

MINISTRY OF EDUCATION AND TRAINING CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY

**GRADUATION THESIS BACHELOR OF ENGINEERING IN
INFORMATION TECHNOLOGY (HIGH-QUALITY PROGRAM)**

Student: Nguyen Van A

Student ID: B1234567

Class: DI20V7F (K47)

Advisor: Dr. Tran Thi B

Can Tho, 12/2024

MINISTRY OF EDUCATION AND TRAINING CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY

**GRADUATION THESIS BACHELOR OF ENGINEERING IN
INFORMATION TECHNOLOGY (HIGH-QUALITY PROGRAM)**

Student: Nguyen Van A

Student ID: B1234567

Class: DI20V7F (K47)

Advisor: Dr. Tran Thi B

Can Tho, 12/2024

APPROVAL PAGE

This thesis, titled “BUILDING A WEB APPLICATION FOR STUDENT MANAGEMENT SYSTEM”, prepared by student Nguyen Van A, Student ID B1234567, has been examined and approved by the thesis examination committee on December 20, 2024.

Advisor	Committee Chairman
Dr. Tran Thi B	Assoc. Prof. Dr. C

Can Tho, December 2024

Dean of College
[Prof. Dr. D]

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Tran Thi B, for her invaluable guidance, continuous support, and encouragement throughout my research and thesis writing process. I am deeply grateful to the faculty members of the Department of Information Technology, College of Information and Communication Technology, Can Tho University, for their excellent teaching and support during my undergraduate studies. I would also like to thank my family and friends for their love, support, and understanding throughout my academic journey. Finally, I acknowledge all the participants who contributed to this research.

Can Tho, December 2024

Nguyen Van A

TÓM TẮT

Luận văn này trình bày việc thiết kế và triển khai hệ thống quản lý sinh viên dựa trên web cho Đại học Cần Thơ. Nghiên cứu giải quyết vấn đề quản lý hồ sơ sinh viên thủ công kém hiệu quả bằng cách phát triển ứng dụng web hiện đại sử dụng React.js cho frontend và Node.js với Express cho backend. Hệ thống triển khai RESTful APIs và sử dụng MongoDB để lưu trữ dữ liệu. Các tính năng chính bao gồm quản lý đăng ký sinh viên, theo dõi điểm số, đăng ký khóa học và tạo báo cáo tự động. Phương pháp nghiên cứu bao gồm phân tích yêu cầu, thiết kế hệ thống sử dụng sơ đồ UML, phát triển agile và kiểm thử toàn diện. Kết quả cho thấy hiệu suất được cải thiện với thời gian xử lý giảm 70% và tỷ lệ hài lòng của người dùng đạt 95%.

Từ khóa: ứng dụng web, quản lý sinh viên, React.js, Node.js, quản lý cơ sở dữ liệu

ABSTRACT

This thesis presents the design and implementation of a web-based student management system for Can Tho University. The study addresses the inefficiency of manual student record management by developing a modern web application using React.js for the frontend and Node.js with Express for the backend. The system implements RESTful APIs and utilizes MongoDB for data storage. Key features include student registration management, grade tracking, course enrollment, and automated report generation. The research methodology encompasses requirements analysis, system design using UML diagrams, agile development, and comprehensive testing. Results demonstrate improved efficiency with a 70% reduction in processing time and a 95% user satisfaction rate.

Keywords: web application, student management, React.js, Node.js, database management

STUDENT DECLARATION

I hereby declare that this thesis, titled “BUILDING A WEB APPLICATION FOR STUDENT MANAGEMENT SYSTEM”, is my original work and has not been submitted in its entirety or in part for any other degree or diploma at this or any other university. All sources used have been acknowledged.

Can Tho, December 2024

Nguyen Van A

TABLE OF CONTENTS

Contents

1	INTRODUCTION	1
1.1	Background and Motivation	1
1.2	Problem Statement	1
1.3	Related Work	1
1.4	Research Objectives	1
1.4.1	General Objective	1
1.4.2	Specific Objectives	1
1.5	Scope and Limitations	2
1.5.1	Scope	2
1.5.2	Limitations	2
1.6	Research Methodology	2
1.7	Thesis Structure	2
2	LITERATURE REVIEW	2
2.1	Overview of Student Management Systems	2
2.1.1	Evolution of SMS	2
2.2	Key Technologies for Web Applications	3
2.2.1	Frontend Technologies	3
2.2.1.1	React.js	3
2.2.1.2	Vue.js	3
2.2.2	Backend Technologies	3
2.2.2.1	Node.js with Express.js	3
2.2.2.2	Python with FastAPI	3
2.2.3	Database Technologies	3
2.2.3.1	MongoDB	3
2.2.3.2	PostgreSQL	3
2.3	Related Work and Case Studies	4
2.3.1	Cloud-Based Student Information Systems	4
2.3.2	Web-Based Educational Management Systems in Vietnam	4
2.3.3	Modern Authentication and Authorization in Educational Apps	4
2.3.4	Real-time Data Synchronization in Distributed Systems	4
2.4	Conclusion	4
3	METHODOLOGY	4
3.1	Research Approach	4
3.2	Agile Development Model	4
3.2.1	Phases of Agile Development	5
3.2.1.1	1. Requirements Analysis	5
3.2.1.1.1	Techniques Used	5
3.2.1.1.2	Key Requirements	5
3.2.1.2	2. System Design	5
3.2.1.2.1	Architectural Design	5
3.2.1.2.2	Database Design	5
3.2.1.2.3	UML Diagrams	5
3.2.1.3	3. Implementation	6

3.2.1.3.1	Frontend Development	6
3.2.1.3.2	Backend Development	6
3.2.1.4	4. Testing	6
3.2.1.4.1	Unit Testing	6
3.2.1.4.2	Integration Testing	6
3.2.1.4.3	User Acceptance Testing (UAT)	6
3.2.1.4.4	Performance Testing	6
3.2.1.5	5. Deployment	6
3.3	Tools and Technologies	7
3.4	Conclusion	7
4	RESULTS AND DISCUSSION	7
4.1	System Implementation Overview	7
4.1.1	Key Features Implemented	7
4.2	Evaluation Metrics and Results	8
4.2.1	Operational Efficiency	8
4.2.2	Data Consistency and Accuracy	8
4.2.3	Accessibility	8
4.2.4	User Satisfaction	8
4.3	Discussion	9
4.3.1	Advantages of the Implemented Solution	9
4.3.2	Comparison with Related Work	9
4.3.3	Limitations and Future Work	9
4.4	Conclusion	9
5	CONCLUSION	10
5.1	Summary of Findings	10
5.2	Contributions of the Research	10
5.3	Recommendations for Future Work	10
5.4	Concluding Remarks	11
5.5	English References	12
5.6	Vietnamese References	12
5.7	Web References	12
5.8	Appendix A: Survey Questionnaire	13
5.9	Appendix B: Code Snippets	13
5.10	Appendix C: Raw Data	13

LIST OF TABLES

Contents

LIST OF FIGURES

Contents

LIST OF ABBREVIATIONS

CTU	Can Tho University
ICT	Information and Communication Technology
UML	Unified Modeling Language
API	Application Programming Interface
REST	Representational State Transfer
GUI	Graphical User Interface
IDE	Integrated Development Environment

1 INTRODUCTION

1.1 Background and Motivation

The rapid advancement of information technology has transformed how educational institutions manage their operations. Traditional paper-based systems are increasingly inadequate for handling the growing volume of student data and administrative tasks. Can Tho University, with thousands of students enrolled annually, faces challenges in efficiently managing student records, course registrations, and academic performance tracking.

Manual processes are time-consuming, error-prone, and limit the accessibility of information to authorized personnel. There is a critical need for an automated, web-based solution that can streamline these processes and provide real-time access to information.

1.2 Problem Statement

The current student management system at Can Tho University faces several challenges:

- **Time-consuming manual processes:** Staff spend excessive time on data entry and record maintenance
- **Data inconsistency:** Multiple versions of student records exist across different departments
- **Limited accessibility:** Information is not readily available to students and faculty
- **Lack of integration:** Various systems operate independently, causing data silos
- **Reporting difficulties:** Generating comprehensive reports requires manual data compilation

These issues result in reduced operational efficiency, increased administrative costs, and diminished user satisfaction.

1.3 Related Work

Several researchers have addressed student management system development:

Smith and Johnson (2020) developed a cloud-based student information system using Java and MySQL, demonstrating 60% improvement in data retrieval speed. However, their system lacked modern user interface design and mobile responsiveness.

According to Nguyen et al. (2021), implementing web-based educational management systems can reduce administrative workload by up to 50%. Their study focused on Vietnamese universities but did not address integration with existing systems.

1.4 Research Objectives

1.4.1 General Objective

To design and develop a comprehensive web-based student management system that improves operational efficiency and user experience at Can Tho University.

1.4.2 Specific Objectives

1. Analyze requirements and design system architecture using modern web technologies
2. Implement core functionalities including student enrollment, course management, and grade tracking
3. Develop a responsive user interface accessible across multiple devices
4. Integrate the system with existing university databases
5. Evaluate system performance and user satisfaction through testing and surveys

1.5 Scope and Limitations

1.5.1 Scope

The system covers:

- Student registration and profile management
- Course enrollment and scheduling
- Grade recording and transcript generation
- User authentication and authorization
- Administrative reporting and analytics

1.5.2 Limitations

This research does not include:

- Financial management and fee collection
- Library management integration
- Human resources management
- Mobile native applications (focus on responsive web)

1.6 Research Methodology

The development follows an agile methodology with the following phases:

1. **Requirements Analysis:** Gather requirements through interviews and surveys
2. **System Design:** Create UML diagrams and database schemas
3. **Implementation:** Develop using React.js, Node.js, and MongoDB
4. **Testing:** Conduct unit testing, integration testing, and user acceptance testing
5. **Deployment:** Deploy on cloud infrastructure with continuous integration

1.7 Thesis Structure

This thesis is organized into five chapters:

Chapter 1: Introduction - Presents the background, problem statement, objectives, and methodology

Chapter 2: Literature Review - Reviews relevant theories, technologies, and related research

Chapter 3: System Design and Implementation - Describes the system architecture, design decisions, and implementation details

Chapter 4: Results and Discussion - Presents experimental results, performance evaluation, and discussion

Chapter 5: Conclusion - Summarizes findings, contributions, and recommendations for future work

2 LITERATURE REVIEW

2.1 Overview of Student Management Systems

Student Management Systems (SMS) are comprehensive software platforms designed to manage student data, academic records, and administrative processes within educational institutions.

Modern SMS often leverage web technologies to provide accessible and efficient services to students, faculty, and administrators.

2.1.1 Evolution of SMS

Early SMS were typically desktop-based applications focused on basic record-keeping. With the advent of the internet, web-based SMS emerged, offering remote access and improved data sharing

capabilities. Cloud-based and mobile-responsive systems represent the latest advancements, emphasizing scalability, accessibility, and user experience.

2.2 Key Technologies for Web Applications

2.2.1 Frontend Technologies

Modern web frontend development often relies on JavaScript frameworks to create dynamic and interactive user interfaces.

2.2.1.1 React.js

React.js is a JavaScript library for building user interfaces, developed by Facebook. Its component-based architecture facilitates the creation of reusable UI elements, promoting modularity and maintainability. React's virtual DOM enhances performance by minimizing direct manipulation of the browser's DOM.

2.2.1.2 Vue.js

Vue.js is a progressive JavaScript framework for building user interfaces. It is designed to be incrementally adoptable, meaning it can be easily integrated into existing projects. Vue is known for its simplicity and ease of learning, offering a flexible and performant solution for frontend development.

2.2.2 Backend Technologies

Backend technologies are responsible for server-side logic, database interaction, and API management.

2.2.2.1 Node.js with Express.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to use JavaScript for server-side programming, enabling a full-stack JavaScript development approach. Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

2.2.2.2 Python with FastAPI

FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints. It is known for its high performance, automatic interactive API documentation (Swagger UI and ReDoc), and ease of use.

2.2.3 Database Technologies

Databases are crucial for storing and managing the vast amounts of data generated by SMS.

2.2.3.1 MongoDB

MongoDB is a NoSQL document database. It stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. This flexibility makes it suitable for handling semi-structured or rapidly evolving data schemas.

2.2.3.2 PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system. It has a strong reputation for reliability, feature robustness, and performance. PostgreSQL supports SQL (relational) and JSON (non-relational) querying, making it versatile for various data storage needs.

2.3 Related Work and Case Studies

2.3.1 Cloud-Based Student Information Systems

Smith and Johnson (2020) developed a cloud-based student information system using Java and MySQL. Their system aimed to improve data retrieval speeds and provide a centralized platform for academic records. While successful in performance, it lacked modern UI/UX principles and mobile responsiveness, which are crucial in today's educational landscape.

2.3.2 Web-Based Educational Management Systems in Vietnam

Nguyen et al. (2021) conducted a study on implementing web-based educational management systems in Vietnamese universities. Their findings indicated that such systems could significantly reduce administrative workloads by up to 50%. However, their research highlighted challenges in integrating new systems with existing legacy infrastructure and ensuring data consistency across different departments.

2.3.3 Modern Authentication and Authorization in Educational Apps

Chandra and Gupta (2019) explored the importance of robust authentication and authorization mechanisms in educational applications. They emphasized the need for secure user management, role-based access control (RBAC), and protection against common web vulnerabilities to safeguard sensitive student data. Their proposed solution leveraged JWT (JSON Web Tokens) for stateless authentication.

2.3.4 Real-time Data Synchronization in Distributed Systems

For systems handling concurrent updates, such as course registration, real-time data synchronization is critical. Lee and Kim (2022) investigated various strategies for ensuring data consistency in distributed educational platforms, including optimistic and pessimistic locking mechanisms, and the use of event-driven architectures to propagate changes across microservices.

2.4 Conclusion

The literature review underscores the necessity of modern, robust, and user-friendly web-based student management systems. Key technologies like React.js, Node.js, and MongoDB offer suitable foundations for building such systems, addressing the shortcomings of traditional approaches. The reviewed works also highlight important considerations for system design, including integration, data consistency, security, and performance. This research aims to build upon these insights to create an efficient and effective solution for Can Tho University.

3 METHODOLOGY

This chapter details the research methodology employed in the design and development of the web-based student management system. An agile development approach was adopted to ensure flexibility, iterative progress, and continuous feedback integration throughout the project lifecycle.

3.1 Research Approach

The project followed a design science research methodology, focusing on creating an innovative artifact (the student management system) to solve a real-world problem. This approach involves identifying a problem, defining objectives, designing and developing an artifact, demonstrating its utility, and evaluating its performance.

3.2 Agile Development Model

An agile software development model, specifically Scrum, was chosen for its iterative and incremental nature. This allowed for adaptability to changing requirements and facilitated close

collaboration with potential users. The project was divided into several sprints, each focusing on delivering a functional increment of the system.

3.2.1 Phases of Agile Development

3.2.1.1 Requirements Analysis

This phase involved gathering and analyzing the functional and non-functional requirements for the student management system.

3.2.1.1.1 Techniques Used

1. **Interviews:** Structured interviews were conducted with key stakeholders, including university administrators, faculty members, and a sample of students, to understand their needs, pain points, and expectations from the new system.
2. **Surveys:** Online surveys were distributed to a wider student population to gather quantitative data on desired features and priorities.
3. **Document Analysis:** Existing documents, such as academic regulations, student handbooks, and current administrative workflows, were reviewed to identify critical data points and processes.

3.2.1.1.2 Key Requirements

1. User authentication and authorization (students, faculty, administrators)
2. Student profile management (personal information, academic history)
3. Course catalog browsing and enrollment
4. Grade entry and viewing
5. Class scheduling and attendance tracking
6. Automated reporting (transcripts, class lists)
7. Responsive user interface
8. Robust data security and privacy

3.2.1.2 System Design

Based on the gathered requirements, the system architecture and detailed design were formulated.

3.2.1.2.1 Architectural Design

The system adopted a three-tier architecture:

1. **Presentation Layer (Frontend):** Developed using React.js, responsible for the user interface and user interaction.
2. **Application Layer (Backend):** Developed using Node.js with Express.js, handling business logic, API endpoints, and data processing.
3. **Data Layer (Database):** MongoDB was chosen for data persistence due to its flexibility and scalability, storing student records, course information, and other system data.

3.2.1.2.2 Database Design

A NoSQL document-oriented approach was used for MongoDB. Collections were designed to store related data, such as students, courses, enrollments, and grades. Relationships between collections were managed through references, optimizing for read performance and schema flexibility.

3.2.1.2.3 UML Diagrams

Unified Modeling Language (UML) diagrams were employed to visualize the system's structure and behavior.

1. **Use Case Diagrams:** Illustrated the interactions between users (actors) and the system.

2. **Class Diagrams:** Depicted the static structure of the system, showing classes, attributes, methods, and relationships.
3. **Sequence Diagrams:** Showed the interactions between objects in a sequential order, particularly for key processes like student enrollment or grade submission.
4. **Activity Diagrams:** Modeled the workflow of various processes within the system.

3.2.1.3 3. Implementation

The system was developed iteratively in accordance with the agile methodology.

3.2.1.3.1 Frontend Development

React.js was used to build a responsive and intuitive user interface. Key components included:

1. User dashboards (student, faculty, admin)
2. Forms for data entry (e.g., course registration, grade submission)
3. Data display components (e.g., student transcripts, course schedules)
4. Navigation and routing using React Router.

3.2.1.3.2 Backend Development

Node.js with Express.js was utilized to create RESTful APIs. Key functionalities implemented included:

1. User authentication and authorization using JWT (JSON Web Tokens).
2. API endpoints for CRUD (Create, Read, Update, Delete) operations on student, course, and grade data.
3. Business logic for enrollment rules, grade calculations, and report generation.
4. Integration with MongoDB using Mongoose ODM (Object Data Modeling).

3.2.1.4 4. Testing

A multi-faceted testing strategy was employed to ensure the system's quality, functionality, and performance.

3.2.1.4.1 Unit Testing

Individual functions and components were tested in isolation to verify their correctness. Jest was used for React.js components, and Mocha/Chai for Node.js backend functions.

3.2.1.4.2 Integration Testing

This involved testing the interactions between different modules and services, particularly between the frontend and backend, and the backend with the database. Automated integration tests ensured that data flowed correctly across the system.

3.2.1.4.3 User Acceptance Testing (UAT)

A selected group of end-users (students, faculty, administrators) participated in UAT. They tested the system against predefined scenarios to confirm that it met their requirements and was user-friendly. Feedback from UAT was incorporated into subsequent development sprints.

3.2.1.4.4 Performance Testing

Load testing was conducted to assess the system's responsiveness and stability under various user loads, ensuring it could handle the anticipated number of concurrent users during peak times (e.g., course registration periods).

3.2.1.5 5. Deployment

The system was deployed to a cloud platform (e.g., Heroku, AWS EC2) to ensure accessibility and scalability. A continuous integration/continuous deployment (CI/CD) pipeline was established to automate the build, test, and deployment processes, facilitating rapid iterations and updates.

3.3 Tools and Technologies

Category	Tool/Technology	Description
Frontend	React.js	JavaScript library for UI development
	HTML5, CSS3	Standard web languages
	Axios	HTTP client for API requests
Backend	Node.js	JavaScript runtime environment
	Express.js	Web framework for Node.js
	Mongoose	ODM for MongoDB
	JWT	JSON Web Tokens for authentication
Database	MongoDB	NoSQL document database
Testing	Jest	JavaScript testing framework (frontend)
	Mocha, Chai	JavaScript testing frameworks (backend)
Version Control	Git, GitHub	Distributed version control system
Deployment	Heroku/AWS EC2	Cloud platform for hosting
UI/UX	Figma	Prototyping and design tool

3.4 Conclusion

The agile development methodology, combined with a robust technology stack, provided a structured yet flexible framework for developing the student management system. The iterative nature of the approach allowed for continuous improvement and ensured that the final product effectively addressed the needs of Can Tho University. The subsequent chapters will present the detailed implementation and evaluation of the system.

4 RESULTS AND DISCUSSION

This chapter presents the results obtained from the implementation and evaluation of the web-based student management system. It also discusses the implications of these results, comparing the system's performance and user satisfaction with the objectives outlined in Chapter 1.

4.1 System Implementation Overview

The student management system was successfully implemented using React.js for the frontend, Node.js with Express.js for the backend, and MongoDB for the database. The system provides a comprehensive suite of features designed to streamline administrative tasks and enhance user experience.

4.1.1 Key Features Implemented

- User Authentication and Authorization:** Secure login for students, faculty, and administrators with role-based access control.
- Student Profile Management:** Functionality for students to view and update personal information, and for administrators to manage student records.
- Course Management:** Administrators can create, update, and delete courses. Students can browse the course catalog and enroll in available courses.
- Grade Management:** Faculty can submit and view grades for their assigned courses. Students can view their academic transcripts and individual course grades.
- Class Scheduling:** Integration with a basic scheduling module to display class timings and locations.

6. **Automated Reporting:** Generation of academic transcripts, class lists, and other administrative reports.
7. **Responsive User Interface:** The system is accessible and fully functional across various devices, including desktops, tablets, and smartphones.

4.2 Evaluation Metrics and Results

The system's performance and effectiveness were evaluated against predefined metrics, including operational efficiency, data consistency, accessibility, and user satisfaction.

4.2.1 Operational Efficiency

The primary goal was to reduce the time and effort associated with manual administrative processes.

1. Reduced Processing Time:

- **Course Enrollment:** Manual enrollment processes typically took 10-15 minutes per student. With the new system, online enrollment takes an average of 2-3 minutes, representing a **70-80% reduction**.
- **Grade Submission:** Faculty reported a 60% reduction in time spent on grade submission due to automated validation and direct entry.
- **Report Generation:** Generating student transcripts and class lists, which previously took hours of manual compilation, now takes seconds.

2. Streamlined Workflows:

- Administrative staff noted a significant decrease in paperwork and manual data entry tasks, allowing them to focus on more critical activities.
- The centralized database eliminated the need for data reconciliation across different departments.

4.2.2 Data Consistency and Accuracy

The system enforced data validation rules at the point of entry and utilized a centralized MongoDB database, leading to improved data consistency.

1. **Error Rate Reduction:** The occurrence of data entry errors, such as incorrect student IDs or course codes, decreased by 90% compared to manual methods.
2. **Real-time Updates:** All stakeholders access the most current data, preventing discrepancies that often arose from outdated spreadsheets or paper records.

4.2.3 Accessibility

The web-based nature and responsive design of the system significantly enhanced accessibility.

1. **24/7 Access:** Students and faculty can access the system anytime, anywhere, facilitating flexible learning and administrative tasks.
2. **Multi-device Support:** The system performed consistently across various browsers and devices, ensuring a seamless user experience regardless of the access point.

4.2.4 User Satisfaction

User satisfaction was assessed through post-implementation surveys and feedback sessions with students, faculty, and administrators.

1. **Overall Satisfaction:** 95% of users reported being satisfied or very satisfied with the new system.
2. **Key Positives:** Users highlighted the ease of use, intuitive interface, and speed of operations as major improvements.

3. **Areas for Improvement:** Minor suggestions included more advanced notification features and deeper integration with external university services (e.g., library systems), which are noted for future enhancements.

4.3 Discussion

The results demonstrate that the developed web-based student management system successfully achieved its objectives of improving operational efficiency and user experience at Can Tho University. The adoption of modern web technologies proved to be a robust solution for addressing the challenges posed by traditional manual systems.

4.3.1 Advantages of the Implemented Solution

1. **Scalability:** The choice of Node.js and MongoDB provides a highly scalable architecture capable of handling a growing number of students and data volumes.
2. **Maintainability:** The component-based approach of React.js and the modular structure of the Node.js backend contribute to easier maintenance and future feature additions.
3. **Cost-Effectiveness:** Open-source technologies (React.js, Node.js, MongoDB) significantly reduced software licensing costs compared to proprietary solutions.
4. **Security:** Implementation of JWT for authentication and adherence to secure coding practices ensured the protection of sensitive student data.

4.3.2 Comparison with Related Work

Compared to systems like the cloud-based solution by Smith and Johnson (2020), our system prioritizes a modern, responsive user interface and enhanced user experience, directly addressing the limitations identified in their work. Similar to Nguyen et al. (2021), our system also achieved substantial reductions in administrative workload, but with a stronger focus on API-driven integration for future scalability. The secure authentication mechanisms align with the recommendations from Chandra and Gupta (2019).

4.3.3 Limitations and Future Work

While the current system is robust, certain limitations provide opportunities for future enhancements:

1. **Integration with External Systems:** Deeper integration with university's financial system, library system, and research management platforms.
2. **Advanced Analytics:** Implementation of advanced data analytics and machine learning for predictive insights into student performance or resource allocation.
3. **Mobile Native Applications:** Development of dedicated mobile applications for iOS and Android platforms to complement the responsive web interface.
4. **Personalized Learning Paths:** Incorporating features for personalized course recommendations or academic advising.
5. **Real-time Collaboration Tools:** Adding features for group projects, online discussions, and virtual office hours.

4.4 Conclusion

The web-based student management system successfully modernized the administrative processes at Can Tho University. The project delivered a high-performing, user-friendly, and secure application that significantly improved efficiency and user satisfaction. The agile methodology allowed for adaptive development, and the chosen technology stack provides a strong foundation for future growth and enhancements. The findings confirm the benefits of adopting contemporary web development practices in educational technology.

5 CONCLUSION

This thesis successfully designed, developed, and evaluated a comprehensive web-based student management system for Can Tho University. The primary objective was to enhance operational efficiency and improve the user experience, addressing the limitations of existing manual and fragmented systems.

5.1 Summary of Findings

The implemented system, built using React.js for the frontend, Node.js with Express.js for the backend, and MongoDB for data persistence, demonstrated significant improvements across various aspects:

1. Operational Efficiency: Processing times for key administrative tasks such as course enrollment and grade submission were reduced by 60-80%. This led to streamlined workflows and a considerable decrease in manual data entry and paperwork.
2. Data Consistency and Accuracy: The centralized database and enforced data validation rules minimized errors and ensured real-time access to accurate information for all stakeholders.
3. Accessibility: The responsive web interface provided 24/7 access from any device, greatly enhancing convenience for students, faculty, and administrators.
4. User Satisfaction: Post-implementation surveys indicated a high level of user satisfaction, with 95% of users reporting positive experiences due to the system's intuitive design and speed.

These findings confirm that the system successfully met its general and specific objectives, providing a robust and modern solution for student management at Can Tho University.

5.2 Contributions of the Research

This research makes several contributions:

1. Practical Solution: Delivers a functional and evaluated web-based student management system that can be deployed at Can Tho University, directly addressing a critical need for modernization.
2. Demonstration of Agile Methodology: Provides a case study for the effective application of agile development principles in a university IT project, showcasing its benefits in adaptability and iterative delivery.
3. Application of Modern Web Technologies: Illustrates the successful integration of React.js, Node.js, and MongoDB in a scalable and performant enterprise application within an educational context.
4. Foundation for Future Development: The modular and API-driven architecture provides a strong foundation for future enhancements, integrations, and the development of more advanced features.

5.3 Recommendations for Future Work

Based on the insights gained and the limitations identified during this project, the following recommendations are proposed for future work:

1. Deep Integration with University Ecosystem: Integrate with other existing university systems, including the financial accounting system for fee management, the library system for resource access, and the alumni portal for post-graduation tracking.
2. Advanced Analytics and Reporting: Implement business intelligence dashboards and predictive analytics capabilities to offer insights into student performance trends, resource allocation optimization, and early intervention for at-risk students.

3. Mobile Native Applications: Develop dedicated mobile applications for iOS and Android platforms to provide an even more tailored and optimized experience for smartphone users, potentially incorporating push notifications for important updates.
4. Personalized Learning Support: Explore the integration of AI-driven features for personalized course recommendations, academic advising, and career guidance based on student performance and interests.
5. Enhanced Communication and Collaboration Tools: Incorporate features like internal messaging, forum functionality, or virtual office hours to foster better communication between students and faculty.
6. Scalability and Performance Optimization: Conduct further load testing and performance tuning to ensure the system can comfortably handle exponential growth in user base and data volume, especially during peak registration periods.

5.4 Concluding Remarks

The successful development and deployment of this web-based student management system mark a significant step forward for Can Tho University in its journey towards digital transformation. By embracing modern technology and agile practices, the university can now manage its student-related operations more efficiently, provide a superior experience to its academic community, and pave the way for future innovations in educational technology. This project serves as a testament to the power of well-applied software engineering principles in solving complex organizational challenges.

REFERENCES

5.5 English References

- Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, 39(4), 12-27.
- Chen, L., & Wang, H. (2022). Microservices architecture for educational management systems. *Journal of Educational Technology*, 45(3), 234-256.
- Facebook. (2023). React - A JavaScript library for building user interfaces. Available from <https://react.dev>, accessed on 15 November 2023.
- Fielding, R.T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. University of California, Irvine.
- MongoDB Inc. (2023). MongoDB Documentation. Available from <https://docs.mongodb.com>, accessed on 20 November 2023.
- Node.js Foundation. (2023). Node.js Documentation. Available from <https://nodejs.org/docs>, accessed on 18 November 2023.

5.6 Vietnamese References

- Nguyễn Văn A., Trần Thị B., & Lê Văn C. (2021). Phát triển hệ thống quản lý sinh viên cho các trường đại học Việt Nam. *Tạp chí Khoa học Công nghệ Thông tin*, 15(2), 45-62.

5.7 Web References

- Can Tho University. (2023). Academic Regulations. Available from <https://www.ctu.edu.vn/academic-regulations>, accessed on 25 November 2023.

APPENDICES

Appendices are supplementary materials that are not essential to the main body of the thesis but provide additional information, such as raw data, detailed code listings, survey questionnaires, or extended explanations.[Include your survey questionnaire here][Include relevant code snippets here][Include any raw data or experimental results not presented in the main chapters]