students are allowed to track their own attendance in lectures and professors can review the attendance rate of their lectures. The main deliverable of an application following this concept would be the possibility for the schedule manager to allocate lecture to classrooms based on the number of attending students, not registered ones, which gives the opportunity to use room resources more efficiently and economically.

The development process was based on several business and IT related methods, which helped the project team to really understand the essence of the problem to be solved and how to approach its solution. The project outcome delivers a basic concept of how an application for this purpose could look like and gives a baseline for future improvements and expansions.



8 Project future

Since the entire project is a proof of concept of an information and communications technology project, the following section analyses the future aspects of the project idea, especially in terms of a product launch into a real market.

In order to evaluate and analyse all influences on the app and the market in which the app would be launched, the PESTEL method was used. With the help of the PESTEL method business influences are divided into six different categories with various factors assigned to the individual categories. This not only allow to identify and classify the impacting factors, it also allows to discover new opportunities.

For the PESTEL analysis a virtual board was created (see Figure 8.1) and based on the sub-problems evaluated during the pre-analysis of the project (see Project Description, chapter 2.1).

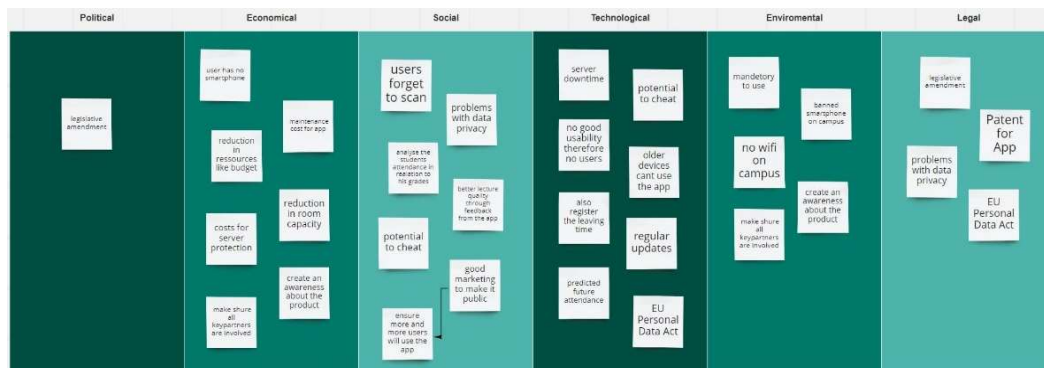
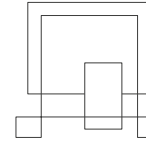


Figure 8.1: PESTEL analysis board



8.1 Political aspects

One of the most important political factors is the change in the legal situation. If the regulations for GDPR and therefore, the framework for data protection is changed, this could lead to a review and possible redesign of all processes that involve private data.

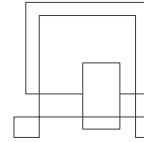
8.2 Economical aspects

Servers are needed to keep the app running. Regular updates must be performed to guarantee the security of user data, but also to ensure that usability is guaranteed for all users. In order to accomplish this work, professional employees are needed to take care of this task. These employees, but also the servers, which must also meet a certain standard, cost a lot of money. If the prices for the infrastructure, such as servers, increase, this can mean that the use of the app is no longer sustainable in the long term.

Besides the cost of the application, the economic potential of the application needs to be considered. A lecture allocation based on the attending students would allow to save costs for rooms as the deviation between attending and registered students could be huge. This could also be a considerable fact if it comes to budget planning and the division of the yearly budget across the different study programs. Programs with a low attendance rate, and therefore low demand of classroom resources could spare the budget for programs with a high number of attending students.

8.3 Social aspects

A good point to see is that the PESTEL analysis not only shows potential weaknesses or dangers, but also new opportunities, especially in the possibilities to expand the APP. One of the best ways is to use the system to relate student attendance to exam performance. In this way, students can see very clearly and meaningfully how attendance and participation in lectures can affect their examination performance.



On the other hand, professors get the opportunity to get idea about their lecture quality. If the major part of the registered students are not attending the lectures (low attendance rate) it could mean that the quality of the lecture could be improved. Furthermore, it could be an idea to deny students the access to exams in case they do not meet a certain level of attendance percentage. Especially, for students in the first semesters this would allow to applicate a more focused and educational improved way of studying.

Another point needed to take into account is the data accuracy. With an application based on the principle of scanning a QR code there is always way to “cheat”. Image the possibility of static classroom-QR code a simple photography of the code is enough to “attend the lecture” from home. For the future, the concept must be expanded to a more secure concept (e.g., dynamic QR codes).

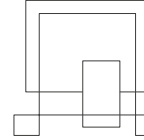
For the schedule manager it could also be considered to change the presentation of classroom occupation rates into lecture attendance rates (similar to the end user professor) to allow a faster and more precise allocation of lecture to the available resources.

8.4 Technological aspects

One of the most critical technological points is that old smartphone devices may not be able to use the app. This would mean that less data is available. To ensure that this does not happen, it must be ensured that all smartphone devices with the respective systems always have the corresponding updates of the app available. This means working closely with the partners who are connected to the app.

8.5 Environmental aspects

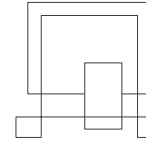
To ensure that the data exchanged with the server when scanning a QR code, for example, an internet connection is required.



This can be a connection via mobile data but also via Wi-Fi. The problem with mobile data is that if it is throttled, a data exchange takes a very long time and therefore in most cases this request is rejected by the server. To ensure that every user can still access the app's functions, it would be advisable to provide a Wi-Fi network for end users at the campuses where the app is used.

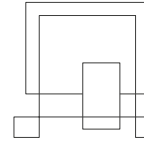
8.6 Legal aspects

The app was developed against the background of working with user data, evaluating it and then using it for planning. However, when working with such sensitive data and storing it for a long time, it is important to pay attention to the privacy of this data. The most important thing here is transparency for the end user, but also the protection of the data from third parties. It is important to ensure the highest possible protection of the end user's sensitive data.

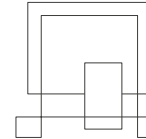


9 Source of information

- Alobaedy, M. M. & Ku-Mahamud, K. R., 2014. Scheduling jobs in computational grid using hybrid ACS and GA approach. *Computing Communications and IT Applications Conference (ComComAp)*, pp. 223-228.
- Altexsoft, 2021. *Functional and Nonfunctional Requirements: Specification and Types*. [Online]
Available at: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/>
[Accessed 04 Dec 2021].
- Banger, D., n.d.. *A Basic Non-Functional Requirements Checklist*. [Online]
Available at: <https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/>
[Accessed 09 Dec 2021].
- Chung, L., NixonEric, B. A. & Mylopoulos, Y., 2000. Non-Functional Requirements in Software Engineering. In: 5, ed. *International Series in Software Engineering*. Boston: Springer, pp. 153-160.
- Dabbagh, M. & Lee, S. P., 2014. An Approach for Integrating the Prioritization of Functional and Nonfunctional Requirements. *The Scientific World Journal*, Volume 2014, p. 13.
- IEEE Computer Society, 2018. *IEEE Recommended Practice for Software Requirements Specifications*. 2nd ed. New York: IEEE.
- IEEE Computer Society, 2012. *IEEE Standard for System and Software Verification and Validation*. New York: IEEE.
- Java T Point, n.d.. *V-Model*. [Online]
Available at: <https://www.javatpoint.com/software-engineering-v-model>
[Accessed 11 Dec 2021].
- Laplante, P. A., 2014. *Requirements Engineering for Software and Systems*. 2nd ed. Boca Raton: Taylor & Francis Group.
- Mathur, S. & Malik, S., 2010. Advancements in the V-Model. *International Journal of Computer Applications (0975 – 8887)*, 1(12), pp. 29-34.
- Scaled Agile, 2021. *Domain Modeling*. [Online]
Available at: <https://www.scaledagileframework.com/domain-modeling/>
[Accessed 09 Dec 2021].
- Scaled Agile, 2021. *Nonfunctional Requirements*. [Online]
Available at: <https://www.scaledagileframework.com/nonfunctional-requirements/>
[Accessed 11 Dec 2021].



- Software Testing Class, 2014. *Difference between Black Box Testing and White Box Testing*. [Online]
Available at: <https://www.softwaretestingclass.com/difference-between-black-box-testing-and-white-box-testing/>
[Accessed 11 Dec 2021].
- Statista, 2021. *Number of registered university students in Denmark from 2010 to 2020*. [Online]
Available at: <https://www.statista.com/statistics/1111224/number-of-registered-university-students-in-denmark/>
[Accessed 12 Dec 2021].
- Usability.gov, n.d.. *User Interface Design Basics*. [Online]
Available at: <https://www.usability.gov/what-and-why/user-interface-design.html>
[Accessed 05 Dec 2021].
- Uxbert Labs, 2017. *10 MOBILE UX DESIGN PRINCIPLES YOU SHOULD KNOW*. [Online]
Available at: <https://uxbert.com/10-mobile-ux-design-principles/#.YblOfDMluU>
[Accessed 05 Dec 2021].
- VIA University College Horsens, 2021. *Campus Horsens*. [Online]
Available at: <https://en.via.dk/life-at-via/campuses/horsens>
- VIA University College, 2021. *About VIA*. [Online]
Available at: <https://en.via.dk/about-via>



10 Appendices

Appendix A User Interface Workflow Student

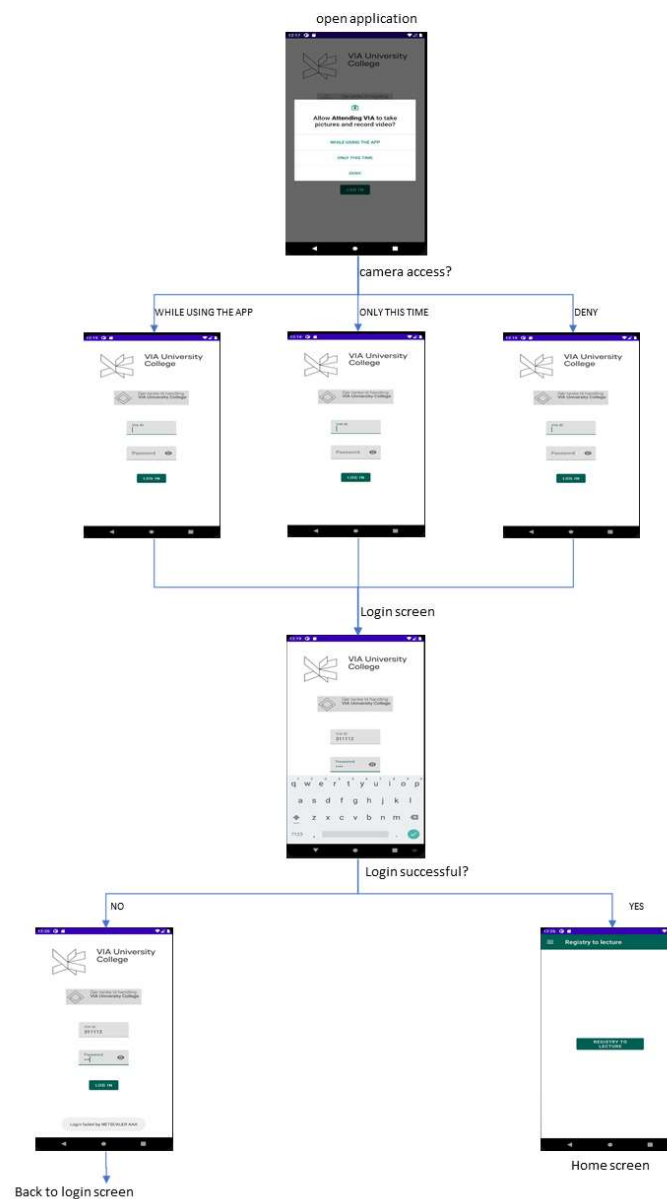


Figure A. 1: workflow login (end user student)

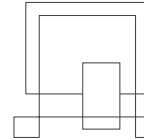


Figure A. 2: workflow registry to lecture (end user student)

ENG-FPRPM-A21: Room utilization optimization via mobile application

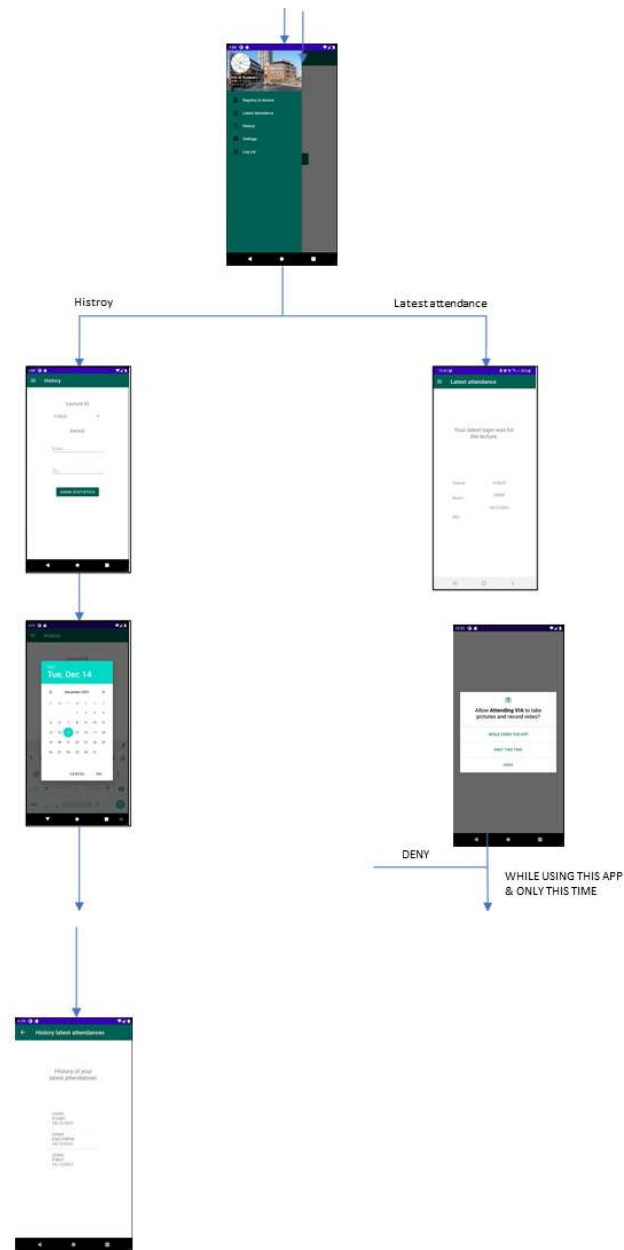
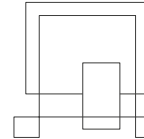
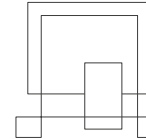


Figure A. 3: workflow histroy & latest attendance (end user student)



Appendix B User Interface Workflow Professor

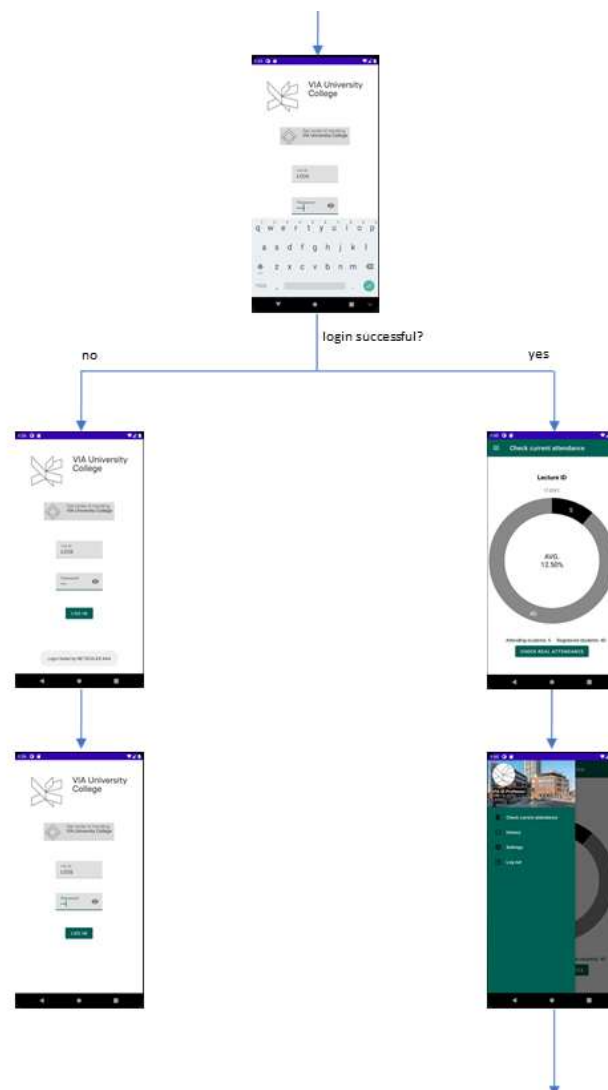


Figure A. 4: workflow attendance rate (end user professor)

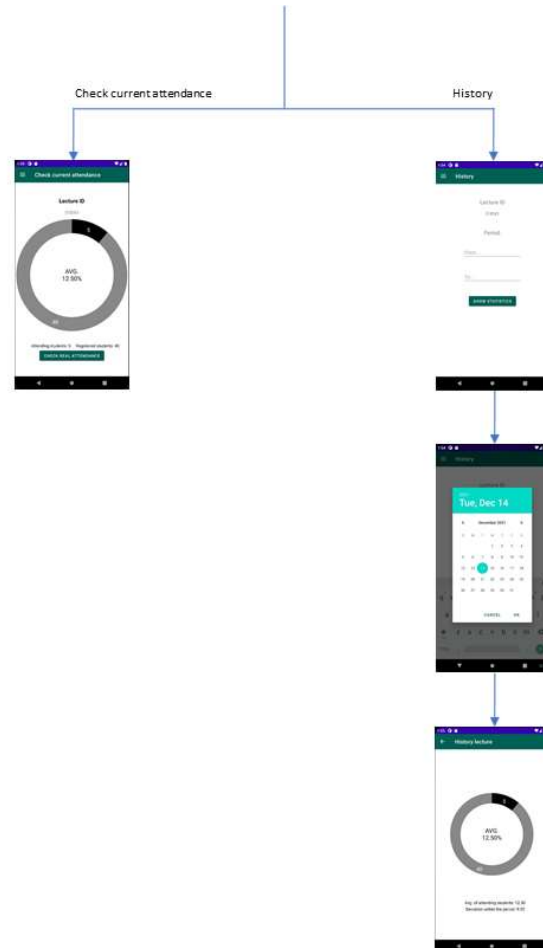
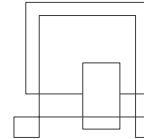
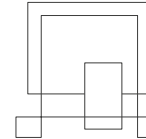


Figure A. 5: workflow history data (end user professor)



Appendix C User Interface Workflow Schedule Manager

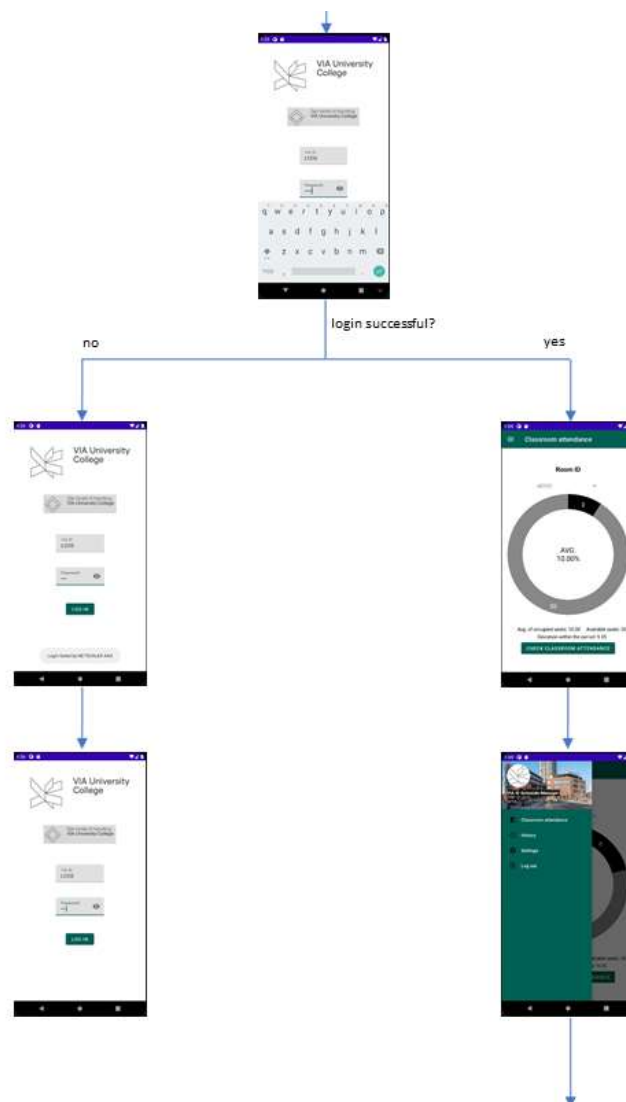


Figure A. 6: workflow average room occupation (end user schedule manager)

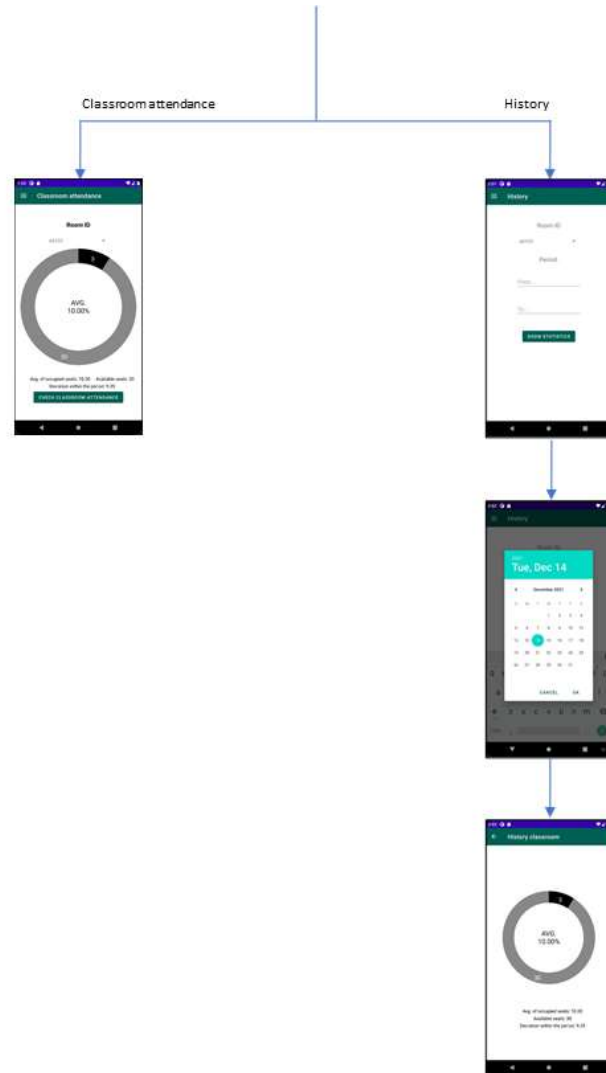
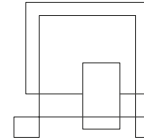


Figure A. 7: workflow history data (end user scheduler)