Academic Administration System for WiseWay Academy

General Sir John Kotelawala Defense University
Faculty of Computing

Group Member Details

Reg.No	Name	Degree
D/BCS/24/0004	W.A.D.S. Wickrama Arachchi	Computer Science
D/BCS/24/0019	S.S.N. Keerthiwardhana	Computer Science
D/BCS/24/0033	L.S.Y. Liyanage	Computer Science
D/BSE/24/0015	P.D.O. Nawanjana	Software Engineering
6872	T.A.T.S. Thennakoon	Computer Science

Supervisor : Dr.H.R.W.P.Gunathilake

March 2024

Abstract

Academic Administration System is a web-based application designed exclusively for administrative users to streamline student registration, payment management, and lecturer information storage. Unlike a Learning Management System, this system focuses solely on administrative functions.

The system enables administrators to register students, generating unique registration numbers automatically. A secure login system ensures restricted access. Payment details are manually entered by administrators, stored in a database, and categorized with status tags such as "Pending Payment" and "Paid" for easy tracking. The Daily Payments feature simplifies the process of recording student payments efficiently.

Additionally, the system maintains a lecturer database, storing essential details and associated class schedules. The dashboard provides quick access to critical features such as student registration, daily payment management, lecturer registration and student profile modifications. A class timetable for the current week is displayed on the login page for easy reference.

This system enhances administrative efficiency by organizing and centralizing student and lecturer data, ensuring a smooth workflow for educational institutions

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background of Studies

Educational institutions often face challenges in managing student registrations, payments, and lecturer records due to manual administrative processes. These inefficiencies can lead to errors, delays, and difficulties in tracking financial transactions. To address this, a centralized Institute Management System is proposed, designed exclusively for administrative users.

This system streamlines student registration, automatically generating unique registration numbers, and allows manual payment entry with categorized status tags such as pending payment and paid. It also maintains a lecturer database, storing essential details and assigned classes.

A daily payments feature simplifies fee recording, while the dashboard provides quick access to core functions like student and lecturer management, salary generation, and timetable viewing. By digitizing these administrative tasks, the system enhances efficiency, accuracy, and organization, ensuring a smoother workflow for educational institutes.

1.2 Problem Identification

Wiseway Academy faces challenges in managing its administrative processes due to outdated manual methods for student registration, payment tracking, and lecturer record-keeping. The absence of an automated system leads to delays, errors, and inefficiencies in retrieving and managing critical data. Currently, student registration lacks a streamlined process for generating unique registration numbers, resulting in inconsistencies. Payment management is also problematic, as transactions are recorded manually without a centralized tracking system, making it difficult to monitor pending and completed payments. Additionally, maintaining lecturer records and organizing class schedules manually increases the risk of mismanagement and scheduling conflicts. Without a centralized dashboard to efficiently handle student registration, payments, and timetable management, administrative workflows become time-consuming and prone to errors. To address these challenges, this project proposes the development of an Institute Management System for Wiseway Academy, providing a structured, digital solution to enhance accuracy, efficiency, and organization in administrative operations.

1.3 Objectives

The primary objective of this project is to develop a centralized Academic Administration System for Wiseway Academy to enhance administrative efficiency by automating key processes such as student registration, payment management, and lecturer record-keeping. The system will provide a structured and user-friendly platform for administrators to manage daily operations effectively.

1. Automate Student Registration – Develop a system that generates unique registration numbers automatically and securely stores student records.
2. Streamline Payment Management – Enable administrators to manually input and track student payments efficiently, categorizing them as "Pending Payment" or "Paid."
3. Maintain a Lecturer Database – Store essential lecturer details, including assigned classes, to facilitate organized scheduling and record-keeping.
4. Provide a Centralized Dashboard – Design an intuitive interface for administrators to access student profiles, payment records, lecturer details, and salary generation.
5. Display a Class Timetable – Integrate a feature that presents the weekly timetable on the login page for quick reference.

Chapter 2 Literature Review

2.1 Importance of an Academic Administration System

An Academic Administration System significantly enhances administrative efficiency within educational institutions by automating time-consuming processes such as student admissions and academic tracking, thereby reducing errors and freeing up valuable time for staff [1]. By centralizing data into a single, accessible platform, an Academic Administration System facilitates informed decision-making and effective resource management, aligning with institutional effectiveness planning goals [2]. Moreover, Academic Administration System solutions improve the quality of education delivery by enabling responsive and flexible learning environments, which are crucial for adapting to the rapidly changing needs of students and educators [3]. By minimizing operational costs and boosting efficiency, an Academic Administration System provides educational institutions with a competitive edge in managing their administrative operations.

2.2 Existing Solutions and Their Limitations

Existing Solutions	Features	Limitations
Fedena - School ERP	- Customizable Dashboards: Admins can configure widgets to display key data, tailoring the interface to their needs. - Cloud-Based with Remote Access: Hosted online, allowing admins to log in from any browser, ideal for distributed management. - Open-Source: Free version available without a licensing fee.	- Customizable Dashboards: Admins can configure widgets to display key data, tailoring the interface to their needs. - Cloud-Based with Remote Access: Hosted online, allowing admins to log in from any browser, ideal for distributed management. - Open-Source: Free version available without a licensing fee.
Ellucian Banner - Student Information System	- Highly Scalable: Handles thousands of students and staff across multiple campuses, ideal for large universities. - Cloud-Hosted with Security: Offers encrypted cloud hosting with compliance to standards like FERPA. - High Availability: 99.9% uptime improves operational and security continuity.	- Overly Complex for Small Institutes: Features like multi-department workflows and advanced analytics may overwhelm small teams needing basic tools, with a steep learning curve for admins. - Expensive Licensing and Implementation: Costs (often $10,000+ annually plus setup fees) are prohibitive for small budgets, requiring dedicated IT staff. - Higher Education Focus: Tailored to universities, its features (e.g., degree auditing) don’t align with smaller institutes’ simpler admin needs.
	- Detailed Reporting: Generates comprehensive reports (e.g., payment summaries, student demographics) exportable as PDFs or spreadsheets, aiding financial audits. - Scalable for Multiple Users: Supports multiple admin logins and can expand to manage	- Dated Interface Hampers Usability: The UI, built on older PHP frameworks, has a minimalistic design and slow navigation. - Limited Automation for Payment Notifications: No built-in alerts for overdue payments

OpenSIS	several campuses with a single instance. - Optional Local Hosting: Can be installed on a local server for institutions preferring data control over cloud reliance.	or automatic status changes (e.g., "Pending" to "Paid"), forcing admins to manually monitor and update records. - Backup Integration with Modern Cloud Solutions is Weak: While locally hosted backups are possible, integrating with cloud services like AWS S3 requires custom configuration.
TUIO	- User-Friendly Interface: Clean, modern design with intuitive navigation, appealing to non-technical users. - Transparent Pricing: Flat-rate plans (e.g., $10/month per user) with no hidden fees, budget-friendly for small schools. - Strong Customer Support: Offers onboarding help and responsive troubleshooting, easing adoption.	- Limited Customization Options: Lacks flexibility to adapt features like payment workflows or lecturer profiles to specific institute needs, restricting admin control. - Limited Scalability: Struggles to handle rapid growth without performance lags or added costs. - Lacks Advanced Reporting: No real-time analytics or detailed financial summaries, limiting decision-making support.
Classter	- Comprehensive Student Information Management: Offers tools for enrollment, attendance tracking, grade reporting, and communication, serving as a one-stop-shop for student data management. - Cloud-Based and User-Friendly: Accessible from any device with an intuitive interface, facilitating easy adoption by staff and students. - Modular Design: Allows institutions to select and pay for only the modules they need, making it cost-effective for various sizes of institutions.	- Complexity in Initial Setup: The wide array of features and customization options can make the initial setup process time-consuming and may require training. - Dependence on Internet Connectivity: Being cloud-based, a stable internet connection is essential for optimal performance, which might be a limitation in areas with unreliable internet access. - Additional Costs for Advanced Features: While the modular approach is cost-effective, adding advanced features can

		increase the overall cost, which might be a consideration for institutions with tight budgets.
PowerSchool	- Integrated Classroom Management: Combines student information system capabilities with classroom management tools, providing a unified platform for teachers and administrators. - Parent and Student Portals: Enhances engagement by allowing parents and students to access grades, attendance, and assignments in real-time. - Customizable Reporting: Offers robust reporting tools that can be tailored to meet the specific needs of the institution.	- High Cost for Small Institutions: The comprehensive feature set comes with a higher price tag, which might be prohibitive for smaller schools or districts. - Steep Learning Curve: The extensive functionalities can be overwhelming, requiring significant training for staff to utilize the system effectively. - Limited Customization for Specific Needs: While offering many features, some institutions may find limitations in customizing the system to their unique processes and workflows.

Table 1: Existing Systems

2.3 Research Gaps

User Experience (UX) and Interface Design

Existing Student Information Systems (SIS) often lack intuitive interfaces, leading to steep learning curves for users. A study by Al-Hunaiyyan et al. (2021) highlighted that many SIS platforms do not adequately consider user-centered design principles, resulting in systems that are not user-friendly. This issue is particularly pronounced in smaller institutions where administrative staff may not have extensive technical expertise. Therefore, there is a need for research focused on designing SIS interfaces that prioritize usability and enhance user adoption rates [4].

Customization and Scalability

Many current SIS platforms are criticized for their rigidity and lack of adaptability to diverse institutional needs. This inflexibility poses challenges for institutions with unique administrative processes or those experiencing growth. Research on developing modular SIS architectures that allow for easy customization and scalability without incurring significant costs or complexity is limited, highlighting a need for further investigation in this area [5].

Data Collection and Reporting Discrepancies

Some institutions face challenges with discrepancies between back-end data collection and front-end reporting in their SIS platforms. Inconsistencies can lead to inaccurate administrative insights and hinder decision-making processes. Research into standardizing data collection and reporting processes within SIS is limited, suggesting a need for studies aimed at improving data consistency and reliability [5].

Data Privacy and Security

With the increasing prevalence of cyber threats, ensuring data privacy and security within SIS platforms is paramount. However, there is a scarcity of comprehensive studies addressing the enhancement of data security measures in SIS, particularly concerning compliance with evolving regulations like the General Data Protection

Regulation (GDPR). This gap underscores the necessity for research focused on developing robust security frameworks for SIS to protect sensitive student data [6].

2.4 Proposed System: Academic Administration System

The proposed Academic Administration System is a web-based tool designed exclusively for admin users to manage student registration, payment tracking, lecturer management, and basic timetable viewing for small-to-medium institutes. It overcomes limitations in existing systems like Fedena, OpenSIS, Ellucian Banner, and TUIO by introducing innovative features that enhance efficiency and usability, tailored to the needs of small administrative teams.

1. Simplified, Admin-Focused Payment Automation

- Gap: Manual updates in Fedena and OpenSIS, basic tracking in TUIO, and overly complex processes in Ellucian Banner burden admins.
- Solution: Smart Daily Payment Entry with Auto-Tagging allows admins to input payments with automatic tag updates (e.g., "Pending" to "Paid") and ID suggestions, streamlining financial tasks.

2. Inadequate Real-Time Timetable Management

- Gap: Static tools in Fedena and OpenSIS, limited TUIO features, and intricate options in Ellucian hinder basic admin needs.

- Solution: Simple Timetable View with Edit Option provides a straightforward, editable timetable display upon login, with minimal complexity.

3. Limited Cost-Effective Scalability

- Gap: High costs in Ellucian, technical barriers in Fedena and OpenSIS, and TUIO’s growth limits restrict small institutes.
- Solution: Lean Scalable Design with Automated Backups supports modular growth and secure data management at a low cost.

2.5 Technology Stack and Justification

Academic Administration System is built using a modern, efficient technology stack tailored for a single admin user on one computer, managing student registration, payment tracking, lecturer management, and a simple timetable view for small-to-medium institutes. The selected technologies React.js for the frontend, Node.js with Express.js for the backend, MySQL as the database, and AWS Lightsail for hosting, with JSON Web Tokens (JWT) for authentication and Tailwind CSS for styling, ensure a lightweight, secure, and scalable solution. This stack supports the system’s core functionalities with simplicity, reliability, and future-proofing in mind.

1. Frontend: React.js

- **Description**: A JavaScript library for dynamic, component-based user interfaces.

- **Justification** :Its reactivity enables an intuitive admin dashboard for viewing and editing student records, payment statuses, and a basic timetable, with real-time updates like auto-suggested IDs during payment entry. React’s component-based design simplifies development of a responsive interface, ensuring usability on the single computer and potential adaptability for future devices, aligning with the need for an efficient, user-friendly experience.

2. Backend: Node.js with Express.js

- **Description**: Node.js is a JavaScript runtime, paired with Express.js for API development.
- **Justification**: Using JavaScript across frontend and backend streamlines development for a small-scale project, reducing complexity. Express.js efficiently handles APIs for student registration, payment updates, and lecturer data, powering features like "Smart Daily Payment Entry with Auto-Tagging" by processing inputs and updating statuses instantly. Its lightweight nature suits a single-user system while supporting future growth.

3. Database: MySQL

- **Description**: An open-source relational database for structured data.

- **Justification**: It organizes relational data—students linked to payments, lecturers to profiles, and timetable entries—with auto-incremented IDs for easy registration. MySQL's reliability and performance ensure quick access to payment logs and lecturer details, while its free availability fits the budget constraints of small institutes, providing a robust foundation for data management.

4. Hosting: AWS Lightsail

- **Description**: A simplified cloud hosting service with virtual servers.
- **Justification**: It organizes relational data—students linked to payments, lecturers to profiles, and timetable entries—with auto-incremented IDs for easy registration. MySQL's reliability and performance ensure quick access to payment logs and lecturer details, while its free availability fits the budget constraints of small institutes, providing a robust foundation for data management.

5. Additional Tools

   i. Authentication: JSON Web Tokens (JWT):

   - **Description**: A standard for secure, token-based authentication.
   - **Justification**: For a single-user, web-based Academic Administration System, JWT ensures secure admin access with minimal complexity, protecting sensitive data (e.g., payment records) by requiring login verification. Its stateless design integrates seamlessly with Node.js and React.js, adding a security layer if the computer is accessed by others and preparing for future multi-user scenarios, supporting the system's long-term viability.

ii. Styling: Tailwind CSS:

- **Description**: A utility-first CSS framework for rapid styling.
- **Justification**: It enables quick development of a clean, modern admin interface, enhancing usability with minimal effort. Tailwind’s utility classes ensure a consistent, professional look for forms and displays, optimizing the single-user experience without requiring extensive custom CSS.

Chapter 3 Proposed Methodology

To ensure efficient, scalable, and secure development, we have chosen the Agile methodology, specifically employing the Scrum Framework. Agile Scrum is a flexible, iterative software development framework that focuses on delivering small, functional increments of the system in short cycles called sprints. This approach enables iterative development, continuous feedback and incremental improvements, making it ideal for a small administrative team with evolving needs.

1. **Requirement Gathering & Product Backlog Creation :** After the discussions with the client, requirements are collected. Once the necessary requirements are gathered, identify key functional and non-functional requirements, convert requirements into user stories and prioritize them in the product backlog and the technology stack is discussed based on system needs.
2. **Sprint Planning :** After creating the product backlog, selecting the high priority user stories for the sprint, finalizing the best tools, framework and technologies and shared tasks among the group members.
3. **Sprint Execution :** In this phase, we work on implementing the selected backlog items within a sprint. This phase involves coding, testing, daily progress tracking, daily scrum meetings and resolving issues to ensure that the features are functional and meet requirements.
4. **Sprint Review :** During this phase, the completed features are presented to stakeholders, demonstrating their functionality and alignment with project

objectives. Feedback is gathered to assess improvements, and necessary adjustments are made to refine future sprint goals.

5. **Sprint Retrospective & Continuous Improvement :** During this phase, we can evaluate the sprint by reviewing what was successful and identifying areas that need improvement. Feedback is analyzed to enhance performance, security, and usability, ensuring continuous refinement of the system for future sprints.

Why Agile Methodology with Scrum Framework is Ideal for the Academic Administration System Project

The Agile methodology with Scrum framework is ideal for the Academic Administration System project due to its flexibility, efficiency, and focus on delivering a high-quality solution. Here are five key reasons:

1. **Incremental and Rapid Delivery**: Scrum's short sprint cycles (2–4 weeks) enable incremental development and delivery of features like student registration, payment tracking, and timetable management, allowing early review and a functional system by November 1, 2025.

2. **Feature Prioritization:** The product backlog ensures critical functionalities (e.g., Smart Payment Entry) are prioritized and completed first, aligning development with the admin's core needs.

3. **Adaptability to Feedback:** Scrum's iterative approach accommodates changes based on admin feedback, ensuring the system evolves to meet institute requirements without derailing progress.

4. **Continuous Testing and Quality:** Testing within each sprint identifies and resolves issues early, delivering a stable, reliable system for the single admin user.

5. **Stakeholder Engagement:** Regular sprint reviews provide opportunities for the admin to assess features and offer input, ensuring the Academic Administration System meets expectations by the final viva on December 1, 2025.

3.1 Flow Diagram

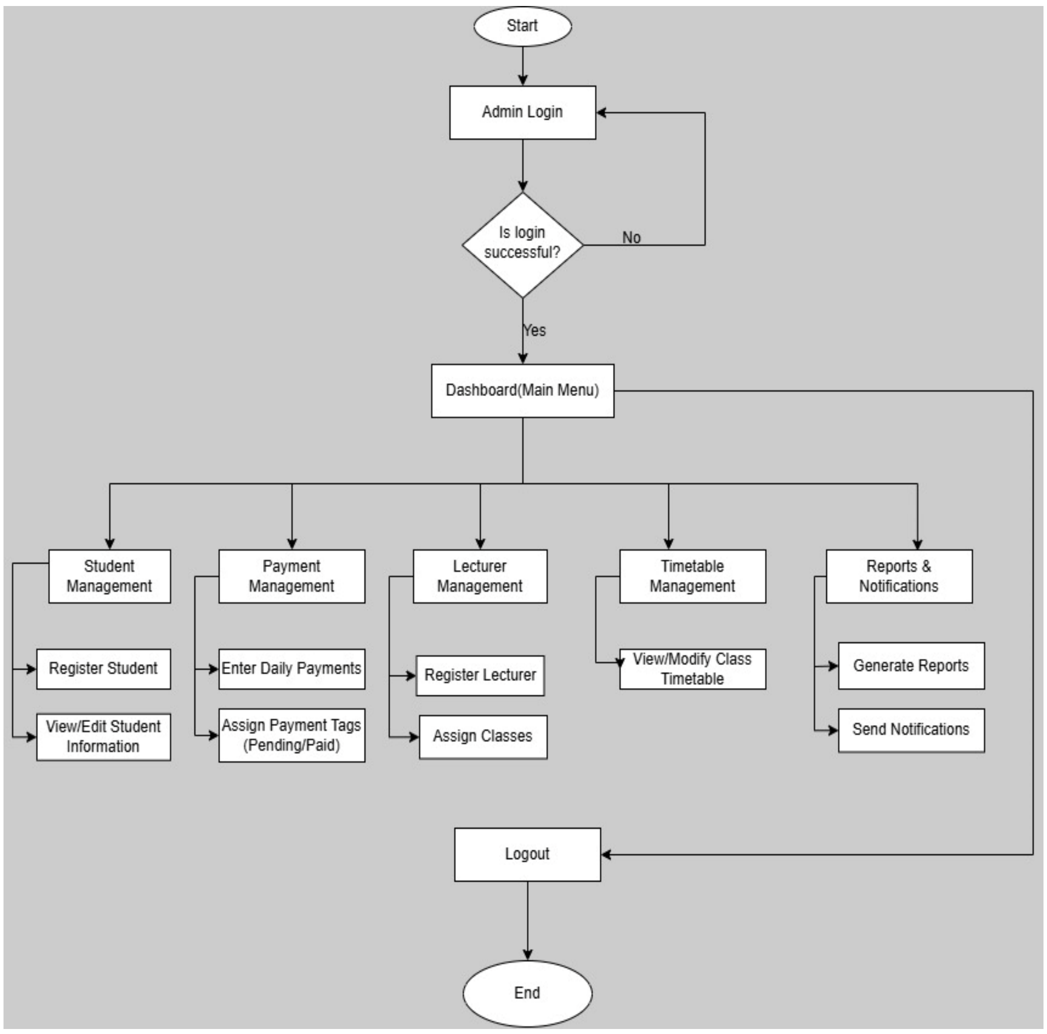

Figure 1: Flow Diagram of the System

3.2 Functional Requirements and Non-Functional Requirements

1. Functional Requirements

   Functional requirements define what the Academic Administration System must do—its specific features and capabilities that the admin can interact with to perform their tasks.

   i. Student Registration

      - The system shall allow the admin to register new students by entering details (e.g., name, contact information).
      - The system shall automatically generate a unique student ID for each registered student.
      - The system shall store student data in a centralized database and display a list of registered students.

   ii. Payment Tracking

      - The system shall enable the admin to manually enter daily payment details (e.g., student ID, amount, date) via a "Smart Daily Payment Entry" feature.
      - The system shall automatically update payment status tags (e.g., "Pending" to "Paid") based on entered payments.
      - The system shall provide an auto-suggest feature for student IDs during payment entry to reduce errors.

- The system shall maintain a payment history viewable by the admin.

iii. Lecturer Management

- The system shall allow the admin to add and update lecturer details (e.g., name, contact, subjects taught).
- The system shall store lecturer data in the database and display a list of lecturers.

iv. Timetable Viewing and Editing

- The system shall display a simple weekly timetable view upon admin login.
- The system shall allow the admin to manually edit timetable entries (e.g., assign lecturers to time slots) with basic modifications saved instantly.

v. Authentication

- The system will require the admin to log in with a username and password to access all features.
- The system shall use JSON Web Tokens (JWT) to secure access and enforce session timeouts (e.g., after 1 hour of inactivity).

2. Non-Functional Requirements

Non-functional requirements specify how the Academic Administration System should perform—its quality attributes like performance, security, and usability, rather than specific actions.

i. Usability

  - The system shall feature an intuitive, user-friendly interface (e.g., clean design with Tailwind CSS) that the admin can navigate without extensive training.
  - The system shall load pages and process inputs (e.g., payment entry) within a few seconds under normal conditions on a single computer.

ii. Security

  - The system shall protect sensitive data (e.g., student records, payment details) with JWT authentication to prevent unauthorized access, even on a single machine.
  - The system shall ensure data integrity by preventing tampering during storage and retrieval.

iii. Scalability

  - The system shall support a "Lean Scalable Design," capable of handling increased numbers of students and lecturers without performance degradation, preparing for potential future expansion.
  - The system shall allow modular upgrades (e.g., additional users or features) with minimal reconfiguration.

iv. Reliability

  - The system shall maintain consistent availability during admin use on the single computer.

- The system shall include automated backups (e.g., daily snapshots via AWS Lightsail) with sufficient retention to recover data in case of failure.

v. Performance

- The system shall process and save payment entries or timetable updates rapidly per transaction.
- The system shall handle database queries (e.g., retrieving student lists) efficiently under typical usage.

vi. Maintainability

- The system will be built with modular code (e.g., React components, Express.js routes) to allow easy updates or bug fixes by a developer.
- The system will use open-source tools (e.g., MySQL, Node.js) to reduce dependency on proprietary software.

vii. Cost-Effectiveness

- The system shall operate within a low-cost hosting budget (e.g., ~$20/month via AWS Lightsail) to suit small institutes.
- The system shall leverage free, open-source technologies to minimize development and operational costs.

3.3 Proposed Testing and Evaluation Method

1. **Unit testing :** To test individual components or features (student registration, payment tracking, lecturer management, timetable viewing and editing) to ensure they work as expected.

   Ex : For the "Student Registration" feature:

   - Verify that student details (e.g., name, contact information) are correctly saved to the database when submitted.
   - Ensure a unique student ID is automatically generated and assigned to each new student upon registration.
   - Test input validation by leaving required fields (e.g., name) blank and confirming that the system displays an appropriate error message.

2. **Integration testing :** To test the interaction between different modules to ensure they function as a whole.

   Ex : For the interaction between "Student Registration" and "Payment Tracking":

   - Register a new student and verify that their details (e.g., student ID, name) are correctly linked to a new payment record in the payment module.
   - Update a student's payments to the system and confirm that this change is accurately reflected in the student profiles with an indicator (Paid /Unpaid/ ending).

3. **System testing :** To ensure the entire system functions as expected in a fully integrated environment and meets the system requirements.

   Ex : Verify the complete system workflow including student registration, payment processing, lecturer management and timetable editing.

4. **Performance testing** : Evaluate the system's speed, responsiveness, and scalability to ensure optimal performance under varying workloads and user interactions.

Ex : Simulate 50+ students registering at the same time and verify that the system does not slow down.

5. **Security testing :** Ensure the protection of sensitive data and prevent unauthorized access by identifying and mitigating potential security vulnerabilities.

   Ex : Attempt to access the admin dashboard without logging in and confirm that unauthorized users are blocked.

6. **Usability testing :** Ensure the system is user-friendly and intuitive for the admin user.

   Ex : Conduct tests with the **real user** to gather feedback on navigation, form usability and ease of access to key functionalities.

7. **User acceptance testing :** Ensure the system complies with business requirements and fulfills user expectations prior to deployment.

   Ex : Allow the institute administrator to test the system for **one day** and provide feedback on student registration, payment tracking, and timetable editing.

Chapter 4 Time Frame

Sprint	Tasks Included	Duration
Sprint1: Project Setup	Install tools(React.js, Node.js, MySQL, AWS Lightsail), plan sprints	Week 1- 2
Sprint 2: Code Modules I	Student registration(UI, API, auto-ID) Lecturer management(CRUD, list)	Week 3- 6
Break: Exam Period	Late April slowdown and May exams, minimal to no work	Week 7-10
Sprint 3:Core Modules II	Payment tracking (Smart Entry, auto-tagging, auto-suggest, history) Timetables	Week 11-15
Sprint 4: Security & UI	JWT authentication, Tailwind CSS styling, UI refinement	Week 16-19
Sprint 5: Testing	Initial testing, Bug fixes, User acceptance testing(UAT)	Week 20-23
Sprint 6: Deployment	Deploy to AWS Lightsail, Configure Backups, Feedback Implementation	Week 24-27
Sprint 7: Finalization	Final Testing, Optimization, Creating user manuals, Training admins, Final Deployment	Week 28-31
Post-Completion	Customer reaction, Error fixes	Week 32- 36

Table 2: Time Frame of the Development

The development will span March 24, 2025, to November 1, 2025 (Weeks 1–31), targeting completion one month before the December 1, 2025. Using Agile (Scrum) with 7 sprints, the timeline starts Week 1 (March 24), slows in Weeks 5–6 (late April), pauses in Weeks 7–10 (May exams), and resumes Week 11 (June 2). Sectors include setup (Weeks 1–2), core modules I (Weeks 3–6), core modules II (Weeks 11–15), security/UI (Weeks 16–19), testing (Weeks 20–23), deployment (Weeks 24–27), and finalization (Weeks 28–31). This ensures a functional system by November 1, with Weeks 32–36 (November 2–December 1) for admin feedback, error fixes, and viva preparation, accommodating exam constraints.

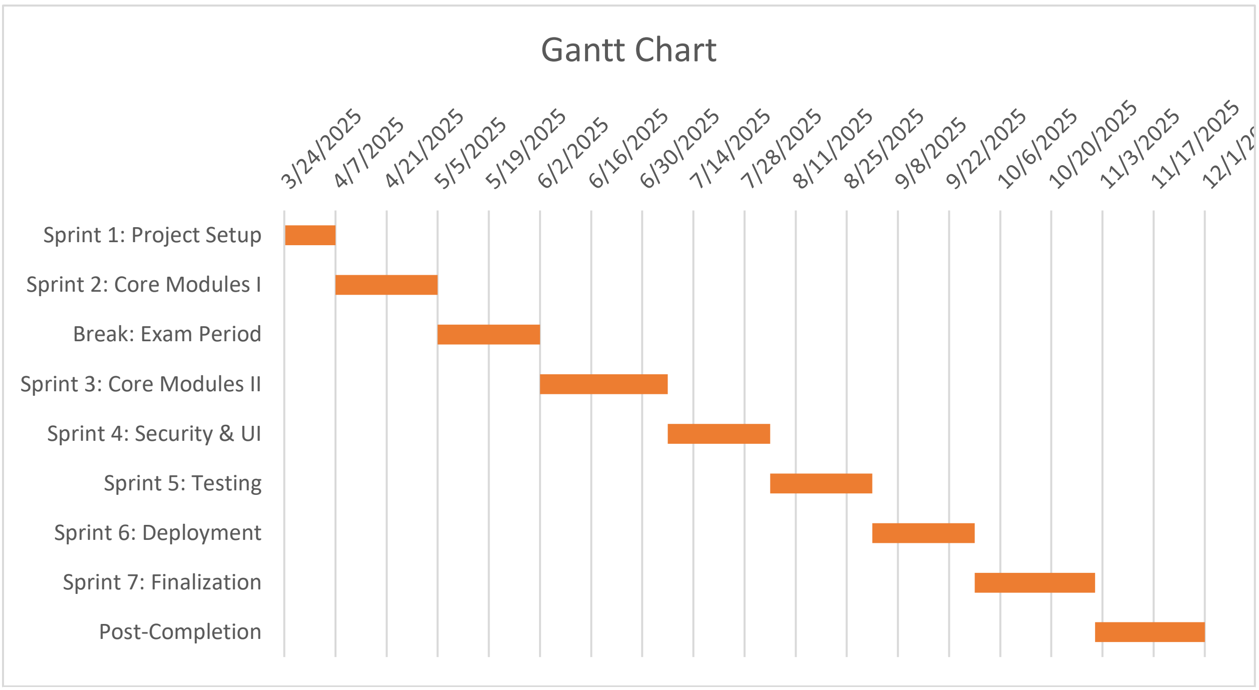

Figure 2: Gantt Chart

Chapter 5 Budget

This chapter outlines the estimated budget required for the development and implementation of the academic administration system. The budget covers expenses related to software development, testing, deployment, and other essential resources.

Development Costs for Frontend, Backend & Database – LKR 80,000

UI/UX Design Costs – LKR 35,000

Hosting and Infrastructure Costs – LKR 38,000

Development Tools & Software Licenses – LKR 30,000

Operational & Maintenance Costs – LKR 20,000

Total Estimated Budget – LKR 203,000

- **Development Costs for Frontend, Backend & Database**

  Development Costs covers frontend, backend, and database development, ensuring a functional and secure system for managing student registrations, payments, and lecturer records.

  Frontend Development (React.js) - LKR 30,000

  Backend Development (Node.js) - LKR 30,000

  Database Development (MySQL)- LKR 10,000

  Authentication & Security - LKR 10,000

- **UI/UX Design Costs**

UI/UX Design costs include wireframing, prototyping, and responsive design to create an intuitive and user-friendly interface.

Wireframes & Prototyping (Figma) - LKR 15,000

Responsive Design Implementation – LKR 10,000

Graphics & UI Components – LKR 10,000

- **Hosting and Infrastructure Costs**

  Hosting & infrastructure costs Accounts for cloud hosting, database server, domain registration, SSL security, and automated backups to maintain system availability and security.

  AWS Lightsail Hosting – LKR 10,000

  Database Server Hosting – LKR10,000

  Domain & SSL Certificate – LKR 8,000

  Backup & Storage Services – LKR 10,000

- **Development Tools & Software Licenses**

  Development tools & software licenses cover necessary frameworks, version control, and API testing tools for efficient software development.

  Development Frameworks & Libraries – LKR 15,000

  Version Control (Bitbucket/GitHub) - LKR 5,000

  API Development & Testing Tools – LKR 10,000

- **Operational & Maintenance Costs**

  Operational & maintenance costs ensure bug fixes, updates, and training support to keep the system running smoothly.

System Maintenance & Bug Fixes – LKR 10,000

Support & Training – LKR 10,000

Budget Conclusion

The proposed budget for the Institute Management System project ensures a cost-effective, scalable, and secure solution for Wiseway Academy’s administrative operations. By allocating funds to development, UI/UX design, hosting, tools, and maintenance, the system will provide a streamlined student registration, payment tracking, and lecturer management experience. The structured budget also includes contingency planning for future scalability and ensuring long-term sustainability. With a total estimated cost of 203,000 LKR, this investment will significantly enhance the efficiency and reliability of administrative processes at Wiseway Academy.

Chapter 6 Conclusion

The Academic Administration System is designed to improve the efficiency of administrative tasks in educational institutions by automating student registration, payment tracking, and lecturer management. By providing a centralized platform, the system ensures that administrators can easily register students with automatically generated unique IDs, track payments with daily updates and status tags, and manage lecturer details, including assigned classes. The dashboard offers a user-friendly interface, allowing quick access to key functions while maintaining security through a protected login system. The development follows a structured approach with five main phases. Documentation, Proposal, Analysis, Design, and Development & Implementation, ensuring systematic progress and timely completion before finalizing in December 2025. The Agile (Scrum) methodology allows flexibility in development, accommodating testing and improvements to create a reliable and secure system. With its structured design, cost-effective implementation, and focus on usability, the Academic Administration System will provide a modern and efficient solution for administrative management, helping institutions reduce errors, save time, and enhance overall workflow efficiency.

Chapter 7 References

[1] "Snatika," 2024. [Online]. Available: https://snatika.com/single-blog/importance-of-education-management-systems-for-educational-institutions?utm_source=chatgpt.com. [Accessed 16 March 2025].

[2] N. Barajas, "EduCase," 18 November 2024. [Online]. Available: https://er.educause.edu/articles/sponsored/2024/11/data-centralization-enhancing-decision-making-for-transformational-change?utm_source=chatgpt.com. [Accessed 16 March 2025].

[3] V. Sisouvong and K. Pasanchay, "Modern Educational Institution Management Strategies," *Journal of Education and Learning,* pp. 23-36, October 2024.

[4] A. Al-Hunaiyyan, R. Alhajri, B. Alghannam and A. Al-Shaher, "Student Information System: Investigating User Experience (UX)," *(IJACSA) International Journal of Advanced Computer Science and Applications,* pp. 80-87, 2021.

[5] S. Mukerjee, "Taylo & Francis- Online," 25 January 2012. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/1360080X.2012.642332?utm_source=chatgpt.com. [Accessed 20 3 2025].

[6] J. Walker, "Effective Education Solutions," 7 January 2016. [Online]. Available: https://www.effectiveeducation.org/2016/01/07/three-missing-features-in-most-student-information-systems-sis/?utm_source=chatgpt.com. [Accessed 20 3 2025].

[7] J. Phillips, "OpenSource," 3 January 2013. [Online]. Available: https://opensource.com/life/13/1/three-open-source-school-management-software-programs-teachers-and-student. [Accessed 3 March 2025].

Appendix

A. Glossary of Terms

- Smart Daily Payment Entry: Feature for manual payment input with automatic status updates (e.g., "Pending" to "Paid").

- JWT Authentication: Token-based security to restrict system access to the admin.
- Lean Scalable Design: System architecture designed for future growth with minimal reconfiguration.

B. Sample UI Mockup

- Payment Entry Page: Simple form with fields for Student ID (auto-suggest), Amount, Date, and a "Save" button; status updates displayed below (e.g., "Paid").

C. Database Schema (Simplified)

- Students: ID (PK, auto-increment), Name, Contact
- Payments: PaymentID (PK), StudentID (FK), Amount, Status, Date
- Lecturers: LecturerID (PK), Name, Contact
- Timetable: EntryID (PK), LecturerID (FK), Day, Time