LAB University of Applied Sciences

Development of web application using MERN Stack

PSP Application for LAB Mechanical Engineering

LAB University of Applied Sciences
Bachelor's Degree Programme in Industrial Information Technology
2025
Firoz Thapa

Abstract

Author(s)	Publication type	Completion year
Firoz Thapa	Thesis, UAS	2025
	Number of pages 35	

Title of the thesis

Development of web application using MERN Stack

PSP Application for LAB Mechanical Engineering

Degree, Field of Study

Bachelor's Degree Programme in Industrial Information Technology

Name, title and organisation of the client.

LAB University of Applied Sciences

Abstract

As educational institutions are facing challenges in managing the study competencies effectively, the need for digital transformation is increasing. One of the major challenges that the department of mechanical engineering in LAB University of Applied Sciences was facing was that the competencies of the project based curriculum were managed by using the excel sheets, which eventually resulted in the difficulty in the tracking of the student progress data as the number increases.

The aim of this thesis was to transform that manual process into a comprehensive digital platform which will help teachers' manage students' competencies more effectively. Implementing MERN Stack (MongoDB, Express.js, React, and Node.js) the successful development of PSP application was carried out. The methodology included the organized requirement analysis through regular meetings, iterative development and designing of user interface prior to the development. As a result, the application is able to provide all the essential authentication, workflow, and communication features. Eventually, this transformation automated curriculum management.

Keywords

MERN Stack, Web Development, Personal Study Plan, RESTful API, Material UI, Redux

Contents

1 Introduction

In this modern world and its developing education system, effective management of the project based curriculum management system has become challenging as the institutions adopt more complex learning frameworks. The mechanical engineering department in the LAB University of Applied Sciences faced difficulties in tracking the students' progress in their project based subject management system while using the excel sheet, especially when the number of students is increasing and the complexity in the curriculum evolved. Depending on the manual processes resulted administrative inefficiencies, limited the accessibility and created a problem in progress monitoring.

The purpose of this thesis was to develop a Personal Study Plan (PSP) Web application using the MERN Stack that would help to evolve from excel based management of project based curriculum to a modern digital platform. This study presents a technical case study showcasing how the MERN Stack(MongoDB, Express.js, React, and Node.js) can be implemented in the development of the web application in the context of education (GeeksforGeeks, 2024). The implementation here explores the fundamental development challenges like creating a single page application, building a flexible NoSQL database schema, developing RESTful APIs, and implementing server-side runtime environments. As shown in Figure 1, the PSP Application Architecture illustrates the clear separation between frontend and backend components with a well-documented Rest API in between, where different user roles (Teacher and Student) interact with the dedicated user interface components which is processed by specific controllers accessing purpose-built MongoDB collections.

The resulting application provides role-based access control, streamlined project submission workflows, curriculum management capabilities, and real-time notification systems that automated the administrative tasks and improved transparency for all stakeholders. These features created a foundation for data-driven curriculum development through enhanced progress tracking.

This thesis is organized into seven different chapters, containing MERN Stack fundamentals, requirement analysis, graphic design, software implementation, system feature, conclusions, limitations, and future implementations, which together provide a comprehensive examination of both technical and practical application in an educational environment.

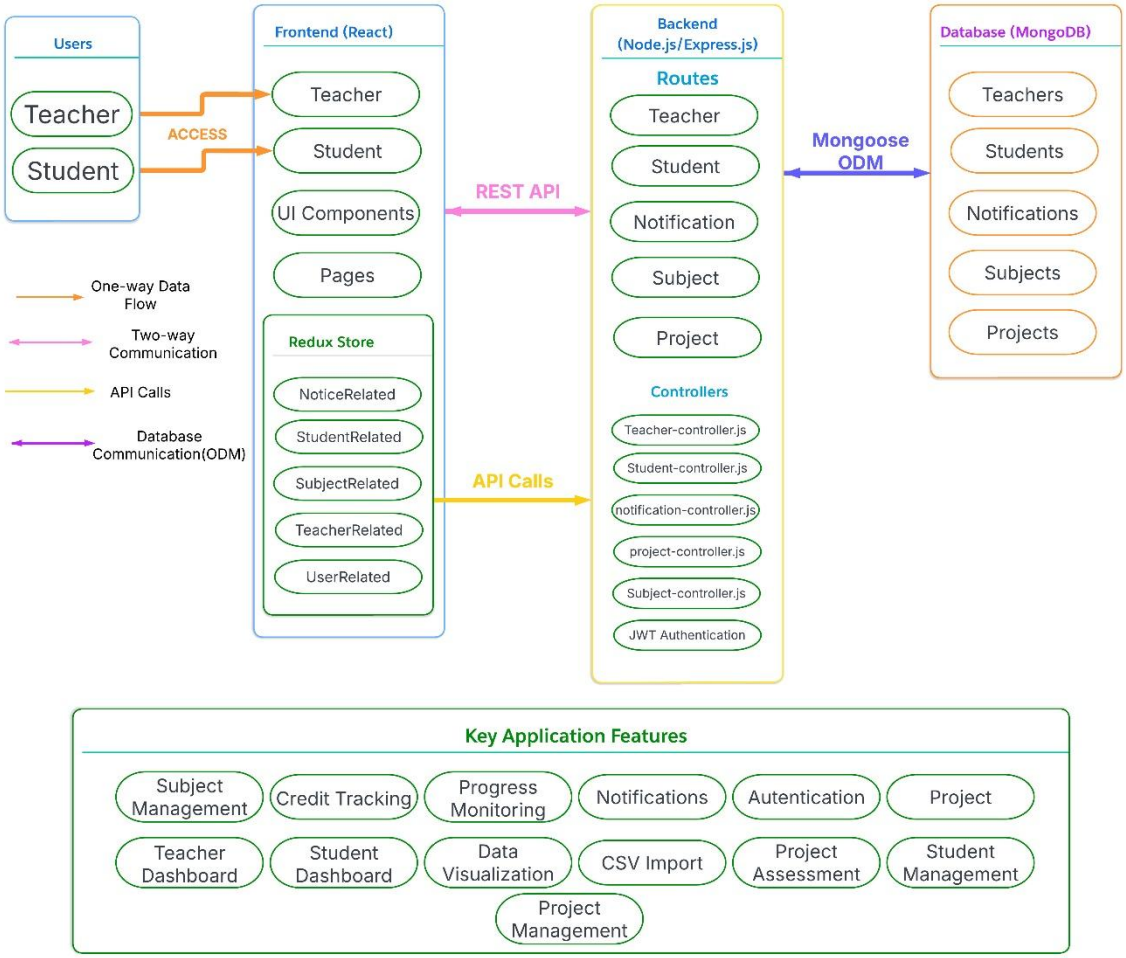


Figure 1. PSP Application Architecture

2 Understanding MERN Stack

The MERN Stack is comprised of four basic technologies in order to develop a unified JavaScript development platform for web applications. As illustrated in Figure 2, every element plays a specific role and works with others to make full-stack development possible. The data flow diagram shows that how these technologies work together to create an entire application framework.

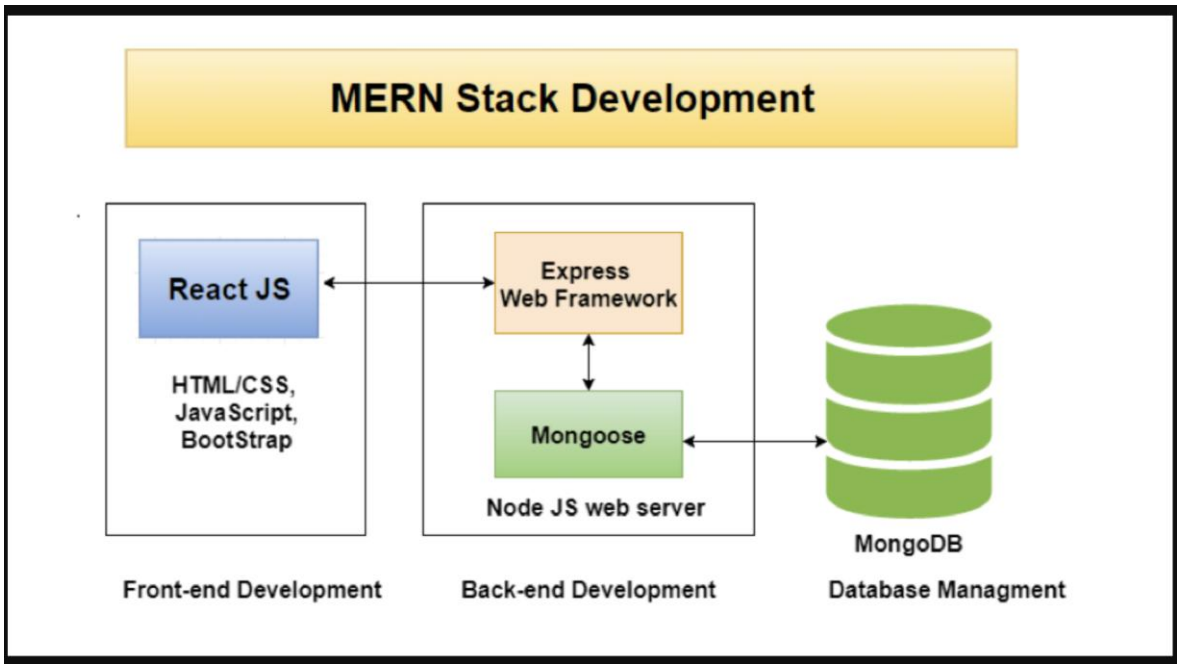


Figure 2. MERN Stack Data Flow Diagram (Bocasay. 2024)

MongoDB: Document-Oriented Database

MongoDB is a special type of database that stores the information in simple documents instead of complicated tables which makes it easy to build apps and grow them bigger when needed (MongoDB, Inc 2023).The MongoDB was used to store information about subjects, outcomes, and student projects in the application easily. Because of this document-oriented database it was possible to store data in JSON-like documents rather than rigid tables. This flexibility helps to adapt easily in the scenario of changing data structures without having to rebuild the entire database. As the application grows, MongoDB's built-in sharing capabilities help to scale horizontally also known as sharding. This is also one of the major reasons to use MongoDB, since its query language is rich and complex, and its indexing is designed to optimize performance for all sorts of query patterns.

Express.js: Server Framework

Express.js provides the application backend with a lean but powerful framework. Middleware design builds a modular request handling pipeline for error handling, logging, and authentication (Express.js 2024). Express's structured routing system organizes API endpoints for educational resource management. This facilitates standardized data exchange between the frontend and database using RESTful API patterns. Centralized error handling ensures consistent responses across the application. For the PSP system, Express manages secure routes for teachers' and students' and implements efficient endpoints for subject and project management.

React: Frontend Library

React transforms frontend development through component-based architecture. This results in a virtual DOM implementation, which is a tree of JavaScript objects indicating the actual DOM that calculates minimal browser updates, thereby enhancing performance (Mojeed 2025). Reacts unidirectional data flow makes state changes predictable and makes the debugging easier. Whereas component based lifecycle methods gives a precise control over rendering and resource management . For the combination of HTML like markup language with JavaScript logic for intuitive user interface development JSX syntax is considered useful. The PSP application interface uses React to create reusable components such as subject cards, outcome accordions and project submission forms while consistently maintaining optimal rendering performance.

Node.js: JavaScript Runtime

Node.js enables server-side JavaScript execution with an event-driven architecture. This means a non-blocking I/O model could be achieved which efficiently manages concurrent connections without thread multiplication (Node.js 2025). Asynchronous programming patterns, like promises and async/await, assisted in the further optimization of operations that involved Input/output (I/O), such as database queries or file operations. Additionally, the extensive npm ecosystem provides ready-made solutions for common development challenges, enabling a single-language environment that eliminates context switching between frontend and backend development. In the PSP application, Node.js effectively manages concurrent teacher and student connections while managing asynchronous database operations.

Integration Architecture

The power of the MERN Stack is found in the seamless integration of the components. API-based communication links React components with Express endpoints using HTTP requests. JSON data format is used as the inter-layer data exchange medium. State management libraries (Redux) synchronize UI and server data. JWT authentication offers secure frontend and backend data exchange. This integrated format enables the PSP application to efficiently synchronize data, feeding real-time updates from instructors into student views.

3 Requirement Analysis

The foundation of this research was laid through a truly comprehensive approach to understand both the functional and technical requirements for the Personal Study Plan (PSP) application at LAB University's mechanical engineering department. By participating in on those weekly collaboration sessions with the faculty, it became possible to really get a handle on how educational workflow works and refine those technical specs continuously. During the weekly meetings with stakeholders, the review of the existing features and planning of the next development iterations were conducted, which ensured that the educational objectives and technical implementation stayed aligned. Implementation of such a collaborative approach was a key to bridging the gap between what the faculty needed from a pedagogical standpoint and what technology could actually deliver.

From a technical view, the MERN Stack selection required specific architectural considerations. MongoDB is known as the optimal database solution because of its document-oriented structure, which adapts to hierarchical nature of educational data models The relationship between subject, learning outcome and student submission was built explicitly for the flexibility of the reporting needs but with careful balancing of the data integrity. Performance and data consistency was a major concern and hence document embedding and referencing strategies were evaluated against typical query patterns. Through stakeholder analysis, authentication emerged as a key functional and security requirement. The system needed permission sets specifically for teachers' and students' and has to be implemented on the basis of role-based access control. JWT (JSON Web Token) authentication, known for its secure method for authenticating users in the web application was chosen for its stateless architecture and compatibility with React's component-based structure (Ruzzan 2024). The authentication system required secure token storage, expiration policies and middleware integration throughout the Express.js backend to enforce access controls on protected routes. The core educational functionality was around curriculum management through subject and learning outcome definition. Teachers' needed to be able to create, modify and organise curriculum content within a framework. This meant a flexible database schema should be developed to hold both standard and custom learning pathways.

The CSV import functionality added technical requirements for data validation, error handling and bulk operations. This involved careful attention to MongoDB's document insertion and error reporting in order to maintain data integrity during imports. Project submission and assessment was another key area. The system needed to manage students' submitting against learning outcomes on an individual basis and teachers' grading and providing feedback. This two-way workflow involved careful state management within the app with careful

attention to concurrency during grading. The Express.js API required strong authentication to ensure data reliability throughout the submission lifecycle and the React frontend needed intuitive interfaces for submission and assessment. Dashboard requirements varied greatly but were technically identical in terms of implementation. Teacher dashboards required the data aggregation for displaying overall student progress and pending grades. Student dashboards focused on individual progress tracking with graphical representation of completion status. Both needed efficient React component design with state management to make sure a responsive user experience. The MongoDB aggregation pipeline was needed to generate the complex datasets which was necessary for useful dashboard visualizations without impacting on the performance. Notification feature became a prime requirement to keep system users in communication. The technical implementation required real time update mechanism with possibilities of polling, WebSocket or MongoDB change streams. The chosen approach had to balance update proximity with system resource utilization especially under concurrent user scenarios. This feature required coordination across all layers of the MERN Stack from database notification storage to API endpoints to React component rendering.

The application successfully addressed Non-functional requirements including performance, security, and considered specific to educational applications. Performance specs included response time targets for common operations and concurrent user support. Security went beyond authentication to include data encryption, input sanitization and protection against common web vulnerabilities. Usability requirements - building navigation patterns, responsive design, and responsive feedback for users of varying technical proficiency. Technical Architecture requirements were involved by each layer of the MERN Stack with more emphasis on the integration points. Validation rules and indexing strategies based on anticipated query patterns were needed in the MongoDB schema design . whereas express.js API architecture has error handling and controller/service layers which helps to separate concerns. Also, to maximize reusability while keeping concerns, separate react component hierarchy is needed to be organized. Node.js environment required to be configured for production deployment with security hardening. Requirements were prioritized using the MoSCoW methodology, which is a four-step approach - must have, should have, could have, and will not have (Brush 2024) where the top considerations were that the most basic feature gets the most attention including educational impacts and technical feasibility.

Core functionality like authentication, subject management and project workflows are must-haves for that minimum viable product. That means detailed reporting, notification systems and advanced filtering capabilities are nice-to-haves that can be added later. Future enhancements like advanced analytics, batch operations and extra visualization tools are nice ideas, but not essential for now.

Validation of all the requirements was carried out in a few diverse ways to make sure they meet both educational needs and technical reality. Regular check-ins with stakeholders provided valuable feedback on how well the functionality and usability were working out. The most critical components were prototyped to make sure project was on the right track before full development began. Performance testing showed that the app would meet response time requirements under expected user loads. The comprehensive validation process ensured the final product will really serve the needs of LAB University's mechanical engineering program and show off the MERN Stack in an educational setting. Whereas the validation process also gave the better confidence and meaningful improvements in the overall solution.

This systematic requirement analysis, presented in this section, paved the way for robust development phases including architectural design along with implementation and testing. By ensuring a balance between educational vs technical deliverables a well working application is developed that would not only serve its purpose but also demonstrate best practice in web application development using the MERN Stack.

4 Graphic Design

Figma, a collaborative web application for interface design (Wikipedia 2024), was used to design the complete user interface before the development of the application which allowed everyone involved to see and agree on how the application would work. As illustrated in the figure 3, the design activity focused on figuring out how users would navigate between the distinct parts of the systems: the teacher and student dashboards, subject control screens and project submission workflows. The developed prototype played a vital role in demonstrating ideas in stakeholder sessions with LAB University's mechanical engineering team. It helped smooth out verification of the proposed application structure and how it would work. By addressing navigation and usability concerns right at the start of the project, the design stage cut down on the number of development iterations that would have been needed later on. That meant the implementation phases could move forward with a clear visual guide to the components and their relationships.

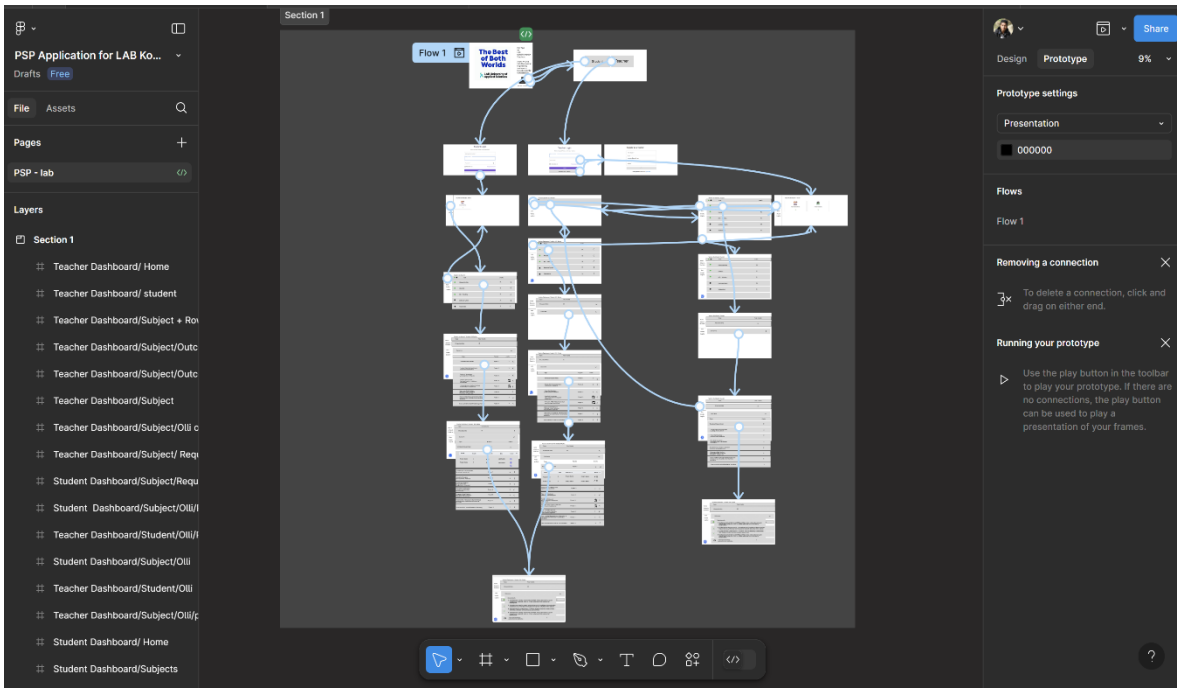


Figure 3. User Interface Design using Figma.

5 Software Implementation

5.1 Configuring the development environment

The configuration of development of environment assisted in the establishment of robust foundation for efficient implementation of the PSP application. Visual Studio Code was chosen as the main integrated development environment because of its rich support for JavaScript ecosystems and easy integration with modern web frameworks. The project's workspace was set up with a clear frontend and backend separation of concerns to allow independent development while keeping integration points stable, as shown in Figure 4.

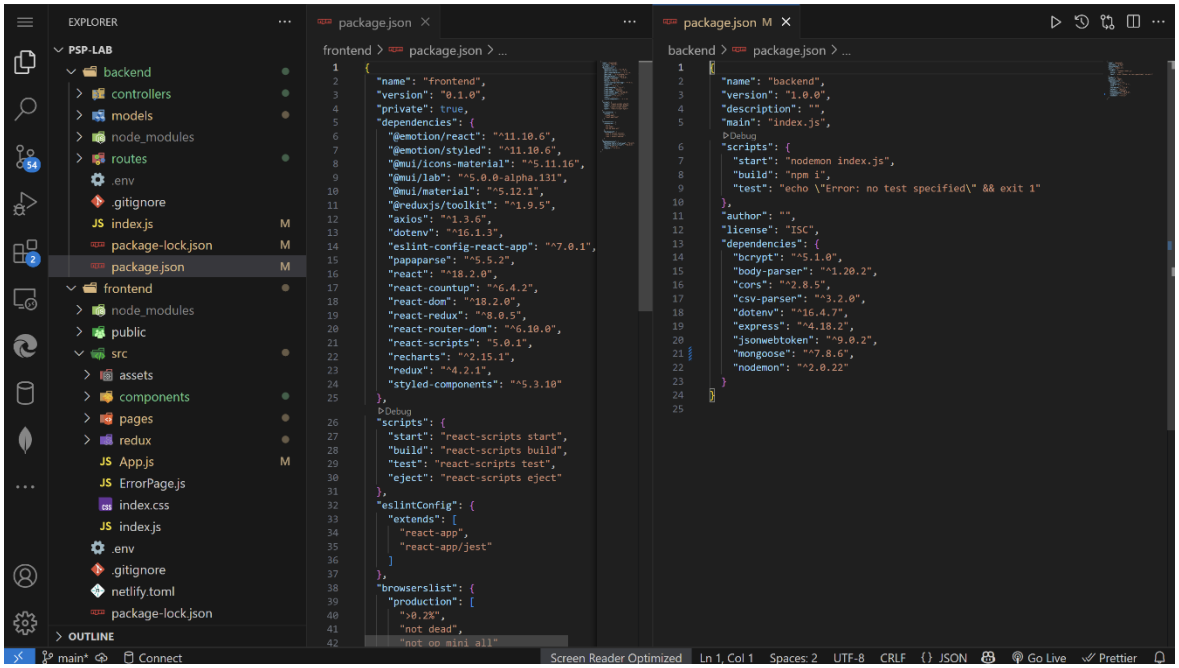


Figure 4. VS Code Project Structure with Frontend and Backend Package.json Files

For editor configuration Visual Studio Code was extended with custom extensions to enhance the JavaScript development experience. The ESLint extension enforced coding conventions on the codebase and language-specific extensions for JavaScript, React and MongoDB provided syntax highlighting, IntelliSense and in-editor documentation. A custom workspace configuration ensured consistent indentation, formatting, and file handling across the development team, which in turn maintained quality during implementation. For the frontend development environment, Create React App (CRA) was used, as stated in the package.json, giving it a zero config set up for webpack bundling, babel transpilation, and development server. This reduces the need for manual setups and provides industry-recognized optimizations for both development and production builds. Hot Module Replacement (HMR) is also turned on by default in CRA, thus allowing immediate viewing of code changes without full-page reloads, which greatly favors the development feedback

loop.The backend environment was set up with Node.js runtime using Nodemon for auto server restart during development, as written in the scripts section of package.json: "start": "nodemon index.js". This ensured that code changes were implemented eventually to the running server without manual intervention, thus keeping the development pace high and reducing possible sync issues between code and runtime state. As shown in Figure 5, the complete development workflow between both frontend and backend is running alongside showing development servers running in the termina and the application is running in the browser.

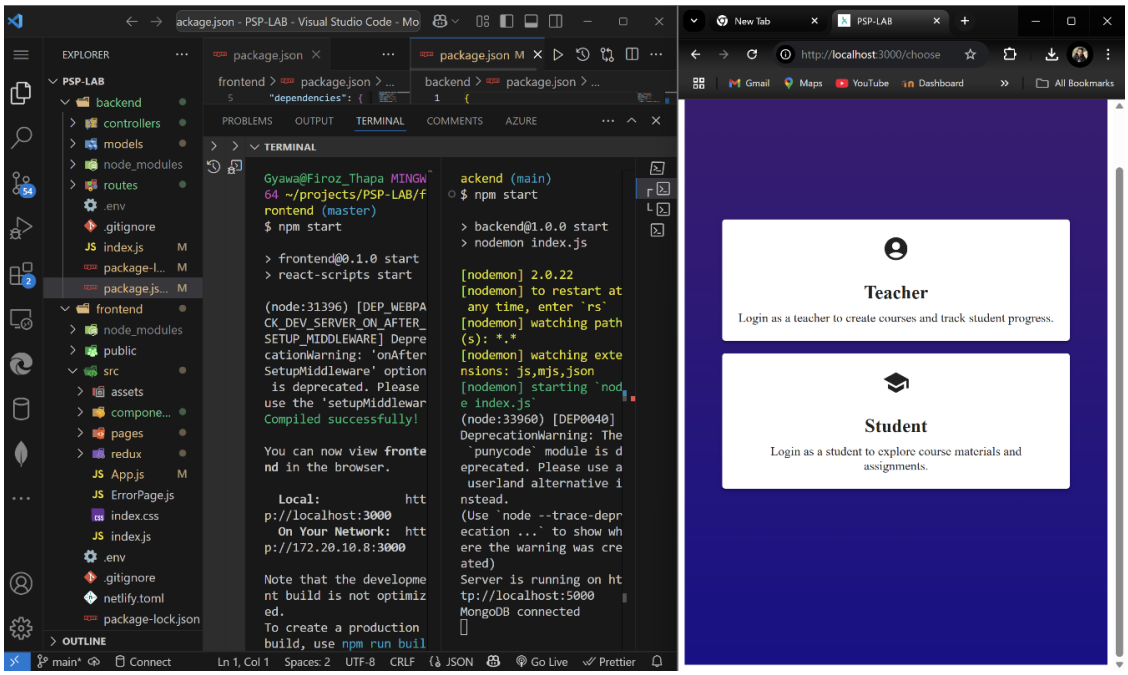


Figure 5. Integrated Development Workflow with Frontend and Backend servers running side by side and app in the browser.

5.2 Node.js and MongoDB Implementation

The backend was set up by installing and configuring the necessary technologies. Node.js helped in the establishment of java script runtime environment for server which was configured with security headers and optimized for parallel connections. The project followed the Model-View-Controller (MVC) design pattern which is a way to manage the application by splitting it into three distinct parts: model, view, and controller (Jaiswal 2020) which assists the code organized into logical components for better maintainability as shown in figure 6. The Express framework followed this pattern, using Mongoose to define schemas that provide structured data validation, relationships and business rules enforced through schema methods and middleware's. While MongoDB is schema less, Mongoose allows to add schema constraints at the application level. NPM (Node Package Manager), which imple-

ments a unique tree-based dependency model (King 2016), was used for dependency management, with package versions locked to make sure consistency across development and production environments.

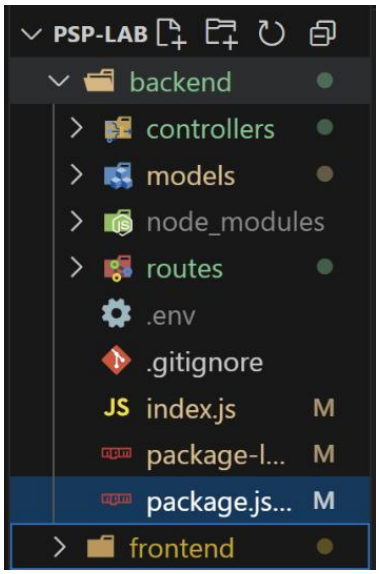


Figure 6. Node.js project structure

As illustrated in Figure 7, MongoDB was used as the primary database management system, with local development instances running on port 27017. The database was structured in around five core collections - notification, project, student, subject, and teacher with well-defined document structure for efficient educational data management. While MongoDB is schema less, schema validation and relationships were enforced at the application level using Mongoose. MongoDB Compass helped in presenting a graphical interface for database visualization and management (MongoDB, Inc 2024). Additionally, Environment variables were used to protect connection strings and authentication credentials from being exposed in the codebase. The database had strategic indexing for performance optimization, including compound indices on frequently queried fields like notification status, project names and subject topics with appropriate collation for case-insensitive operations.

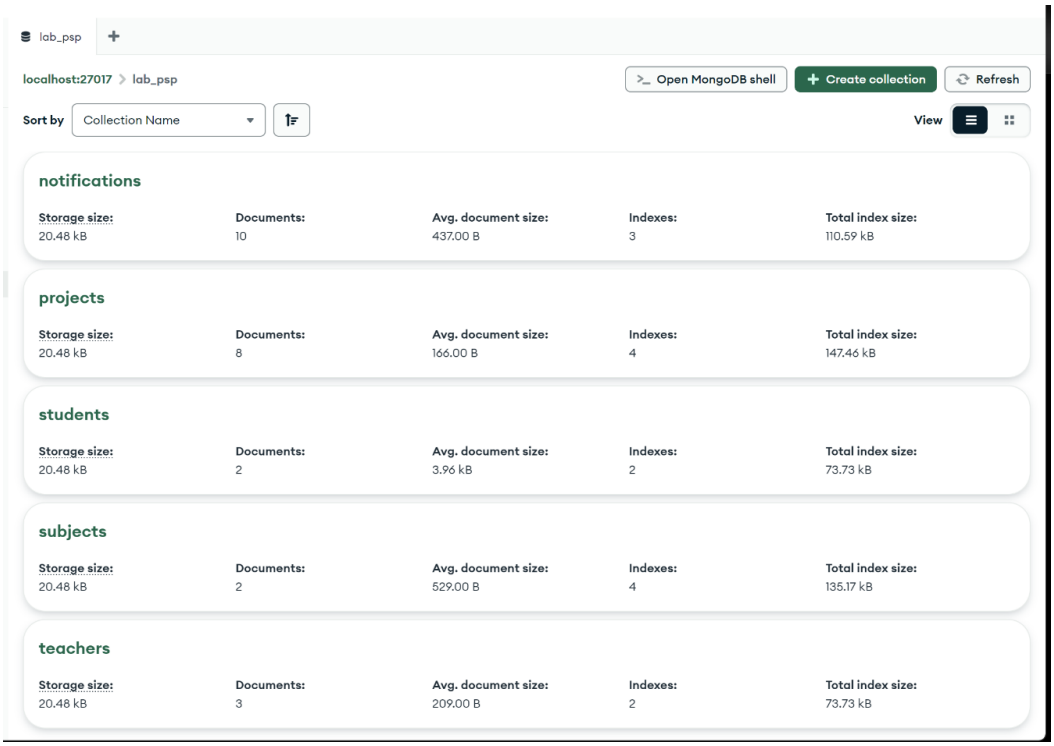


Figure 7. MongoDB Compass interface

The data model used MongoDB's document-oriented structure to represent complex educational relationships efficiently. The notification schema kept the track of student project submissions and linked them to the students, subjects, and outcomes. Where each subject had a comprehensive structure showing what was needed, and student record showed which subjects they had and what project they submitted. This demonstrates the MongoDB flexible document oriented data modelling capabilities to store hierarchical education data with the help of referential integrity via typed ObjectId reference.

For API testing and validation, Postman, a tool that is used for building, managing, and testing the APIs easily was utilized throughout the development process to ensure the endpoints worked before frontend integration(Postman 2025). As shown in Figure 8, it shows the successful retrieval of subject data via a GET request to the subject's endpoint, showing the hierarchical structure of subjects with nested outcomes, credits, and requirements. The JSON response shows the proper MongoDB document relationships, and the efficient document-oriented data structure used in the application.

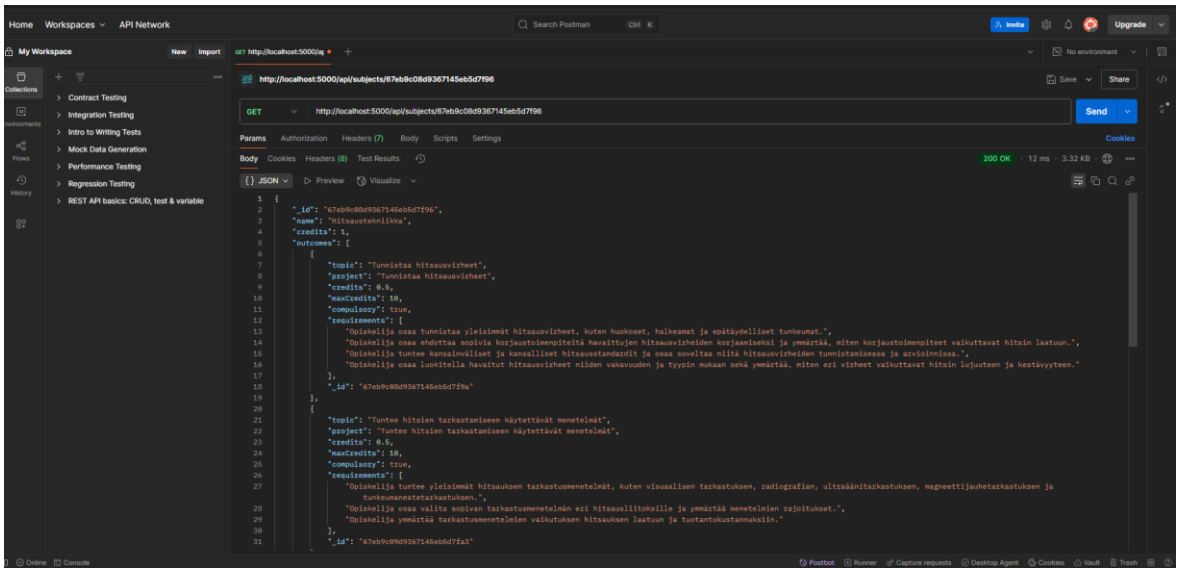


Figure 8. Postman API testing showing successful retrieval of subject data with nested outcomes.

5.3 Essential plugin installation

After thorough consideration, proper NPM packages and plugins were chosen for the PSP application's core MERN Stack. The selected packages and their integration essentially met our development needs, with a strong focus on performance and security. Two separate package environments were used for frontend (React-related dependencies) and the backend (server-side components), which eventually allowed to keep the code organized and have a separate configuration for each environment.

Frontend plugins

For the frontend React application, Material UI, a popular React library with ready-to-use, customizable components which is based on Google's Material(Material UI 2025) Design was chosen as the primary component library. One of the main reasons for selecting this library was because of its comprehensive pre-styled set of interface components that accurately hold to modern design guidelines. This choice facilitated the rapid development of uniform, accessible user interface components while still allowing for flexibility through the Material UI theming system. Redux toolkit was used to manage the state where the state was made centralized to the application, with user authentication, subject and user notifications were offered as separate slices of state. Custom middleware was built for async work and persistence of state to ensure the application behaved the same way across usage sessions. For just-in-time route navigation, React Router DOM 6.10.0 was implemented which allowed route definitions to be modelled using declarative components. Browser routing was used for client-side navigation and memory routing for any test environments that were required. Whereas Protected route components were implemented to ensure users

will meet the right access controls based on their authentication and role permissions. For data visualization, recharts 2.15.1 was implemented as a composable charting library using React components. When building data visualizations for student progress tracking and academic performance metrics, composability allowed for nicely responsive visualizations that included interactivity. Its component-based structure matched the architecture of React. To manage the CSV data, Papa Parse 5.5.2 was used for a robust parsing option with a whole suite of error handling capabilities, Papa Parse which is also known for its powerful CSV parser library that supports features such as steaming, automatic delimiter detection and worker threads(NPM 2023). Custom own wrapper functions for validation and transformation were included within the CSV parsing to confirm any imported data met the application's schema before adding the data to the database.

The technical design managed HTTP request management through Axios (version 1.3.6), implementing a centralized API service with interceptors for authentication token management and error handling. The Axios implementation involved manual-coded instance creation with base URL configuration that could change based on the development and production environments, timeout configurations, and response transforming functions. Testing API endpoints used Postman to supply a vital development tool facilitating systematic checks for REST endpoints, authentication sequences, and data consistency prior to frontend integration. Such an integration of tools provided the assured communication between the React frontend and Express backend throughout the software development life cycle.

Styled-components (version 5.3.10) perfected the Material UI implementation by providing additional component styling capability in the guise of CSS-in-JS methods. This method enabled the custom styled components to be defined and maintained theme consistency with the Material UI while offering greater control over component appearance and feel.

Backend plugins

At the server side, a set of complementary plugins supplemented the capabilities of Express.js and Node.js, as illustrated in the Figure 4 at the backend package.json. The security model employed authentication with the use of bcrypt (version 5.1.0) for hashing passwords and JSON Web Tokens (version 9.0.2) for stateless authentication. In order to balance security and performance requirements during password validation operations the application configured bcrypt with a work factor of ten.

The safe configuration of Cross-Origin Resource Sharing (CORS) was ensured by the use of version 2.8.5 of the cors middleware, which strengthened appropriate security policies across API access. The configuration specified allowed origins, methods, and headers to

enable secure cross-domain communication while restricting unauthorized use of sensitive API endpoints. The server structure employed body-parser (version 1.20.2) to manage various request payload structures, including JSON data and form data that enhanced the data parsing. The solution was configured by this middleware with proper size limits and parsing options in order to prevent any possibilities of denial of service attacks while sustaining efficient request processing.

The database solution was synchronized with MongoDB integration through Mongoose (version 7.8.6) for schema validation, middleware support, and simplified query construction. The environment included optimizations within connection pooling, index definition tools, and advanced schema custom methods for educational data management operations. The approach enhanced the application data integrity while simplifying complex document interconnectivities.

In order to work efficiently, the engineering process configured Nodemon (version 2.0.22) to monitor file changes and automatically restart the Node.js server in development mode. The setup involved particular custom watch patterns targeted towards the core server pieces and excluded test files and documentation in order to optimize the development feedback loop without unnecessary restarts.

Application framework standardized front and backend environment variable management with dotenv packages, version 16.4.7 on the back end and version 16.1.3 on the front end. This security feature protected sensitive configuration data such as database connection strings, JWT secret keys, and API endpoints and made them unavailable in the source code repository.

For server-side CSV processing, csv-parser (version 3.2.0) complemented the frontend PapaParse support, which provided stream-based CSV file handling for efficient processing of large data sets in import operations. Whereas the backend implementation included custom transformation of streams for data validation and normalization which made sure that the imported educational content maintained consistency with the application's data models.

The strategic selection and integration of these plugins established a robust development ecosystem while maintaining application performance and security. Each dependency addressed specific technical requirements of the PSP application, extending the core MERN Stack capabilities to fulfil the educational management needs of LAB University's mechanical engineering program.

5.4 User Interface Development

The process of developing the user interface for PSP (Personal Study Plan) application was conducted using React.js with Material UI components which eventually helped in creating the responsive, intuitive, and visually consistent experience across the platform. This phase translated the Figma prototype documented in the Graphic Design section into functional React components, maintaining particular focus towards supporting the distinctive workflows for teachers' and students' roles. The realization kept component reuse in consideration, state management optimality, and consistent design language throughout the app. Both the teacher and student dashboards provide essential overviews that are tailored to each of their requirements, as demonstrated in Figure 8 and 9.

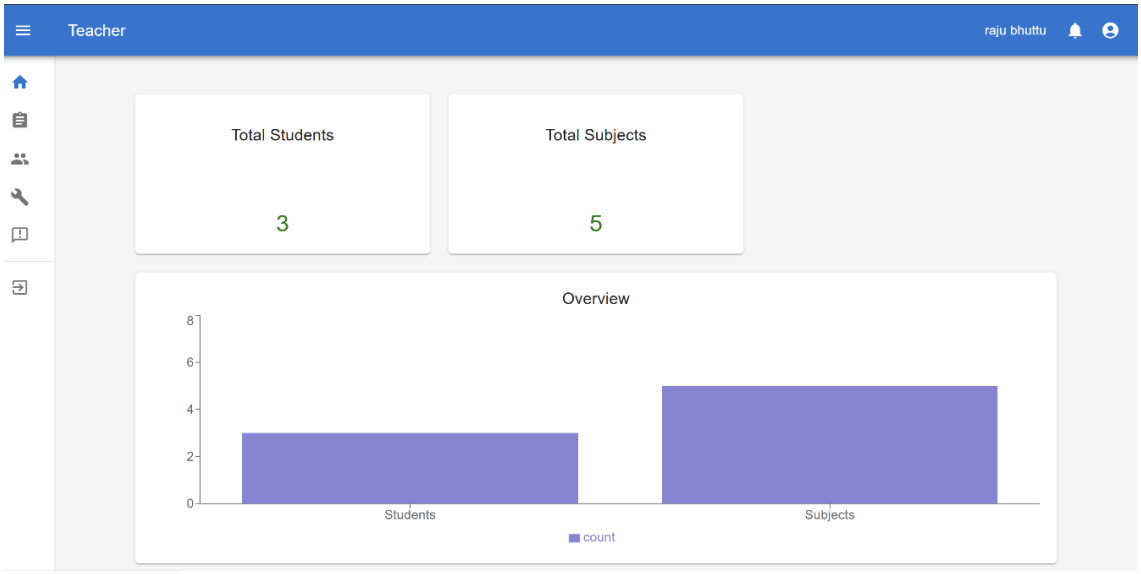


Figure 8. Teacher Dashboard Interface

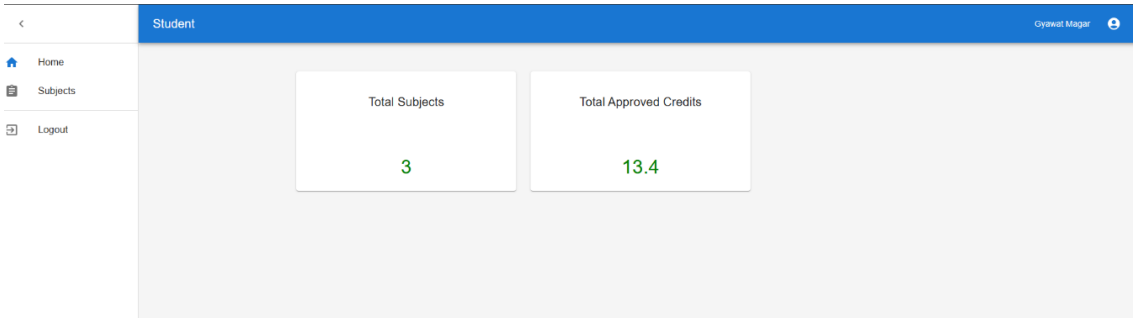


Figure 9. Student Dashboard Interface

The UI architecture was built on a components-based model with clear separation of concerns. This allowed the interface to be broken down into maintainable and reusable units. After the careful consideration, Material UI was chosen as our components library because

it offers a robust collection of usable components, a uniform design language and full theming capability that could be used to match LAB University's visual identity. That helped us in the development efficiently without sacrificing consistency across the application.

Several key design principles guided the interface implementation, such as:

- Role-based interfaces for teachers' and students', each with their own navigation and features.
- Managing the architecture from overview (dashboard) level down to specific details.
- Disclosure of complex information shown step by step to enhance usability.
- Maintaining a standard colour schemes, spaces, and typography throughout the application.
- Ensuring the design is responsive and adapts across different device sizes.

As illustrated in Figure 10, the mobile view of the project management interface layout is able to adapt to smaller screen sizes while maintaining all the critical functionality. The navigation sidebar collapses, and content elements reflow to fit the narrower viewport. Teachers' and students' both get a coherent user experience despite the different functional focuses of their interfaces.

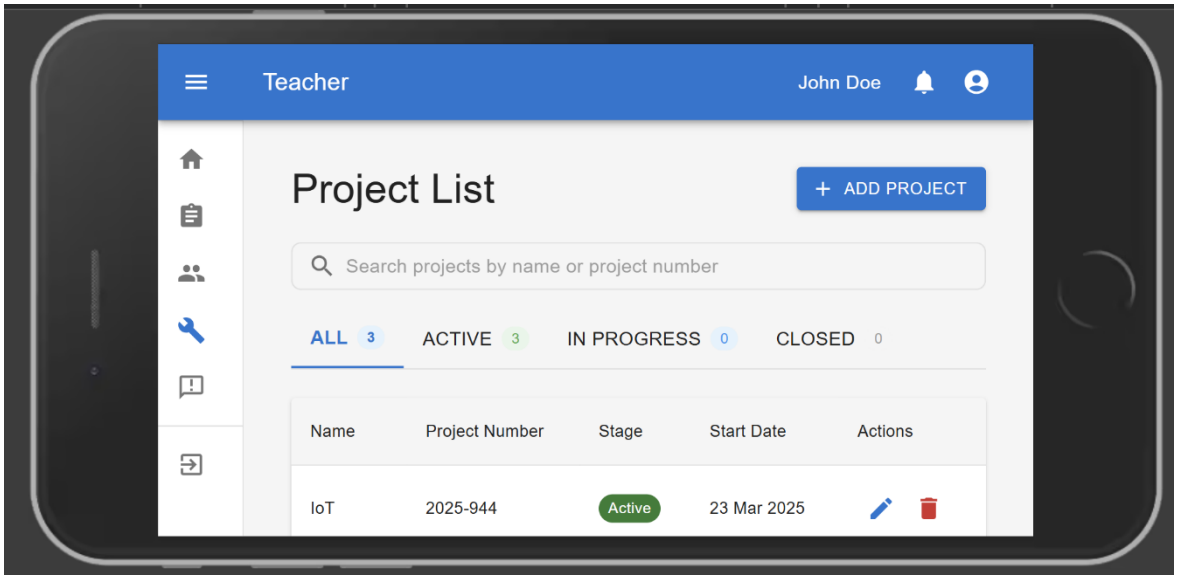


Figure 10. Mobile Responsive View of Project Management Interface

In the dashboard interfaces, teachers' are able to see the comprehensive statistics focused on student progress, subject management, and pending notifications, while student interfaces highlight the credit accumulation, available subjects, and project submission opportunities. Regardless of these different functional focuses, both interfaces are able to manage a consistent design language and navigation patterns to ensure a reliable user experience.

The component structure was managed into three main categories: layout components that provides structural foundation (App Bar, Drawer, Container), page-level components that are representatives of main application views (Homepage, Subject Management, Student Progress), and reusable UI components providing consistency throughout the application (Table View Template, Search Bar, Popup). The subject management interface uses an accordion-style expandable row pattern to let teachers to navigate the hierarchical subject structure and their outcomes in an efficient way.

State management was a hybrid approach as redux was implemented for global application state (authentication, subject data, notifications) and React's use State hook for component-specific state (form inputs, modal dialogs, pagination). This strategy helped in the optimization of performance by centralizing critical data management while keeping user interface specific state encapsulated within relevant components.

Moreover, the authentication interface lets users identify as either teachers' or students', determining which interface they are accessing. Teacher dashboards provide comprehensive statistics on student progress, subject management, and pending notifications. Meanwhile, student interfaces focus on credit accumulation, available subjects, and project submission opportunities.

Subject management uses a tabbed interface that provides filtering capabilities, search functionality and an accordion-style expandable row pattern. This interface also supports both individual subject creation and batch importing through CSV files. The feature of importing using CSV facilitates that they verify imported data before they make any modifications to the database.

The student management interface in a teachers' interface is also a searchable table of enrolled students' which is linked to detailed progress tracking views. These views display subject completion status, project submissions, assessment status and credit accumulation through visual indicators and progress bars. This comprehensive visualization of student progress provides teachers to efficiently monitor and assess academic advancement.

Project management organizes projects by status (Active, In Progress, Closed) with filtering tabs for each category. Additionally, each project entry shows automatically generated identifiers, status indicators and relevant metadata, with inline editing capabilities.

Additionally, the notification system provides alerts about student submissions requiring review, with status filtering, search functionality, and direct assessment capabilities as illustrated in Figure 11.During the development of the user interface, several technical changes appeared, one of these were solved successfully by the implementation of color-coded notification status which allowed users to quickly identify items visually. These were addressed through careful component design, robust parsing logic, efficient state management and centralized calculation approaches. The resulting user interface has improved upon the previous Excel-based system in LAB University's mechanical engineering program.

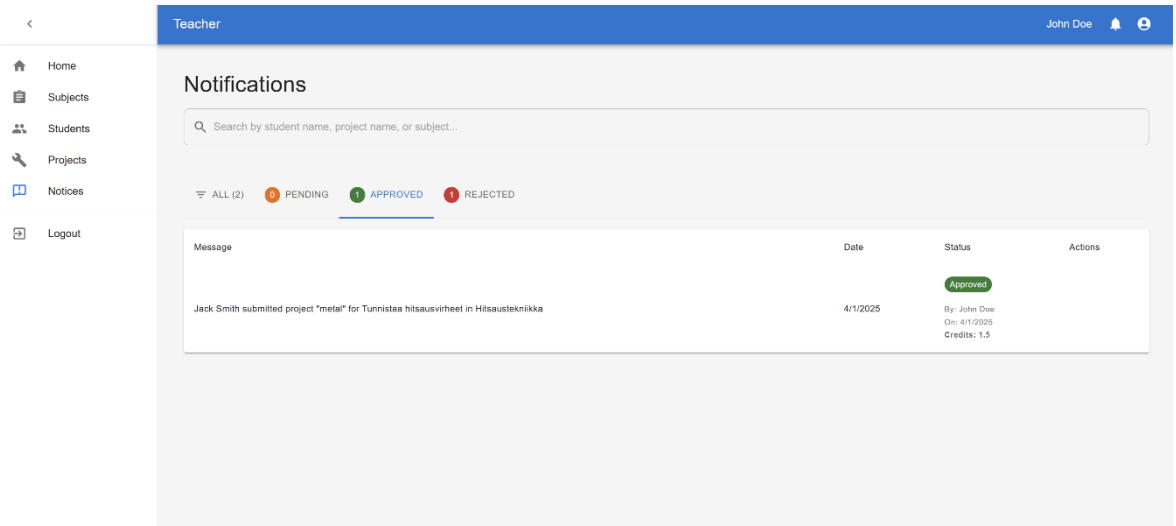


Figure 11. Notification page interface

6 System Features

6.1 User Authentication and Role Management

The PSP application has a full authentication framework with role-based access control, so data is secure, and the system is used correctly. For the authentication feature JSON Web Token (JWT) was utilized for stateless authentication, which is known for its secure and efficient way to verify identity across the application. As the authentication is successful, the server generates a token with the user's encrypted credentials and role information which is then verified for each protected resource request.

Role separation between teachers' and students' is implemented by using different sets of permissions. Teachers' have privileges to manage curriculum, create subjects, define outcomes, assess students' and registering new students'. The authentication system checks and verifies the teachers' credentials before allowing access to administrative functions like project approval workflows or requirement specification. Whereas, students' have access to view their assigned subjects, track their progress, submit projects for assessment meaning they can only see their own academic records and submission workflows.

Password security is managed through industry standard bcrypt hashing with a configurable work factor of ten, so it is secure but not too slow (npm 2024). This way even if the database is compromised the password is not exposed.

6.2 Project Submission and Evaluation Workflow

The PSP application has a structured and transparent assessment process through a project management framework. The system has a sophisticated project lifecycle starting with teacher defined project templates that include critical metadata: unique identifiers, names, current stage (active, in-progress, closed) and temporal tracking information. These templates are the tools for assessment across the curriculum.

Submission process allows the students' to select projects for learning outcomes, to specify credit requests within parameters delineated and mark evidence of proficiency. Each submission is timestamped, tracked as status, and tied to curriculum items.

On submitting the application puts into motion automated notice workflow to send notification of the appropriate teaching staff through central assessment queue. This queue shows submissions in chronological order with clear visual indicators of pending assessments so they can be processed on time. The assessment interface gives teachers' a full evaluation toolkit:

- Binary decision options (approve/reject) with visual indicators.
- Credit allocation controls with validation against specified parameters.
- Mechanisms of structured feedback for qualitative evaluation documentation.
- Automating the status updates for transparency.

The system keeps a full audit trail of all submissions and assessments, original requests, modifications, final decisions, and timestamps. This assists with academic honesty and supplies details for continuous improvement.

6.3 Academic Content Orchestration System

The PSP application has a curriculum management system that allows for precise educational content organisation. The curriculum model is hierarchical and maps pedagogical relationships:

- Subjects are the large organisational units with credits values.
- Learning outcomes are competency specific objectives within subjects.
- Requirements state correct assessment criteria for each outcome.
- Projects are real world demonstrations vehicles for competency validation.

Administrators benefit from comprehensive subject management capabilities that support curriculum development through intuitive interfaces for creation, modification, and sequencing of educational content. Throughout these operations the system maintains referential integrity to ensure curriculum coherence.

Learning outcome definition is conducted through a structured interface while capturing essential educational parameters such as:

- Clear topic identifiers for clear communication
- Minimum and maximum credit allocations to reflect outcome significance.
- Compulsory/optional designations to show what is required in the curriculum.
- Requirement specifications that express precise competency expectations.

The application has an advanced framework for requirements that lets to articulate assessment criteria in as much detail as needed. This framework supports both standardized and customized requirement sets across the curriculum.

That framework is what makes educational content management so much easier. Data could be imported in bulk via standardized CSV templates, and the system validates, detects duplicates and manages errors to keep data integrity intact during those bulk operations.

6.4 Real-time Communication and Alert Framework

Real-time communication is key to keeping stakeholders informed. The PSP application has a notification system that lets users to exchange information on the fly. The notification system tracks academic workflows and generates alerts as needed. It uses an end-to-end event tracking mechanism to keep everything on record. The system records submission activity, grading decision, requirement updates and status changes to keep stakeholders informed.

Project based notifications have submission metadata like project id, student information, requested credits and timestamp information. These are displayed in teachers' interfaces with priority indicators to help with workflow.

Assessment notifications have rich decision information for students', such as:

- Approval status with appropriate icons.
- Awarded credit values with variance calculation done from requests.
- Instructor name for accountability purposes.
- Qualitative feedback for improvement.

The notification management interface is equipped with filtering by message status (pending, approved, rejected) that allows users to focus only on specific types of messages. This feature is helpful in high volume environments where it assists in updating the workflow and improving the efficiency by quickly filtering out through large numbers of notifications.

7 Conclusion

This thesis has demonstrated the effective application of the MERN Stack in developing a comprehensive Personal Study Plan (PSP) application for the mechanical engineering department at LAB University of Applied Sciences. The primary goal of the study was to transform the manual Excel-based curriculum management into a digital platform that improves educational administration.

In this study, the main research question " How an effective digital application for managing the students' competencies in a project based education system could be carried out using MERN Stack?" was successfully addressed by exploring several key technical areas. Firstly, the examination of MERN Stack fundamentals was carried out, explaining how MongoDB, Express.js, React, and Node.js integrate together to form a unified JavaScript development. Secondly, Systematic requirement analysis was developed with the help of regular weekly collaboration with the faculty members. Additionally, MoSCoW method was taken into consideration to prioritize features based on the educational impact and technical feasibility.

The designing of the user interface was developed using Figma. During the designing of the application Figma helped in the confirmation of workflows before implementation as well as minimized development iterations by resolving navigation and usability issues at the early stage itself. The software implementation was carried out in a step by step technical progression from environment setup to feature deployment, emphasizing architectural choices supporting crucial educational workflows. The testing of the application successfully proved that MongoDB's document-driven architecture efficiently modelled educational hierarchies, whereas Express.js was able to establish secure role-based API access implementing JWT authentication. Additionally, the component-driven architecture by React improved both development productivity as well as user experience by utilizing reusable UI parts, and Node.js implemented a single programming environment, thereby simplifying development.

In order to ensure the consistency of design throughout the application, Material UI elements were implemented for the development of a uniform visual language which was applied across the platform to present a cohesive user experience across various functional priorities for students' and educators'. The application is able to provide dynamic robust system features that integrate user authentication using role-based access control, workflow for project submission and assessment, academic content management, and notification systems for real-time alerts. Combining these features helps to enable a responsive

digital platform for improved administration efficiency and academic clarity, permitting both students' as well as educators' to better interact with the process of managing curriculum.

In conclusion, the PSP application that is developed specifically for the mechanical department of LAB University of applied sciences is able to solve the curriculum management problems using the MERN Stack efficiently. The application shows how modern web technologies can be implemented to develop education management systems that are both technically sound as well as pedagogically effective, digitizing manual processes to improve convenience for all parties involved.

Limitations

During the process of development this project has encountered some limitations also. Firstly, the application was designed specifically for the mechanical department only meaning it might not work well for the other department without some specific changes. The deployment was not carried out to the university servers yet, which means we were not able to know what problems might arise when collaborating with existing university systems. Finally, the focus of this thesis was on making a web version of the application rather than a mobile application, even though many students' prefer using their phones for educational tools.

Future implementation

The next step in development would be creating a dedicated mobile application for the PSP system. A mobile application would extend the platform's accessibility beyond traditional classroom settings, allowing students' to check their progress and receive real-time notifications on their personal phones. This mobile version would enable students' and teachers' to efficiently manage their competencies more conveniently from anywhere, addressing the increasing reliance on mobile devices for educational tools. Additionally, the application could be further enhanced for other faculties like Information Technology, Business, Health by incorporating specific changes suitable for the departments as education system continues to evolve in our digital age.

References

Bocasay. 2024. MERN Stack data flow diagram. Bocasay Blog. Retrieved on 2 April 2025. Available at https://www.bocasay.com/how-does-the-mern-stack-work/

Express.js. 2024. Guide for using middleware. Express Documentation. Retrieved on 2 April 2025. Available at https://expressjs.com/en/guide/using-middleware.html

Mojeed, I. 2025. What is the virtual DOM in React? LogRocket Blog. Retrieved on 23 April 2025. Available at https://blog.logrocket.com/the-virtual-dom-react/

GeeksforGeeks. 2024. MERN Stack: Introduction and Architecture. Tutorial. Retrieved on 22 April 2025. Available at https://www.geeksforgeeks.org/mern-stack/

Jaiswal, A. 2020. Understanding the MVC Pattern with NodeJS. Blog. Retrieved on 27 March 2025. Available at https://dev.to/abhishekjaiswal_4896/understanding-the-mvc-pattern-with-nodejs-1fpg

King, A. 2016. Understanding the npm dependency model. Blog. Retrieved on 9 April 2025. Available at https://lexi-lambda.github.io/blog/2016/08/24/understanding-the-npm-dependency-model/

Material UI. 2025. Getting Started with Material UI. Documentation. Retrieved on 30 March 2025. Available at https://mui.com/material-ui/getting-started/

MongoDB, Inc. 2023. Introduction to MongoDB. MongoDB Documentation. Retrieved on 2 April 2025. Available at https://www.mongodb.com/docs/manual/introduction/

MongoDB, Inc. 2024. What is MongoDB Compass. MongoDB Compass. Retrieved on 7 April 2025. Available at

https://www.mongodb.com/docs/compass/current/#:~:text=MongoDB%20Compass%20is%20a%20powerful,macOS%2C%20Windows%2C%20and%20Linux

Node.js. 2025. Overview of blocking vs non-blocking. Node.js Documentation. Retrieved on 9 April 2025. Available at https://nodejs.org/en/learn/asynchronous-work/overview-of-blocking-vs-non-blocking

NPM. 2023. What is react-papaparse? NPM package repository. Retrieved on 9 April 2025. Available at https://www.npmjs.com/package/react-papaparse

npm. 2024. How safely store a password using bcrypt package? Documentation. Retrieved on 20 April 2025. Available at https://www.npmjs.com/package/bcrypt

Postman, Inc. 2025. API Development and Testing Tools. Postman Documentation. Retrieved on 1 April 2025. Available at https://www.postman.com/product/api-client

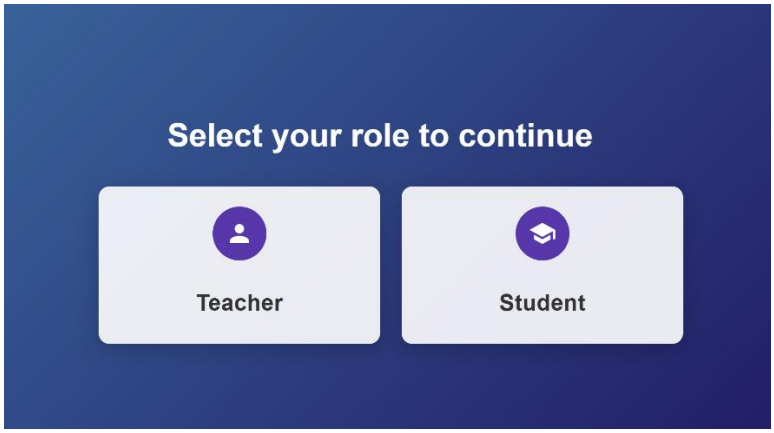
Ruzzan, A. 2024. How to implement JWT authentication with MERN Stack. Documentation. Retrieved on 9 April 2025. Available at https://medium.com/@aaqil.ruzzan/how-to-implement-jwt-authentication-with-the-mern-stack-4948d4423c64

Brush, K. 2024. What is MoSCoW method? TechTarget. Retrieved on 23 April 2025. Available at https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method

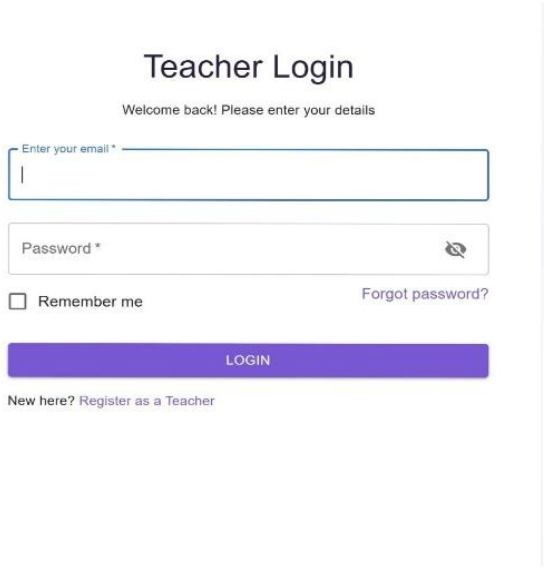
Wikipedia. 2024. Figma (software). Wikipedia. Retrieved on 9 April 2025. Available at https://en.wikipedia.org/wiki/Figma

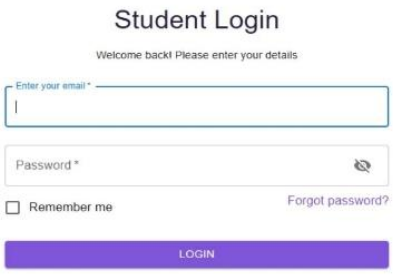
Appendix. Development prototypes and interfaces

Choose User page.

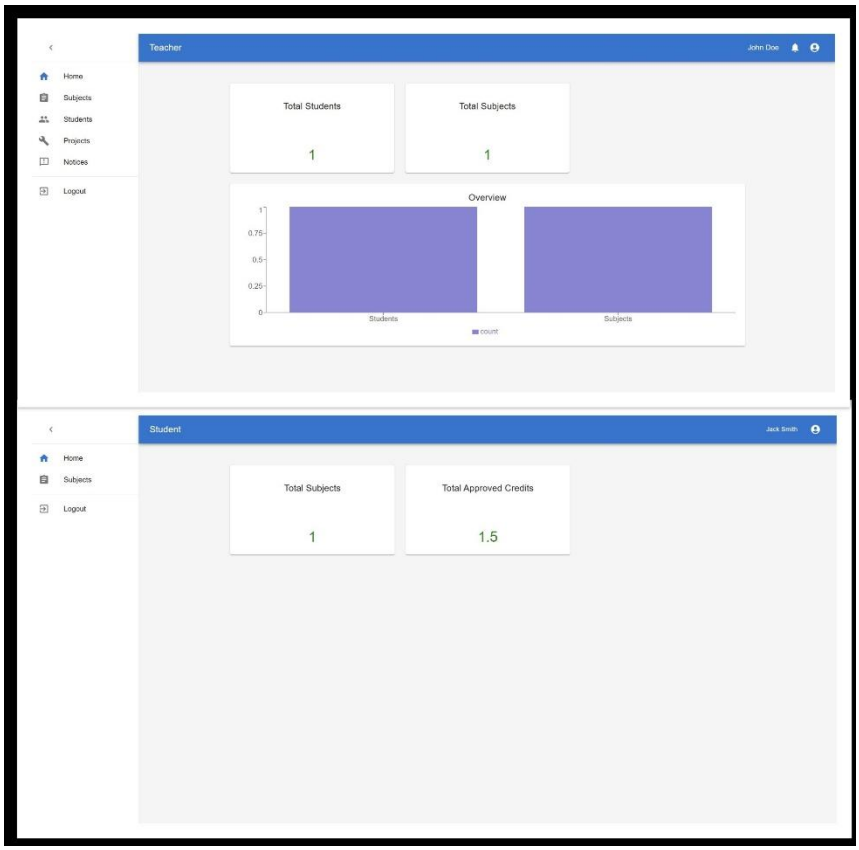


Login page.

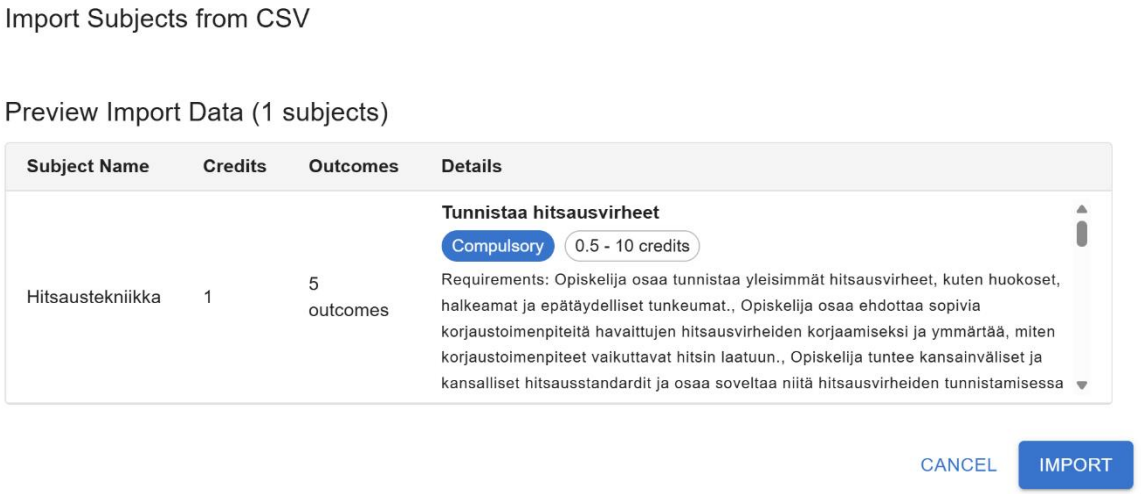




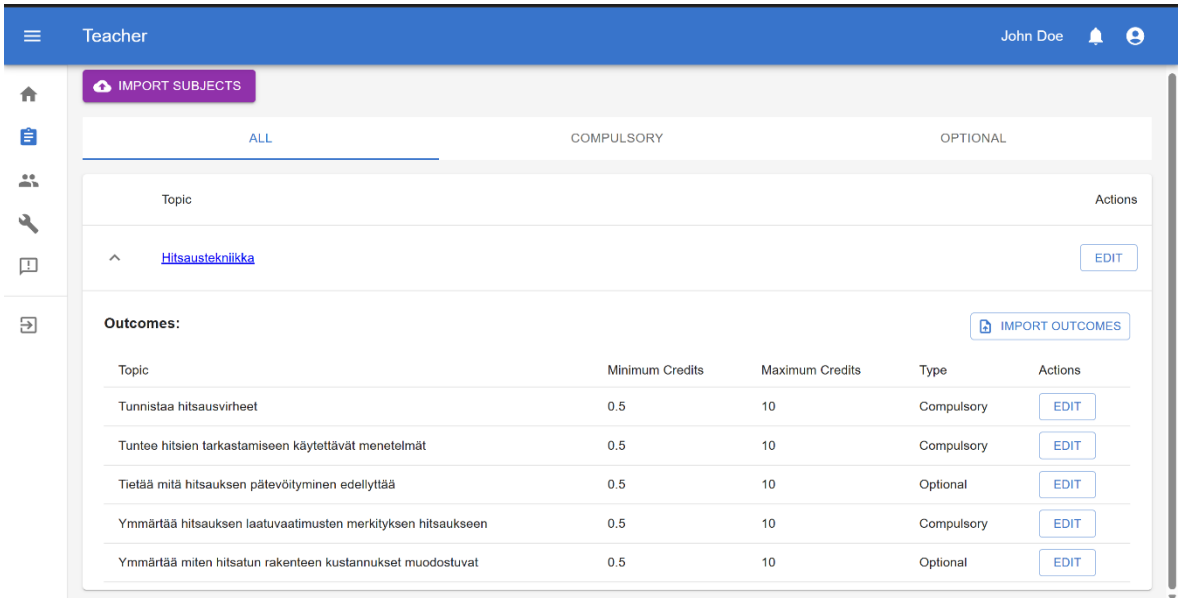
Dashboard (Teacher and Student).



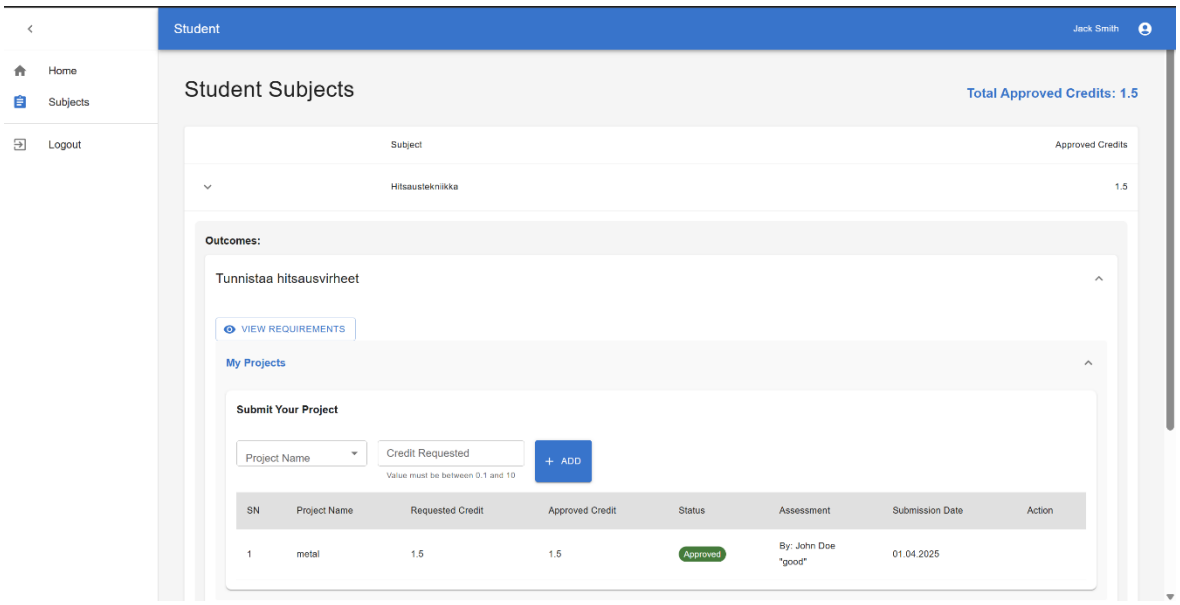
CSV Import.



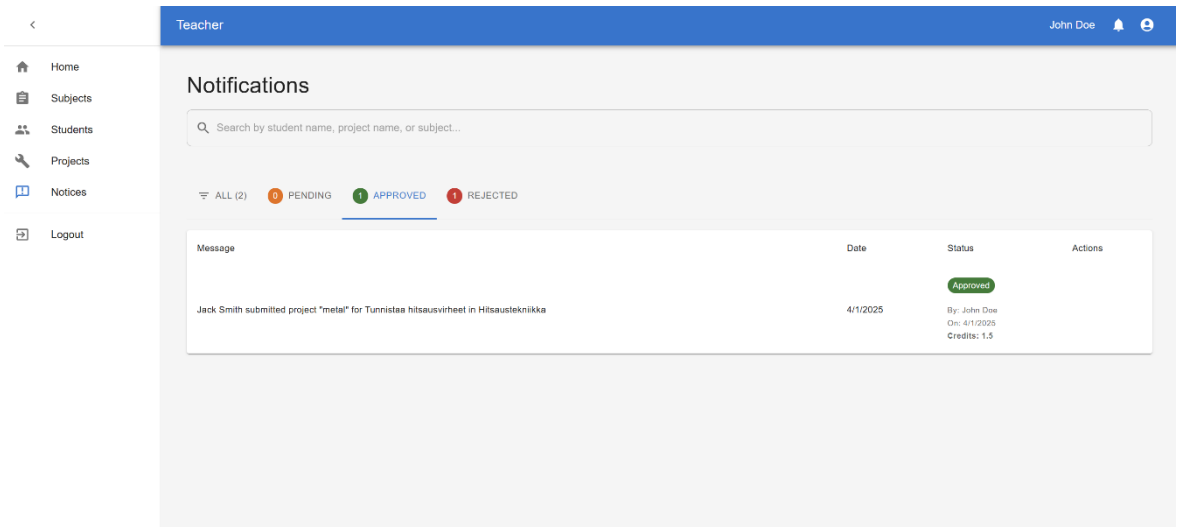
Teacher Subjects page.



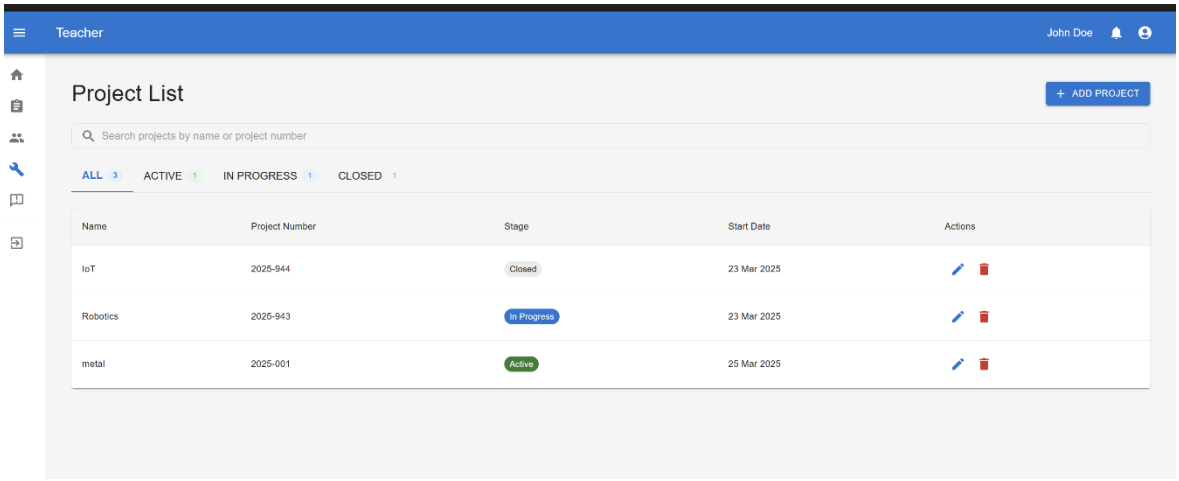
Student subject page.



Notification page.



Project page.



Requirement Page