Requirement Specification

The application is called Student Attendance System, henceforth **SAS**.

Purpose of the SAS application:

- Currently, in many schools, teachers are spending valuable time performing a roll call at the start of every lecture, while they could be teaching. The system satisfies the need to save the teacher the time it takes to perform a roll call at the start of each lecture.

SAS will consist of:

- A front-end application
- A back-end application
- A database

Glossary

Teacher: One who is assigned topics to teach, and does so in classrooms populated with students.

Student: One who has a classroom assigned to them, and attends lectures at a school, to the best of their abilities.

Class: A semester as is currently defined in KEA fronter. Fx. SW22, WD22.

Subject: A subject of the curriculum. (Example: “Development of Large Systems”)

Lecture: A time period of 1 hour and 30 minutes of a subject.

User: An end-user of our system; can be either a student or a teacher.

(Attendance) Code: A short string of random characters used in the attendance process.

Attendance process: The process of marking attendances to a lecture where the teacher presents an attendance code. The students are then supposed to then use this code to mark that they attended the lecture in question.

Roll Call: The process of calling out a list of names to establish who is present.

Cheating: Marking attendance when the student is not physically present at the school, and attending their lecture.

Code: An 8-character string composed of only uppercase characters used when marking one’s attendance in a lecture.

System: Our software solution meets the requirements.

Instance: One specific component that has a specified responsibility within the system as a whole.

Functional Requirements

Sample table: Explains the format of the requirements

FR #	Functional Requirement Number
FR Name	Functional Requirement Name
User-Story	User story, describing the functionality from the users perspective
MoSCoW	**MUST HAVE** / **SHOULD HAVE** / **COULD HAVE** / **WON'T HAVE**
User Type	Student / Teacher / Student and Teacher
Notes	**General**: Additional notes to better understand the functional requirement to avoid any confusion for the readers of this document, the developers working on this project, and the project owner. **Prerequisites**: Notes the prerequisites for the functional requirement / user story, For example, that the user has to have logged in successfully in order to make use of the function. **Requirements**: Describes the requirement in detail, for example describing the view that the user will have, such as the information displayed, buttons, input fields, etc,. Along with additional information, such as input value constraints, place holder text, clickable fields, etc,.

FR #	1
FR Name	Authentication
User-Story	As a user I want to be able to sign-up and login using a third party authentication broker.
MoSCoW	**MUST HAVE**
User Type	Student and Teacher

Notes	**General**: Authentication is used to ensure that only valid users are allowed access to the system. Additionally, only teachers will see the teacher page. Students have their own view of the system.
	
	Prerequisites:
	
	• User has a Microsoft account.
	• Microsoft account has to be a KEA account, where the email extension can be one of the following:
	  • `@stud.kea.dk` - for students
	  • `@kea.dk` - for teachers
	
	Requirements:
	
	The third party authentication broker has be Microsoft.
	
	The system should present the user a webpage that contains a login button that will redirect the user to the third party authentication broker of choice.
	
	The webpage should also contain a navigation bar, as seen in the top of figures 5 and 6.
	
	The page's styling options should match the Figma design seen in figure 1 as close as possible.
	
	The Student Attendance System should grant access based on the account type.
	
	The system will validate with Microsoft that the user is who they claim to be.
	
	Once that is done, the system will retrieve the user from the database to be used in queries and requests.

FR #	2
FR Name	Student - Attendance
User-Story	As a student I want the option to submit a code to attend a lecture.
MoSCoW	**MUST HAVE**
User Type	Student

Notes	**Prerequisites**:
	• Student has logged in successfully.
	Requirements:
	The system should present the user a webpage that contains the following:
	• An input field that allows the user to type in the attendance code
	  • It should have the placeholder text: “Class code”
	  • It should only allow the user to input uppercase letters
	  • It should only allow the user to input a maximum of 8 characters
	• A button that will attempt to mark the user’s attendance to a lecture for the inputted attendance code
	  • The button text should be: “Attend class”
	  • When clicked, it should validate that the code is made up of uppercase letters and has a size of exactly 8 characters, and attempt to mark the user’s attendance
	  • If the code is invalid, it should display an error message
	  • If the code is valid, it should display a modal notifying the user that they successfully attended the lecture
	• A button that will redirect the user to a page where they can check their attendances
	  • The button text should be: “Check attendance”
	  • It should redirect to a “404 - Not Found” page, as seen in figure 7
	The webpage should also contain a navigation bar, as seen in figure 5.
	The page's styling options should match the Figma design seen in figure 3 as close as possible.
	The code is generated and shared by the teacher of that lecture.
	The system validates that the code is correct, and marks the student as attended.

FR #	3
FR Name	Teacher - Attendance Start
User-Story	As a teacher I want to be able to generate an attendance-code for a lecture, and I want the code to re-generate every 15 seconds until the attendance process is finished.
MoSCoW	**MUST HAVE**
User Type	Teacher

Notes	**Prerequisites**: • Teacher has logged in successfully. • Teacher has selected a subject from the list of subjects. **Requirements**: The system should present the user a webpage that contains the following: • A paragraph that displays the name of the current lecture's subject (Example: Testing) • A paragraph that displays the current available attendance code • A button that takes the user to the previous page in their history and stops the attendance process • It should contain the text: "Stop attendance" The webpage should also contain a navigation bar, as seen in figure 5. The page's styling options should match the Figma design seen in figure 2 as close as possible. The attendance code is retrieved by calling the `/class-code/:lectureId` endpoint with a **GET** request. It requires the following path parameters: • `lectureId` this is the ID of the lecture for which an attendance must begin. The system will generate a lecture with the provided ID and then a code • The code is a random string of exactly 8 capital letters. • The system responds with the generated code Once an attendance has been started, the endpoint **GET** `class-code/:lectureId` can be used to get a new code for the same lecture id. The system will keep calling the **GET** `class-code/:lectureId` endpoint every 15 seconds until the end of the attendance problem.

FR #	4
FR Name	Subjects by Teacher
User-Story	As a teacher I want an overview of all my subjects
MoSCoW	**MUST HAVE**
User Type	Teacher

Notes	**Prerequisites**:
	
	• Teacher has logged in successfully.
	• The teacher is supposed to teach some lectures this semester.
	
	Requirements:
	
	The system should present the user a webpage that contains the following:
	
	• A table containing a list of the user's current subjects:
	  • The table headers should be as follows: Class , Subject, Students.
	  • When clicking on any of the table entries, the user should be redirected to a new page specified in FR #3
	  • The click functionality of the entries can remain unfinished until FR #3 is added.
	• A search input field that allows the user to filter the table entries:
	  • The placeholder text should be: "Search"
	  • The filtering functionality is not part of this user story
	• An add button with the text of: "Add"
	  • The button functionality is not part of this user story
	
	The webpage should also contain a navigation bar, as seen in figure 5.
	
	The page's styling options should match the Figma design seen in figure 4 as close as possible.
	
	The attendance code is retrieved by calling the **GET** `/subjects/by-teacher/:teacherId` endpoint.
	
	It requires the following path parameters:
	
	• `teacherId` this is the ID of the teacher for to whom the subjects are linked to
	
	The system will return the list of lectures that the teacher is a part of.
	
	The response contains an array of the following information in JSON format:
	
	• `name` in the `String` format
	• `startedAt` a date in the UTC time in ISO 8601 format `YYYY-MM-DDTHH:mm:ss.xxZ`
	• `endedAt` a date format of `YYYY-MM-DD HH:ss`
	• `classId` in the `Int` format

FR #	5
FR Name	Teacher - Attendance Stop
User-Story	As a teacher I want to be able to stop the attendance process for a lecture.
MoSCoW	**MUST HAVE**
User Type	Teacher
Notes	**Prerequisites**: • Teacher has logged in successfully. • Teacher has selected a subject from the list of subjects. • The attendance process has started. • The system should present the user a webpage as described in FR #3. **Requirements**: The "Stop attendance" button should stop the attendance process by deleting the attendance code, ending the lecture, and taking the user to the previous page in their history. The system will remove the access code from its whitelist and confirms the action.

Non-Functional Requirements

Performance:

Performance requirements are applicable to the system for up to a load of 6.000 concurrent requests to the back-end system.

Accepted performance requirements for the system are as follows:

- 99% of requests should be resolved within 400ms.

- The remaining 1% of the requests should not take longer than 2000ms.

Where the metrics represent the time it takes the backend system to process and respond to the request internally.

Scalability:

The goal of the system in terms of scalability is that it should be able to support the higher education facility KEA (Copenhagen School of Design and Technology).

The application should be able to scale horizontally or vertically as needed, to accommodate any amount of potential users.

Portability:

The system should be running on a virtual machine, ensuring that the system is easily portable from one platform to another if needed.

Compatibility:

The website should appear fully functional using the following browsers in desktop mode:

- Mozilla Firefox
- Safari
- Google Chrome
- Brave

Reliability:

Users can access the system's functionality 98% of the time without failure.

Functionalities might have constraints that don't allow for usage within a given time frame or have some prerequisites defined, which do not count towards the remaining 1%.

It is considered a failure if the user can not access the functionality as it is intended to be used, for any reason, whether internal or external system faults cause it.

Availability:

The system should be highly available by having an availability level of at least 99,9%. This means that our system is not allowed to be down for more than 8,77 hours per year, 10,08 minutes per week, or 1,44 minutes per day.

Maintainability:

The system should be tested with at least 90% code coverage. It is also important that the tests cover the most sensitive components and highly used functions. The system must be documented.

Serviceability:

The system should be easy to install and configure.

The system should be easily monitored by logging every important event and program state, enabling developers to quickly identify and fix the cause of failures in the system.

The system should provide health checks that can be used to determine if certain components are working as intended.

Security:

The system will be accessed only by users with the roles defined in the functional requirements. The system should be secured from SQL injections, and protect user information.

Usability:

The system should not require any training to operate, and its controls should be self-explanatory.

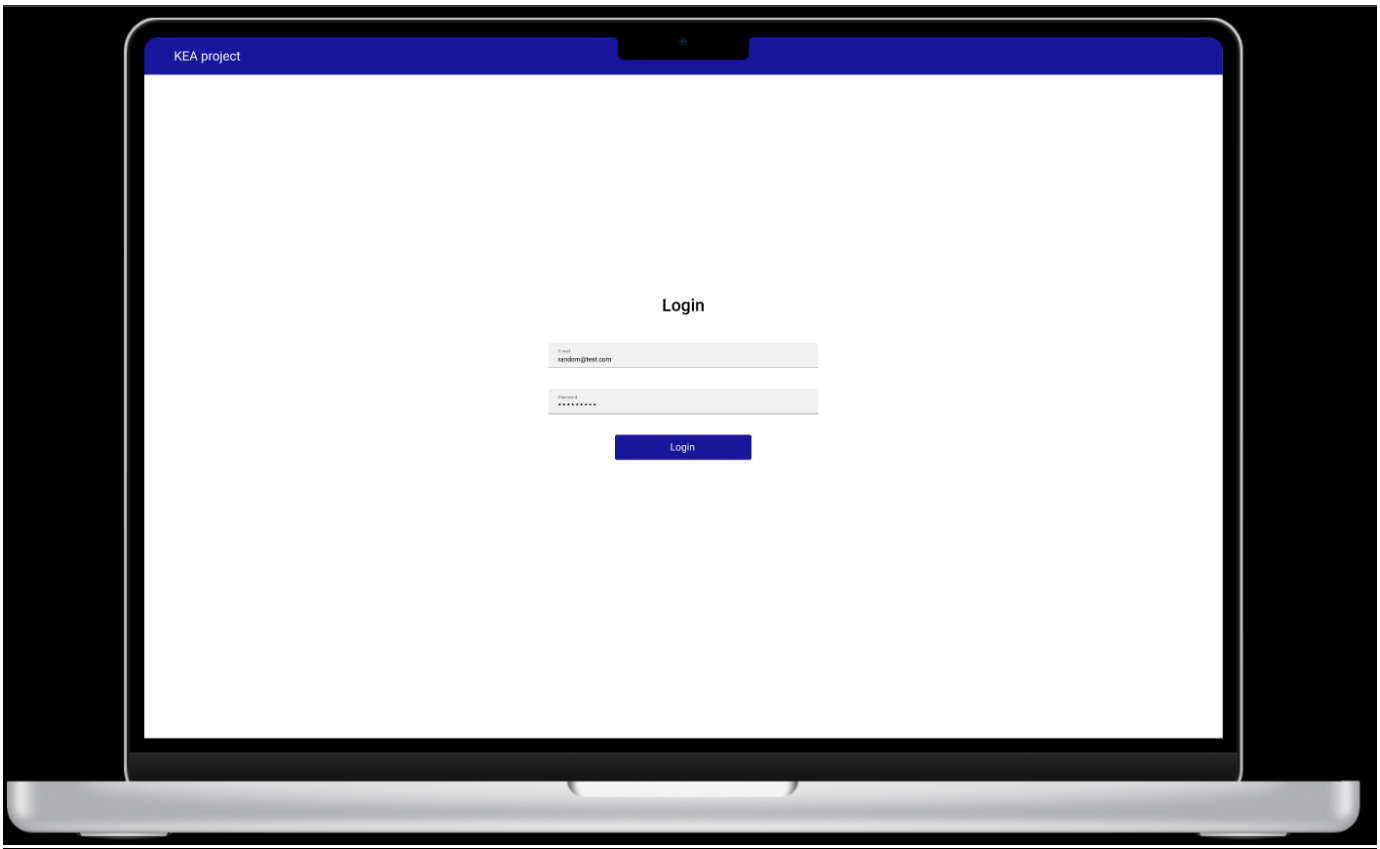
The system should have a simple UI. It should not allow the student to make any changes to the system except to view data and register their attendance to a lecture.

The system should require as few steps as possible in order to achieve the main functionalities for both user groups of the system.

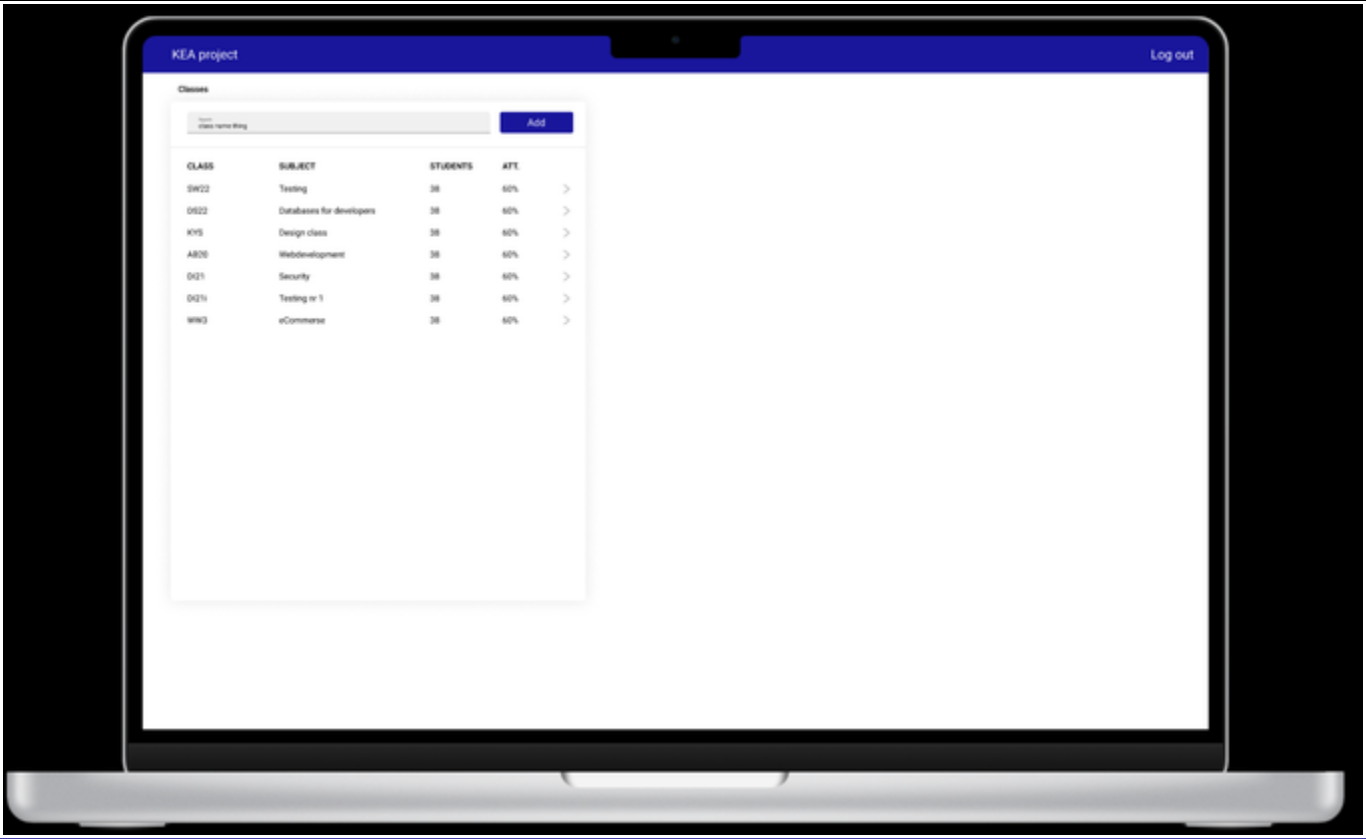
The teacher will have minimum functionality which will keep it easy to use and reduce the time it takes to perform a roll call.

Usability testing can be done throughout the development of the system in order to ensure that the system is user-friendly and to expose user experience defects.

Figures:


KEA project
Login
random@test.com
Login


KEA project
Log out
Development of Large Systems
Code:
FKJSCRH
Back



Role Call
Att. class
Check Att.
Log out

Role Call
Manage Classes
Log out


Role Call
Manage classes
Page not Found 404