Bilkent University

Department of Computer Engineering

Senior Design Project

Tutoryum

Final 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, MySQL for relational databases, and a RESTful API for communication between the front and backends.

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 features an intuitive user interface and straightforward design that guides users to the information they need. Tutoryum creates an environment where users feel comfortable and confident, enhancing their overall learning experience.

Performance: Tutoryum prioritizes performance to ensure that the platform operates efficiently and effectively. The video conferencing technology is optimized for speed and reliability to ensure a smooth and uninterrupted experience for both tutors and students. The platform also features 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 has robust backup systems in place to ensure that users can access their materials and participate in sessions at all times. Tutoryum also ensures that all user data is secure and protected.

Marketability: Tutoryum creates a platform that is appealing and competitive in the market. The platform has been designed with user experience in mind, providing features that are relevant and useful to both tutors and students. Tutoryum creates a unique and attractive platform that stands out in the market and attracts users.

Extendibility: Tutoryum creates a platform that is easily extendable and can adapt to the evolving needs of its users. The platform has been designed to allow for the integration of new features and functionality as needed. Tutoryum creates 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 has robust security protocols in place to protect user data and ensure that all interactions on the platform are secure. Tutoryum creates an environment where users can feel safe and confident.

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

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

Flexibility: Tutoryum creates a platform that is flexible and can adapt to the unique needs of its users. The platform has been 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 creates a platform that is modular and easy to work with. The platform has been designed with a modular architecture, making it easy to add, remove, or modify components as needed. Tutoryum creates 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.

MySQL: A 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.

1.4. Overview

Tutoryum is a web-based application that provides 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 has several features, such as an interactive whiteboard, slide sharing, and optimized whiteboard recording and storage from the whiteboard at the end of the session. Additionally, the platform provides live streaming and live interaction with the lecturer and the student, making the experience more engaging and interactive.

The platform also offers 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 utilizes WebRTC technology for setting up video conferencing and interactive whiteboard features. Relational databases MySQL is used for data storage. The platform also uses a RESTful API to communicate the backend with the frontend.

Tutoryum is 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. Requirements Details

2.1. Functional Requirements

- Users can register to the system.
- Students can see the expertise of an instructor from their profile.
- Students can reserve times from instructors.
- Students and tutors can meet at peer-to-peer video conferences.
- Users can use educational tools interactively.
- Educational tools such as interactive whiteboards can be saved and used later by the tutor.
- An automated system creates a whiteboard show from the changing status of the whiteboard throughout the session.
- Students can pay for the courses.
- Tutors can get their pay from the payment.
- Tutors can create courses.
- Tutors can add course materials to their courses.

2.2. Non-Functional Requirements

- **Security:** Tutors' documents are protected. Payments are also safe.
- **Scalability:** The system is able to handle 500 concurrent meetings.
- **Extensibility:** The system is suitable for new features to be added.
- **Usability:** The application is simple enough to access broad communities.

2.3. Pseudo Requirements

2.3.1 Implementation Constraints

- The project is implemented as a website on the Internet.
- Git is used for version control of the codebase. [1]
- GitHub is used for cooperatively working on the project's repository. [2]
- Asana is used for task management. [3]
- The project uses a third-party library to implement video conferencing logic.

2.3.2 Economic Constraints

- Since many users may use the application, multiple servers are needed to scale the application. For development purposes, a free or relatively cheap server was enough. In the case of production, using one of the Cloud Providers might be needed. In that case, the price will be related to the number of users.
- For all the development tools, free versions are used.
- We have bought a domain name to publish our project.

2.3.3 Ethical Constraints

- Tutors' information is viewed without leaking too much personal information.
- Documents of tutors are confidential, so they are securely stored.
- Contents of the meetings are encrypted so that adversaries cannot watch them.
- The payment system is safe, so we don't have a leaking problem.

2.3.4 Sustainability Constraints

- The application will be reviewed regularly. The purpose of it is to keep technologies and tools used in the project updated, enhance performance, and fix unwanted behaviors and bugs.
- New features regarding education tools might be added.
- In case of a surprising number of users, the project has a scalability logic so that the application will keep running without intervention.

2.3.5 Social Constraints

- There is a request & complaint email for solving disputes between students and teachers. For example, teachers can be different people from how they introduced them to the system.

2.3.6 Technological Constraints

- Users need a microphone and a camera.
- Users need a sufficient internet connection for the best experience since it is necessary for both interactive real-time education tools and video conferences.

2.3.7 Time Constraints

- All the basic functionalities are finished before the deadline, such as login & register logic of students and teachers, viewing teachers, setting up & canceling meetings, and uploading documents.

2.3.8 Ethical Constraints

- Users are allowed to report any inappropriate content published by the tutors.
- The payment system included in the application follows each online payment-related law of the countries in which the payment system is implemented.
- Users are informed and ask for permission for each user's collected and stored data.
- Any personal information of the users that are stored are stored securely.
- Users have an option to report others they are involved with in a lecture session.

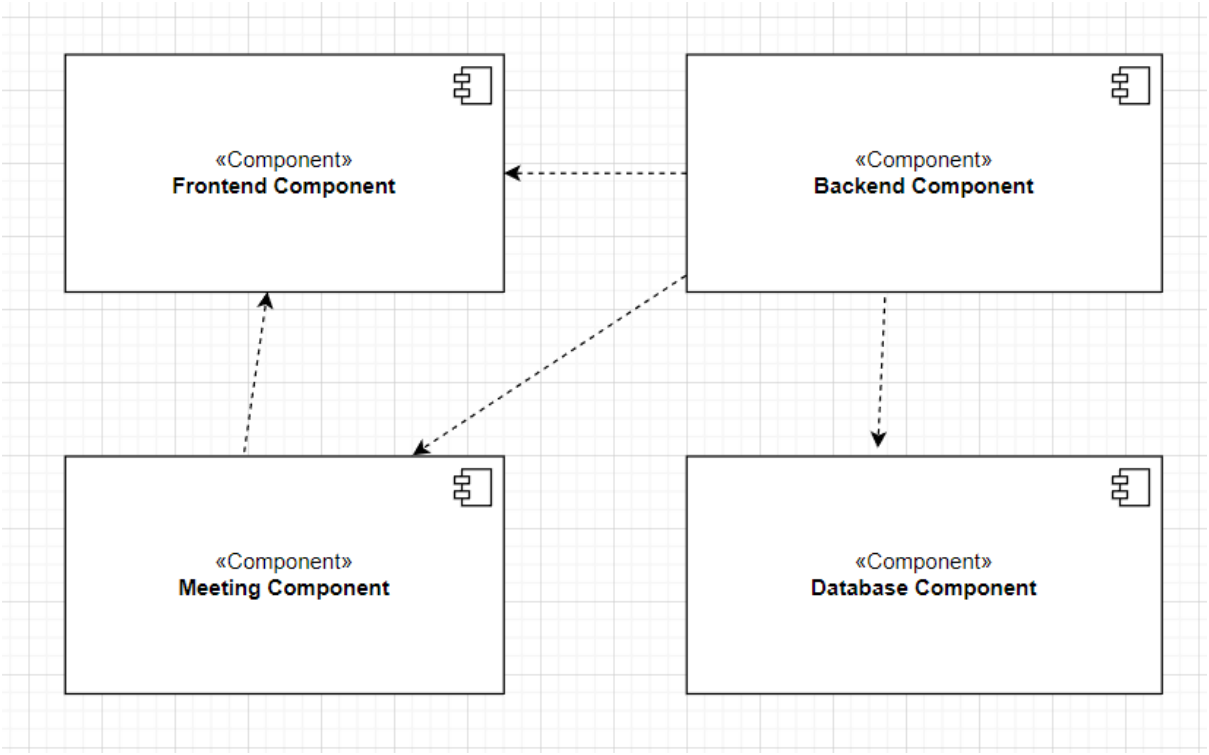
3. Final Architecture and Design Details

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 is open to adding new features seemingly for the future since the video communication baseline is 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 have been implemented in different languages and are independent of each other.
2. It was easier to share the tasks in the team environment since teammates focused on their parts and were responsible for their components. That way, when someone needed to ask questions, he could ask the responsible

teammate, and that teammate knew the details because of the partition of workflow.
3. It was easier to add new features since all the components work independently, and this provided 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 is. Users use the system here, and all the calculations are abstracted to provide a better experience. The backend component is responsible for handling API requests, handling the database system, and creating connections between users for video communication using the meeting component. The database component is 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

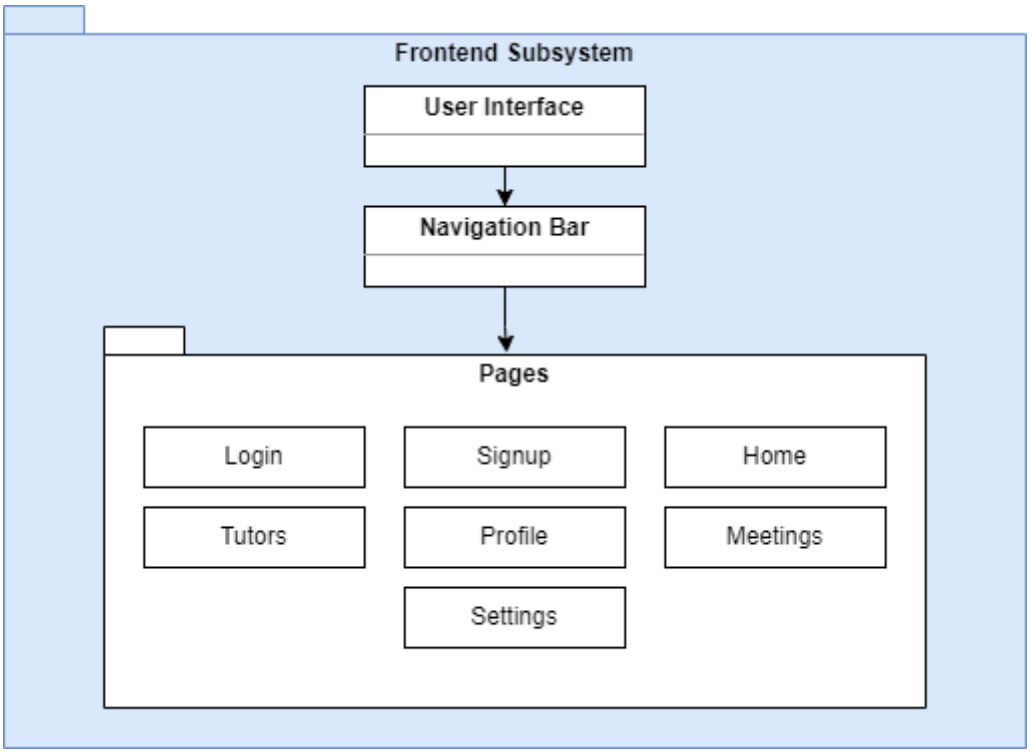
The system does not need a huge dedicated database to store the necessary data. Since our system works peer-to-peer, video communication is not a burden on the server or the database system. The database system is responsible for storing the user data and meeting information; the data users have stored 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 MySQL, and it needs no specific optimization because user data is not huge. Whiteboard screenshots are 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 logs on to the system using their credentials, and passwords are not stored in the database directly. SHA values of the password are stored, and authentication is done using these values. Also, the system uses authentication for all server requests, and each endpoint is protected for its users. In other words, a person is not able to make API requests to fetch the data of other users.

4. Development/ Implementation Details

4.1. Frontend Subsystem



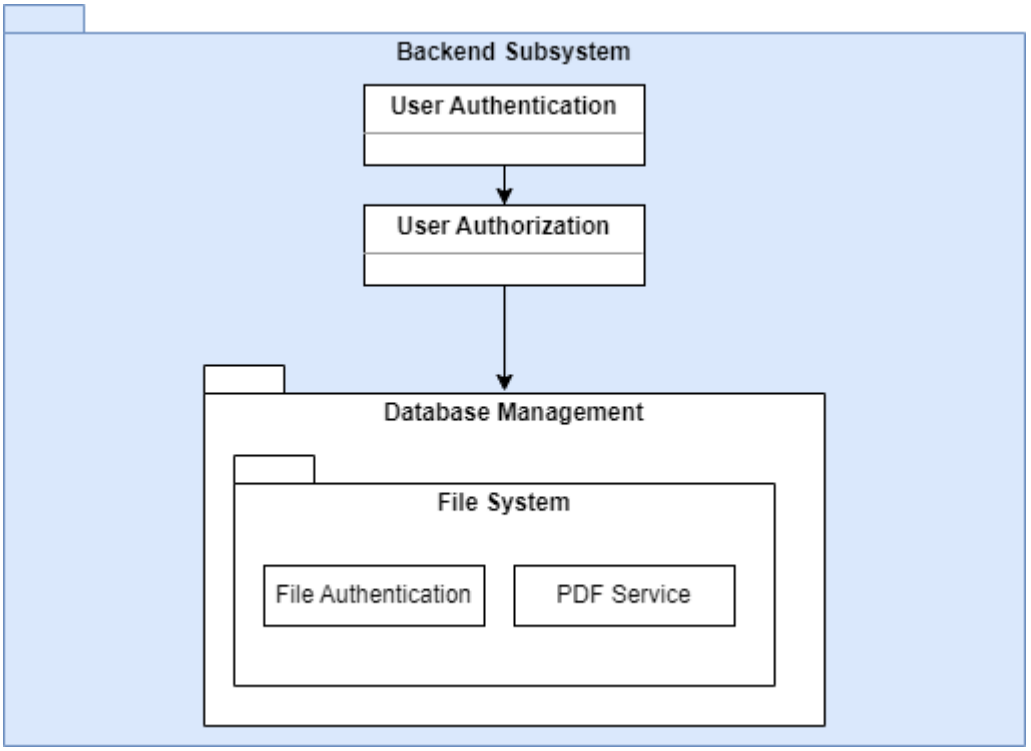
The Frontend subsystem is responsible for providing users with an intuitive and user-friendly interface to navigate the website's different pages. The website includes various pages, such as Login, Signup, Settings, Home, Tutors, Profile, and Meetings pages, which serve different functions.

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



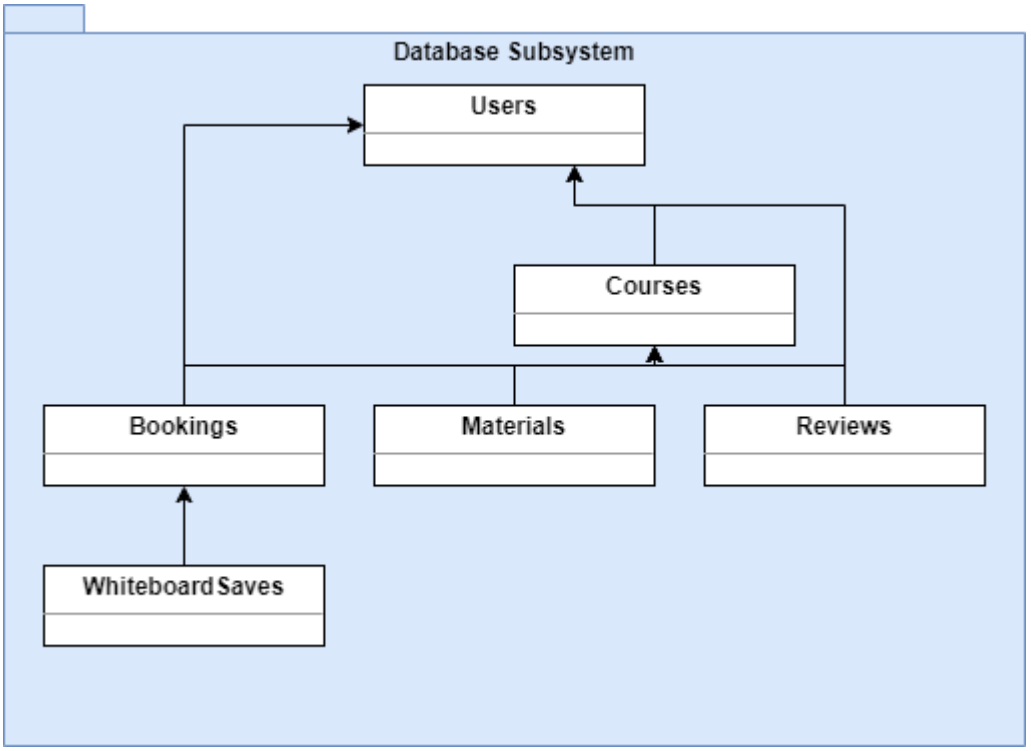
The Backend Subsystem is responsible for managing user accounts, authentication, and authorization. Moreover, it is also used for database management. One of the primary functions is establishing new connections between two users. When the meeting time comes, the connection between the two users must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

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 a whiteboard show from the saved movements.

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

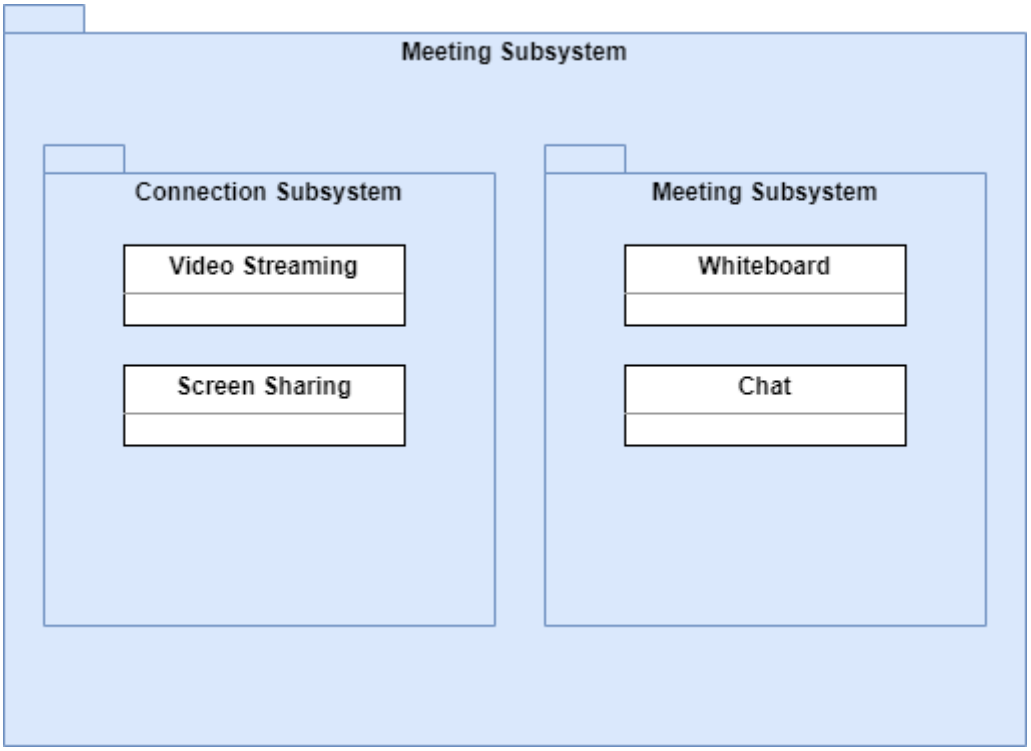


The Database Subsystem is responsible for the storage and retrieval of data related to users, their profiles, their preferences, and their interactions with the overall system. This subsystem provides a reliable, scalable, and secure way to store and access data.

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 whose paths 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 whiteboard shows after 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 pdf 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. This tool is implemented using PIXI.js library. 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 history of that meeting.

5. Test Cases and Results

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	~Pass
Disclaimer	*Create the course* part is done. *Verify the course* feature is dropped.

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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 where 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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Pass

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.
Tester	Barış Ogün Yörük
Test Date	18 May 2023
Status	Fail
Notes	Not applicable due to the registration flow.

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.
Tester	Halil Özgür Demir
Test Date	12 April 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	12 April 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	15 April 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	10 May 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	10 May 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	10 May 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	14 May 2023
Status	~Pass
Disclaimer	Saving as an image file is dropped. It is saved as a text 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.
Tester	Halil Özgür Demir
Test Date	14 May 2023
Status	~Pass
Disclaimer	Saving as a pdf file is dropped. It is saved as a text file.

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.
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	Fail
Notes	Chat is dropped.

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”
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	Fail
Notes	Chat is dropped.

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.
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	Fail
Notes	Chat is dropped.

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.
Tester	Halil Özgür Demir
Test Date	10 May 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	14 May 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	~Pass
Disclaimer	Rotation is dropped. Scale and translation is added.

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.
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	Pass

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.
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	~Pass
Disclaimer	Only at the end of the meeting, the whiteboard is saved.

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.
Tester	Halil Özgür Demir
Test Date	19 May 2023
Status	Fail
Notes	Chat is dropped.

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.
Tester	Oğuzhan Özçelik
Test Date	12 April 2023
Status	Pass

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
Tester	Oğuzhan Özçelik
Test Date	12 April 2023
Status	Pass

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
Tester	Oğuzhan Özçelik
Test Date	14 May 2023
Status	Pass

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.
Tester	Oğuzhan Özçelik
Test Date	14 May 2023
Status	Pass

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
Tester	Oğuzhan Özçelik
Test Date	19 May 2023
Status	Pass

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
Tester	Oğuzhan Özçelik
Test Date	19 May 2023
Status	Pass

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
Tester	Oğuzhan Özçelik
Test Date	19 May 2023
Status	Pass

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
Tester	Oğuzhan Özçelik
Test Date	19 May 2023
Status	Fail

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
Tester	All team
Test Date	19 May 2023
Status	Pass

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.
Tester	Mustafa Cagri Durgut
Test Date	19 May 2023
Status	Pass

Test ID	*F_UI_02*
Description	Students should be able to review the courses they have been registered for on the “Passed 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.
Tester	Mustafa Cagri Durgut
Test Date	19 May 2023
Status	~Pass
Disclaimer	UI needs improvements.

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.
Tester	Mustafa Cagri Durgut
Test Date	19 May 2023
Status	Fail

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.
Tester	Mustafa Cagri Durgut
Test Date	19 May 2023
Status	Fail

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.
Tester	Yusuf Miraç Uyar
Test Date	19 May 2023
Status	Fail

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.
Tester	Yusuf Miraç Uyar
Test Date	19 May 2023
Status	Pass

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.
Tester	Yusuf Miraç Uyar
Test Date	19 May 2023
Status	Pass

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.
Tester	Yusuf Miraç Uyar
Test Date	19 May 2023
Status	Pass

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.
Tester	Yusuf Miraç Uyar
Test Date	19 May 2023
Status	Fail
Notes	Filter will be implemented.

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.
Tester	Yusuf Miraç Uyar
Test Date	19 May 2023
Status	Pass

6. Maintenance Plan and Details

6.1. Maintenance of Development Environments

When it comes to handling the development and production environments for our platform, we lean on several dependency management and development tools.

6.1.1 Frontend Maintenance

For our frontend, we primarily use npm and vite.

npm (Node Package Manager) is our chosen tool for managing JavaScript dependencies. It allows us to specify and automatically manage all the libraries our frontend code relies on. By defining these dependencies, we can ensure a consistent environment, making our application more predictable and easier to test and debug.

Vite is a modern frontend build tool we utilize. It allows for faster and leaner development workflows. With features like hot-module replacement, it improves the efficiency of our frontend development process.

6.1.2 Backend Maintenance

For the backend, we use poetry and gunicorn.

Poetry is a tool for dependency management in Python. It allows us to handle Python package installation and ensure that the correct versions of each package are used across our development, testing, and production environments. It keeps our Python environment consistent, which is crucial for maintaining application reliability.

Gunicorn is a Python WSGI HTTP Server for UNIX. We use it as a middleman between our web server and our web applications. It allows us to run our Python web applications in a production environment, taking care of things like process management and load balancing.

6.1.3 Meeting Maintenance

For the maintenance of our meeting components, we rely primarily on Express.js and Yarn.

Express.js is a fast, unopinionated, and minimalist web framework for Node.js. We use it to structure our web application and handle HTTP requests for our meeting components. It's an efficient and flexible tool, allowing us to write handlers for requests with different HTTP verbs at different URL paths. This simplifies the routing and handling of requests, helping us manage the meeting functionality smoothly.

Yarn is a fast, reliable, and secure JavaScript dependency management tool. We use Yarn to manage our project’s dependencies. It helps us ensure that we install the correct versions of the libraries/packages used in our meeting components, allowing us consistent installations across all environments. This way, we can prevent possible bugs caused by slight differences in package versions.

6.2. Proactive Maintenance

We are making our database better and optimized by regularly cleaning out old data we don't need, using indexes to find data faster, and updating the statistics used by the system that decides how to get data the quickest way possible. This keeps our database running smoothly and efficiently.

We are keeping our system software up to date. We regularly install the latest patches for our operating system, updates for the libraries we use in our software, and keep our software dependencies current. This helps our system run at its best and fixes any security holes that might have been found.

We regularly making backups of our databases, user data, and how we've set up our systems, or "system configurations." We also check these backups to make sure they work and practice using them so we're ready if something bad happens and we need to recover our data.

6.3. Reactive Maintenance

We are in the process of setting up a system to handle sudden problems, which is called an incident management process. The idea is to quickly and clearly communicate with all parties involved, including stakeholders and users, whenever an issue pops up. Once we've taken care of the problem, we plan to carry out a deep-dive investigation, known as a root cause analysis (RCA). This analysis will help us understand why the issue happened and how we can prevent similar problems from happening in the future. Right now, we're doing a basic version of this process, but we're planning to enhance it soon.

6.4. Documentation

Ensure proper documentation for all aspects of the system. This should include system architecture, design decisions, operational procedures, incident management procedures, and common troubleshooting steps.

6.5. CI/CD

GitHub Actions is used for Continuous Integration and Continuous Deployment (CI/CD) for all three of our components. We created a workflow to automatically build and test code on every commit or pull request.We automatically deploy changes to the staging environment for further testing. After successful testing, changes can be deployed to the production environment.Rollback mechanisms should be in place to restore the previous stable version if needed.

6.6. Security Maintenance

For securing our users data and systems. We monitor our systems for breach. In future we plan to hire a small team to research and investigate further.

6.7. Security Monitoring

All of Tutoryum’s components, frontend, backend and the meeting component are registered with monitoring tools. This allows us to monitor the system-wide usage, resource utilization and error rates. Therefore we can easily track the issues at the system and take immediate action if needed.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

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

7.1.1. 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

7.1.2. 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

7.1.3. 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

7.1.4. 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 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

7.1.5. 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

7.1.6. 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

7.1.7. 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

7.1.8. 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

7.2. Ethics and Professional Responsibilities

Tutoryum is a bridge between people, and these people may have different ethnicities and sexualities. Throughout the development of the Tutorium, we always kept this in mind. Therefore, we are proud to say that our application accomplished the inclusion at its best. Here some ethical and professional responsibilities that we implemented in our solution:

- While accepting tutors to the system, it does not consider any of these characteristics of people.
- If one student who attends the tutor’s conferences insults the tutor's characteristics or vice versa, Tutoryum has a report section. After a proper inspection, the reported users may be banned from the system.
- The reporting system can also be used to resolve payment issues between tutors and users.
- Another responsibility is the security of the personal information of tutors. As tutors will be uploading their formal exam documents to the system, the system securely stores them and deletes them when they are no longer needed.
- Content publications of tutors are checked to verify the published content belongs to them.
- There is an option for users to report any inappropriate content published by the tutors.
- The payment system included in the application follows each online payment-related law of the countries in which the payment system is implemented.
- Users are informed and asked for permission for the data collected and stored.

- Any personal information of the users that are stored are secured with best practices.

7.3. Teamwork Details

7.3.1. 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

7.3.2. Helping creating 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 [3]. 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 [4]. 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

7.3.3. 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.

7.3.4. Meeting objectives

In this section, we will discuss the objectives we set in the analysis report and at which level these objectives are met:

1. Until the third week, we will finish the implementation of HTML and CSS.
2. Until the third week, we will initialize the backend and database parts.

3. Until the third week, we will implement and test the viability of the streaming API primitively.
4. Until the third week, we will implement and test the viability of the interactive whiteboard primitively.
5. Until the first demo, we will finish the implementation of the front end but leave the backend connection out.
6. Until the first demo, we will finish the implementation of most of the backend services but leave the edge cases and infrastructure out.
7. Until the first demo, we will finish the video streaming so that two connections on the local host can be established.
8. Until the first demo, we will finish the interactive whiteboard so that primitive drawings can be shared between users.
9. Until the second demo, we will enhance the UI/UX of the front end, add a user manual, and finish the backend connection.
10. Until the second demo, we will finish the implementation of the backend services so that all the features in the requirements are satisfied.
11. Until the second demo, we will finish the implementation of the video connection so that it can work over the Internet. Also, we will add additional features such as mute, screen sharing, and chat.
12. Until the second demo, we will finish the implementation of the interactive whiteboard so that screenshots can be saved, different types of pens can be used, and the size of the board can be changed.
13. Until the second demo, we will serve every component on the Internet via infrastructure solutions.
14. Until the second demo, we will conduct end-to-end tests to make sure that everything works properly.

We have successfully achieved all of our objectives up to the first demo (element 8). While we have made significant improvements to the UI, it is not yet fully completed. However, all the backend services are currently functioning properly. We have published our meeting component on the internet and added the necessary functionality to connect peers. However, there are still some issues with certain parts, as we are currently relying on manual peer ID for peer-to-peer connections. The whiteboard functionalities have been fully implemented, with only a few minor bug fixes and UI improvements remaining. All components are accessible on the internet. We have now started conducting end-to-end tests.

7.4. New Knowledge Acquired and Applied

Coverage of the Tutoryum project is extensive and requires knowledge about several tools and technologies. These tools and technologies are as follows:

- HTML
- CSS
- JavaScript
- Semantic UI

- React
- FastAPI
- MySQL
- Docker
- WebRTC
- Computer Networks
- Software Project Management
- PIXI.js

Since each team member is assigned different parts of the project, they are required to know each tool used in their parts. From previous experiences, each team member knows some tools. While dividing team members into each part, prior knowledge was considered. However, of course, there are some missing parts, and each team member is required to learn these missing parts before starting the implementation.

Some team members have known Computer Networks and Software Project Management since they took these courses in previous semesters. These members will be responsible explicitly for these concepts. However, if it is required for another team member who does not know these to learn them, people knowing these will help them out.

The rest of the tools and technologies can be learned online. To learn these, the following approaches will be used:

- Official Documentations
    - Tutorials: These help boost new learners. They usually tend to be precise and direct to the point. They help to give learners an overview of the tool. However, most of the time, they are insufficient.
    - Reference Pages: These include all the information regarding functions, classes, etc., inside the new tool. Of course, it is impractical to sit and read all these pages, but new learners can utilize them when they are stuck on a specific part of the tool.

- Online Courses: These resources help you learn new technologies in depth. Usually, these courses include all the required knowledge to develop a specific tool. Moreover, they often have exercises in coding along while learning, which is extremely useful to reinforce what is learned. Each team member who does not know a tool was suggested to follow one of the online courses so that before starting the project, they had enough information about the tool to develop with it

8. Conclusion and Future Work

In developing and maintaining Tutoryum, we've created a platform that enhances the educational experience by fostering a rich and interactive learning environment. Through diligent system monitoring, maintenance, and security measures, we've built a stable, scalable and secure platform that users can trust.

Our maintenance plan's implementation—covering database optimization, regular system updates, backup and recovery, incident management, and security practices—has played a critical role in ensuring the platform's smooth operation. Our use of tools like npm, vite, poetry, gunicorn, express, and yarn has helped us keep our development and production environments consistent and manageable.

8.1 Future Work

Looking ahead, we plan to further enhance and expand our platform. Some key areas we aim to focus on include:

Frontend improvements: We follow the developments in the React ecosystem and want to push further with the React Server Components. Currently Tutoryum is a Single Page Application which does not allow search engines to crawl necessary data. We can improve this by using RSC or frameworks like Next.js. Also this approach will make the repo monolith which will reduce the need for the REST API.

Redis and other Caching Alternatives: We plan to improve our UX by reducing the db responses with using caching or another key value database alternative. This will improve the

Incident Management Process: We aim to advance our incident management process. This includes refining our root cause analysis (RCA) procedures to prevent recurrence of issues and enhancing communication protocols during incident resolution.

Security: We intend to fortify our security measures. This involves expanding our security team, conducting regular security audits, and investing in advanced security tools. We will also strengthen access controls and implement role-based access control (RBAC) to further secure user data.

System Enhancements: We will continue to update and upgrade our system based on user feedback and technological advancements. This includes improvements in user interface design, system features, and performance optimization.

By continuously improving and innovating, we hope to keep Tutoryum at the forefront of educational platforms, providing a seamless, engaging, and effective learning experience for all users.

9. 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

RSC: React Server Components

REST: Representational State Transfer

10. References

[1] https://git-scm.com/about [Accessed: Oct 12, 2022]

[2] “Let’s Build from Here”. https://github.com/about [Accessed: Oct 12, 2022]

[3] “We're in Business to Help You thrive”. https://asana.com/company [Accessed: Oct 15, 2022]

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