Tartu University

Faculty of Social Studies

Narva College

Study program “Information Technology Systems Development”

Ilia Bogatyrev

DEVELOPMENT OF AN APPLICATION FOR REGISTERING FOR RETAKING OF WRITTEN WORK AT IDA-VIRUMAA VOCATIONAL EDUCATION CENTRE.

Bachelor thesis

Supervisor: Andre Sääsk, M.Sc.

Narva 2024

Tartu Ülikool

Sotsiaalteaduste valdkond

Narva Kolledž

Õppekava „ Infotehnoloogiliste süsteemide arendus “

Ilia Bogatyrev

IDA-VIRUMAA KUTSEHARIDUSKESKUSE KIRJALIKU TÖÖDE JÄRELEVASTAMISELE REGISTREERIMISE RAKENDUSE ARENDAMINE.

Bakalaureusetöö

Juhendaja: Andre Sääsk, M.Sc.

Narva 2024

TABLE OF CONTENTS

RESÜMEE

Käesolev lõputöö käsitleb rakenduse arendamist, mis on mõeldud Ida-Virumaa Kutsehariduskeskuse (IVKHK) kirjalike tööde järeletegemise registreerimise protsessi haldamiseks. Rakenduse eesmärk on leevendada olemasolevaid administratiivseid probleeme, mis tulenevad praegusest käsitsi hallatavast süsteemist, mis kasutab Microsoft Forms'i registreerimiseks ja käsitsi koostatud nimekirju osalemise jälgimiseks.

Arendatud rakendus pakub kasutajasõbraliku ja efektiivse lahenduse, mis võimaldab hõlpsasti registreeruda järeltöödele, valida ajad ja kohad ning toetab sujuvat suhtlust ja järeltööprotsessi haldamist õpilaste, õpetajate, klassijuhatajate ja järeltööde korraldajate vahel. Üks rakenduse peamisi omadusi on QR-koodide kasutamine õpilaste kohaloleku kinnitamiseks, mis kõrvaldab vajaduse käsitsi nimekirjade haldamiseks ja vähendab vigade tekkimise võimalust.

Rakenduse arendamisel kasutati MEVN-tehnoloogiapakki (MongoDB, Express.js, Vue.js, Node.js), mis tagab süsteemi paindlikkuse, skaleeritavuse ja hooldatavuse. Valitud tehnoloogiad võimaldavad arendada jõudlusele ja skaleeritavusele keskendunud lahendust. MongoDB valik andmebaasiks tagab andmete haldamise ja töötlemise paindlikkuse ja efektiivsuse.

Olulised funktsioonid, nagu QR-koodi integreerimine kohaloleku kinnitamiseks, kasutajasõbralikud liidesed sündmuste ajastamiseks ja sujuv kasutajate registreerimine Microsoft Entra autentimise kaudu, on hoolikalt kavandatud ja rakendatud. Need funktsioonid lihtsustavad administratiivset töökoormust ja parandavad kasutajakogemust nii õpilaste, õpetajate kui ka administraatorite jaoks.

Rakenduse juurutusstrateegia, mis kasutab Dockerit konteineriseerimiseks ja DigitalOceani majutamiseks, koos CI/CD torujuhtmetega GitHub Actions'i kaudu, tagab sujuva ja tõhusa juurutusprotsessi. Nginx ja Cloudflare integratsioon suurendavad rakenduse turvalisust, jõudlust ja skaleeritavust, muutes selle hästi varustatud süsteemiks IVKHK administratiivsete protsesside haldamiseks.

Kokkuvõttes on käesolev lõputöö edukalt arendanud ja juurutanud rakenduse, mis mitte ainult ei vasta IVKHK vajadustele, vaid loob aluse tulevastele täiustustele ja laiendustele. Innovatiivne kaasaegsete veebitehnoloogiate kasutamine ja kasutajakeskne disain

lähenemine rõhutavad tehnoloogia potentsiaali hariduse administratiivsete protsesside muutmisel, luues efektiivsema, täpsema ja kasutajasõbralikuma süsteemi haridusasutustes.

INTRODUCTION

In the rapidly evolving field of educational technology, the efficiency of administrative processes significantly influences both the quality of education and the operational ease within educational institutions. This thesis identifies and addresses a critical challenge encountered by the Ida-Virumaa Vocational Education Centre (IVVEC) - the lack of an effective system for managing the retaking process of written work. The current system, which heavily relies on Microsoft Forms for registration, followed by manual compilation and printing of lists, is not only cumbersome but also prone to errors. This method complicates the monitoring of student attendance for retakes, leading to an increased administrative burden and potential inaccuracies in tracking student participation.

The absence of a streamlined process for these administrative tasks results in confusion among students and educators, unnecessary delays, and an overall increase in the administrative workload. The need for a dedicated application to manage this process efficiently has become apparent, representing a significant step towards enhancing the operational infrastructure of educational institutions.

The goal of this thesis is to develop a functional application specifically designed for IVVEC that facilitates the easy registration for retakes, selection of times and locations, and supports seamless communication and management of the retake process for students, teachers, class heads, and retake organizers. A key feature of this application will be the implementation of a QR code solution to simplify the verification of student attendance at retakes, thereby eliminating the need for manual list management and reducing the potential for errors.

To accomplish this objective, the following tasks have been outlined:

1. Design and develop an application that addresses IVVEC's unique requirements for managing the retake process of written work, with a focus on streamlining administrative tasks and improving accuracy in student attendance tracking.
2. Incorporate critical features and interfaces, including a QR code-based attendance verification system, to engage students, teachers, class heads, and retake organizers effectively.
3. Ensure the application’s security and seamlessly integrate it with Microsoft Entra for authentication, providing a reliable and secure platform for all users.

This thesis will leverage the MEVN technology stack (MongoDB, Express.js, Vue.js, Node.js) for the development of the application and will adhere to agile development

methodologies to ensure flexibility and efficiency throughout the development process. The integration with Microsoft Entra will address critical aspects of security and user authentication, ensuring the application's reliability and institutional viability.

The anticipated outcome of this work extends beyond a mere technological solution; it aims to significantly enhance the administrative operations of IVVEC, leading to a more efficient and streamlined educational process. By implementing this initiative, the thesis seeks to establish a benchmark for the development of similar educational tools that can effectively address administrative challenges and contribute to the enhancement of the educational landscape.

1 EXISTING SOLUTIONS ON THE MARKET

In the contemporary educational landscape, the fusion of technology and administrative processes has become a cornerstone for enhancing both the efficiency and effectiveness of learning institutions. This chapter delves into the pivotal role of Learning Management Systems (LMS) and other educational technologies in streamlining administrative tasks, with a special focus on managing student attendance and the retaking of assessments. The significance of this research lies in its potential to transform the administrative hurdles currently faced by the Ida-Virumaa Vocational Education Centre (IVVEC) into streamlined processes that support both educators and learners.

The advent of digital solutions presents an opportunity to address the manual and time-consuming tasks associated with the retake process and attendance tracking. Yet, despite the availability of numerous tools and platforms, gaps remain in their ability to fully cater to the specific needs of vocational education centres like IVVEC. This chapter seeks to explore existing technologies within this domain, namely Moodle, to understand their capabilities, limitations, and how they align with the needs of IVVEC.

Our investigation is guided by several key objectives:

1. **Examine Existing Solutions:** By analysing current tools and technologies, particularly Moodle, we aim to understand the landscape of available solutions for educational administrative management.

2. **Identify Gaps and Shortcomings:** This involves pinpointing the limitations of these tools in the context of managing retakes and student attendance efficiently at IVVEC.

3. **Propose Solutions to Address Identified Needs:** Based on the gaps identified, this research will explore modifications, customizations, or integrations that could enhance Moodle’s utility for IVVEC, focusing on simplifying the retake and attendance tracking processes.

Through a meticulous exploration of existing tools and an insightful analysis of their applicability and limitations, this chapter endeavors to pave the way for a tailored solution that meets IVVEC's unique requirements. The goal is not just to critique what exists but to envision practical enhancements that leverage technology for administrative ease, thereby

contributing to the broader discourse on educational technology's role in optimizing educational administration.

1.1 Learning Management Systems: Moodle

Learning Management Systems (LMS) are crucial in the digital transformation of educational processes, offering a robust platform for course management, content delivery, and various administrative tasks. Among these, Moodle stands out as a widely adopted and highly flexible open-source LMS. Let's delve into Moodle's capabilities, focusing on how it can be leveraged for managing student attendance, registrations, and particularly for organizing retake exams.

1.1.1 Moodle: An Overview

Moodle (Modular Object-Oriented Dynamic Learning Environment) is an open-source learning platform designed to provide educators, administrators, and learners with a single robust, secure, and integrated system to create personalized learning environments. It is used worldwide by universities, schools, companies, and independent teachers (About Moodle - MoodleDocs, 2024). Moodle is highly customizable, supporting plugins and themes to extend its functionality and appearance.

1.1.2 Key Features Relevant to Attendance and Retake Management

Attendance Module: Moodle offers an Attendance module that allows teachers to mark student attendance and students to view their attendance records. This module can be customized for different class schedules and includes features for marking presence, absence, lateness, and excused absences.

Quiz and Assignment Modules: For managing retakes, Moodle's Quiz and Assignment modules can be configured for multiple attempts, allowing students to retake quizzes or submit assignments more than once. This feature is useful for organizing retake exams or additional coursework submissions.

Booking Module: Although not part of Moodle's core modules, the Booking module can be added to enable students to register for classes, exams, or events. This module could be adapted to manage registrations for retake exams, allowing students to select time slots and locations for their retakes.

Calendar and Scheduling: Moodle's Calendar feature helps in scheduling exams, classes, and meetings. It can be used to inform students about retake dates and deadlines, integrating with the Booking module for a seamless registration process.

1.1.3 Addressing the Gap

While Moodle offers extensive features for course management and learning activities, its use for specifically managing the retake process, especially with a focus on simplifying attendance verification and student registration for retakes, may require additional customization or the integration of external plugins. The current challenge at IVVEC highlights the need for a more streamlined and automated process for managing retakes, including the efficient tracking of student attendance.

1.1.4 Proposed Solution with Moodle

To address the identified gaps, particularly the efficient management of retake exams and attendance verification, a combination of Moodle's core and additional modules, along with custom development, could be considered. For example, integrating a QR code attendance system directly within Moodle could automate attendance tracking, linking student presence at retakes with their academic records in Moodle. Furthermore, developing a custom module or adapting the Booking module for retake registrations could streamline the process, making it more accessible for students and easier to manage for administrators and educators.

1.1.5 Conclusion

Moodle's flexibility and the vast ecosystem of plugins make it a potentially viable platform for addressing IVVEC's challenges with managing retake exams and student attendance. However, achieving the desired level of functionality may require exploring additional plugins or custom development to tailor the system to IVVEC's specific needs.

1.2 MyAttendanceTracker

MyAttendanceTracker is an online tool designed to simplify the process of tracking attendance for teachers, schools, and organizations. This platform is especially beneficial for educators looking to move away from traditional paper-based methods or complex spreadsheets for attendance tracking. Its user-friendly interface and online accessibility make

it a convenient choice for managing attendance in a variety of educational settings (Ohannessian, n.d.).

1.2.1 Key Features Relevant to Attendance Management

Online Attendance Recording: MyAttendanceTracker allows teachers to record attendance for their classes through an online interface. This eliminates the need for manual paperwork and makes attendance data accessible from anywhere, at any time.

Student Accounts: Students can have their own accounts, where they can log in to check their attendance records, view class schedules, and receive notifications for upcoming classes or activities. This feature enhances transparency and encourages students to be more accountable for their attendance.

Customizable Reports: The platform offers the ability to generate customizable reports based on attendance data. These reports can provide insights into attendance patterns, identify students with frequent absences, and help in making informed decisions regarding student performance and participation.

Email Notifications: MyAttendanceTracker supports automatic email notifications to students and/or their guardians for absences or reminders for upcoming classes. This feature aids in improving communication between teachers and students or parents, ensuring all parties are informed about attendance-related matters.

1.2.2 Addressing the Gap

While MyAttendanceTracker provides a robust solution for tracking attendance, its focus is primarily on the recording and reporting of attendance data. For the specific challenge faced by IVVEC - managing the retake process for written work, including registrations for retakes and verifying student attendance at these events - MyAttendanceTracker may not offer direct functionalities. The platform is centered around attendance management rather than the comprehensive organization of exams, retakes, and specific academic events.

1.2.3 Proposed Solution with MyAttendanceTracker

To integrate MyAttendanceTracker into a solution for IVVEC’s needs, it could be used in tandem with other systems or custom solutions that handle exam and retake registrations.

MyAttendanceTracker could specifically manage the attendance verification aspect for retakes, ensuring accurate and efficient tracking of which students attend their scheduled retake exams. However, to manage the entire retake process effectively - from registration to scheduling and communication - additional tools or custom developments would be necessary. For example, developing an interface or integration that connects MyAttendanceTracker's attendance data with a scheduling system for retakes could offer a more seamless solution, providing a comprehensive approach to managing retakes, including both registration and attendance verification.

1.2.4 Conclusion

MyAttendanceTracker offers valuable functionalities for attendance management that could play a role in a larger ecosystem of tools designed to streamline IVVEC's retake process. However, as a standalone solution, it would need to be part of a more extensive system that includes functionalities for scheduling, registering, and managing retake exams. Custom integrations or the addition of complementary tools would be essential to address all aspects of the challenge, ensuring a holistic solution that meets IVVEC's specific needs.

1.3 ExamSoft: A Comprehensive Examination and Assessment Platform

ExamSoft is a sophisticated software solution designed to streamline the entire assessment process, from exam creation to administration, grading, and analysis. It serves educational institutions, professional organizations, and certification providers, offering a secure and efficient platform for managing both in-person and remote examinations. ExamSoft's capabilities extend beyond mere assessment creation, encompassing comprehensive tools for academic integrity, data analysis, and student performance tracking (About ExamSoft | Innovative Assessment Software, n.d.).

1.3.1 Key Features Relevant to Examination Management

Secure Exam Environment: ExamSoft provides a secure testing environment that minimizes the risk of cheating and content theft. This is achieved through features like lockdown browsers and device control, ensuring that assessments are conducted with integrity.

Customizable Exam Creation: Educators can create diverse types of assessments, including multiple-choice questions, essays, and practical tasks. The platform supports a

wide range of question formats, enabling the creation of exams that accurately reflect course content and learning objectives.

Real-Time Performance Feedback: ExamSoft offers immediate feedback on student performance, allowing educators to identify areas where students struggle and adjust their teaching strategies accordingly. This feature enhances the learning and teaching experience by highlighting knowledge gaps and areas for improvement.

Advanced Data Analysis: The platform provides detailed analytics on exam performance, offering insights into trends, student progress, and the effectiveness of teaching methods. This data-driven approach facilitates informed decision-making and personalized student support.

Remote Proctoring Options: For institutions that conduct exams online, ExamSoft offers remote proctoring features that ensure the integrity of assessments. These include identity verification, audio and video monitoring, and anomaly detection, which help maintain high standards of academic honesty in a virtual environment.

1.3.2 Addressing the Gap

ExamSoft is primarily focused on the creation, administration, and analysis of exams and assessments, offering robust features for ensuring academic integrity and facilitating detailed performance analysis. However, when it comes to the specific challenges faced by IVVEC, such as managing registrations for retake exams and verifying student attendance at these events, ExamSoft's core functionalities may not directly address these needs. The platform excels in the secure administration of exams and analyzing results but does not inherently provide tools for scheduling retakes or handling logistical aspects of exam retake processes.

1.3.3 Proposed Solution with ExamSoft

To leverage ExamSoft in addressing IVVEC's challenges, it could be integrated with additional systems or processes designed for managing the logistical aspects of retakes. For instance, while ExamSoft can ensure that retake exams are conducted securely and efficiently, separate systems might be needed for scheduling retakes, registering students, and verifying attendance. Custom integrations or the use of complementary tools, such as scheduling software or attendance tracking systems, could provide a comprehensive

solution. These tools could manage the administrative side of retakes, while ExamSoft would handle the secure administration and analysis of the retake assessments themselves.

1.3.4 Conclusion

ExamSoft offers a powerful platform for managing the assessment aspect of retakes, ensuring academic integrity, and providing valuable insights into student performance. However, to fully address IVVEC's needs around managing retake exams, ExamSoft would need to be part of a broader ecosystem of tools. This ecosystem would handle the end-to-end process of retakes, from registration and scheduling to conducting and analyzing the exams. By integrating ExamSoft with other systems tailored to IVVEC's specific requirements, a holistic approach to the retake process could be achieved, enhancing efficiency and streamlining administrative tasks.

2 OVERVIEW OF WEB DEVELOPMENT TECHNOLOGIES

The landscape of web development technologies is vast and continuously evolving, offering an array of tools, frameworks, and languages designed to meet the diverse needs of digital projects ranging from simple websites to complex web applications. This rich ecosystem is divided into two main areas: front-end development, focusing on the user interface and experience, and back-end development, which handles the server-side logic and database interactions. The choice of technologies impacts not just the development process but also the efficiency, scalability, and security of the final application.

Web development technologies can be broadly categorized into several key areas, each playing a crucial role in building and maintaining web applications:

- Front-end Frameworks: These are libraries or frameworks used to build the user interface of web applications. They provide developers with a set of pre-written, standardized code to create interactive and dynamic web pages. Common front-end technologies include HTML, CSS, and JavaScript, along with frameworks like React, Vue.js, and Angular (Wallis, n.d.).
- Back-end Frameworks: Back-end frameworks are used on the server side to manage database operations, user authentication, and application logic. They often provide the scaffolding for creating the architecture of a web application, with popular options including Node.js with Express.js, Django (Python), and Ruby on Rails (Ruby).
- Databases: Essential for storing, retrieving, and managing data, databases are a critical component of web development. They can be classified into SQL (Structured Query Language) databases like MySQL and PostgreSQL, and NoSQL databases like MongoDB, each with its own set of advantages depending on the application's data requirements.
- Authentication Solutions: Security is paramount in web development, particularly for applications handling sensitive user information. Authentication technologies ensure that access to resources is granted only to authenticated and authorized users. Solutions range from traditional session-based authentication to modern token-based systems like OAuth, JWT (JSON Web Tokens), and secure identity services like Microsoft Entra.

The selection of technologies for a web development project is influenced by several factors, including the project requirements, developer expertise, community support, and the scalability and performance needs of the application. Each technology and framework offers unique features and benefits, making it essential to carefully evaluate them in the context of the project goals and constraints.

In the subsequent sections, we will delve deeper into these categories, examining the features, use cases, and comparative advantages of specific technologies. This exploration

will not only highlight the diversity of the web development ecosystem but also provide insights into the rationale behind the selection of the MEVN stack (MongoDB, Express.js, Vue.js, Node.js) for developing a robust and efficient application tailored to the needs of the Ida-Virumaa Vocational Education Centre.

2.1 Front-end Technologies

The realm of front-end development is where the visual and interactive elements of a web application come to life. It is the part of web development that interacts directly with users, encompassing everything they see and interact with within their web browser. This area of development relies heavily on a combination of languages and frameworks to create responsive, dynamic, and user-friendly interfaces (Ferreira, 2021). Here, we explore the core languages of front-end development and highlight some of the most prominent frameworks that have shaped the modern web.

2.1.1 Core Languages of Front-end Development

HTML (HyperText Markup Language): The backbone of any website, HTML provides the basic structure of pages, which is then enhanced and manipulated by CSS and JavaScript (HTML Tutorial, n.d.).

CSS (Cascading Style Sheets): CSS is used for styling HTML elements on the web. It controls the layout, colors, fonts, and overall appearance of a website, allowing for responsive designs that adapt to different screen sizes (What is CSS, n.d.).

JavaScript (JS): JavaScript brings web pages to life. It is used to create dynamic content, control multimedia, animate images, and pretty much everything else (What is JavaScript?, 2024). With JavaScript, developers can build rich interactive web applications.

2.1.2 Leading Front-end Frameworks

Frameworks and libraries have become essential in front-end development, providing developers with a set of pre-written code to speed up development and solve common issues. Here are some of the leading front-end frameworks and their key features:

- **React (by Facebook):** React is a declarative, efficient, and flexible JavaScript library for building user interfaces (What is React?, n.d.). It lets developers create large web

applications that can change data without reloading the page. Its main advantage is its component-based architecture, allowing for reusable UI components .

- **Vue.js:** Vue is a progressive JavaScript framework used for building UIs and single-page applications (Introduction | Vue.js, n.d.). It is designed from the ground up to be incrementally adoptable. The core library focuses on the view layer only, making it easy to pick up and integrate with other libraries or existing projects. Vue is also perfectly capable of powering sophisticated single-page applications when used in combination with modern tooling and supporting libraries (Merced, 2020).
- **Angular (by Google):** Angular is a platform and framework for building single-page client applications using HTML and TypeScript (Angular - What is Angular?, 2023). Angular provides developers with an efficient way to build desktop and mobile applications. It has a comprehensive solution that includes tools for everything from testing, animation, and UI components to a whole suite of HTTP services.

Each of these frameworks has its own strengths and is chosen for projects based on specific needs. React's component-based architecture is excellent for projects that require a dynamic interface with high user interaction. Vue.js is known for its simplicity and ease of integration, making it ideal for projects that need to be incrementally upgraded or for developers seeking a gentle learning curve. Angular offers a robust solution for enterprise-level applications, providing a wide range of features out of the box.

2.1.3 Justification for Choosing Vue.js

For the project at hand, Vue.js has been selected as the front-end technology of choice. This decision was driven by Vue's simplicity and its model of progressive enhancement. It offers a gentle learning curve without sacrificing the ability to scale for larger projects. Vue's component-based approach allows for the development of reusable and maintainable code, a crucial factor for the longevity and flexibility of the application. Additionally, Vue's ecosystem, including Vue Router and Vuex, provides a comprehensive solution for developing sophisticated single-page applications (SPAs), which aligns with the goals of developing an efficient and user-friendly application for the Ida-Virumaa Vocational Education Centre.

2.2 Back-end Frameworks

In the world of web development, back-end frameworks are the cornerstone of any robust web application. They serve as the backbone, handling the server-side logic, database interactions, database management, and integration with other systems. Back-end frameworks are responsible for the behind-the-scenes functionality that powers the user interface, making them indispensable for creating scalable, secure, and efficient web

applications. In this section, we explore the essentials of back-end development and highlight a selection of prominent frameworks that are pivotal in the industry.

2.2.1 The Role of Back-end Frameworks

Back-end frameworks provide the tools and libraries necessary for building the server-side of web applications. They offer pre-written, reusable code segments that handle common web development tasks including routing, session management, and authentication, significantly speeding up the development process. Moreover, they support various databases and ensure secure communication between the server and the client. The choice of a back-end framework can impact an application's performance, scalability, and ease of maintenance.

2.2.2 Leading Back-end Technologies

Several back-end technologies have emerged as leaders in the field, each with its own set of features, benefits, and use cases:

- **Express.js:** Built on top of Node.js, Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications (Express - Node.js web application framework, n.d.). It simplifies the server-side scripting process through its numerous HTTP utility methods and middleware, making it easier to create a strong API. Express is known for its performance, high test coverage, and being unopinionated, giving developers the freedom to build applications as they see fit.

- **Django:** A high-level Python Web framework that encourages rapid development and clean, pragmatic design (Django, n.d.). Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel (Stromer, n.d.). It's free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

- **Ruby on Rails (Rails):** Rails is a server-side web application development framework written in Ruby under the MIT License (Miller, 2021). It is a model-view-controller (MVC) framework, providing default structures for a database, a web service, and web pages. Rails emphasize the use of well-known software engineering

patterns and principles, such as convention over configuration (CoC), don't repeat yourself (DRY), and the active record pattern (Ruby on Rails, n.d.).

2.2.3 Justification for Choosing Express.js and Node.js

For this project, the back-end technology selected is Express.js, coupled with Node.js for the runtime environment. This decision is anchored in several key considerations:

- **Performance and Scalability:** Node.js offers a non-blocking I/O model that ensures efficient performance even under heavy loads, making it ideal for applications that require high concurrency. Express.js builds on this strength, offering a lightweight framework that doesn't bog down the application.

- **Unified JavaScript Development:** Using Node.js and Express.js allows for a JavaScript-only development environment, streamlining the development process by using a single language across both the front-end and back-end. This contributes to faster development cycles and easier maintenance.

- **Community and Ecosystem:** Node.js and Express.js benefit from a vast and active community, providing a wealth of resources, libraries, and tools that enhance development efficiency. This ecosystem ensures that developers have access to cutting-edge solutions and support for emerging technologies.

- **Flexibility:** Express.js is unopinionated, offering the flexibility to structure the application as needed without imposing rigid patterns. This allows for customized solutions that precisely meet the project's requirements.

By leveraging the strengths of Express.js and Node.js, the project aims to create a back-end infrastructure that is not only efficient and scalable but also cohesive with the chosen front-end technology, Vue.js, ensuring a seamless development process and a robust final product.

2.3 Databases

A fundamental component of any web application is its ability to store, retrieve, and manipulate data efficiently. This is where databases come into play, acting as the central repository for the application's data. Databases are classified into two main types: SQL (Structured Query Language) databases, which are relational, and NoSQL (Not Only SQL)

databases, which are non-relational. The choice between SQL and NoSQL databases depends on the specific needs of the application, including the nature of the data being handled, the scalability requirements, and the complexity of data queries.

2.3.1 SQL Databases

SQL databases, such as MySQL, PostgreSQL, and SQLite, use a structured query language for defining and manipulating data. This type of database is characterized by its use of tables to store data, making it a good choice for applications that require complex queries, transactions, and strong data integrity. SQL databases are ideal for applications with a clear schema that is not expected to change frequently over time.

- **MySQL:** An open-source relational database management system that is widely used for web applications. It is known for its reliability, compatibility with all major hosting providers, and ease of integration with various programming languages (What is MySQL: MySQL Explained for Beginners, 2024).

- **PostgreSQL:** An advanced open-source relational database that emphasizes standards compliance and extensibility (PostgreSQL: About, n.d.). It offers advanced features such as complex queries, foreign keys, triggers, views, transactional integrity, and multiversion concurrency control.

2.3.2 NoSQL Databases

NoSQL databases, such as MongoDB, Cassandra, and CouchDB, are designed to store, distribute, and access data using means other than the tabular relations used in relational databases. These databases are well-suited for handling large sets of distributed data and are known for their flexibility, scalability, and high performance in dealing with a wide variety of data types.

- **MongoDB:** A document-oriented database that stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time (What Is MongoDB?, n.d.). This model allows for the storage of complex hierarchies and is efficient for operations involving large volumes of data and real-time applications.

- **Cassandra:** A distributed NoSQL database designed to handle large amounts of data across many commodity servers, providing high availability without a single point of failure (Apache Cassandra Documentation, n.d.).

2.3.3 Justification for Choosing MongoDB

For this project, MongoDB has been selected as the database technology. This choice is informed by several considerations:

- **Schema Flexibility:** MongoDB's schema-less nature allows for the storage of flexible, JSON-like documents. This flexibility is advantageous for the project as it accommodates the evolving data requirements of the Ida-Virumaa Vocational Education Centre's application without the need for significant schema modifications.

- **Scalability:** MongoDB is designed with scalability in mind, supporting horizontal scaling through sharding and capable of handling large volumes of data and high traffic levels efficiently.

- **Development Speed:** The document model of MongoDB aligns well with the object-oriented programming paradigm, simplifying the development process by allowing for straightforward mapping of application objects to database documents.

- **Ecosystem and Community Support:** MongoDB has a vast and active community, ensuring access to a wide range of tools, extensions, and support. This ecosystem is beneficial for rapid development and addressing any challenges that may arise during the project.

By leveraging MongoDB, the project aims to create a robust and scalable database solution that can adapt to the changing needs of the application, ensuring efficient data management and retrieval for the Ida-Virumaa Vocational Education Centre.

3 THE APPLICATION

3.1 Design Phase

3.1.1 Overview of Application Architecture

The architecture of the application is foundational to its functionality and performance. For this project, a client-server architecture has been implemented using the MEVN stack. This architecture choice effectively separates concerns, enhancing maintainability and scalability.

- **Client-Server Model:** The application adopts a traditional client-server model where the client (front-end) is built using Vue.js, and the server (back-end) utilizes Express.js running on Node.js. MongoDB serves as the database, managing all data storage needs. This separation ensures that user interface concerns are decoupled from data management and processing logic, improving the application's responsiveness and user experience.

- **Architecture Decision:** Initially, a microservices architecture was considered to allow for more modular scaling and development. However, for simplicity and to keep the initial overhead low, a client-server model was deemed more appropriate. This approach provides a straightforward development and deployment process, suitable for the scope of this project.

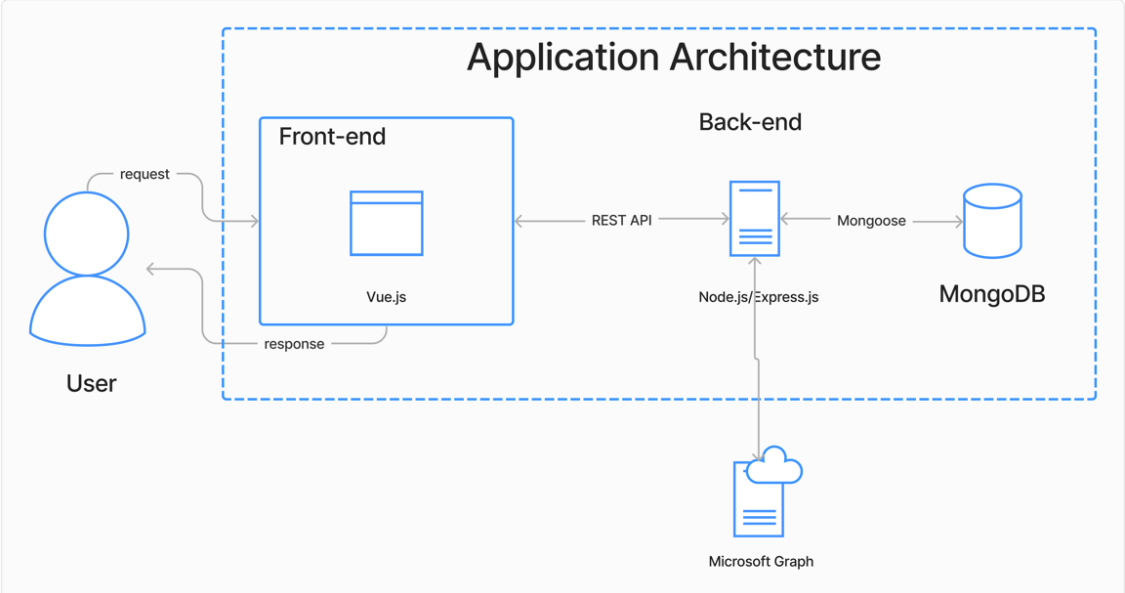

Figure 1 Application Architecture (source: Author)

3.1.2 Component Design

The application's structure is logically divided into three main components, each handling distinct aspects of the application's functionality.

- **Front-end:** The user interface, developed with Vue.js, provides interactive and dynamic web pages. It sends requests to the server based on user interactions.
- **Back-end:** The server, developed using Express.js, processes all business logic, handles API requests, and performs CRUD operations with the database.
- **Database:** MongoDB stores all application data, including user profiles and event data. It interacts with the back-end to retrieve, store, and update data as needed.

3.1.3 Technology Stack Integration

Each component of the MEVN stack plays a crucial role in the application's overall functionality:

- **MongoDB:** Integrated to handle all data storage requirements, MongoDB's schema-less nature allows for flexibility in managing different data types and structures that the application might require.
- **Express.js and Node.js:** These are used to set up the server and routing logic. Express.js, running on Node.js, manages request handling, middleware integration, and response sending.
- **Vue.js:** Chosen for its efficiency in building dynamic and reactive user interfaces, Vue.js enhances the user experience by making the application responsive and intuitive to interact with.

3.1.4 Technology Dependencies

The application's functionality is supported by various libraries and frameworks, integrated within the front-end and back-end. Here’s a detailed breakdown:

- **Front-end Dependencies:**

- **axios**: For making HTTP requests to the back-end.
- **date-fns**: Used for handling and formatting dates.
- **qrcode.vue**, **vue-qrcode-reader**: For generating and reading QR codes within the application.
- **vue-router**: Manages navigation within the application.
- **Diagram Needed:** Include a package dependency diagram for the front-end.

- **Back-end Dependencies:**
  - **@azure/msal-node**, **@microsoft/microsoft-graph-client**: For handling authentication via Microsoft.
  - **express-session**, **jsonwebtoken**: For managing sessions and token-based authentication.
  - **mongoose**: To model and manage database interactions in MongoDB.
  - **Diagram Needed:** A similar package dependency diagram for the back-end.

3.1.5 RESTful API

The application's server-side functionality is exposed through a RESTful API, facilitating communication between the front-end and the database. Key endpoints include:

- **Users:** Handles user data retrieval, update, and authentication checks.
- **Events:** Manages event creation, modification, and querying.
- **Registrations:** Deals with user registrations for events, including presence management.

3.1.6 Detailed API Endpoints

The API for the application is designed to facilitate smooth interaction between the front-end, the server, and the database. This section details the various endpoints, the HTTP methods they employ, and their specific roles within the system. Each endpoint is critical for

handling specific aspects of the application such as user management, event handling, and registration processes. Diagrams will be included to illustrate the relationships and data flows between these endpoints and the MongoDB database.

Users:

- **Get User by ID**
  - Method: GET
  - Endpoint: /user_by_id/:id
  - Description: Retrieves a user's data based on their unique identifier.
  - Database Interaction: Queries the MongoDB database to fetch user details.
- **Handle Microsoft Redirect**
  - Method: GET
  - Endpoint: /redirect
  - Description: Handles the OAuth redirection for Microsoft authentication.
  - Database Interaction: May interact with the database to update or verify user authentication status.
- **Get User Roles**
  - Method: GET
  - Endpoint: /user-roles
  - Description: Fetches the roles assigned to the authenticated user.
  - Database Interaction: Queries user roles from the database based on user credentials.
- **Get User Groups by User ID**
  - Method: GET
  - Endpoint: /:id/groups
  - Description: Retrieves all groups associated with a specific user.
  - Database Interaction: Fetches groups from the database where the user is a member.
- **Delete User**
  - Method: DELETE
  - Endpoint: /delete/:id
  - Description: Deletes a user from the application.
  - Database Interaction: Removes the user's record from the MongoDB database.

Events:

- **Create Event**
  - Method: POST
  - Endpoint: /create
  - Description: Allows for the creation of a new event.
  - Database Interaction: Inserts a new event into the database.
- **Get Events**
  - Method: GET
  - Endpoint: /get
  - Description: Retrieves all events.
  - Database Interaction: Queries all events from the database.
- **Get Event by ID**

    - Method: GET
    - Endpoint: /get/:id
    - Description: Fetches detailed information about a specific event.
    - Database Interaction: Looks up a single event by its ID in the database.
- **Update Event**
    - Method: PUT
    - Endpoint: /update/:id
    - Description: Updates information about an existing event.
    - Database Interaction: Modifies an event's details in the database.
- **Delete Event**
    - Method: DELETE
    - Endpoint: /delete/:id
    - Description: Removes an event from the system.
    - Database Interaction: Deletes an event from the database.

Registrations:

- **Register for Event**
    - Method: POST
    - Endpoint: /create
    - Description: Registers a user for an event.
    - Database Interaction: Adds a new registration record to the database.
- **Get All Registrations**
    - Method: GET
    - Endpoint: /getAll
    - Description: Retrieves all registrations for all events.
    - Database Interaction: Fetches all registration data from the database.
- **Get Registrations by User**
    - Method: GET
    - Endpoint: /get/:userId
    - Description: Fetches all registrations made by a specific user.
    - Database Interaction: Queries the database for registrations associated with a particular user.
- **Update Presence**
    - Method: PUT
    - Endpoint: /update_presence/:registrationId/:isPresent
    - Description: Updates the attendance status of a user at an event.
    - Database Interaction: Modifies a registration record to reflect the attendance status.
- **Delete Registration**
    - Method: DELETE
    - Endpoint: /delete/:registrationId
    - Description: Removes a registration from the system.
    - Database Interaction: Deletes a registration record from the database.

3.2 Development Phase

The development phase of the application involved configuring the development environment, implementing core features, and integrating various components and external

services to ensure smooth operation and functionality. This section details these essential aspects of the application's construction.

3.2.1 Environment Setup

A well-structured development environment is crucial for efficient development and collaboration. Here's how the environment was configured for this project:

- **Node.js and npm**: Node.js serves as the runtime environment with npm managing package dependencies. Installation ensures compatibility with various libraries used throughout the project.
- **Vite**: Instead of Vue CLI, Vite was chosen to scaffold the Vue.js frontend. Vite provides a faster and more modern development environment, optimizing loading time and simplifying the configuration.
- **MongoDB Atlas**: Instead of a local MongoDB setup, MongoDB Atlas, a cloud database service, was used. This allowed for scalable data management and remote access without the need for local database maintenance.
- **IDE and Tools**: JetBrains WebStorm was selected as the primary IDE, known for its robust support for JavaScript and frontend technologies, along with integrated tools for debugging and testing.
- **Version Control**: Git, in conjunction with GitHub, facilitated version control. This setup supported feature branching, pull requests, and code reviews to maintain high code quality and facilitate collaborative development.

3.2.2 Implementation of Core Features

The application's functionality is designed around several pivotal features, each crafted to meet the specific needs of the users and streamline processes within the educational environment. This section elaborates on the implementation of core features like user registration, retake scheduling, and QR code integration for attendance tracking.

3.2.2.1 User Registration Feature

The user registration feature in the application is critical as it serves as the entry point for users into the system, establishing their credentials and roles. It integrates seamlessly with Microsoft's identity platform, using OAuth for authentication and Microsoft Graph for data retrieval. Below is a detailed breakdown of the user registration implementation, including schema design and the logic for creating or updating a user in the database.

The **User** schema is structured to capture essential information retrieved from the user's Microsoft account, as well as additional data relevant to the application's context. Here's a detailed explanation of each field in the **UserSchema**:

- **microsoftId**: A unique identifier provided by Microsoft, used as a primary key to distinguish each user.
- **email**: The user's email address, required and must be unique within the database to ensure each email corresponds to one user.
- **role**: The user's role within the application, which could vary and is not mandatory upon initial setup.
- **name**: The full name of the user, as displayed in the Microsoft account.
- **groups**: An array of strings, each representing a student group the user belongs to. These are parsed from the group information fetched from Microsoft.
- **additional_info**: A flexible field using a map data structure to store additional attributes about the user, which might include settings or preferences as strings.

The **createOrUpdateUser** function handles both the creation of new users and the updating of existing user records in the database. Here's how it operates:

1. **Extract and Map Data**: The function begins by destructuring the **profileData** object to extract relevant fields such as **id**, **jobTitle**, **displayName**, and **userPrincipalName**. The **jobTitle** is used to determine the user's role through a mapping function, which converts Estonian roles to their English equivalents.

2. **Identify Student Groups**: Using the **groupsData** array, the function filters and maps over each group to identify those that are designated as student groups. This is done using a regular expression that matches groups ending with two digits, indicative of student groups like "JPKM22".

3. **Check for Existing User**: The function then checks if a user already exists in the database with the given **microsoftId**. If not, a new user is created.

4. **Create or Update User Record**:

    - **New User**: If no existing user is found, a new **User** document is created with the retrieved and mapped data, then saved to MongoDB.

    - **Existing User**: If a user already exists, the record is updated with the new data, ensuring the user's information is current.

5. **Save or Update Operation**: The database operation either saves the new user or updates the existing one, handling any errors that might occur during this process.

Figure 2 Screenshot of login page (Source: author)

The user registration feature is designed to be robust, leveraging Microsoft's secure authentication system and automating the data handling process to minimize user input errors and streamline user integration into the system. By ensuring that user data is

consistently up-to-date and correctly categorized, the application can provide a personalized and secure user experience. This setup not only simplifies the initial user onboarding but also maintains the integrity and accuracy of user data throughout their interaction with the application.

3.2.2.2 Retake Scheduling Feature

The retake scheduling functionality is a core component of the application, designed to manage and coordinate events such as exams or class retakes efficiently. This section dives into the details of the event model, the operations performed by the backend to handle event data, and the frontend implementation that interacts with these operations.

The **EventSchema** captures all necessary details about each event:

- **title**: A string that names the event, making it easily identifiable.
- **date**: The scheduled date and time of the event.
- **locations**: An array storing the possible locations where the event can be held.
- **description**: A brief description of what the event entails.
- **registrationOpens** and **registrationCloses**: These fields define the window during which students can register for the event, calculated relative to the event date.
- **responsivePerson**: An optional field identifying the main contact person for the event.
- **createdBy**: Stores the identifier of the user who created the event.
- **createdOn**: The date and time when the event was created.
- **status**: Indicates whether the event is active, canceled, or completed, defaulting to 'active'.

The backend controllers manage the creation and retrieval of event data:

- **createEvent**: This function handles the creation of a new event. It converts the provided event date from a string to a **Date** object and calculates the registration open

and close dates based on the event date. The event is then saved to the database, and a successful response is returned with the event data.

- **getEvents**: Fetches all events from the database. This function can be extended to filter events based on their active status or other criteria to optimize data retrieval.
- **getEventById**: Retrieves a specific event by its ID, useful for viewing detailed information about a single event.
- **getLocationsByEventId**: Fetches the location details for a specific event, which is crucial for logistical preparations and informing participants.

The Vue.js component for retake scheduling interacts with the backend to fetch and display event data:

- **Data Fetching**: Using Axios, the component sends a GET request to retrieve all current and upcoming events where registration is open. It filters these events based on the current date and the registration window.
- **User Interface**: Each event is displayed using the **EventCard** component, which presents the event details in a clear and interactive format. Users can view events and register for them within the defined registration period.
- **Reactivity and Accessibility**: The Vue.js setup ensures that the list of events is reactive and updates dynamically as new data is fetched or events are updated.

Figure 3 Screenshot of Scheduling Page

The retake scheduling feature is a complex component that requires meticulous attention to detail in both data management and user interface design. It plays a critical role in ensuring that students and staff can manage their schedules efficiently, with clear information on upcoming events and necessary actions. By carefully implementing and documenting each aspect of this feature, the application not only enhances user experience but also supports robust data integrity and operational reliability.

3.2.2.3 Registration for Event Feature

The registration feature is essential for managing student enrollments for various events like retakes or additional classes. It involves detailed mechanisms to handle registration limitations, store comprehensive event and user data, and ensure a smooth user experience through an interactive interface.

The **RegistrationSchema** is designed to capture all necessary details related to a user's registration for an event:

- **eventId**: References the specific event for which the registration is made.
- **userId**: Identifies the user registering for the event.
- **place**: Specifies the location where the event will take place.
- **teacherId**: The identifier for the teacher responsible for the event.
- **subject**: The subject or topic of the event.
- **work_title**: A title or description of the work to be done or discussed in the event.
- **status**: Tracks the registration status, such as 'pending', 'confirmed', or 'canceled'.
- **isPresent**: A boolean indicating whether the user attended the event.

The backend functionality supports creating and managing registrations, enforcing business rules such as registration limits:

- **registerForEvent**: Handles the creation of new registrations with a limit of three per user per event. It first checks existing registrations to ensure this limit is not

exceeded. If the limit is reached, it denies the registration and provides feedback. Otherwise, it proceeds to create a new registration entry.

- **getRegistrations**: Retrieves all registrations associated with a specific user, allowing users to view their event commitments.
- **getAllRegistrations**: Fetches all registrations from the database, useful for administrative purposes to monitor overall event enrollments.
- **getByEventAndLocation**: Returns all registrations for a particular event at a specified location, aiding in logistical planning and attendee management.
- **getPresence**: Retrieves the attendance status of a specific registration, essential for tracking participant attendance.
- **updatePresence**: Updates the attendance status for a registration, allowing event organizers to mark participants as present or absent.

The **RegisterForEvent** Vue component facilitates the user interaction for event registration:

- **Data Fetching**:
    - **Employees**: Fetches a list of potential teachers or responsible persons from the backend.
    - **User Data**: Retrieves the current user's data to pre-fill forms and manage registrations.
    - **Locations**: Gathers available locations for the event to offer choices to the user.
- **Registration Form**: Users fill out a detailed form to register for an event. The form includes fields for selecting the location, teacher, subject, and work title, bound to the **registrationData** reactive variable.
- **Form Submission**: On submission, the form data is posted to the backend. If successful, the user is redirected to their profile or another relevant page. In case of an error, such as exceeding the registration limit, an error message is displayed.

- **Error Handling**: Displays error messages to inform the user of any issues during the registration process, enhancing user experience by providing clear feedback on action results.

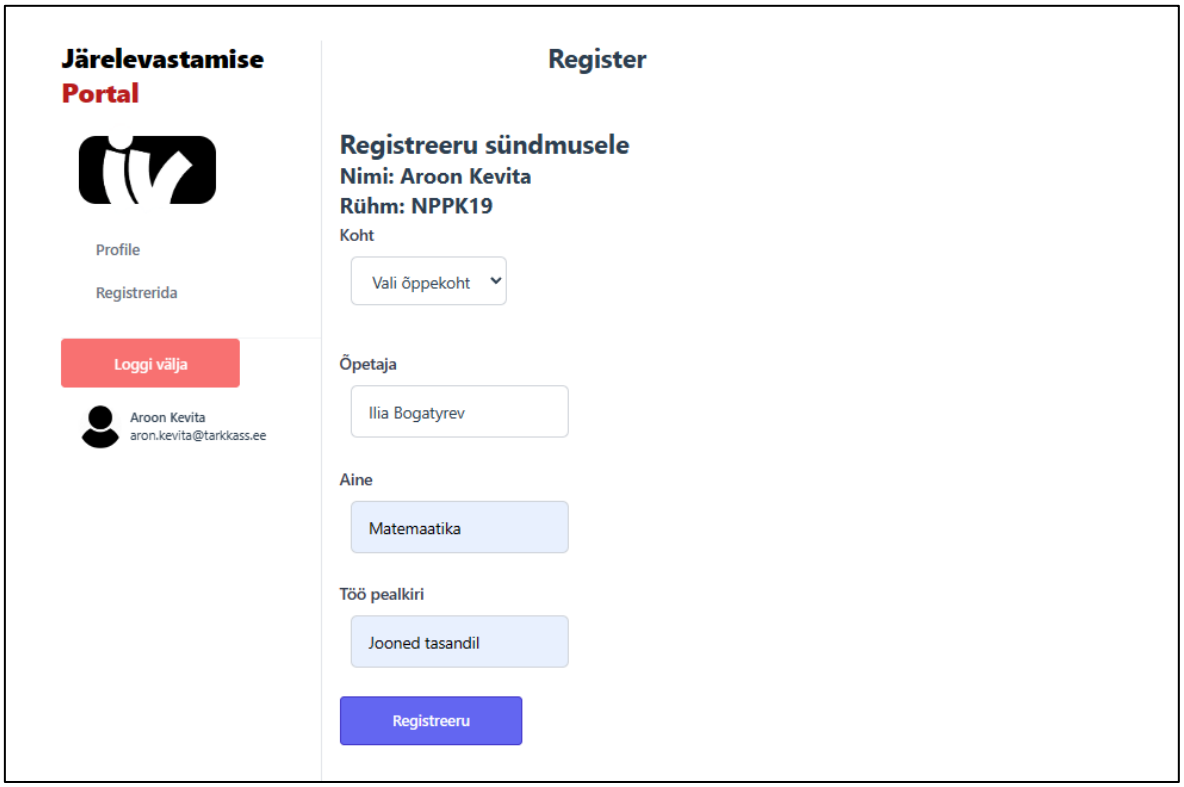

Figure 4 Screenshot of Registration Form (Source: author)

The event registration feature is a crucial aspect of the application, designed to handle user registrations efficiently while enforcing rules such as registration limits. By integrating frontend components with robust backend logic, the feature not only enhances user engagement but also ensures accurate and secure data handling. This comprehensive approach facilitates effective management of educational events, contributing to a streamlined educational experience.

3.2.2.4 QR Code Integration for Attendance

The QR code integration for attendance is a vital component of the system, designed to automate and streamline the process of marking attendance at events. This feature reduces administrative burden and improves accuracy by leveraging modern technology. Below is an in-depth look at how this feature is implemented within the application.

The **StudentCard** component in Vue is responsible for displaying student information along with a dynamically generated QR code that contains the student's user ID. This QR code is essential for the attendance process, as it will be scanned at the event to verify attendance.

- **QrcodeVue**: This component generates a QR code based on the user ID. The QR code is displayed on each student's card, ready to be scanned at the event.

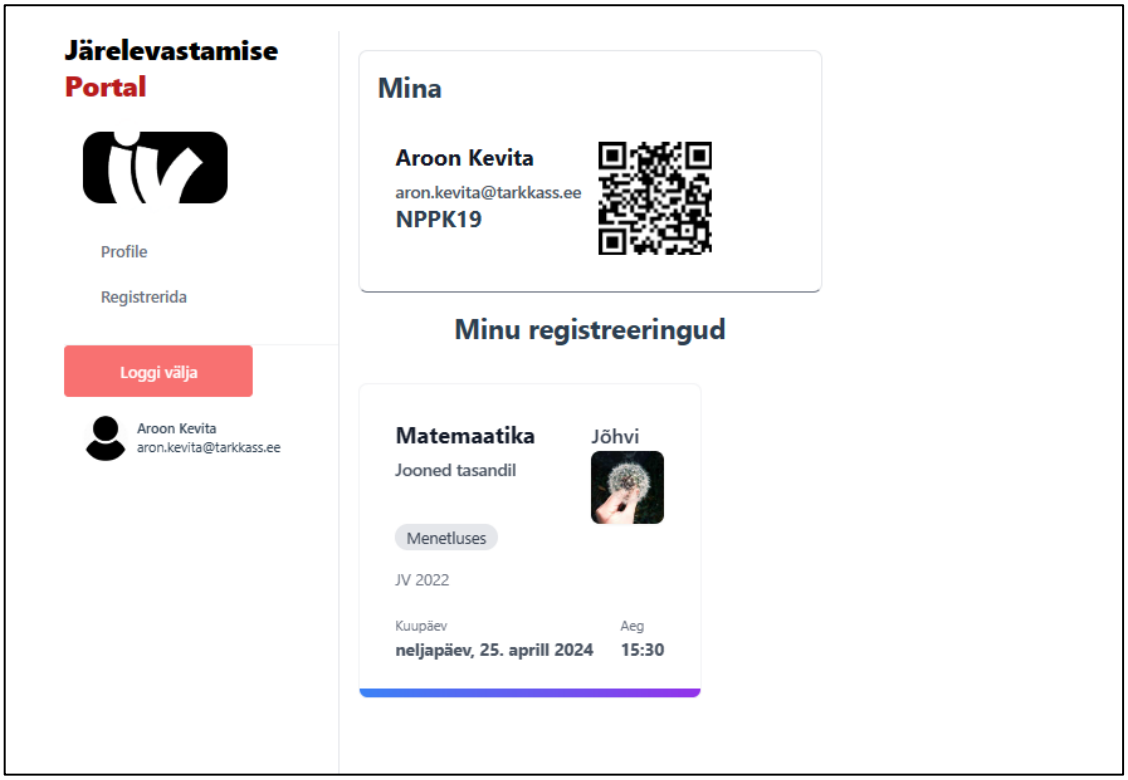

Figure 5 Screenshot of Profile Page (Source: author)

The **Journal** component serves as the interface for event organizers to manage attendance. It lists all the registrations for a specific event and provides the functionality to open a scanner interface.

- **ScanerId Component Integration**: This sub-component is triggered to open within the **Journal** when the "Scan QR Code" button is clicked. It uses the camera to scan QR codes presented by students as they arrive.

The **ScanerId** component is crucial for scanning QR codes and updating attendance status in real-time.

- **QR Code Detection**: Utilizes **QrcodeStream** to detect QR codes. Once a code is detected, it extracts the user ID encoded within.

- **Fetch Registration Data**: After detecting a QR code, the component fetches registration data for that user and the specific event, ensuring that the student is registered for the event.

- **Update Attendance**: If the student's registration is confirmed, their presence is marked in the system. This action updates the database to reflect that the student has attended the event.

The backend supports these operations through several API endpoints:

- **Get Registrations by Event and User**: Retrieves registration details to confirm if the user is registered for the event before marking them as present.

- **Update Presence**: Updates the attendance status in the database once the QR code is scanned and the user is verified to be present at the event.

Flow and Interactions:

1. **StudentCard Display**: Each participant has a **StudentCard** with a QR code.

2. **Journal Management**: Event organizers view registrations and open the QR scanner.

3. **QR Scanning**: **ScanerId** scans the QR code and fetches the relevant registration data.

4. **Attendance Confirmation**: Upon successful verification, the system updates the attendance status to "present."

Integrating QR codes for attendance tracking enhances the efficiency and accuracy of managing event participation. By automating the attendance verification process, the system not only saves time but also ensures that attendance data is precise and reliable. The implementation of this feature using Vue.js components and backend services demonstrates an effective use of technology to address practical needs in educational and event management settings.

3.2.3 Integration Processes

Seamless integration of components and external services is critical for the application's functionality:

- **Microsoft Authentication**:
    - Implemented using Microsoft's authentication library to manage secure login sessions. This setup involves handling OAuth redirections and using Microsoft Graph to fetch user data during the registration process.
- **API Integration**:
    - The backend provides RESTful APIs for the frontend to interact with MongoDB Atlas. These APIs are crucial for fetching event data and handling user registrations.
- **Middleware and Security**:
    - Express middleware handles CORS and security headers to secure API communication. JWT tokens are used for authenticating API requests and managing user sessions.
- **Continuous Integration/Continuous Deployment (CI/CD)**:
    - CI/CD pipelines are implemented using GitHub Actions, automating testing and deployment processes. This ensures that any updates to the codebase are automatically tested and deployed to the production environment after passing required checks.

3.2.4 Conclusion

This phase of development established a robust technical foundation, implemented essential functionalities, and ensured smooth integration across different system components. The use of diagrams in this documentation will further clarify the architecture and workflow, providing a valuable reference for understanding system operations and facilitating future maintenance and scalability enhancements.

3.3 Deployment and Configuration

Deploying and configuring the application in a production environment involves several steps to ensure that the software is delivered efficiently, securely, and is maintainable. This section covers the deployment strategy and configuration management for the application, highlighting the use of DigitalOcean, Docker, Nginx, GitHub Actions, and Cloudflare.

3.3.1 Deployment Strategy

The application is deployed on a DigitalOcean droplet, utilizing Docker to containerize both the front-end and back-end components. Here are the detailed steps for the deployment process:

1. **Containerization with Docker**:

    - **Front-end and Back-end Containers**: Each part of the application is packaged into separate Docker containers. This approach ensures isolation of environments, managing dependencies effectively, and simplifying updates and scaling.

    - **Docker Images**: Docker images are created for both the front-end and back-end, specifying the environment, dependencies, and the runtime configuration in Dockerfiles.

2. **Nginx Configuration**:

    - **Server-Level Nginx**: On the DigitalOcean droplet, Nginx is configured to handle incoming HTTP requests. It serves as the reverse proxy, directing traffic to the appropriate Docker container based on the request path.

    - **Front-end Nginx**: Inside the front-end container, Nginx is also configured to serve the static files and handle caching, providing efficient content delivery.

    - **Routing Rules**: Requests starting with **/api** are routed to the back-end container, ensuring that API calls are processed by the server-side application.

3. **Deployment Automation with GitHub Actions**:

- **CI/CD Pipeline**: GitHub Actions is set up to automate the continuous integration and deployment process. Upon a commit to the main branch, the action triggers the build process for Docker containers.

- **DockerHub Integration**: Once built, the Docker images are pushed to DockerHub. The latest images are then pulled from DockerHub to the DigitalOcean droplet as part of the deployment process.

- **Container Deployment**: Scripts within the GitHub Actions pipeline handle stopping the current containers, pulling the latest images, and running the new containers, ensuring minimal downtime and seamless updates.

3.3.2 Configuration Management

Configuration management is critical to ensure that the application runs smoothly in different environments and is easy to maintain:

1. **Environment Variables**:

    - **Separation of Configurations**: Environment variables are used to separate configuration from code, defining critical settings such as database connections, API keys, and service endpoints. This method enhances security and flexibility.

    - **.env Files**: For local development, settings are stored in **.env** files. In production, these variables are securely set in the DigitalOcean droplet environment and passed to Docker containers.

2. **SSL Configuration via Cloudflare**:

    - **SSL Management**: To simplify SSL certificate management and enhance security, Cloudflare is used. It handles SSL/TLS termination and encrypts traffic between Cloudflare and end-users.

    - **DNS and CDN Services**: Cloudflare also provides DNS services and acts as a content delivery network (CDN), improving the application's performance and resilience against DDoS attacks.

3. **Maintenance and Monitoring**:

    - **Logging and Monitoring**: Systematic logging is configured in both the front-end and back-end containers to capture runtime errors and access logs. Tools like DigitalOcean Monitoring and third-party services can be integrated to monitor the health and performance of the application.

    - **Update Procedures**: Regular updates to the Docker images and Nginx configurations are managed through the CI/CD pipeline, ensuring that the application remains secure and up-to-date with minimal manual intervention.

3.3.3 Conclusion

The deployment and configuration of the application are designed to be robust, secure, and scalable. The use of Docker, DigitalOcean, Nginx, GitHub Actions, and Cloudflare creates a flexible and maintainable infrastructure that supports continuous development and efficient deployment. This strategy not only addresses the technical requirements of the application but also aligns with best practices for web application deployment and maintenance.

4 CONCLUSION

In the pursuit of advancing educational administration at the Ida-Virumaa Vocational Education Centre (IVVEC), this thesis has developed a comprehensive application tailored to manage the registration and retaking process of written work. Through a detailed exploration and implementation of modern web technologies, the application addresses the critical challenges faced by IVVEC, streamlining administrative tasks, and improving accuracy in tracking student attendance for retakes.

The application leverages the MEVN stack (MongoDB, Express.js, Vue.js, Node.js) to provide a robust, scalable, and maintainable solution. The decision to utilize Vue.js for the front-end, coupled with Express.js and Node.js for the back-end, ensures a cohesive development environment with a focus on performance and scalability. MongoDB's flexibility and scalability make it an ideal choice for managing the varied data needs of the application.

Key features such as QR code integration for attendance verification, user-friendly interfaces for event scheduling, and seamless user registration through Microsoft Entra authentication have been meticulously designed and implemented. These functionalities not only simplify the administrative workload but also enhance the user experience for students, teachers, and administrators alike.

The deployment strategy, using Docker for containerization and DigitalOcean for hosting, along with CI/CD pipelines through GitHub Actions, ensures a streamlined and efficient deployment process. Nginx and Cloudflare integration further enhance the security, performance, and scalability of the application, making it well-equipped to handle the demands of IVVEC's administrative processes.

By addressing the specific needs of IVVEC and providing a tailored solution, this thesis contributes significantly to the field of educational technology. The application serves as a benchmark for similar institutions seeking to modernize their administrative workflows, demonstrating how technology can be harnessed to enhance educational operations and reduce administrative burdens.

Future work could explore the integration of additional features such as automated notifications, enhanced analytics for attendance data, and broader integrations with other

educational tools and platforms. Continuous feedback from users will be essential in iterating and improving the application, ensuring it remains aligned with the evolving needs of the educational environment.

In conclusion, this thesis has successfully developed and deployed an application that not only meets the immediate needs of IVVEC but also sets a foundation for future enhancements and expansions. The innovative use of modern web technologies and a user-centered design approach underscores the potential of technology to transform educational administration, paving the way for more efficient, accurate, and user-friendly systems in educational institutions.

5 REFERENCES

About ExamSoft | Innovative Assessment Software. (n.d.). Retrieved from https://examsoft.com/about-examsoft

About Moodle - MoodleDocs. (2024, January 27). Retrieved from https://docs.moodle.org/403/en/About_Moodle

Angular - What is Angular? (2023, August 15). Retrieved from https://angular.io/guide/what-is-angular

Apache Cassandra Documentation. (n.d.). Retrieved from Apache Cassandra Web Site: https://cassandra.apache.org/_/index.html

Django. (n.d.). Retrieved from MozillaWiki Web Site: https://wiki.mozilla.org/Django

Express - Node.js web application framework. (n.d.). Retrieved from Express - Node.js web application framework: https://expressjs.com/

Ferreira, F. &. (2021). On the (un-)adoption of JavaScript front-end frameworks. *Software: Practice and Experience*.

HTML Tutorial. (n.d.). Retrieved from w3schools.com: https://www.w3schools.com/html/

Introduction | Vue.js. (n.d.). Retrieved from https://vuejs.org/guide/introduction.html

Merced, A. (2020, August 22). *Hello World in Vue*. Retrieved from LinkedIn: https://www.linkedin.com/pulse/hello-world-vue-alex-merced/

Miller, S. (2021, September 09). *What Is Ruby on Rails?* Retrieved from Codecademy Web Site: https://www.codecademy.com/resources/blog/what-is-ruby-on-rails/

Ohannessian, T. (n.d.). Retrieved from MyAttendanceTracker :: Student/Class Grading Attendance Dashboards: https://www.myattendancetracker.com/about

PostgreSQL: About. (n.d.). Retrieved from The PostgreSQL Global Development Group Web Site: https://www.postgresql.org/about/

Ruby on Rails. (n.d.). Retrieved from learnamic: https://www.learnamic.com/topics/ruby-on-rails

Stromer, M. (n.d.). *Intro to Django*. Retrieved from Michael Stromer Web Site: https://www.djangoproject.com/start/overview/

Wallis, J. (n.d.). *Demystifying Tech Stacks: Unveiling the Secrets of Technology Stacks*. Retrieved from Intuji Web Site: https://intuji.com/tech-stacks-unveiling-the-secrets/

What is CSS. (n.d.). Retrieved from w3schools.com: https://www.w3schools.com/whatis/whatis_css.asp

What is JavaScript? (2024, January 24). Retrieved from Mozilla Foundation Web Site: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

What Is MongoDB? (n.d.). Retrieved from MongoDB Web Site: https://www.mongodb.com/company/what-is-mongodb

What is MySQL: MySQL Explained for Beginners. (2024, January 19). Retrieved from Hostinger Tutorials: https://www.hostinger.com/tutorials/what-is-mysql

What is React? (n.d.). Retrieved from W3Schools: https://www.w3schools.com/whatis/whatis_react.asp

6 APPENDIX

6.1 Code source

- Front-end finmipt/retake-front: The front-end (github.com)
- Back end finmipt/retake-back (github.com)

6.2 Licence

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Ilia Bogatyrev**

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

Development Of An Application For Registering For Retaking Of Written Work At Ida-Virumaa Vocational Education Centre.

mille juhendaja on **Andre Sääsk**,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Ilia Bogatyrev

19.05.2024