

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Anna Ivanova 2107371IABB

**Solution Proposal for the EDUKOHT
Programming School Intranet**

Bachelor's Thesis

Supervisor: Bahdan Yanovich
BSc

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Anna Ivanova 2107371IABB

**Programmeerimiskooli EDUKOHT siseveebi
lahenduse ettepanek**

Bakalaureusetöö

Juhendaja: Bahdan Yanovich
BSc

Tallinn 2024

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Anna Ivanova

23.05.20204

Abstract

This thesis examines the development and partial implementation of the proposed user interface for the EDUKOHT programming school. The goal of the study was to analyze system requirements and create interface layouts that are intuitive and easy to use. Based on an analysis of existing platforms, functional requirements were formulated and a use case diagram was developed with three main roles: administrator, mentor and client. The work involved creating layouts in Figma, including a dashboard, course page, and mentor page, and implementing forms for adding events and mentors.

During the implementation phase, registration and login functionality were developed, as well as a calendar for viewing and adding events. Mentor pages with the ability to edit information have been developed. During the work, it was revealed that the project was not developing as quickly as originally planned, and the lack of clear deadlines made it difficult to fully complete the tasks.

For further development of the project, it is recommended to set clear deadlines, expand the functionality of the platform, regularly update and test the system, and integrate user feedback. These steps will help improve the platform and increase its value for EDUKOHT Coding School users.

This thesis is written in English and is 66 pages long, including 4 chapters, 36 figures, 16 table.

Annotatsioon

Programmeerimiskooli EDUKOHT siseveebi lahenduse ettepanek

Käesolevas lõputöös vaadeldakse programmeerimiskooli EDUKOHT väljapakutud kasutajaliidese väljatöötamist ja osalist juurutamist. Uuringu eesmärk oli analüüsida süsteeminõudeid ning luua intuiitivseid ja hõlpsasti kasutatavaid liidese paigutusi. Olemasolevate platvormide analüüsi põhjal formuleeriti funktsionaalsed nõuded ja töötati välja kasutusjuhtude diagramm kolme peamise rolliga: administraator, mentor ja klient. Töö hõlmas Figma paigutuste, sealhulgas armatuurlaua, kursuse lehe ja mentorilehe loomist ning sündmuste ja mentorite lisamise vormide rakendamist.

Rakendusfaasis töötati välja registreerimise ja sisselogimise funktsionaalsus ning kalender sündmuste vaatamiseks ja lisamiseks. Välja on töötatud mentorilehed, millel on võimalus infot muuta. Töö käigus selgus, et projekt ei arenenud nii kiiresti kui algselt planeeritud ning selgete tähtaegade puudumine raskendas ülesannete täielikku täitmist.

Projekti edasiseks arendamiseks on soovitatav seada selged tähtajad, laiendada platvormi funktsionaalsust, regulaarselt uuendada ja testida süsteemi ning integreerida kasutajate tagasiside. Need sammud aitavad platvormi täiustada ja tõsta selle väärtust EDUKOHT Coding Schooli kasutajate jaoks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 66 leheküljel, 4 peatükki, 36 joonist, 16 tabelit.

List of abbreviations and terms

API	<i>Application Programming Interface</i>
Back-end	<i>the part of a website which runs on the server, handling data processing, server-side scripting, and database interactions</i>
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
Front-end	<i>the part of a website that users interact with directly in their web browsers, including everything users experience directly</i>
Git	<i>a distributed version control system designed to process any project quickly and efficiently.</i>
Html	<i>Hyper Text Markup Language</i>
JWT	<i>JSON Web Tokens</i>
LMS	<i>Learning Management System</i>
Moodle	<i>Modular Object-Oriented Dynamic Learning Environment</i>
NPM	<i>Node Package Manager</i>
Scratch	<i>A visual programming language targeted primarily at children</i>

Table of Contents

Author’s declaration of originality	3
Abstract.....	4
Annotatsioon.....	5
List of abbreviations and terms	6
Table of Contents	7
List of Figures.....	9
List of Tables	11
1 Introduction	12
1.1 Background and Problem.....	12
1.2 Goal and Expected Results	13
1.3 Work Structure.....	13
2 Methodology.....	15
2.1 Object Description	15
2.2 Description of Tools	16
2.3 Description of Technologies.....	17
2.3.1 Back-End.....	17
2.3.2 Front-End	18
2.4 Description of Workflow	19
2.5 Feedback Collection	20
2.5.1 Mentor surveys.....	20
2.5.2 Interview with the client.....	20
2.5.3 Author’s own observations.....	20
3 Work Results	22
3.1 Description of Initial Solution	22
3.1.1 Notion.....	22
3.1.2 Moodle	25
3.1.2.1 Login page.....	26
3.1.2.2 Home page.....	27
3.1.2.3 Course page.....	29
3.2 Analysis of Main Users Feedback	31

3.3 Formation of Final Requirements	32
3.4 Prototyping	43
3.5 Technical Solution	47
3.5.1 Database	47
3.5.2 Back-end.....	47
3.5.3 Front-end	48
3.5.4 Some technical aspects of the solution.....	49
3.6 Testing and Validation.....	50
4 Analysis of Results	52
4.1 Technology and Tools	52
4.2 Project Workflow	53
4.3 Requirements	54
4.4 Implemented Functionalities.....	55
4.4.1 Welcome Page.....	55
4.4.2 Login and Registration Pages.....	56
4.4.3 Calendar	57
4.4.4 Mentor Page	59
4.5 Testing and Validation.....	61
4.6 Further Work.....	61
Summary.....	62
References	63
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis	64
Appendix 2 - Login and registration buttons example	65
Appendix 3 - Registration form example	66

List of Figures

Figure 1. Notion Quik links.....	22
Figure 2. Notion Kanban board with tasks.....	23
Figure 3. Notion locations (Estonia)	23
Figure 4. Notion additional hours reporting.....	23
Figure 5. Notion calendar	24
Figure 6. Notion course blocks.....	25
Figure 7. Notion course lessons.....	25
Figure 8. Moodle log-in page	26
Figure 9. Main pages of Moodle	27
Figure 10. Profile settings.....	27
Figure 11. Moodle courses	28
Figure 12. Moodle courses	28
Figure 13. Moodle calendar.....	29
Figure 14. Lessons navigation menu	30
Figure 15. Students' attendance page.....	30
Figure 16. Session form.....	30
Figure 17 Use Case Diagram.....	34
Figure 18. Registration wireframe.....	43
Figure 19. Dashboard wireframe	44
Figure 20. "Add event" form wireframe	44
Figure 21. Courses page wireframe	45
Figure 22. Course page wireframe	45
Figure 23. Course's module page wireframe.....	46
Figure 24. Mentor page wireframe	46
Figure 25. Session routing in `routes.js` file	49
Figure 26. 'server.js' file example	50
Figure 27. Welcome page.....	56
Figure 28. Register page.....	56
Figure 29. Login page.....	57
Figure 30. Dashboard page with calendar	57

Figure 31. Calendar views	58
Figure 32. New event form.....	58
Figure 33. Event information	59
Figure 34. Mentor page	59
Figure 35. Create new mentor form.....	60
Figure 36. Mentor details page.....	60

List of Tables

Table 1. Feedback from mentors	31
Table 2. Log-in function.....	35
Table 3. User registration function	35
Table 4. Calendar of events	36
Table 5. Add event function	36
Table 6. Event change function	37
Table 7. Event delete function.....	37
Table 8. Courses page.....	38
Table 9. Add course function	38
Table 10. Edit course function.....	39
Table 11. Delete course function.....	39
Table 12. Mentors page	40
Table 13. Add a mentor function.....	40
Table 14. Edit mentor function.....	41
Table 15. Delete mentor function.....	41
Table 16. Automatic calculation of mentors' salaries.....	42

1 Introduction

In this chapter, the author gives a brief overview of the developed solution, goals, and problems is provided. Additionally, the created functionality and further structure are described in a generalized manner.

1.1 Background and Problem

"EDUKOHT OÜ"[1] (hereafter referred to as "EDUKOHT") stands as an esteemed hobby school, distinguished by its provision of programming courses tailored for children and youth. The institution operates across four Estonian cities — Tallinn, Narva, Kohtla-Järve, and Jõhvi — engaging over 900 students and more than 30 young mentors in its educational pursuits. Predominantly, EDUKOHT offers courses on web development and Scratch, with the former concentrating on the foundational aspects of HTML, CSS, and JavaScript, and the latter being a block-based visual programming language and website, designed primarily as an educational instrument for children.

The educational and operational mechanisms of the organization encompass a multitude of complex tasks, including but not limited to, the facilitation of communication with mentors, the meticulous tracking of attendance, the diligent monitoring of student progress, and the precise recording of employees' regular and overtime hours. Presently, the management of these tasks is executed through a variety of online tools, such as social media chats, Google Sheets, and Learning Management System (LMS) Moodle [2], which has been developed specifically within the EDUKOHT framework.

Nonetheless, this segmented approach to information management incurs significant inconvenience, necessitating considerable time and effort to aggregate and interrelate data from disparate platforms. Furthermore, this decentralization of data accentuates the propensity for inaccuracies, with the omission of critical information, erroneous tracking, or oversight potentially culminating in a range of adverse outcomes. These may include failure to disburse mentors' remunerations, inaccuracies in student financial accounting, and other implications.

1.2 Goal and Expected Results

Given the complexities caused by such a dispersed management approach, the organization's founders decided to initiate the development of an intranet that would serve all stakeholders involved in the educational process.

The goal of the project is to propose a solution for EDUKOHT programming school intranet.

The expected results are:

1. New intranet system requirements
2. Mock-ups
3. Some implemented features

A unified platform, consolidating data from the Learning Management System (LMS), Google Sheets, Type forms, and other sources, would significantly streamline and automate administrative operations while improving communication among all participants.

Considering the substantial amount of time required for the project, the owners of EDUKOHT do not expect the author to implement the full range of functions.

1.3 Work Structure

This thesis is structured into four main parts, each dedicated to a distinct phase of the project lifecycle, from initial methodologies to the final analysis of the results.

This chapter lays the foundation for the project, detailing the object of the project, and the tools and technologies employed. It covers everything from the initial description of the project object to the workflow and the technologies used for both backend and frontend development. Additionally, this section discusses the methods used for collecting feedback, such as mentor surveys, client interviews, and observations.

This part delves into the application of the initial solutions such as Notion and Moodle, and explores their functionality including the login, home, and course pages. It also discusses the prototyping process using Figma wireframes, the technical solution including database and backend development, and the subsequent phases of testing and validation to ensure the system's efficacy and security.

The final part of the thesis evaluates the system's performance through various analytical methods and discusses the implemented functionalities such as the Welcome Page, login and registration interfaces, and additional features like the calendar and mentor pages. This chapter culminates with a comprehensive assessment of the project outcomes, highlighting both the successes and areas for future work.

2 Methodology

In this chapter, the author describes the project and a complete list of the technologies and tools used in it. This chapter also includes a detailed description of the workflow process.

2.1 Object Description

Initially, the idea of developing a proprietary intranet for a programming school belonged to one of the company's co-founders. By the time the project was handed over to the author of this work, a project had already been created in the company's personal GitHub, and there was a Kanban board with project tasks on the Jira project management platform.

Before beginning the work, the author was provided with all the necessary components for the job:

1. Access to all necessary materials and development environments.
2. Support — the co-founder of EDUKOHT (hereafter the project client) provided support during the analysis process and subsequently during the development process when problems arose.

The author's first step was to conduct a business analysis to clarify and formalize the requirements for the intranet. Interviews were conducted with key users and stakeholders to determine their needs and expectations of the system. This information was systematized and transformed into clearly formulated requirements, which were then approved by the client. The Jira board was used to manage the requirements and their priorities, ensuring transparency of the process and ease of tracking changes and additions to the project.

The decision was made to use Vue.js [4] for developing the user interface and the open-source backend framework Node.js. [5] This choice was made in favor of these technologies because they were proposed by the project client, and moreover, the author already had experience developing web applications using these technologies.

2.2 Description of Tools

GitHub is an internet hosting service specializing in software development and version control using the Git [6] system. In this project, GitHub [7] was used as the primary repository for storing source code. Its features for monitoring changes and additions to the code significantly facilitated centralized management and development coordination.

Jira [8] is a project management platform widely used to organize work processes. In this project, Jira was utilized for planning, coordinating, and tracking task completion. Thanks to its capabilities, the author was able to organize the workflow, structure tasks, and monitor their execution in real time, which contributed to increased work efficiency.

Figma is an interface design tool that allows teams to create, collaborate, and share design components in real time. Figma was used in this project for developing and implementing interface wireframes. With its help, the author could visualize the structure of the proposed solutions, easing the process of verifying design concepts and ensuring effective interaction with the development team.

WebStorm [9] development environment was chosen for developing the client side of the application using the Vue.js framework. WebStorm is ideally suited for web development, providing powerful tools for working with JavaScript, CSS, and HTML. The environment supports modern frameworks such as Vue.js and offers features that simplify the development of interactive user interfaces, including built-in debugging, profiling, and testing tools, as well as integration with version control systems.

IntelliJ IDEA [10] was used for developing the server side on Node.js. Thanks to its versatility and powerful capabilities, it effectively supports development in JavaScript and Node.js. IntelliJ IDEA provides comprehensive support for Node.js, including dependency management through npm, application running and debugging, and support for asynchronous programming.

2.3 Description of Technologies

This chapter contains detailed information about the key technologies used in the development of this application.

2.3.1 Back-End

For the server-side part of this project, Node.js technology was used in combination with the Express.js framework [11]. Node.js is a JavaScript runtime built on the V8 engine from Google Chrome, enabling the development of both client-side and server-side web applications in JavaScript. Its asynchronous, event-driven environment is ideal for creating scalable network applications. Express.js, a minimalist and flexible web application framework for Node.js, was chosen to simplify the development process. It provides an extensive set of features for web and mobile applications, simplifying routing, middleware integration, and HTTP utility methods.

The decision to use Node.js and Express.js was based on several factors. Node.js uses a non-blocking, event-driven architecture, allowing it to handle multiple connections simultaneously and efficiently. Unlike traditional multi-threaded servers, Node.js operates on a single-threaded event loop, which simplifies memory management and reduces the likelihood of synchronization errors. Using JavaScript on both the client and server sides offers versatility. This allows developers to reuse code, streamline development processes, and reduce the learning curve associated with new languages.

NPM (Node Package Manager) provides access to many open-source packages and modules. This rich ecosystem accelerates development by offering pre-built solutions for common tasks, thus reducing the time and effort required to implement complex features. Node.js excels in performance and scalability, making it a leading platform for server application development. Its ability to handle numerous simultaneous connections with high throughput is particularly beneficial for real-time applications. Using Express.js with Node.js significantly simplifies the development process. Express.js offers a structured yet flexible

approach to building web applications, allowing developers to focus on writing business logic rather than dealing with the intricacies of server configuration.

In this project, SQLite [12] was chosen as the database to complement the Node.js and Express.js stack. SQLite is a self-contained, serverless, and zero-configuration SQL database engine. This choice was made due to its simplicity, portability, and ease of integration with Node.js. SQLite stores data in a single file, which makes it ideal for small to medium-sized applications and development environments where setup overhead needs to be minimal. Additionally, SQLite's lightweight nature ensures that it requires minimal system resources while still providing full SQL capabilities.

The combination of Node.js, Express.js, and SQLite in this project enhances development efficiency, reduces time to market, and ensures the creation of high-performance, scalable applications. This technological stack continues to be a preferred choice for modern web application development due to its robustness, flexibility, and ease of use.

2.3.2 Front-End

Vue.js was chosen as the primary framework for the front-end development of the project. Vue.js is a progressive JavaScript framework designed for building user interfaces. One of its key features is its incremental adaptability, which allows it to be used as either a simple library for small projects or as a full-fledged framework for complex solutions. This flexibility gives developers the freedom to choose the necessary tools depending on the requirements and complexity of the project. Moreover, Vue.js features an intuitive and compact API, which facilitates quick learning and accelerates the development process. The framework also has a convenient modular component structure, simplifying code reuse and maintenance, and ensures high interface reactivity through efficient change tracking and automatic DOM updates.

Bootstrap was also used to enhance the user interface and ensure the project's adaptability [13]. This frontend toolkit enables rapid creation of modern, responsive, and visually appealing interfaces. Bootstrap includes an extensive set of ready-to-use components and

utilities, which significantly speeds up development and allows focusing on the unique aspects of the project without needing to deal with routine styling.

Using Vue.js in combination with Bootstrap is ideal for creating powerful yet easily maintainable applications, providing excellent compatibility with Node.js. This combination of technologies allows developers to achieve high scalability and flexibility, making the choice of Vue.js and Bootstrap an excellent solution for our project.

2.4 Description of Workflow

The workflow was initiated by providing access to the project through the corporate platforms GitHub and Jira, which created the basis for starting work. At the outset, general requirements and technology descriptions were presented in a vague manner, highlighting the need for a thorough business analysis. This analysis became a central part of the process, as it allowed us to formulate specific requirements for the functionality of the developed platform and identify key tools for their implementation.

The implementation of business analysis included collecting feedback from the parties and analyzing current business processes in the company, which helped to identify all the necessary functional and technical requirements. This data was critical to forming a clear picture of the project.

Instead of a traditional approach with strict time frames, meetings between myself and the project client occurred on an as-needed basis.

At any time, the author of the work had the opportunity to contact the customer for clarification or assistance. This provided effective support and prompt resolution of emerging issues. Maintaining ongoing dialogue and knowledge sharing between parties had a positive impact on workflow and results.

Thus, the work process was organized in such a way as to provide maximum flexibility and the ability to quickly adapt to changing conditions and project requirements.

2.5 Feedback Collection

This chapter introduces 3 main methods for collecting feedback for the future analysis: mentor surveys, interview with client and surveillance.

2.5.1 Mentor surveys

In the process of collecting feedback, it was decided to conduct a survey among mentors to evaluate their experience of using the current educational platform. The survey was conducted in an informal manner: mentors were given the opportunity to leave their comments in free text without asking questions. This approach allowed for the collection of honest and detailed feedback, expressing participants' personal opinions and suggestions for improving the system.

2.5.2 Interview with the client

To gain a deep understanding of the customer's requirements, a video meeting was organized, during which the customer shared his impressions of the shortcomings of the current platform. This session allowed us not only to clarify specific aspects that require improvement, but also to discuss potential areas for functionality development. The interview helped to identify key problems affecting the ease of use and efficiency of the educational process.

2.5.3 Author's own observations

The author of this work, actively participating in the learning process as a mentor at the EDUKOHT programming school, had the opportunity to personally evaluate the functionality and ease of use of the discussed platforms. This experience allowed to observe real-life applications of the systems and identify problems faced by users on a daily basis.

Direct participation in the educational process and the use of platforms made it possible to form a valuable opinion on the necessary improvements and additions.

3 Work Results

3.1 Description of Initial Solution

This chapter provides a description of the initial tools used by the EDUKOHT Programming School before the development of the project began.

3.1.1 Notion

Notion, a universal tool for project management and workflow organization, plays an important role, although it has not been used in the EDUKOHT organization for over a year. The author emphasizes the significance of Notion, particularly in the context of project requirement analysis, but notes a significant reduction in its use.

On the main page of Notion, there are links for quick access to important information (Figure 1), as well as a Kanban board for tracking tasks (Figure 2). It also includes a list of cities where the organization is active, although some of them are currently inactive (Figure 3 and 4). A feature for mentors to register additional work hours is available.

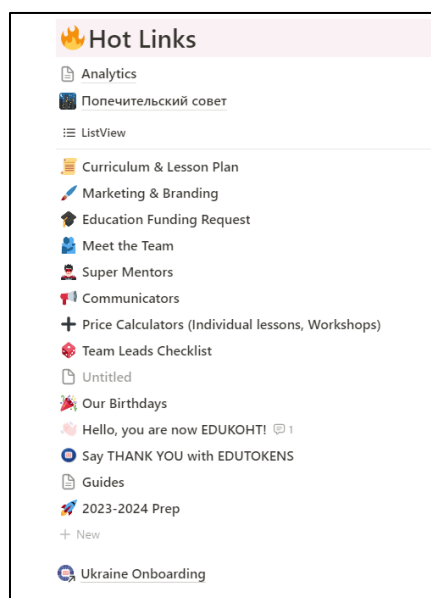


Figure 1. Notion Quik links

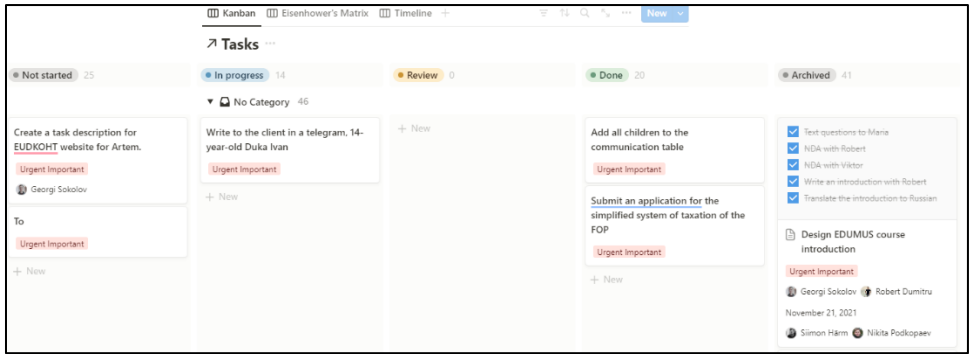


Figure 2. Notion Kanban board with tasks

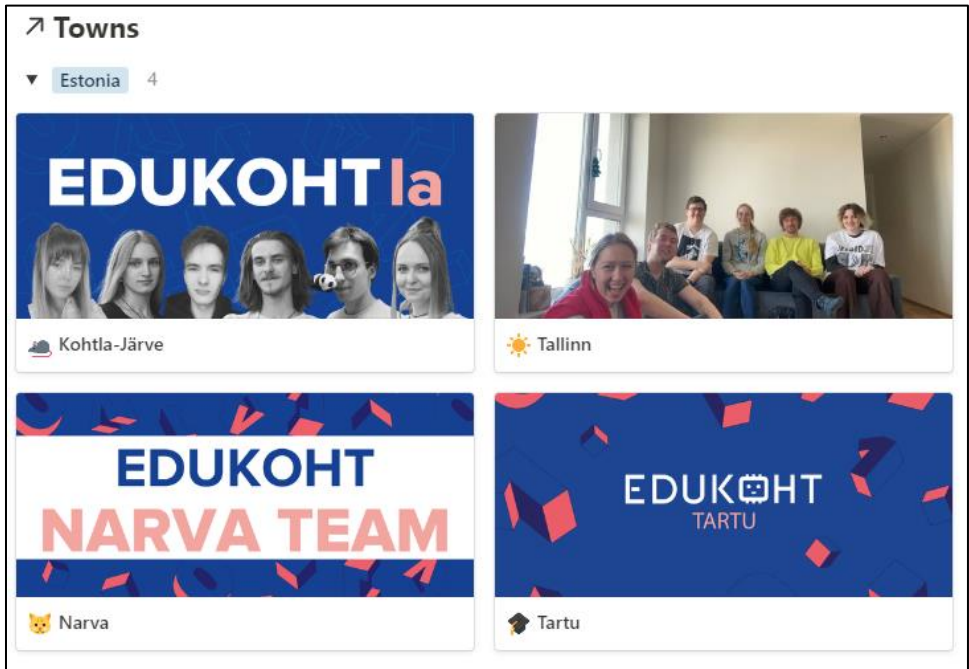


Figure 3. Notion locations (Estonia)

Activity description	Team Member Name	Date Range	Location	Activity type
Try it out	Kirill Fomin	January 7, 2023	Tallinn	Trial Lesson
Try it out	Anna Ivanova	January 7, 2023	Tallinn	Trial Lesson
Feedback	Oliver Tamm	January 7, 2023	Tartu, Kohtla-Järve	General
Crashing computers	Henry-Gerret Grüning	January 2, 2023	Tartu	General
Repairing computers	Oliver Tamm	January 2, 2023	Tartu	General
Roblox	Henry-Gerret Grüning	January 1, 2023 – January 31, 2023	Tartu	Curriculum
Try it out	Henry-Gerret Grüning	December 1, 2022 – December 31, 2022	Tartu	Trial Lesson
Try it out	Jürmo Mihhalov	December 1, 2022 – December 31, 2022	Tartu	Trial Lesson
Try it out	Oliver Tamm	December 1, 2022 – December 31, 2022	Tartu	Trial Lesson

Figure 4. Notion additional hours reporting

The location page includes information about the status of various places, whether active or inactive, with the city of Tallinn as an example. It contains links that provide detailed information about each location, including logistical aspects and security procedures. A notable feature is the calendar, which helps mentors track lessons and update information on student attendance and progress (Figure 5).

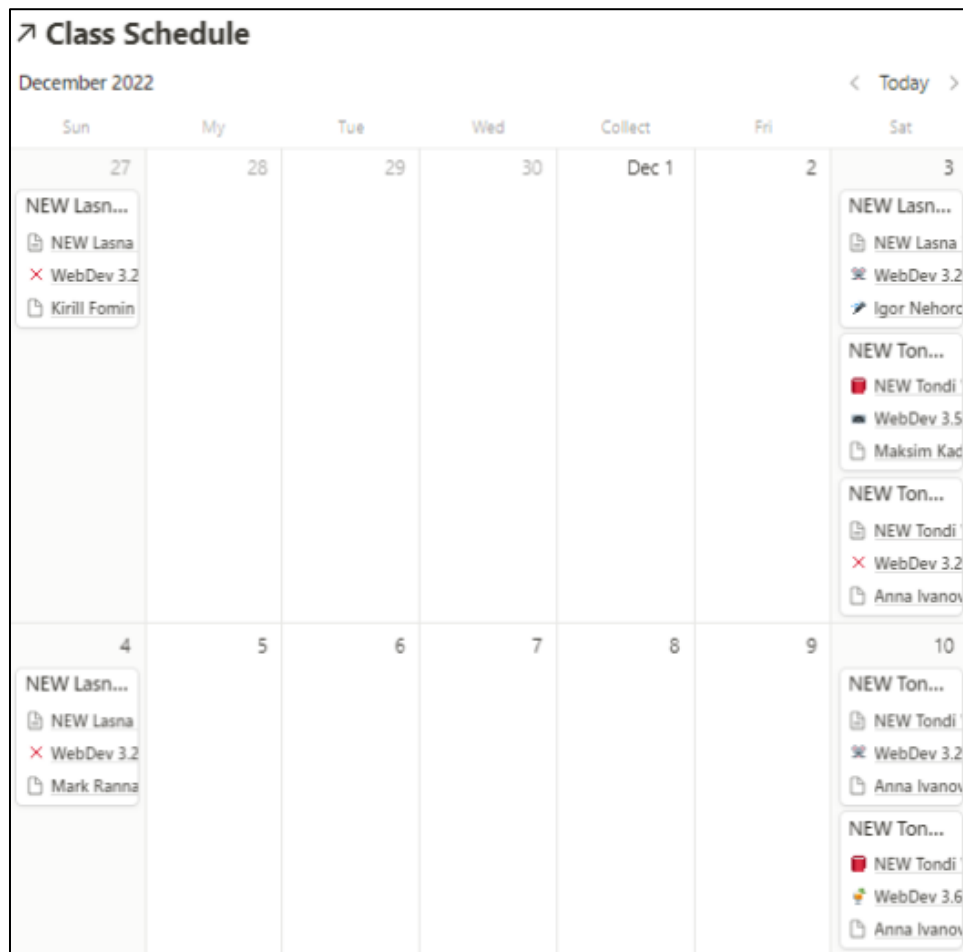


Figure 5. Notion calendar

The course page provides a structured overview of courses, such as web development. It describes the content of the course, its objectives, and the technologies used, as well as thematic blocks with details of each session (Figure 6 and 7).

Blocks

Add a new block

Block 1: HTML and CSS Elements

Direction Goals

Students should be able to construct their own basic pages using HTML, CSS and JS.

Students should be able to find the answer to an IT question on the internet.

Description

The block focuses on learning the basics of HTML and CSS. Students learn how to make the base structure of the web page, how to make styling of the elements on it using tag name selectors, classes, and ids, and how to make an interactive design using pseudo-class hover. Students learn about headings, images, paragraphs, break lines, thematic break (<hr>) tags, and other HTML tags. They will be able to style elements using color, width, height, background color, margin, border properties, and other CSS elements.

Figure 6. Notion course blocks

Lesson Plan	Lesson Objective	Eduskills	New Content
WebDev 1.1 Text Tags 2	1. Explore HTML, CSS	Text Formati...	<p> <h1>-<h6> <style> color background-color
WebDev 1.1 Text Tags ENG	1. Explore HTML, CSS	Text Formati...	<p> <h1>-<h6> <style> color background-color
WebDev 1.2 Formatting Tags 6	1. Use text formatting	Text Formati...	 <i> <small> <mark>
WebDev 1.3 Media, Lists, Sections 4	1. Embed videos and i	Text Formati...	 <iframe> <hr> width height
WebDev 1.4 Div and Border 2	1. Investigate the bloc	Text Formati...	<div> border border-radius :hover <header>
WebDev 1.5 CSS Layout	1. Position elements c	Element Styli...	position: relative position: absolute z-index
WebDev 1.6 Synthesis: 1.1 - 1.7 2	1. Practice applying te	Text Formati...	

Figure 7. Notion course lessons

Without delving into the details of each feature, the author highlights the reduced yet conceptually important use of Notion in the current activities of EDUKOHT, aligning with the organization's strategy to decrease dependency on this platform.

3.1.2 Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) is a free, open-source learning management system (LMS) used to create interactive online courses and websites. It is one of the most popular platforms for e-learning and is particularly favored in academic and educational institutions worldwide due to its flexibility, customizability, and active user community. Moodle offers extensive features including course management, assessment of

learning achievements, discussions (forums), file repositories, automated testing, and much more.

The EDUKOHT team switched to using Moodle as their main platform in March 2023. Subsequent chapters will provide descriptions of the main pages in Moodle.

3.1.2.1 Login page

On the Moodle login page, there is a field for entering a username or email address and a separate field for the password, allowing users to input their credentials. To log into the system, users can click the "Log in" button. If a user has forgotten their password, they can use the "Lost password?" link to start the recovery process. The page also includes a section for new users with the question "Is this your first time here?" and a "Create new account" button, which allows new visitors to register on the site. These elements provide convenient access to an account or help create a new account if the user is visiting the site for the first time.

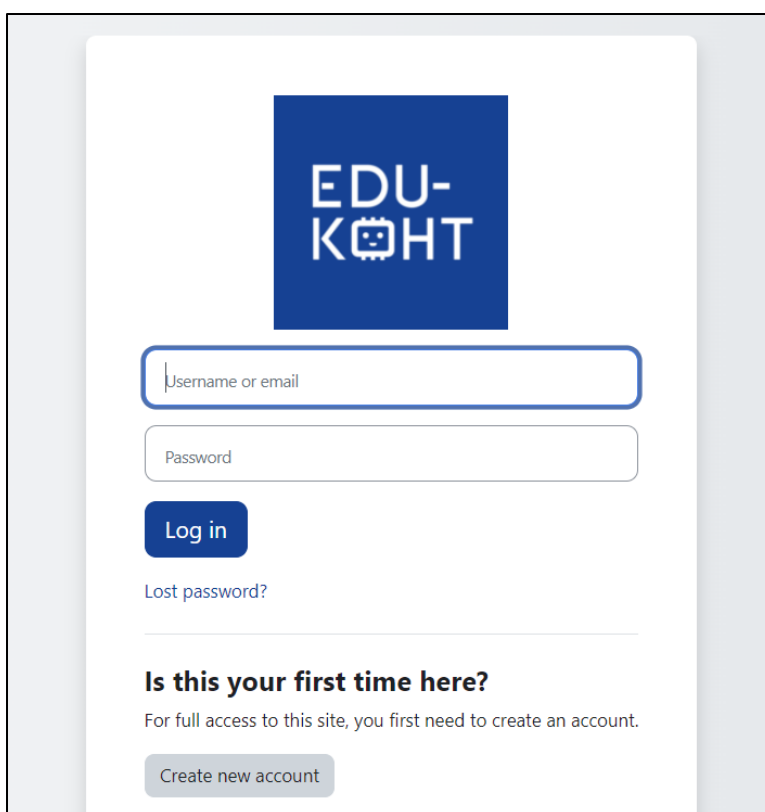


Figure 8. Moodle log-in page

3.1.2.2 Home page

With the transition to the Moodle platform, the core functionality previously available in Notion has been preserved.

Moodle on the EDUKOHT platform includes three main pages: the homepage, the control panel, and the "My Courses" section. The top panel features icons for notifications, messages, and user photos (Figure 9). Clicking on the user avatar provides access to their personal profile, private files, calendar, and reports. Additionally, the user can change the interface language, gender, or log out of the system (Figure 10).



Figure 9. Main pages of Moodle

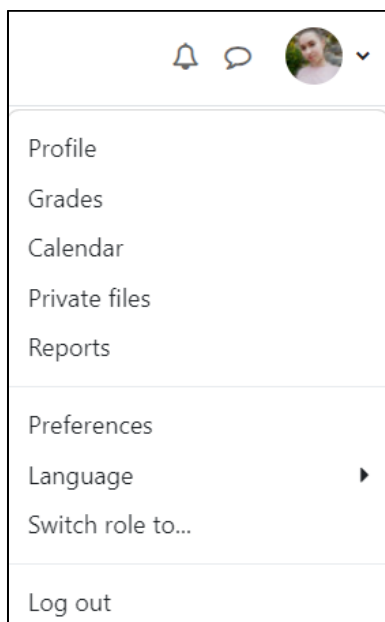


Figure 10. Profile settings

On the homepage, there is a list of active courses which can be accessed by clicking on their respective links. In addition to the main courses, the platform also features additional sections: "Welcome to EDUKOHT", "Mentor Statistics", and "Guide to Using Moodle" (Figure 11 and 12).

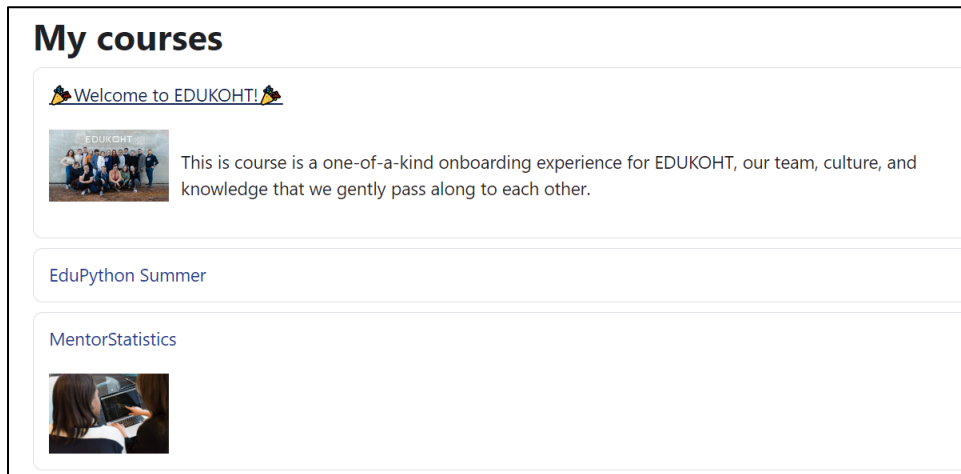


Figure 11. Moodle courses



Figure 12. Moodle courses

The platform also retains the functionality of a calendar, which allows users to add events. This tool provides the ability to plan and organize various activities within educational operations, ensuring effective time and resource management. The calendar is easily integrated with other platform features, making it convenient for daily use in an educational environment (Figure 13).

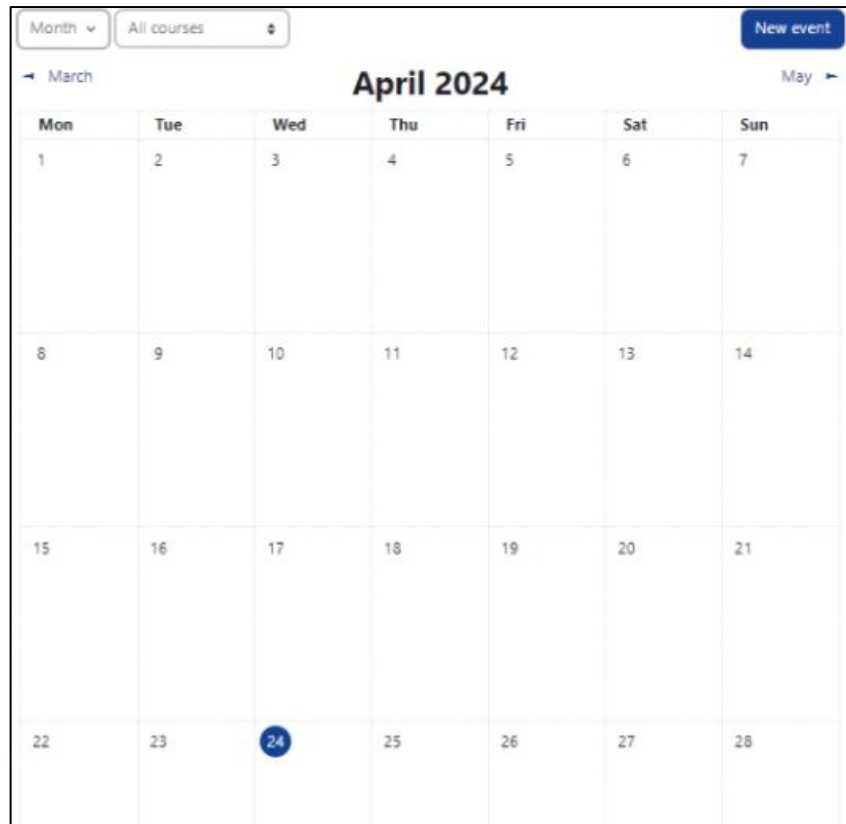


Figure 13. Moodle calendar

3.1.2.3 Course page

The course page maintains a similar structure to that of Notion, where the material is divided into thematic blocks. On the left, there is a navigation menu that allows for quick access to the necessary lessons (Figure 14).

Unlike Notion, the attendance tracking of students is now conducted through a special attendance page (Figure 15). The mentor has the option to add a session using the corresponding button. After the form is filled out, the data about the lesson is displayed in a general table (Figure 16).

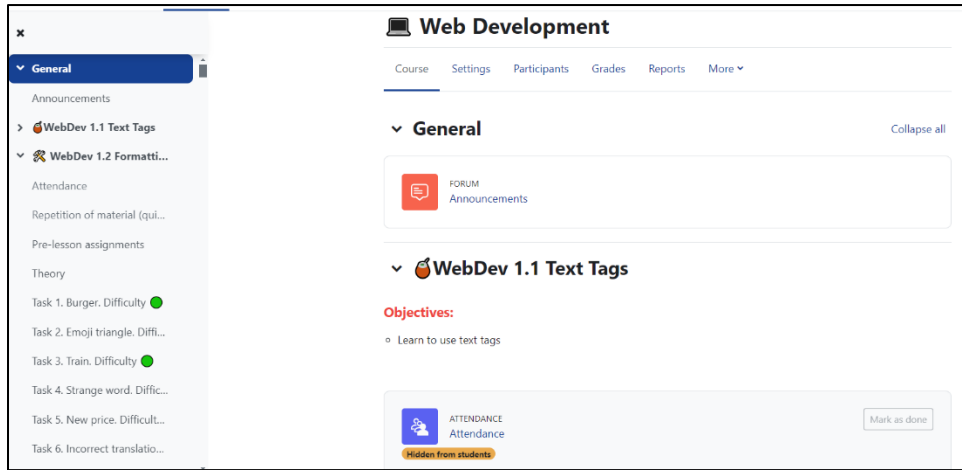


Figure 14. Lessons navigation menu

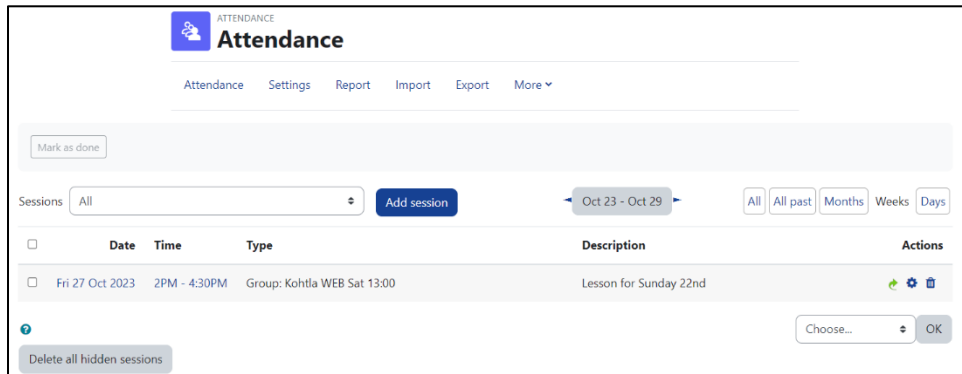


Figure 15. Students' attendance page

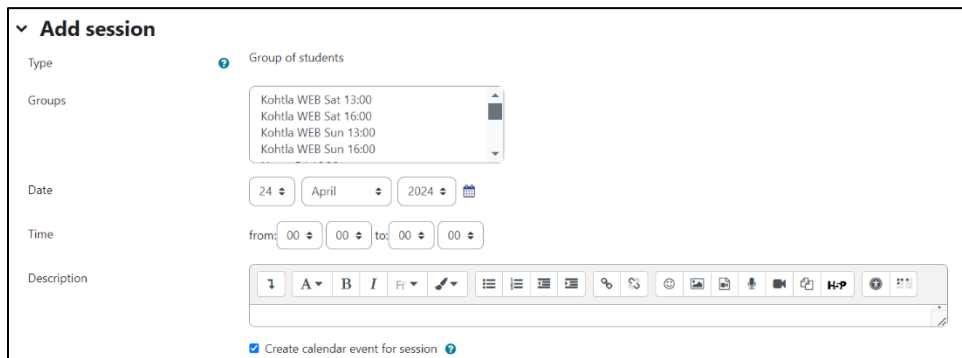


Figure 16. Session form

3.2 Analysis of Main Users Feedback

In this chapter, we delve into collecting and analyzing feedback from mentors specifically regarding their use of Moodle. The focus is to identify and understand the existing challenges that mentors encounter with Moodle. The analysis reveals that while Moodle offers various tools, there were significant requests for enhancements to support user interaction. The feedback underscores key weaknesses that hinder mentors' ability to manage study groups and courses are effectively outlined. The main feedback points are detailed in Table 1.

List management system	Mentors face difficulties due to the lack of filtering capabilities in list management systems, which complicates the segmentation and management of students. The absence of dashboards also makes it difficult to track attendance and workload. It is recommended to implement enhanced filtering features and dashboards that would provide mentors with deeper analysis and better control over the learning process.
Corner case handling	Challenges arise in handling special cases, such as when a student is also a mentor. Current solutions in the Moodle system are assessed as suboptimal. Development of more flexible mechanisms to account for exceptional situations is needed, which would improve UX and simplify user management.
Feedback and lesson coordination with the topic	The lack of a structured feedback process and rigid linkage of lessons to specific topics limit the flexibility of the educational process. It is proposed to create mechanisms for more flexible integration of feedback into curricula and develop methodologies that allow mentors to adapt educational materials based on the needs and requests of students.
Mentor limitations	Mentors working with various courses experience difficulties with switching between groups and accessing resources from different cities. It is important to provide a unified integrated management system that allows mentors to freely move between courses and groups, and ensures reliable and secure access settings.
Interface and settings	A complex and cluttered interface with an excessive number of filters complicates the work of mentors. It is recommended to redesign the interface to make it more intuitive and minimalist, which would improve the overall user experience.

Table 1. Feedback from mentors

In addition to improvements based on current feedback, the client expressed interest in developing a feature for automatic salary calculation after each conducted lesson. This feature should be integrated not only into the administrative interface but also made available to mentors to enhance transparency and control over their finances, as well as to avoid errors in salary calculations.

3.3 Formation of Final Requirements

This chapter describes the process of generating specifications for the key functional requirements of an educational platform. An effective requirements specification not only ensures clarity of technical implementation, but also ensures that the final product meets user expectations and is tailored to current and future educational needs. The chapter also covers the key features needed to make the platform fully functional, including course and mentor pages, content addition features, an event calendar and location information, as well as the specification of the login function.

The use case diagram is a crucial element in visual modeling that aids in analyzing a system's functional requirements. It shows various actors (users or external systems) who interact with the system, and the use cases, which describe the actions that can be performed by the actors with the system [14].

The system defines three main types of users: Administrator (Admin), Mentor, and Client (Parent).

Administrator Capabilities

The Administrator has the most extensive powers in the system:

- Course management: adding, removing, and editing courses.
- Mentor account management: adding, removing, and editing mentor accounts.
- Client account management: adding, removing, and editing client accounts.
- Viewing and managing payments: automatic payment calculation based on worked hours, issuing invoices.

- Managing information about locations: adding, removing, and editing information about course venues.

Mentor Capabilities

Mentors in the system have opportunities for interacting with courses and materials:

- View assigned courses: access to the list of courses that the mentor leads.
- Adding comments to course materials: ability to leave notes and comments on materials to improve courses.
- View payment information: access to information about the number of hours worked and corresponding payments.
- Access to course materials: viewing educational materials for preparation and conducting classes.

Client (Parent) Capabilities

Clients (parents) using the system can manage their children's education and interact with mentors:

- View available courses and enroll children: choosing courses for their children's education.
- View children's progress and attendance records: monitoring children's performance and attendance.
- Make payments/view payment history: managing financial issues related to education.
- Communicating with mentors and administrators: opportunity for feedback and managing requests.
- Access resources and materials: receiving educational and additional materials from the organization

Use cases are visualized on Figure 17.

Use Case Diagram

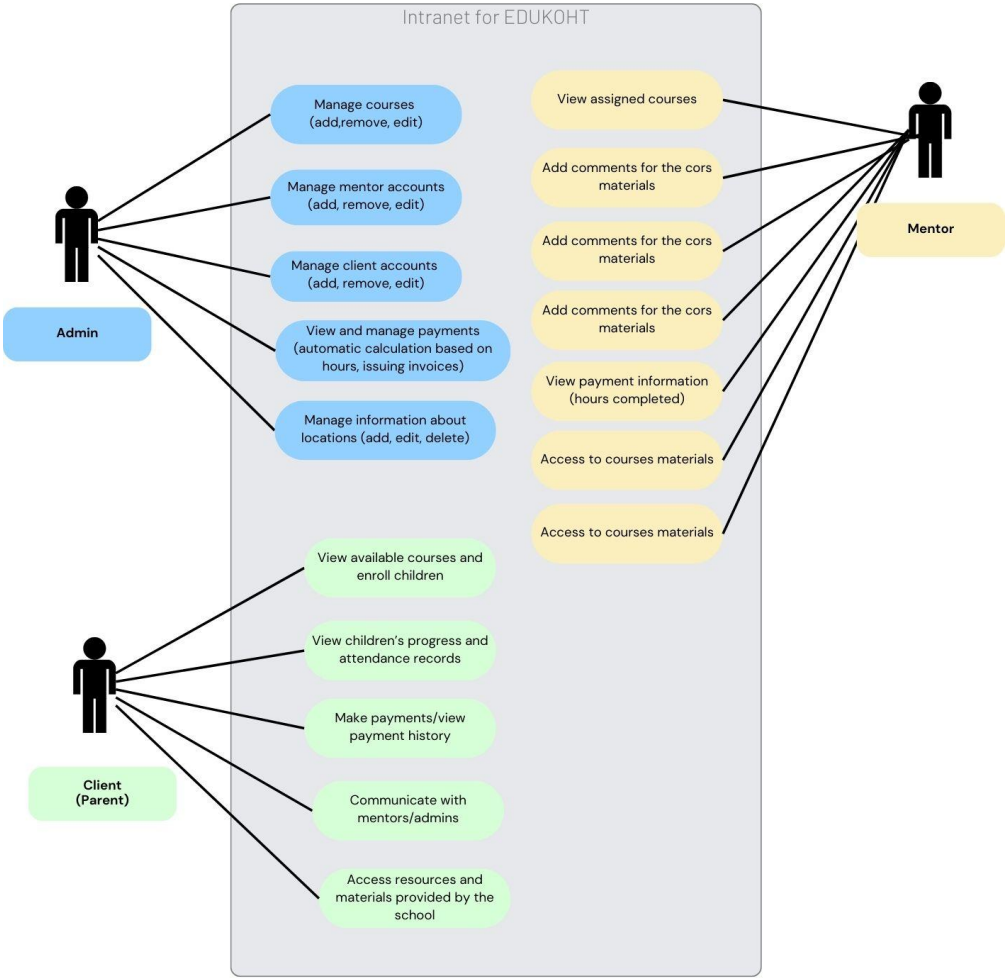


Figure 17 Use Case Diagram

Specifications for key functional requirements are presented in Tables 2 - 16.

Log-in function	
Functional	Secure user access to their accounts.
Technical requirements	Integration with security system for authentication and authorization.
Interface	Simple login form with options to restore access.
Acceptance criteria	Authentication security, user data protection, login speed.

Table 2. Log-in function

User registration function	
Functional	Allows new users to create an account on the platform by entering the necessary personal data such as name, email, password and possibly additional profile data.
Technical requirements	Integration with user database to save new entries. A function is required to check the entered information for uniqueness and validity. Implementation of security mechanisms to protect user data.
Interface	A registration form that contains data entry fields, instructions for users about data requirements (eg, email format, password requirements), and a registration confirmation button.
Acceptance criteria	The user should successfully create an account without errors. All data must be stored correctly in the database. The user is automatically sent an email confirming registration.

Table 3. User registration function

Calendar of events	
Functional	Displaying main events, lessons, and important dates for users.
Technical requirements	Integration with calendar management system, support for multiple time zones. Validation for date, event cannot take place in the past.
Interface	Intuitive calendar with customization and presentation options. It is possible to view specific days, weeks and months separately
Acceptance criteria	Accuracy of event display, ease of setup, multilingual support.

Table 4. Calendar of events

Function of adding an event to the calendar	
Functional	Allows users to add events to the calendar, including setting the time, date, location, participant selection, and event description. Supports the creation of one-time and recurring events.
Technical requirements	Integration with calendar database to store all events. API support for synchronization with external calendar services such as Google Calendar and Microsoft Outlook. Implement notification functions to alert users about upcoming events.
Interface	A form for adding an event with fields for entering all the necessary details: selecting a date and time through an interactive control, fields for entering a location and description, functions for selecting participants from the contact list. The interface should provide checkboxes to configure the repetition of events.
Acceptance criteria	Correct execution of functions for adding and editing events, accurate saving of user-specified event details into the database. The interface should be intuitive and provide high speed data entry. Security of event data and protection from unauthorized access to the calendar.

Table 5. Add event function

Event change function	
Functional	Allows users to change details of already scheduled events, including time, date, location, list of participants and description.
Technical requirements	User rights to edit an event must be checked. Database integration to update event information in real time.
Interface	An event editing form that allows you to change all aspects of the event through convenient interface elements such as calendar pickers, text fields and drop-down lists.
Acceptance criteria	Changes should be correctly displayed in the calendar of all event participants. The system should automatically notify all participants about changes to the event.

Table 6. Event change function

Event delete function	
Functional	Allows users to remove scheduled events from the calendar. This feature should only be available to users who have permission to edit the event.
Technical requirements	Integration with a calendar management system to provide the ability to delete events. User rights to delete a specific event must be verified to avoid unauthorized access.
Interface	A delete button or icon located next to each event on the calendar. A confirmation dialog should appear asking the user if they are sure they want to delete the event
Acceptance criteria	The deletion must be quick and accurate, and the event must completely disappear from everyone's calendar. Once deleted, a notification must be sent to all event participants that it has been cancelled.

Table 7. Event delete function

Courses page	
Functional	View available courses and the ability to register for them.
Technical requirements	Ensuring connection with the course database, support for filtering and searching.
Interface	Clear and intuitive list of courses with detailed information about each course.
Acceptance criteria	Completeness of information presentation, page loading speed, ease of navigation.

Table 8. Courses page

Add course function	
Functional	Allows administrators to add new courses to the platform, including course title, description, start and end dates, and mentor information.
Technical requirements	Integration with course database to save new entries. Checking the uniqueness of course names to avoid duplication.
Interface	Form for adding a course with fields for entering all the necessary details. Drop-down lists should be provided for selecting mentors, if necessary.
Acceptance criteria	The course is successfully added to the system without technical errors. All data is displayed correctly in the general list of courses.

Table 9. Add course function

Edit course function	
Functional	Allows administrators to change information about existing courses, including title, description, dates, and associated mentors.
Technical requirements	Database integration to update existing records. A function to check changes for conflicts or errors is required.
Interface	A course editing form that allows the administrator to change all aspects of the course through convenient interface elements.
Acceptance criteria	Changes must be correctly saved and displayed in the system. The system should provide feedback on the successful saving of changes.

Table 10. Edit course function

Delete course function	
Functional	Allows administrators to remove courses from the system.
Technical requirements	Database integration for deleting course records. There must be precautions to prevent accidental removal.
Interface	Delete option in the course management interface with confirmation of the action through a dialog box.
Acceptance criteria	The course must be completely deleted from the system without the possibility of recovery, with confirmation of deletion for the administrator.

Table 11. Delete course function

Mentors page	
Functional	Providing information about mentors, their qualifications, specializations and availability.
Technical requirements	Communication with the personnel database, ensuring the relevance and accuracy of the data.
Interface	Mentor profiles with detailed descriptions and the possibility of direct communication.
Acceptance criteria	Data relevance, ease of use, protection of personal information.

Table 12. Mentors page

Add a mentor function	
Functional	Allows administrators to add new mentors to the system, including their personal details, qualifications and courses they teach.
Technical requirements	Integration with the mentor database to create new entries. Checking for uniqueness of data to prevent duplication.
Interface	A form for adding a mentor with fields for entering data and the ability to select courses that the mentor will teach.
Acceptance criteria	The mentor is added to the system without errors, and his data is displayed correctly in the list of mentors.

Table 13. Add a mentor function

Edit mentor function	
Functional	Allows administrators to change information about mentors, including their personal details, qualifications and courses they teach.
Technical requirements	Database integration to update existing records. A function to check changes for conflicts or errors is required.
Interface	A form for editing mentor data, allowing the administrator to change all the necessary information through convenient interface elements.
Acceptance criteria	Changes must be correctly saved and displayed in the system. The system should provide feedback on the successful saving of changes.

Table 14. Edit mentor function

Delete mentor function	
Functional	Allows administrators to remove mentors from the system.
Technical requirements	Database integration to delete mentor records. There must be precautions to prevent accidental removal.
Interface	Delete option in the mentor management interface with confirmation of the action through a dialog box.
Acceptance criteria	The mentor must be completely removed from the system, with confirmation of removal for the administrator.

Table 15. Delete mentor function

Automatic calculation of mentors' salaries	
Functional	Automation of the process of calculating salaries for mentors based on hours worked and established tariffs, as well as on the basis of additional hours of work entered.
Technical requirements	Database integration to track mentors' time in class and their rates.
Interface	Panel for mentors and administrators to enter, view and approve work data.
Acceptance criteria	Accuracy of calculations, ease of use of the interface, data security.

Table 16. Automatic calculation of mentors' salaries

The tables presented above describes the key specifications for various functions that are important for managing and servicing an educational platform. Each function — from adding and editing courses to managing mentor profiles — is carefully designed to ensure efficiency, security, and ease of use. The specifications include a clear definition of functionality, technical requirements, user interface, and acceptance criteria. These elements are critically important to ensure that each function not only meets technical standards and business requirements but also provides the best possible experience for end users—administrators, mentors, and students. The described processes and standards help guarantee that all aspects of the system work cohesively and effectively, supporting seamless and productive educational activities.

3.4 Prototyping

Chapter delves into the prototyping phase to explore and refine ideas before the start of full-scale development.

During the implementation of the approved procedure for forming functional requirements, detailed mockups were developed using Figma software. These mockups significantly facilitated the approval process of the platform's functionality from the client's side. Subsequently, these same mockups were used in the project's technical implementation.

Registration and Login

The "EDUKOHT" registration and login page is presented as a simple and clean form, allowing users to quickly log into the system or register (Figure 18). The "Log In" and "Register" buttons are clearly highlighted, ensuring easy navigation. This interface element is critical, as it represents the user's first interaction with the platform.

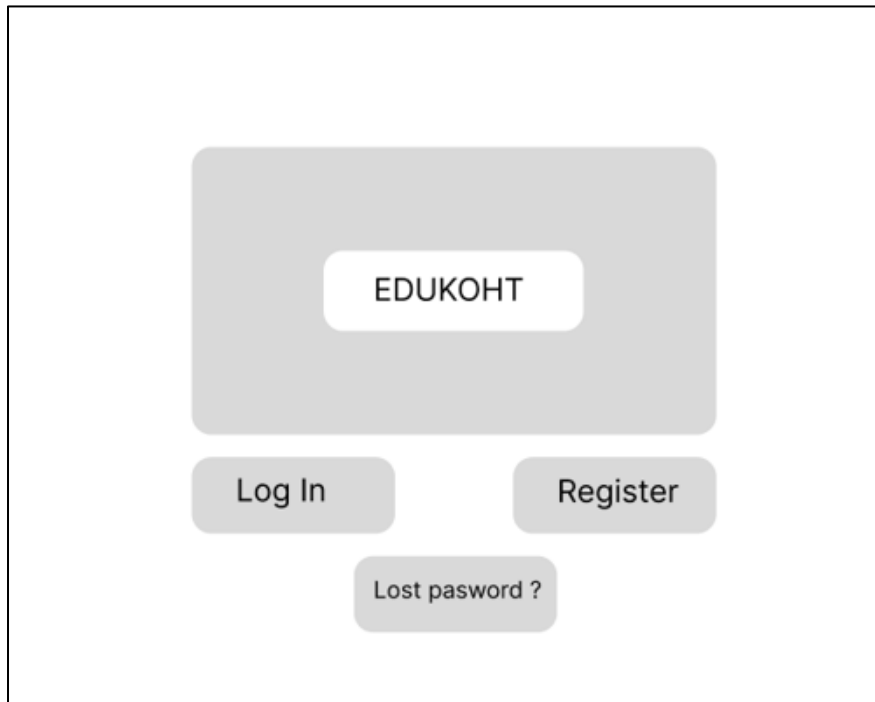


Figure 18. Registration wireframe

Dashboard

The "EDUKOHT" dashboard includes widgets for quick access to key features such as an event calendar, access to courses, and other resources (Figure 19-20). Also featured on the dashboard is a calendar with events. The "Add Event" button allows users to add events to the calendar, which is an important function for planning the educational process.

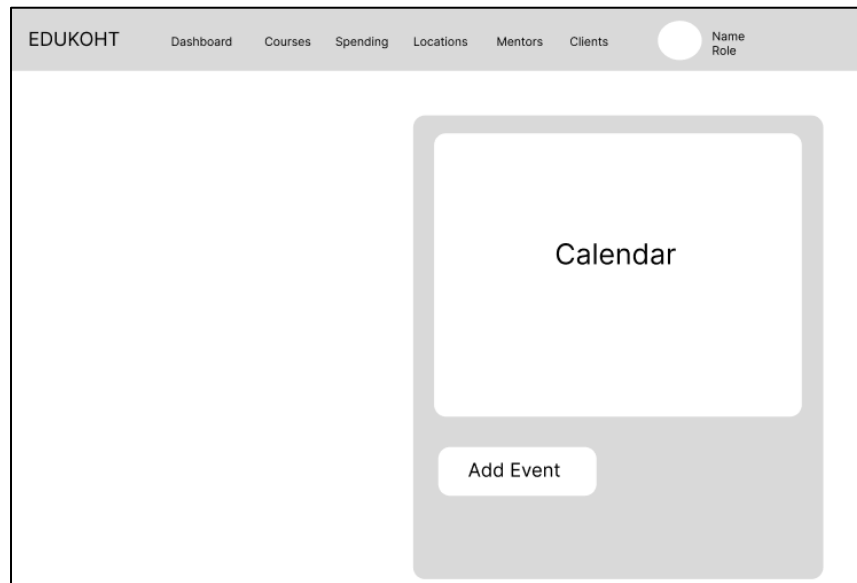


Figure 19. Dashboard wireframe

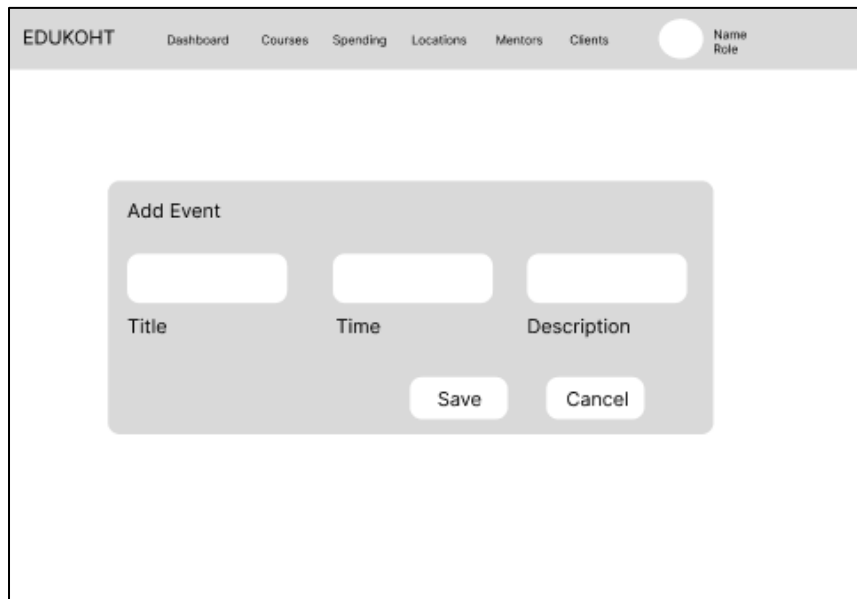


Figure 20. "Add event" form wireframe

Course Management

The course page features cards for each course, allowing easy navigation through available training programs (Figure 21-23). Individual course pages provide detailed information and functionality for managing modules and lessons, ensuring flexibility in administering course content.

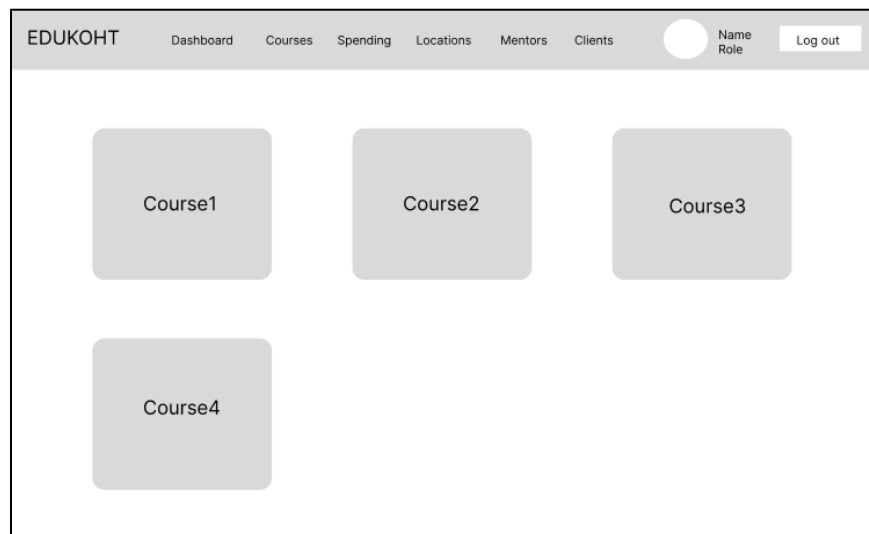


Figure 21. Courses page wireframe

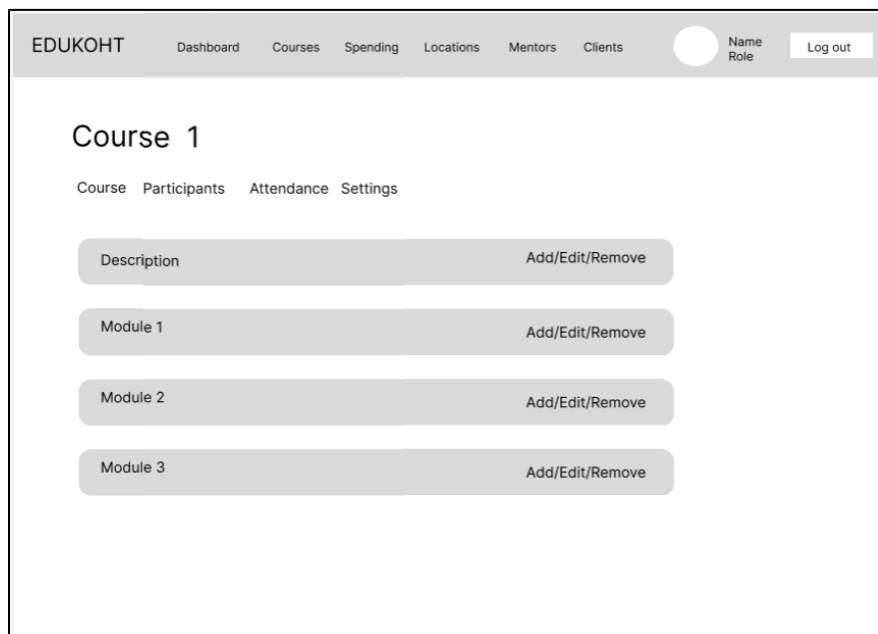


Figure 22. Course page wireframe

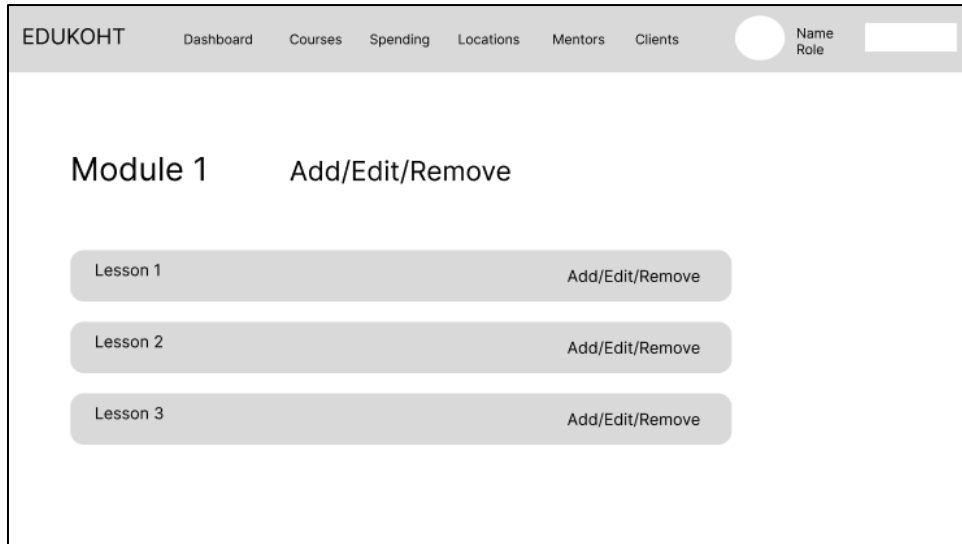


Figure 23. Course’s module page wireframe

Mentor Management

The mentor management page facilitates easy handling of information about instructors, including the ability to add, edit, and delete mentors (Figure 24). This is crucial for maintaining up-to-date information about instructors and their availability to students.

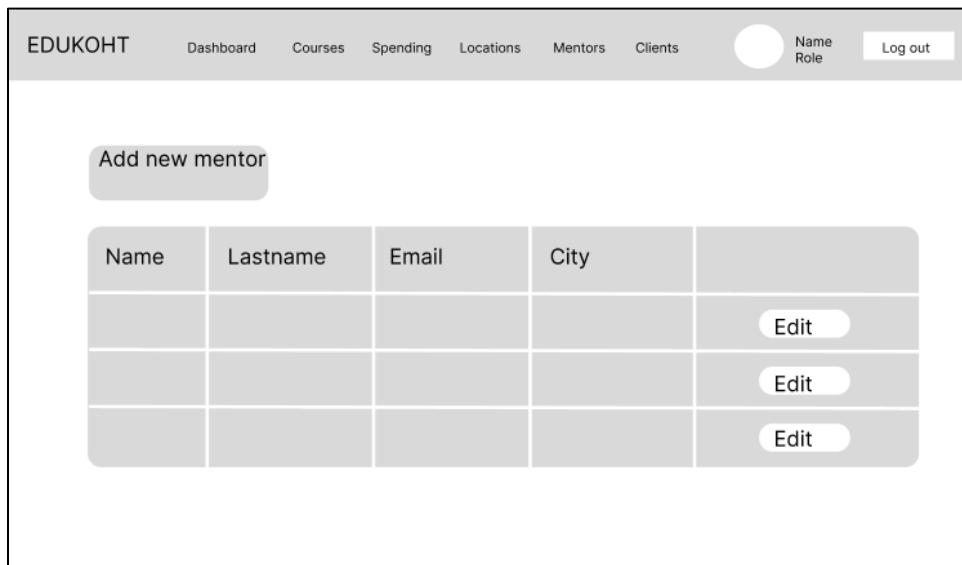


Figure 24. Mentor page wireframe

3.5 Technical Solution

As the author still managed to implement some of the functionalities, this chapter briefly describes the technical solution.

3.5.1 Database

The project uses SQLite as the database management system, providing convenience and simplicity in setup, making it ideal for development and testing. The configuration and management of the database are carried out through the Sequelize ORM (Object-Relational Mapping) tool, which allows developers to abstract database interactions using an object-oriented approach.

3.5.2 Back-end

The server-side of this project handles request processing, data management, and interactions between various services. It was developed using the Node.js platform and the Express.js framework. Detailed descriptions of these platforms and the reasons for their use in the project can be found in chapter 2.3.

The project consists of several key modules, each responsible for a specific functionality of the system. All server logic is organized according to the principles of "Clean Code," ensuring clear structure and ease of maintenance.

Project Structure:

models - contains data models that define the structure of SQLite database tables and methods for data processing.

sqlitedb - contains configuration files for managing the SQLite database and setting up the local development environment.

jwtProvider.js - responsible for implementing authentication and security mechanisms using JSON Web Tokens (JWT).

routes.js - contains all API endpoints for each service.

Server.js - the main server file, which is the entry point of the application and performs server initialization.

.env - contains environment variables for application configuration.

The architecture is based on the "router-controller-model" pattern, which ensures a clear separation of responsibilities between different system components. Each request passes through a router that directs it to the appropriate controller, which handles the request logic and interacts with the database through models.

3.5.3 Front-end

The front-end of our solution employs a modular approach, utilizing reusable components to streamline development and ensure consistency across the user interface. These components, such as login and registration buttons, are designed to be flexible and adaptable to various parts of the application, enhancing the user experience and reducing the overall development time.

For example, our login page features standardized buttons for user authentication processes, which are not only visually consistent but also functionally uniform across different scenarios. Detailed code snippets of these reusable components, demonstrating their implementation and integration, can be found in Appendix 2, which showcases the login and registration button example, and Appendix 3, which details the registration form structure.

By leveraging such reusable components, we ensure that the application maintains a cohesive look and feel, while also supporting scalability and maintainability of the codebase.

3.5.4 Some technical aspects of the solution

Routing

Routing in the project is implemented in the `routes.js` file, which defines all API endpoints and handles incoming requests by directing them to the appropriate controllers (Figure 25). This allows for a clear structure of requests and their handling logic.

```
router.get('/events', passport.authenticate("jwt", { session: false })), (req, res) => {
  Event.getAllEvents((err, data) => {
    if (err) {
      res.status(500).send(err.message);
    } else {
      res.status(200).send(data);
    }
  });
});

router.post('/events', (req, res) => {
  const { title, start, end, description } = req.body;
  Event.createEvent(title, start, end, description, (err, id) => {
    if (err) {
      res.status(500).send(err.message);
    } else {
      res.status(201).send({ id, title, start, end, description });
    }
  });
});
```

Figure 25. Session routing in `routes.js` file

Authentication

The `jwtProvider.js` module is responsible for implementing authentication and security mechanisms using JSON Web Tokens (JWT). This ensures system security by verifying and validating user sessions.

Main server file

The main server file, `server.js`, is the entry point of the application, where all components are integrated and configured (Figure 26). Here, server initialization occurs, connecting all necessary middleware for parsing JSON, managing sessions, and handling errors, ensuring smooth server operation. This file configures the server, connects all necessary middleware, and launches the server, listening for incoming requests on the specified port.

```
const express = require("express");
const cors = require("cors");
const jwtProvider = require('./jwtProvider');
const routes = require('./routes');
const {passport} = require("./jwtProvider");

require('dotenv').config();

const app = express();
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(express.json());
app.use(passport.initialize());

app.use('/', routes);

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Figure 26. 'server.js' file example

3.6 Testing and Validation

This chapter details the methods and outcomes of testing processes.

To ensure the operational integrity and reliability of the software, manual testing was extensively conducted. This involved structured walkthroughs and scenario-based testing to simulate real-world usage, ensuring that all parts of the system functioned correctly under varied conditions. Manual testing was pivotal in identifying and rectifying unforeseen errors and usability issues.

Feedback was exclusively collected from the client during the testing phase. The client conducted thorough reviews and provided insights that were instrumental in the final tuning of the system. Feedback from client confirmed that the solution meets all the functional requirements at this project stage.

4 Analysis of Results

This chapter presents a comprehensive analysis of the results obtained from the implementation of the project. It evaluates the performance and effectiveness of the technological solutions.

4.1 Technology and Tools

As detailed in Chapter 2.3 (Technology Description), the technology stack for the project was chosen with the necessity of flexibility and scalability in mind. In this context, Node.js was selected for the server side of the project due to its asynchronous, event-driven architecture, which is ideal for developing scalable network applications. Currently, only a portion of the server capabilities have been implemented, which allowed for the testing of asynchronous data processing and basic functions under controlled conditions. This decision provided the opportunity to dynamically add new features and services as the project developed.

At the front end, the Vue.js framework combined with Bootstrap was chosen based on recommendations outlined in the same chapter. Vue.js is noted for its incremental adaptability and modularity, allowing it to be used in both small and large-scale projects. At present, only the basic framework of the interface has been developed, which includes a few basic user interactions. This allows for flexible responses to changes in requirements and user feedback in the early stages of development. Bootstrap added ergonomics and ease of use to the interface, which is critically important for gathering initial consumer feedback.

During the interface layout development process, Figma was used, as described in Chapter 2.3. This tool allowed for quick creation and modification of design in line with the project's evolving needs. Working in Figma facilitated effective interaction between developers and designers, which is especially important at a stage when the product has not yet been released and significant revisions are forthcoming.

Thus, although the full implementation of the project and the product launch are not yet complete, the choice of technologies, based on the analysis in Chapter 2.3, allows for the efficient development of the project, quickly adapting to changing requirements and responding to preliminary user feedback. This ensures the creation of a flexible and scalable system capable of gradually integrating new features and improvements.

4.2 Project Workflow

Throughout the project, several conclusions were drawn that significantly impacted its progression. Initially, the project did not progress as planned due to the absence of strict deadlines and a structured workflow, hindering the effective completion of tasks. This lack of structure not only delayed project milestones but also affected the overall scope of functionalities that could be achieved.

To address these issues, we adopted Jira for task planning and resource allocation. Jira facilitated the structuring of workflows and organization of tasks, which improved the project's overall efficiency. Additionally, GitHub was utilized to monitor code changes effectively, allowing for better version control and collaboration across the development team.

Despite the technological tools in place, the project faced several challenges in terms of feedback collection and requirements validation. The feedback process primarily involved the client, and while this ensured that the project adhered to their specifications, it limited broader input from potential end users. This was a significant drawback as user feedback is crucial for tailoring solutions that genuinely meet user needs and enhance usability.

The development process was reviewed to determine if it was effectively aided by these tools. While Jira and GitHub provided substantial assistance in managing tasks and code respectively, the overall benefit to the project needed to be critically assessed. In retrospect, while these tools helped in some areas, the absence of a more comprehensive feedback system involving end users was a missed opportunity for gathering invaluable insights.

In conclusion, while the project met the basic requirements set forth by the client, the lack of structured deadlines initially and limited user feedback were major challenges. Future projects should consider these as areas for improvement, ensuring that deadlines are clearly defined from the beginning and that a mechanism for extensive user feedback is integrated into the project workflow to enhance the relevance and usability of the solution.

4.3 Requirements

The requirements for the developed system were formulated to address specific educational and administrative needs while considering the feedback from stakeholders to optimize functionality and user satisfaction.

Positive Aspects:

Improved Access to Information:

The system centralizes all educational materials and resources, making them easily accessible to students in one location. This centralization aims to simplify the learning process and enhance the interaction between students and teachers, thus contributing positively to the educational environment.

Automation of Educational Processes:

One of the key requirements was the automation of routine educational processes. The system integrates features such as course enrollment, performance tracking, and task distribution. These automated processes are designed to minimize manual labor and reduce the occurrence of errors, thereby increasing the efficiency of administrative procedures and allowing educators more time to focus on teaching and student interaction.

Risks and Challenges:

Integration Process:

Integrating the new system into the existing infrastructure poses significant challenges. There are concerns about potential technical errors that might hinder the platform's

functionality, especially during the initial period after launch. Mitigating these risks requires careful planning and testing to ensure compatibility and smooth transition from old systems.

Resistance to Change:

Another critical requirement was addressing the potential resistance from users accustomed to the previous system. The change management strategy includes comprehensive training sessions, user support, and gradual implementation phases to ease the transition. Ensuring user adoption is crucial for the new system to be effective and for it to enhance the overall educational experience.

The requirements for the new system were designed to leverage technological advancements to enhance educational outcomes and administrative efficiency. However, the potential risks associated with integration and user adoption are recognized as significant factors that could impact the success of the implementation. These elements will need to be managed carefully to ensure the system delivers its intended benefits without disrupting the existing educational processes.

4.4 Implemented Functionalities

This chapter provides an overview of the key features that have been successfully integrated into the system.

4.4.1 Welcome Page

The Welcome Page (Figure 27) serves as the entry point to the application and offers users two main options: logging in and registering. The page design was implemented with minimalism, centering the key controls for user convenience.

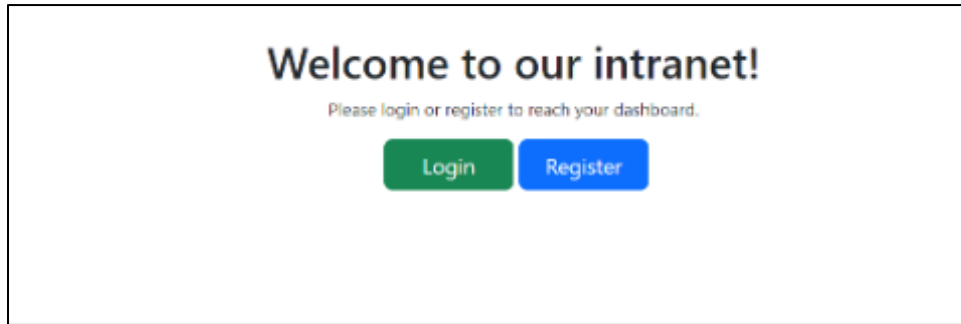


Figure 27. Welcome page

4.4.2 Login and Registration Pages

The Login Page (Figure 29) and Registration Page (Figure 28) are designed for authentication and registration of new users, respectively. Both pages follow a unified style and design, providing clean forms with a minimal set of fields: username and password. Vue.js and Bootstrap have facilitated data form validation and handling.

Figure 28. Register page

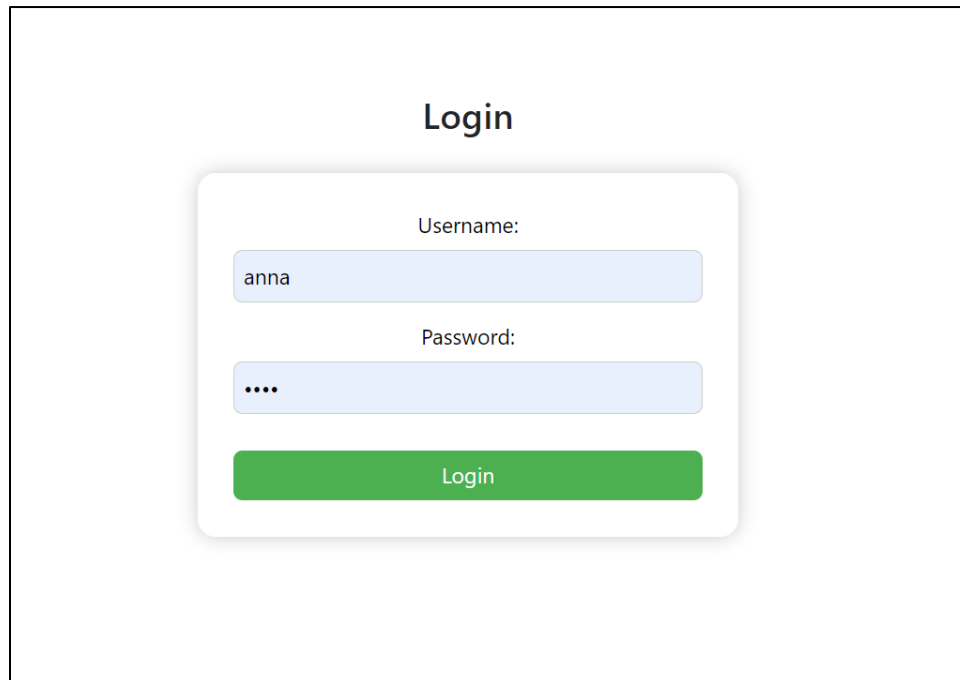


Figure 29. Login page

4.4.3 Calendar

In the EDUKOHT project (Figure 30), the calendar is a key component of the user interface, designed for managing and scheduling events. The calendar integration was accomplished using the Schedule-X.dev library [15]. This library provides the flexibility to display the calendar in various views: daily, weekly, and monthly, as illustrated in Figure 31.

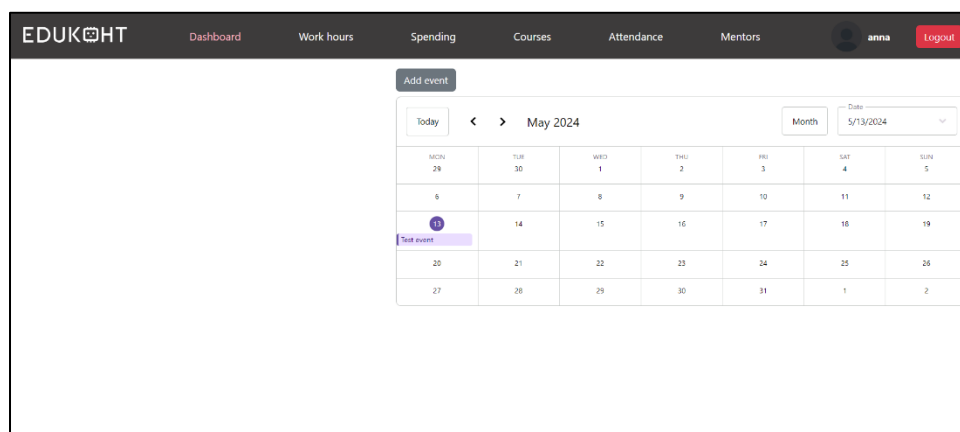


Figure 30. Dashboard page with calendar

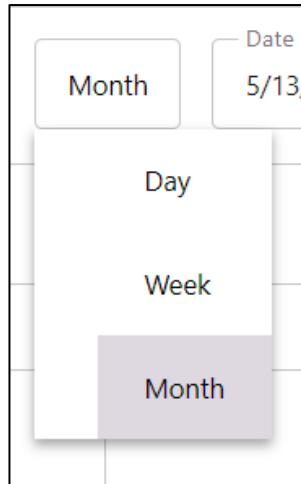


Figure 31. Calendar views

Adding a new event to the calendar (Figure 32) is critically important for the usability of the application. The event addition form allows users to input details such as the title, date, start and end times, and a description. After filling out the form and pressing the "Save" button, the event is added to the calendar and becomes visible (Figure 33). Hovering over an event in the calendar displays additional information about it. The form is developed using the Vue.js framework, ensuring its integration with the calendar and the instant display of added events.

The implementation of the calendar functionality and the event addition form using Schedule-X and Vue.js significantly simplifies the scheduling and time management process for users, enhancing the functionality and convenience of the application for daily use.

New event

Title: Chose the date: Start time: Ending time: Description:

Cancel Save

Figure 32. New event form

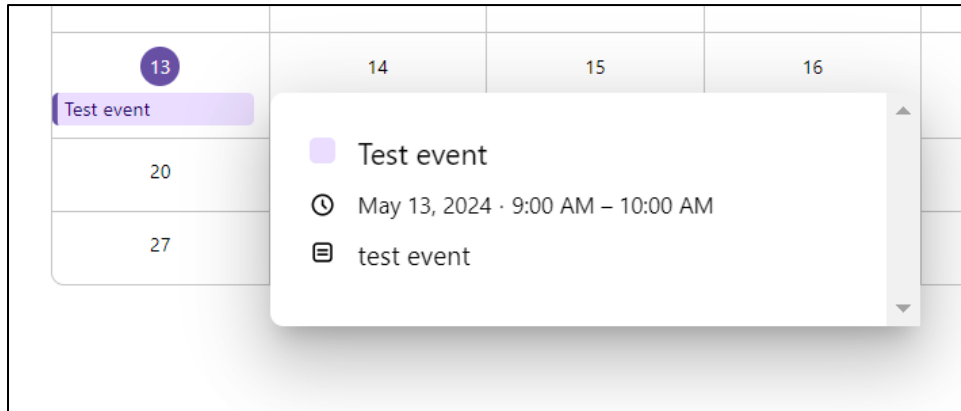


Figure 33. Event information

4.4.4 Mentor Page

As part of the project, a "Mentors" page was also implemented, which allows managing information about mentors in the educational environment. This page includes the following functionalities: viewing a list of mentors, adding, removing, and editing already saved mentors.

Viewing the list of mentors (Figure 34) is presented on the "Mentors" page as a list of all mentors with basic information about each, including their first name, last name, email, and city. The interface provides buttons for editing and deleting each mentor and the ability to add a new mentor.

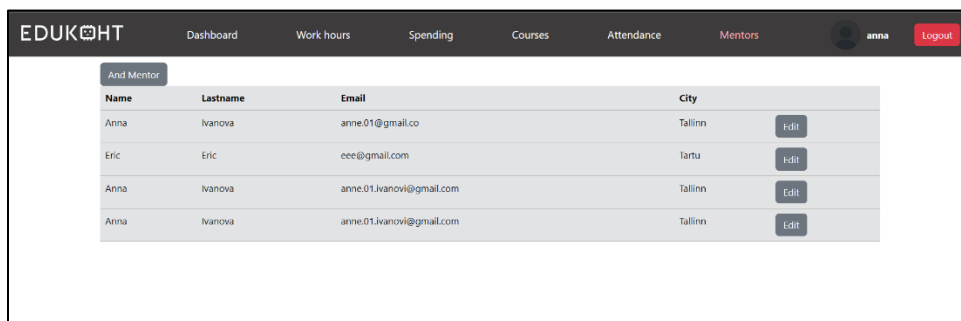


Figure 34. Mentor page

Using the "Create New Mentor" form (Figure 35), users can enter details for a new mentor, such as first name, last name, email, and city. After filling out the form, pressing the "Save" button adds the mentor to the system, allowing them to participate in the educational process.

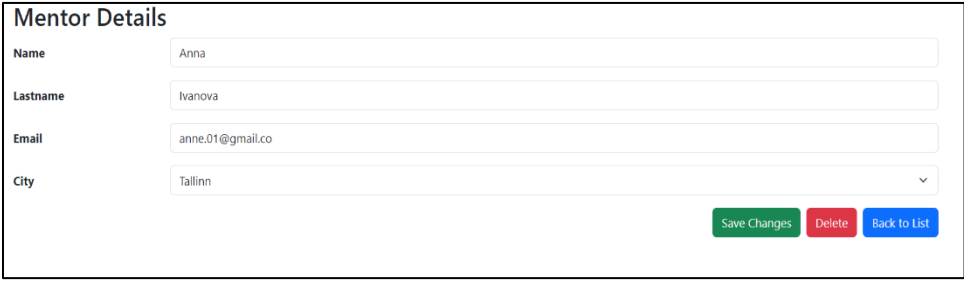
When selecting a mentor from the list on the "Mentors" page, the "Mentor Details" form opens, where existing information can be changed. The form allows updating the mentor's data, including their first name, last name, email, and city. Changes are saved in the system upon pressing the "Save Changes" button (Figure 36).

The "Mentors" page interface also includes a function for removing a mentor. This is done by pressing the "Delete" button next to each mentor in the list. The deletion of a mentor from the system occurs with confirmation of the action to avoid accidental removal.



The screenshot shows a form titled "Create New Mentor". It has four input fields: "Name", "Lastname", "Email", and "City". The "City" field is a dropdown menu. At the bottom right, there are two buttons: "Submit" (green) and "Cancel" (red).

Figure 35. Create new mentor form



The screenshot shows a form titled "Mentor Details". It has four input fields: "Name" (Anna), "Lastname" (Ivanova), "Email" (anne.01@gmail.co), and "City" (Tallinn). At the bottom right, there are three buttons: "Save Changes" (green), "Delete" (red), and "Back to List" (blue).

Figure 36. Mentor details page

4.5 Testing and Validation

Testing and validation of the technical solution were initially conducted manually, which provided a basic level of assurance that the system met its design specifications. However, to enhance the quality of the testing process, it is recommended to integrate automated testing (unit tests) to ensure thorough coverage and identify defects more efficiently.

Additionally, mockups were used to validate the system with potential users to identify any usability issues. Despite this, there remains a need for more robust user feedback to assess whether the requirements truly address user problems. Implementing more comprehensive user testing, including larger focus groups and usability tests, will help in obtaining more detailed feedback, thereby ensuring that the system is not only functional but also user-friendly and effective in solving the problems it is designed to address.

This dual approach of integrating both automated and manual testing, along with extensive user involvement, will significantly enhance the reliability and usability of the system, providing a more accurate assessment of its performance in real-world scenarios.

4.6 Further Work

Based on the analysis, it can be assumed that the implementation of the proposed solution could bring significant improvements to the learning process at EDUKOHT school. However, to minimize potential risks, the following is recommended:

1. **Conducting Testing:** Organize system testing at all stages of development to identify and rectify technical defects early on.
2. **Training and Adaptation Programs:** Develop and implement user support programs to facilitate the transition to the new system.
3. **Regular Updates and Testing:** Implement a schedule for regular updates and thorough testing to ensure the platform remains functional and user-friendly.
4. **User Feedback Integration:** Collect and integrate feedback from users to continuously improve the platform and meet their needs more effectively.

Summary

The goal of the project was to propose a solution for EDUKOHT programming school intranet.

The expected results were :

1. New intranet system requirements
2. Mock-ups
3. Some implemented features

During the work, the following results were achieved. Firstly, the functional requirements were formulated based on the analysis of previously used platforms. A use case diagram was developed, which included three main roles: administrator, mentor, and client. It is important to note that the project is still under development, and the requirements may change or be supplemented in the future.

Secondly, user interface mockups were created in Figma. A dashboard mockup with a calendar was created, as well as mockups for the course page, mentor page, and forms for adding events and mentors.

Thirdly, some components defined earlier as functional requirements were partially implemented. The author developed the functionality to register and log in to the platform using a username and password. Additionally, a calendar was implemented, allowing the addition and viewing of events. A mentor page was developed with a table containing information about each mentor, such as first name, last name, email, and the city they work in. The ability to view detailed information about a mentor on a separate page, where data can also be edited, was implemented. The functionalities to add new mentors and delete existing mentors were also realized.

Author described future work and made suggestions on the project further implementation. By implementing those suggestions, the platform can evolve to better meet its objectives and provide enhanced value to its users.

References

- [1] “EDUKOHT” official website [Online material]. Available: <https://www.edukoht.ee/>. [Used: 06.03.2024].
- [2] “What is an LMS? Learning management systems explained”, 31.012022 [Online material]. Available: <https://moodle.com/news/what-is-an-lms-learning-management-systems-explained/>. [Used: 06.03.2024].
- [3] Figma Design [Online material]. Available: <https://www.figma.com/design/>. [Used:06.03.2024]
- [4] Vue.js, “The Progressive JavaScript Framework” [Online material]. Available: <https://vuejs.org/> [Used: 06.03.2024]
- [5] Node.js, “Run JavaScript Everywhere” [Online material]. Available: <https://nodejs.org/en> [Used:14.03.2024]
- [6] git Documentation [Online material]. Available: <https://git-scm.com/doc> . [Used:14.03.2024]
- [7] GitHub,” GitHub Docs” [Online material]. Available: <https://docs.github.com/en> . [Used:14.03.2024]
- [8] Jira, “Welcome to Jira” [Online material]. Available: <https://www.atlassian.com/software/jira/guides/getting-started/introduction> [Used: 10.04.2024]
- [9] “Getting started with WebStorm” 11.02.2024 [Online material]. Available: <https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html> [Used:20.04.2024]
- [10] “IntelliJ IDEA overview”, 19.04.2024 [Online material]. Available: <https://www.jetbrains.com/help/idea/discover-intellij-idea.html> [Used:20.04.2024]
- [11] Express.js [Online material]. Available: <https://expressjs.com/> [Used: 20.04.2024]
- [12] “About SQLite”, 10.10.2023 [Online material]. Available: <https://www.sqlite.org/about.html> [Used: 20.04.2024]
- [13] “Get started with Bootstrap” [Online material]. Available: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> [Used:17.05.2024]
- [14]”What is Use Case Diagram” [Online material]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> [Used:17.05.2024]
- [15] shedule-x [Online material]. Available: <https://schedule-x.dev/docs/calendar> [Used:17.05.2024]

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis¹

I Anna Ivanova

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Solution Proposal for the EDUKOHT Programming School Intranet”, supervised by Bahdan Yanovich
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

23.05.2024

¹ The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Login and registration buttons example

```
<template>
  <div
    class="home d-flex flex-column flex-grow-1 align-items-center
justify-content-center"
  >
    
    <h1 class="text-center">Welcome to our intranet!</h1>
    <p>Please login or register to reach your dashboard.</p>
    <div class="buttons d-flex flex-row">
      <router-link class="router-link-button" to="/login">
        <button class="btn btn-success btn-lg">Login</button>
      </router-link>
      <router-link class="router-link-button" to="/register">
        <button class="btn btn-primary btn-lg">Register</button>
      </router-link>
    </div>
  </div>
</div>
</template>
```

Appendix 3 - Registration form example

```
<template>
  <div class="register-form">
    <h2>Register</h2>
    <form @submit.prevent="register">
      <div>
        <label for="username">Username:</label>
        <input type="text" id="username" v-model="username" required />
      </div>
      <div>
        <label for="password">Password:</label>
        <input type="password" id="password" v-model="password" required
/>
      </div>
      <button type="submit">Register</button>
    </form>
  </div>
</template>
<script>
import { ref } from 'vue';
import { useAuthStore } from "@store/auth.store";
export default {
  setup() {
    const authStore = useAuthStore();
    const username = ref('');
    const password = ref('');

    const register = async () => {
      try {
        await authStore.register({ username: username.value, password:
password.value });
      } catch (error) {
        console.error('Registration failed', error);
      }
    };

    return {
      username,
      password,
      register,
    };
  }
}
</script>
```