

Chapal Shaik

Robotics Lab Management System

A Web-Based Platform for Managing and Organizing a Robotics Lab

Robotics Lab Management System

A Web-Based Platform for Managing and Organizing a Robotics Lab

Chapal Shaik
Bachelor's Thesis
Spring 2024
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology

Author: Chapal Shaik

Title of the thesis: Robotics Lab Management System

Thesis examiner(s): Meija Lohiniva

Term and year of thesis completion: Spring 2024

Pages: 46

This thesis focused on developing a web-based demonstration of a concept user interface for robotics lab operations aimed at simplifying administrative and student workflows and improving resource management. The primary goals were to enhance the system's usability and security, streamline the tracking of lab instruments, and assist the lab administrator in performing tasks efficiently. Inspiration was drawn from the Oulu University of Applied Sciences robotics lab, which was used as the basis for designing this system. The existing system kept all records manually in an Excel sheet for equipment borrowing and returning, which was time-consuming. Developing a new system with increased functionality was necessary to address these challenges and provide a seamless and efficient interface.

The theoretical framework analyzed the system utilized in the OAMK robotics lab and its difficulties in managing lab operations, particularly its proneness to errors and lack of collaboration features. The framework explored the development of a web-based system to effectively manage lab operations and keep them updated with the latest technology. It integrated concepts from web-based systems, database management, user-centered design, and agile development approaches to tackle these difficulties, providing a more efficient and secure solution. Using these principles, the study aimed to show the advantages of a web-based robotics lab management system (RLMS) in improving resource management and operational efficiency.

The document covered the background, objectives, methodology, and implementation of the RLMS system application, which involved role-based access control, inventory management modules, lab resource tracking features, and communication tools.

The thesis concluded by analyzing the project's achievements, practical expertise, and future development and expansion directions. This research contributed to the field by demonstrating the potential of a web-based UI built with ASP.NET API, React UI, and an MSSQL database to enhance robotics lab operations and develop a more efficient and innovative research environment.

Keywords: Robotics Lab Management, Resource Management, Web development, Database

CONTENTS

| | |
|---|----|
| LIST OF ABBREVIATIONS..... | 6 |
| 1 INTRODUCTION | 7 |
| 2 OBJECTIVES | 9 |
| 3 SYSTEM ANALYSIS | 10 |
| 3.1 User Management Module | 11 |
| 3.2 Inventory Management Module | 11 |
| 3.3 Lab Resource Management Module | 12 |
| 3.4 Communication and Collaboration Module..... | 13 |
| 4 TECHNICAL INTRODUCTION | 14 |
| 4.1 Backend Development | 15 |
| 4.1.1 ASP.NET Web API | 15 |
| 4.1.2 Microsoft SQL Server (MSSQL)..... | 16 |
| 4.1.3 Clean Architecture Design | 16 |
| 4.2 Frontend Development..... | 17 |
| 4.2.1 React.js..... | 17 |
| 4.2.2 Benefits of this Technical Stack | 19 |
| 5 IMPLEMENTATION..... | 20 |
| 5.1 Front Page Layout..... | 20 |
| 5.1.1 Login/Signup Button..... | 21 |
| 5.1.2 Slider..... | 22 |
| 5.1.3 Main Modules..... | 22 |
| 5.1.4 University News | 22 |
| 5.1.5 Latest Research..... | 22 |
| 5.1.6 Featured Content..... | 23 |
| 5.2 User Management Module | 23 |
| 5.3 Inventory Management Module | 24 |
| 5.3.1 Equipment Management..... | 24 |
| 5.3.2 Purchase Order..... | 26 |
| 5.3.3 Low-Stock Alerts | 26 |
| 5.3.4 Inventory Control..... | 27 |
| 5.4 Lab Resource Module | 28 |

| | | |
|-------|---|----|
| 5.4.1 | Equipment Request System..... | 28 |
| 5.4.2 | Resource Allocation | 29 |
| 5.4.3 | Result Sharing | 30 |
| 5.5 | Communication and Collaboration Module..... | 31 |
| 5.6 | Backend | 32 |
| 6 | TESTING | 35 |
| 6.1 | Testing Methodologies | 36 |
| 7 | SECURITY | 38 |
| 7.1 | Security Measures..... | 38 |
| 7.2 | Vulnerability Assessment | 39 |
| 8 | DEPLOYMENT | 40 |
| 8.1 | Configuring ASP.NET Core Project..... | 40 |
| 8.2 | Deployment Steps | 41 |
| 9 | CONCLUSION..... | 42 |
| | REFERENCES | 43 |

LIST OF ABBREVIATIONS

| | |
|-------|---|
| RLEMS | Robotics Laboratory Equipment Management System |
| RLMS | Robotics Lab Management System |
| SQL | Relational database language |
| HTTP | Application protocol for distributed, collaborative, hypermedia information systems |
| XML | Extensible Markup Language |
| UI | User interface |
| UX | User Experience Design |
| API | Application Programming Interface |
| DOM | Document Object Model |
| CORS | Cross-Origin Resource Sharing |
| IIS | Internet Information Services |
| XSS | Cross-Site Scripting |

1 INTRODUCTION

Integrating automation into robotics laboratories is essential for efficient and effective work management. Researchers and students can allocate more time to crucial components of their research projects by automating repetitive processes like data collecting, analysis, and equipment management. This tool aims to simplify laboratory activities, ultimately improving efficiency and productivity in research.

Practical robotics courses have become increasingly popular, requiring more extensive labs with more instruments (1). The ongoing challenges in effectively controlling equipment usage, storage, and documentation lead to confusion and inefficiency. In addition, unofficial lending activities put equipment security at risk. The integration of equipment data, maintenance logs, and borrowing processes into a single platform is lacking in the current systems.

Implementing a comprehensive system that monitors equipment usage, ensures availability for research and instructional purposes, classifies items into specific categories for effective retrieval, and facilitates prompt reporting and repair to reduce downtime and malfunction rates is imperative to address these issues. Therefore, building a laboratory equipment management system for business process innovation is essential to improving laboratory management (2).

Automating lab procedures enables us to gather information on the frequency and type of equipment usage. This data is invaluable for determining future equipment requirements and scheduling maintenance tasks. By leveraging this information, we can enhance lab productivity and reduce costs.

This project suggests creating and implementing a special robotics laboratory equipment management system (RLEMS) that uses automation to solve the difficulties associated with overseeing intricate robotics labs. The RLEMS seeks to optimize lab operations to free up researchers and students for their primary research tasks.

The Robotics Lab Management System (RLMS) is being developed, utilizing real-world applications and recent research in laboratory management systems. This initial part examines the literature to pinpoint approaches and prospective fields for future innovation. Robotics Lab Specific

Systems, Equipment Management Systems, and Laboratory Management Systems (LMS) are essential areas of study. Research on Laboratory Management Systems (LMS) offers important insights into system features, problems, and user needs. Creating an efficient RLMS requires understanding how LMS handles issues like resource allocation, user access control, and data management (3).

Additionally, studies on equipment management systems illuminate features, including automatic low-stock alerts, maintenance scheduling, and inventory tracking. Investigating specialized equipment management systems can improve RLMS effectiveness (4). This will be accomplished by offering a consolidated system for maintenance logs and equipment data and utilizing policies. This thesis investigates methods to establish a more structured and influential future for robotics research projects by implementing such a system.

2 OBJECTIVES

Creating an innovative Robotics Lab Management System (RLMS) that tackles the main issues facing modern robotics labs is the goal of this project. The objective is to provide a smooth, effective atmosphere that benefits scholars and learners. The following are the main goals.

Maximize Efficiency and Reduce Mistakes: The RLMS will simplify operations by automating equipment usage, borrowing, and maintenance procedures. This will reduce human labor and mistakes.

Boost equipment availability and tracking: The RLMS will offer real-time insight into equipment's position and data through a centralized platform, guaranteeing the best possible resource allocation for research initiatives.

User Management and Borrowing Equipment: Formal requesting procedures and organized user access restrictions will reduce confusion, improve equipment management, and reduce security threats related to the informal sharing of equipment.

Data-Driven Decision Making: The RLMS will facilitate well-informed decisions on resource allocation and future equipment purchases by gathering and evaluating equipment usage and maintenance data.

Workflow Coordination and Collaboration: The RLMS's features, such as discussion forums and announcement boards, will facilitate communication in the lab setting and encourage teamwork.

Increasing Lab Output: The RLMS aims to dramatically boost lab productivity through automation and effective resource use, leading to higher research output and possibly lower costs.

3 SYSTEM ANALYSIS

The System Analyst bridges the gap between researcher demands and technology capabilities to produce a streamlined Robotics Lab Management System (RLMS) by automating lab procedures and designing a user-friendly system. This section delves into the design of each module within the RLMS. While any software inherently has a structure, sound design principles lead to a system with simplicity, consistency, and a clear underlying logic. Conversely, poorly designed systems can be cluttered and inefficient (5; 6).

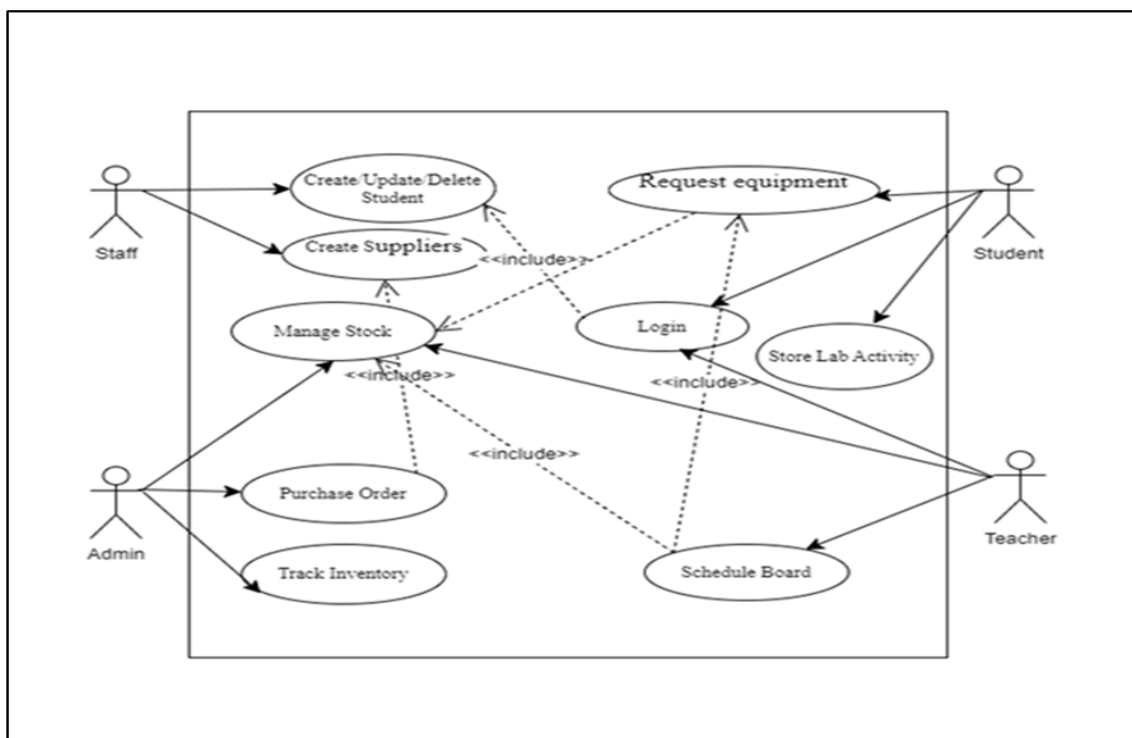


FIGURE 1. System Analysis User Case Diagram (7).

The use case figure 1 graphic shows how staff members may add, delete, and update objects. Administrators are responsible for issuing purchase orders, maintaining inventory, and managing stock; they depend on staff to generate suppliers. Here, "supplier" refers to the equipment the laboratory procures. When the laboratory purchases products or equipment from suppliers, it typically involves issuing purchase orders to formalize the transaction. Once the purchase orders are sent to suppliers, they fulfill the orders by delivering the equipment to the laboratory. Students use staff-created logins to access the system and request equipment depending on stock availability. Similarly, teachers use credentials given by staff to organize activities according to de-

mands and supply availability. An understandable overview of RLMS features and user interactions is presented in figure 1.

Using user-centric design and automation, the system analyst may make the Robotics Lab Management System (RLMS) more efficient. The logical structure, consistency, and simplicity of design are the cornerstones of every RLMS module, guaranteeing maximum use and functionality (8). Staff, administrators, students, and teachers have different responsibilities and interactions within the system, ranging from activity organizing to equipment management, as shown in the use case graphic. This thorough overview directs the design process, making it easier to create an RLMS that is user-friendly and efficient and that maximizes lab operations while empowering users.

3.1 User Management Module

This module focuses on managing user data in the laboratory setting. Administrators can add, remove, edit, and query user data. However, once logged in, students' access is restricted mainly to reading designated data and completing prescribed data input activities.

3.2 Inventory Management Module

This module tracks lab equipment. Administrators can add, edit, remove, and search equipment data. It will also facilitate supplier management, purchase order creation, and inventory control. Low-stock alerts and reporting features can also be integrated into this module. Robotics labs manage a unique inventory of equipment and supplies. Businesses use techniques like EOQ (ordering the right amount) and safety stock (having extra supplies) to avoid delays. Unlike factories, however, they may not use Just-in-Time methods because specialized equipment needs to be readily available for research. Analyzing equipment usage data helps them predict future needs and keep research running smoothly (9).

Figure 1 depicts the user experience within the Robotics Lab Management System (RLMS). Administrators establish user accounts and define access privileges. Staff and teachers primarily interact with the Inventory Management module, where they manage equipment data, identify shortages through low-stock alerts and quantity checks, initiate purchase orders, and analyze

usage patterns. Students utilize the Student Portal to view lab schedules, book available slots for research, and submit their work upon completion. The flowchart highlights the LMS's role-based functionalities and decision points, illustrating how user actions navigate the system for efficient lab operations.

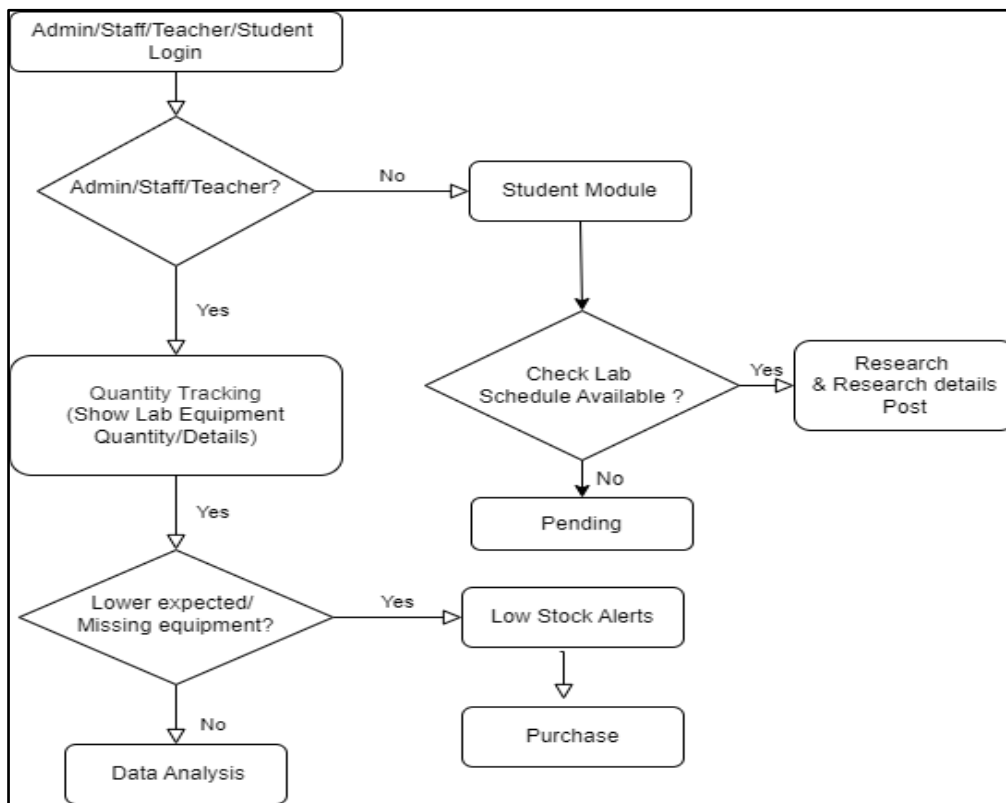


FIGURE 2. User Management with Inventory Management (10).

The Inventory Management module becomes a central hub for optimizing resource allocation and research continuity in the robotics lab. The figure 2, visually reinforces this concept, demonstrating how administrators, staff, and teachers seamlessly interact with the system to manage equipment, identify and address stock issues, and ensure research projects have the resources to thrive. This user-centric design empowers lab personnel to focus on their core tasks while the RLMS handles the administrative burden, fostering a more efficient and streamlined research environment.

3.3 Lab Resource Management Module

This module aims to expedite the permissions-granting procedure for staff and make it easier for students to register for equipment use. Students can keep papers related to their studies or lab

activity and establish lab profiles using them. The module also makes it possible to track resource availability in real-time.

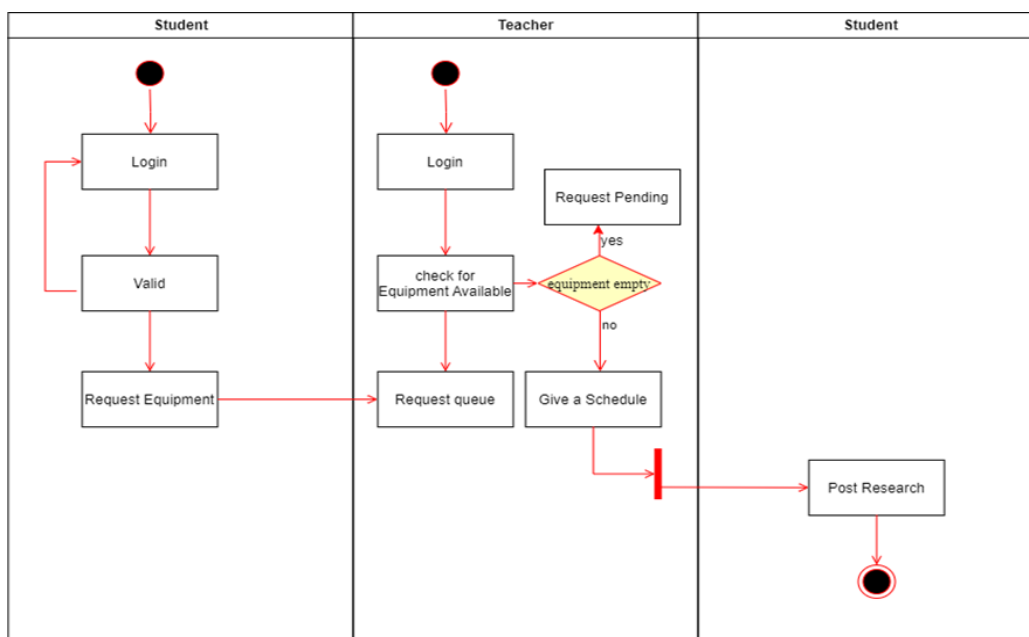


FIGURE 3. Lab Module System (11).

The sequence diagram depicted in figure 3, illustrates how teachers, students, and the system interact. While teachers carry out a comparable login process and verify that equipment is available, students start the login process and make equipment requests. Teachers start research projects and share their findings on the site if the necessary equipment is accessible. This succinct graphic illustrates the smooth progression of events, from equipment distribution and research start-up to login authentication, underscoring the effectiveness of RLMS in supporting lab activities.

3.4 Communication and Collaboration Module

This section focuses on improving teamwork and communication in the lab. It entails creating a scheduling module for lab equipment and resource reservations, an announcement board for exchanging important information, and, if desired, incorporating a message system to help staff and students communicate.

4 TECHNICAL INTRODUCTION

This project uses modern web technologies to create a reliable and accessible robotics lab management system (RLMS). The system uses a clean architectural methodology to guarantee the code's testability, scalability, and maintainability. Architects bridge communication gaps by acting as boundary objects and keeping projects adaptable to evolving technologies and customer requirements. Their efforts are essential for the successful design and implementation of large-scale systems. (12.)

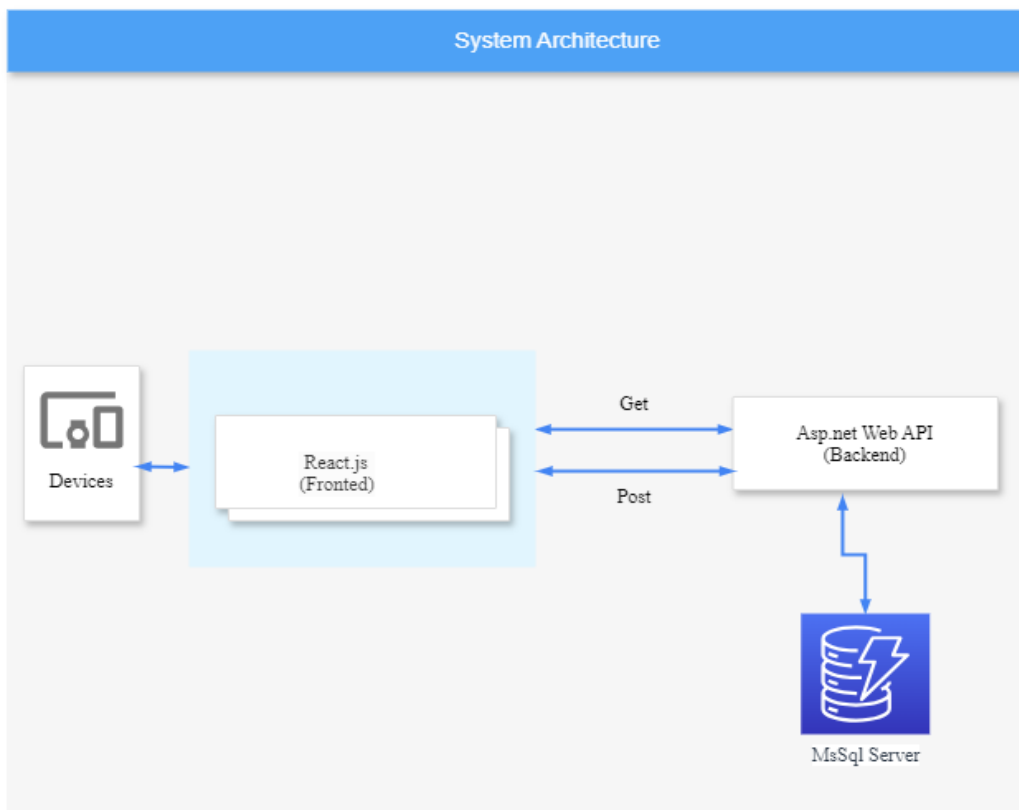


FIGURE 4. System Architecture (13).

The System Architecture figure 4, depicts a software architecture where a React.js frontend communicates with an ASP.NET API, which interacts with a Microsoft SQL Server (MSSQL) database. Users can access the system from various devices, such as mobile phones, tablets, laptops, and desktop computers, thanks to the responsive design of the React.js frontend. React.js serves as the frontend framework, offering dynamic and interactive user interfaces. Complementing this, the ASP.NET API orchestrates backend functionalities, enabling seamless communication between the frontend and the underlying Microsoft SQL Server (MSSQL) database.

4.1 Backend Development

Backend refers to the server side of a web application, where the core logic and data management processes reside. It bridges the frontend user interface and the database, handling requests, processing data, and generating responses. The backend is essential for executing tasks that cannot be performed on the client side, such as interacting with databases, implementing business logic, and managing user authentication.

4.1.1 ASP.NET Web API

ASP.NET Web API is a framework for building HTTP services that can reach many clients, including browsers and mobile devices. It allows developers to create RESTful APIs, enabling efficient communication between the frontend and backend layers of an application. With ASP.NET Web API, developers can implement endpoints to handle HTTP requests, define routes, and serialize and deserialize data in various formats like JSON or XML (12). Several libraries were used in the creation of the ASP.NET Web API project.

1. **Entity Framework:** Entity Framework is utilized for data access and persistence within the ASP.NET Web API project. This Object-Relational Mapping (ORM) framework simplifies interactions with the underlying database by abstracting database operations into object-oriented code (14). With Entity Framework, developers can work with domain-specific objects and queries rather than directly interacting with database tables and SQL queries. This enhances productivity, code maintainability, and database portability across different providers.
2. **Identity Framework:** The Identity Framework manages authentication and authorization within the ASP.NET Web API application. This framework provides robust features for user authentication, including user registration, login, password management, and role-based access control. By integrating the Identity Framework, the API can ensure secure access to resources and enforce permissions based on user roles (15).
3. **Swagger:** Integration of Swagger automates the generation of interactive API documentation. This inclusion allows developers to explore conveniently and test API endpoints di-

rectly from the browser, thereby enhancing understanding and utilization of the API by both internal teams and external clients (16).

4. **Serilog:** Serilog is integrated to manage logging within the ASP.NET Web API application (17). Serilog configuration captures pertinent logs for monitoring, troubleshooting, and performance analysis. Leveraging Serilog's structured logging capabilities, crucial events, errors, and diagnostic information are logged in a format conducive to easy search and analysis (18).
5. **Automapper:** Automapper integration streamlines object mapping between Data Transfer Objects (DTOs) and domain models within the ASP.NET Web API project. Data transfer between different application layers is automated by defining mapping profiles, reducing redundant code, and enhancing code readability. Automapper also contributes to maintaining the separation of concerns and fostering cleaner code architecture (19).

4.1.2 Microsoft SQL Server (MSSQL)

MSSQL is a relational database management system (RDBMS) developed by Microsoft. It serves as the backend database solution for storing and managing application data, including user information, equipment details, booking records, and communication logs. MSSQL offers features such as ACID compliance, scalability, security, and robust query optimization capabilities, making it suitable for enterprise-level applications (20).

4.1.3 Clean Architecture Design

The idea of "clean architecture" in software design emphasizes the independence of components and the division of responsibilities inside an application. It advocates for a layered architectural approach where the core application logic (domain layer) is isolated from infrastructure concerns such as persistence (database access) and presentation (user interface) (21). By decoupling these layers, clean architecture promotes maintainability, testability, and flexibility, allowing developers to change one layer without affecting the others. This design principle enables developers to create scalable and maintainable applications that are easier to understand and evolve.

4.2 Frontend Development

The goal of front-end development is to create the user interface (UI) and user experience (UX) elements of a web application that users would directly interact with. It entails creating and executing the program's interface design, layout, and functionality.

4.2.1 React.js

React.js is a widely used JavaScript library for building interactive user interfaces (22). It utilizes a component-based architecture, encapsulating UI elements into reusable and modular components. React.js enables developers to manage the UI state, update the DOM (Document Object Model), and create dynamic and interactive web applications. Its declarative approach simplifies building complex UIs by breaking them into smaller, manageable components, enhancing code reusability, and improving maintainability. Additionally, React.js provides excellent performance optimizations, such as virtual DOM rendering, which results in faster and smoother user experiences. Overall, React.js is an ideal choice for frontend development in the RLMS, enabling the creation of a modern, responsive, and intuitive user interface. The frontend React project uses a specific library, as mentioned below.

1. **Chakra-UI:** A UI library for React applications that provides a set of accessible and customizable components to build user interfaces quickly. Chakra UI is a more robust, layout-focused library that also provides developers with UI components similar to those that Material UI provides but has a greater focus on the creation of flexible, composable, and scalable code (23). This React library prioritizes modern development practices. It leverages React hooks for component functionality, utilizes CSS-in-JS for streamlined styling, and boasts exceptional customizability for crafting unique themes. Furthermore, its comprehensive documentation and exclusive TypeScript implementation ensure a smooth development experience with type safety (24).
2. **Axios:** The project's dependency on a web API for all data retrieval and posting makes Axios an ideal choice. Axios, a popular promise-based HTTP client for React applications, simplifies network requests with its ease of use and comprehensive feature set. It offers advantages such as excellent browser support, automatic response parsing, and

built-in functionalities for handling common scenarios like authorization, request cancellation, error handling, and CSRF protection. These features make Axios a valuable tool for building robust and efficient React applications, ensuring smooth communication with the web API while maintaining code simplicity and reliability (25).

3. **Redux:** Redux is a well-liked JavaScript state management library. Intricate React apps with several state management requirements may find it beneficial, though more straightforward applications may become more sophisticated as a result. This post offers a detailed how-to for integrating Redux into a React application, which includes setting up a store, actions, reducers, and components. Given how much state our React application handles, Redux is an excellent option for effective state management (26).
4. **DOM:** React Router DOM is crucial for crafting dynamic single-page applications (SPAs) in React. It streamlines navigation between application views, pages, or components without requiring full page reloads. This feature enhances user experience by delivering faster and smoother interactions. React Router DOM organizes routing functionality into a hierarchical structure.
 - **Browser Router:** Serves as the parent component, managing all routing capabilities within the application.
 - **Routes:** Nested within Browser Router, Routes specify the navigation paths available in the SPA.
 - **Route components:** These components match the current URL to defined paths and render the corresponding component when a match occurs.
 - **Link components:** Generate user-friendly route navigation elements, facilitating seamless navigation between different views.

React Router DOM simplifies React SPA development by utilizing this centralized routing system, giving application users a more effective and convenient experience. (27.)

4.2.2 Benefits of this Technical Stack

The RLMS leverages a modern technology stack, promoting efficiency throughout the development lifecycle. ASP.NET Web API facilitates the creation of a robust and scalable RESTful API, guaranteeing effective communication between the backend services and the user interface. Microsoft SQL Server (MSSQL) provides a secure and reliable foundation for storing and managing all application data. Furthermore, the system adheres to clean architecture principles, separating concerns and promoting modularity. This approach fosters code maintainability, allowing more accessible updates and adaptations as the lab's needs evolve. Additionally, the user interface is built with React.js. This popular JavaScript library enables a responsive and interactive user experience for researchers and students and allows for efficient component-based development, further contributing to code maintainability (28; 29).

5 IMPLEMENTATION

Each module and feature of the RLMS was meticulously implemented to align with the specifications outlined during the system analysis and design phases. Below is a detailed description of how each module and feature was developed.

5.1 Front Page Layout

The Robotics Lab Management System (RLMS) front page is the platform's gateway, providing easy access to crucial functionalities and information. The layout is designed to feature a login/signup button, a slider showcasing essential updates, and quick access to the five main modules of the system. Additionally, body sections highlight University News, Latest Research, and Featured Content to engage users and provide valuable insights.

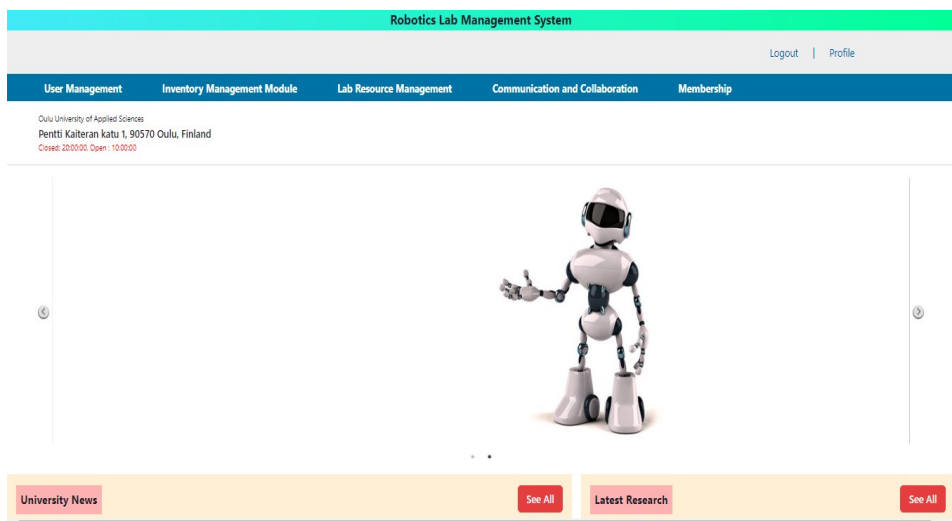


FIGURE 5. Front Page 1

On the front page, figure 5, presents the Robotics Lab Management System (RLMS), designed to offer users easy access to crucial functionalities and information. It prominently features module buttons for quick navigation to essential system modules, a login/signup button for user authentication, and a dynamic slider showcasing essential updates. This user-centric layout aims to engage users and provide valuable insights while facilitating efficient navigation within the RLMS platform.

5.1.1 Login/Signup Button

Description: A prominent button is displayed at the top right corner of the page. It allows users to log in to their accounts if they have existing credentials or sign up for a new account if they are new users.

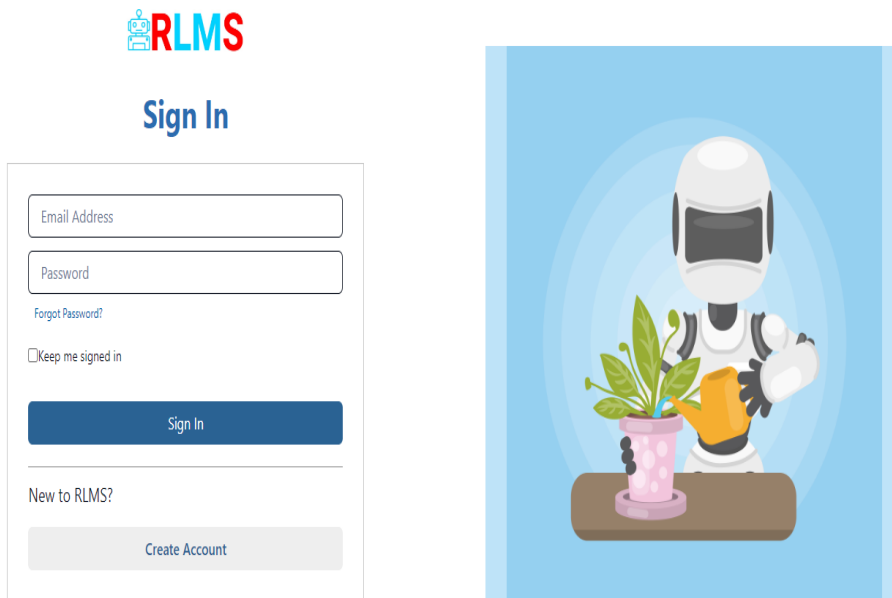


FIGURE 6. Login Page

On the login page, figure 6, serves as the point of entry for authorized users. It features slots for entering a username and password, providing a straightforward and user-friendly interface for secure login. Authentication and access control measures ensure the protection of private data and uphold system security.

Implementation: The visual design of the button uses React, and JavaScript handles the backend functionality to facilitate user authentication and registration processes.

5.1.2 Slider

Description: A dynamic slider positioned prominently on the front page, featuring slides that showcase essential updates, announcements, or events related to the robotics lab or university.

Implementation: The slider uses JavaScript for interactivity and transitions between slides. Content for the slider is dynamically fetched from the backend database to ensure timely updates.

5.1.3 Main Modules

Description: The RLMS's five main modules are prominently displayed on the front page, providing quick access to essential functionalities: Communication and Collaboration, Lab Resource Management, Inventory Management, and User Management.

Implementation: Each module is represented by a visually appealing card or icon, with clickability implemented using HTML/CSS and JavaScript to navigate users to the corresponding module pages within the RLMS.

5.1.4 University News

Description: A section dedicated to displaying the top two news articles or announcements from the university, highlighting important events, achievements, or updates relevant to the academic community.

Implementation: News articles are fetched from a designated data source, such as an RSS feed or a custom database table and displayed dynamically on the front page. Clickable links allow users to read the full articles.

5.1.5 Latest Research

Description: Another section showcases the top two latest research findings or publications from the robotics lab or related academic departments, providing insights into ongoing research activities and contributions to the field.

Implementation: Research findings are retrieved from a database or external data source and presented in a visually appealing format on the front page. Links to all research papers or articles are provided for users to explore further.

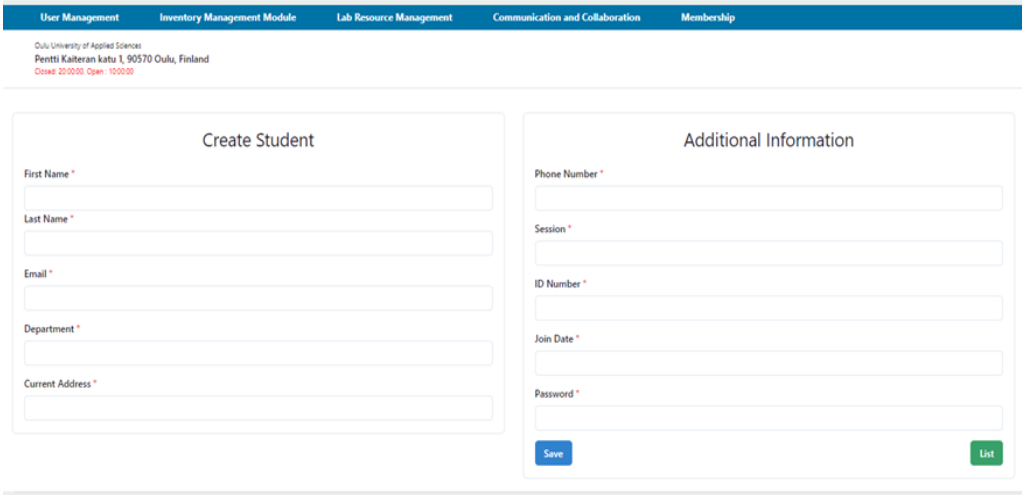
5.1.6 Featured Content

Description: A section featuring the top two pieces of featured content, such as blog posts and articles curated to engage users and provide valuable insights into robotics and related topics. This section is accessible to guests without requiring login.

Implementation: Featured content is selected and displayed dynamically on the front page, with content retrieved from a designated data source. Clickable links are provided for users to read or view the full content.

5.2 User Management Module

Description: The User Management Module enables administrators to perform essential tasks related to user data management. Administrators have enhanced privileges to add, remove, edit, and query user data. On the other hand, regular users, such as students, have restricted access upon logging in, primarily limited to viewing designated data and completing prescribed input activities.



The screenshot displays a web application interface for creating a student. At the top, a blue navigation bar contains the following menu items: 'User Management', 'Inventory Management Module', 'Lab Resource Management', 'Communication and Collaboration', and 'Membership'. Below the navigation bar, the application header identifies the user as 'Dulu University of Applied Sciences' and provides the address 'Penttili Kaizeran katu 1, 90570 Oulu, Finland' along with contact information 'Oulu: 020020, Open: 100020'. The main content area is divided into two columns. The left column, titled 'Create Student', contains input fields for 'First Name *', 'Last Name *', 'Email *', 'Department *', and 'Current Address *'. The right column, titled 'Additional Information', contains input fields for 'Phone Number *', 'Session *', 'ID Number *', 'Join Date *', and 'Password *'. At the bottom of the right column, there are two buttons: a blue 'Save' button and a green 'List' button.

FIGURE 7. Create Student

Figure 7, illustrates the interface where staff or administrative personnel can input all necessary student data. This includes fields for First Name, Last Name, Email, Department, Current Address, Additional Information, Phone Number, Session, ID Number, Join Date, and Password.

Implementation: The module used ASP.NET Identity Framework for user authentication and authorization. Customized views and controllers were developed for user registration, login, and profile management functionalities. Role-based access control (RBAC) was enforced to ensure users only have access to functionalities appropriate for their roles. The Student List Access Control feature empowers administrators with comprehensive control over student data within the Robotics Lab Management System (RLMS). Administrators can efficiently manage student information, including editing, terminating, or granting access rights. This functionality ensures strict adherence to data integrity and access policies, fostering a secure and well-maintained student database.

5.3 Inventory Management Module

Implementing the Lab Resource Management Module involves developing robust features and functionalities to manage lab resources effectively. Below are the key aspects of the implementation.

5.3.1 Equipment Management

Description: The module allows administrators to track equipment data, including adding, editing, removing, and searching for equipment records. This ensures that the inventory database remains comprehensive and up to date.

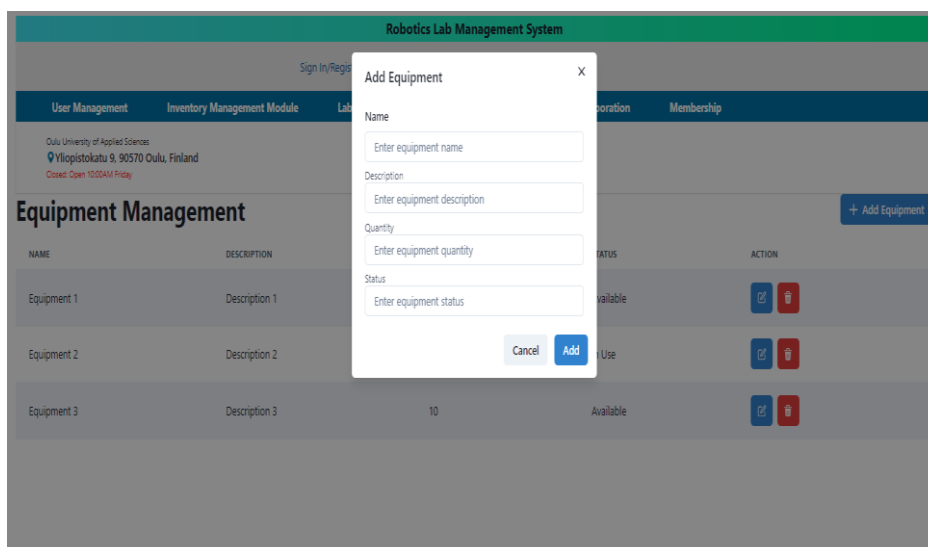
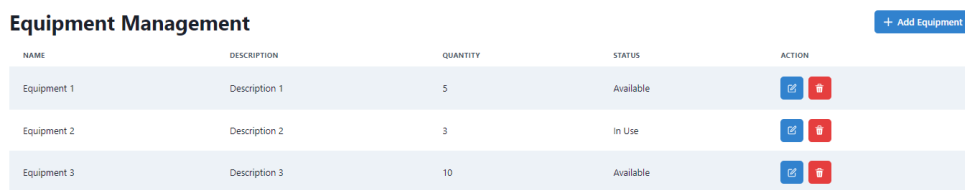


FIGURE 8. Equipment Management Add Module

Figure 8, showcases the Equipment Management Add Module, providing users a streamlined interface for adding new equipment to the system.

Implementation: User interfaces are developed to facilitate seamless interaction with equipment records, enabling administrators to perform CRUD (Create, Read, Update, Delete) operations efficiently. Backend APIs handle data processing and validation, ensuring data integrity and consistency.









| NAME | DESCRIPTION | QUANTITY | STATUS | ACTION |
|-------------|---------------|----------|-----------|---|
| Equipment 1 | Description 1 | 5 | Available |   |
| Equipment 2 | Description 2 | 3 | In Use |   |
| Equipment 3 | Description 3 | 10 | Available |   |

FIGURE 9. Equipment Management List

Figure 9, displays the Equipment Management List, showcasing the functionalities for updating or deleting equipment by administrators or staff members.

5.3.2 Purchase Order

Description: The module enables administrators to create purchase orders to acquire necessary equipment and supplies from suppliers. Purchase orders include item quantities, delivery dates, and supplier information.

Implementation: User interfaces are developed to facilitate purchase order creation, allowing administrators to specify order details and submit purchase requests. Backend processes handle order validation, submission, and tracking, ensuring timely resource procurement.

FIGURE 10. Create Equipment Page

Figure 10, illustrates the Create Equipment Page, where users can input details such as quantity, price, and instructions for usage upon successful purchase order and receipt of the equipment.

5.3.3 Low-Stock Alerts

Description: The module includes functionalities to generate alerts when equipment or supplies reach predefined low-stock thresholds, prompting administrators to take necessary actions such as initiating purchase orders or adjusting inventory levels.

Implementation: Backend processes monitor inventory levels and trigger notifications when stock levels fall below specified thresholds. Administrators receive alerts via email, SMS, or within the RLMS interface, enabling proactive management of low-stock situations.

Low-stock alerts and reporting features were integrated to facilitate proactive management of equipment stock.

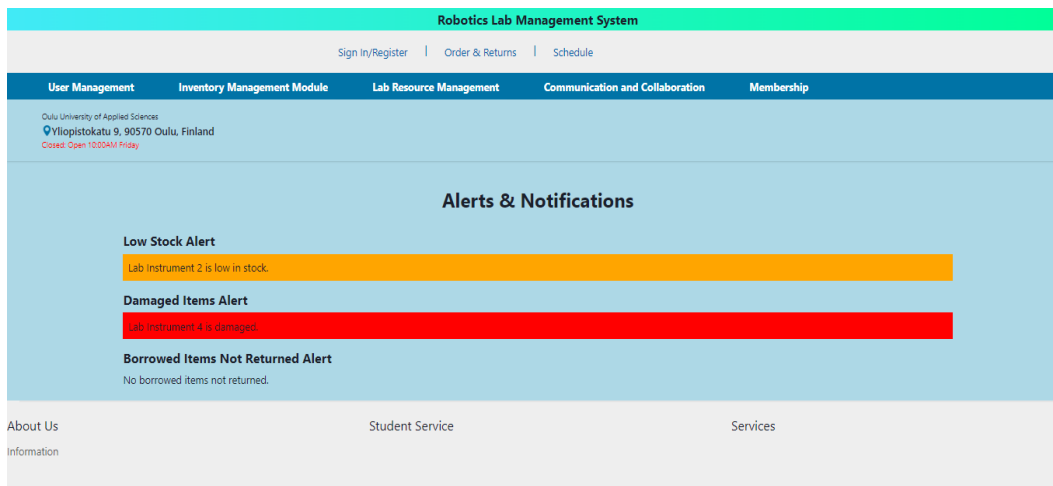


FIGURE 11. Low-stock alerts page

Figure 11, showcases the Low-stock Alerts Page module, which incorporates functionalities designed to generate alerts when equipment or supplies reach predefined low-stock thresholds. This prompts administrators to take necessary actions, such as initiating purchase orders or adjusting inventory levels.

5.3.4 Inventory Control

Description: Inventory control functionalities allow administrators to monitor stock levels and manage inventory effectively. This includes tracking the quantity of items in stock, updating inventory records, and facilitating inventory adjustments.

Implementation: User interfaces provide access to inventory management features, allowing administrators to view real-time inventory data, perform stock updates, and manage inventory levels. Backend processes handle inventory tracking, stock adjustments, and notifications for low-stock levels.

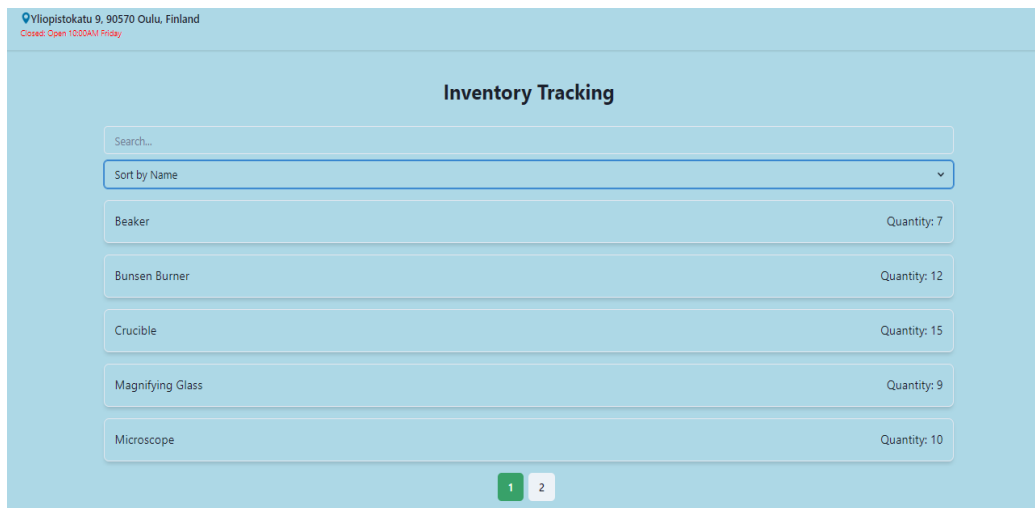


FIGURE 12. Inventory Tracking Page

Figure 12, presents the Inventory Tracking Page, featuring inventory control functionalities empowering administrators to monitor stock levels and manage inventory effectively.

5.4 Lab Resource Module

The Lab Resource module is crucial in managing equipment requests, approvals, and allocations within the robotics lab environment. It consists of several key components to streamline resource management and enhance collaboration among staff and students.

5.4.1 Equipment Request System

Description: The equipment request system allows students to request specific equipment required for their studies or lab activities. It includes a user-friendly form where students can upload their research findings, providing detailed information such as the topic, result, and description of the research.

Implementation: Implemented as a user interface component within the RLMS frontend, the equipment request system facilitates seamless interaction between students and the system. The form allows students to input relevant details and submit their requests, which are processed by the backend for further review and approval.

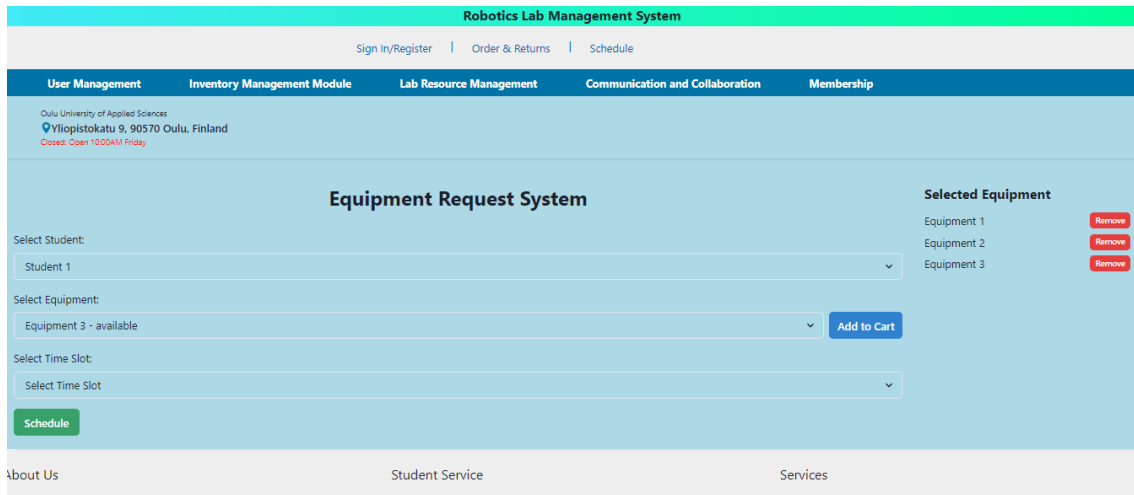


FIGURE 13. Equipment Request System

Figure 13, introduces the Equipment Request System, designed to enable students to request specific equipment for their studies or lab activities.

5.4.2 Resource Allocation

Description: This component enables teachers to allocate equipment to specific projects based on their requirements. Teachers can assign equipment resources strategically to ensure that projects have access to the necessary tools and resources for successful completion.

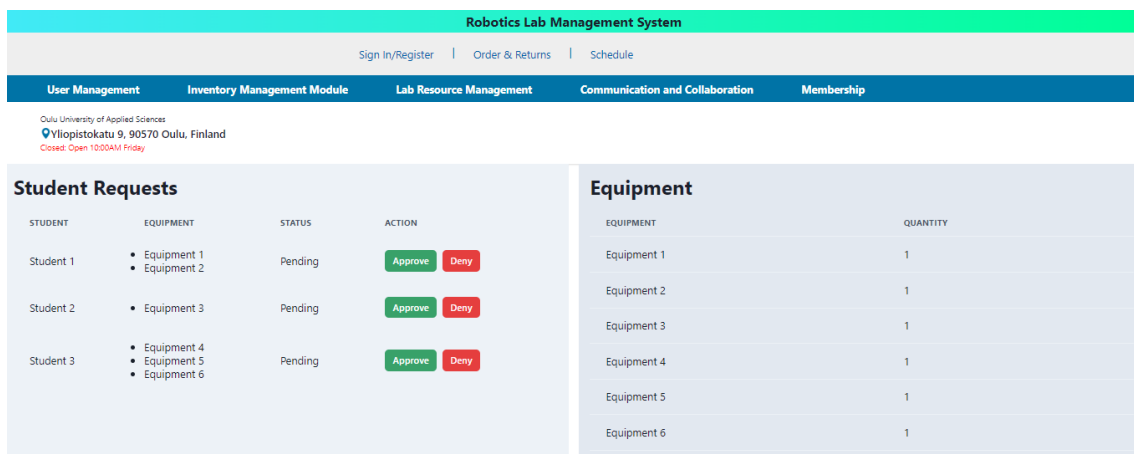


FIGURE 14. Resource Allocation

Figure 14, introduces the Resource Allocation component, empowering teachers to allocate equipment to specific projects based on their unique requirements.

Implementation: Integrated into the RLMS backend, the resource allocation feature provides teachers with tools to manage equipment assignments efficiently. Teachers can view project details, assess resource availability, and allocate equipment accordingly, ensuring optimal utilization and project success.

5.4.3 Result Sharing

Description: The Result Sharing component allows students to share their research findings on the platform. It provides input fields for the topic, result, and description of the research. This component proves particularly valuable for projects requiring long-term, day-to-day data collection, such as student long-term projects. In such scenarios, the Result Sharing component becomes instrumental for efficient data storage and sharing.

The screenshot shows the 'Robotics Lab Management System' interface. At the top, there is a navigation bar with links for 'Sign In/Register', 'Order & Returns', and 'Schedule'. Below this is a menu with 'User Management', 'Inventory Management Module', 'Lab Resource Management', 'Communication and Collaboration', and 'Membership'. The main content area is titled 'Upload Research Findings' and contains three input fields: 'Topic' (with placeholder 'Enter topic'), 'Result' (with placeholder 'Enter result'), and 'Description' (with placeholder 'Enter description'). Below these fields is a green 'Upload' button. At the bottom of the page, there is a footer with 'About Us', 'Student Service', and 'Services'.

FIGURE 15. Result Sharing

Figure 15, depicts the Result Sharing component, a critical functionality on the platform that facilitates the dissemination of student research findings. The "Upload" button is disabled until all required fields are filled, ensuring that users provide the necessary information before uploading their research findings.

Implementation: The component utilizes React state hooks to manage the input values for the topic, result, and description. The handle upload function is triggered when the user clicks the "Upload" button. It logs the uploaded data to the console and clears the input fields afterward. The button is disabled if any required fields are empty, ensuring data completeness before submission.

5.5 Communication and Collaboration Module

Description: The Communication and Collaboration Module focuses on improving teamwork and communication within the lab. It includes features such as a scheduling module for lab equipment and resource reservations, an announcement board for sharing important information, and a messaging system for staff and students to communicate.



The screenshot displays the 'Robotics Lab Management System' interface. At the top, there is a navigation bar with links for 'Sign In/Register', 'Order & Returns', and 'Schedule'. Below this is a menu with categories: 'User Management', 'Inventory Management Module', 'Lab Resource Management', 'Communication and Collaboration', and 'Membership'. The main content area is titled 'Scheduling Module' and contains four dropdown menus for selection: 'Select Student', 'Select Equipment', 'Select Teacher', and 'Select Time Slot'. A green 'Schedule' button is positioned at the bottom of the form.

FIGURE 16. Scheduling Module

Figure 16, showcases a Scheduling Module designed to enhance teamwork and communication within the lab environment. This module offers several key features to streamline lab operations and facilitate collaboration.

Implementation: This module included developing functionalities for creating and managing lab schedules, posting announcements, and facilitating communication between lab members. Customized views and controllers were designed to provide an intuitive and user-friendly interface.

Integration with real-time messaging services or email systems may be utilized for communication features.

5.6 Backend

The Swagger UI provides an interactive interface for exploring and testing the Robotics Lab Management System API Server's endpoints. It allows users to visualize the API's structure, view detailed documentation for each endpoint, and execute requests to interact with the server in real time.

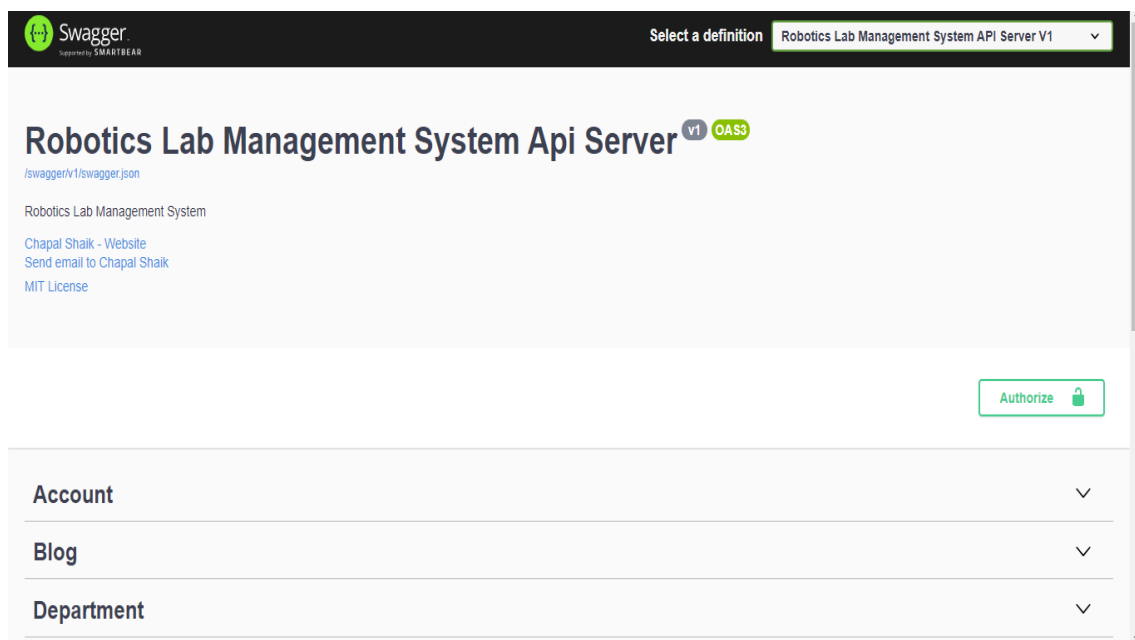


FIGURE 17. Swaggar UI

Figure 17, displays a list of controllers that provide Swagger UI, which offers several API development and documentation benefits. Swagger, now known as the Open API Specification, is a powerful tool for describing, producing, consuming, and visualizing RESTful APIs.

The Swagger UI offers a user-friendly experience that presents the API endpoints in a hierarchical manner, organized by resource categories such as Equipment, Purchase Order, Research, Student, and University. Each endpoint is accompanied by detailed descriptions of its purpose, parameters, request/response schemas, and possible status codes. Additionally, the Swagger UI provides input fields for specifying request parameters and headers, making it easy for users to customize and execute API requests.

Description: The Robotics Lab Management System API Server (RLMS API Server) serves as the backend infrastructure for the Robotics Lab Management System. It provides a set of endpoints and functionalities to facilitate communication between the frontend user interface and the underlying database. The API server follows the Open API Specification (OAS3) standard, offering a standardized interface for accessing resources and performing operations within the system. It handles various aspects of lab management, including equipment management, purchase orders, research data, student information, and more. Additionally, the API server integrates with authentication mechanisms to ensure secure access to protected resources.

Implementation: The RLMS API Server uses modern web technologies, including Asp.net, to build the server-side logic and handle HTTP requests. The server is designed to follow RESTful principles, with each endpoint corresponding to a specific resource or action within the system. Data persistence is achieved through integration with a relational database management system (e.g., MS SQL) using an ORM (Entity Framework) library such as Sequelize. The API server utilizes middleware for request parsing, authentication, and error handling to enhance reliability and security.

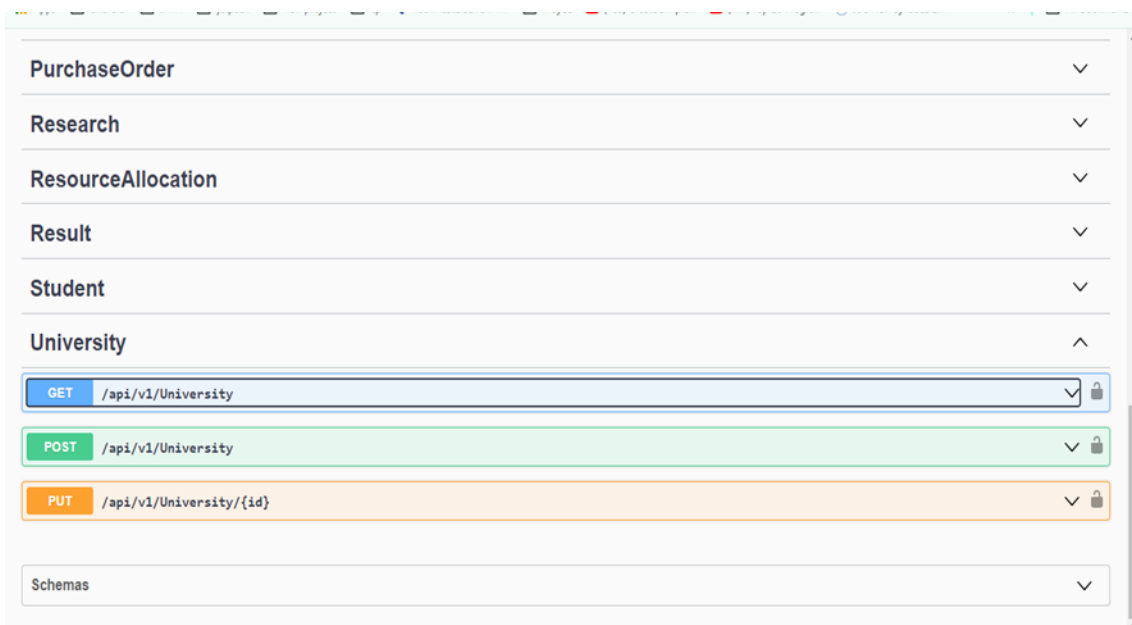


FIGURE 18. Controller List

Figure 18, displays a list of controllers providing Swagger UI, showcasing all the controllers the backend manages.

Endpoints are organized into logical categories such as Equipment, Purchase Order, Research, Student, and University, each corresponding to a specific module or feature of the RLMS. These endpoints accept various HTTP methods (GET, POST, PUT, DELETE) to perform CRUD (Create, Read, Update, Delete) operations on corresponding resources. Additionally, the API server incorporates authorization mechanisms to restrict access to specific endpoints based on user roles and permissions. Documentation for the API endpoints is provided as an Open API Specification (OAS3) file, accessible at the `/swagger/v1/swagger.json` endpoint, enabling developers to explore and understand the available functionalities. Overall, the RLMS API Server is a robust backend infrastructure, enabling seamless communication and data management within the Robotics Lab Management System.

6 TESTING

Testing is an integral component of the development lifecycle for the Robotics Lab Management System (RLMS). These phases ensure that the system meets its objectives, adheres to quality standards, and delivers a seamless user experience. This chapter thoroughly summarizes the procedures, standards, and outcomes used in the RLMS project's testing.

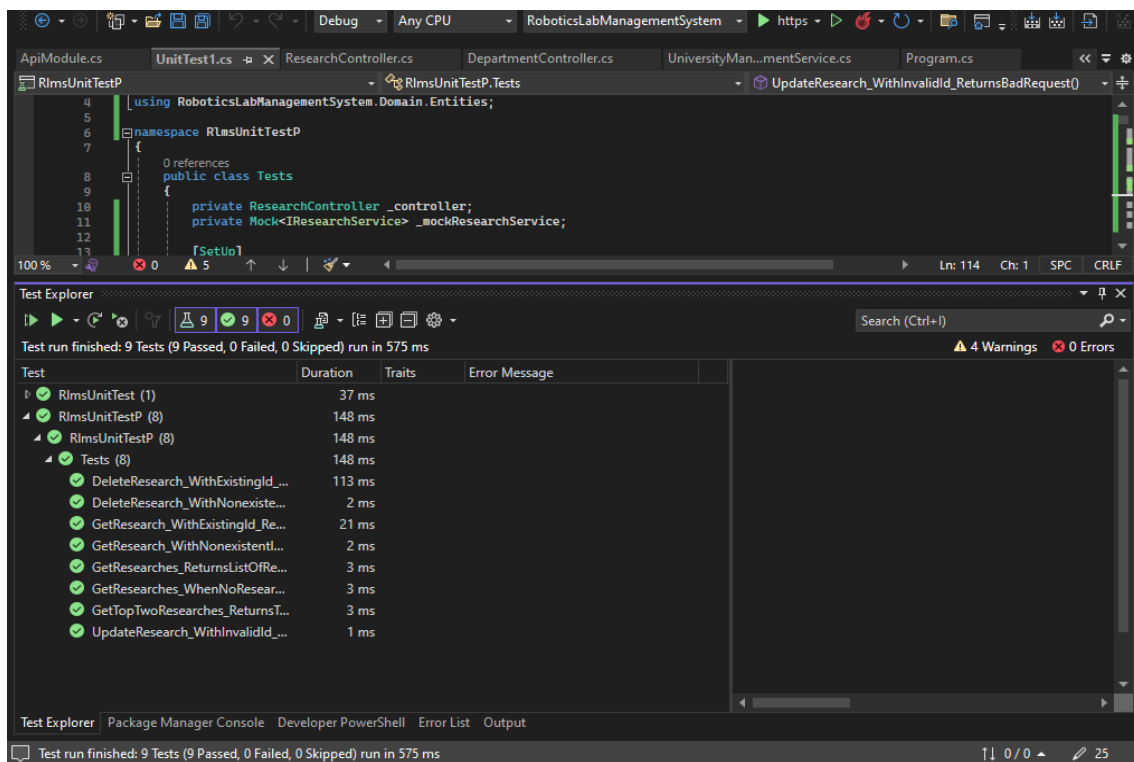


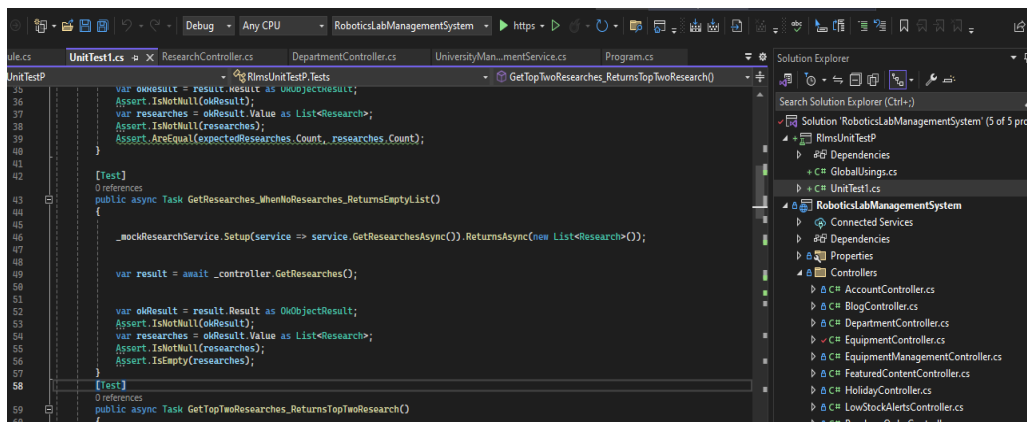
FIGURE 19. Research Controller Unit Test Result

Figure 19, contains unit test results for the ResearchController in the Robotics Lab Management System project. These tests are written using the NUnit framework and Moq library to mock the iResearchService dependency. The Setup method initializes the mock service and creates an instance of the ResearchController, ensuring that each test starts with a clean state. Each test method focuses on a specific action of the ResearchController and follows an Arrange-Act-Assert pattern. The Arrange section sets up the necessary dependencies and defines the expected behavior of the mocked service. By running these tests, the correctness of the ResearchController is verified, and ensuring the code behaves as expected under various conditions promotes reliability, early issue detection, and overall software quality.

6.1 Testing Methodologies

This chapter provides a comprehensive overview of the methods and procedures used in the testing RLMS project.

Unit Testing: Unit testing involves the verification of individual components or units of code in isolation. By isolating each unit, developers could assess its functionality independently, ensuring that it performed as expected and identifying any defects or errors early in the development process (30).



```
UnitTestP
36 var okResult = result.Result as OkObjectResult;
37 Assert.IsNotNull(okResult);
38 var researches = okResult.Value as List<Research>;
39 Assert.IsNotNull(researches);
40 Assert.AreEqual(expectedResearches.Count, researches.Count);
41 }
42
43 [Test]
44 References
45 public async Task GetResearches_WhenNoResearches_ReturnsEmptyList()
46 {
47     _mockResearchService.Setup(service => service.GetResearchesAsync()).ReturnsAsync(new List<Research>());
48
49     var result = await _controller.GetResearches();
50
51     var okResult = result.Result as OkObjectResult;
52     Assert.IsNotNull(okResult);
53     var researches = okResult.Value as List<Research>;
54     Assert.IsNotNull(researches);
55     Assert.IsEmpty(researches);
56 }
57
58 [Test]
59 References
60 public async Task GetTopTwoResearches_ReturnsTopTwoResearch()
```

FIGURE 20. Unit Test Code

Figure 20, presents a unit test that verifies the behavior of the GetResearches method within the controller. This test ensures that when no research data is available, the process returns an empty list. It accomplishes this by configuring a mock research service to return an empty list of Research objects. Upon invoking the controller's GetResearches method, the test examines the result to confirm it is an OkObjectResult. Subsequently, it verifies that the value of the result, a non-null list, is indeed empty. This rigorous testing guarantees that the controller adeptly handles scenarios without research entries to return.

This test verifies that the controller method behaves correctly when no research is available without involving the actual implementation of the IResearchService. It focuses solely on the behavior of the GetResearches method in isolation, making it a unit test.

Unit testing plays a crucial role in software development by allowing developers to verify the correctness of individual units or components of a software system. By isolating code sections and

testing them in isolation, developers can identify and fix bugs early in the development cycle, saving time and costs associated with later-stage testing. Additionally, unit testing facilitates code reuse, helps developers understand the code base, and enables them to make changes quickly (31).

Integration Testing: Integration testing ensures that it focuses on small pieces of code because it examines how multiple parts of the system collaborate to achieve specific tasks. Integration tests examine everything that relies on, like databases and networks, and how it handles requests and responses. Integration tests provide a higher level of confidence in the correctness and functionality of the entire system, as they test the interactions between components in a more realistic environment.

Unit tests verify the correctness of individual units or components in isolation and integration tests validate the collaboration and compatibility of different system elements when integrated. The System Under Test (SUT) is a common term used in integration testing discussions to refer to the tested project, usually an ASP.NET Core app (32).

System Testing: System testing is a type of software testing in which the entire integrated software product is assessed to ensure it satisfies the requirements and performs as intended in the intended setting. In system testing, the entire software application, including all its components, modules, and interfaces, is tested to verify that they work together seamlessly and fulfill the desired functionality.

The primary objective of system testing is to validate the software system's behavior against its requirements, functional specifications, and user expectations. This testing phase typically occurs after integration testing and before acceptance testing. System testing is conducted in an environment that resembles the production environment to simulate real-world usage conditions as closely as possible.

7 SECURITY

Security is critical to ensuring the integrity and confidentiality of data and resources within the Robotics Lab Management System (RLMS). This section outlines the security measures, vulnerabilities, and recommendations for enhancing the RLMS's security posture, mainly focusing on web API security.

7.1 Security Measures

Robust security measures are implemented to safeguard against potential threats.

Encryption Protocols: Data encryption techniques protect sensitive information from unauthorized access. Encryption technology is utilized to store passwords, ensuring the privacy of users information (33).

Access Controls: Role-Based Access Control (RBAC) mechanisms restrict user access based on predefined roles and permissions, ensuring that users only have access to relevant resources and functionalities. The Identity framework technology is utilized to manage authentication for different user roles, including Admins, staff, and teachers.

Authentication Mechanisms: Token-based authentication using JSON Web Tokens (JWT) ensures secure user authentication and authorization (34).

Cross-Origin Resource Sharing (CORS): CORS policies are implemented to thwart unauthorized access to APIs from other domains. This restricts the domains that can access our RLMS APIs, reducing the risk of unauthorized data access and manipulation (35).

```
93
94
95 builder.Services.AddCors(options =>
96 {
97     options.AddPolicy("AllowSites",
98         builder =>
99         {
100             builder.WithOrigins("http://localhost:4200", "https://localhost:7307", "https://localhost:7070", "http://localhost:5173")
101                 .AllowAnyMethod()
102                 .AllowAnyHeader();
103         });
104     });
105
```

FIGURE 21. Implement CORS

Figure 21, provides further configuration for CORS in the ASP.NET Core application.

Rate limiting is implemented using ASP.NET Core rate limiting middleware to limit the number of calls to API endpoints within a given time frame. This helps protect against brute-force attacks and ensures the stability and availability of the API. This helps protect against brute-force attacks and ensures the stability and availability of our API.

7.2 Vulnerability Assessment

Despite robust security measures, the RLMS may be susceptible to various vulnerabilities.

SQL Injection: Improperly sanitized user inputs may lead to SQL injection attacks, compromising database integrity. Proper input validation and parameterized queries are essential to mitigate this risk (36).

Cross-Site Scripting (XSS): Vulnerabilities in web application inputs may allow malicious scripts to be injected, potentially compromising user data (37). Input validation and output encoding techniques are employed to prevent XSS attacks.

8 DEPLOYMENT

An application must be deployed for users to use its features and functions and be available. In addition to accessibility, deployment allows scalability to meet growing user demand, testing in real-world settings, and the application of security controls to safeguard confidential information. It also makes continuous integration deployment and performance optimization easier, guaranteeing the smooth delivery of updates and new features. Ultimately, deployment improves user happiness by giving consumers a dependable and effective platform to engage with the program, enhancing their entire experience.

This chapter delves into deploying ASP.NET Core applications to Internet Information Services (IIS). ASP.NET Core hosting differs significantly from traditional ASP.NET hosting, necessitating a nuanced understanding for successful deployment. IIS, as a web server operating within the Windows OS, plays a pivotal role in hosting applications built on ASP.NET Core. The steps in configuring ASP.NET Core applications for IIS hosting will be explored, and the advantages of utilizing IIS for ASP.NET Core hosting will be discussed. Deploying an ASP.NET Core app to IIS may appear complex initially, but it becomes a manageable task with proper guidance. Deploying ASP.NET Core applications to Internet Information Services (IIS) requires careful consideration and correct configuration. By understanding the nuances of ASP.NET Core hosting and configuring the project accordingly, the application can be successfully deployed to an IIS server. Leveraging the capabilities of IIS offers numerous advantages, including increased performance, scalability, and management capabilities.

8.1 Configuring ASP.NET Core Project

Before proceeding with deployment, ensuring that our ASP.NET Core project is configured correctly is imperative. This involves evaluating the server environment to ascertain its compatibility and adequacy for hosting the application. Considerations include hardware specifications, software dependencies, and operating system compatibility. Additionally, compiling the developed application into a deployable format is crucial. This encompasses packaging all necessary files, libraries, and configurations for execution into a coherent and portable unit.

8.2 Deployment Steps

Project Publishing: Visual Studio is used to publish the application. This process generates optimized files for deployment.

IIS Configuration:

- **Application Pool Creation:** Create a dedicated application pool in IIS Manager for the specified application. This isolates it from other applications, improving security and resource management.
- **Website Setup:** Create a new website or virtual directory pointing to the published application folder. Configure bindings (domain name/IP address) and port assignments for user access.
- **UseIISIntegration():** In Program.cs file, configure the WebHostBuilder to use IIS Integration with the UseIISIntegration() method. This enables seamless communication between the application and IIS.

Utilizing Internet Information Services (IIS) for hosting applications, particularly ASP.NET Core, offers several advantages. With optimized performance, scalable architecture, robust security features, streamlined management tools, seamless integration with the Windows ecosystem, and dedicated support for ASP.NET Core, IIS provides a reliable and efficient platform for deploying web applications. Its comprehensive feature set ensures high availability, performance, and security, enabling organizations to deliver seamless user experiences while maximizing productivity and minimizing operational complexities (38).

9 CONCLUSION

Automation in the lab management system is essential for reducing errors and decreasing the time required to manage lab operations. Additionally, it simplifies the administrators' tasks, enhancing overall efficiency and effectiveness in laboratory management.

The thesis was inspired by OAMK's need to create a web-based Robotics Lab Management System (RLMS). It was essential to move away from outdated lab management methods. Management tasks were done using Excel spreadsheets and files stored in the cloud or the administrator's computer. Completing the Robotics Lab Management System (RLMS) represents a significant milestone in advancing robotics labs, offering an innovative solution to address various operational challenges.

Robotics Lab Management System transforms equipment management by providing real-time tracking capabilities that improve resource allocation and support well-informed decision-making. Using centralized monitoring, lab administrators can make data-driven choices that optimize efficiency and resource use, providing crucial insights into equipment availability and consumption trends.

The RLMS's robust user control and equipment borrowing features are vital components. These features encourage responsible use and strengthen lab security. The solution ensures smooth communication and protects lab resources from misuse or illegal access by enforcing strict access rules and simplifying request procedures.

Moreover, the RLMS gives managers practical insights from extensive data collection on equipment use and upkeep. Utilizing this abundance of data, lab managers may optimize lab operations and boost efficiency by making well-informed decisions about resource allocation, future acquisitions, and strategic planning.

REFERENCES

1. Johnson, Anna 2024. Reasons to study Robotics. Search date 29.5.2024. <https://www.robotlab.com/blog/reasons-to-study-robotics>
2. BO, Gao 2009. Design and implement multimedia equipment management system based on .Net and three-tier architecture. Journal of Chongqing Institute of Technology. Search date 3.5.2024.
3. BOYAR, Kyle, PHAM, Andrew, SWANTEK, Shannon, WARD, Gary and HERMAN, Gary 2021. Laboratory Information Management Systems (LIMS). In: Springer eBooks. Search date 3.5.2024.
4. LI, Dongqi, LIU, Peng, HUANG, Guotai, MA, Yuan, XIE, Zheyu, LI, Yunfeng and HUANG, Xin 2021. Design and application of intelligent equipment management platform. Search date 3.5.2024. <https://doi.org/10.1088/1742-6596/1983/1/012098>
5. Grinter, Systems Architecture: Product Designing and Social Engineering, Bell Labs, Lucent Technologies. Search date 3.5.2024.
6. Waldo, Jim 2006. On System Design, Sun Microsystems Laboratories, of Sun Microsystems, Inc., pp. 1-5. Search date 3.5.2024. <https://scholar.harvard.edu/files/waldo/files/ps-2006-6.pdf>
7. Creately, 2022. Use Case Diagram Tutorial (Guide with Examples). Search date 3.5.2024. <https://creately.com/guides/use-case-diagram-tutorial/>
8. Zheng, Boli 2017. Design and Implement a Laboratory Management System based on Web ResearchGate. Search date 3.5.2024. https://www.researchgate.net/publication/312865492_Design_and_Implement_of_Laboratory_Management_System_based_Web
9. Snow, John. Laboratory Logistics Handbook. Search date 3.5.2024. https://pdf.usaid.gov/pdf_docs/Pnadb082.pdf

10. Creately,2022. Ultimate Flowchart Tutorial | Learn What a Flowchart is and How to Create a Flowchart. Search date 10.5.2024. <https://creately.com/guides/flowchart-guide-flowchart-tutorial/>
11. Creately,2022. Sequence Diagram Tuto. Search date 10.5.2024. <https://creately.com/guides/sequence-diagram-tutorial/>
12. Wadepickett,2022. Create a web API with ASP.NET Core. Search date 11.5.2024. <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio>
13. InfoQ, 2020. A Seven-Step Guide to API-First Integration. Search date 11.5.2024. <https://www.infoq.com/articles/api-first-integration/>
14. Ajcvickers,2021. Overview of Entity Framework Core - EF Core. Search date 11.5.2024. <https://learn.microsoft.com/en-us/ef/core/>
15. Anderson, Rick 2024. Introduction to identity on ASP.NET core. Search date 12.5.2024. <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio>
16. Zuckerthoben,2024. Get started with Swashbuckle and ASP.NET Core. Search date 13.5.2024. <https://learn.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-8.0&tabs=visual-studio>
17. Serilog,2022. Serilog — simple .NET logging with fully structured events. Search date 13.5.2024. <https://serilog.net/>
18. Centeio, Brucy 2023. Adding SeriLog to ASP.NET Core .NET 7 & 8. Search date 14.5.2024. <https://medium.com/@brucycenteio/adding-serilog-to-asp-net-core-net-7-8-5cba1d0dea2>
19. Chinta, Merwan 2024. Auto Mapper in C# .NET Core with Unit tests. Search date 14.5.2024. <https://medium.com/codenx/automapper-in-net-core-778f9c874164>

20. SQL Server tutorial,2021. SQL Server tutorial – The practical SQL Server tutorial. Search date 15.5.2024. <https://www.sqlservertutorial.net/>
21. Celep, Beyza 2022. Sequence. Introduction to Clean Architecture. date 15.5.2024. <https://celepbeyza.medium.com/introduction-to-clean-architecture-acf25ffe0310>
22. React,2023. Getting started – react. Search date 15.5.2024. <https://legacy.reactjs.org/docs/getting-started.html>
23. Chakra Ui,2022. Chakra UI - A simple, modular and accessible component library that gives you the building blocks you need to build your React applications. Search date 3.5.2024. <https://v2.chakra-ui.com/>
24. Yotam, Bloom 2023. Why we chose Chakra-UI for our design system. Search date 16.5.2024. <https://medium.com/meliopayments/why-we-chose-chakra-ui-for-our-design-system-part-1-a9f988127dab>
25. Da, Naveen 2022. A detailed guide to using Axios in your React App - JavaScript in plain English. Search date 16.5.2024. <https://javascript.plainenglish.io/a-detailed-guide-to-using-axios-in-your-react-app-7396f79fb4c2>
26. Thenuka, Thisura 2022. Add Redux to your React app in 6 Simple Steps. Search date 3.5.2024. <https://dev.to/thisurathenuka/add-redux-to-your-react-app-in-6-simple-steps-43bb>
27. GeeksforGeeks,2024. What is react-router-dom? Search date 3.5.2024. <https://www.geeksforgeeks.org/what-is-react-router-dom/>
28. Lip,2022. Building Responsive User Interfaces with ReactJS and CSS. Search date 3.5.2024. <https://taglineinfotechus.medium.com/building-responsive-user-interfaces-with-reactjs-and-css-d21ed1b99da4>
29. Machado, Breno 2023. Building a Project Management Tool with ASP.NET 7 and React. Search date 17.5.2024. <https://medium.com/@brenobm/building-a-project-management-tool-with-asp-net-f5f40cd84a04>

30. Pritomsarkar, 2022. Unit testing in ASP.NET Core Web API - C# Programming. Search date 17.5.2024. <https://medium.com/c-sharp-programming/unit-testing-in-asp-net-core-web-api-b2e6f7bdb860>
31. GeeksforGeeks,2024. Unit testing software testing. Search date 17.5.2024. <https://www.geeksforgeeks.org/unit-testing-software-testing/>
32. Rick, Anderson 2024. Integration tests in ASP.NET Core. Search date 17.5.2024. <https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-8.0>
33. Chen, Stephen 2023. What is Data Encryption, and Why Is It Important? Search date 18.5.2024. <https://www.titanfile.com/blog/what-is-data-encryption-and-why-is-it-important/>
34. Okta,2022. What is Token-Based Authentication? Search date 18.5.2024. <https://www.okta.com/identity-101/what-is-token-based-authentication/>
35. Wikipedia,2024. Cross-origin resource sharing. Search date 19.5.2024. https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
36. Cloudflare,2020. What is SQL injection? Search date 19.5.2024. <https://www.cloudflare.com/learning/security/threats/sql-injection/>
37. Portswigger,2022. What is cross-site scripting (XSS) and how can it be prevented? Search date 20.5.2024. <https://portswigger.net/web-security/cross-site-scripting>
38. Anderson, Rick,2024. Installing and configuring Web deployment on IIS 8.0 or later. Search date 20.5.2024. <https://learn.microsoft.com/en-us/iis/install/installing-publishing-technologies/installing-and-configuring-web-deploy-on-iis-80-or-later>