Bilkent University

Department of Computer Engineering

Senior Design Project

Tutoryum

Detailed Design Report

Barış Ogün Yörük

Halil Özgür Demir

Mustafa Çağrı Durgut

Oğuzhan Özçelik

Yusuf Miraç Uyar

Academic Supervisor: Can Alkan

Industry Expert: Mehmet Gök

Jury Members: Erhan Dolak, Tağmaç Topal

1. Introduction

Are you tired of traditional exam preparation methods that fail to provide direct applicability of education to exams? Have you ever wished for a platform where you can communicate directly with someone who has excelled in the exam you want to take? If so, you're in luck! Tutoryum is a revolutionary platform that connects students with instructors who have aced their exams and have registered their achievements in the Tutoryum system.

The education system is constantly evolving, but one thing remains constant: the need for testing. Exams are an inevitable part of the education system, and they play a crucial role in determining a student's future. However, one of the biggest problems with traditional exam preparation methods is that they fail to test the direct applicability of education to exams. Many instructors specialize in only one subject, and while they may provide detailed information about their field, their approach may not align with students' motivations to pass exams or get a degree.

This is where Tutoryum comes in. Our instructors not only have expertise in their respective fields, but they also have a deeper understanding of the exam and its methods than any other traditional instructor. As Einstein once said, "If you can't explain it simply, you haven't understood it deeply." Our instructors understand their fields so deeply that they can explain them in simple terms, making it easier for students to grasp complex concepts and pass their exams.

Traditional teaching aid methods may copy the education system's method, but they often fail to motivate students. Many students find their education process boring and unproductive, and this is where Tutoryum's approach is unique. We connect students with instructors who can not only provide education but also motivate and inspire them to achieve their goals. Our platform provides an all-in-one experience, where students don't have to register with any other platform to join the meeting. All of the video meetings are conducted inside the platform, making it easier for students to access the knowledge they need.

Tutoryum is a web application designed to provide students and instructors with an easy-to-use platform that helps them find what they need quickly. Our platform is built on the principles of usability, performance, reliability, marketability, extendibility, security, scalability, maintainability, flexibility, modularity, aesthetics, and more. We use modern technologies like WebRTC for video conferencing and interactive whiteboard features, PostgreSQL for relational databases, Redis for cache techniques, and a RESTful or GraphQL API for communication between the front and back ends.

Our platform is not just limited to video conferences; we provide easy-to-use class materials that instructors can use to improve education. They can create virtual courses by uploading their materials to the system, making it easier for them to teach their classes. We also offer built-in whiteboard and slideshow facilities, making it easier for instructors to share their knowledge with students.

In conclusion, Tutoryum is more than just a platform; it's a revolution in the education system. Our mission is to make education accessible to all, regardless of their location or background. We want to empower students and instructors alike, providing them with the tools they need to succeed in their academic pursuits. With Tutoryum, students can achieve their goals, and instructors can share their knowledge and inspire the next generation of leaders.

1.1. Purpose of the system

The purpose of the Tutoryum platform is to provide a convenient and efficient way for students to connect with instructors who have excelled in their exams and achieve their academic goals. Traditional exam preparation methods often fall short when it comes to preparing students for specific exams, as the education provided may not be directly applicable to the exam. Tutoryum's approach addresses this issue by offering a one-on-one video interview system with instructors who have first-hand experience in excelling in the exam.

1.2. Design goals

Usability: One of the primary goals of Tutoryum is to create a user-friendly platform that is easy to navigate and use. To achieve this, the platform will feature an intuitive user interface and straightforward design that guides users to the information they need. Tutoryum aims to create an environment where users feel comfortable and confident, enhancing their overall learning experience.

Performance: Tutoryum will prioritize performance to ensure that the platform operates efficiently and effectively. The video conferencing technology used will be optimized for speed and reliability to ensure a smooth and uninterrupted experience for both tutors and students. The platform will also feature fast load times, minimal latency, and a seamless user experience.

Reliability: Tutoryum recognizes the importance of reliability and strives to provide a platform that is always accessible and dependable. The platform will have robust backup systems in place to ensure that users can access their materials and participate in sessions at all times. Tutoryum will also ensure that all user data is secure and protected.

Marketability: Tutoryum aims to create a platform that is appealing and competitive in the market. The platform will be designed with user feedback in mind, providing features that are relevant and useful to both tutors and students. Tutoryum aims to create a unique and attractive platform that stands out in the market and attracts users.

Extendibility: Tutoryum aims to create a platform that is easily extendable and can adapt to the evolving needs of its users. The platform will be designed to allow for the integration of new features and functionality as needed. Tutoryum aims to create a platform that can grow and evolve with its user base.

Security: Tutoryum recognizes the importance of security and aims to provide a platform that is safe and secure for all users. The platform will have robust security protocols in place to protect user data and ensure that all interactions on the platform are secure. Tutoryum aims to create an environment where users can feel safe and confident.

Scalability: Tutoryum aims to create a platform that can scale with the growth of its user base. The platform will be designed to accommodate a large number of users, ensuring that it can handle high traffic volumes without compromising performance or reliability. Tutoryum aims to create a platform that can grow with its users.

Maintainability: Tutoryum aims to create a platform that is easy to maintain and manage. The platform will be designed with maintenance in mind, making it easy for developers to update and maintain the platform as needed. Tutoryum aims to create a platform that is reliable and easy to maintain.

Flexibility: Tutoryum aims to create a platform that is flexible and can adapt to the unique needs of its users. The platform will be designed to allow for customization and configuration, ensuring that users can tailor the platform to their needs. Tutoryum aims to create a platform that is flexible and adaptable.

Modularity: Tutoryum aims to create a platform that is modular and easy to work with. The platform will be designed with a modular architecture, making it easy to add, remove, or modify components as needed. Tutoryum aims to create a platform that is easy to work with and customize.

1.3. Definitions, acronyms, and abbreviations

Tutoryum: The name of the proposed platform that offers one-on-one video interviews with instructors who have excelled in an exam.

WebRTC: Web Real-Time Communications is an open-source project that provides browsers and mobile applications with real-time communication via simple application programming interfaces.

PostgreSQL: An open-source relational database management system used for storing and managing data.

RESTful API: A Representational State Transfer (REST) API is an architectural style for building APIs that use HTTP requests to GET, PUT, POST, and DELETE data.

GraphQL API: An API that uses the GraphQL query language to define the data structure and retrieve data from the server.

1.4. Overview

Tutoryum is a web-based application that aims to provide a platform where users can arrange one-on-one video conferences with instructors who have excelled in their fields and have their achievements registered in the system. The platform is designed to help students prepare for exams and provide a better education experience.

The proposed system will have several features, such as an interactive whiteboard, slide sharing, optimized whiteboard recording and storage, and automated slide creation from the whiteboard at the end of the session. Additionally, the platform will provide live streaming and live interaction with the lecturer and the student, making the experience more engaging and interactive.

The platform will also offer easy-to-use class materials for improving education, and tutors can create virtual courses by uploading their materials to the system. Tutors can share their availability schedule on their page, and anyone interested in having a one-on-one video interview with that instructor can book an available time slot.

To achieve its design goals of usability, performance, reliability, marketability, extendibility, security, scalability, maintainability, flexibility, modularity, and aesthetics, the platform will utilize WebRTC technology for setting up video conferencing and interactive whiteboard features. Relational databases such as MySQL will be used for data storage. The platform will also use a RESTful API or GraphQL API to communicate the back end with the front end.

Tutoryum will be built with progressive web application principles in mind, utilizing service workers, manifests, and other web-platform features in combination with progressive enhancement to give users an experience on par with native apps.

2. Current Software Architecture

Video conferencing tools like Zoom [1], Microsoft Teams [2], and Google Meet [3] can be considered competitors of our virtual classroom system as they also provide online communication and collaboration tools for remote learning.

Let’s shift our attention to the software architecture of Zoom, one of the leading virtual meeting platforms. Here are the main components of Zoom's software architecture [7]

2.1. User Interface

Zoom's user interface is built using web technologies like HTML, CSS, and JavaScript. The user interface is responsible for displaying the video conference and providing controls for users to interact with it.

Zoom provides a complete solution for any type of device, whether it is a mobile or a desktop device. Zoom’s client has native solutions for each of them. Currently, there are a web interface, a desktop application, a native Android application, and a native IOS application for Zoom’s client. All of these apps provide the same experience, but they use the merits of being a native application.

2.2. Web Server

The web server component is in charge of processing and delivering requests from user web browsers. **Node.js** and **Express.js** are used to build the web server, and **REST API**s are used to connect it to other system parts.

Millions of users and concurrent video conferencing sessions may be handled using Zoom's backend architecture, which is very **scalable**. The cloud-based architecture used by Zoom is built on top of **Amazon Web Services (AWS)** and other external cloud service providers. Zoom can dynamically scale its infrastructure up or down in response to variations in user demand thanks to this cloud-based architecture.

Zoom's usage of a **distributed cloud infrastructure** is one of the main aspects of its backend architecture. This architecture, which comprises numerous servers spread across various regions, enables Zoom to split its workload among various data centers and reduce latency for users throughout the globe. **Redundancy** is another feature of the distributed architecture that guarantees consumers will continue to receive service even if one server fails.

The backend of Zoom similarly employs a **microservices architecture**, in which each service is made to carry out a particular task, such as maintaining users, dealing with video and audio feeds, or handling chat messages. Since each microservice can be scaled individually, Zoom is able to expand particular parts of its infrastructure to meet rising demand.

To ensure scalability, Zoom's backend architecture also includes features like **load balancing, auto-scaling,** and **elastic storage**. Load balancing distributes incoming traffic across multiple servers, ensuring that no single server becomes overwhelmed. Auto-scaling allows Zoom to automatically adjust its infrastructure in response to changes in user demand, while elastic storage ensures that Zoom can quickly provision additional storage capacity as needed.

2.3. Media Server

The media server is a critical component of Zoom's architecture that is responsible for managing audio and video streams between participants in a video conference. It uses **WebRTC** technology to establish real-time communication between participants.

WebRTC is a free and open-source technology that enables real-time communication between web browsers and other applications. It uses a combination of **UDP** (User Datagram Protocol) and **STUN** (Session Traversal Utilities for NAT), and **TURN** (Traversal Using Relays around NAT) servers to establish direct peer-to-peer connections between participants.

Zoom's media server is built using a combination of technologies, including **Node.js**, **WebRTC**, and **WebSocket**. Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to build server-side applications using JavaScript. WebRTC provides the real-time communication capabilities necessary for video conferencing; WebSocket is a protocol that enables bi-directional communication between web browsers and servers.

2.4. Cloud Storage

Zoom's cloud storage component is used to store data related to users, meetings, and recordings. It is built using **Amazon Web Services (AWS)** and provides scalable storage for the platform.

2.5. Chat Service

Zoom's chat service allows users to send text messages to each other during a video conference. It is built using **XMPP** technology and provides real-time messaging capabilities to users.

2.6. Zoom API

The Zoom API allows developers to integrate Zoom's functionality into their own applications. It provides a set of REST APIs for managing users, meetings, and recordings.

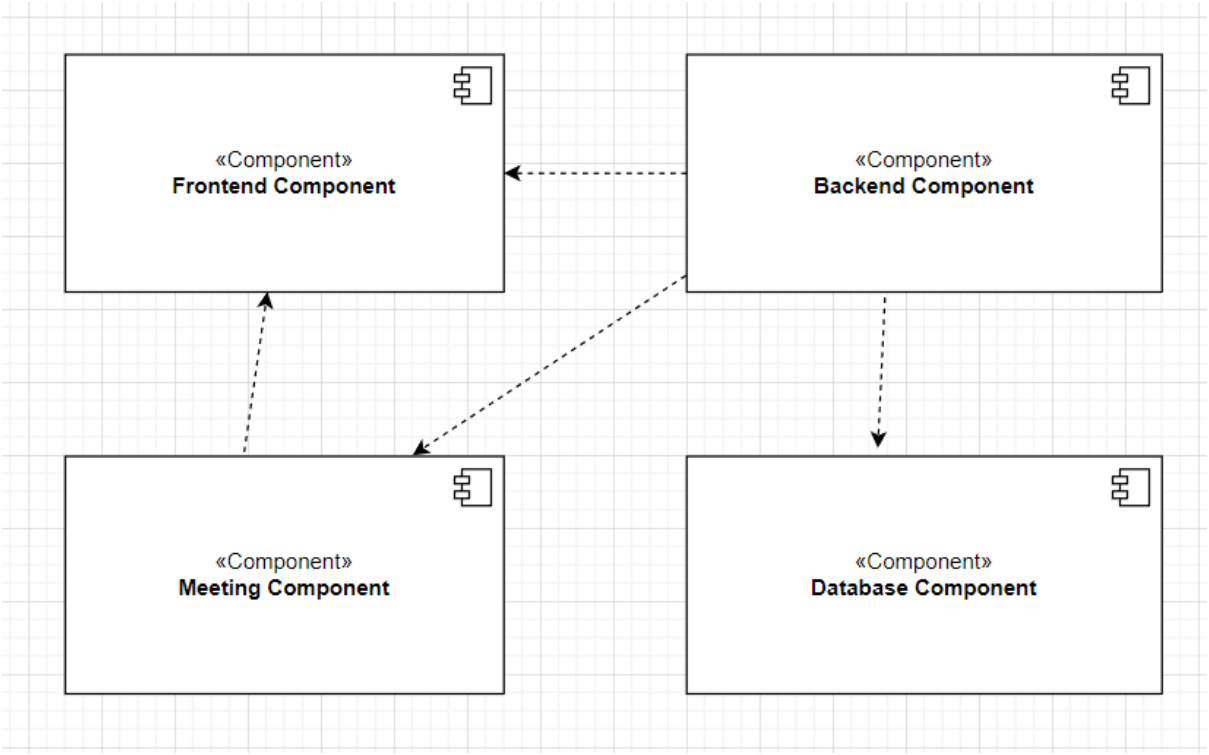
3. Proposed Software Architecture

3.1. Overview

Tutorium is designed to be a reliable and scalable platform to provide a good experience for its users. One of the main focuses of software architecture is maintainability. In other words, the Tutorium codebase will be open to adding new features seemingly for the future since the video communication baseline will be useful for many different areas, not only education. In this section, subsystem decomposition, persistent data management and access control and security will be explained.

3.2. Subsystem decomposition

In Tutorium, the system is decomposed into four main components: frontend component, backend component, database component and meeting component.



The details of these components are explained in section 4 explicitly.

In a higher sense, this is how the system looks like. The reason behind the decision to break the system into such components are:

1. These components will be implemented in different languages and are independent of each other.
2. It is easier to share the tasks in the team environment since teammates will focus on their parts and will be responsible for their components. That way, when someone needs to ask questions, he can ask the responsible

teammate, and that teammate will know the details because of the partition of workflow.

3. It will be easier to add new features since all the components work independently, and this will provide maintainability. This is a big part that the team focuses on because after implementing core features, the base of Tutorium can be open to endless possibilities.

The frontend component is where the user experience will be. Users will use the system here, and all the calculations are abstracted to provide a better experience. The backend component will be responsible for handling API requests, handling the database system, and creating connections between users for video communication using the meeting component. The database component will be responsible for storing the necessary data.

3.3. Hardware/software mapping

The system does not include any hardware components, and it is all software centered. Cameras are used, but it is not a part of the system. The OS handles the communication, and we use the visuals that are provided by the camera without any extra work.

3.4. Persistent Data Management

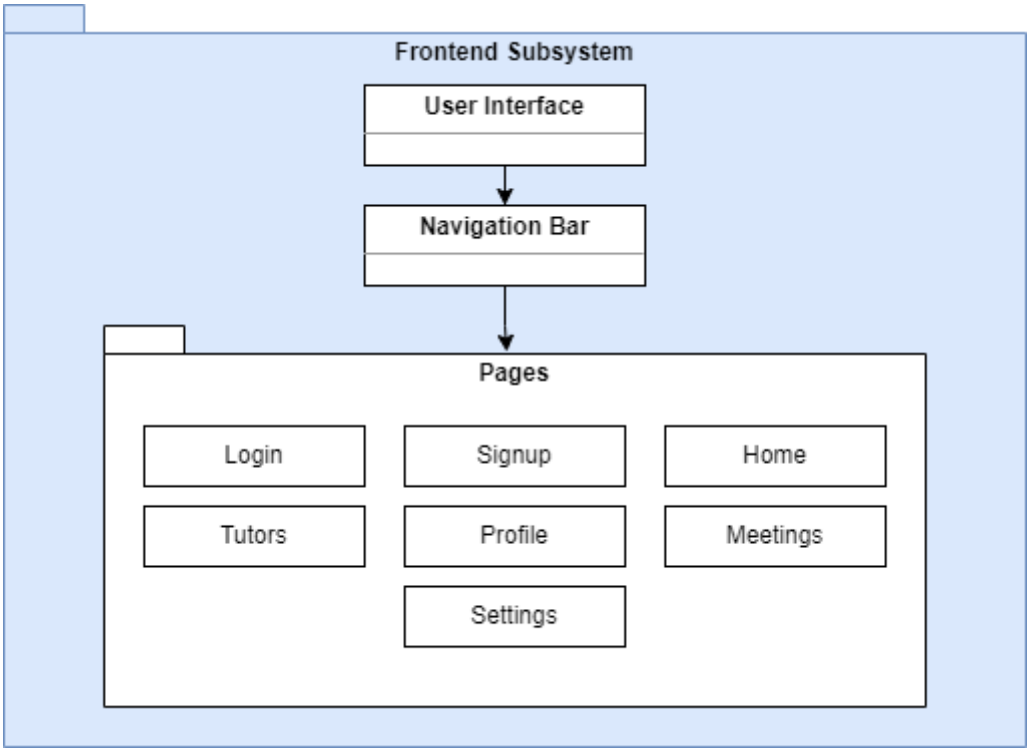
The system does not need a huge dedicated database to store the necessary data. Since our system works peer-to-peer, video communication will not be a burden on the server or the database system. The database system will be responsible for storing the user data and meeting information; the data users have stored during the meeting to take a look at another time or use for different purposes, such as whiteboard screenshots or lecture documents. User data is stored using PostgreSQL, and it needs no specific optimization because user data is not huge. Whiteboard screenshots will be stored by storing vector data, and it is more optimized than storing it as a png/jpg file.

3.5. Access control and security

In Tutorium, each user will log on to the system using their credentials, and passwords will not be stored in the database directly. SHA values of the password will be stored, and authentication will be done using these values. Also, the system will use authentication for all server requests, and each endpoint will be protected for its users. In other words, a person will not be able to make API requests to fetch the data of other users.

4. Subsystem Services

4.1. Frontend Subsystem



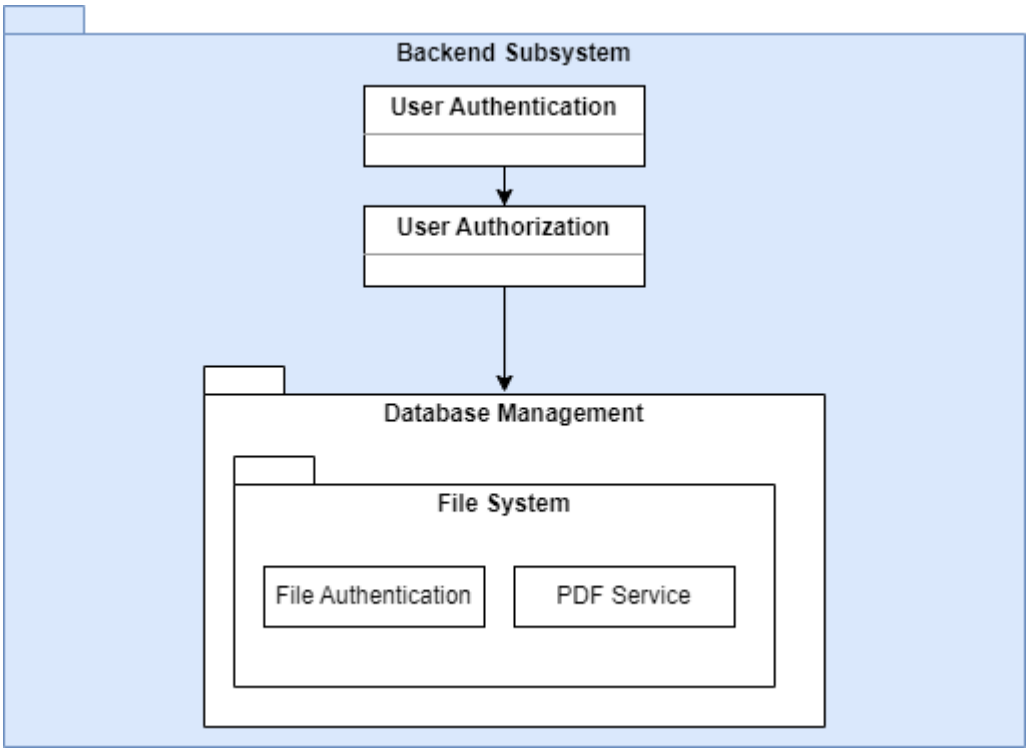
The Frontend subsystem is responsible for providing users with an intuitive and user-friendly interface to navigate the website's different pages. The website includes various pages, such as Login, Signup, Settings, Home, Tutors, Profile, and Meetings pages, which serve different functions.

The Login and Signup pages allow users to create accounts and log in to the system. Once users have logged in, they can search for tutors on the Tutors page and access their own tutor preferences on the Profile page. The Meetings page allows users to view and join upcoming virtual conferences.

To facilitate easy navigation between these pages, the website includes an upper bar on each page. This bar enables users to move quickly between different pages, enhancing their overall user experience. Additionally, users can modify their personal information on the Settings page.

In summary, the Frontend subsystem's primary objective is to enhance the website's user experience by providing users with intuitive user interface elements and easy navigation between different pages.

4.2. Backend Subsystem



The Backend Subsystem is responsible for managing user accounts, authentication, and authorization. Moreover, it is also used for database management. One of the primary functions is establishing new connections between two users. When the meeting time comes, the connection between the two users must be formed, and they are directed to the Meeting Subsystem for other functionalities of a meeting.

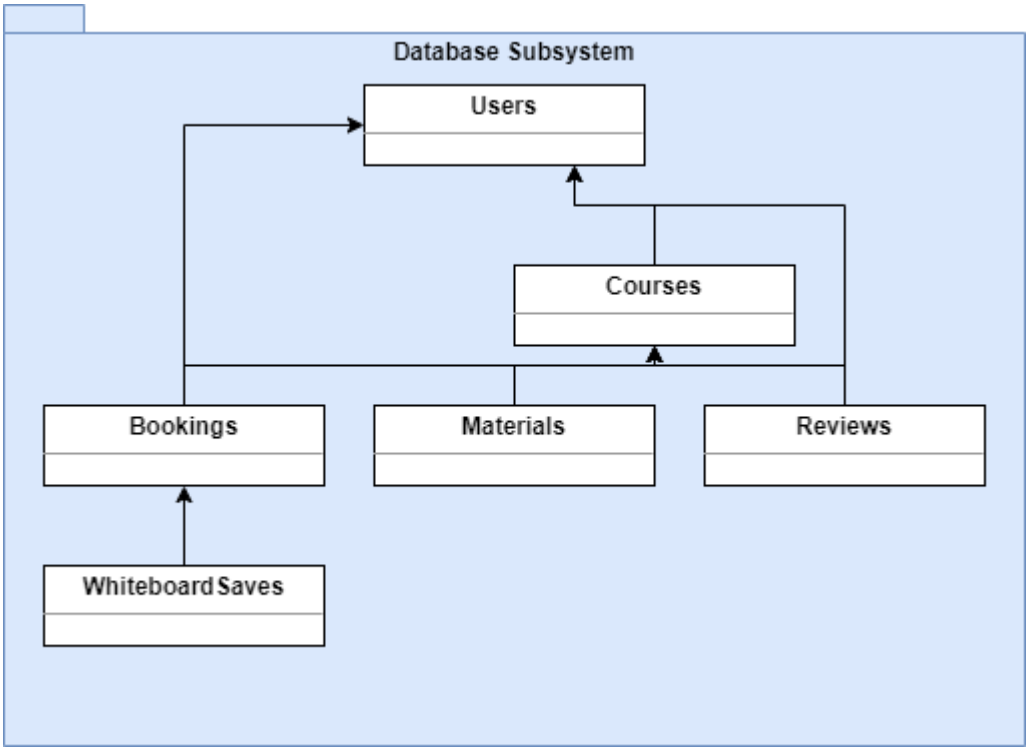
Another function of this subsystem is authenticating tutors. While signing in, tutors upload the required document to the system, and Backend Subsystem controls the viability of this document. If the document is accepted, the new tutor will be added to the system. Also, Backend Subsystem adds other users to the database. For the security of the system, Backend Subsystem handles user authorization. This happens when the user tries to log in to the system.

Backend Subsystem is also responsible for managing the database. Elements like courses, reviews made on tutors, users, and course materials are controlled by the Backend Subsystem. It forms a bridge between the frontend and the database for these elements. Also, this subsystem uses file service, which is used for making required alterations on those files and saving them on the database if required.

In order to save the movements in interactive tools, Meeting Subsystem calls Backend Subsystem and saves those into the database. It can also create pdf from the saved tools, which are not limited to whiteboard. Screenshots of shared screens or any source provided by the tutor can be used to create this pdf. Backend Subsystem also controls the meeting from time to time in order to find if there is a problem with the connection. If a connection problem is found, it informs users using the Frontend Subsystem.

Overall, the Backend Subsystem is a critical component of this application. It provides essential services for website management. Also, it is responsible for the connection between users and maintenance of it.

4.3. Database Subsystem

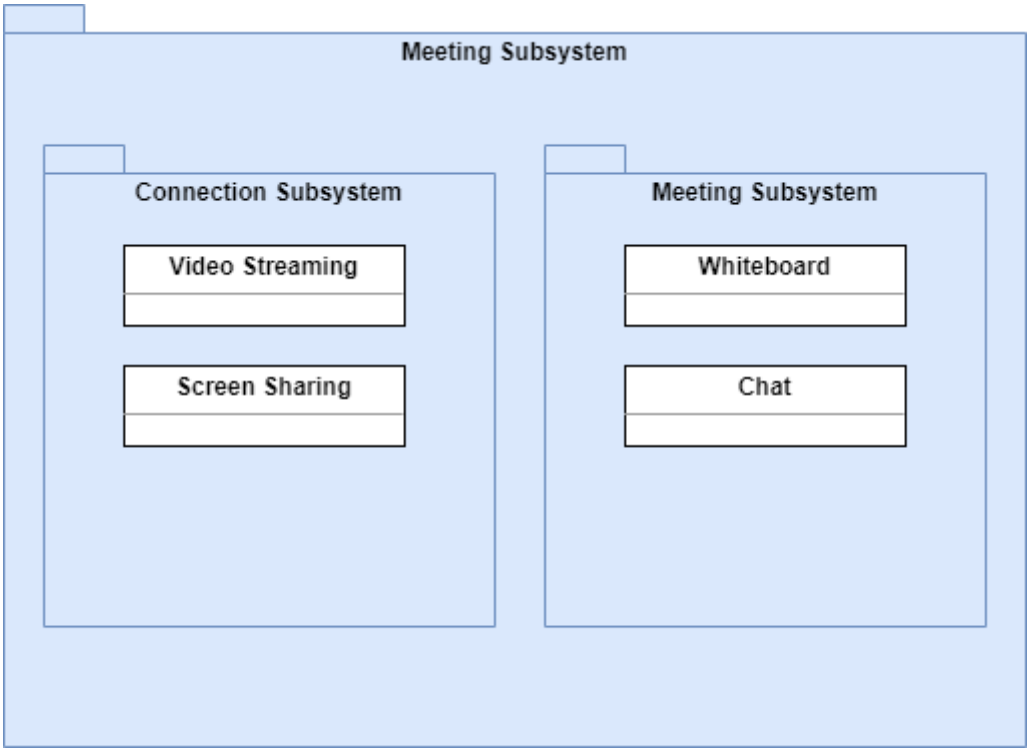


The Database Subsystem is responsible for the storage and retrieval of data related to users, their profiles, their preferences, and their interactions with the overall system. This subsystem provides a reliable, scalable, and secure way to store and access data.

Courses, users, bookings, and reviews are stored in the database. These are controlled by the interactions between the frontend and the backend. Frontend retrieves this data from the database to show the proper information to the user. Also, by retrieving required information from the frontend, the backend changes the database accordingly. Tutors can also upload course materials as files that are stored in the database. These files can be used by users who are enrolled in that course.

This subsystem is also used for storing meeting information and interactive tools used in those meetings. When any problem occurs in one meeting, the required information about that meeting is retrieved from the database, and the same meeting is formed again. Moreover, as interactive tools are also saved, they can be retrieved as pictures or presentations after meetings or during meetings. All the drawings on the whiteboard are stored as vector components; therefore, storing these will not threaten the scalability of the system.

4.4. Meeting Subsystem



After the meeting has been planned between a tutor and a student, they are directed to a Meeting Component where all conferences are held. This subsystem has two components which are the connection subsystem and interactive tools.

4.4.1. Connection Subsystem

Meetings between two users are happening peer to peer. For this connection, WebRTC technology is used. This subsystem is responsible for video streaming, screen sharing, and slide sharing. Also, changes happening on interactive tools (like a whiteboard) are shared between users using this system. Lastly, these changes are also shared with the server so they can be used later.

4.4.2. Interactive Tools

As this application tries to serve an educational environment to its users, using interactive tools is one way to do this. One of these tools is the whiteboard. Both student and tutor will use the whiteboard. All the drawings on this whiteboard will be mapped to vectors which are made to ease the storage of these drawings. Users can save the whiteboard as a picture, or at the end of the meeting, whiteboard activity can be used to create presentations. Another tool is chat. This chat can also be retrieved as a transcript at the end of the meeting.

5. Test Cases

We will use the following format for test ids. X stands for the requirement, and Y stands for the component of the application.

Test ID Description: X_YY_NO

X: F (Functional), N (Nonfunctional)
YY: BE (Backend), MT (Meeting Tools), UI (User Interface), VC (Video Connection)
NO: Number of the test in that system

Test ID	*F_BE_01*
Description	Tutors should be able to create verifiable courses with the following information: description, duration, name, subject, and verification document. If the verification document is valid, the course should be verified.
Priority	High
Dependency	F_BE_08
Precondition	- The tester should be a tutor. - The tester should have a valid document to verify the course s/he creates. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the course creation endpoint. The request should include an authorization scheme and the information regarding the course: description, duration, name, subject, and verification document. 2. Tester should check the database entities. 3. Tester should check the return value of the request.
Expected Result	- A new course entity should be added to the database. - The new entity should include the given information. - The new entity should have a foreign key relationship with the user of the tester. - The verification documents should be checked and the course should be verified in case it is valid. - The expiration date should be set to the expiration date of the document in case it is valid. - The returned body should include all the information given as well as it should include the database ID of the newly created entity. - The return code should be successful.

Test ID	*F_BE_02*
Description	Students should be able to review the courses they have been registered for.
Priority	Medium
Dependency	F_BE_01
Precondition	- The tester should be a student. - The tester should be registered for a course. - The tester should not have reviewed the same course before. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the review creation endpoint. The request should include the authorization scheme and the information regarding the review: comment, and rating. 2. Tester should check the database entities. 3. Tester should check the return value of the request.
Expected Result	- A new review entity should be added to the database. - The new entity should include the given information. - The new entity should have a foreign key relationship with the user of the tester. - The new entity should have a foreign key relationship with the course that the tester reviewed. - The new entity should have its *createdAt* field filled with the request time. (it can be later than the actual time due to the network connection) - The return value of the request should include all the information given as well as it should include the database ID of the newly created entity. - The return code should be successful.

Test ID	*F_BE_03*
Description	Tutors should be able to upload supplementary material to courses they have created.
Priority	Medium
Dependency	F_BE_01
Precondition	- The tester should be a tutor. - The tester should have created a course. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the material creation endpoint. The request should include an authorization scheme and the information regarding the material: description, display name, and file. 2. Tester should check the database entities. 3. Tester should check the return value of the request.
Expected Result	- A new material entity should be added to the database. - The new entity should include the given information. - The new entity should have a foreign key relationship with the course that the tester uploaded the material. - The new entity should have its *createdAt* field filled with the request time. (it can be later than the actual time due to the network connection) - The return value of the request should include all the information given as well as it should include the database ID of the newly created entity. - The return code should be successful.

Test ID	*F_BE_04*
Description	Tutors should be able to delete courses they created.
Priority	Medium
Dependency	F_BE_01
Precondition	- The tester should be a tutor. - The tester should have created a course. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the course deletion endpoint. The request should include an authorization scheme and the course ID. 2. Tester should check the database entities. 3. Tester should check the return value of the request.
Expected Result	- A course entity should be deleted from the database. - The reviews that belong to the course should be deleted from the database. - The material that belongs to the course should be deleted from the database. - The verification document that belongs to the course should be deleted from the database. - The return code should be successful.

Test ID	*F_BE_05*
Description	Students should be able to update their reviews related to courses they have booked for.
Priority	Low
Dependency	F_BE_02
Precondition	- The tester should be a student. - The tester should be registered for a course. - The tester should have reviewed the same course before. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the review update endpoint. The request should include the authorization scheme and the information regarding the updated review: comment, and rating. 2. Tester should check the database entities. 3. Tester should check the return value of the request.
Expected Result	- The information in the database should be changed according to the new information. - The updated entity should have its *updateAt* field filled with the request time. (it can be later than the actual time due to the network connection) - The return value of the request should include all the information of the previous entity with updated fields. - The return code should be successful.

Test ID	*F_BE_06*
Description	Students should be able to view supplementary materials from the courses they have bookings for.
Priority	Medium
Dependency	F_BE_03
Precondition	- The tester should be a student. - The tester should have bookings from a course. - The course should have materials. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the course get endpoint. The request should include an authorization scheme and the course ID. 2. Tester should note the material ids from the return value of the previous request. 3. Tester should make another HTTP request to the API. The request's URL should be the material get endpoint. The request should include the authorization scheme and the material ID. 4. Tester should check the return value of the request. 5. Tester should check the downloaded files.
Expected Result	- The return value of the first request should include a material ID. - The second request should automatically download the material with the given ID. - The downloaded material should be the same as the uploaded material by the tutor. (check the test F_BE_03) - The return code should be successful.

Test ID	*F_BE_07*
Description	Users should be able to create accounts in the system
Priority	High
Dependency	
Precondition	- The tester should have a tool to make HTTP requests.
Test Steps	1. Tester should connect to the relevant API endpoint 2. Tester should fill in the necessary fields 3. Testers should authenticate their accounts using a 3rd party confirmation (magic link / OAuth / secret)
Expected Result	- The user should be successfully registered in the system. - The return code should be successful.

Test ID	*F_BE_08*
Description	Users should be able to log in to the system.
Priority	High
Dependency	F_BE_07
Precondition	- The tester should have a tool to make HTTP requests. - The tester should have a registered and confirmed account in the system
Test Steps	1. Tester should connect to the relevant API endpoint 2. Tester should fill in the necessary fields
Expected Result	- Users should be successfully logged in to the system. - Users should get an ID token to make their request with. - The return code should be successful.

Test ID	*F_BE_09*
Description	Users should be only able to interact with the parts of the system that they are authorized to interact with.
Priority	High
Dependency	F_BE_08
Precondition	- The tester should have a tool to make HTTP requests. - The tester should have a registered and confirmed account in the system
Test Steps	1. Testers should make a request to an API endpoint to which they have access. 2. Testers should make a request to an API endpoint that they don't have access to.
Expected Result	- The first request should work as normal and the second should be rejected and not have any impact on the state of the backend/database - The resulting code of the first request should start with 200. - The resulting code of the second request should be 401, unauthorized. - The return code should be successful.

Test ID	*F_BE_10*
Description	Users should be able to view a list of courses.
Priority	High
Dependency	F_BE_01
Precondition	- The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key. - There should be at least one course in the database.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the course list get endpoint. The request should include an authorization scheme. 2. Tester should check the return value of the request.
Expected Result	- The return value should include a list of courses with their information. - The return code should be successful.

Test ID	*F_BE_11*
Description	Users should be able to view a single course by its ID.
Priority	High
Dependency	F_BE_01
Precondition	- The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key. - There should be at least one course in the database.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the course detail endpoint with a valid course ID. 2. Tester should check the return value of the request.
Expected Result	- The return value should include all the information of the course entity. - The return code should be successful.

Test ID	*N_BE_01*
Description	Users who do not have any bookings from a course should not be able to download materials of that course due to security.
Priority	High
Dependency	F_BE_03
Precondition	- The tester should be a student. - The tester should not have bookings from a course. - The course should have materials. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the course get endpoint. The request should include an authorization scheme and the course ID. 2. Tester should check the return value of the first request. 3. Tester should make another HTTP request to the API. The request's URL should be the material get endpoint. The request should include an authorization scheme and the material ID. (material ID should belong to a course that the tester does not have bookings for) 4. Tester should check the return value of the second request. 5. Tester should check the downloaded files.
Expected Result	- The return value of the first request should not include any material IDs. - The return value of the first request should include publicly available course information. - The return code of the first request should be successful. - The second request should not download the material with the given ID. - The return code of the second request should be unsuccessful.

Test ID	*N_BE_02*
Description	Users who do not belong to a particular meeting should not be able to see the link to the meeting for security reasons.
Priority	High
Dependency	F_BE_01
Precondition	- The tester should have an account. - There should be a booking between two other accounts. - The tester should have a tool to make HTTP requests. - The tester should be logged in, and therefore, should have an authorization key.
Test Steps	1. Tester should make an HTTP request to the API. The request's URL should be the booking get endpoint. The request should include an authorization scheme and the booking ID. (note that the user's account does not belong to this meeting) 2. Tester should check the return value of the request.
Expected Result	- The return value of the request should not include any URL for the meeting. - The return code of the first request should be unsuccessful.

Test ID	*N_BE_03*
Description	The system should be able to handle 500 concurrent meetings for the scalability requirement. From the backend point of view, it is enough to return meeting URLs for concurrent calls since the rest of the meeting is peer-to-peer.
Priority	Medium
Dependency	F_BE_01
Precondition	- The tester should have five hundred students and five hundred tutor accounts. - Each student account and each tutor account should have one booking. - Each booking time should be the same. - The tester should have a tool to make concurrent HTTP requests. - The tester should be logged in, and therefore, should have an authorization key for each account.
Test Steps	1. Tester should make concurrent HTTP requests to the API. The request's URL should be the booking get endpoint. The request should include authorization schemes and booking IDs. 2. Tester should check the return value of the requests.
Expected Result	- The return value of each request should include the unique meeting URL. - The status code of each request should be successful.

Test ID	*N_BE_04*
Description	The system should be able to handle 1.000 concurrent website usage for the scalability requirement. Therefore, the backend should return the main page requests (list of tutors) for 1.000 concurrent calls.
Priority	Medium
Dependency	
Precondition	- There should be at least eight tutors. (since the main page displays information about tutors) - The tester should have a tool to make concurrent HTTP requests.
Test Steps	1. Tester should make concurrent HTTP requests to the API. The request's URL should be the tutors list get endpoint. 2. Tester should check the return value of the requests.
Expected Result	- The return value of each request should include the list of tutors' information. - The status code of each request should be successful.

Test ID	*N_BE_05*
Description	The system should be able to handle 100.000 registered users for the scalability requirement. Therefore, the backend should be able to register 100.000 users.
Priority	Low
Dependency	F_BE_07
Precondition	- The tester should have a tool to make concurrent HTTP requests.
Test Steps	1. Tester should make HTTP requests to the API. The request's URL should be the register post endpoint. The request should include unique email addresses for each registration. 2. Tester should check the return value of the requests.
Expected Result	- All new users should be in the database as user entities. - The return value of each request should include the registration information. - The status code of each request should be successful.

Test ID	*F_MT_01*
Description	Whiteboard can be visible on the meeting page.
Priority	High
Dependency	
Precondition	- There must be a meeting between users.
Test Steps	1. The user opens the whiteboard tool. 2. The user changes different views.
Expected Result	- The whiteboard tool is visible. - The position of the whiteboard changes according to view.

Test ID	*F_MT_02*
Description	Drawings on the whiteboard should be correctly transformed into vector drawings.
Priority	High
Dependency	F_MT_01
Precondition	- There must be a meeting between users.
Test Steps	1. The user opens the whiteboard tool. 2. Pick a drawing tool. 3. Draw on a whiteboard. 4. Release the mouse button.
Expected Result	- Users should successfully select the drawing tool. - During drawing, lines should be raster graphics. - The drawing should be turned into a vector after finished.

Test ID	*F_MT_03*
Description	Users are able to erase the vectors drawn on the whiteboard.
Priority	Medium
Dependency	F_MT_02
Precondition	- There must be a meeting between users. - There are some drawings on the whiteboard.
Test Steps	1. The user selects the eraser tool. 2. Hold the mouse button and pass it through any drawing.
Expected Result	- The user should successfully select the eraser tool. - The vector component mouse passed through should be erased.

Test ID	*F_MT_04*
Description	Users can control the whiteboard by scaling, translating, or rotating it.
Priority	Low
Dependency	F_MT_02
Precondition	- There must be a meeting between users. - There are some drawings on the whiteboard.
Test Steps	1. The mouse should be brought to the borders of the whiteboard. 2. Mouse wheel is scrolled up and down. 3. Mouse is moved while holding the middle mouse button (mouse wheel). 4. Mouse is moved while holding the right mouse button.
Expected Result	- The whiteboard is scaled up and down. It can be observed by the change of sizes of drawings. - The whiteboard is rotated according to the middle point of the whiteboard. It can be observed by looking at rotations of drawings. - The whiteboard is moved according to the movement of the pointer. It can be observed by looking at the positions of drawings.

Test ID	*F_MT_05*
Description	Users can change the colors and sizes of drawing tools.
Priority	Low
Dependency	F_MT_02
Precondition	- There must be a meeting between users. - The whiteboard should be open. - The drawing tool is selected.
Test Steps	1. Draw a line on the whiteboard. 2. Draw two other lines by increasing and decreasing the scroll under the drawing tool. 3. Select another color from the color selector under the drawing tool. 4. Draw lines by selecting different colors.
Expected Result	- The first three lines must have different weights. Increasing the scroll under the drawing tool should also increase the weight of the drawing. - After selecting a new color, drawn lines should be in that color.

Test ID	*F_MT_06*
Description	Alterations made on one board should be visible to the other user in the meeting.
Priority	High
Dependency	F_MT_03
Precondition	- There must be a meeting between users. - The whiteboard should be open. - Tester must see two users' whiteboards.
Test Steps	1. The first user selects the drawing tool and draws two lines. 2. The first user selects the eraser tool and erases one of the lines. 3. The second user selects the drawing tool and draws two lines. 4. The second user selects the eraser tool and erases one of its lines and the line drawn by the first user. 5. The first user erases the line drawn by the second user.
Expected Result	- The first two lines should be visible to both users. - One line should be erased in both users. - Two additional lines should be visible to both users. - Two lines should be erased in both users. - The last line should be erased in both users.

Test ID	*F_MT_07*
Description	Whiteboard can be saved as an image file.
Priority	Medium
Dependency	F_MT_02
Precondition	- There must be a meeting between users. - The whiteboard should be open. - There must be drawings on the whiteboard.
Test Steps	1. Click on the save button. 2. Draw other things. 3. Click on erase all button. 4. Click save on the pop-up.
Expected Result	- Contents of the whiteboard must be saved as an image file. - Whiteboard with updated contents must be saved as an image file.

Test ID	*F_MT_08*
Description	After the meeting is finished, the whiteboard can be exported as a pdf file.
Priority	Medium
Dependency	F_MT_07
Precondition	- There must be a meeting between users. - The whiteboard must be erased several times.
Test Steps	1. The meeting is ended by the tutor. 2. Chose export as a pdf from the pop-up.
Expected Result	- The pdf of the whiteboard should be exported. Each page must be composed of the contents of the whiteboard when the save or erase all button is pressed.

Test ID	*F_MT_09*
Description	Chat can be visible on the meeting page.
Priority	Low
Dependency	
Precondition	- There must be a meeting between users.
Test Steps	1. The user opens the chat. 2. The user changes different views.
Expected Result	- The Chat is visible. - The position of the chat changes according to view.

Test ID	*F_MT_10*
Description	Users should be writing to the chat and communicating through the chat.
Priority	Low
Dependency	F_MT_09
Precondition	- There must be a meeting between users. - Chat should be opened. - Tester must see the chats of two users.
Test Steps	1. The first user writes on a chat. 2. Second user writes on a chat.
Expected Result	- Both users must successfully write on their chats. - Messages written by the other user should be visible on the other chat. - Order of messages on two chats must be “First User -> Second User”

Test ID	*F_MT_11*
Description	Chat transcript should be seen by the users in that meeting after the meeting.
Priority	Low
Dependency	F_MT_10
Precondition	- There must be a meeting between users. - Chat must be used throughout the meeting.
Test Steps	1. The meeting is finished. 2. Tutor selects download transcript from the pop-up.
Expected Result	- Messages on the transcript should be the same as the chat history. - There should be an indicator on each message that which time of the meeting is the message sent. - There should be an indicator of the real-world time of the message.

Test ID	*F_MT_12*
Description	Users can type on the whiteboard.
Priority	Low
Dependency	F_MT_01
Precondition	- There must be a meeting between users. - Whiteboard is open.
Test Steps	1. The user selects typing tool. 2. The user selects font sizes and types from this tool. 3. The user clicks anywhere on the whiteboard. 4. The user types something. 5. The user presses enter.
Expected Result	- The script user wrote should be visible on the whiteboard. - The other user should see the script on their whiteboard.

Test ID	*F_MT_13*
Description	Users can put images on the whiteboard.
Priority	Low
Dependency	F_MT_01
Precondition	- There must be a meeting between users. - Whiteboard is open.
Test Steps	1. The user selects the image addition tool. 2. The user selects an image from their local computer.
Expected Result	- When the users click on add image, there must be a pop-up to select an image from the local computer. - The image added to the whiteboard should be visible to both users.

Test ID	*F_MT_14*
Description	Users can scale, rotate and move images.
Priority	Low
Dependency	F_MT_13
Precondition	- There must be a meeting between users. - Whiteboard is open. - There must be an image added to the whiteboard.
Test Steps	1. The user left-clicks on the image without selecting any tool. 2. The user scales the image by holding the corners of the image. 3. The users move the image by clicking on the image and holding it. 4. The user brings the mouse button to a little outside of the corner of the image and when the pointer changes, holds the button and rotates it.
Expected Result	- The image must be selected when the user clicks on it. - The image must be scaled corresponding to the movement of the pointer. - The image must be moved according to the movement of the pointer. - The image must be rotated according to the movement of the pointer.

Test ID	*N_MT_01*
Description	Storing whiteboard information in the database should not take up too much space in order to guarantee the scalability of the system.
Priority	Low
Dependency	F_MT_08
Precondition	- There should be another system to increase the saves made to the system for test purposes.
Test Steps	1. Start with storing a daily expected number of new whiteboards in the system. 2. Increase the saves made in order to find how many days can save system work without intervention.
Expected Result	- The system should support the save function for daily expected usage. - System should work without intervention between maintenance.

Test ID	*N_MT_02*
Description	System should support concurrent vector drawing information. Each time any user draws something, this must be saved to the database so the system should support this.
Priority	Low
Dependency	F_MT_08
Precondition	- There should be another system to increase the number of vector drawings sent to the system by different meetings.
Test Steps	1. Tester should draw lines and erase lines from the whiteboard continuously. 2. Tester should increase the number of information sent to the database using the test system.
Expected Result	- The system should support at least 500 concurrent meeting information. - The system shouldn’t completely break down after an unexpected number of concurrent meetings.

Test ID	*N_MT_03*
Description	System should support concurrent chat information. Each time any user writes something on chat, this must be saved to the database so the system should support this.
Priority	Low
Dependency	F_MT_11
Precondition	- There should be another system to increase the number of chat messages sent to the system by different meetings.
Test Steps	1. Tester should write on chat continuously. 2. Tester should increase the number of information sent to the database using the test system.
Expected Result	- The system should support at least 500 concurrent meeting information. - The system shouldn't completely break down after an unexpected number of concurrent meetings.

Test ID	*F_VC_01*
Description	After joining a meeting, there should be a peer-to-peer connection that is opened between the users.
Priority	High
Dependency	
Precondition	- Both users should join the meeting via the user interface
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting
Expected Result	- The meeting component should create the connection between the peers, in such a case this will be in the logs.

Test ID	*F_VC_02*
Description	Users should be able to video stream to each other.
Priority	High
Dependency	F_VC_01
Precondition	- The camera of the users should be working.
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. Both users allow their browsers for sharing their camera footage
Expected Result	- User1 should be seeing User2's stream - User2 should be seeing User1's stream

Test ID	*F_VC_03*
Description	Users should be able to share screens with each other.
Priority	Medium
Dependency	F_VC_01
Precondition	
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. Users try sharing their screens via a button on the user interface
Expected Result	- The other user should be able to see the shared screen

Test ID	*F_VC_04*
Description	After leaving the meeting, a peer-to-peer connection should be terminated between the users.
Priority	High
Dependency	F_VC_01
Precondition	- Both users should join the meeting via the user interface
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. Connection is set 4. Both users leave the meeting
Expected Result	- The meeting component should terminate the connection between the peers, in such a case this will be in the logs.

Test ID	*F_VC_05*
Description	Users should be able to stop video streams.
Priority	High
Dependency	F_VC_02
Precondition	- Users must be already sharing video streams.
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. User1 allows his/her browsers for sharing the camera footage 4. User1 stops the video stream by clicking a button
Expected Result	- User2 should no longer be able to see User1’s stream

Test ID	*F_VC_06*
Description	Users should be able to stop sharing the screens.
Priority	Medium
Dependency	F_VC_03
Precondition	- Users must already be sharing the screens
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. User1 tries sharing the screen via a button on the user interface 4. User1 terminates the screen share
Expected Result	- User2 should no longer be able to see the shared screen

Test ID	*F_VC_07*
Description	Users should be able to enable their voice to be shared with the other user.
Priority	High
Dependency	F_VC_01
Precondition	- Users must already be in the meeting. - User1 must have hardware that accepts audio input for streaming
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. User1 tries to enable voice sharing via a button on the user interface
Expected Result	- User2 should be able to hear User1

Test ID	*F_VC_08*
Description	Users should be able to disable their voice to be shared with the other user.
Priority	High
Dependency	F_VC_07
Precondition	- Users must already be in the meeting. - User1 must have hardware that accepts audio input for streaming
Test Steps	1. User1 joins the meeting 2. User2 joins the meeting 3. User1 tries to enable voice sharing via a button on the user interface 4. User1 then tries to disable the voice sharing using the same button
Expected Result	- User2 should not be able to hear User1 even if User1 is talking

Test ID	*N_VC_01*
Description	System should be able to host 500 concurrent meetings
Priority	Medium
Dependency	
Precondition	- There should be two endpoints for meetings since it will be easy to do it on LAN, we should test it outside the LAN.
Test Steps	1. Tester opens concurrent mock meetings between two endpoints and starts video streaming
Expected Result	- Each meeting should be able to run without ease

Test ID	*F_UI_01*
Description	Tutors should be able to see new courses added by them on their profile page. Upon creating a new course from the “Create a Course” page, a new course must be visible on the tutor's profile page.
Priority	Medium
Dependency	F_BE_01
Precondition	- The tester should be logged in as a tutor. - The tester should have a valid document to verify the course s/he creates.
Test Steps	1. Tester should log in to his account. 2. Tester should create a course from the “Create a Course” page. 3. Tester should check his profile page
Expected Result	- A new course should be appearing on the tester's profile page.

Test ID	*F_UI_02*
Description	Students should be able to review the courses they have been registered for on the “Upcoming Meetings” page.
Priority	Medium
Dependency	F_BE_02
Precondition	- The tester should be logged in as a student. - The tester should be registered for a course.
Test Steps	1. Testers should attend a course. 2. Tester should try to make a comment on the attended course. 3. Tester should check the course page.
Expected Result	- A new comment should be appearing on the course page.

Test ID	*F_UI_03*
Description	Tutors should be able to upload supplementary material to courses they have created on the course page.
Priority	Medium
Dependency	F_BE_03
Precondition	- The tester should be logged in as a tutor. - The tester should have created a course.
Test Steps	1. Tester should navigate to his course from his profile page. 2. Tester should add new material from the course page. 3. Tester should check the materials from the course page.
Expected Result	- New material should be appearing on the course page.

Test ID	*F_UI_04*
Description	Tutors should be able to delete courses they created from the course page.
Priority	Medium
Dependency	F_BE_04
Precondition	- The tester should be logged in as a tutor. - The tester should have created a course.
Test Steps	1. Tester should navıgate to the course page. 2. Tester should clıck on the delete button using the UI. 3. Tester should check his profile page.
Expected Result	- Deleted course should not be appearing on the user's profile page.

Test ID	*F_UI_05*
Description	Students should be able to update their reviews related to courses they have booked for.
Priority	Low
Dependency	F_BE_05
Precondition	- The tester should be logged in as a student. - The tester should be registered for a course. - The tester should have reviewed the same course before.
Test Steps	1. Tester should navigate to the course information page. 2. Tester should change his comment. 3. Tester should refresh the page and check his comment.
Expected Result	- The content of the comment should be changed.

Test ID	*F_UI_06*
Description	Students should be able to view supplementary materials from the courses they have bookings for from the course page.
Priority	Medium
Dependency	F_BE_06
Precondition	- The tester should be logged in as a student. - The tester should have bookings from a course. - The course should have materials.
Test Steps	1. Tester should navigate to the course page he registered. 2. Tester should try to check the course material provided on the course page.
Expected Result	- Tester should be able to review the content of the course material.

Test ID	*F_UI_07*
Description	Users should be able to create their profiles in the system.
Priority	High
Dependency	F_BE_07
Precondition	Tester should have a valid email account.
Test Steps	1. Tester should provide the necessary information. 2. Tester should verify his email. 3. Tester should try to log in to the system.
Expected Result	- Tester should be successfully logged in to the system.

Test ID	*F_UI_08*
Description	Users should be able to change their names on the user settings page.
Priority	Low
Dependency	
Precondition	- Tester should be logged in to a valid account.
Test Steps	1. Tester should navigate to the settings page. 2. Tester should change their name via UI. 3. Tester should check his profile page.
Expected Result	- Updated user name should be appearing at the profile page.

Test ID	*F_UI_9*
Description	Users should be able to view a list of courses and filter them by subject and/or duration on the “Meetings” page.
Priority	High
Dependency	F_BE_09
Precondition	- Tester should log in on a valid account. - Tester should have attended or/and create meetings.
Test Steps	1. Tester should navigate to the “Meetings” page. 2. Tester should check his meetings.
Expected Result	- All of the meetings should be visible to the user.

Test ID	*N_UI_01*
Description	The system should not let users access other users’ settings by editing their user key.
Priority	Low
Dependency	
Precondition	- The tester should log in to the system. - Another user account should exist.
Test Steps	1. Tester should change his user key via the page source. 2. Tester should check the settings page.
Expected Result	- Tester should be accessing his settings page instead of the other user’s.

6. Consideration of Various Factors in Engineering Design

While designing the product, we made some judgments about different factors.

6.1. Economic Factors

While designing this product, it is intended to be a commercial one. By using this product, the ones with significant success can share their experiences with those in need and earn money. Besides, the owners of this product, us in this case, must get some proportion from this trade to maintain the product. Therefore, the design of the product should support this intention.

One of the main concerns was that even though tutors in this system had proven their successes, users may need more convincing to pay them to get their knowledge. So, we gave them a chance to convince users by giving them free courses. By doing this, they can increase their positive comments and easily convince other users for charged courses.

Another concern is that all the product users will trust us for their payment information and security. This leads us to two design concerns: securing all the users' payment information safely and ensuring that transactions between users happen as neatly as possible. For the security of payment information, information like credit card numbers shouldn't be accessible by others. For the integrity of money transactions between users, we have to decide what to do about situations like; the tutor does not come to the conference, the user/tutor wants to cancel the appointment, the conference can't be held because of technical issues, etc.

Level of effect: 10/10

6.2. Safety

Our product requires us to hold some information about users. For us, it is crucial to secure the privacy of the users' data since people have to trust the system to use the product. Since the payment information of the users is important to keep secure, it is essential to decide which payment processor system to be used in the application. We are also considering a two-factor authentication method for securing the personal information of the users. This will be an important decision to make between ease of use and safety.

Level of effect: 8/10

6.3. Social Factors

This product aims to be a bridge between tutors and the ones who want to gather knowledge from tutors' experiences. This makes this product a social product, leading us to think about social factors. By forming this bridge, both sides benefited. Tutors earn money from his/her success and help others to benefit from his/her previous experiences. However, for both sides to benefit from this product optimally, we decided to add comments to the tutors. By doing this, users can easily evaluate tutors according to their comments which makes the unpredictable side of the product more predictable.

Level of effect: 5/10

6.4. Welfare

Our application's main audience is successful university students who are looking for a source of income. To keep and grow our tutor users, we are considering several economic models for our application to ensure it will be both profitable enough for both our tutors and us. In this matter, we are considering how much cut-off we will set for course payment and whether we should use fixed course prices or let lecturers decide that. Thus, we made and are still thinking about some decisions on the welfare of our tutors while ensuring the maintainability of our product.

Level of effect: 4/10

6.5. Environmental Factors

Maintaining a server consumes energy which has a bad effect on the environment. Using peer-to-peer video streaming, Tutoryum solves this problem in addition to scalability and cost. Therefore, we only need to use the server to track which conferences are currently and previously held, information about the users, and course materials. This will decrease the server requirement by a noticeable amount.

Level of effect: 1/10

6.6. Public Health

We did not consider factors related to public health since they won't be applicable to our application.

Level of effect: 0/10

6.7. Global Factors

In the current state, our application will aim for tutors that are studying in Turkish universities. As a result, we did not consider any global factors for our application.

Level of effect: 0/10

6.8. Cultural Factors

Cultural factors also do not affect our engineering design since they won't be applicable to our product.

Level of effect: 0/10

	Economic	Safety	Social	Welfare	Environmental	Public Health	Global	Cultural
Level of effect	10	8	5	4	1	0	0	0

7. Teamwork Details

7.1. Contributing and functioning effectively on the team

Since Tutoryum was a rather big and complex project, proper teamwork should have been ensured. Ideally, each person had responsibilities and specific tasks. We were a team of five people, and if each person contributed to the project equally, the project could be finished in approximately six months. Each person should have completed their tasks without delay since delay might have caused others to be delayed.

To assign tasks to five people, we first assigned each person to a specific part of the project. The project had three main components: Backend Component, Frontend Component, and Meeting Component. Each member contributed to the following components:

Barış Ogün Yörük: Backend Component
Halil Özgür Demir: Meeting Component
Mustafa Çağrı Durgut: Backend Component
Oğuzhan Özçelik: Meeting Component
Yusuf Miraç Uyar: Frontend Component

We also divided these three components into subparts so that project development could be highly parallelized. Thanks to this, the task creation step was easier, and we could make up the big picture out of smaller pictures. This subpart logic also eased the project management and testing of individual parts, so we had a higher level of confidence that everything on the demo was working correctly. The project had seven subparts: API, Database, DevOps, Meeting Tools, UI/UX Design, UI/UX Implementation, and Video Connection. Each member contributed to the following subparts:

Barış Ogün Yörük: API, Database, DevOps
Halil Özgür Demir: Meeting Tools, Video Connection
Mustafa Çağrı Durgut: API, Database
Oğuzhan Özçelik: Meeting Tools, Video Connection
Yusuf Miraç Uyar: UI/UX Design, UI/UX Implementation

7.2. Helping create a collaborative and inclusive environment

To create a collaborative and inclusive environment, we used several tools. These tools helped us to assign tasks to everyone in a fair manner. Since these tools had auditing features, we were able to keep track of every member included in the project management. Also, these tools had collaboration features so that we could communicate on tasks and work on them if needed. Now, tools that were used to ensure the creation of a collaborative and inclusive environment will be discussed briefly.

Asana: As a task management application, we used Asana [1]. Thanks to Asana, we had a SaaS that enabled us to divide tasks and organize the project's development.

- **Creating a Collaborative Environment:** Since all team members could see what was going on with the tasks, Asana helped us to create a collaborative environment. On tasks, we were able to add comments and raise questions. This helped us to puzzle our brains on the tasks and come up with creative ideas. Moreover, we were able to change the priority of tasks so that we could agree on which tasks should be done first.
- **Inclusive Environment:** Since each task was assignable to a team member, we were able to observe who got overwhelmed and who was free. This enabled us to be inclusive regarding each team member.

GitHub: We used GitHub as a remote repository service [2]. It enabled us to share code and manage our project's codebase. We used several features of GitHub and Git to manage our codebase properly with contemporary techniques. Moreover, GitHub was also used for auditing purposes. It was visible how many commits each person created, how many pull requests each person opened, and how many lines of code each person added. However, it should be noted that each task had varying difficulties. Some required more research than others, and some included many lines, but they were just boilerplate statements. We did not treat these indicators as the only indicator for ensuring equal teamwork.

- **Creating a Collaborative Environment:** Thanks to the pull request feature of GitHub, each member was able to observe the latest changes and see how each task was implemented. Moreover, each member could add comments and reviews to the pull requests so that if one point was not clear on a commit or one feature was not implemented properly, we were able to raise our heads.
- **Inclusive Environment:** Since we were able to keep track of how many commits each member contributed, it was easy to detect who was not included much in the effort. This enabled us to be inclusive of all the team members.

Discord: We used Discord to communicate with the team and organize documents and artifacts [3]. Thanks to Discord, we had transparent and open conversations with each other so that ambiguities in the project or different proposals were resolved as soon as possible. Additionally, having dedicated artifacts and links channels saved us time.

- **Creating a Collaborative Environment:** We were able to discuss our ideas and proposals on Discord. While writing the documents, we sent the artifacts over Discord so that we could give feedback to each other. This helped us to double-check what we wrote and fix it according to the messages.
- **Inclusive Environment:** With regular and emergency meetings, we met over Discord. This helped us to include every team member in the effort. We discussed and determined the project path together.

Each member used the following applications to create a collaborative and inclusive environment:

Barış Ogün Yörük: Asana, Discord, GitHub
Halil Özgür Demir: Asana, Discord, GitHub
Mustafa Çağrı Durgut: Asana, Discord, GitHub
Oğuzhan Özçelik: Asana, Discord, GitHub
Yusuf Miraç Uyar: Asana, Discord, GitHub

7.3. Taking a lead role and sharing leadership on the team

To ensure that everyone takes the lead role and shares leadership on the team, we assigned each member to one or two parts of the project as a leader (in a mutually exclusive, collectively exhaustive manner). This approach ensured that each part of the project was done correctly and went synchronously. Each member is assigned to the following parts to be a leader:

Barış Ogün Yörük: API, DevOps
Halil Özgür Demir: Meeting Tools
Mustafa Çağrı Durgut: Database
Oğuzhan Özçelik: Video Connection
Yusuf Miraç Uyar: UI/UX Design, UI/UX Implementation

Leaders' responsibilities were as the following:

1. Keep track of the tasks regarding their parts.
2. Setting up regular meetings with members who are also working on the same part.
3. Reporting the status of their parts on the weekly meetings.
4. Preparing documents regarding their parts on the deliverables.

Note that each member successfully carried out their leadership role with respect to the assignment above.

7.4. Fulfilled tasks

In this section, fulfilled tasks by each member will be given as a list. Note that tasks had varying difficulties, and the number of tasks completed by each member does not necessarily mean they spent more time.

Task name	**Task component**	**Completed by**
Add whiteboard service	API	Barış Ogün Yörük
Add material service	API	Barış Ogün Yörük
Add review service	API	Barış Ogün Yörük
Add course service	API	Barış Ogün Yörük
Add file service	API	Barış Ogün Yörük
Add pdf service	API	Barış Ogün Yörük
Setup Circle CI	DevOps	Barış Ogün Yörük
Prepare ‘activity diagrams’	Document	Barış Ogün Yörük
Prepare ‘state diagrams’	Document	Barış Ogün Yörük
Prepare ‘planning for new knowledge’	Document	Barış Ogün Yörük
Prepare ‘ensuring proper teamwork’	Document	Barış Ogün Yörük
Add simple whiteboard	Meeting Tools	Halil Özgür Demir
Make the whiteboard online interactable	Meeting Tools	Halil Özgür Demir
Make UI ready for demo	Meeting Tools	Halil Özgür Demir
Prepare ‘sequence diagrams’	Document	Halil Özgür Demir
Prepare ‘professional responsibilities’	Document	Halil Özgür Demir
Prepare ‘risks and alternatives’	Document	Halil Özgür Demir
Prepare ‘consideration of various factors’	Document	Halil Özgür Demir
Add authentication service	API	Mustafa Çağrı Durgut
Add database models	Database	Mustafa Çağrı Durgut
Prepare ‘scenarios’	Document	Mustafa Çağrı Durgut
Prepare ‘introduction’	Document	Mustafa Çağrı Durgut
Prepare ‘requirements’	Document	Mustafa Çağrı Durgut
Prepare ‘overview’	Document	Mustafa Çağrı Durgut

Prepare ‘use cases’	Document	Mustafa Çağrı Durgut
Initialize meeting component	Meeting Tools	Oğuzhan Özçelik
Set two connections on LAN	Video Connection	Oğuzhan Özçelik
Add peer-to-peer streaming	Video Connection	Oğuzhan Özçelik
Add screen sharing	Video Connection	Oğuzhan Özçelik
Prepare ‘class model’ for client-side	Document	Oğuzhan Özçelik
Prepare ‘class model’ for server-side	Document	Oğuzhan Özçelik
Design website	Document	Yusuf Miraç Uyar
Prepare ‘project plan’	Document	Yusuf Miraç Uyar
Implement home page	Frontend	Yusuf Miraç Uyar
Implement tutors page	Frontend	Yusuf Miraç Uyar
Implement profile page	Frontend	Yusuf Miraç Uyar
Implement schedule page	Frontend	Yusuf Miraç Uyar
Implement meetings page	Frontend	Yusuf Miraç Uyar

8. Glossary

API: Application Programming Interface

CSS: Cascading Style Sheets

HTML: HyperText Markup Language

HTTP: Hypertext Transfer Protocol

LAN: Local Area Network

SaaS: Software as a Service

SQL: Structured Query Language

UI: User Interface

UX: User experience

XMPP: Extensible Messaging and Presence Protocol

9. References

[1] “We Deliver Happiness”. https://explore.zoom.us/en/about/ [Accessed: Feb 25, 2023]

[2] “What is Microsoft Teams”. https://www.microsoft.com/en-us/microsoft-teams/group-chat-software [Accessed: Feb 25, 2023]

[3] “Secure Video Conferencing for Everyone”. https://meet.google.com/ [Accessed: Feb 25, 2023]

[4] “We're in Business to Help You thrive”. https://asana.com/company [Accessed: Mar 2, 2022]

[5] “Let’s Build from Here”. https://github.com/about [Accessed: Mar 2, 2022]

[6] “Create Space for Everyone to Find Belonging”. https://discord.com/company [Accessed: Mar 2, 2022]

[7] A study of Zoom's Video Conferencing Architecture & System Design. https://www.cometchat.com/blog/zoom-video-technology-architecture [Accessed March 13, 2023]