Republic of the Philippines
Western Mindanao State University
COLLEGE OF COMPUTING STUDIES
DEPARTMENT OF INFORMATION TECHNOLOGY

STUDENT MANAGEMENT SYSTEM: ENHANCED GRADE TRACKING, ATTENDANCE MANAGEMENT AND ADVANCED LECTURE INTEGRATION.

A CAPSTONE PROJECT

Presented to the Faculty of the
College of Computing Studies
Western Mindanao State University

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Information Technology

By

ABRAJANO, JOHN PAULO O.
AQUINO, LEONARD M.
SESPEÑE, NIFLED JERICHO M.

VIN CZAR R. JAILANI
Faculty Adviser

DECEMBER 16, 2024

TABLE OF CONTENTS

Page

TITLE …… i

APPROVAL SHEET …….. ii

EXECUTIVE SUMMARY …….. iii

DEDICATION …… iv

ACKNOWLEDGEMENT ……... v

LIST OF TABLES ……... xi

LIST OF FIGURES …… xii

LIST OF APPENDICES …….. xiii

LIST OF ABBREVIATIONS ……. xiv

CHAPTER

1		**INTRODUCTION**	**2**
	1.1	Project Context	2
	1.2	Purpose and Description	2
	1.3	Project Objectives	3
		1.3.1 General Objectives	3
		1.3.2 Specific Objectives	4
	1.4	Scope and Limitations	4
	1.5	Significance of Study	4
	1.6	Definition of Terms	6
2		**REVIEW OF RELATED LITERATURE**	**7**
	2.1	Introduction	7
		2.1.1 Vevox	7
		2.1.2 MyAttendanceTracker	8
		2.1.3 Top Hat	8
		2.1.4 Accuclass	8
	2.2	Summary	10
	2.3	Synthesis	10
3		**TECHNICAL BACKGROUND**	**11**
	3.1	Conceptual Framework	11
	3.2	Software and/or Hardware Requirements	14

4 DESIGN AND METHODOLOGY 16

4.1 Introduction 16

4.2 Context Diagram 17

4.2.1 Context Diagram 17

4.2.2 Data Flow Diagram 18

4.2.3 Flowchart 19

4.2.3.1 Student Flowchart 19

4.2.3.2 Adviser Flowchart 20

4.2.3.3 Admin Flowchart 21

4..2.4 Use Case 22

4.3 Requirement Specification 24

4.3.1 Functional Requirements 25

4.3.1.1 Software Functionality 23

4.3.1.1.1 Grade Tracking 25

4.3.1.1.2 Attendance 25

4.3.1.1.3 Supplementary Lecture Materials 26

4.3.1.2 User Characteristics 26

4.3.2 Non-Functional Requirements 26

4.3.2.1 Technical Requirements 26

4.3.2.2 Performance Requirements 27

4.3.2.3 Assumptions and Dependencies 27

4.3.2.4 Security Requirements 28

4.4 System Design 28

4.4.1 Database Design 29

4.4.2 Entity Relationship Diagram 30

4.4.3 Database Fields 31

4.4.4 User Interface Design 40

5 DEVELOPMENT AND TESTING 72

5.1 Development 72

5.2 Integration of Technology 72

5.3 Testing 73

5.3.1 Alpha Testing 74

5.3.2 Beta Testing 87

5.3.3 User Acceptance Testing 96

5.4 Implementation Plan 96

5.4.1 Infrastructure/Deployment 97

5.4.2 Processes/Policies/Personnel 97

6 RESULT AND DISCUSSIONS 98

6.1 Introduction 98

6.2 Summary of Findings 98

6.3 Testing Result 98

6.3.1 Alpha Testing Result 99

6.3.2 Integration System Result 104

7 CONCLUSION AND RECOMMENDATIONS 111

7.1 Conclusions 111

7.2 Recommendations 111

REFERENCES 113
APPENDIX A: DATA GATHERING **115**
APPENDIX B: SOURCE CODE **116**
APPENDIX C: TESTING **134**
APPENDIX D: EVALUATION FORM **137**
APPENDIX E: GANTT CHART **141**

LIST OF TABLES

Table		Page
1	Definition of Terms	6
2	Review of Related Works and System Comparison	9
3	TABLE Activities	31
4	TABLE Activity_Attachement	31
5	TABLE Activity_Submission	31
6	TABLE Admin	31
7	TABLE Admin_auto_notifications	32
8	TABLE Admin_notes	32
9	TABLE Admin_notifications	32
10	TABLE Admin_reminders	32
11	TABLE Archived_semesters	32
12	TABLE Attendance	32
13	TABLE Classes	33
14	TABLE Classes_meetings	33
15	TABLE Classes_students	33
16	TABLE Course_requirements_laboratory	34
17	TABLE Course_requirements_lecture	34
18	TABLE Current_semester	34
19	TABLE Laboratory_rubrics	34
20	TABLE learning_resources	34
21	TABLE Lectures	35
22	TABLE Lecture_rubrics	35
23	TABLE Messages	35
24	TABLE Rubrics	35
25	TABLE Rubric_requirements	35
26	TABLE class_ semester	36
27	TABLE Specific_rubrics	36
28	TABLE Staff_accounts	36
29	TABLE Staff_advising	36
30	TABLE Staff_notification	37
31	TABLE Students	37
32	TABLE Student_enrollment	37
33	TABLE Student_grades	37

34 TABLE Student_info 38
35 TABLE Student_notification 38
36 TABLE Student_pictures 38
37 TABLE Subject 38
38 TABLE Subject_meetings 38
39 TABLE Subject_schedules 39
40 TABLE Teacher_notes 39
41 TABLE Teacher_reminders 39

LIST OF FIGURES

Figure		Page
1	Conceptual Framework	11
2	Context Diagram	17
2.1	Data Flow Diagram	18
2.2	Student Flowchart	19
2.3	Adviser Flowchart	20
2.4	Admin Flowchart	23
3	Use Case	22
4	Database Design	28
4.1	Entity Relationship Diagram	30
5	Home Page View	40
5.1	Admin Dashboard	42
5.2	Create Semester	44
5.3	Semester Management	42
5.4	Create a Subject	46
5.5	Creating Staff Accounts	47
5.6	Creating a Class	48
5.7	staff landing page	50
5.8	teacher dashboard	51
5.9	Class management	52
6	Create new class	53
6.1	View classes	54
6.2	View Attendance Record	55
6.3	Setting the grades in rubrics	56
6.4	People Section	57
6.5	Setting Activities	58
6.6	Scores	59
6.7	Grades	60
6.8	Learning Resources	61
6.9	Staff Information	62
7	Student Login Page/Creation	63
7.1	Student Dashboard	64
7.2	Activity Section	65
7.3	Grades Section	67

7.4 Lecture Materials Resources 67
7.5 Attendance 68
7.6 Activities 69
7.7 Grades 70
7.8 Communication Tool 70
7.9 Notification Tool 71

LIST OF APPENDICES

Appendix	Page
Appendix A: Data Gathering	115
Appendix B: Source Code	116
Appendix C: Testing	134
Appendix D: Evaluation Form	137
Appendix E: Gantt Chart	141

CHAPTER 1

INTRODUCTION

1.1 Project Context

In today's educational landscape, students often find themselves juggling numerous responsibilities. They are not only attending multiple classes across different subjects but also participating in various organizations and extracurricular activities. For instance, a student might be involved in a student council, a debate club, and a sports team, all while taking a full course load. This intense schedule leaves them exhausted and struggling to keep up with the demands of their academic and extracurricular commitments.

Amidst this chaos, keeping track of grades can feel like an overwhelming task. Many students spend precious time manually calculating their grades, which detracts from their focus on learning and personal growth. This is especially challenging when students have to compute grades for multiple subjects, each with its own set of quizzes, assignments, and exams. The lack of a streamlined, automated system means students often rely on various methods to track their academic progress, leading to inconsistencies and errors.

Additionally, the frequent shuffling of class schedules and extracurricular commitments makes it difficult for students to maintain an organized overview of their academic performance. Without a reliable reference or validation system, students are sometimes unsure if their grades are computed correctly. This uncertainty can lead to anxiety and a reluctance to check their grades on the portal, further hindering their ability to manage their academic responsibilities effectively.

Recognizing these challenges, the proposed Student Management System aims to address these issues by providing an integrated, user-friendly platform that simplifies grade tracking and academic management. This system will ensure students can easily access accurate, up-to-date information about their grades, attendance, and supplementary materials, thereby reducing their administrative burden and empowering them to focus on their academic and personal growth.

1.2 Purpose and Description

The purpose of this project is to develop a comprehensive Student Management System designed to address the challenges faced by students in managing their academic responsibilities amidst their busy schedules. The proposed system aims to simplify grade tracking and enhance overall academic management, thereby reducing the administrative burden on students and allowing them to focus on learning and personal growth.

1.3 Project Objectives

The primary objectives of the proposed Student Management System are to create an integrated platform that simplifies academic management and supports student success.

This section presents the general objective and the specific objectives of the study.

1.3.1 General Objective

1. Develop a feature for real-time computation and storage of grades from quizzes, assignments, and exams.
2. Ensure easy access to grades for students.
3. Implement an attendance feature to promote accountability and participation.
4. Provide a user-friendly interface for viewing and managing attendance records.
5. Offer a centralized location for supplementary lecture materials.
6. Ensure continuous access to essential learning resources.
7. Design an intuitive and user-friendly interface for easy navigation.
8. Facilitate efficient management of academic information.
9. Consolidate grades, attendance, and supplementary materials into one accessible platform.
10. Enable students to maintain an organized overview of their academic progress.
11. Reduce administrative burdens on students.
12. Free up time and energy for learning, growth, and personal development.

1.3.2 Specific Objectives

1. Develop an Automated Grade Tracking Feature
2. Implement a Digital Attendance Tracking Feature
3. Create a Centralized Repository for Supplementary Lecture Materials
4. Provide a Comprehensive Academic Overview
5. Streamline the Academic Management Process

1.4 Scope and Limitations

The proposed Student Management System will encompass automated grade tracking, digital attendance management, a centralized repository for supplementary materials, and a user-friendly interface, all aimed at enhancing the academic management experience for students. The system will automatically compute and store grades, provide real-time access to academic performance, accurately record attendance, and offer an intuitive platform for viewing and managing both grades and attendance. Additionally, it will include a centralized repository for lecture notes and resources, ensuring continuous and easy access to essential learning materials.

However, the system faces several limitations, such as potential challenges in integrating with existing school systems, dependence on accurate data entry by instructors, and the need for reliable internet access and compatible devices. Security and privacy concerns will require robust measures to protect sensitive student data, and successful implementation will depend on adequate training and user adoption. Furthermore, ongoing maintenance and technical support will be necessary to address any issues and ensure the system remains functional and up-to-date, which will involve additional costs and resources

1.5 Significance of the Study

The development of the proposed Student Management System is poised to significantly enhance the academic experiences of students and improve the administrative efficiency of educational institutions. By automating grade tracking and providing real-time access to academic performance, the system will alleviate the burden of manual grade computations, reducing errors and freeing up valuable time for students to focus on learning and personal development. Additionally, the digital attendance management feature will promote accountability and

participation, enabling students to easily track their attendance records and ensure consistent class attendance.

Moreover, the centralized repository for supplementary materials will offer continuous access to essential learning resources, supporting students in their academic pursuits and fostering a more organized and effective study environment. This integration ensures that students have all necessary materials readily available, enhancing their ability to study and prepare for classes. The user-friendly interface of the system will further improve its usability and adoption, allowing students to navigate their academic responsibilities with greater ease and confidence. This will lead to a more streamlined and less stressful academic experience.

For educational institutions, the system will streamline administrative tasks, improve data accuracy, and facilitate better communication between students and faculty. By integrating various academic management functions into a single platform, the system will lead to more efficient operations and a more cohesive academic experience. Ultimately, this study aims to contribute to the development of a more supportive and efficient educational environment, fostering a culture of academic success and personal growth. By addressing the current challenges faced by students in managing their academic responsibilities, the proposed system will play a crucial role in enhancing the quality of education and supporting students in achieving their educational goals.

1.6 Definition of Terms

Student Management System - A software platformed designed to help students and educational institutions manage academic information such grades, attendance, and supplementary materials.

Automated Grade Tracking - A feature within the SMS that calculates and records student grades automatically from quizzes, assignments, exams, ensuring accuracy and providing real-time access. Digital Attendance Management - A system feature that tracks and records student attendance electronically, allowing students and instructor to monitor attendance records efficiently.

Digital Attendance Management - system feature that tracks and records student attendance electronically, allowing students and instructor to monitor attendance records efficiently.

Supplementary Materials Repository - A centralized digital storage within the SMS where lecture notes, readings, and additional learning resources are uploaded and accessed by students.

User-Friendly Interface - The capability of SMS to provide immediate and up-to-date information regarding grades, attendance, and other academic data.

Administrative Burden - The time and effort required by the students and instructors to manage academic records and task manually.

Data Accuracy - The precision and correctness of information recorded and displayed within the SMS, crucial for reliable academic tracking.

CHAPTER 2

REVIEW OF RELATED LITERATURE

2.1 Introduction

The review of related literature and systems focused on improving Student Management Systems (SMS) offers a comprehensive understanding of existing academic tools, their functionalities, and the technological advancements that have shaped this domain. This review identifies key features and methodologies within comparable systems, highlighting their strengths and limitations to inform the development of the proposed SMS. By examining prior studies and innovations, we gain valuable insights into the academic management system landscape, allowing us to pinpoint opportunities for refinement and innovation.

The need for efficient, scalable, and cost-effective student management systems has never been greater as educational institutions strive to optimize their operations and enhance the student experience. The advent of technologies such as QR-based attendance tracking, real-time data analytics, and modular dashboards has unlocked new possibilities for developing advanced solutions that address institutional needs. These tools are lauded for their affordability, versatility, and ease of implementation, making them integral to modernizing academic processes. The following literature and systems exhibit comparable structures and thematic relevance.

The following literature and systems demonstrate similar structures and thematic relevance,

2.1.1 Vevox

Vevox is a web-based platform designed to enhance educator-student interaction, focusing on attendance tracking and student engagement. Rated #1 for ease of use and support, Vevox seamlessly integrates with existing Learning Management Systems (LMS), making it a user- friendly solution for both educators and institutions.

- Attendance Management
- Engagement Tools
- LMS Integration
- User-Friendly Interface

Vevox stands out for its seamless integration with Learning Management Systems (LMS), providing

an innovative solution that not only tracks attendance but also enhances student engagement through interactive tools, making it highly adaptable to modern classroom needs. Ahmed et al. (2023)

2.1.2 MyAttendanceTracker

MyAttendanceTracker is an online tool that enables educators to easily log and monitor student attendance with a simple interface. Known for its flexibility, it also allows tracking of grades and notes, providing a comprehensive solution beyond just attendance.

- Attendance Grade Tracking
- Online Access

As noted by Silva and Peterson (2023), MyAttendanceTracker is an exemplary tool for educators due to its flexibility in managing attendance and grades, offering features like exportable reports and real-time updates that cater to dynamic classroom environments. Silva, P., & Peterson, L. (2023)

2.1.3 Top Hat

Is a comprehensive education platform designed to enhance the teaching and learning experience. It seamlessly integrates various teaching tools with efficient attendance tracking, making it an ideal choice for educators seeking to streamline their educational processes. The platform supports automated grading and assignment distribution, allowing instructors to focus more on teaching and less on administrative tasks. Additionally, Top Hat is compatible with existing Learning Management Systems (LMS), ensuring easy integration into your current teaching ecosystem.

- Automated Grading
- Comprehensive Attendance Tracking

Jones et al. (2024) highlight Top Hat as a transformative platform in education, emphasizing its ability to integrate attendance tracking, automated grading, and interactive teaching tools, which significantly reduce administrative workload and foster a more engaging learning environment."
Jones, K., Adams, S., & Lee, T. (2024).

2.1.4 Accuclass

Accuclass is an advanced attendance tracking software designed to deliver precise and comprehensive attendance data for educational institutions. It offers real-time class sign-ins and sign-out capabilities, allowing for accurate tracking of student attendance. With its focus on analytics,

Accuclass empowers educators to identify attendance trends and patterns, facilitating informed decisions for academic planning and intervention strategies.

- Real-Time Class Sign -Ins with Sign-Out Options
- Customizable Attendance Statuses

According to Fernandez and Yoon (2024), Accuclass offers unparalleled precision in real-time attendance tracking, coupled with advanced analytics that empower educators to identify patterns and make data-driven decisions for academic interventions "Fernandez, R., & Yoon, J. (2024.

Table 2 : Review of Related Works and System Comparison Table

Feature	Vevox	MyAttendanceTracker	Top Hat	Acccuclass	Student Management System
Grade Tracking		✓	✓	✓	✓
Automated Grade Computation	✓	✓	✓		✓
Supplementary Lecture Materials Attachement			✓		✓
Attendance Management	✓	✓	✓	✓	✓
User-Friendly Interface	✓	✓	✓	✓	✓
Notifications					✓
Online Access for Instructor and Students	✓	✓	✓	✓	✓
Real-Time Sign-in and Sign-out	✓	✓	✓	✓	✓
QR Code for Attendance					✓

The table show most of the modules that the system will be having has similar modules with the three related literature that was presented in this project. Furthermore, even though there are more similarities still the system to be develop has some unique features.

2.2 Summary

Student Management System is a web-based system aims to creates an integrated platform that simplifies academic management and supports student access. It includes features such as automated grade tracking, stores and compute grades, digital attendance monitoring, and a repository for supplementary lecture materials, ensuring continuous access to essential learning resources. Vevox is a top-rated web-based system that enhances educator-student interaction and simplifies attendance tracking, seamlessly integrating with existing LMS and offering features to boost student engagement. MyAttendanceTracker is an online tool known for its simple interface and flexibility, allowing easy logging of attendance, grades, and notes, with customizable, exportable reports. Top Hat is a holistic education platform with integrated teaching tools, automated grading, and efficient attendance tracking, compatible with existing MS. Accuclass provides accurate, real-time attendance data with customizable statuses and robust analytics to help educators understand attendance trends and improve academic planning.

2.3 Synthesis

The review of related highlights advanced applications enhancing academic management and advising through automation and sophisticated data management. The Student Management System integrates advanced features from leading academic tools to enhance academic management and student support. It includes automated grade tracking, digital attendance monitoring, and a repository for supplementary materials, ensuring continuous access to essential resources. Similar to Vevox, it boosts engagement and integrates with LMS for seamless attendance tracking. It adopts MyAttendanceTracker's flexibility for logging attendance, grades, and notes with exportable reports. Top Hat's holistic approach and Accuclass's precise attendance data and analytics are also incorporated. This system provides a user-friendly platform that simplifies academic management, promotes accountability, and supports student success.

CHAPTER 3

TECHNICAL BACKGROUND

3.1Conceptual Framework

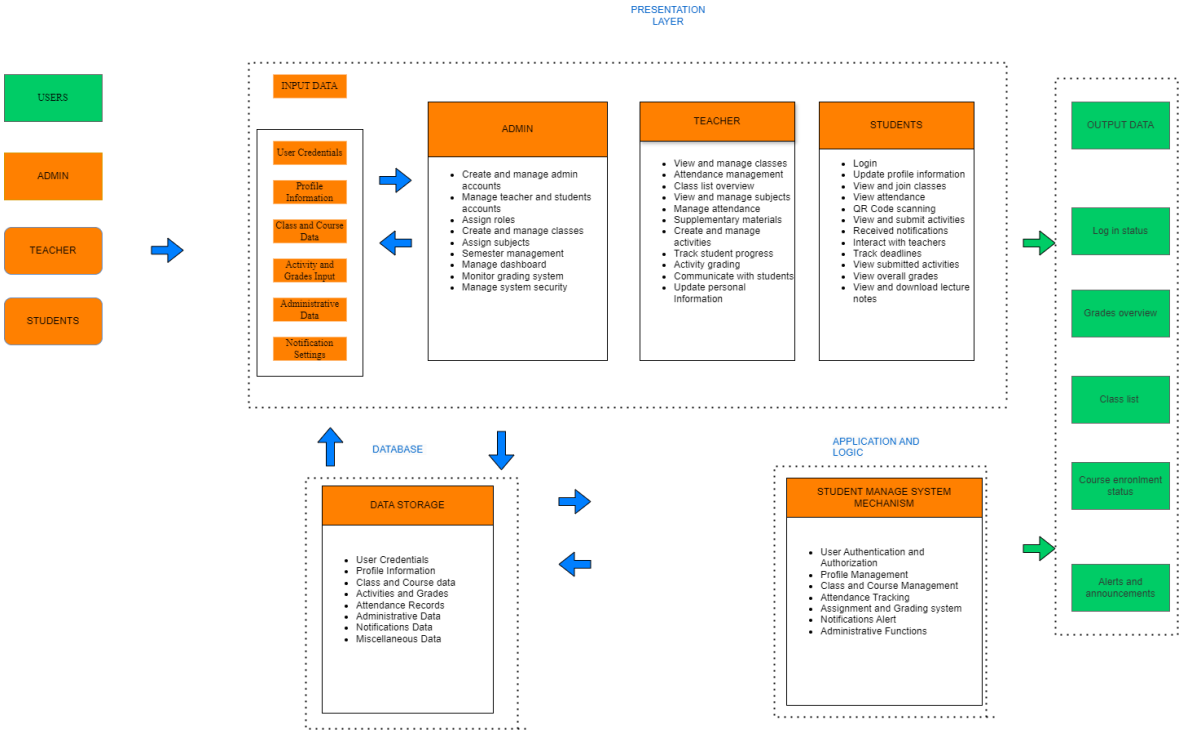


Figure 1: Conceptual framework

The Student Management System (SMS) is designed to cater to three primary user roles: administrators, teachers, and students. Each user role interacts with the system via a user-friendly interface, which can be accessed through web browsers, providing real-time access to key academic and administrative features.

The main interface for thorough academic administration and evaluation is the online application. It gives the teacher the ability to enter curriculum details, control subjects and classes, and monitor each student's overall academic performance. The student has the ability to view their entire academic record.

Additionally, this interface manages the vital task of academic evaluation, processing students' attendance records, grades, and academic records to support their academic work.

This feature helps the students by giving them precise, data-driven information so they may make wise decisions and concentrate on their own development.

The system provides a reliable and scalable solution by utilizing a number of contemporary technologies. The website application, developed using PHP ensures compatibility and optimal performance on various devices, enabling students to easily view their automated grades, access lectures and laboratory materials, and upload their completed tasks, activities, and assignments.The web application or online application's responsive and user-friendly interface is made possible by the usage of HTML , CSS, and JavaScript in its construction. PHP is utilized for the web server, ensuring efficient handling of server-side operations, while MySQL serves as the database management system, offering reliable data storage and retrieval capabilities. This technological stack guarantees that the application is not only functional and efficient but also safe and scalable, ready to meet the demands of a big user base. PHP is used for server-side activities and MySQL is used for database management.

The Frameworks divided into three main layers: Presentation, Application and Logic, and Database.

Presentation Layer

layer includes user interface for students, students and administrators. The primary goal of this layer is to ensure that users can interact with the system easily:

Admin Interface: Admins can create and manage user accounts (both teachers and students), assign roles, manage subject and courses, and oversee the grading system. It also allows the admins to manage semester schedules, system security, and the overall dashboard for effective academic monitoring.

Teacher Interface: Teacher manage classes and attendance, view class list, handle grading, and create supplementary materials or activities for students. Teachers can also communicate with students and track their academic progress.

Student Interface: Student can log in, update personal information, join classes, view attendance track their academic progress, and submit assignments.

Application Logic Tier

The Application ang logic layer handles core business rules and system logic. This layer ensures that user input is processed efficiently, permissions are checked, and appropriate actions are

performed based on user roles.

Student Management System mechanism is to streamline and automate the academic management processes within an educational institution. The system is designed to simplify tasks for students, teachers, and administrators by providing an integrated platform for managing grades, attendance, and academic materials.

Data Validation and Security: This component ensures that all data inputted by users is validated to prevent errors and security vulnerabilities such as SQL injection. User credentials are encrypted, and access control is enforced based on user roles to maintain data security and integrity.

Database Layer

This layer utilizes a MySQL database to handle and organize various types of information, ensuring structured and efficient data retrieval. Key data types stored include user credentials, which authenticate and authorize admins, teachers, and students, as well as profile information that contains personal details of all users.

3.2 Software and Hardware Requirements

Software Requirements

1. Operating System
   Server: A reliable operating system such as Ubuntu Linux Server or Windows Server is required to host the web server and database. Client: The application should be accessible via modern web browsers (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge) on Windows, macOS, or Linux. For mobile devices, the Android operating system is necessary.

2. Web Server:
   Apache HTTP Server: Used to run server-side application. Because of its strong design, wide module support, and customizable configuration choices, Apache HTTP Server excels in serving a wide range of web content and effectively managing multiple concurrent client connections.

3. Database Management System:

MySQL: Chosen for its reliability and scalability, MySQL will manage the data storage needs, ensuring efficient data retrieval and manipulation.

4. Development Tools:
   Web Development Tools: Visual Studio Code, XAMMP

5. Programming Languages and Libraries:
   HTML, CSS, Javascript: For front-end development.
   Apache HTTP Server, PHP, and MySQL for the web server and for Backend services.

6. Version Control:
   Git: For coordinating developer work and source code management.

Hardware Requirements

1. Server Hardware:
   Processor: A multi-core processor (e.g., Intel Xeon or AMD Ryzen) to handle concurrent user requests efficiently.
   Memory: At least 16GB of RAM to support multiple operations and database transactions. Storage: Solid-State Drives (SSD) with a minimum of 500GB for fast read/write operations and sufficient data storage capacity.

2. Client Devices:
   Desktop/Laptop: Users accessing the web application will need a computer with at least 4GB of RAM, a modern multi-core processor, and an up-to-date web browser. Mobile Devices: Android smartphones with at least Android 6.0 (Marshmallow), 2GB of RAM, and a functioning camera for QR code scanning.

3. Networking:
   Internet Connection: A stable and high-speed internet connection is essential for accessing the cloud-hosted application and ensuring smooth data transmission between clients and servers.

CHAPTER 4

DESIGN AND METHODOLOGY

4.1 Introduction

The design and methodology of the Student Management System (SMS) are focused on developing an efficient, user-friendly platform that addresses the academic management needs of students, teachers, and administrators. The design process follows a systematic approach, ensuring that the system incorporates the necessary features, such as grade tracking, attendance management, communication, and notification. The methodology follows standard software development practices to ensure the system is reliable, scalable, and secure.

The SMS is structured into three primary layers: the presentation layer, which includes user interfaces for all stakeholders; the application and logic layer, which processes user input and manages business rules; and the database layer, which securely stores user data and academic records. The use of modern web technologies like HTML, CSS, and JavaScript ensures a responsive interface, while PHP and MySQL provide a robust backend infrastructure for managing data and system operations efficiently.

Throughout the development process, the system design is guided by both functional and non-functional requirements, ensuring that it meets user expectations for usability and performance. The methodology includes user requirement analysis, system design, iterative development, testing, and evaluation. This approach guarantees that the system is not only functional but also adaptable to future needs, ensuring continuous improvement and scalability.

4.2 Context Diagram

4.2.1 Context Diagram

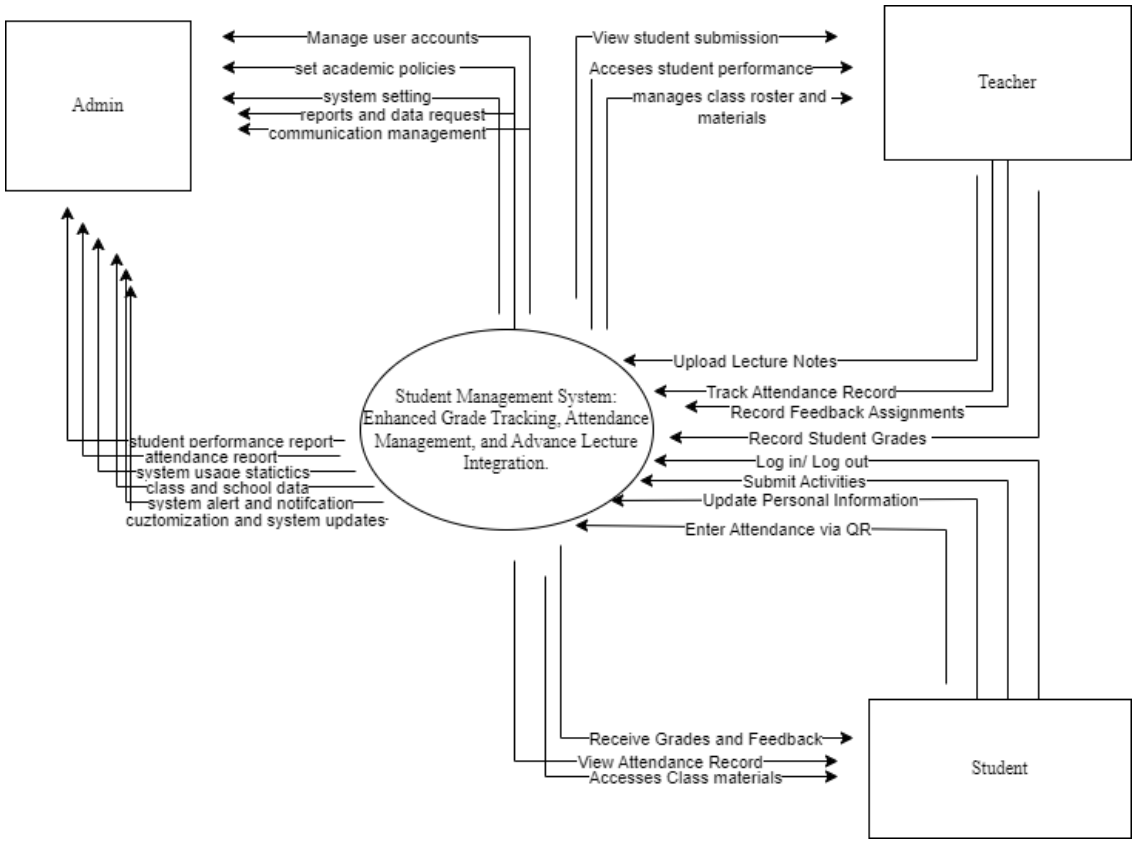


Figure 2: Context Diagram

The Student Management System (SMS) is a comprehensive platform designed to streamline academic management and enhance interactions between students, teachers, and administrators. The system supports a seamless, efficient process for managing academic records, attendance, grades, and class materials, providing users with all necessary tools to support learning and teaching.

Students are central users of the system. They can create an account or log into an existing account, which enables them to access personalized academic information. Students can view their grades, class schedules, and attendance records.

Additionally, the system allows students to submit assignments and view feedback on their submissions. Students are also able to download lecture materials and submit any activities or homework required by their teachers.

The SMS tracks students' performance across various subjects, allowing them to monitor their academic progress over time. They receive notifications regarding upcoming deadlines, class cancellations, or updates related to their courses. Furthermore, students can view their overall attendance and grades through an easy-to-navigate dashboard.

Teachers have a more dynamic role in the system. They can log in, manage their profiles, and handle class records such as attendance and grades. Teachers are responsible for entering grades for assignments, quizzes, and exams into the system. They can upload lecture notes, assignments, and activities, which are then accessible to students.

Teachers also manage attendance, marking students as present or absent and tracking trends in class participation.

Additionally, teachers are able to review and provide feedback on students' submitted assignments, ensuring timely and comprehensive assessments. Their dashboard allows them to manage multiple classes, view class performance summaries, and track overall student progress.

Administrators oversee the entire system. They manage the accounts of teachers, creating, modifying, or deactivating user accounts as needed. Administrators can also define grading scales, assign teachers to courses, and schedule class times for the academic year.

Administrators have access to a variety of reports, such as student performance summaries, attendance statistics, and system usage metrics. These reports help administrators make informed decisions about student support, teacher workloads, and overall system health. Administrators are also responsible for managing system settings, including data security, permissions, and access levels, ensuring the integrity and smooth functioning of the SMS.

4.2.2 Data Flow Diagram

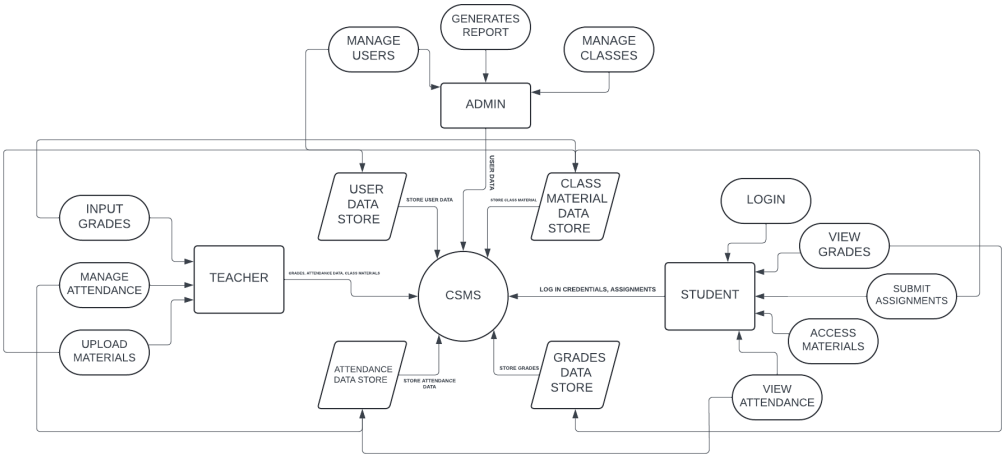


Figure 2.1: Dataflow Diagram

The figure above shows the inner workings of the system. It illustrates how different parts of the system work together in one main process, giving a clear picture of how the system functions internally.

4.2.3 Flowchart

4.2.3.1 Student Flowchart

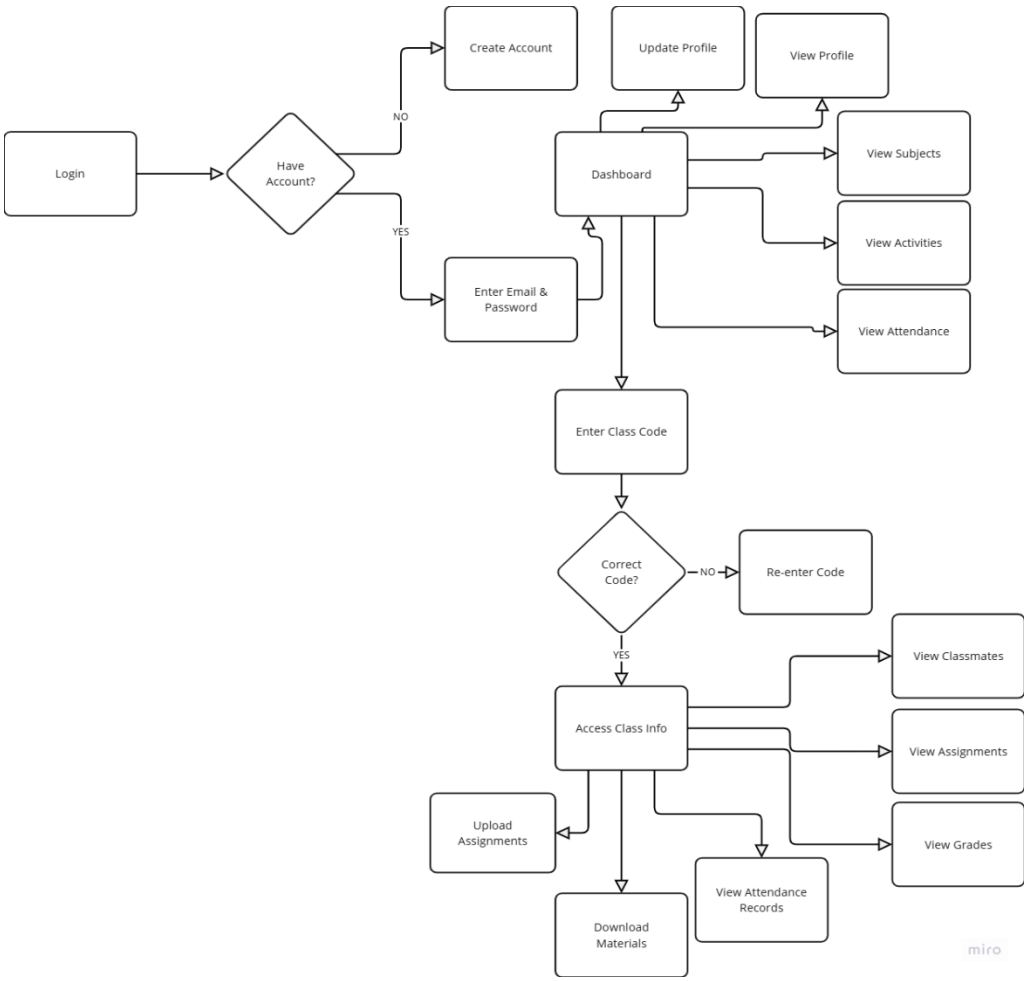


Figure 2.2: Student Flowchart

In this flowchart when a student logs in to the system, they begin by entering their email and password. If they don't have an account yet, they can create one. Once logged in, the system recognizes them as a student and presents different functionalities. The student can update their profile information, such as changing their course, year, or even their password. They can also view their profile anytime and make changes when needed.

In the Dashboard, the student can see their subjects, any pending or missed activities, and details about their attendance. To access a specific class, they need to enter a Class Code. If the code is correct, they

are given access to important class information, like who else is in the class, the assignments they need to complete, their grades, attendance records, and any lecture materials that can be downloaded. The system also allows students to upload assignments, view their grades, and keep track of their attendance. This system is designed to help students manage their academic tasks, and class-related information easily.

4.2.3.2 Adviser Flowchart

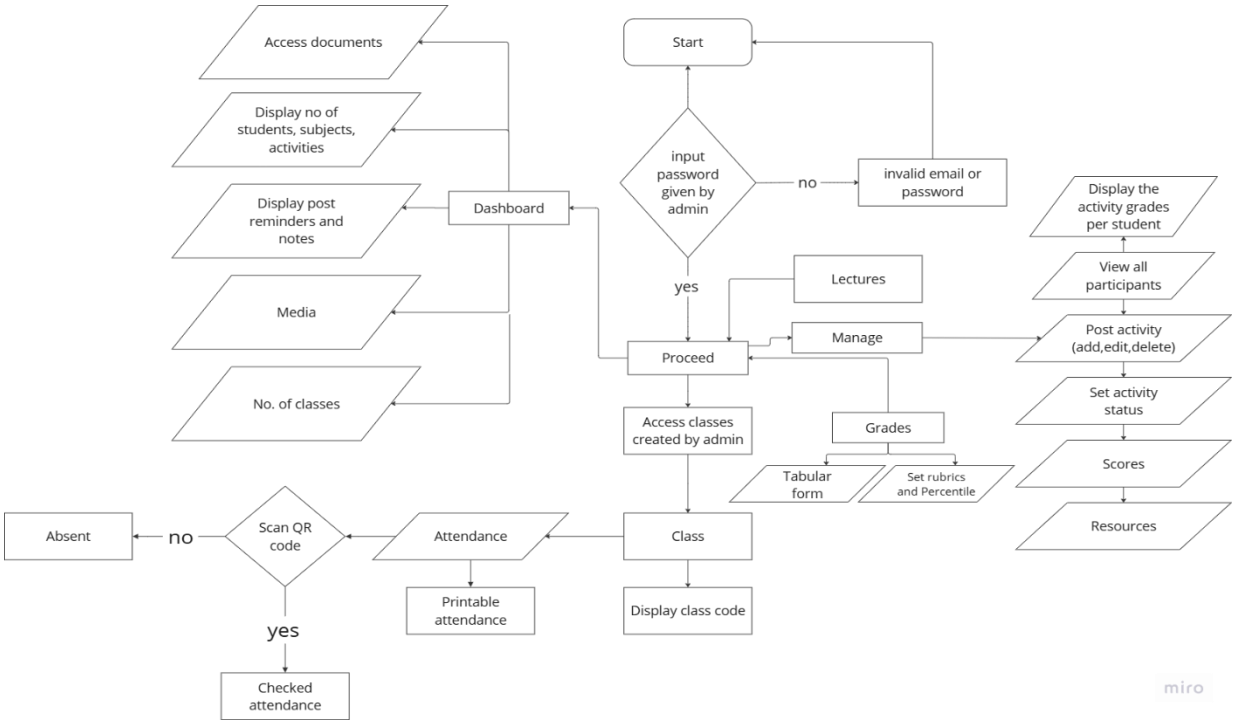


Figure 2.3: Adviser Flowchart

The diagram illustrates the workflow of a teacher management system, starting with a login process where the teacher enters credentials provided by the admin. If the login is successful, the user is directed to a dashboard displaying information like the number of students, subjects, and activities, along with options to post reminders and access subjects quickly. From the dashboard, the teacher can proceed to manage various classroom tasks, including accessing subjects and classes created by the admin, viewing participants, and managing activities (such as posting, editing, or deleting them). The system also allows teachers to monitor student progress by displaying activity statuses and overall grades. In addition, the teacher can manage class operations through features like attendance, where they can scan QR codes to mark student presence, print attendance records, and note absences. The system also includes tools for creating and editing rubrics, managing grading criteria, and uploading or downloading files related to lectures. Lastly,

the teacher can access printable and editable worksheets, ensuring seamless management of class materials and student progress.

4.2.3.3 Admin Flowchart

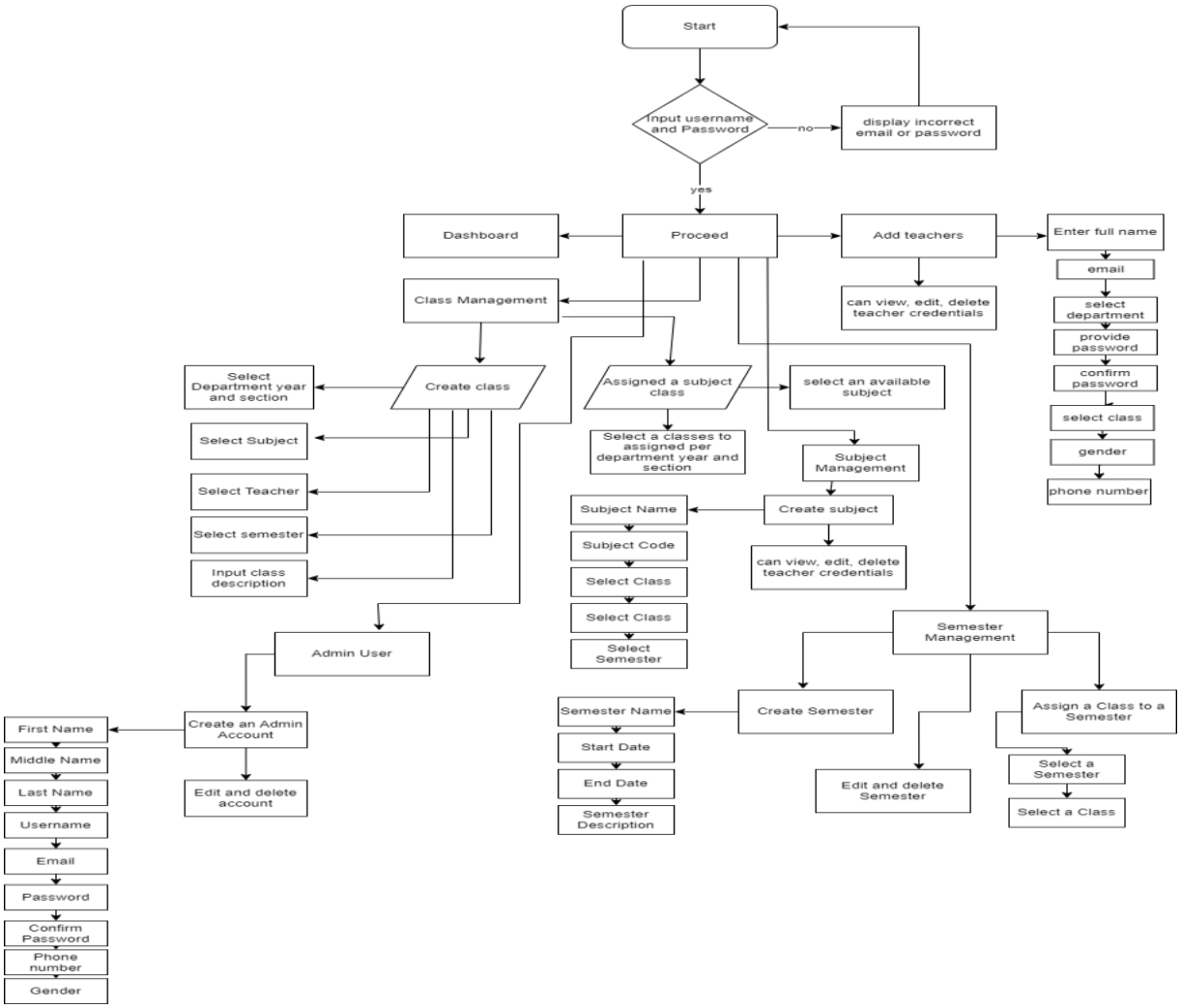


Figure 2.4: Admin Flowchart

It begins with a login process where the user inputs their username and password provided by the programmer, incorrect credentials result in an error message, while correct ones grant access to the dashboard. From the dashboard, the admin can add teachers by inputting their personal information and assigning them to departments, with the ability to view, edit, or delete teacher profiles. In the class management section, the administrator can create classes by selecting a department, year, section, subject, teacher, and semester, along with providing a class description. There's also a subject management section where new subjects can be created and linked to classes. Semester management allows the creation and editing of semesters by specifying details

like the semester name, start and end dates, and class assignments.

Additionally, the admin user management feature enables the creation and editing of admin accounts with personal details such as name, email, and password. This entire system facilitates structured control over various school operations.

4.2.4 Use Case

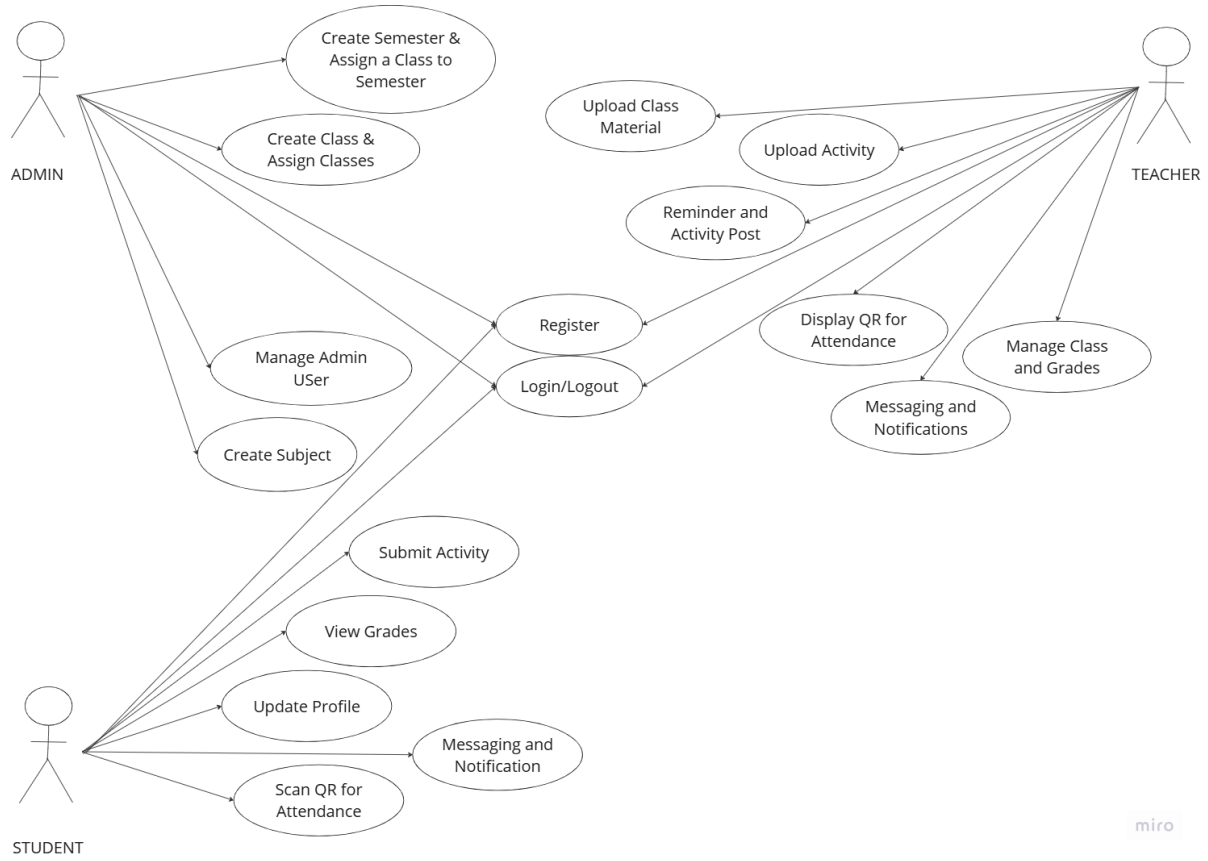


Figure 3: Use Case

Use Case #1: Log-in

User: Administrator, Teacher and Student

Description: The system must permit user to access the system.

Fit Criterion: The user must input the given username/email and password, and it should match the credentials to access the assigned module after log-in.

Use case scripts:

1. The log-in form is displayed.
2. The user enters a valid username/email and password
3. Upon clicking the “log-in” button, the system checks the entered credentials.
4. If valid, the user is granted access, displaying the message: “Login Succesful!”
5. If invalid, the system returns: “Account login error!”.

Use Case #2: Register Account

User: Administrator, Teacher and Student

Description: This allows the users to create an account in the system by providing necessary information.

Fit Criterion: The system verifies that all required fields are filled and that the username/email is unique before allowing registration.

Use case scripts:

1. The registration form is displayed.
2. The user inputs personal details
3. The system validates the inputs, checking for completeness and uniqueness of the username.
4. If all checks pass, the system registers the user and display: “Admin Addition successful!”
5. If there are errors like duplicate username, the system displays an appropriate error message.

Use Case #3: Upload Class Material User: Teacher

Description: This allows teachers to upload educational materials for their assigned classes.

Fit Criterion: The uploaded files should be linked to the correct class and made available to the students in that class

Use case scripts:

1. The teacher navigates to the “Upload Material” section.
2. The teacher selects the class and uploads relevant files.
3. The system verifies the file type and uploads it to the server
4. If successful, a message displays: “Material uploaded successfully.”
5. If an error occurs, the system display an error message and prompts the user to try again.

Use Case #4: Submit Activity User: Student

Description: This allows students to submit activities uploaded by the teacher.

Fit Criterion: The submitted activity is saved in the system and linked to the correct activity and student

Use case scripts:

1. The student navigates to the "Submit Activity" section.
2. The student selects the assignment and uploads the completed work.
3. The system checks the file type and saves the submission.
4. If successful, a message displays: "Activity Submitted Successfully"
5. If the upload fails, the system provides an error message.

Use Case #5: View Grades User: Student

Description: This allows students to view their grades for various assignments and overall class performance.

Fit Criterion: The system retrieves the student's grades from the database and displays them in an easy-to-read format.

Use case scripts:

1. The student selects the "Grade" option.
2. The system fetches the grades for the current semester from the database.
3. The grades are displayed, showing individual activity grades and overall scores.
4. If there are no grades available, the system displays a message: "Grades Not Yet Available."

4.3 Requirements Specification

The Requirements Specification for the Student Management System (SMS) outlines both the functional and non-functional requirements that the system must meet to fulfill its intended purpose. This document provides a detailed guide for developers, stakeholders, and project managers to ensure the system's features are well defined. The functional requirements detail the system's capabilities, including user account management, grade tracking, attendance

management, and content sharing between students and teachers. The non-functional requirements specify the system's performance criteria, security protocols, and usability standards to ensure reliable an

4.3.1 Functional Requirements

Functional requirements define the specific behavior, functions, and capabilities that the system must possess. These requirements ensure that the system performs the tasks needed to support research management and academic collaboration effectively. Key functional requirements for the system include:

4.3.1.1 Software Functionality

This section describes the modules that the Student Management System uses to meet the goals and objectives of developing the application.

4.3.1.1.1 Grade Tracking

This feature ensures that access to the system's functionalities is restricted based on the user's role (student, or teacher), with additional support for grading rubrics and editable criteria.

- Student Role: Can view their grades for assignments, quizzes, exams, and overall performance, including a breakdown of how their grades were determined based on the teacher's grading rubrics and criteria.
- Teacher Role: Can input, update, and manage student grades, create and customize grading rubrics with editable criteria for different assignments and assessments. Teachers can also adjust rubrics as needed and apply them consistently across multiple classes or students.

4.3.1.1.2 Attendance

This feature ensures that access to the system's functionalities is restricted based on the user's role (student or teacher), with the use of QR codes for attendance tracking.

- Student Role: Can access a unique QR code generated upon account creation, which is used for attendance. Students can view their attendance records to track their presence in each class. Attendance records can be monitored for accuracy and viewed as needed.
- Teacher Role: Can scan each student's unique QR code to mark them as present. The system automatically updates the attendance records in real time. Teachers can also view,

update, and generate attendance reports for the entire class, and print these reports for documentation or review.

4.3.1.1.4 **S***upplementary Lecture Materials*

This feature allows the student to download supplementary materials uploaded by the teachers. This will help lessen the burden from the students to easily accessed the needed files to study or review.

4.3.1.5 User Characteristics

- The Students are the primary users of the system since they frequently use it to track their grades, access supplementary materials, and manage their attendance. Their role involves regular interaction with the system, such as submitting assignments and viewing academic records.
- The Administrator are the key users as they have the highest level of control and access. They manage teacher and student accounts, assign roles, manage subjects and courses, oversee the grading system, schedule semesters, and ensure the overall functioning of the system
- Teachers have more of a managerial role but are secondary to students in terms of direct system use. They manage class records, grades, and attendance, and also upload lecture materials for students.

4.3.2 NonFunctional Requirements

Nonfunctional requirements define the system's performance and operational criteria, ensuring that it meets quality standards and provides a satisfactory user experience. Key nonfunctional requirements include:

4.3.2.1 Technical Requirements

The technical requirements for the Student Management System (SMS) are detailed and structured across both hardware and software specifications. The system utilizes a web-based platform developed using modern technologies such as PHP for server-side processing and MySQL for database management, ensuring efficient and secure data handling. The front-end is built with HTML, CSS, and JavaScript, providing a responsive, user-friendly interface that adapts across devices. The SMS runs on a reliable web server like Apache, hosted on operating systems like Ubuntu

or Windows Server, and requires compatibility with modern web browsers (e.g., Chrome, Firefox, Edge). The system also necessitates robust server hardware, such as multi-core processors (e.g., Intel Xeon), 16GB RAM, and solid-state drives (SSD) with at least 500GB storage to support concurrent operations and secure data storage.

Additionally, the SMS needs reliable network connectivity and requires that client devices, including desktops and Android smartphones, meet certain specifications for smooth performance.

4.3.2.2 Performance Requirements

The performance requirements for the Student Management System (SMS) specify that the system must handle concurrent user operations efficiently, ensuring that all users (students, teachers, administrators) can access and interact with the system without delays or lags. The system must provide real-time data retrieval, especially for features such as grade tracking, attendance, and academic material uploads. It is also critical that the system can scale effectively to accommodate a growing number of users without degrading performance.

Additionally, the system should ensure quick loading times for all web-based interfaces, maintaining optimal responsiveness across a variety of devices and browsers.

4.3.2.3 Assumptions and Dependencies

The assumptions and dependencies of the Student Management System (SMS) include the need for reliable internet access and compatible devices (modern computers and Android smartphones) to support system functionality. It also depends on accurate data input from instructors and administrators and assumes that users possess basic technical knowledge to interact with the system. The system's performance is dependent on the proper setup and maintenance of the web server (e.g., Apache) and the database (e.g., MySQL) to ensure scalability and reliable operations.

Reliable internet access for real-time system use.

Compatibility with modern devices (computers, Android smartphones).

Accurate data input by instructors and administrators.

Basic technical proficiency from users.

Proper configuration and maintenance of web server and database systems.

4.3.2.4 Security Requirements

Given that the Project System will be hosted on a cloud server, data security and user privacy are paramount. Several measures are implemented to ensure the security of user data and prevent unauthorized access:

Authentication: The system must implement a robust authentication mechanism requiring users (administrators, teachers, and students) to provide valid credentials (username/email and password) for access. This ensures that only authorized users can access the system's functions.

Role-based Access Control: The system must enforce access control based on user roles. Administrators, teachers, and students should have specific privileges, ensuring that sensitive information like grades and attendance records are only accessible to authorized users.

Data Encryption: Sensitive data, including user credentials and personal academic information, must be encrypted during transmission to protect it from interception and unauthorized access.

Data Validation: The system should validate all input data to prevent common security vulnerabilities such as SQL injection, which could compromise the integrity of the database.

Regular Backups: The system must perform regular data backups to prevent data loss in the event of system failure or security breaches.

4.4 System Design

The System Design phase is a pivotal step in developing the Student Management System (SMS). This phase converts the collected requirements into a detailed blueprint that outlines how the system will function and how its components will interconnect. It encompasses both high-level architectural design and the detailed design of individual system modules, ensuring the system is efficient, scalable, and user-friendly. By addressing both functionality and performance, the design phase ensures the SMS will meet the academic management needs of students, teachers, and administrators while remaining maintainable and adaptable for future enhancements.

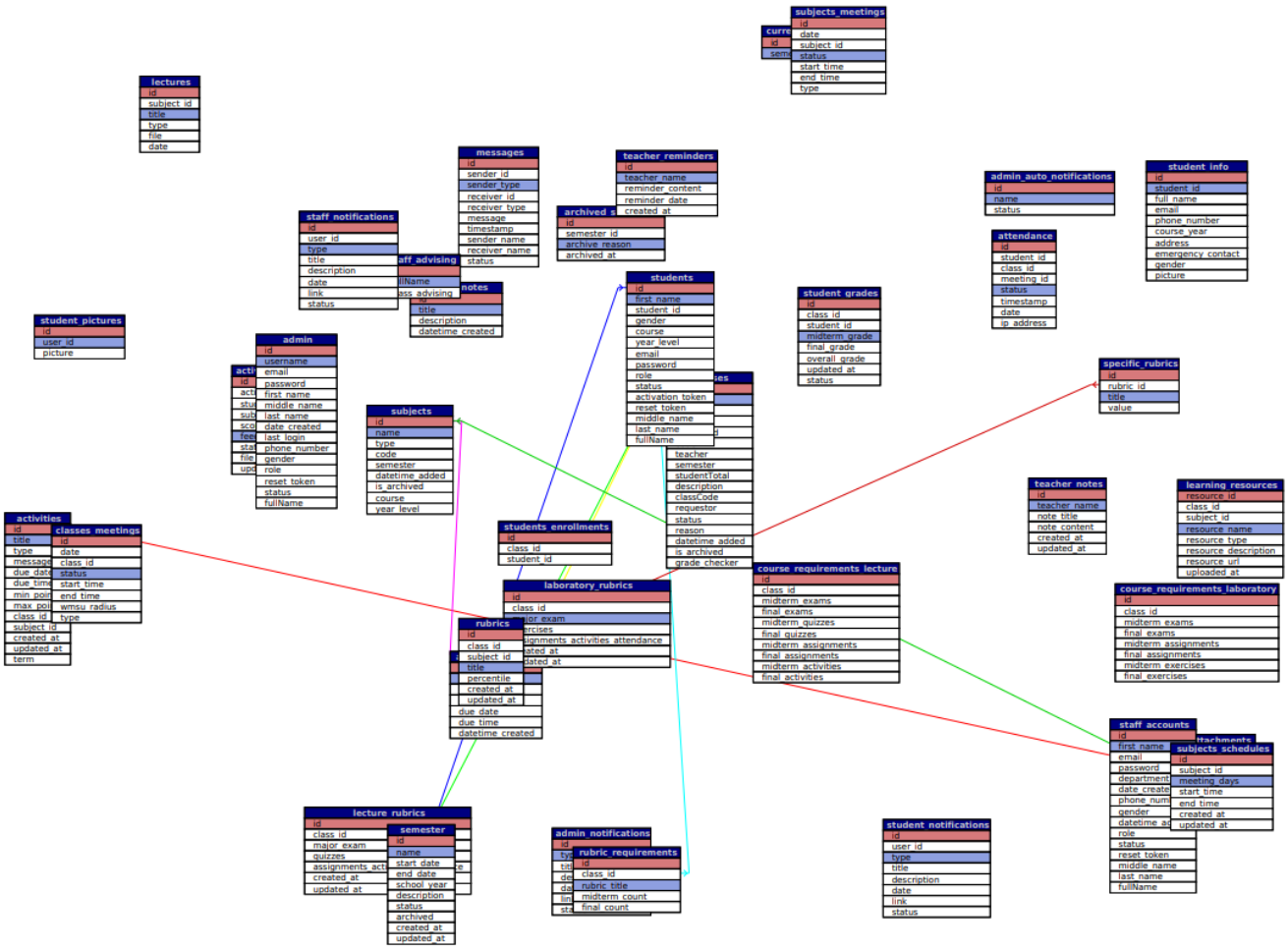


Figure 4: Database Design

The database design ensures that all tables are interconnected to maintain relationships between different data entities. For instance, the student table is linked to the grades table to store academic performance, while also being connected to the attendance table to track student participation. The teacher table is linked to the classes table, allowing teachers to manage class- specific records, such as grades and attendance. Similarly, the administrator table is connected to user accounts, enabling the creation, modification, and deletion of user roles. This interconnected structure ensures that all data is accessible and related, enabling seamless data retrieval and updates. It also supports data analysis, helping the system generate reports on student performance, attendance patterns, and overall academic progress. By linking these tables, the system ensures that all interactions between users (students, teachers, and administrators) are recorded and properly stored for future use in analytics and system improvements.

4.4.2 Entity Relationship Diagram

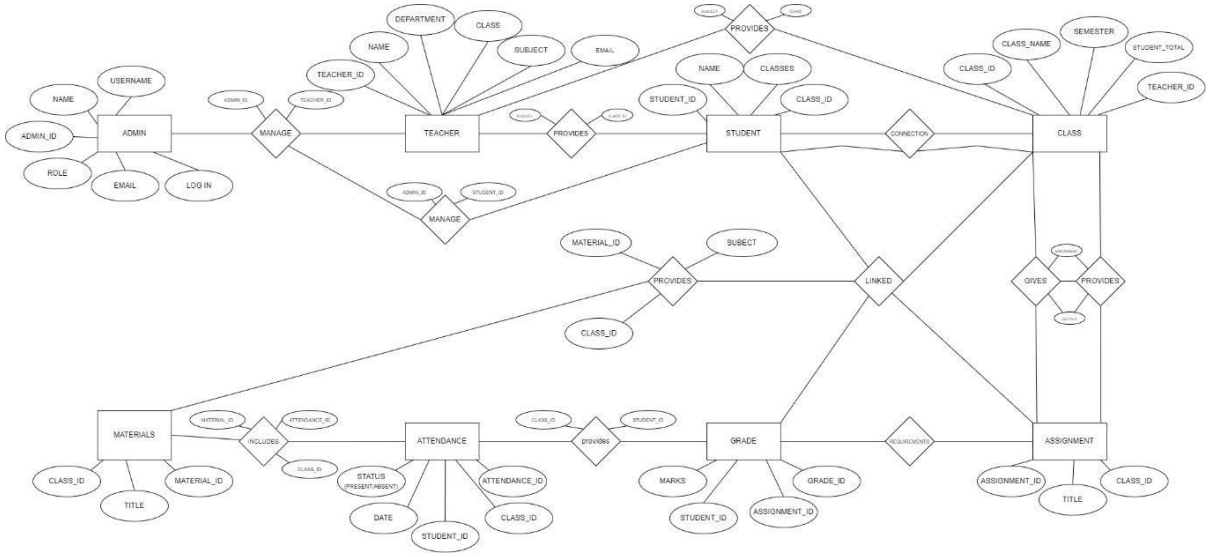


Figure 4.1: Entity Relationship Diagram

This diagram appears to be an Entity-Relationship Diagram (ERD) for a school management system. It models the relationships between various entities such as Admin, Teacher, Student, Class, Materials, Attendance, Grade, and Assignment. The admin entity manages the system with attributes like username, role, and email. Teachers manage classes and provide materials to students. Students are linked to classes, and each class has a subject, semester, and other related details. Materials are associated with specific classes and can be included in assignments. Attendance records track student participation, while grades are assigned to students based on their performance in assignments. The diagram highlights key relationships such as “MANAGE,” “PROVIDES,” and “LINKED,” illustrating how data flows between different components of the system.

4.4.3 DATABASE FIELDS

TABLE 3: activities

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Icon	TEXT			No
Type	ENUM	…		No
Title	TEXT			No
Description	TEXT			No
Date	DATETIME			No
Link	TEXT			No
Status	ENMU	…		No

TABLE 4: Activity_Attachement

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Activity_id	INT	11		No
File_name	VARCHAR	255		No
File_path	VARCHAR	255		Yes
Uploaded_at	DATETIME			No

TABLE 5: Activity_submission

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Acitivty_id	INT	11		No
Student_id	INT	11		No
Submission_date	INT	11		Yes
Score	DATETIME			Yes
Feedback	TEXT			Yes
Status	ENUM			Yes
File-path	VARCHAR	255		Yes
Updated_at	DATE			Yes

TABLE 6: Admin

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
username	VARCHAR	255		No
email	VARCHAR	255		No
password	VARCHAR	255		Yes
First_name	VARCHAR	255		No
Middle_name	VARCHAR	255		No
Last_name	VARCHAR	255		No
Date_created	DATE			Yes
Last_login	DATETIME			Yes
Phone_number	VARCHAR	255		No
Gender	TEXT			No
Role	TEXT			No
Reset_tokens	TEXT			Yes

Status	**TEXT**			**Yes**
fullName	**TEXT**			**Yes**

TABLE 7: Admin_auto_notifications

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
name	**TEXT**			**No**
status	**TEXT**			**No**

TABLE 8: Admin_notes

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Title	**VARCHAR**	**255**		**No**
Description	**TEXT**			**No**
Datetime_created	**TEXT**			

TABLE 9: Admin_notifications

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Type	**ENUM**			**No**
Title	**TEXT**			**No**
Description	**TEXT**			**No**
Date	**DATETIME**			**No**
Link	**TEXT**			**No**
Status	**ENUM**			**No**

TABLE 10: Admin_reminders

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Title	**TEXT**			**No**
Description	**TEXT**			**No**
Level	**TEXT**			**No**
Due_date	**TEXT**			**No**
Due_time	**TEXT**			**No**
Datetime_created	**TEXT**			**No**

TABLE 11: Archived_semesters

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Semester_id	**INT**	**11**		**No**
Archive_reason	**TEXT**			**No**
Archived_at	**TIMESTAMP**			**No**

TABLE 12: Attendance

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Student_id	**INT**	**11**		**No**
Class_id	**INT**	**11**		**No**

Meeting_id	**INT**	**11**		**No**
Status	**ENUM**			**No**
Timestamp	**TIMESTAMP**			**No**
Date	**DATE**			**No**
Ip-address	**TEXT**			**Yes**

TABLE 13: Classes

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Name	**TEXT**			**No**
Type	**TEXT**			**Yes**
Subject	**TEXT**			**No**
Subject_id	**INT**	**11**		**No**
Code	**TEXT**			**Yes**
Teacher	**TEXT**			**No**
Semester	**TEXT**			**No**
Student_total	**INT**	**11**		**No**
Description	**TEXT**			**No**
Class_code	**TEXT**			**Yes**
Requestor	**TEXT**			**Yes**
Status	**TEXT**			**No**
Reason	**TEXT**			**Yes**
Datetime_added	**DATETIME**			**No**
Is_archived	**INT**	**11**		**No**
Grade_checker	**TEXT**			**Yes**

TABLE 14: Classes_meetings

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Date	**DATE**			**No**
Class_id	**INT**	**11**		**No**
Status	**TEXT**			**No**
Start_time	**TEXT**			**No**
End_time	**TEXT**			**No**
Wmsu_radius	**TEXT**			**No**
Type	**TEXT**			**No**

TABLE 15: Classes_students

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Class-id	**INT**	**11**		**No**
Student_id	**INT**	**11**		**No**
Enrollment_date	**DATETIME**			**Yes**
Status	**ENUM**			**Yes**
Grade	**DECIMAL**	**5,2**		**Yes**

TABLE 16: Course_requirements_laboratoy

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Class_id	INT	11		No
Midterm_exams	INT	11		No
Final_exams	INT	11		No
Midterm_assignments	INT	11		No
Final_assignments	INT	11		No
Midterm_exercises	INT	11		No
Finalk_exercises	INT	11		No

TABLE 17: Course_requirements_lecture

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Class_id	INT	11		No
Midterm_exams	INT	11		No
Final_exams	INT	11		No
Midterm_quizzes	INT	11		No
Final_qquizzes	INT	11		No
Midterm_assignments	INT	11		No
Final_assignments	INT	11		No
Midterm_activity	INT	11		No
Finalk_activity	INT	11		No

TABLE 18: Current_semester

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Semester	TEXT			No

TABLE 19: Laboratory_rubrics

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Class_id	INT	11		No
Major_exam	DECIMAL	5,2		No
Exercises	DECIMAL	5,2		No
Assignments_activities_attendance	DECIMAL	5,2		No
Final_qquizzes	TIMESTAPM			No
Midterm_assignments	TIMESTAMP			No

TABLE 20: Learning_resources

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Class_id	INT	11		No
Subject_id	DECIMAL	5,2		No
Resource_name	VARCHAR	255		No
Resource_type	TEXT	5,2		No
Resource_description	TEXT			Yes
Resource_url	VARCHAR	500		No
Uploaded_at	TIMESTAMP			No

TABLE 21: Lectures

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Subject_id	INT	11		No
Title	TEXT			No
Type	ENUM			No
File	TEXT			No
Date	DATE			No

TABLE 22: Lecture_rubrics

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Class_id	INT	11		No
Major_exam	DECIMAL	5,2		No
Quizzes	DECIMAL	5,2		No
Assignments_activities_attendance	DECIMAL	5,2		No
Created_at	TIMESTAMP			No
Updated_at	TIMESTAMP			No

TABLE 23: Messages

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Sender_id	INT	11		No
Sender_type	ENUM			No
Receiver_id	INT	11		No
Receiver_type	ENUM			No
Messages	TEXT			No
Timestamp	TIMESTAMP			No
Sender_names	VARCHAR	255		No
Receive_name	VARCHAR	255		No
Status	TEXT			No

TABLE 24: Rubrics

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Class_id	INT	11		No
Subject_id	INT	11		No
Title	VARCHAR	255		No
Percentile	DECIMAL	10,0		Yes
Created_at	TIMESTAMP			No
Updated_at	TIMESTAMP			No

TABLE 25: Rubric_requirements

Attribute Name	Data Type	Max Length	Key Type	Null

Id	**INT**	**11**	**Primary Key**	**No**
Class_id	**INT**	**11**		**No**
Rubric_Title	**VARCHAR**	**255**		**No**
Midterm_count	**INT**	**11**		**No**
Final_count	**INT**	**11**		**No**

TABLE 26: Semester

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Name	**VARCHAR**	**255**		**No**
Start_date	**DATE**			**No**
End_date	**DATE**			**No**
School_year	**TEXT**			**No**
Description	**TEXT**			**Yes**
Status	**ENUM**			**Yes**
Archived	**TINYINT**	**1**		**Yes**
Created_At	**TIMESTAMP**			**No**
Updated_at	**TIMESTAMP**			**No**

TABLE 27: Specific_rubrics

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Rubric_id	**INT**	**11**		**No**
Title	**VARCHAR**	**255**		**No**
Value	**DECIMAL**	**5,2**		**No**

TABLE 28: Staff_accounts

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
First_name	**TEXT**			**No**
Email	**TEXT**			**No**
Password	**TEXT**			**No**
Department	**TEXT**			**Yes**
Date_created	**DATE**			**No**
Phone_number	**TEXT**			**Yes**
Gender	**TEXT**			**No**
Datetime_Added	**DATETIME**			**No**
Role	**TEXT**			**No**
status	**TEXT**			**No**
Reset_token	**TEXT**			**Yes**
Middle_name	**TEXT**			**No**
Last_name	**TEXT**			**No**
fullName	**TEXT**			**No**

TABLE 29: Staff_advising

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**

fullNAme	**TEXT**			**No**
Class_advising	**TEXT**			**No**

TABLE 30: Staff_notifications

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
User_id	**INT**	**11**		**No**
Type	**ENUM**			**No**
Title	**TEXT**			**No**
Description	**TEXT**			**No**
Date	**DATETIME**			**No**
Link	**TEXT**			**No**
Status	**ENUM**			**No**

TABLE 31: Students

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
First_name	**TEXT**			**No**
Student_id	**INT**	**11**		**No**
Gender	**TEXT**			**No**
Course	**TEXT**			**Yes**
Year_level	**DATE**			**No**
Email	**TEXT**			**Yes**
Password	**TEXT**			**No**
Role	**DATETIME**			**No**
Status	**TEXT**			**No**
Activation_token	**TEXT**			**Yes**
Reset_token	**TEXT**			**Yes**
Middle_name	**TEXT**			**No**
Last_name	**TEXT**			**No**
fullName	**TEXT**			**No**

TABLE 32: Students_enrollment

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Class_id	**INT**	**11**		**No**
Student_id	**INT**	**11**		**No**

TABLE 33: Student_grades

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Class_id	**INT**	**11**		**No**
Student_id	**INT**	**11**		**No**
Midterm_grade	**TEXT**			**No**
Final_grade	**TEXT**			**No**
Overall_grade	**TEXT**			**No**
Updated_at	**TIMESTAMP**			**No**
status	**TEXT**			**Yes**

TABLE 34: Student_info

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Student_id	VARCHAR	50	Primary Key	No
Ful_name	VARCHAR	255		No
Email	VARCHAR	255		No
Phone_number	VARCHAR	20		No
Course_year	VARCHAR	255		No
address	TEXT			No
Emergency_contact	VARCHAR	20		No
Gender	ENUM			No
Picture	TEXT			Yes

TABLE 35: Student_notifications

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
User_id	INT	11		No
Type	ENUM			No
Title	TEXT			No
Description	TEXT			No
Date	DATETIME			No
Link	TEXT			No
Status	ENUM			No

TABLE 36: Student_pictures

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
User_id	TEXT			No
Picture	TEXT			No

TABLE 37: Subject

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Name	TEXT			No
Type	TEXT			No
Code	TEXT			No
Semester	TEXT			No
Datetime_added	DATETIME			No
Is_archived	INT	11		No
Course	VARCHAR	100		No
Year_level	VARCHAR	50		No

TABLE 38: Subjects_meetings

Attribute Name	Data Type	Max Length	Key Type	Null
Id	INT	11	Primary Key	No
Date	DATE			No

Subject_id	**INT**	**11**		**No**
Status	**TEXT**			**No**
Start_time	**TEXT**			**No**
End_time	**TEXT**			**No**
Type	**TEXT**			**No**

TABLE 39: Subjects_Schedules

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Subject_id	**INT**	**11**		**No**
Meeting_days	**ENUM**			**No**
Start_time	**VARCHAR**	**555**		**No**
End_time	**VARCHAR**	**555**		**No**
Created_at	**TIMESTAMP**			**No**
Updated_at	**TIMESTAMP**			**No**

TABLE 40: Teaher_notes

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Teacher_name	**VARCHAR**	**255**		**No**
Note_title	**VARCHAR**	**255**		**No**
Note_content	**TEXT**			**No**
Created_at	**TIMESTAMP**			**No**
Updated_at	**TIMESTAMP**			**No**

TABLE 41: Teaher_reminders

Attribute Name	Data Type	Max Length	Key Type	Null
Id	**INT**	**11**	**Primary Key**	**No**
Teacher_name	**VARCHAR**	**255**		**No**
Reminder_content	**TEXT**			**No**
Reminder_date	**DATE**			**Yes**
Created_at	**TIMESTAMP**			**No**

4.4.4 User Interface Prototype

Figure 5

Home Page View

This is our landing page where the red logo is for logging in for all types of users. All users whether it is a Student, Staff or Admin, all users has the same Landing Page.

Accessing the Login Page:

- Open your web browser and navigate to the login page of the system

Entering User Credentials:

- Once on the login page, you will typically see fields prompting you to enter your credentials given to who created the system.
- Enter your username in the designated field. Ensure that you input the correct username associated with your account.

Inputting Password:

- After entering your username, navigate to the password field.
- Input your password given by the developer carefully, ensuring that it matches the password associated with your username

Clicking the Login Button:

- Once you have entered your username and password, locate the "Login" button on the page.
- Click on the "Login" button to initiate the authentication process and access the system.

Accessing Dashboard (Upon Successful Login):

- Upon successful authentication, you will be granted access the system.
- Once granted the access to the system you will be directed to dashboard where you can see the overall view of the system.

Navigating the System:

- Once logged in, you can navigate the various functionalities of the system using the menu option and buttons

Explore the different sections of the systems as needed to perform your task and access relevant information.

Dashboard

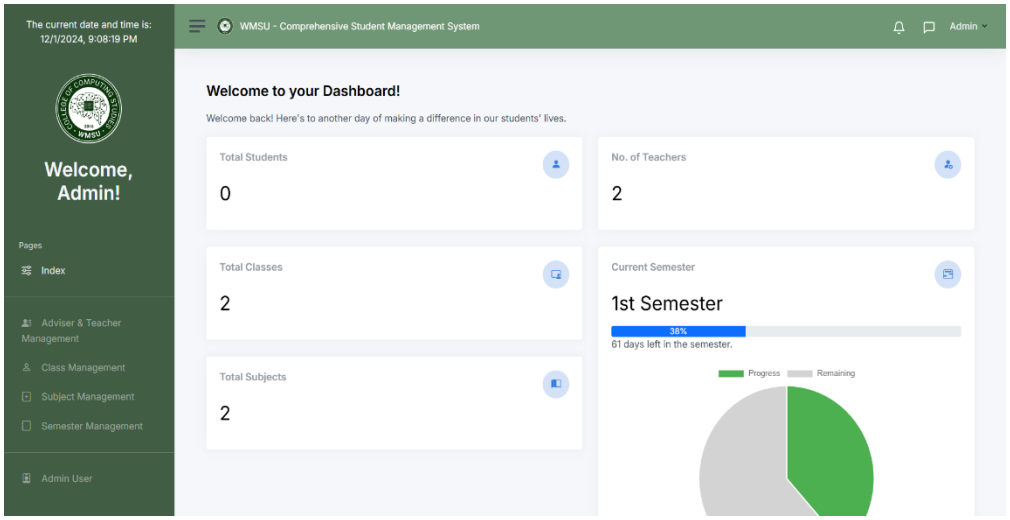


Figure 5.1

Dashboard

This feature gives you the overall view of the system. It displays the total of the students, Total of the teachers, total of classes, total subjects, and the current semester. Also has a quick access where you can click one of the modules to be directed to the other modules you would like to go.

Accessing Dashboard (Upon Successful Login)

- Upon successful authentication, you will be granted access to the system.
- As an Admin you will be directed to an Admin Dashboard page where will be displaying relevant information and options.

Navigating to specific Sections:

The dashboard/home features a user-friendly navigation bar menu designed for easy access to various sections within the system. This menu comprises essential tabs such as Dashboard, Class Management, Teacher Management, Subject Management, Semester Management, and Admin User.

Through this intuitive interface, users can seamlessly delve into critical metrics, including:

-Total Students
-Total Teachers
-Total Classes
-Total Subjects
-Current Semester
-Quick Access Buttons

Create a Semester

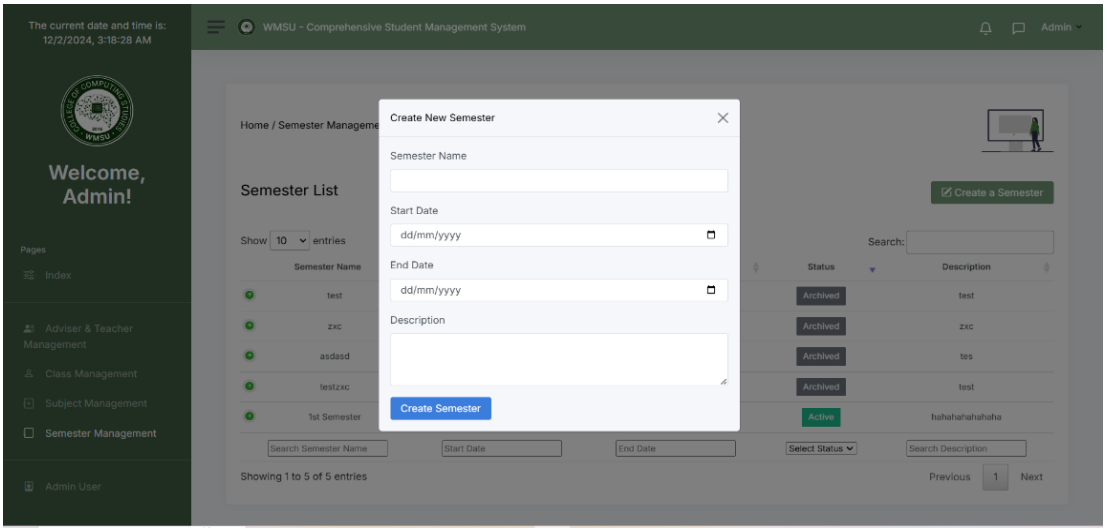


Figure 5.2

Create a Semester

This section allows the admin to create a new semester by entering the necessary details into the form provided.

Entering Semester Information:

- The "Create a Semester" form includes the following fields:
- **Semester Name:** The name or identifier for the semester
- **Start Date:** The starting date of the semester.
- **End Date:** The Ending date of the semester.
- **Semester Description:** A brief description or note regarding the semester

Create Semester

- **Create Semester:** Click this button to save the entered semester details. After Clicking the "create semester" the sweet alert will pop up.

Navigating the System:

- Admins can open this form by clicking the "Create a Semester" button on the semester Management Page.
- After successful creation, the newly added semester will appear in the semester list on the dashboard.

Semester Management

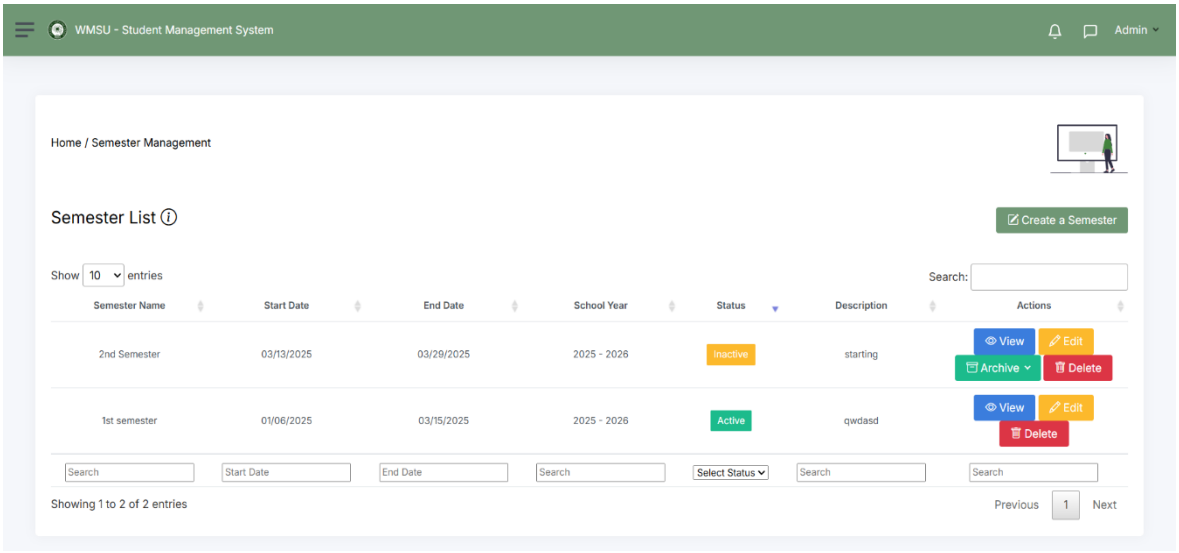


Figure 5.3
Semester Management

This section allows the admin to manage semester details such as view, edit, delete button to navigate and control the semester.

View Semester

- Upon accessing the Semester Management section, you can view the semester created to clarify or to check if there are any errors to change onwards.

Edit Semester:

- The admin can click on the “Edit Semester” button to begin the process of editing or updating any information in the semester created.
- If all of the requirements are completed you can go to this button in click “make active” to activate the semester created
-

Delete the Semester:

- Admin can easily delete the semester created.

Archiving;

- In this section the admin can manually archived the semester created.
- But if the users want automatically archived the semester, the system can automatically archived based on the end date of the semester.

Subject Management

Creating a Subject

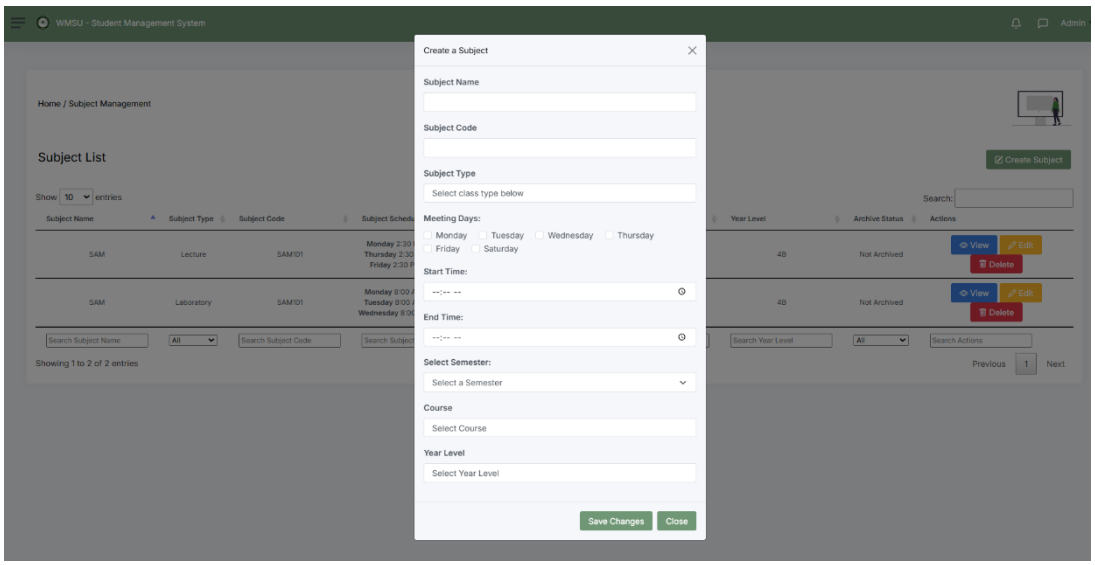


Figure 5.4

Creating a Subject

This section allows the admin to create a subject for the semester

- To create a subject click the “create subject” and manually input all the information’s needed such as “subject name”, “subject code”, “subject type”, if it is laboratory or lecture, “meeting days”, the “start time and end time”, “assigned semester”, the “course and the year level”.
- After fulfilling the needed information click “save changes” to save the subject created then “close button” to exit.
- “view” button is the viewing of the information of the subject created
- “edit” button for editing any wrong information
- “delete” button for deleting the subject created.

Adviser and Teacher Management

Creating of Staff Accounts

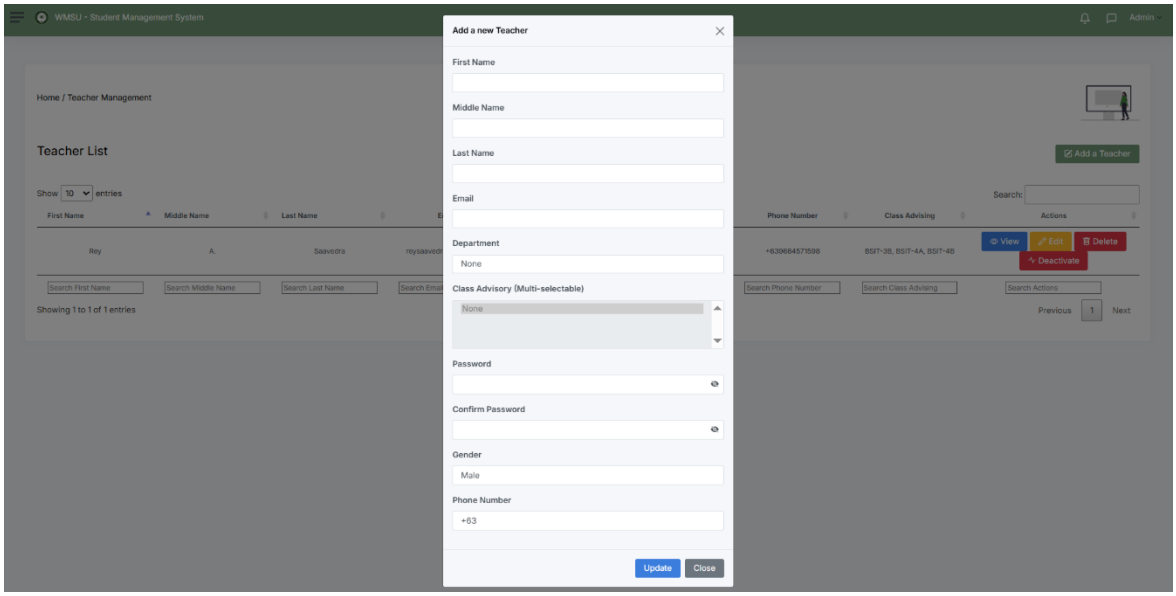


Figure 5.5

Creating of Staff Accounts

This section allows the admin to create the staff accounts for the semester.

- Put all the information needed first is to enter their “First name, “middle name”, “last name”, “email, “department”. For class advisory you can assign the staff created into multiple sections. Next is to set their “password” the “confirm password”, “gender”, and lastly their “contact number”.
- Click “update button” to save the teacher credentials information.
- “view” button for viewing the staff credentials.
- “edit” button for editing any wrong information
- “delete” button for removing accounts created
- “activate/deactivate” activate for activating the staff accounts to functions, and deactivate.

Class Management

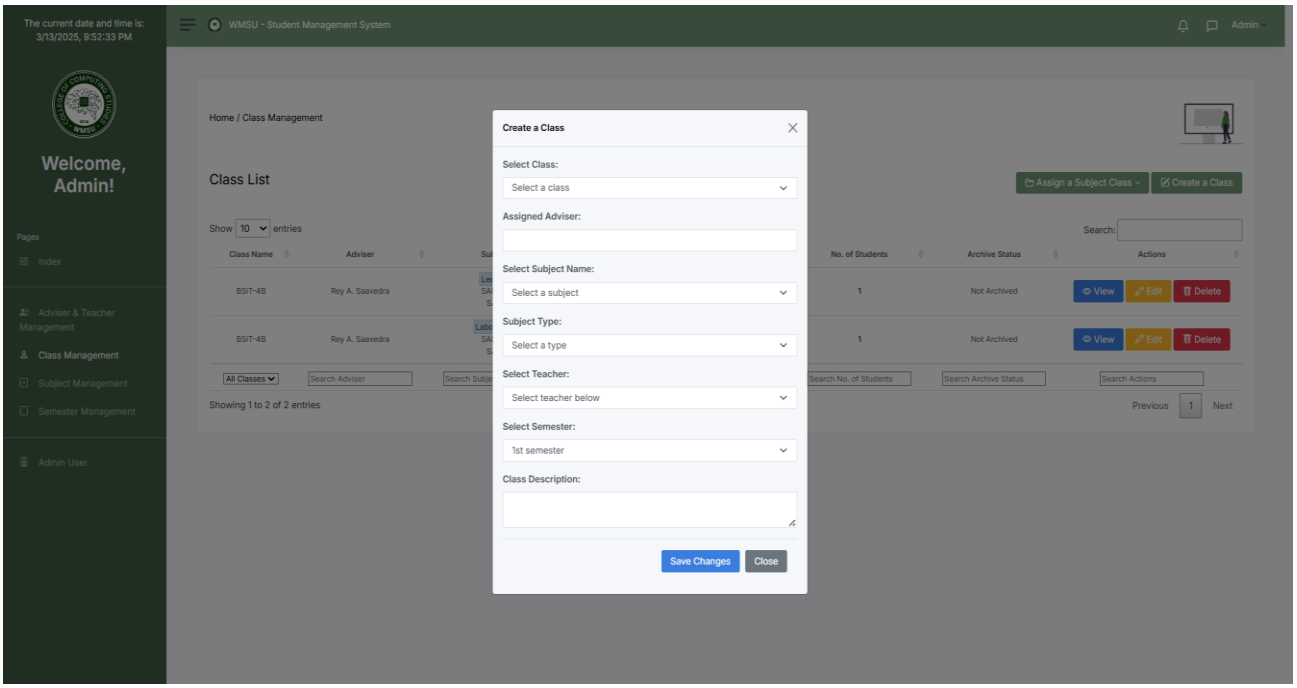


Figure 5.6

Creating a Class

This section allows the admin for creating a class in the semester.

- In this section you can select or assign a class, staff, subject name, the subject type, and select staff again to manage the class, the semester and last the class description.
- After successfully filled the information needed click “save changes” to save the class created and “close button” to exit.
- After creating the class created you can now access the buttons in the right corner such as “view” for viewing, “edit” for editing, and “delete” for deleting the class created.
- “assigned” a subject class” for shortcuts of assigning the staff to a class.

Admin User Section

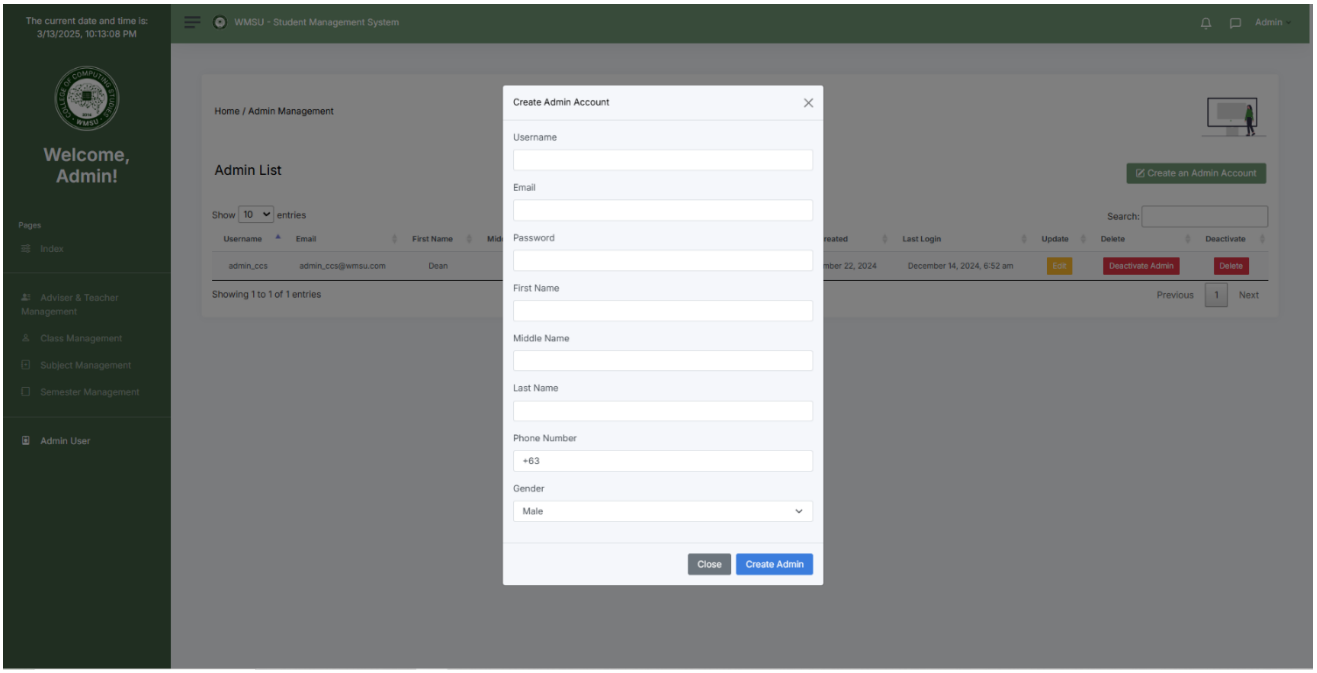


Figure 5.7

Adding new admin users

- In this section the admin users can add new user accounts for admin by manually inputting the information needed such as username, email, password, then its first name, middle name, last name, phone number, and lastly their gender.
- Click the “create admin” button for saving new user accounts and close button to exit
- “edit” button for updating any wrong information then save.
- “activate/deactivate” button to activate and deactivate admin accounts.
- “delete” button for deleting admin accounts.

Staff Dashboard

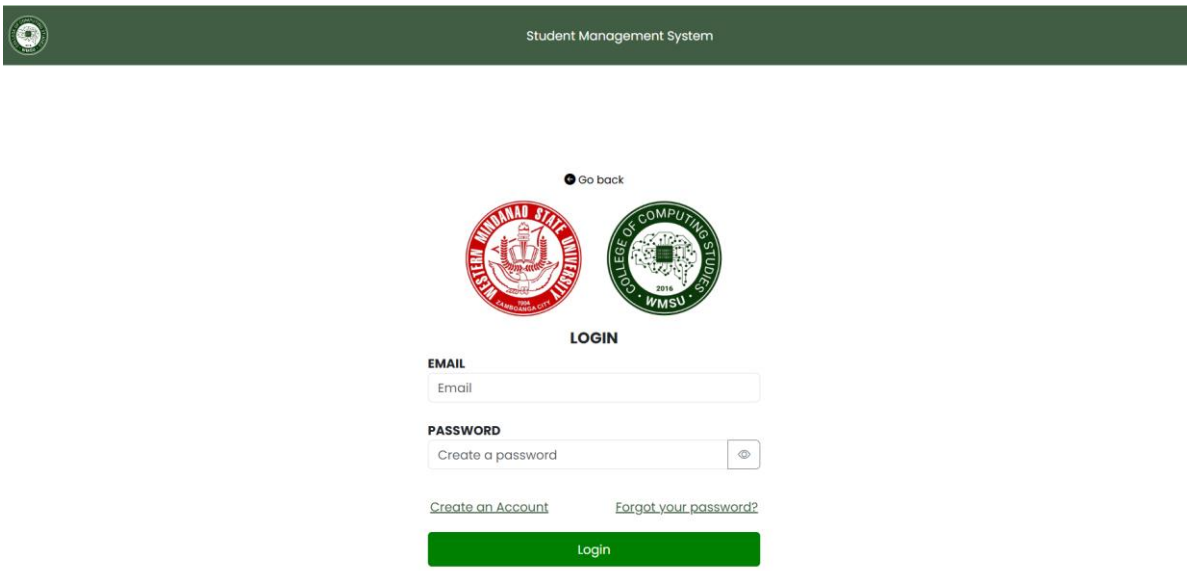


Figure 5.7

Landing Page

This is the landing page for the staff and students for creating and logging in.

- In this section this is where the staff input their email and password created by the admin

Staff Dashboard

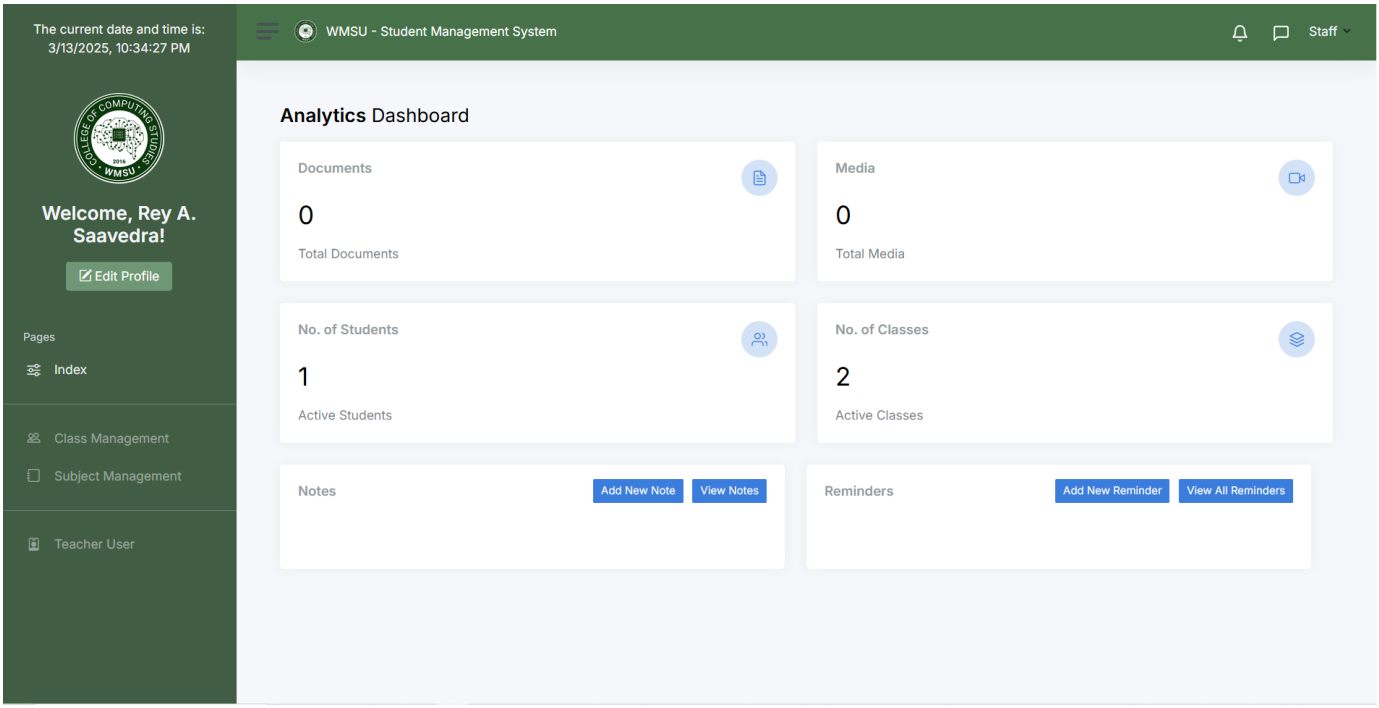


Figure 5.8

Teacher Dashboard

The Teacher Dashboard provides an intuitive interface for managing classes, subjects, and student's performance. There are four section in a nav bar for the staff to navigate. The Class Management, Subject Management, and Teacher User. Its also has a analytics dashboard for providing insights on students performance, no. of students enrolled, notes, reminders, media, and no. of classes.

Class Management/Subject Management

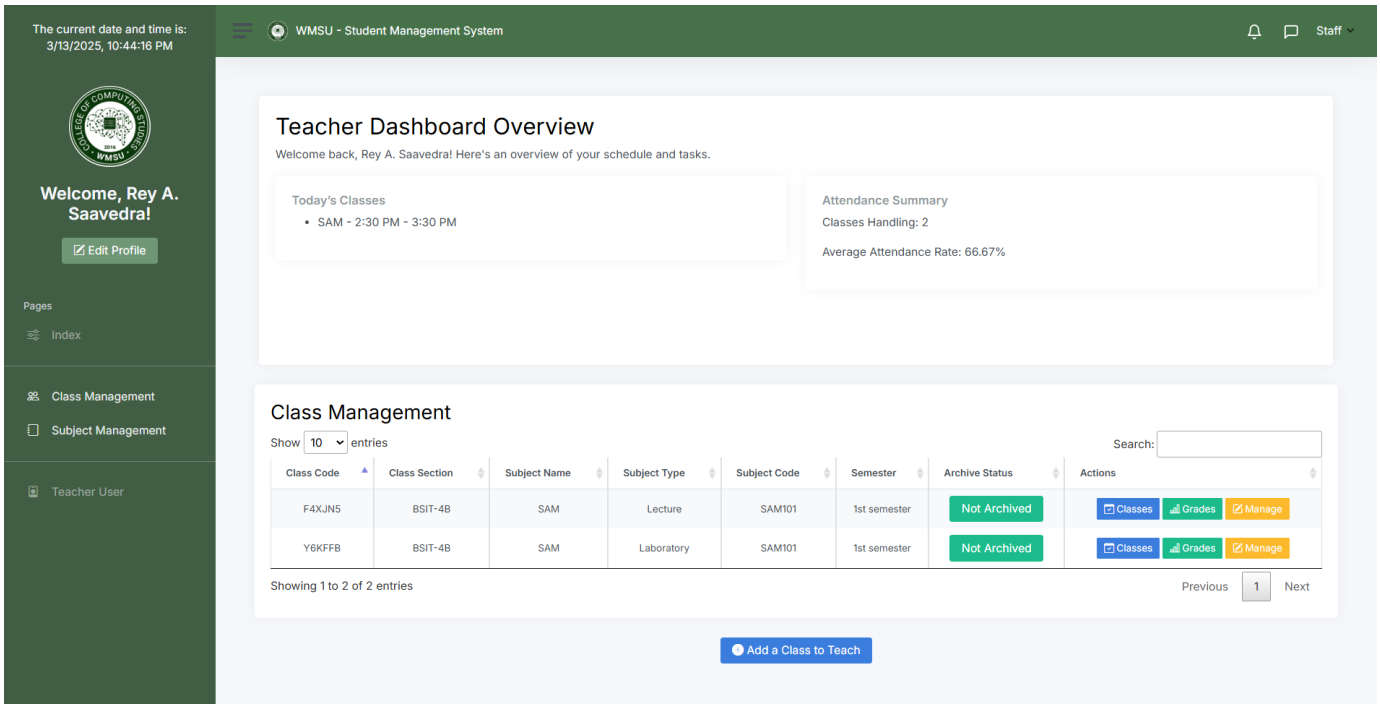


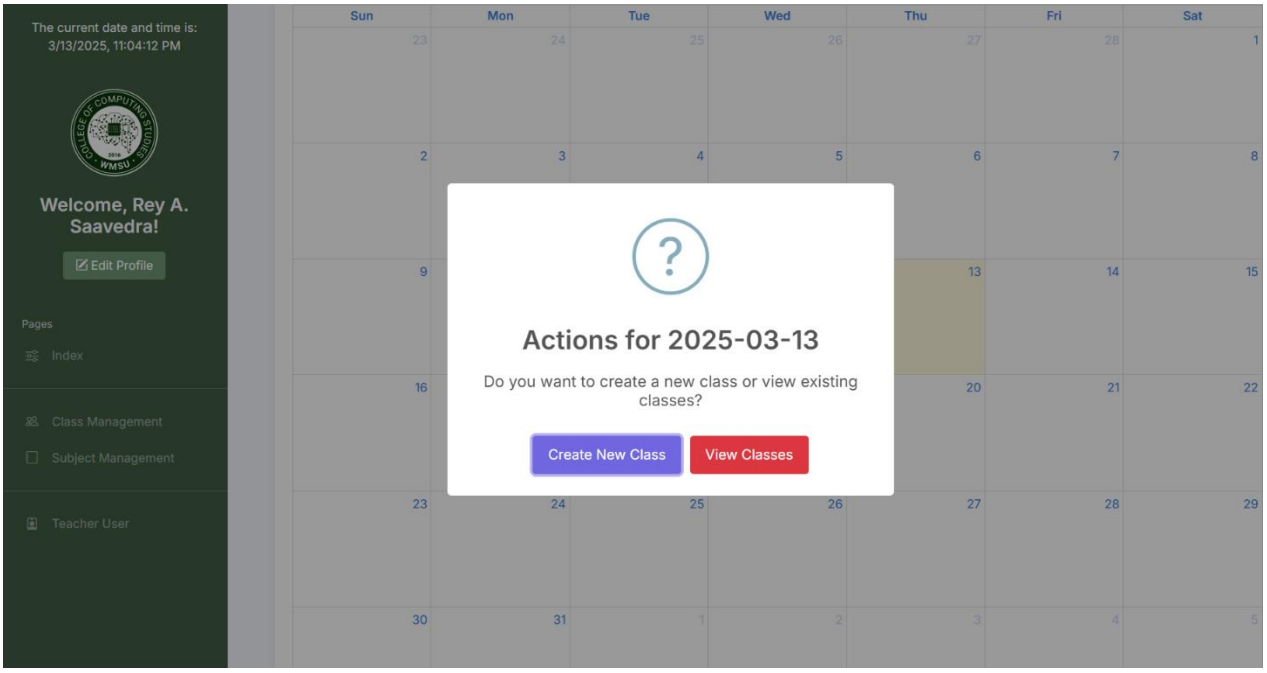
Figure 5.9

Class Management/Subject Management

In this section the staff can view all the assigned subjects and classes created by the admin.

- This section shows the Classes and subject created by the admin where it indicates the Class code, the class section, subject name, subject type, subject code, the semester, the status for archiving, and the actions
- There are 3 different actions in class management, the Classes for handling attendance, grades for recording the student performance, and manage for managing the student's activities, scores, grades, and resources.

Classes Sections



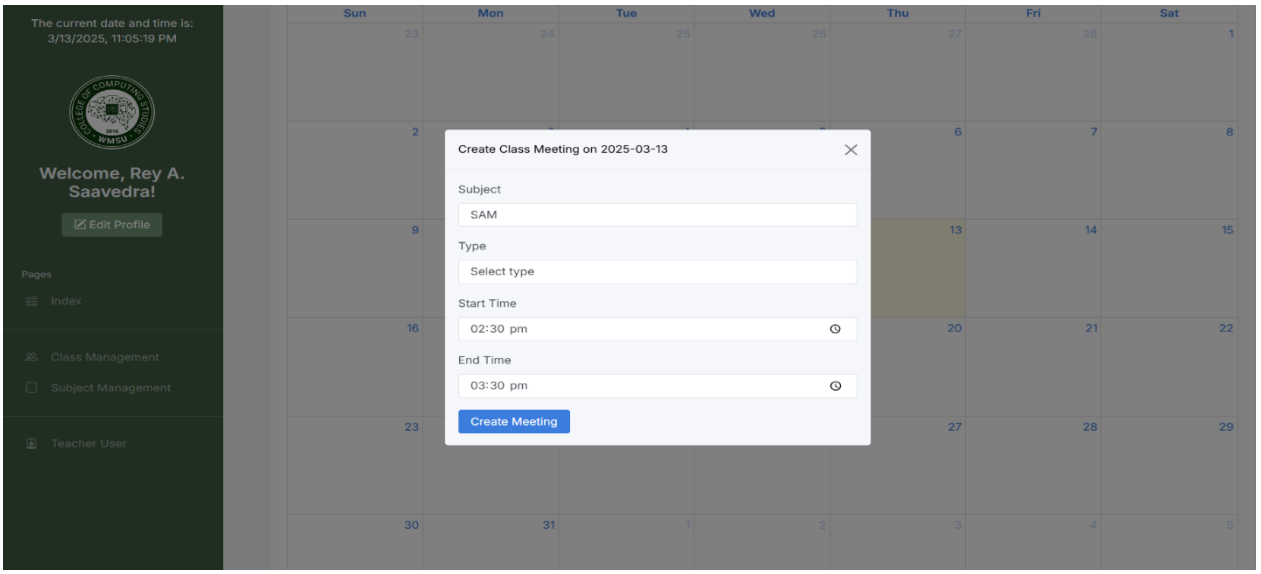


Figure 6

Create new class

- This section allows the staff to create a new class
- To create a class click “create new class” then click “type” to select a type of the class to be conducted, if it is “regular”, “late”, or “make-up class”.
- Next is click the “Create meeting” to conduct the class.

View Classes

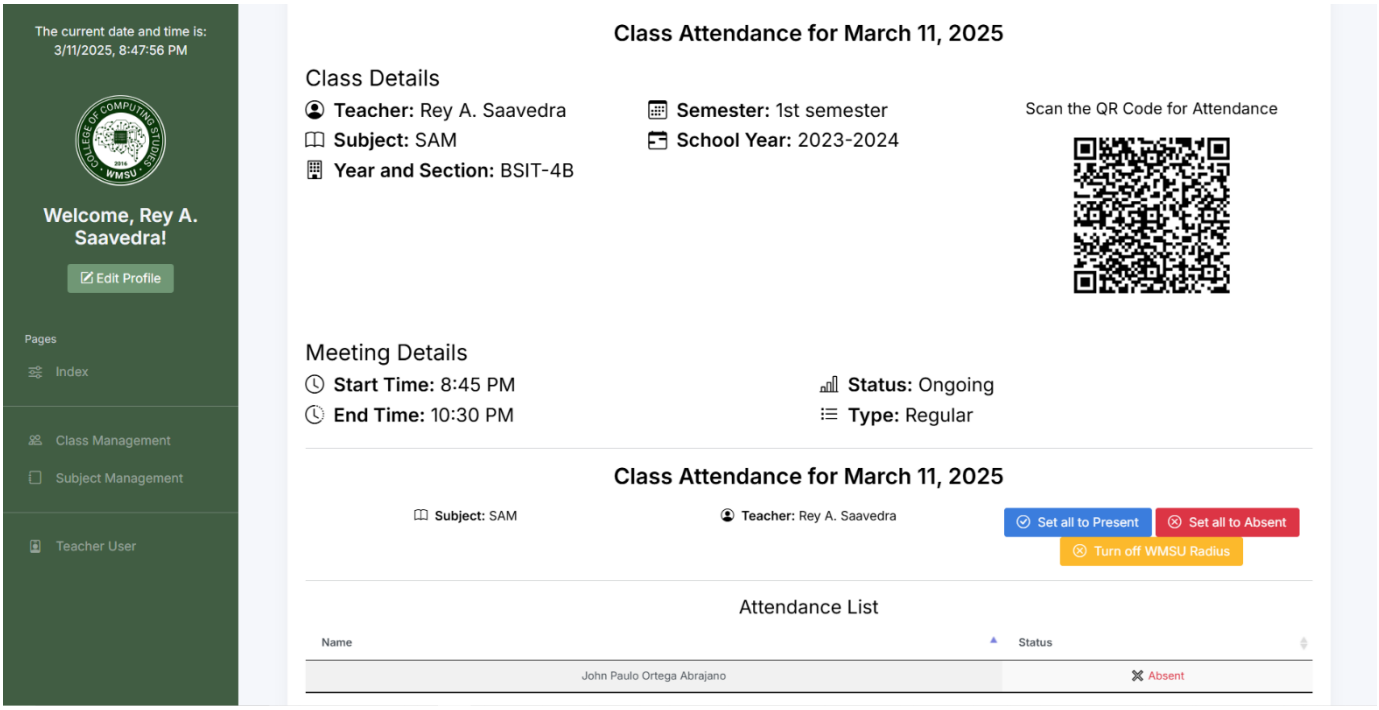


Figure 6.1

View classes

In the View classes section there are 3 buttons which are Attendance, Edit, and delete

- The picture above represents the attendance section in “view classes’ where the students scan the QR code for them to be present in class.
- The Turn on and off radius, is a feature that if the radius is turned on it set a 20 meter radius. The status of their attendance will be recorded if they are in the radius conducting the scanning of attendance if off it’s the opposite.
- after the students scanned the QR, their status will be marked as present, late, or absent.
- The 2 remaining buttons are edit and delete, edit for editing class status, start time, and end time while delete button is for deletion of the class created.

Attendance

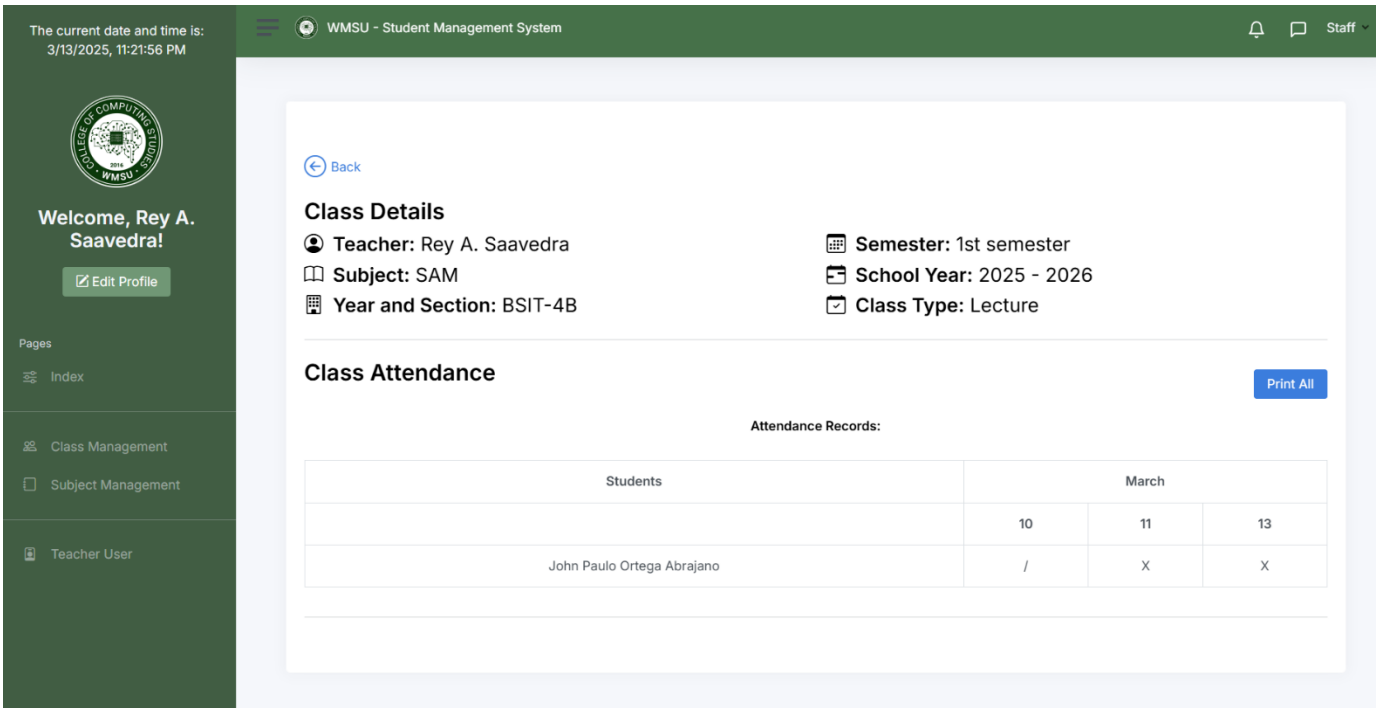


Figure 6.2

View Attendance Record

- The section allows the teacher to view the attendance record of the students from the first day of the classes conducted to the last day of the semester.

Grades

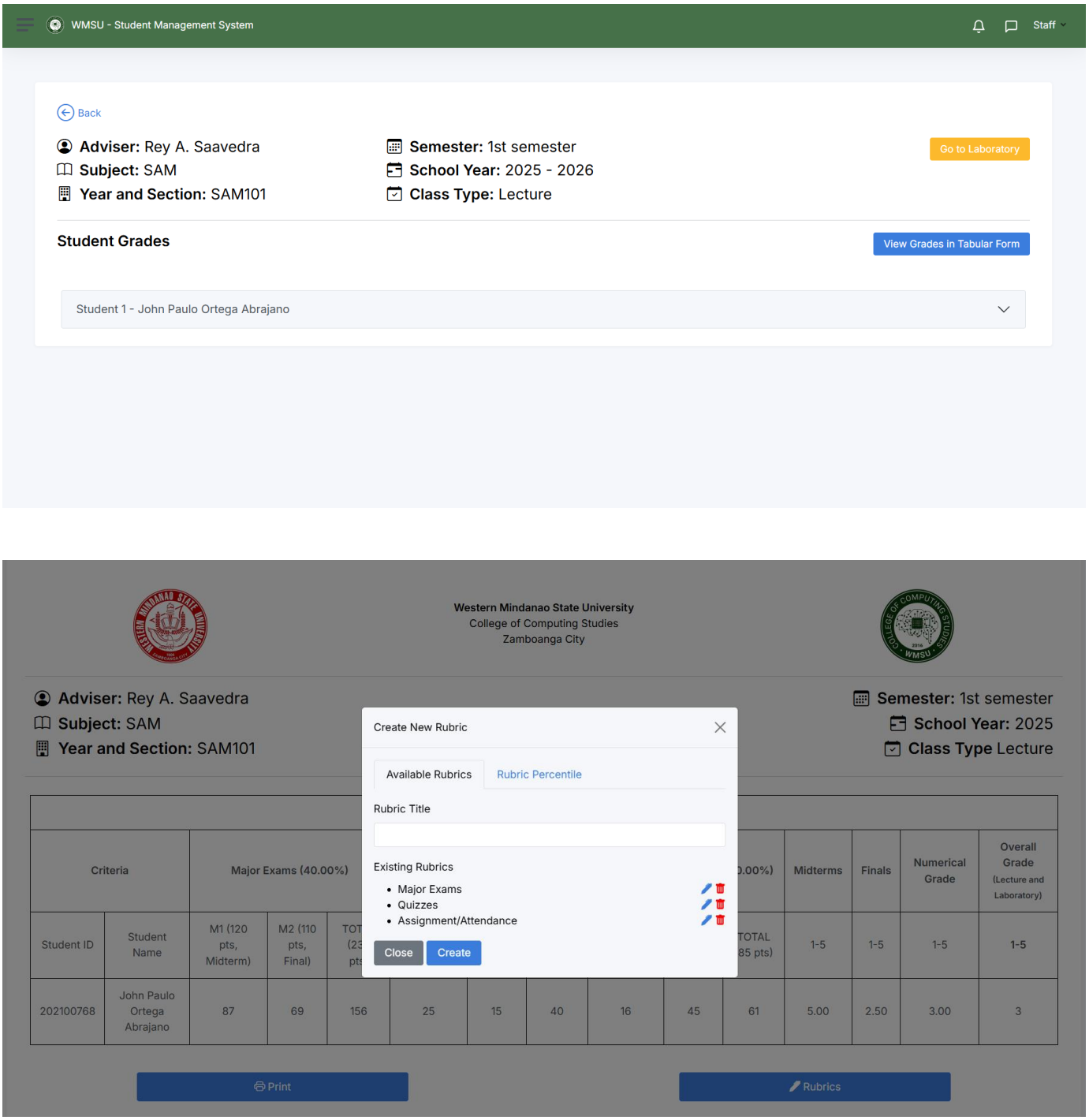


Figure 6.3

Setting the Grades in Rubrics

- In Grade Action this is where the staff set the rubrics of the subject based on the standard of the university.
- Click the “Tabular Button” for accessing the rubrics.
- Find the “rubrics” button for setting the grades/percentile.
- This section also allows the staff to monitor the student’s performance, grades and any activities conducted by the teacher.

Manage

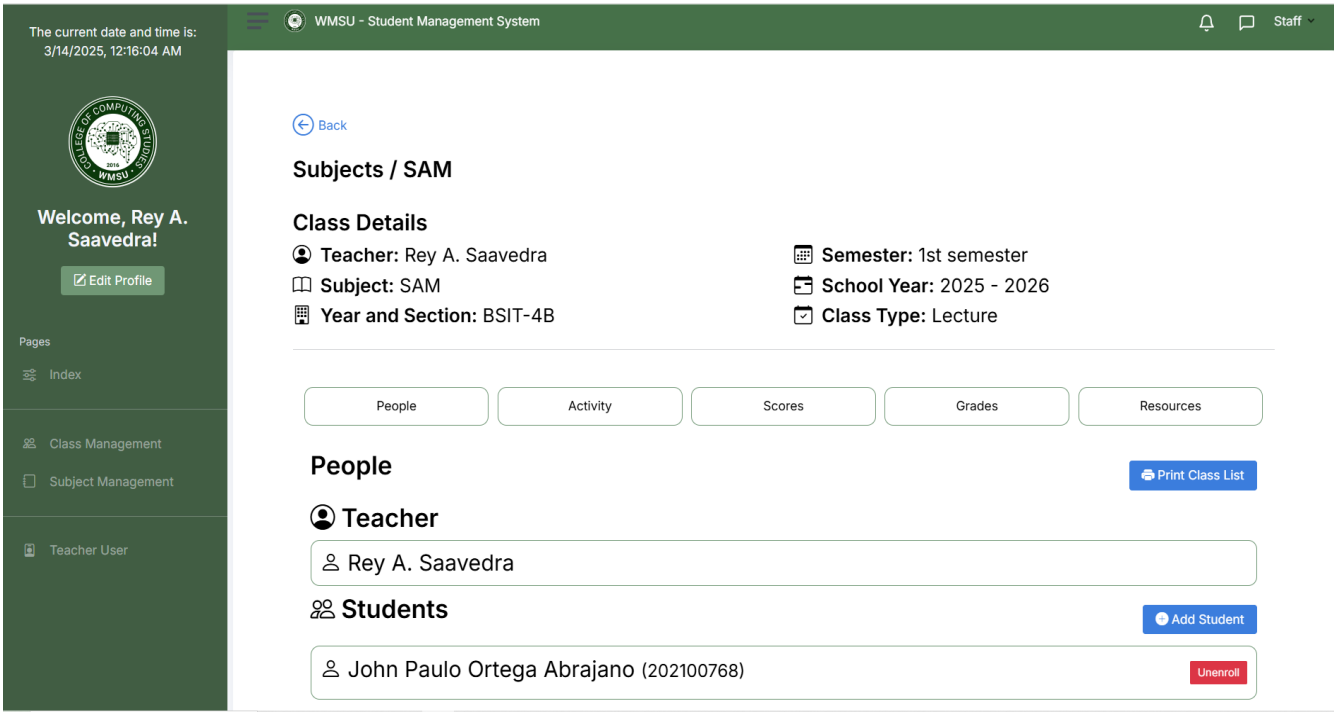


Figure 6.4

People Section

This section allows the teacher to view the students enrolled in class, and they can also add a student to enroll in their class.

Activity

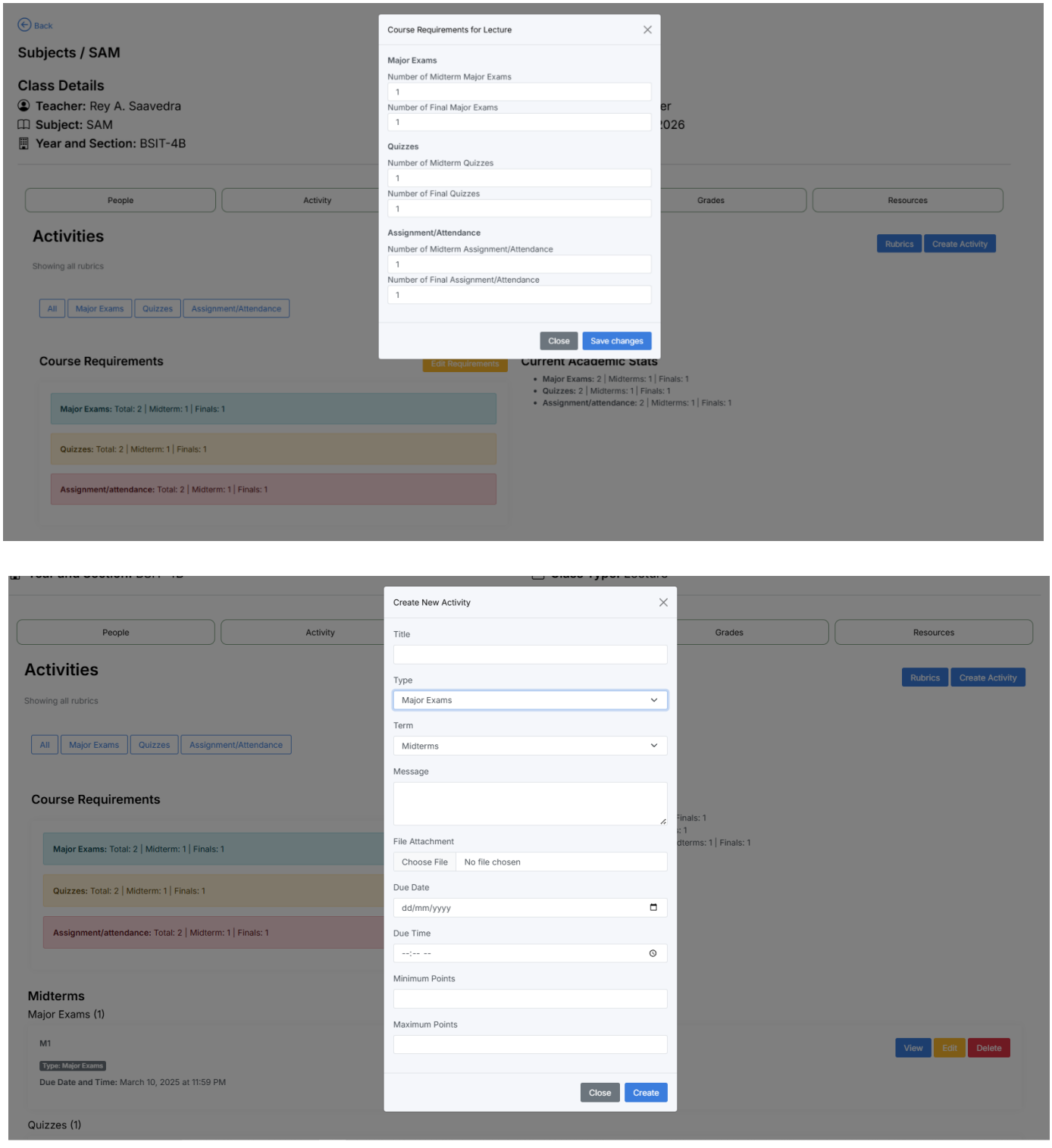


Figure 6.5

Setting Activities

This section allows the staff to manage the number of course requirements needed to met the criteria for grading.

- In order to set the requirements, click the “edit requirements” to set a number of course requirements set by the teacher.
- After creating the set requirements you can now proceed to create activities by clicking the “activity” button to create types of activities or exams first input the title, type of activities if it is major exams, quizzes, or activities, then the term for the activities created, message, files of the activity, the due date, and the due time, ‘set the minimum points for the activity and also the maximum.
- Click “save” button to save the activity created, and close to exit.
- Below is the activity created by the teacher, and they can view, edit, and delete the activities created.

Scores

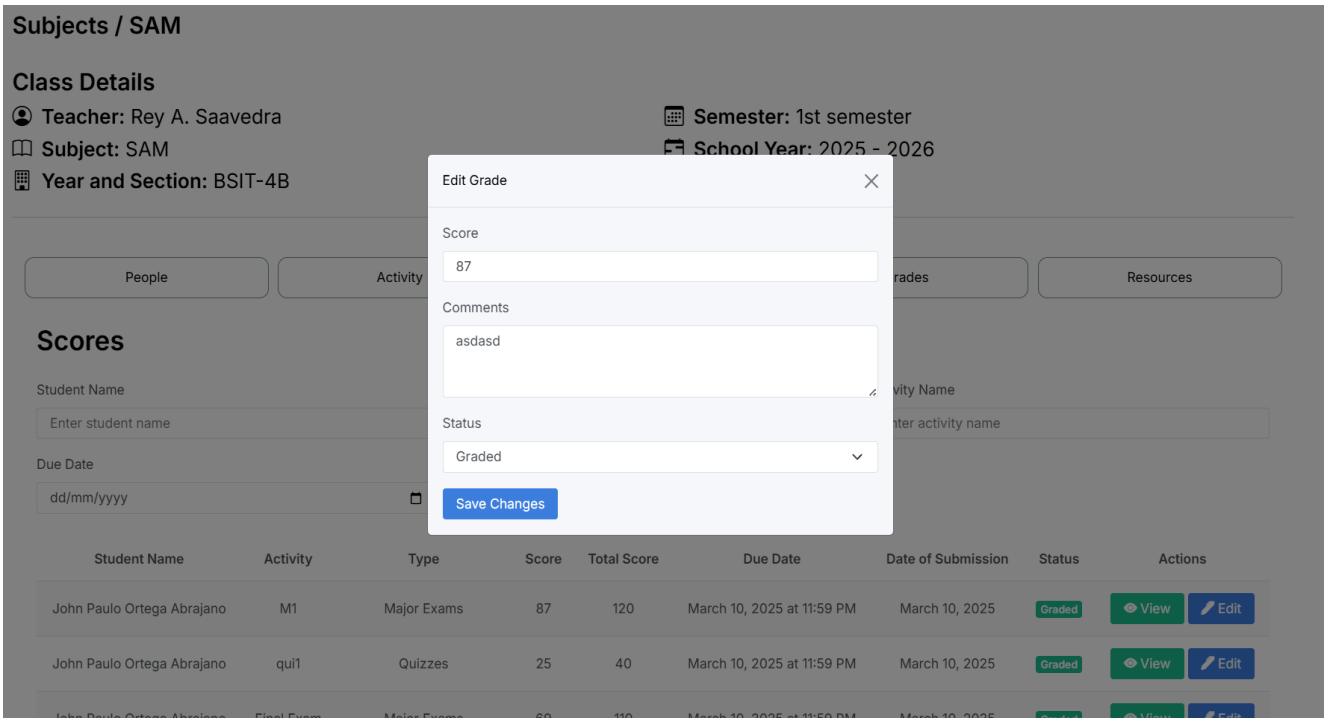


Figure 6.6

Scores section

In this section it allows the teacher to view and edit the activities of the students and put a grade in it.

Grades

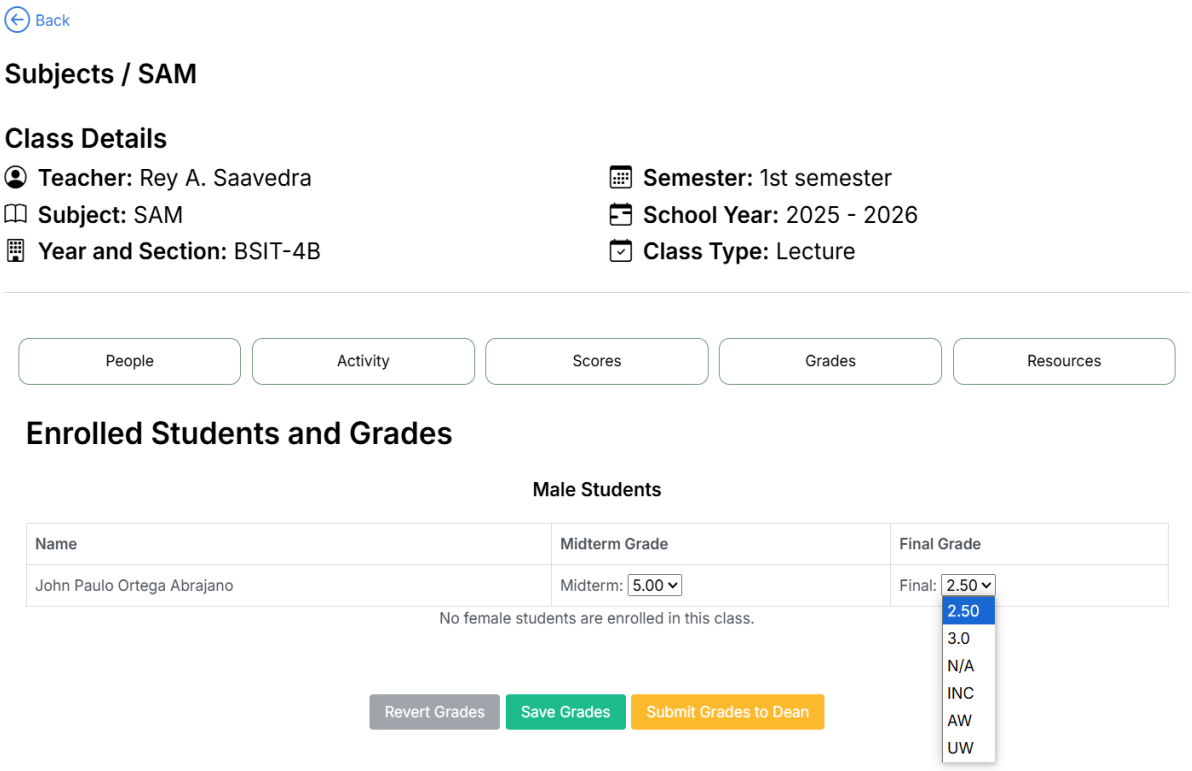


Figure 6.7

Grading Section

- In this section the teacher can overview the final average of the students that the system automatically computed the grades through its performance.
- The teacher also can manipulate the student's status if it is NA, INC, AW, or UW.
- The teacher can now save the grades and submit the grades to the dean for approval for finalization.
- The teacher also can revert the grades if there are any changes.

Resources

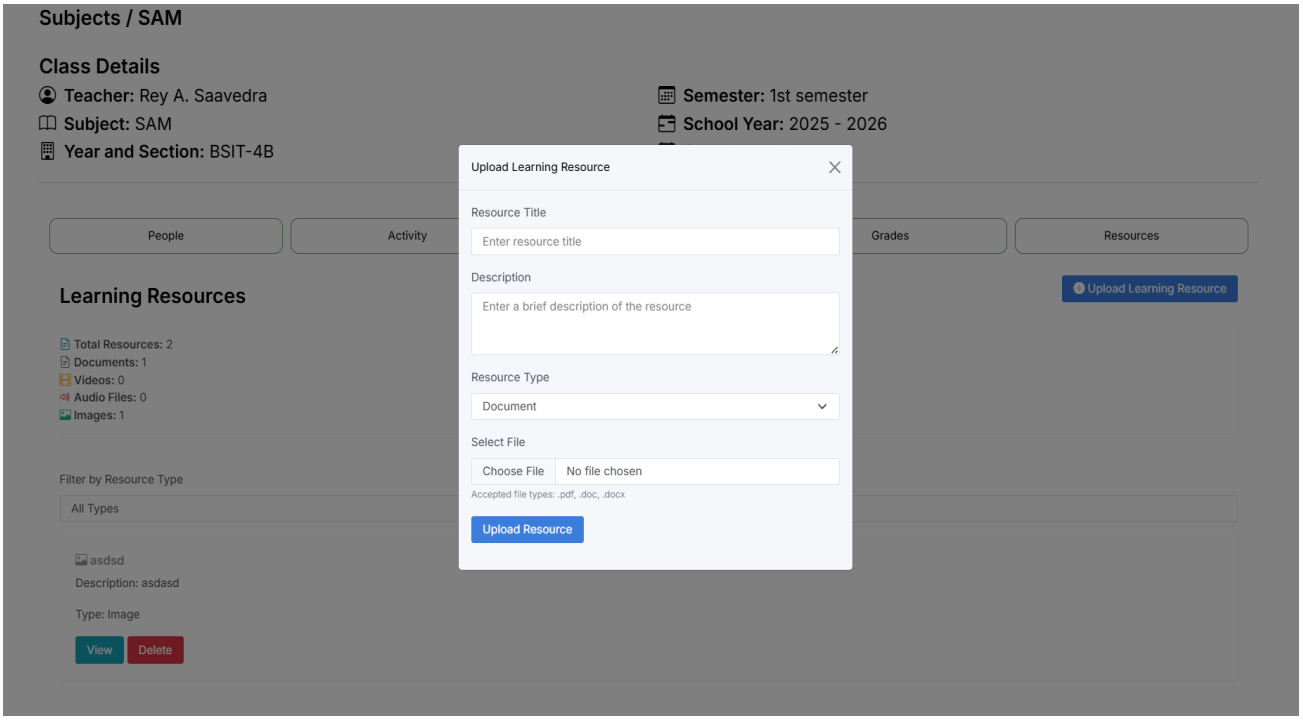


Figure 6.8

Learning Resources

In this it allows the teacher to provide learning resources for the students

- Click the “upload learning resources” button and fill the information such as resource title, description, resource type if it is document, video, image, or audio, and choose a file to upload.
- Then click “upload resources” to save the materials.
- The teacher can view and delete the materials uploaded.

Teacher User

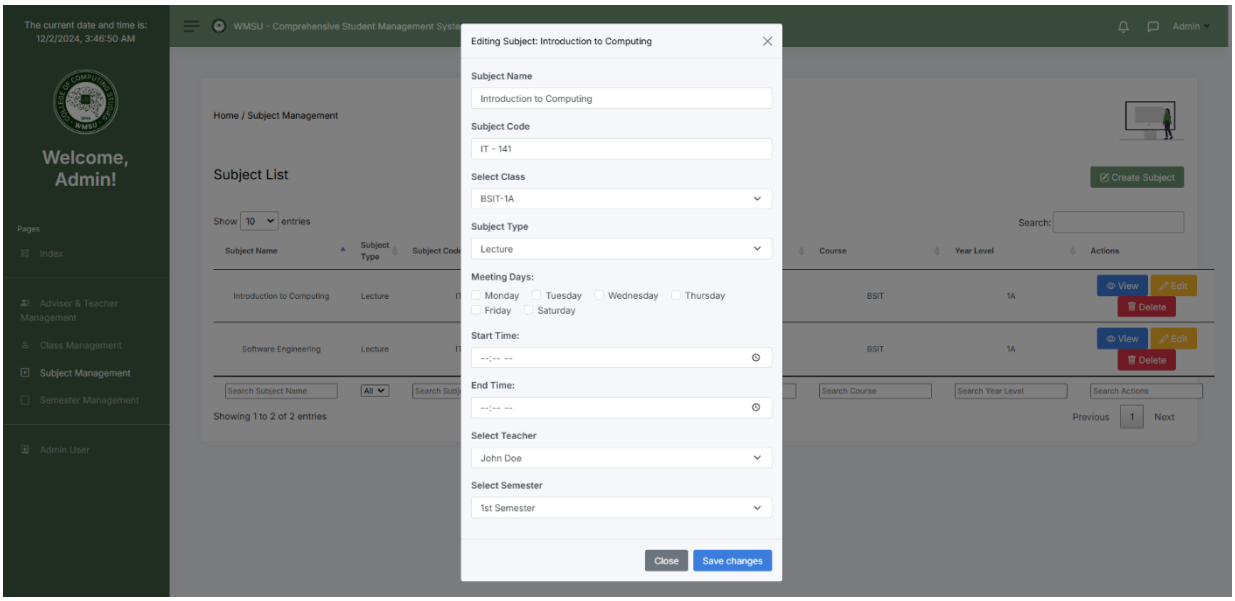
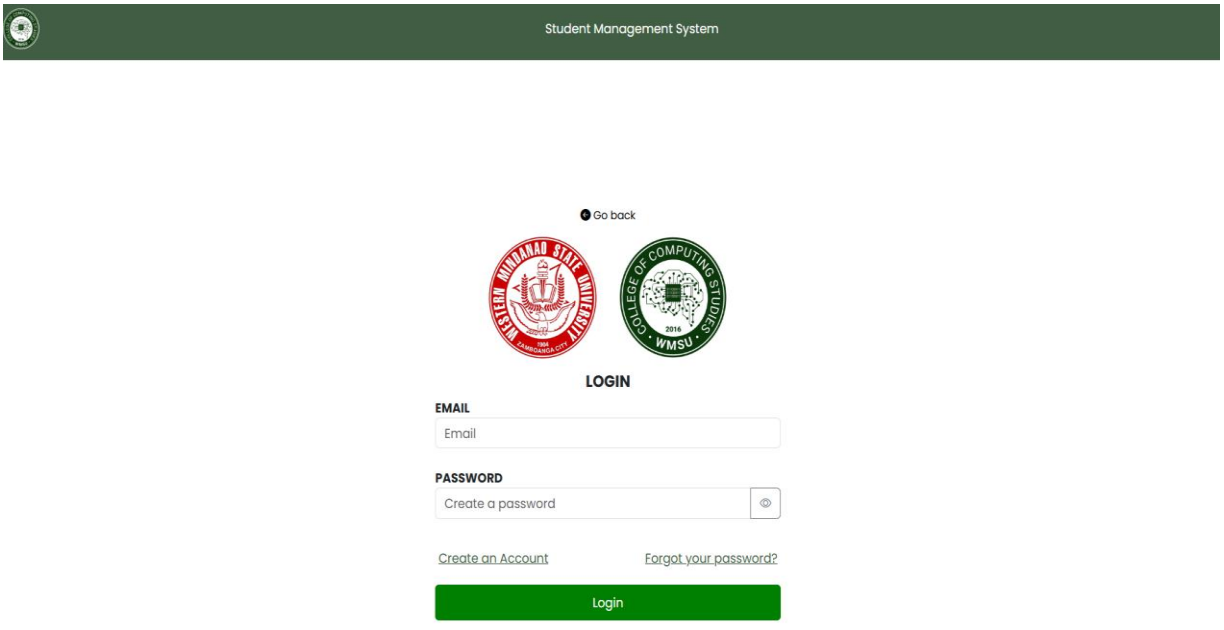


Figure 6.9

Staff Information

In this section it allows the teacher to view their credentials created by the admin

Student Login Page/Creation



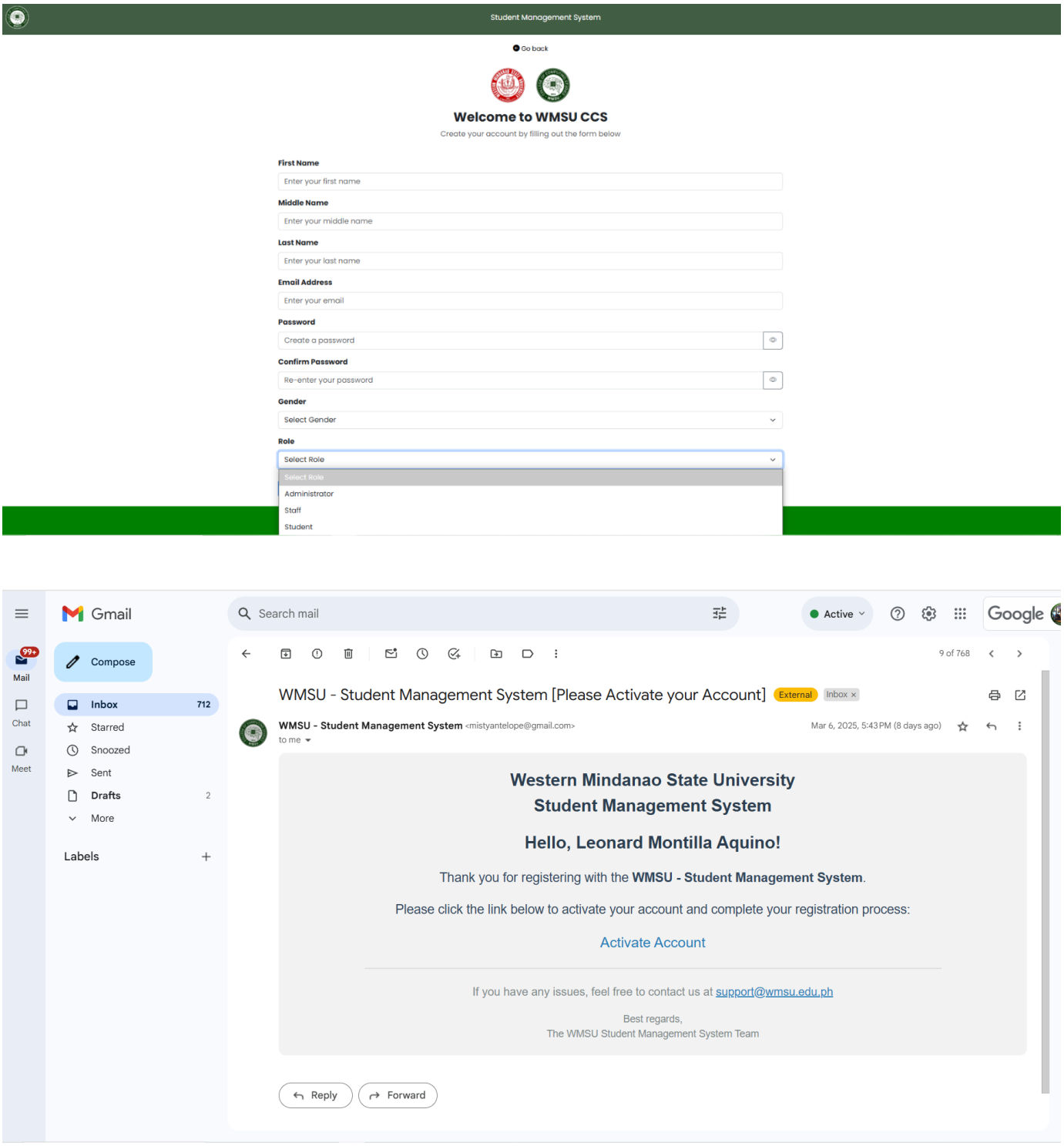


Figure 7.0

Student Log in Page/Creation

This section allows the student to create an account by inputting their wmsu account credentials

- Enter the student first name, middle name, last name, WMSU email Address, password, confirm

password, gender, and role.

- After filling up the information click the register button to register the account.
- Open your google account and click “activate” to activate your account in the system.
- Go back to the login page and enter your user credentials created in the registration
- Input your “wmsu email” and “password” then click login.

Student Dashboard

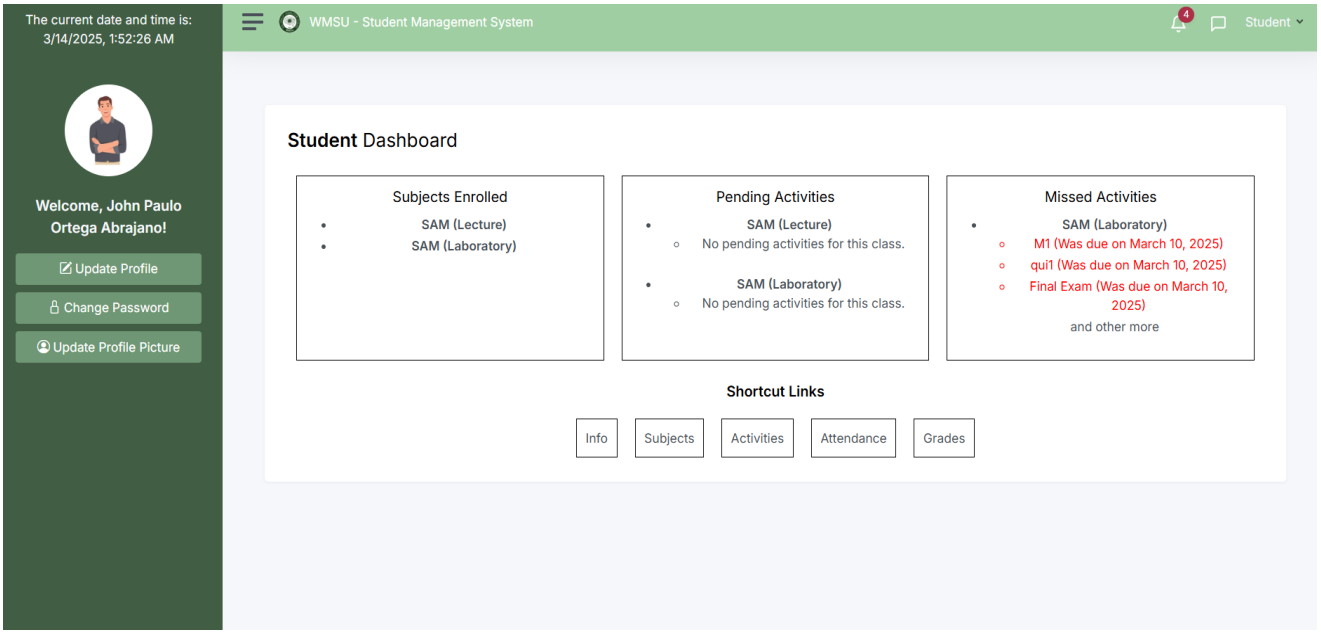
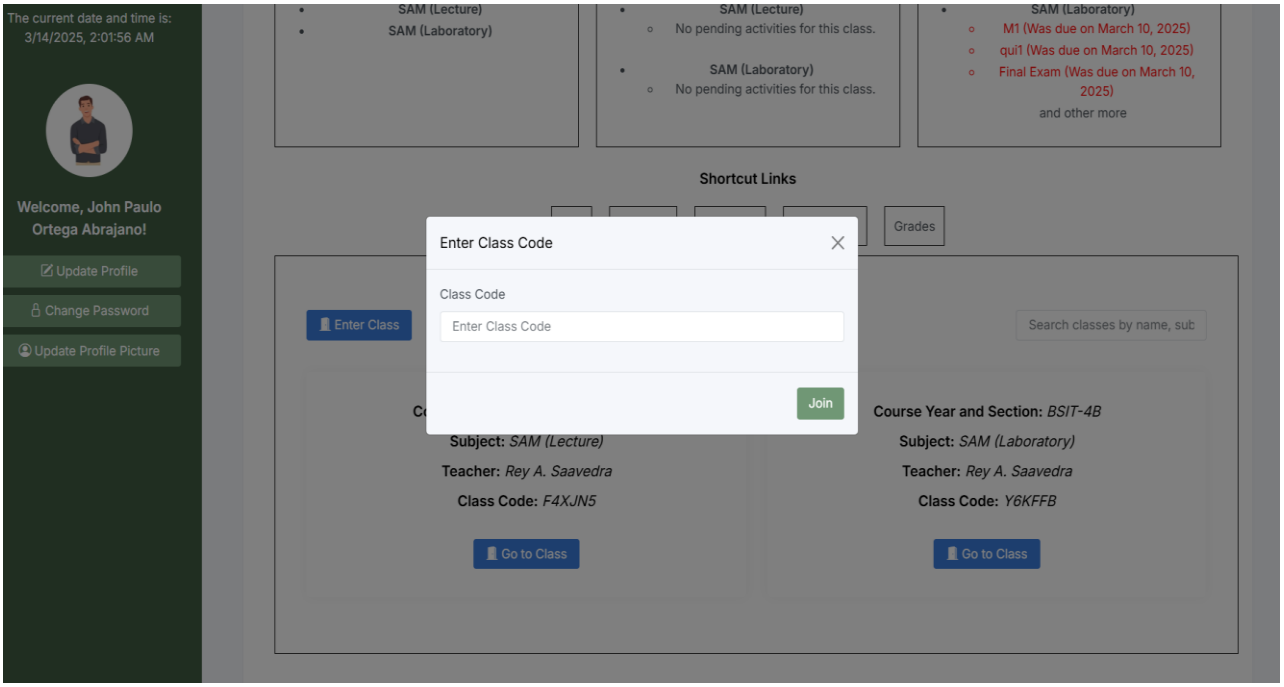


Figure 7.1

Student Dashboard

This is the student dashboard of the students where it shows the information in subject enrolled on the class, the pending and missed activities and short links, containing the information of the students, subject enrolled, activities to comply, attendance record, and grades. There are 3 buttons in side nav bar, the update profile, to update the information, change password and update profile picture.

Subjects



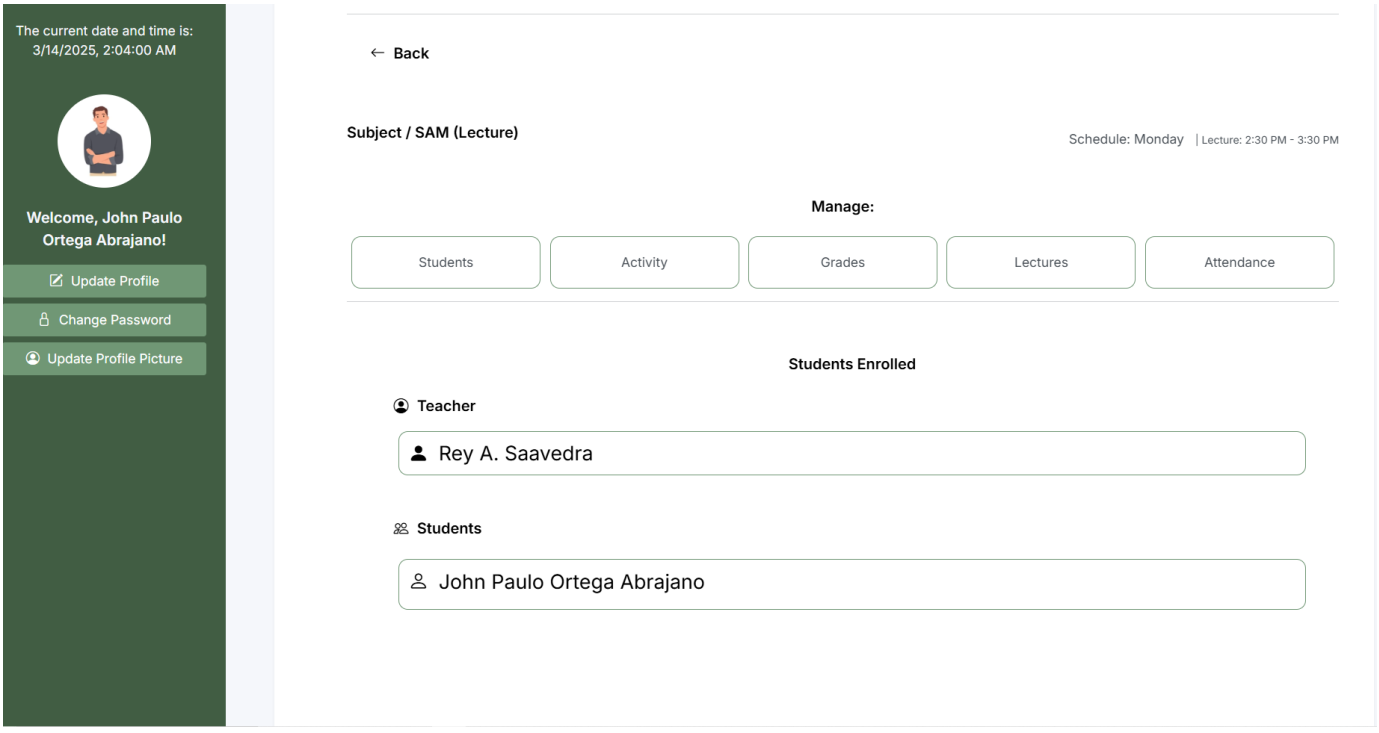


Figure 7.2

Enter Class Code

The system allows the students to enter the class code in order for them to enrolled in the class

- After inputting the class code, the student may now accessed the subject showing the number of the students enrolled in the class, the activities, grades, lectures, and attendance.

Activity

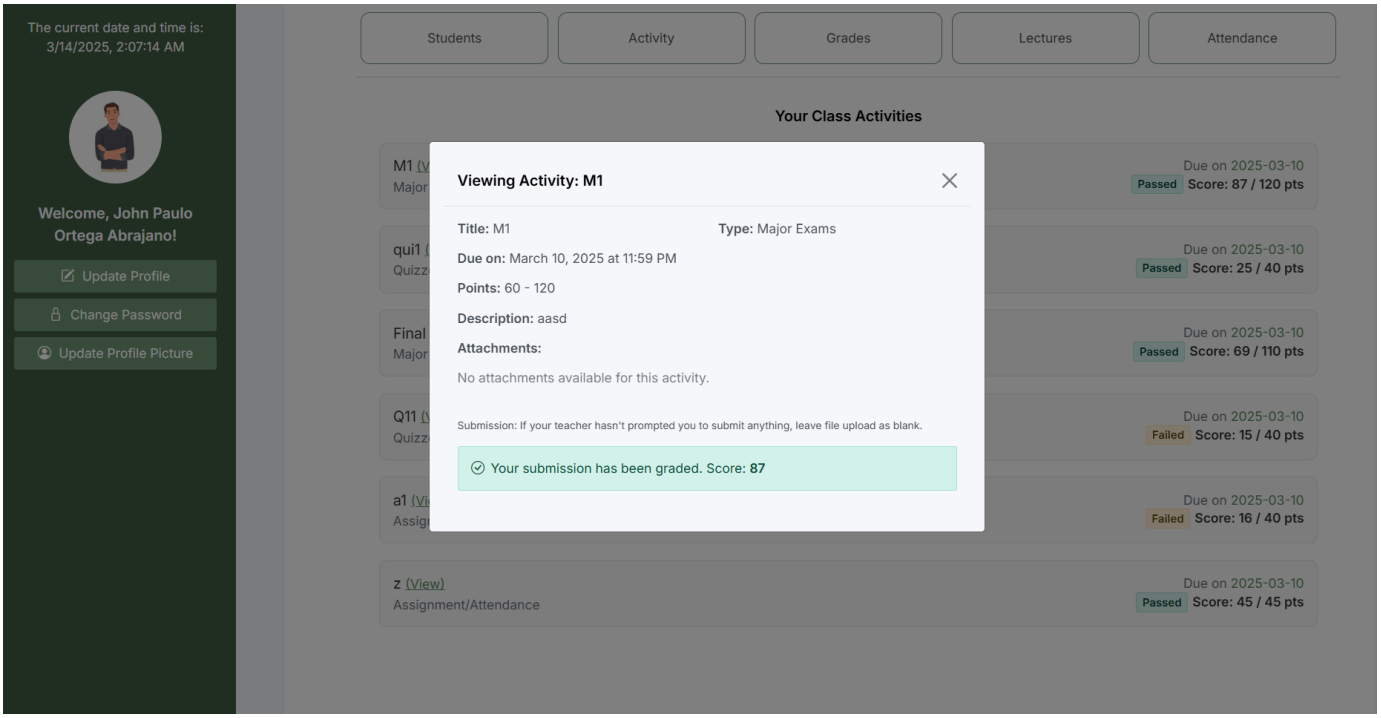


Figure 7.2

Activity Section

In this section the students can view the activity created by their adviser.

- This allow the students to see the activities set by their adviser
- The student must submit first their activity to be graded by the teacher and later they can see their results.

Grades

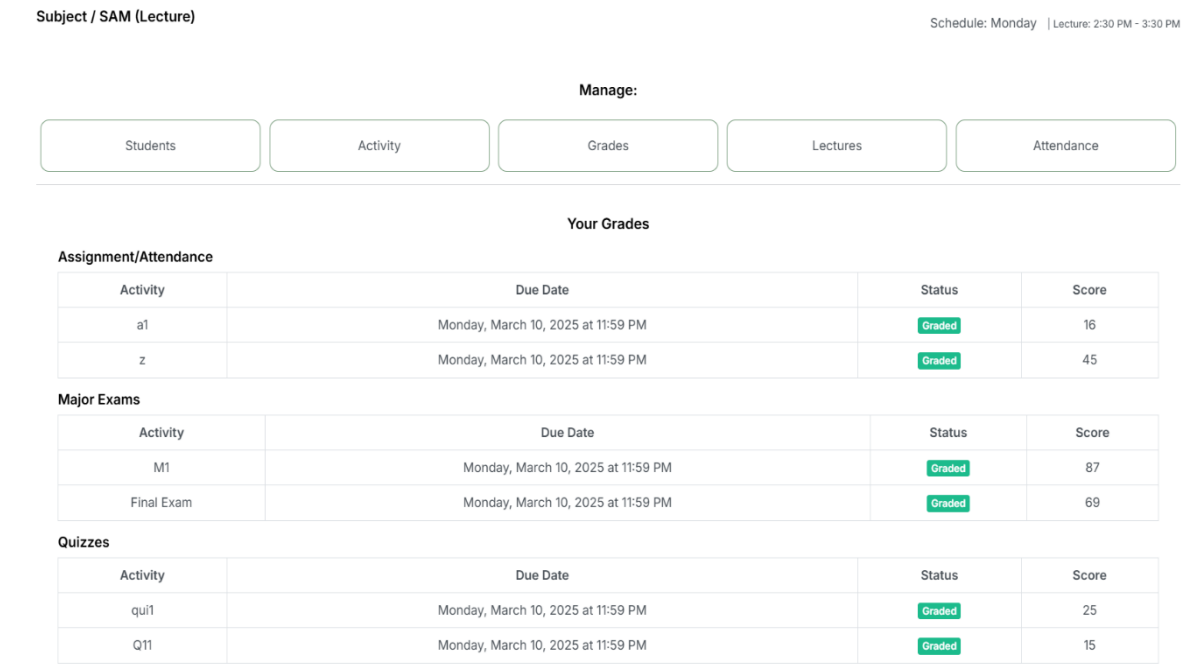


Figure 7.3

Grades Section

This Section allows the students to view their grades recorded by their teacher.

Lectures

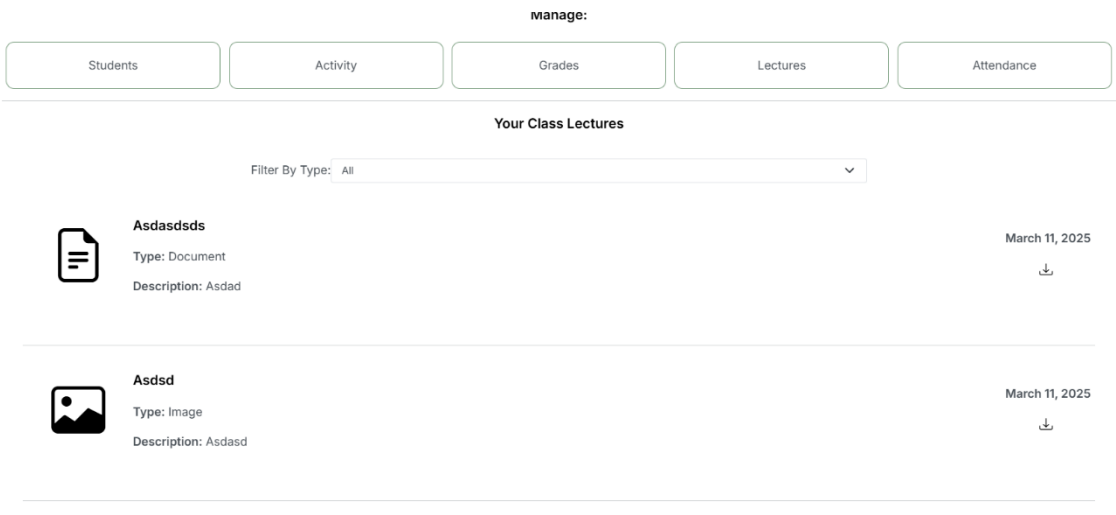


Figure 7.4

Lecture Materials Resources

This section allows the student to view the learning materials resources provided by their adviser for them to study or review.

Attendance

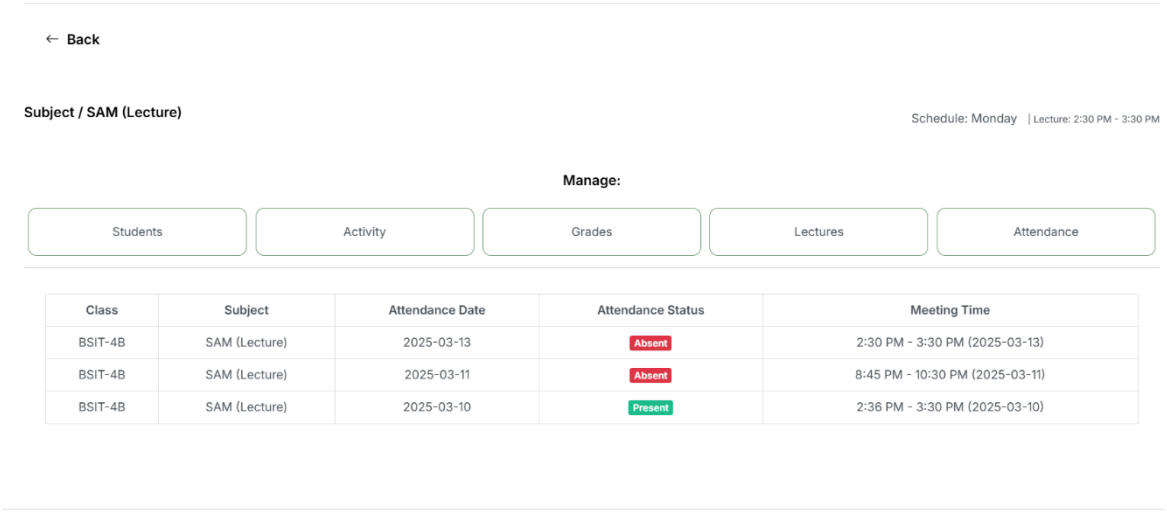


Figure 7.5

Attendance

This section allows the student to view their attendance record from the first day of class conducted to end of semester.

Shortcut Links

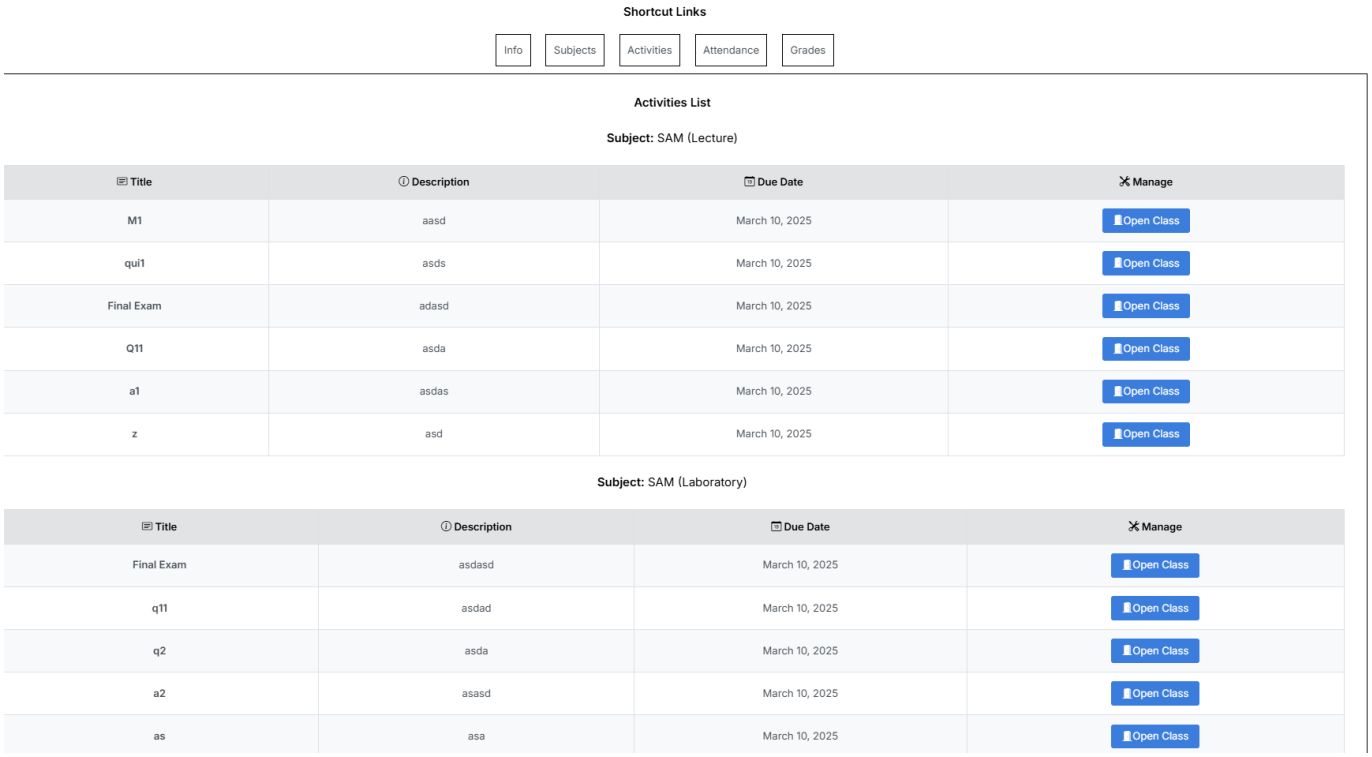


Figure 7.6

Activities

This sections allows the student to view directly the assigned activities for them not to miss any of it.

Grades

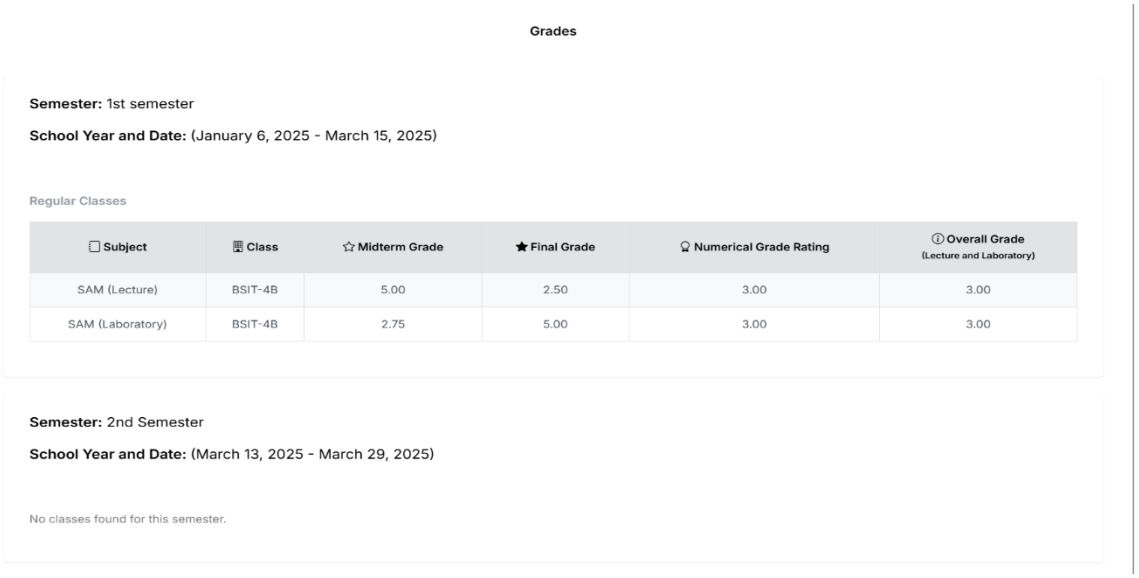


Figure 7.7

Grades

This section allows the student to view their indicating the overall grades from both laboratory and lecture for the students to have a reference for all of their performance.

Communication Tools

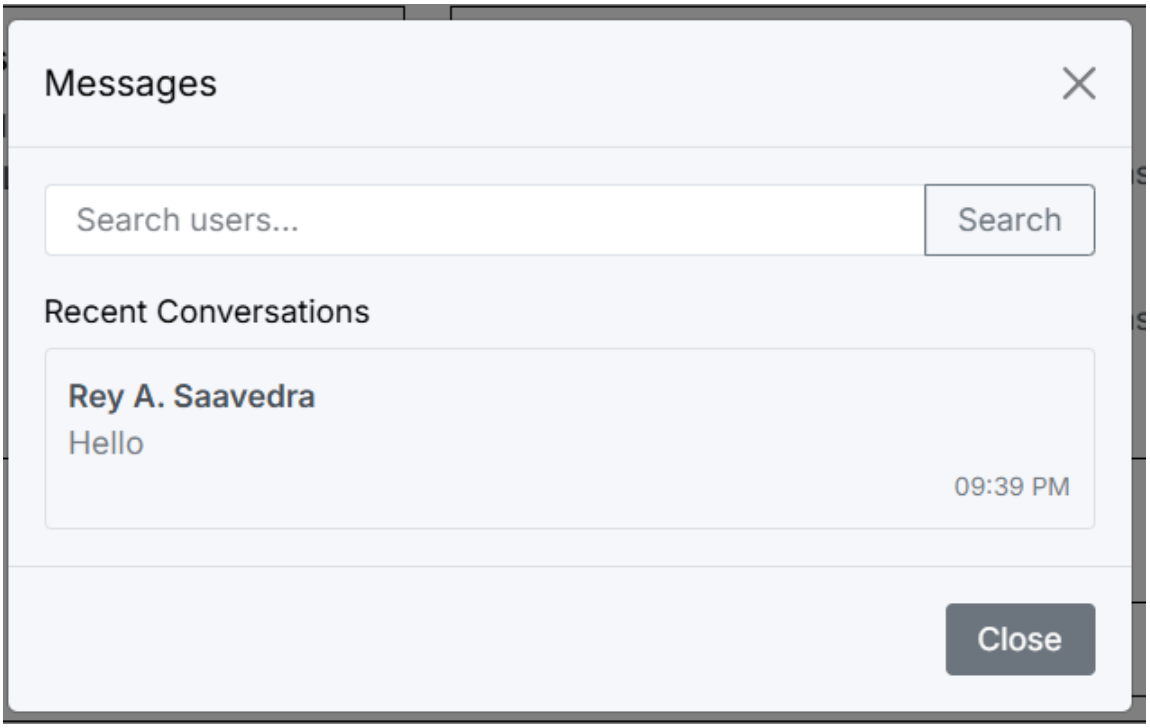


Figure 7.8

Communication Tools

This tool provides the users to have interaction with the other users such as admin to staff, then teacher to students.

Notification Tools

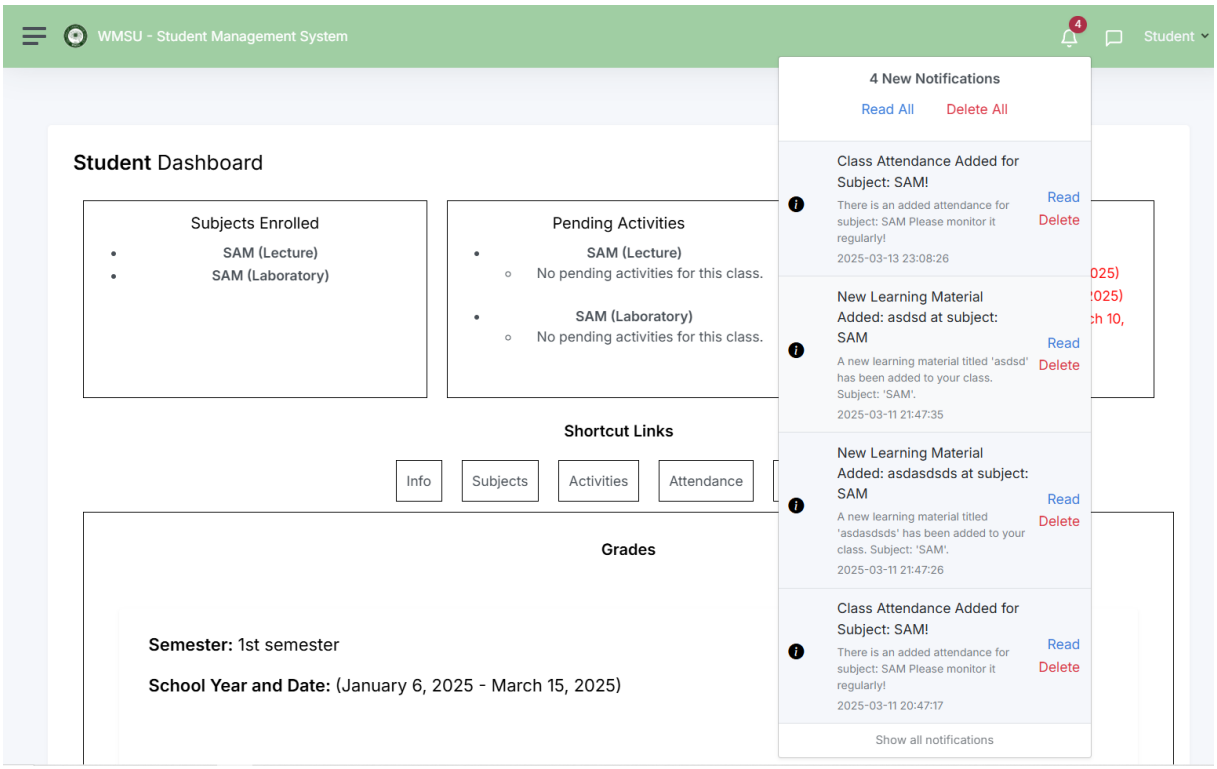


Figure 7.9

Notification Tools

This feature allows all of the users to notified all of the activities conducted inside the system, to help them not to miss any of it.

CHAPTER 5

DEVELOPMENT AND TESTING

5.1 Development

This project utilized the Agile Methodology, a dynamic approach emphasizing iterative progress, stakeholder feedback, and user-centered design. The development followed key stages, starting with Planning and Requirement Gathering, where detailed functional and non-functional requirements were documented after consultations with stakeholders. The Design phase focused on creating system and database architecture alongside user interfaces that prioritized accessibility and ease of use. During Coding, the team employed modern programming languages and frameworks such as PHP, MySQL, HTML, CSS, and JavaScript, with a modular approach for scalability and maintainability. In the Testing phase, unit, integration, and system testing ensured component compatibility, while user acceptance testing validated that the system met stakeholder needs. Finally, the Deployment phase involved configuring hosting environments, establishing databases, and publishing the system in both controlled and live settings. Team roles were clearly defined, with Programmers handling coding and module integration, System Analysts managing requirements alignment, and Data Gatherers ensuring accurate data collection and validation. Challenges such as implementing complex algorithms and ensuring a user-friendly interface were addressed through the use of established frameworks and iterative feedback processes. Development tools included IDEs like Visual Studio Code and version control with Git, while environments ranged from local development to live server testing, with considerations for mobile accessibility.

5.2 Integration of Technology

The project integrated PHP, MySQL, HTML, CSS, and JavaScript to create a web-based application. PHP powered the backend, while MySQL managed data storage and retrieval efficiently. The front end used HTML for structure, CSS for design, and JavaScript for dynamic interactions, ensuring a responsive and user-friendly interface.

Data flowed seamlessly between the web application and the database, with PHP scripts handling input and output. Frameworks like Bootstrap and jQuery enhanced responsiveness and simplified JavaScript tasks.

Security measures, including input validation, encryption, and safeguards against SQL injection, ensured the integrity and confidentiality of user data. This combination of technologies provided a cohesive, secure, and efficient system.

5.3 Testing

Testing is crucial phase in the development process to ensure the system functions correctly and meet user requirements. This decision is divided into (3) three sub-sections to cover different testing stages:

Focus Areas for Testing

1. Hardware Reliability

- Power Management: Testing the system's ability to operate under different power conditions, including server outages and power fluctuations.
- Connectivity Stability: Evaluating the reliability of the system's connection to the database and other backend services under various network conditions.
- Device Compatibility: Ensuring the system works seamlessly on a range of devices (desktop, tablet, mobile).

2. Software Usability and Functionality

- System Latency: Measuring the time it takes for grades, attendance, or profile updates to be processed and displayed in real-time.
- Error Handling: Testing how the system manages missing or invalid data entries, failed uploads, or incomplete requests.
- Data Integrity: Verifying the correctness, consistency, and reliability of student data (grades, attendance, profiles) during creation, retrieval, and modification.

3. Notification Mechanism

- Threshold Testing: Testing notification triggers for low attendance warnings or deadline reminders to ensure they activate at appropriate conditions.
- Notification Accuracy: Ensuring alerts (email/SMS) are sent to the correct users with clear, actionable messages.
- Redundancy Checks: Verifying fail-safes for notification mechanisms, such as retrying failed notifications.

4. User Interface (UI) and User Experience (UX)

- Ease of Use: Testing the intuitiveness of the student, teacher, and admin dashboards, ensuring effortless navigation and task execution.
- Accessibility: Ensuring the system is accessible across browsers and devices, including screen

readers and other assistive technologies.

- Consistency: Verifying that the UI maintains a consistent design and layout across all modules and user roles.

5. Security and Privacy

- Authentication: Ensuring that login processes are secure, passwords are hashed, and role-based access is implemented.
- Data Privacy: Testing that sensitive student data (grades, attendance records, personal info) is accessible only to authorized users.
- Attack Prevention: Verifying the system's resilience to attacks such as SQL injection, XSS (Cross-Site Scripting), and brute force.

6. Academic Modules

- Grade Management: Testing that grades can be entered, modified, and displayed accurately and that cumulative results are calculated correctly.
- Attendance Management: Verifying that attendance records are accurate, including QR code scanning functionality.
- Activity Submissions: Ensuring students can upload assignments successfully and that submissions are retrievable by teachers.

5.3.1 Alpha Testing

In terms of Alpha Testing, the development team utilized (3) laptops and (1) desktop computer connected via a localhost connection exposed via XAMPP's server using the I.P address to monitor and test the overall functionality of the Student Management System between the admin, teacher and student side.

As per se the requirements of the systems used indicated in the document in 3.2 for Hardware Requirements, all of the system has had stronger capabilities in terms of their hardware processes where the testers have had little to no problems in terms of loading, processing and fetching of data.

The development team conducted the Alpha Testing in Nifled's household where they had setup it in a Local Area Network infrastructure connected via a wireless router where John Paulo O. Abrajano acted as the admin side, Nifled Jericho Sespene acted as the teacher side and Leonard Aquino acted as the student side.

Below are the illustrations of the system being utilized for alpha testing.

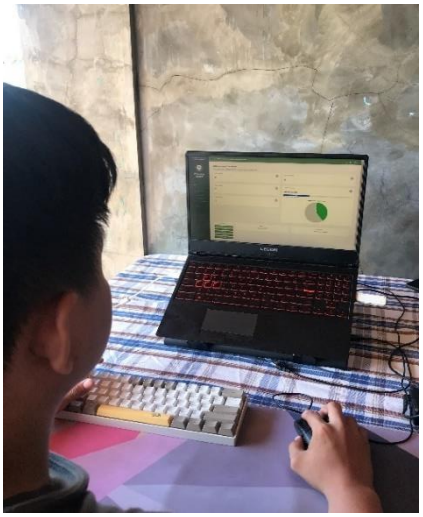


Figure 10: Admin Testing

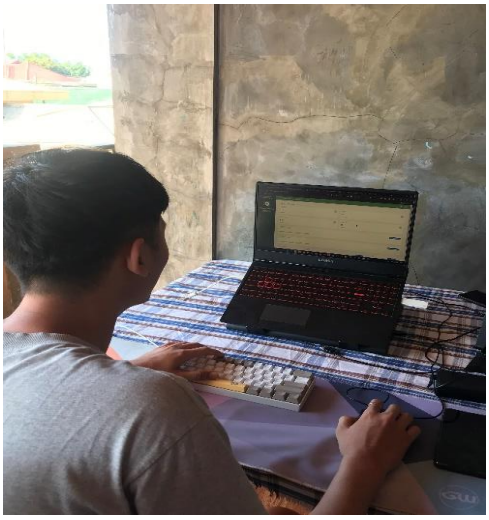


Figure 10: Staff Testing

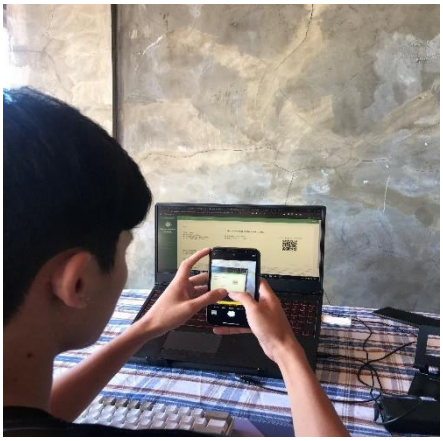


Figure 10: Student Testing

5.3.1.4 Unit Testing

TABLE 27: Unit Testing

Unit Testing	
Project Number:	1.0
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Type:	Unit Testing
Test Description:	Unit testing is a method of testing individual units or components of a software application.
Date:	From December 02, 2024 to December 08, 2024
App Version No:	N/A
Total Duration:	7 days
Total no. of passed test cases:	**6**
Total no. of failed test cases:	**0**
Total no. of tested test cases:	**0**
Total no. of test cases:	**6**

STATUS	
% Passed Test Cases	100%
% Failed Test Cases	0%
% Tested Test Cases	100%

A. Login Module

Login Module					
Project Name:		Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.			
Test Case #:		1			
Type of Test Case:		Unit Testing			
Test Case Description:		The user will log in to the system using email and password.			
Date Created		December 02, 2024			
Completed Date:		December 02, 2024			
Executed By:		John Paulo O. Abrajano			
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
TC001	Admin enters valid email and password	Admin successfully logs in and is redirected to the Admin Dashboard screen.	Admin enters valid credentials; system redirects to Admin Dashboard.	Pass	User is able to log in as Admin.
TC002	Teacher enters valid email and password	Teacher successfully logs in and is redirected to the Teacher Dashboard screen.	Teacher enters valid credentials; system redirects to Teacher Dashboard.	Pass	User is able to log in as Teacher.
TC003	Student enters valid email and password	Student successfully logs in and is redirected to the Student Dashboard screen.	Student enters valid credentials; system redirects to Student Dashboard.	Pass	User is able to log in as Student.
B. Declined Scenario					
TC004	Admin enters invalid email and/or incorrect password	Login fails, and an error message "Invalid email or password" is displayed.	Admin enters invalid credentials; system displays error: "Invalid email or password".	Pass	Login fails due to incorrect credentials.
TC005	Teacher enters invalid email and/or incorrect password	Login fails, and an error message "Invalid email or password" is displayed.	Teacher enters invalid credentials; system displays error: "Invalid email or password".	Pass	Login fails due to incorrect credentials.
TC006	Student enters invalid email and/or incorrect password	Login fails, and an error message "Invalid email or password" is displayed.	Student enters invalid credentials; system displays error: "Invalid email or password".	Pass	Login fails due to incorrect credentials.

B. Logout Module

Logout Module					
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.				
Test Case #:	2				
Type of Test Case:	Unit Testing				
Test Case Description:	The user will log out from the system.				
Date Created	December 02, 2024				
Completed Date:	December 02, 2024				
Executed By:	John Paulo O. Abrajano				
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
TC007	Admin logs out from the web application by clicking the Log Out button	Admin is logged out and redirected to the login screen.	Admin logs out using browser; system logs out and redirects to login page, receiving a logout successful dialog box.	Pass	Logout functionality works on a browser.
TC008	Teacher logs out from the web application by clicking the Log Out button	Teacher is logged out and redirected to the login screen.	Teacher logs out using browser; system logs out and redirects to login page, receiving a logout successful dialog box.	Pass	Logout functionality works on a browser.
TC009	Student logs out from the web application by clicking the Log Out button	Student is logged out and redirected to the login screen.	Student logs out using browser; system logs out and redirects to login page, receiving a logout successful dialog box.	Pass	Logout functionality works on a browser.

C. Account Creation Module

Account Creation Module	
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Case #:	3
Type of Test Case:	Unit Testing
Test Case Description:	The user can create an account based on a chosen role.

Date Created		December 02, 2024			
Completed Date:		December 02, 2024			
Executed By:		John Paulo O. Abrajano			
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
TC010	Admin enters valid details for account creation: email, password and other important credentials	Admin account is successfully created, and user is redirected for Login.	Admin enters valid details; system creates account and redirects to Login Page	Pass	Account created successfully.
TC011	Teacher enters valid details for account creation: email, password and other important credentials	Teacher account is successfully created, and user is redirected for Login.	Teacher enters valid details; system creates account and redirects to Login Page	Pass	Account created successfully.
TC012	Student enters valid details for account creation: email, password and other important credentials	Student account is successfully created, and user is redirected for Login.	Student enters valid details; system creates account and redirects to Login Page	Pass	Account created successfully.
A. Declined Scenario					
TC013	Admin attempts to create an account with an already taken email.	Error message: "Email already taken. Please choose another."	Admin attempts to create account with a taken username; system displays error: "Email already taken."	Pass	Email uniqueness validation works.
TC014	Teacher attempts to create an account with an already taken email.	Error message: "Email already taken. Please choose another."	Teacher attempts to create account with a taken username; system displays error: "Email already taken."	Pass	Email uniqueness validation works.
TC015	Student attempts to create an account with an already taken email.	Error message: "Email already taken. Please choose another."	Student attempts to create account with a taken username; system displays error: "Email already taken."	Pass	Email uniqueness validation works.

D. User Account Module

User Account Module					
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.				
Test Case #:	3				
Type of Test Case:	Unit Testing				
Test Case Description:	This feature allows the user to create, update, and search user accounts. This feature is created for security purposes and access level permission. The user of the system is the system administrator, QMS officer, internal auditor, and client.				
Date Created	December 02, 2024				
Completed Date:	December 02, 2024				
Executed By:	John Paulo O. Abrajano				
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
UA-ADM-01	Admin logs in with valid username and password.	Admin is redirected to the admin dashboard.	Admin is redirected to the admin dashboard.	Pass	Ensure role-specific dashboards load.
UA-ADM-02	Admin creates a new student account with valid data.	Student account is created with a confirmation message.	Student account is created with a confirmation message.	Pass	Validate account creation process.
UA-ADM-03	Admin updates their profile information.	Profile updates are saved and displayed immediately.	Profile updates are saved and displayed immediately.	Pass	Verify update queries function properly.
B. Declined Scenario					
UA-ADM-04	Admin enters an invalid password during login.	Error message: "Invalid credentials."	Error message: "Invalid credentials."	Pass	Test error messages for clarity.
UA-ADM-05	Admin attempts to create a user with missing fields.	Error message: "All fields are required."	Error message: "All fields are required."	Pass	Ensure all mandatory fields are validated.
UA-ADM-06	Admin uses an invalid email format during profile update.	Error message: "Invalid email address."	Error message: "Invalid email address."	Pass	Test for email format validation.

E. User Account Module Teacher

User Account Module (Teacher)					
Project Name:		Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.			
Test Case #:		2			
Type of Test Case:		Unit Testing			
Test Case Description:		This module allows the user to create, update, and search user accounts. This feature is created for security purposes and access level permission. The user of the system is the system administrator, QMS officer, internal auditor, and client.			
Date Created		December 03, 2024			
Completed Date:		December 03, 2024			
Executed By:		John Paulo O. Abrajano			
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
UA-TEA-01	Teacher logs in with valid credentials.	Teacher is redirected to their dashboard.	Teacher is redirected to their dashboard.	Pass	Ensure proper access control for teachers.
UA-TEA-02	Teacher updates their password.	Password is updated successfully with confirmation message.	Password is updated successfully with confirmation message.	Pass	Verify password hashing mechanism.
UA-TEA-03	Teacher views and edits their profile information.	Profile updates are saved and displayed.	Profile updates are saved and displayed.	Pass	Ensure secure update queries.
B. **Declined Scenario**					
UA-ADM-04	Admin enters an invalid password during login.	Error message: "Invalid credentials."	Error message: "Invalid credentials."	Pass	Test error messages for clarity.
UA-ADM-05	Admin attempts to create a user with missing fields.	Error message: "All fields are required."	Error message: "All fields are required."	Pass	Ensure all mandatory fields are validated.
UA-ADM-06	Admin uses an invalid email format during profile update.	Error message: "Invalid email address."	Error message: "Invalid email address."	Pass	Test for email format validation.

F. User Account Module (Admin)

User Account Module (Admin)					
Project Name:		Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.			
Test Case #:		2			
Type of Test Case:		Unit Testing			
Test Case Description:		This module allows the user to create, update, and search user accounts. This feature is created for security purposes and access level permission. The user of the system is the system administrator, QMS officer, internal auditor, and client.			
Date Created		December 03, 2024			
Completed Date:		December 03, 2024			
Executed By:		John Paulo O. Abrajano			
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
UA-ADM-01	Admin logs in with valid username and password.	Admin is redirected to the admin dashboard.	Admin is redirected to the admin dashboard.	Pass	Ensure role-specific dashboards load.
UA-ADM-02	Admin creates a new student account with valid data.	Student account is created with a confirmation message.	Student account is created with a confirmation message.	Pass	Validate account creation process.
UA-ADM-03	Admin updates their profile information.	Profile updates are saved and displayed immediately.	Profile updates are saved and displayed immediately.	Pass	Verify update queries function properly.
B. **Declined Scenario**					
UA-ADM-04	Admin enters an invalid password during login.	Error message: "Invalid credentials."	Error message: "Invalid credentials."	Pass	Test error messages for clarity.
UA-ADM-05	Admin attempts to create a user with missing fields.	Error message: "All fields are required."	Error message: "All fields are required."	Pass	Ensure all mandatory fields are validated.
UA-ADM-06	Admin uses an invalid email format during profile update.	Error message: "Invalid email address."	Error message: "Invalid email address."	Pass	Test for email format validation.

G. User Account Module (Student)

User Account Module (Student)					
Project Name:		Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.			
Test Case #:		2			
Type of Test Case:		Unit Testing			
Test Case Description:		This module allows the user to create, update, and search user accounts. This feature is created for security purposes and access level permission. The user of the system is the system administrator, QMS officer, internal auditor, and client.			
Date Created		December 03, 2024			
Completed Date:		December 03, 2024			
Executed By:		John Paulo O. Abrajano			
Test Case ID	**Test Case Scenario / Condition**	**Expected Result**	**Actual Result**	**Status**	**Remarks**
A. Successful Scenario					
UA-STU-01	Student registers with valid email, password, and name.	Account is created with a confirmation message.	Account is created with a confirmation message.	Pass	Validate uniqueness of email address.
UA-STU-02	Student logs in using valid credentials.	Student is redirected to their dashboard.	Student is redirected to their dashboard.	Pass	Ensure student-specific dashboard loads.
UA-STU-03	Student updates their profile picture.	Profile picture is uploaded and displayed correctly.	Profile picture is uploaded and displayed correctly.	Pass	Validate image file type and size.
B. **Declined Scenario**					
UA-STU-04	Student enters mismatched passwords during registration.	Error message: "Passwords do not match."	Error message: "Passwords do not match."	Pass	Validate client-side and server-side checks.
UA-STU-05	Student logs in with an inactive account.	Error message: "Account is inactive."	Error message: "Account is inactive."	Pass	Verify account activation checks.
UA-STU-06	Student uploads an image exceeding size limit (e.g., >5MB.	Error message: "File size exceeds limit."	Error message: "File size exceeds limit."	Pass	Ensure proper file upload validation.

5.3.1.5 Integration Testing

Integration Testing	
Project Number:	1.0
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Type:	Integration Testing
Test Description:	Integration testing is a method of testing how different units or components of a software application interact with each other.

Date:	**December 03, 2024**
App Version No:	N/A
Total Duration:	7
Total no. of passed test cases:	6
Total no. of failed test cases:	1
Total no. of tested test cases:	0
Total no. of test cases:	6

STATUS	
% Passed Test Cases	90%
% Tested Test Cases	10

E. User Account Module and Login Module

User Account Module and Log-in Module	
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Case #:	1
Type of Test Case:	Integration Testing
Test Case Description:	The Administrator with Log-in Module is integrated to be able to show if the username and password that is created by the admin is accessible in the log-in module.

Date Created	December 04, 2024		
Completed Date:	December 04, 2024		
Executed By:	John Paulo O. Abrajano		
Test Case	**Expected Result**	**Actual Result**	**Remarks**
Successful Scenario			
Integration of Log-in Module and Dashboard Module	After logging into the system, users should seamlessly access the dashboard and other features.	After logging into the system, users successfully access the dashboard and other features seamlessly.	Pass
Failed Scenario			
ser attempts to log in with incorrect credentials	Error message: "Invalid username or password."	Error message is displayed correctly: "Invalid username or password."	Pass
Admin account disabled attempts to log in	Error message: "Account is inactive. Please contact support."	Error message is displayed correctly: "Account is inactive. Please contact support."	Pass

5.3.1.6 System Testing

System Testing	
Project Number:	1.0
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Type:	System Testing
Test Description:	System testing tests all the processes of the system.

Date:	December 04, 2024
App Version No:	**1.0**
Total Duration:	**7**
Total no. of passed test cases:	**8**
Total no. of failed test cases:	**2**
Total no. of tested test cases:	**8**
Total no. of test cases:	**6**

STATUS	
% Passed Test Cases	80%
% Tested Test Cases	20%

F. Customer Complaint and Validation Process

Customer Complaint Validation process	
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Case #:	1
Type of Test Case:	System Testing
Test Case Description:	This test case verifies the system's functionality in automating the issuance of nonconformity through valid complaints.
Date Created	December 04, 2024
Completed Date:	December 04, 2024
Executed By:	John Paulo O, Abrajano
Expected Result:	Nonconformity will be automatically issued to the respective unit or office in response valid complaint.
Status:	
Remarks:	

Test Case ID	**Test Case Scenario / Condition**
A. Scenario	
Step 1	Customer Complaint Review - Examine the details of the customer complaint.

	Condition: Admin revies
Step 2	Verification and Logging - Verify the authenticity of the customer complaint and log the verification details.
Step 3	Nonconformity Issuance - If the complaint is valid, the system automatically issues a nonconformity.
Step 4	Verification - Confirm the issuance of the nonconformity report by checking the nonconformity list for the relevant office or unit.

5.3.2 Beta Testing

In terms of beta testing, the development team has had only the alpha testing done wherein the beta testing will occur within this following week.

The beta testing will have actual users such as dean as an admin, teachers from the faculty overseeing the system and students from College of Computing Studies using the system.

5.3.2.1 Selection of Beta Tester

The selection of beta testers was a crucial step in the testing process to ensure that the feedback received was comprehensive, relevant, and valuable for refining the system before its wider release. The goal was to gather insights from a diverse group of users who could assess the system's functionality, usability, and overall user experience from various perspectives. A carefully selected group of participants was chosen from different categories of stakeholders, including faculty members, students, and administrators at WMSU. The testers were selected not only based on their familiarity with the system's features but also on their willingness to actively engage in the testing process. To ensure that the feedback covered a wide range of use cases and experiences, the selection process considered the following criteria:

1. User Role Representation

Feedback was collected from a well-balanced group, representing administrators, teachers, and students. Each group provided unique insights to address their specific needs and system interactions.

- Administrators assessed functionality related to managing users, classes, and semester configurations.
- Teachers offered perspectives on handling attendance, grade management, and supplementary material uploads.
- Students provided feedback on accessing grades, attendance tracking, and user interface navigation.

2. Technical Proficiency and System Experience

A mix of users with varying levels of familiarity with digital academic tools participated in the testing.

- Experienced users compared features with other platforms and highlighted areas for enhancement.
- Inexperienced users assessed the intuitiveness and accessibility of the system for first-time interaction, ensuring that the system is user-friendly for all skill levels.

3. Constructive Feedback Willingness

Participants were selected based on their ability to provide detailed and actionable feedback.

- Testers provided both positive and critical assessments of system features, functionality, and overall usability.
- Their input guided the iterative improvement of the system, addressing usability issues and refining performance. Notifications and reminders are clear and timely.

5.3.2.2 Scope of Testing

The scope of beta testing was designed to encompass a wide range of functionalities to thoroughly assess the system's performance, usability, and overall effectiveness. The primary areas tested included:

1. **User Interface (UI) and Usability**

➢ Beta testers were tasked with evaluating the system's usability and interface intuitiveness. The assessment focused on the following key areas:

✓ Navigation

Testers examined the ease of navigating between modules such as grades, attendance, and supplementary materials. The goal was to ensure that the system's layout was intuitive and provided logical transitions between sections.

✓ Accessibility

The system was evaluated for its usability by users with different levels of technical expertise. The design aimed to accommodate novice users while still providing advanced functionality for experienced users.

✓ Mobile Compatibility

Given the necessity of mobile access, the system's responsiveness on various devices was tested. Beta testers ensured the interface adapted seamlessly to different screen sizes, from desktops to

smartphones, providing consistent functionality and usability.

This thorough assessment guaranteed that the platform met usability standards while addressing the diverse needs of its user base.

2. **Core Functionalities**

   This section was a key focus during beta testing to ensure that the system's primary features performed effectively for all user roles:

   ✓ Grade Management

   Teachers tested the grade input and computation features, ensuring that grades could be entered, updated, and displayed accurately. Students reviewed their grades for clarity and ease of access, while administrators verified the accuracy of grade storage and retrieval processes.

   ✓ Attendance Tracking

   The QR code-based attendance system was evaluated by teachers and students. Student scanned Class' QR codes to record attendance, while students checked their attendance records for accuracy. Testers confirmed the system's ability to handle real-time updates and generate attendance reports.

   ✓ Supplementary Material Management

   Teachers uploaded lecture materials and activities, ensuring that files were properly stored and accessible to students. Students verified that materials were easy to download and navigate.

   ✓ Role-Specific Functions

   Each user role was tested to confirm smooth operation of assigned tasks:

- **Students:** Successfully accessed grades, attendance, and lecture materials.
- **Teachers:** Managed classes, uploaded materials, and tracked attendance effectively.
- **Administrators:** Oversaw user accounts, class assignments, and ensured proper data management.

3. **Performance and Usability**

   To ensure the system's reliability under real-world conditions, beta testers evaluated its performance across various scenarios:

   ✓ Simultaneous Users

   Testers simulated multiple users logging in and accessing the system simultaneously. This assessed the platform's ability to maintain stability and responsiveness under heavy traffic.

✓ **Data Processing and Uploads**

Testers evaluated the system's responsiveness during the upload of student assignments, lecture materials, and large files. The goal was to confirm that the system could handle these uploads without experiencing significant delays or performance issues.

✓ **Search Functionality**

The efficiency of the search feature was tested by submitting multiple queries simultaneously. Testers assessed how quickly the system retrieved records, such as attendance logs or class schedules, and verified the accuracy of search results.

4. **Security**

✓ Security was a vital aspect of the beta testing process, with testers evaluating the system's ability to protect user data and prevent unauthorized access:

✓ **Restricted Access**

Testers attempted to access restricted areas, such as admin settings and confidential student or teacher data, to ensure that role-based access controls were functioning effectively and unauthorized access was blocked.

✓ **Input Validation**

Various forms of data, including special characters, SQL injection attempts, and invalid inputs, were submitted to test the robustness of input validation measures. This ensured that the system was protected against common security vulnerabilities.

5.3.2.3 **Method Used to Collect Feedback**

To ensure meaningful and actionable feedback during beta testing, a blend of structured and informal techniques was implemented. These approaches provided the development team with both quantitative metrics and qualitative user insights to improve the system effectively.

1. Bug Reports

Identifying and resolving technical issues was a core part of the beta testing process. Testers were encouraged to document bugs, errors, or crashes systematically. A dedicated reporting mechanism was established to:

Log Issues Promptly: Testers reported problems immediately after encountering them, ensuring timely tracking.

Detail Reproduction Steps: Reports included clear descriptions of the issues, reproduction steps, and supporting evidence like screenshots or logs.

Monitor Resolution Progress: A tracking system allowed testers to view the status of their reported issues, ensuring transparency and accountability.
This process allowed the development team to prioritize and resolve issues efficiently, contributing to a robust and reliable final product.

User Surveys and Feedback Forms

Testers provided structured feedback through surveys and forms, focusing on system usability, feature performance, and overall satisfaction. Questions covered topics like ease of navigation, response times, and the clarity of displayed information.

Observation and Interviews

Informal feedback was collected through observation and direct interviews. Beta testers shared their real-time experiences, providing insights into potential pain points and suggestions for enhancements.

2. Surveys

Structured surveys were provided to all beta testers to capture both quantitative and qualitative feedback.
Tester should answer these questioners.

Introduction

Dear Participants,

Thank you for participating in the evaluation of the Student Management System as part of out Capstone Project. Your Feedback is crucial for enhancing the system and ensuring it meets user expectations. We value your time and insights, which will greatly contribute to refining the system.

This evaluation uses a 4-point raising scale to capture your experience with various aspects of the system. Please choose the response that best reflects your experience. All responses will be kept confidential Thank you. Kindly return the completed form to the system proponents.

Instruction: Please indicate your response by checking the corresponding rating number.

Part I. Graphical User Interface	**Ratings**			
Questions	**1 (Strongly Disagree)**	**2 (Disagree)**	**3 (Agree)**	**4 (Strongly Agree)**
The design elements (e.g., buttons, icons) are consistent throughout the interface.				

The font size and style used in the GUI are clear and readable.				
Overall, I find the interface easy to use.				
The design of the system is visually appealing				
Font style and size is easy to read				
The colors used in the interface were pleasing and appropriate.				
Buttons and icons are easy to identify.				
The layout of the pages is consistent and well-structured.				
The onboarding process (e.g., log in, sign up) is straightforward.				
The interface provides a clear and logical workflow for completing tasks.				

Comments:

_____

_____

_____

_____

Instruction: Please indicate your response by checking the corresponding rating number.

Part II. Usability	**Ratings**			
Questions	**1 (Strongly Disagree)**	**2 (Disagree)**	**3 (Agree)**	**4 (Strongly Agree)**
The SMS is easy to learn.				
Error messages within the system are clear and informative.				
Overall, I am satisfied with the usability of the SMS.				
The system loads quickly when first accessed.				
The system responds quickly when interacting with the features.				
There were no delays in completing tasks.				
The system runs smoothly without any crashes.				
The system performs consistently well under normal use.				

The system is stable even with multiple users.				
The interface provides a clear and logical workflow for completing tasks.				
The navigation between different features (e.g., grades, attendance) is intuitive and efficient.				

Comments:

_____

_____

_____

Instruction: Please indicate your response by checking the corresponding rating number.

Part III. Functionality	**Ratings**			
Questions	**1 (Strongly Disagree)**	**2 (Disagree)**	**3 (Agree)**	**4 (Strongly Agree)**
The grade tracking feature is reliable and accurate.				
The attendance tracking feature is easy to use and effective.				
The repository for supplementary materials is intuitive and comprehensive.				
The User Account Login/Logout feature provides a reliable experience.				
The system effectively consolidates grades, attendance, and supplementary materials				
The Dashboard offers a clear and well-structured interface for users.				
Notifications and reminders are clear and timely.				
The dashboard provides clear and well-organized summaries of academic data.				
The search feature allows me to locate specific information (e.g., grades, subjects) easily and quickly.				

Comments:

Respondent: _____

Signature over printed name

3. **User Interviews**

   Follow-up interviews were conducted with selected beta testers to gain deeper insights into their experiences. These interviews aimed to:

   Explore Specific User Experiences: Delve into particular aspects of usability and interface design to understand user interactions and challenges.

   Gather Qualitative Insights: Obtain detailed feedback that surveys might not fully capture, providing a richer understanding of user perspectives.

   Solicit Improvement Suggestions: Encourage testers to propose enhancements for the interface, workflow, and additional features to boost user satisfaction.

   The interviews were semi-structured, allowing for open-ended responses to foster candid discussions about user experiences.

4. **Usage Analytics**

   To complement subjective feedback, usage analytics tools were integrated into the system to monitor:

   User Actions: Tracking which features were most frequently accessed, time spent on various sections, and completion rates of specific tasks.

   Problem Areas: Identifying sections where users faced difficulties, such as high abandonment rates during certain processes or prolonged load times.

   Behavioral Insights: Analyzing patterns like click rates, scrolling behavior, and navigation paths to understand user engagement and identify areas needing improvement.

   This data-driven approach provided objective insights into user interactions, highlighting issues that may not have been evident through direct feedback alone.

5.3.2.4 Goals of Beta Testing

The beta testing phase was designed to rigorously evaluate the system's functionality, performance, and user experience to ensure readiness for deployment. The following objectives were emphasized:

Usability Assessment

The testing aimed to verify that users could effortlessly navigate the platform, understand its features, and complete essential tasks, such as accessing grades, attendance records, and supplementary materials, without encountering confusion or usability challenges.

Performance Evaluation

The system was tested under real-world conditions, including scenarios with multiple simultaneous users. This ensured that the platform maintained speed, stability, and reliability while handling tasks

such as grade input, attendance tracking, and file uploads.

User Satisfaction Feedback

Beta testers provided feedback to determine whether the system met the expectations and needs of its stakeholders, including students, teachers, and administrators.

Bug and Issue Identification

The testing phase served as an opportunity to uncover bugs or technical issues that might have been overlooked during earlier development. Identified problems were documented and resolved to enhance system stability.

Feature Functionality Validation

Each core feature, including grade computation, attendance tracking via QR codes, and material uploads, was tested to confirm that they operated as intended and aligned with user requirements.

By achieving these goals, the beta testing phase ensured that the Student Management System was fully prepared for widespread implementation.

5.3.3 User Acceptance Testing

In alpha testing, all of the users who has had utilized 4 devices wherein it contained 3 laptops and 1 computer connected via a local area network connection where the users has had satisfactory experience in testing the overall system ranging from the website design, database design and infrastructure and system functionalities utilizing the overall seamless integration of learning experiences as an admin, teacher and student of College of Computing Studies in Western Mindanao State University.

In terms of beta testing in showcasing the overall user acceptance testing, the development team will roll out the system this week to further amplify the whole development and usage of the Student Management System.

5.4 Implementation Plan

The project was developed over a span of 150 days, from August 2024 to December 2024, beginning with the design of the user interface and continuing through the final development phase. The development process followed an iterative approach, which was established early in the project. Initially, the researchers proposed the project to the subject adviser, and upon receiving approval, the study commenced. To gather essential data, two types of questionnaires were created for two distinct groups of respondents: teachers and students. The implementation phase was broken into three main categories: data collection, user interface design, and the development phase. For data collection, the

team utilized Google Forms to gather responses. A brainstorming session was held to determine the design, font, and art style that would best align with the concept of the application. After collecting all necessary information, the development phase began, where all the data was incorporated into the system. Throughout the development, each module was tested multiple times to ensure no errors occurred. The testing process consisted of unit testing, which evaluated the accuracy of the source code; integration testing, where all modules were combined and tested as a whole; and acceptability testing, in which the system was reviewed by actual users to ensure it met their needs and expectations. This comprehensive development and testing process ensured the system was fully functional and user-friendly by the end of the project.

5.4.1 Infrastructure /Deployment

The system was deployed on a dedicated server with a multi-core processor, 16GB RAM, and 500GB SSD to handle user requests and database transactions. The server environment was set up using Apache HTTP Server, and MySQL for database management. PHP was configured for backend processing, and the application files were uploaded.

Backup procedures included daily automated backups to cloud storage, with recovery mechanisms in place for quick restoration. To ensure high availability and scalability, load balancing and caching were implemented.

The deployment took 1-2 hours, with minimal downtime. Users were notified in advance, ensuring a smooth transition to the new system.

5.4.2 Processes /Policies/Personnel

The successful implementation and management of the system relied on clear processes, robust policies, and a dedicated team. The deployment began with planning and setup, where IT staff configured the server environment and installed the application. This was followed by comprehensive testing conducted by developers to ensure the system’s stability and functionality before launch. Training sessions were provided to users, including academic advisors and administrators, to ensure they could effectively navigate and utilize the system. Policies were established to safeguard data security and privacy, including role-based access control to restrict access based on user roles. Regular system monitoring and maintenance were implemented to keep the system up-to-date, reliable, and responsive to user needs, ensuring its continued success in supporting academic management.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Introduction

This chapter presents the results of the development, testing, and evaluation of the "Student Management System," which aimed to streamline academic management and improve the efficiency of tracking grades, attendance, and supplementary materials. The chapter assesses the effectiveness of the system in meeting its objectives through alpha, beta, and user acceptance testing. The goal was to determine whether the system successfully achieved its key objectives of providing real-time grade tracking, efficient attendance management, and seamless user accessibility across different user roles.

6.2 Summary of Findings

The Student Management System successfully met its core objectives, integrating various technologies and providing a user-friendly interface. Testing validated both the functionality and performance of the system's hardware and software components.

Software Integration

- The System Database, powered by MYSQL, accurately stored and retrieved data.
- QR code generation for attendance was reliable, allowing for seamless tracking.
- User authentication was secure and efficient handled through hashed password storage.

Software Performance

- PHP and MySQL were integrated effectively, ensuring smooth handling of backend operations.
- The user interface displayed grades and attendance data with minimal delay, ensuring up-to-date information was available to students, teachers, and administrators.
- Notifications via email were timely, ensuring users were promptly informed of updates such as grade changes or attendance issues.

6.3 Testing Result:

In terms of testing results, there are three major stages which are: alpha testing, beta testing and user acceptance result.

6.3.1 Alpha Testing Result:

1. **Create Account Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-STU-01	Student registers with valid email, password, and name.	Account is created with a confirmation message.	Account is created with a confirmation message.	Pass	Validate uniqueness of email address.
UA-STU-02	Student logs in using valid credentials.	Student is redirected to their dashboard.	Student is redirected to their dashboard.	Pass	Ensure student-specific dashboard loads.
UA-STU-03	Student updates their profile picture.	Profile picture is uploaded and displayed correctly.	Profile picture is uploaded and displayed correctly.	Pass	Validate image file type and size.

2. **Login Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-LOGIN-01	Test login with valid credentials (email and password)	User should be successfully logged in and redirected to the dashboard.	User logged in successfully and redirected to the dashboard.	Pass	Ensure correct authentication flow.
UA-LOGIN-02	Test login with invalid email format	Error message: "Invalid email format."	Error message displayed: "Invalid email format."	Pass	Validate email format input.
UA-LOGIN-03	Test login with incorrect password	Error message: "Incorrect password."	Error message displayed: "Incorrect password."	Pass	Ensure correct password validation.
UA-LOGIN-04	Test login with unregistered email	Error message: "Email not found. Please register."	Error message displayed: "Email not found."	Pass	Ensure correct user existence check.
UA-LOGIN-05	Test login with empty email and password fields	Error message: "Please fill in both fields."	Error message displayed: "Please fill in both fields."	Pass	Ensure validation for empty fields.
UA-LOGIN-06	Test login with expired session (user tries to log in after session expiry)	Error message: "Session expired. Please log in again."	Error message displayed: "Session expired."	Pass	Ensure session expiration handling.

3. **Student Account Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-STU-01	Test student account creation with valid details (email, password, etc.)	Student account should be created successfully, and confirmation email should be sent.	Student account created successfully, confirmation email sent.	Pass	Ensure proper handling of student data and email notification
UA-STU-02	Test student account creation with invalid email format	Error message: "Invalid email format."	Error message displayed: "Invalid email format."	Pass	Validate email format input.
UA-STU-03	Test student account creation with missing required fields	Error message: "Please fill out all required fields."	Error message displayed: "Please fill out all required fields."	Pass	Ensure validation for mandatory fields.
UA-STU-04	Test student account creation with existing email	Error message: "Email already in use."	Error message displayed: "Email already in use."	Pass	Ensure email uniqueness validation.
UA-STU-05	Test student account login with valid credentials (email and password)	Student should be successfully logged in and redirected to their dashboard.	Student logged in successfully and redirected to dashboard.	Pass	Ensure proper authenticati on flow.
UA-STU-06	Test student account login with incorrect password	Error message: "Incorrect password."	Error message displayed: "Incorrect password."	Pass	Validate password check for login.
UA-STU-07	Test student account login with unregistered email	Error message: "Email not found. Please register."	Error message displayed: "Email not found."	Pass	Ensure correct user existence check.
UA-STU-08	Test student profile update with valid information (name, course, etc.)	Profile updates should be saved and displayed correctly.	Profile updated successfully and displayed.	Pass	Ensure profile update functionalit y.

4. **Student Dashboard Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-DASH-01	Test if student dashboard displays correct total number of Subject enrolled, Pending Activities, Missed Activities.	Dashboard should show the correct total numbers for each category.	Dashboard correctly displays the total number of students, teachers, classes, and subjects.	Pass	Ensure data consistency across the dashboard.
UA-DASH-02	Test student dashboard quick access to subject details.	Clicking on subject should open the detailed class information.	Clicking on a subject redirects to class information correctly.	Pass	Verify correct redirection to subject details.
UA-DASH-03	Test if student can view their activities on the dashboard.	Activities should be listed with status (pending/completed) and due dates.	Activities are displayed with status and due dates.	Pass	Ensure proper status and deadline display.
UA-DASH-04	Test if student dashboard displays accurate attendance record.	Dashboard should show a summary of attendance for the current semester.	Attendance is displayed correctly on the dashboard.	Pass	Validate accuracy of attendance records.
UA-DASH-05	Test if student can access profile from the dashboard.	Clicking on the profile should open the student's profile page.	Profile is accessible via the dashboard with no errors.	Pass	Ensure smooth profile navigation.

5. **Grade Tracking Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-GRADE-01	Test grade retrieval for a student to ensure correct display of grades	Grades for the student’s activities, quizzes, and exams should be displayed.	Grades displayed correctly, showing individual and overall performance.	Pass	Verify accurate grade retrieval.
UA-GRADE-02	Test grade entry by the teacher for an assignment	Grades should be entered successfully and saved in the system.	Grades entered successfully for the assignment, visible on the dashboard.	Pass	Ensure correct grade input functionality.
UA-GRADE-03	Test grade updates for a student after teacher modification	Updated grades should reflect changes immediately in the student’s profile.	Grade update reflected correctly in the student’s profile.	Pass	Validate real-time grade updates.

UA-GRADE-04	Test viewing of grades by students for the current semester	The system should display grades for each subject and activity.	Grades for each subject and activity displayed correctly.	Pass	Ensure student access to grades.
UA-GRADE-05	Test grade breakdown display for each assignment, quiz, and exam	Breakdown of grades should show individual scores and total percentage.	Grade breakdown displayed correctly, including individual scores and totals.	Pass	Verify accurate grade breakdown

6. **Attendance Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-ATT-01	Test student scanning the teacher's QR code for attendance in a class	Student's attendance should be recorded automatically upon scanning the correct QR code for the class.	Student's attendance marked as present after scanning the correct teacher's QR code for the class.	Pass	Ensure correct QR code scanning per class.
UA-ATT-02	Test student scans QR code for the class and then tries to scan it again	Student should not be marked multiple times for the same class.	Attendance is recorded once per class, even if the QR code is scanned again.	Pass	Prevent multiple attendance markings.
UA-ATT-03	Test student scanning the QR code after the class has ended	Error message: "Class attendance period has ended."	Error message displayed: "Class attendance period has ended."	Pass	Ensure time-based validation for attendance.

7. **Messaging Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-MSG-01	Test sending a message from a teacher to a student	The message should be sent successfully, and the student should receive it in their inbox.	Message sent successfully, student received the message in inbox.	Pass	Ensure proper message sending and receiving.
UA-MSG-02	Test sending a message with an empty content field	Error message: "Message content cannot be empty."	Error message displayed: "Message content cannot be empty."	Pass	Ensure content validation for messages.
UA-MSG-03	Test sending a message to multiple students simultaneously	The message should be sent successfully to all recipients.	Message sent successfully to all selected students.	Pass	Verify multiple recipient handling.

8. **Notification Module**

Test Case ID	Test Case Scenario / Condition	Expected Result	Actual Result	Status	Remarks
UA-NOT-01	Test sending a notification to a student about upcoming class activity	The notification should be sent successfully and displayed on the student’s dashboard.	Notification sent successfully and displayed on the student’s dashboard.	Pass	Ensure notifications are delivered properly.
UA-NOT-02	Test notification for a missed class/attendance	Error message: "Missed attendance notification sent."	Notification about missed attendance sent and displayed.	Pass	Verify missed attendance notifications.
UA-NOT-03	Test if student receives notification when grades are posted	The student should receive a notification when their grades are posted.	Student receives notification when grades are posted successfully.	Pass	Ensure grade posting notifications work.

6.3.2 Integration System Result

Beta Testing Result:

In terms of beta testing, the team will roll out the testing with actual admin, teachers and students from College of Computing Studies from Western Mindanao State University that will occur within this week.

User Acceptance Result:

Below is the user acceptance result which ranges from the testing occurred at alpha and beta testing – since alpha testing has had only occurred, it will only be the testing that will play a primary role in the evaluation result for the meantime.

Evaluation Result:

In terms of evaluation result, the development team has had alpha testing occur in order to evaluate not only the overall functionalities and features of the Student Management System but also the overall design of the web application.

Graphical User Interface (GUI) and User Experience (UX)

The Graphical User Interface (GUI) and User Experience (UX) are essential parts of the system's architecture, ensuring easy navigation and efficient functionality for all user roles (admin, instructor, and student). The following are significant descriptions of the GUI and UX design principles used:

Admin Side

An admin side contains a dashboard that is the central hub displaying key statistics, such as the number of students, teachers, and pending tasks, through visually appealing widgets and charts. A navigation bar that includes collapsible menus for quick access to Student Management, Teacher Accounts, Class Schedules, and reports and forms and tables containing intuitive form layouts for adding or updating records and responsive tables with filtering, sorting, and pagination for viewing data.

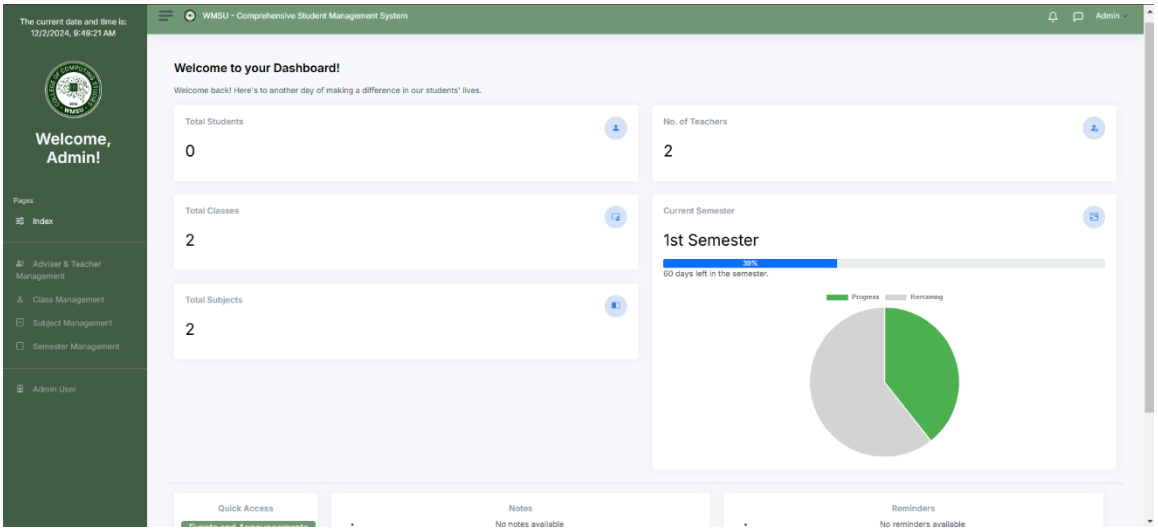


Figure 10.1

Admin Side Dashboard

Teacher Side

The Class Management Page is designed to provide teachers with an organized and efficient interface to manage their assigned classes. It allows them to view class details, mark attendance, and upload grades seamlessly. The page utilizes tabs to separate core functionalities such as attendance tracking and grading, ensuring a clear and focused workflow.

The interface includes a calendar view that displays schedules, deadlines, and upcoming school events, with color-coded markers for easy reference. Key elements of the page include a calendar widget, an attendance tracker with toggles for marking students present or absent, and a gradebook table where teachers can input and save grades directly, all designed for an intuitive user experience.

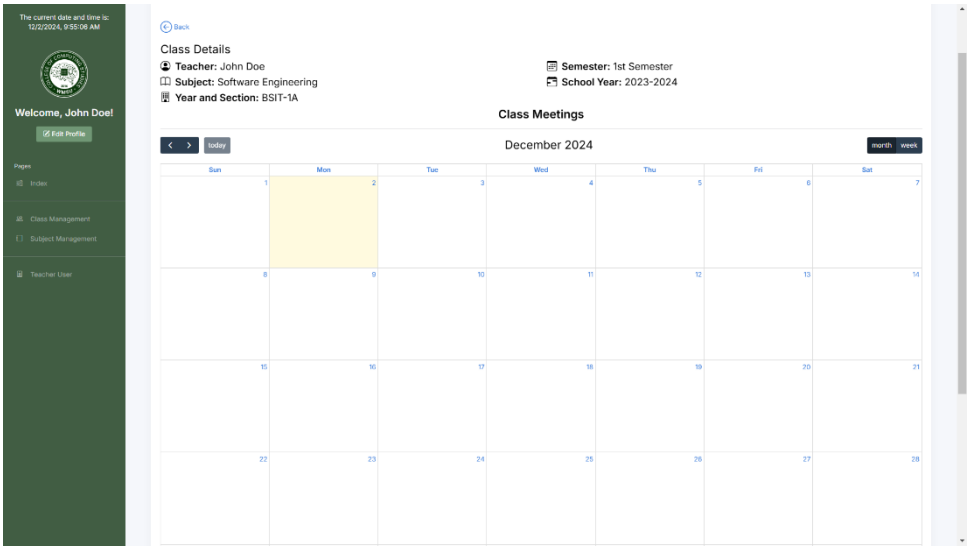


Figure 10.2

Teacher Side Calendar

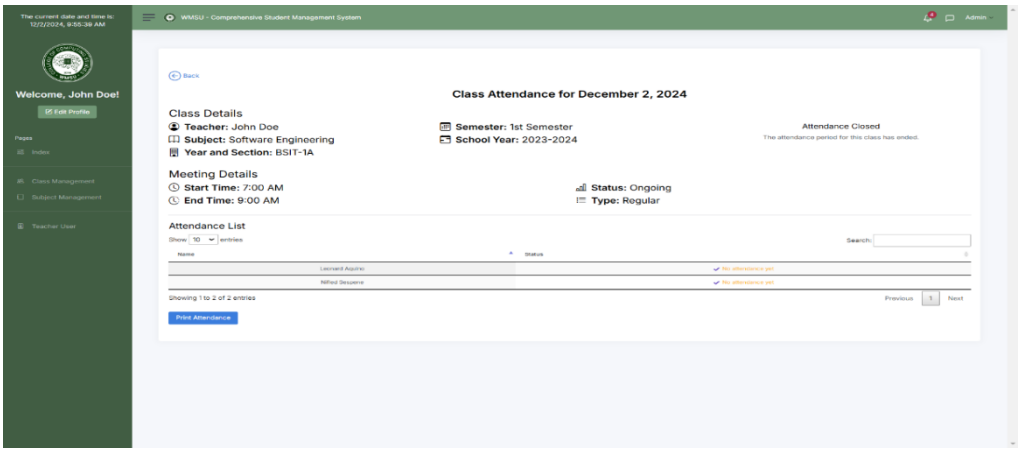
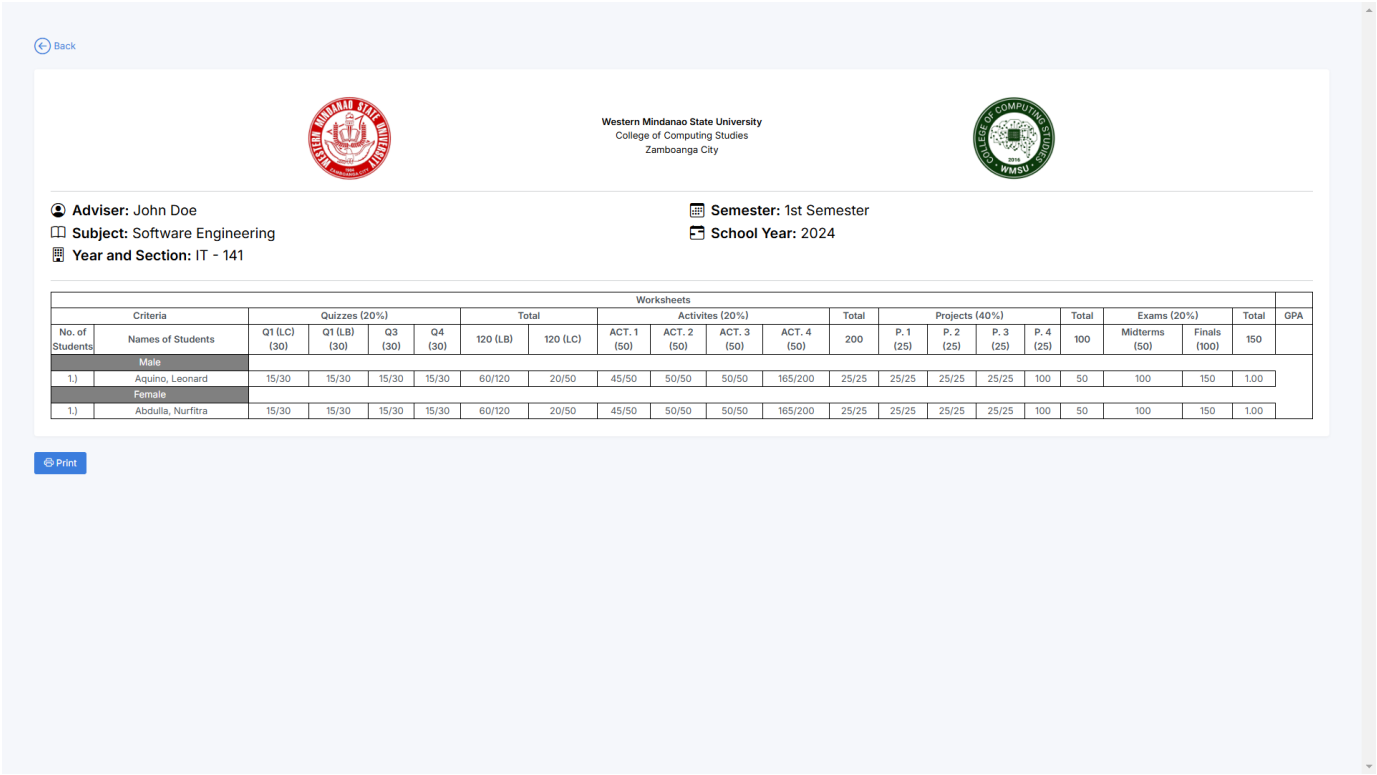


Figure 10.3

Teacher Side Attendance Tracking



Worksheets																						
Criteria		Quizzes (20%)				Total		Activites (20%)				Total	Projects (40%)				Total	Exams (20%)			Total	GPA
No. of Students	Names of Students	Q1 (LC) (30)	Q1 (LB) (30)	Q3 (30)	Q4 (30)	120 (LB)	120 (LC)	ACT. 1 (50)	ACT. 2 (50)	ACT. 3 (50)	ACT. 4 (50)	200	P. 1 (25)	P. 2 (25)	P. 3 (25)	P. 4 (25)	100	Midterms (50)	Finals (100)	150	
Male																					
1.)	Aquino, Leonard	15/30	15/30	15/30	15/30	60/120	20/50	45/50	50/50	50/50	165/200	25/25	25/25	25/25	25/25	100	50	100	150	1.00	
Female																					
1.)	Abdulla, Nurfitra	15/30	15/30	15/30	15/30	60/120	20/50	45/50	50/50	50/50	165/200	25/25	25/25	25/25	25/25	100	50	100	150	1.00	

Figure 10.4

Teacher Side Grade Tabular Form

Student Side

The Student Side of the system provides an intuitive interface designed to meet students' academic needs. The Profile Page presents a clean layout showcasing essential details such as the student's ID, course, year level, and email address. The Subject Dashboard offers a comprehensive

view of enrolled subjects, featuring subject cards with the subject name, instructor, and quick links to assignments, grades, and announcements. Notifications for new announcements or assignments ensure students stay updated and engaged with their academic responsibilities.

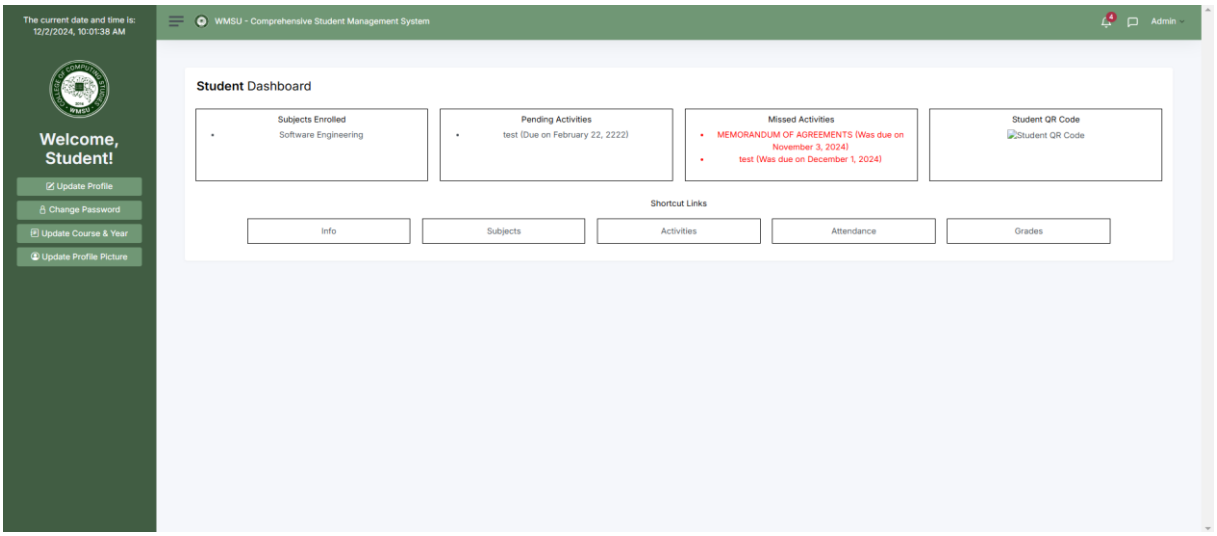


Figure 10.5

Student Side Dashboard

System Usability

In terms of the system usability, upon alpha testing, the system is designed with usability at its core, featuring a user-friendly interface with intuitive navigation, clear labels, and responsiveness for various devices. Accessibility considerations ensure inclusivity for users with disabilities, while performance optimizations allow for quick loading times and smooth handling of multiple users simultaneously. Robust error handling with validation messages and tooltips minimizes user errors, ensuring a seamless experience. Additionally, a feedback mechanism enables users to report issues or request assistance, fostering continuous improvement.

The development team will roll out the survey in terms of Frequency Distribution of Participants Response or FD for students and Post-Study System Usability Questionnaire or PSSUQ for teachers.

User Feedback on Each of Use

Most users found the system intuitive and user-friendly, particularly when navigating key features such as grade tracking, attendance monitoring, and accessing supplementary materials. The clear layout and responsive design were highlighted as major strengths, allowing users to accomplish tasks efficiently.

Positive Feedback: End-users, including students, teachers, and administrators, praised the platform's organized interface. Students appreciated the ability to quickly view their grades and attendance, while teachers found the QR code-based attendance feature easy to use and reliable.

Minor Suggestions: Some users suggested improvements to the class registration process, recommending a more streamlined approach for adding or joining classes, especially for first-time users unfamiliar with the system.

Error Rate and Error Handling

The QR code-based attendance system was designed to handle errors seamlessly and automatically log attendance without requiring prompts. Testing revealed a few scenarios where error handling needed refinement. For instance, when a student scanned the QR code for their class, the system correctly marked them as "Present" or "Late" based on the time of scanning. However, issues arose when another student attempted to use the same device for scanning; the system initially failed to block this action effectively. Post-testing, the system was updated to validate IP addresses to prevent multiple students from using the same device for attendance.

Positive Feedback: The system's automatic attendance marking was praised for its efficiency. Students appreciated the absence of prompts, as their attendance status was updated seamlessly. Teachers also noted that the system correctly logged attendance without manual intervention.

Defects Identified: During initial testing, the system allowed multiple scans from the same device in specific scenarios, resulting in duplicate entries. This was resolved by ensuring that the IP address of the scanning device was registered uniquely for each student. Additionally, any attempt to re-scan using the same device now triggers an automatic rejection with no manual prompt required.

Core Features

The system's core features, including grade tracking, QR code-based attendance, and supplementary material uploads, were thoroughly tested and validated by users. All essential workflows functioned seamlessly, enabling teachers, students, and administrators to carry out their tasks efficiently. The attendance tracking system using QR codes proved to be reliable, and students found it easy to access grades and class materials.

Positive Feedback: Teachers appreciated the straightforward process for generating unique QR codes for each class session, while students valued the quick access to their grades and attendance records. The ability to upload and access supplementary materials was also highly praised for enhancing the learning experience.

Suggestions for Improvement: Some users recommended improving the filtering and sorting options for grades and attendance records, allowing for quicker retrieval of specific information. Additionally,

a request was made to add a notification feature for upcoming deadlines or missed attendance to keep users better informed.

Performance and Reliability

The system's performance was stable and reliable during typical operations, such as scanning QR codes, accessing grades, and viewing attendance records. It efficiently handled everyday workloads, ensuring smooth interactions for both students and teachers. However, some issues were noted under high-load scenarios, such as simultaneous scanning by multiple students or accessing large amounts of data.

Positive Feedback: Users reported that the system was responsive and performed well during regular use. Attendance marking, grade tracking, and supplementary material access were quick and seamless.

Defects Identified: Performance issues were observed during scenarios involving simultaneous QR code scanning from multiple devices in a single session. Additionally, delays were noted when generating detailed reports for larger class sections. These issues have been flagged for further optimization to enhance the system's performance under heavy loads.

User Satisfactions and Functionalities

Users expressed high satisfaction with the system's functionalities, particularly its ease of use and effectiveness in managing grades, attendance, and class materials. The core features met user expectations, enabling efficient workflows for students and teachers alike. Minor enhancements were suggested to improve navigation and usability.

Positive Feedback: Students appreciated the simplicity of viewing grades and tracking attendance, while teachers found the QR code attendance system and grade input features highly convenient. Administrators praised the user management and class organization tools.

Suggestions for Improvement: Users recommended adding automated notifications for attendance and grade updates, as well as providing detailed performance analytics for administrators to monitor class performance trends more effectively.

System Functionalities

In terms of system functionalities, upon alpha testing, the system supports comprehensive functionalities tailored to different user roles with permission-based access. It ensures secure data management for storing and updating information, alongside built-in communication tools for seamless interaction. Teachers can manage classes and schedules, track attendance, and upload grades, while students access detailed profiles, assignment tracking, and real-time notifications for

updates and deadlines. The system also provides reporting capabilities, generating attendance summaries, grade analyses, and performance metrics, and integrates smoothly with external platforms like learning management systems and cloud storage solutions.

The development team will roll out the survey in terms of Frequency Distribution of Participants Response or FD for students and Post-Study System Usability Questionnaire or PSSUQ for teachers.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

The proposed Student Management System (SMS) was developed with the aim of streamlining the management of academic processes and fostering student success. The general objectives focused on creating a comprehensive platform that simplifies grade computation and storage, improves access to academic resources, and enhances student accountability through attendance tracking. Through its intuitive and user-friendly interface, the system consolidates key academic data—grades, attendance, and supplementary materials—into a single platform, facilitating efficient management for both students and administrators.

By developing features such as automated grade tracking, digital attendance management, and a centralized repository for supplementary materials, the SMS addresses the challenges students face in staying organized and accessing essential learning resources. These objectives aim to not only reduce administrative burdens but also empower students with the tools they need to maintain an overview of their academic progress, allowing them to focus more on learning and personal development.

In conclusion, the system achieves its goal of promoting a more efficient and organized academic environment, significantly enhancing the learning experience for students while easing the administrative load for educators. The integration of these features into a seamless platform will support student success, providing them with the necessary resources for academic achievement and growth.

7.2 Recommendations

While the system was tested primarily in a Local Area Network (LAN) setup for development and alpha testing, its planned deployment on a dedicated domain will significantly enhance its accessibility and scalability. Transitioning to a domain-hosted infrastructure will allow users, including administrators, teachers, and students, to interact with the system remotely, bypassing the geographical limitations of LAN-based access. This deployment will enable greater flexibility for users and support schools or institutions with multiple branches or remote learning needs.

To ensure a seamless transition to a domain-based environment, it is recommended to implement robust server-side security protocols, including SSL certificates, firewalls, and regular vulnerability assessments. This is critical to protect sensitive user information and prevent unauthorized access in

a web-hosted setting. Furthermore, hosting the system on a domain allows for the integration of advanced features such as real-time notifications, data synchronization, and remote backup services.

This would enhance the system's reliability and provide failover mechanisms to mitigate potential downtime or data loss. It is also advisable to optimize the system for different bandwidth conditions and test its performance under various loads to ensure consistent functionality for all users. Providing mobile-friendly access and leveraging cloud-based hosting solutions could further increase usability and availability, catering to a diverse range of devices and user scenarios. Ultimately, the transition to a domain-hosted platform should be accompanied by comprehensive user training and support resources, ensuring that the system's functionality and potential are fully understood and utilized by all stakeholders.

REFERENCES

SOURCES OF LITERATURE

Chung, C. H., Pasquini, L. A., & Koh, C. E. (2013). Web-based learning management system considerations for higher education. Learning and Performance Quarterly, 1(4), 24-37. https://d1wqtxts1xzle7.cloudfront.net/35605084/Web-based_Learning_Management_System_Considerations_for_Higher_Education-libre.pdf?1416219344=&response-contentdisposition=inline%3B+filename%3DWeb_based_Learning_Management_System_Con.pdf&Expires=1733110836&Signature=G0y0JDGOMAfYQGdCMdMw4zAdqIKaMjFIJGo8d0D0jhBMzG8ggvvCmmH1haizC2vXvqKfx1LKzvCm9E-a0IL06bEhg9faBf2s0J-TX1H3M04PoqHJvnYDDKvVmZyn8cpYG~eygg4a3YcmChAx7g78itcXmlf06VPbI12HHKMzNDV2L3zlJuUvcdtu1BqfoysXpqyfCBGvrBG3DdSHIs7UtO8TC2M5EsYMCXlQ8cpjvVXcDgmf1r30tC6kQ46lfdClhc2USf3OjH8MDdPMl07ZKUnoQd25Lv7N4IXlZ2NspCpU4~P~efTfxh8WnB79f5tid4VSX4UopcrTFeXi0P~DSw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Unal, Z., & Unal, A. (2011). Evaluating and comparing the usability of web-based course management systems. Journal of Information Technology Education: Research, 10(1), 19-38. https://www.learntechlib.org/p/111510/

Adewale, O. S., Nıgerıa, A., Ibam, E. O., & Alese, B. K. (2012). A web-based virtual classroom system model. Turkish Online Journal of Distance Education, 13(1), 211-223. https://dergipark.org.tr/en/pub/tojde/issue/16899/176132

Chang, C. K. (2001). Refining collaborative learning strategies for reducing the technical requirements of web-based classroom management. Innovations in education and teaching international, 38(2), 133-143. https://www.tandfonline.com/doi/abs/10.1080/13558000010030185

Moura, J. G., Brandão, L. O., & Brandão, A. A. (2007, October). A web-based learning management system with automatic assessment resources. In 2007 37th Annual Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports (pp. F2D-1). IEEE. https://ieeexplore.ieee.org/abstract/document/4418100/

Singh, M., Khan, M. A., Singh, V., Patil, A., & Wadar, S. (2015, February). Attendance management system. In 2015 2nd International Conference on Electronics and Communication Systems (ICECS) (pp. 418-422). IEEE. https://ieeexplore.ieee.org/abstract/document/7124938/

Arif, Z. H., Ali, N. S., Zakaria, N. A., & Al-Mhiqani, M. N. (2018). Attendance management system for educational sector: critical review. International Journal of Computer Science and Mobile Computing, 7(8), 60-66. https://d1wqtxts1xzle7.cloudfront.net/92689101/V7I8201814-libre.pdf?1666174258=&response-content-

disposition=inline%3B+filename%3DAttendance_Management_System_for_Educati.pdf&Expires=1733111153&Signature=dxdtkanKrFNkNfeeWlnhn1SgC9txuEIAq57zuQ59LBVZLc1LJN2tP4mLYyi0vJTJayrZazCot93a9mo4CBymcoRW03H2KkeUJkxspFjqOtq8dQFw~Y3bLTqx6EFN-HDSqT3rpN1aLAPWT5uZAQx1oHVN6~fJFxZYwbyzmRFELV0E~sJ4ApRQw~zYSGnHjsM~Pod8cT54QjxvSkrLokhYQ7-YIiGG8IADEG~YcSaWy7TJXH341hVdzHBq41j185KsghwnCC58p5pCMNxYnUnKwBNx-ZUohHCypUL6GobQ2jiHWHkbRuFQgbFuX5gW6e7Ut5EIsNQoOgBNnP3MyHgI~A__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Yang, P., Sun, G., He, J., Zhou, P., & Liu, J. (2017). A student information management system based on fingerprint identification and data security transmission. Journal of Electrical and Computer Engineering, 2017(1), 9598581. https://onlinelibrary.wiley.com/doi/full/10.1155/2017/9598581

Alwi, N. H. M., & Fan, I. S. (2010). E-learning and information security management. International Journal of Digital Society (IJDS), 1(2), 148-156. https://infonomics-society.org/wp-content/uploads/ijds/published-papers/volume-1-2010/E-Learning-and-Information-Security-Management.pdf

APPENDIX A

APPENDIX A: DATA GATHERING

Method Use to Collect Feedback

To ensure the system's functionalities met the needs of its users, a combination of structured and informal methods was utilized to gather actionable feedback during testing. This allowed developers to refine the system based on real-world usage scenarios.

1. **Bug Reports**

Testers were encouraged to identify and report technical issues such as system errors, usability problems, or crashes. A streamlined process for bug submission was implemented, enabling users to:

- Describe issues in detail, including steps to reproduce the problem.
- Attach relevant screenshots or logs to provide additional context.
- Track the resolution status of reported bugs through a dedicated system interface.

2. **Surveys**

Structured surveys were distributed to beta testers, focusing on system usability, functionality, and performance. These surveys included:

- Likert-scale questions to gather quantitative insights.
- Open-ended questions to collect qualitative feedback for deeper analysis.

3. **User Interview**

Follow-up interviews were conducted with a selected group of testers to:

- Explore specific usability issues and areas for improvement.
- Understand user preferences and expectations.
- Gather suggestions for enhancing features such as attendance tracking, grade management, and supplementary materials.

APPENDIX B

APPENDIX B: SOURCE CODE

Notifications

```
<?php
session_start();
require ‘../../server/conn.php’;
if ($_SERVER[‘REQUEST_METHOD’] == ‘POST’ && isset($_POST[‘id’])) {
    $id = intval($_POST[‘id’]);
    // Update the notification status in the database
    $stmt = $pdo->prepare(“UPDATE admin_notifications SET status = ‘read’ WHERE id = ?”);
    $stmt->execute([$id]);
    // Redirect or return a success message
    header(‘Location: /path/to/notifications’);
}
?>
```

Source Code for Reading Notifications Individually

```
<?php
require '../../server/conn.php'; // Adjust the path based on your directory structure

// Assuming you have a `status` column in your `admin_notifications` table
try {
    $stmt = $pdo->prepare("UPDATE admin_notifications SET status = 'read' WHERE status =
'unread'");
    $stmt->execute();
    echo json_encode(['success' => true]);
} catch (PDOException $e) {
    echo json_encode(['success' => false, 'message' => $e->getMessage()]);
}
?>
```

Source Code for Reading Notifications as One

```
<?php
// delete.php
session_start();
require '../../server/conn.php'; // Adjust the path based on your directory structure
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['id'])) {
    $id = intval($_POST['id']);
    // Delete the notification from the database
    $stmt = $pdo->prepare("DELETE FROM admin_notifications WHERE id = ?");
    $stmt->execute([$id]);
    // Redirect or return a success message
    header('Location: /path/to/notifications'); // Change to your redirect path
}
?>
```

Source Code for Deleting Notifications One-by-One

```
<?php
require '../../server/conn.php'; // Adjust the path based on your directory structure
try {
    $stmt = $pdo->prepare("DELETE FROM admin_notifications");
    $stmt->execute();

    echo json_encode(['success' => true]);
} catch (PDOException $e) {
    echo json_encode(['success' => false, 'message' => $e->getMessage()]);
}
?>
```

Source Code for Deleting Notifications as One

Chat

```
script>
        const userId = <?php echo isset($_SESSION['user_id']) ? $_SESSION['user_id'] :
'null'; ?>;
        const userType = '<?php echo isset($_SESSION['user_type']) ?
$_SESSION['user_type'] : ''; ?>';        console.log('userId:', userId);
        console.log('userType:', userType);
        if (userId === null || userType === '') {
            console.error('User ID or User Type is not properly set.');
        } else {
            console.log('User is logged in:', userId, userType);
        }
    </script>
    <script>
        document.getElementById('messagesModal').addEventListener('shown.bs.modal', () =>
{
            fetchRecentConversations(userId, userType);
        });
        async function searchUsersAcrossApp(query, userType = 'all') {
            try {
                const response = await
fetch(`messaging.php?action=search_users&query=${encodeURIComponent(query)}&user_type=${u
serType}`);
                const data = await response.json();
                if (data.success) {
                    return data.data; // Return the list of users
                } else {
                    console.error('Search Error:', data.message); // Log the error m
                    return [];
                }
            } catch (error) {
                console.error('Request failed', error);
                return [];
```

```
            }
        }
        document.getElementById('userSearch').addEventListener('input', async function ()
{
            const query = this.value.trim();
            const recentConversations = document.getElementById('recentConversations');
            const searchResults = document.getElementById('searchResults');
            const resultsContainer = searchResults.querySelector('.list-group');
            if (query.length > 0) {
                recentConversations.style.display = 'none';
                searchResults.style.display = 'block';
                resultsContainer.innerHTML = '<p class="text-muted text-
center">Searching...</p>';
                const results = await searchUsersAcrossApp(query);
                resultsContainer.innerHTML = results.length
                    ? results.map(user => `
                    <a href="#" class="list-group-item list-group-item-action"
onclick="openConversation(${user.id}, '${user.type}', '${user.name}')">
                        <span>${user.name}</span>
                    </a>
                `).join('')
                    : '<p class="text-muted text-center">No users found.</p>';
            } else {
                recentConversations.style.display = 'block';
                searchResults.style.display = 'none';
            }
        });
        async function fetchRecentConversations(userId, userType) {
    try {
        console.log('Fetching recent conversations...');
        const response = await
fetch(`messaging.php?action=get_recent_conversations&user_id=${userId}&user_type=${userTy
pe}`);
        const data = await response.json();
        console.log('Recent Conversations:', data);
        if (data.success) {
            const recentConvoList = document.getElementById('recentConvoList');
            recentConvoList.innerHTML = ''; // Ensure the list is cleared before
            if (data.data.length > 0) {
                const seenUsers = new Set(); // Used to prevent multiple conversation
                data.data.forEach(convo => {
    const userPair = [Math.min(convo.sender_id, convo.receiver_id),
Math.max(convo.sender_id, convo.receiver_id)].join('_');
    if (!seenUsers.has(userPair)) {
        seenUsers.add(userPair);
        const convoItem = document.createElement('a');
        convoItem.classList.add('list-group-item', 'list-group-item-action');
        convoItem.href = '#';
        convoItem.onclick = (e) => {
            e.preventDefault();
```

```
            openConversation(
                (convo.sender_id === userId) ? convo.receiver_id : convo.sender_id,
                (convo.sender_id === userId) ? convo.receiver_type : convo.sender_type,
                (convo.sender_id === userId) ? convo.receiver_name : convo.sender_name
            );
        };
        convoItem.innerHTML = `
            <strong>${convo.conversation_name}</strong><br />
            ${convo.message}<br />
            <small class="text-muted"> - ${new
Date(convo.timestamp).toLocaleString()}</small>
        `;
        recentConvoList.appendChild(convoItem);
    } else {
        console.log(`Duplicate conversation skipped: ${userPair}`);
    }
});
            } else {
                recentConvoList.innerHTML = '<p>No recent conversations.</p>';
            }
        }
    } catch (error) {
        console.error('Error:', error);
        alert('Error loading recent conversations. Try again.');
    }
}
async function openConversation(receiverId, receiverType, userName) {
    const currentUserId = userId;
    const currentUserType = userType;

    const modalTitle = document.getElementById('modalTitle');
    modalTitle.textContent = `Conversation with ${userName}`;
    document.getElementById('searchSection').style.display = 'none';
    document.getElementById('conversationSection').style.display = 'block';
    document.getElementById('backButton').style.display = 'inline-block';
    const conversationContent = document.getElementById('conversationContent');
    conversationContent.dataset.userId = receiverId;
    conversationContent.dataset.userType = receiverType;

    if (!conversationContent.innerHTML || conversationContent.dataset.userId !==
receiverId) {
        conversationContent.innerHTML = '<p class="text-muted text-
center">Loading...</p>';
    }    let messages = await fetchConversation(receiverId, receiverType, currentUserId,
currentUserType);
    updateConversation(messages, currentUserId, currentUserType, receiverId,
receiverType);
    const pollingInterval = setInterval(async () => {
        const newMessages = await fetchConversation(receiverId, receiverType,
currentUserId, currentUserType);
```

```
        if (newMessages.length > messages.length) {
            messages = newMessages;
            updateConversation(messages, currentUserId, currentUserType, receiverId,
receiverType);
        }
    }, 3000);
    document.getElementById('messagesModal').addEventListener('hidden.bs.modal', () => {
        clearInterval(pollingInterval);
    });
    function updateConversation(messages, currentUserId, currentUserType, receiverId,
receiverType) {
        conversationContent.innerHTML = '';
        if (!messages || messages.length === 0) {
            conversationContent.innerHTML = '<p class="text-muted text-center">No
messages yet.</p>';
        } else {
            messages.forEach(message => {
                const isCurrentUserSender = message.sender_id === currentUserId &&
message.sender_type === currentUserType;
                const alignment = isCurrentUserSender ? 'end' : 'start';
                const bgColor = isCurrentUserSender ? 'bg-primary text-white' : 'bg-
light';
                const messageHtml = `
                    <div class="d-flex justify-content-${alignment} mb-2">
                        <div class="p-2 rounded ${bgColor}">
                            ${message.message}
                        </div>
                    </div>`;
                conversationContent.innerHTML += messageHtml;
            });
            conversationContent.scrollTop = conversationContent.scrollHeight;
        }
    }
}
async function fetchConversation(receiverId, receiverType, senderId, senderType) {

 try {

        const url =
`messaging.php?action=get_conversation&receiver_id=${receiverId}&receiver_type=${receiver
Type}&sender_id=${senderId}&sender_type=${senderType}`;
        const response = await fetch(url);
        const data = await response.json();
        console.log('Fetched Conversation:', data);
        return data.success ? data.data : [];
    } catch (error) {
        console.error('Error fetching conversation:', error);
        return [];
    }
}
```

```
        async function sendMessage() {
    const messageInput = document.getElementById('messageInput');
    const message = messageInput.value.trim();
    const conversationContent = document.getElementById('conversationContent');
    const receiverId = conversationContent.dataset.userId;
    const receiverType = conversationContent.dataset.userType;
    const receiverName =
document.getElementById('modalTitle').textContent.replace('Conversation with ',
'').trim();
    if (message && receiverId && receiverType) {
        try {
            const response = await fetch('messaging.php?action=send_message', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({
                    message,
                    receiver_id: receiverId,
                    receiver_type: receiverType,
                    receiver_name: receiverName
                })
            });
            const data = await response.json();
            if (data.success) {
                const messageHtml = `
                    <div class="d-flex justify-content-end mb-2">
                        <div class="bg-primary text-white p-2 rounded">${message}</div>
                    </div>`;
                conversationContent.innerHTML += messageHtml;
                messageInput.value = '';
                conversationContent.scrollTop = conversationContent.scrollHeight;
            } else {
                alert(data.message || 'Failed to send message. Try again.');
            }
        } catch (error) {
            console.error('Error:', error);
            alert('Error sending message. Try again.');
        }
    } else {
        alert('Message, receiver ID, and receiver type are required.');
    }
}        function goBackToMessages() {
            const modalTitle = document.getElementById('modalTitle');
            const conversationSection = document.getElementById('conversationSection');
            const searchSection = document.getElementById('searchSection');
            const backButton = document.getElementById('backButton');
            modalTitle.textContent = 'Messages';
            conversationSection.style.display = 'none';
            searchSection.style.display = 'block';
            backButton.style.display = 'none';
        }
```

```
    </script>
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);
header('Content-Type: application/json');
session_start();
require 'processes/server/conn.php'; // Include your PDO connection setup
$action = $_GET['action'] ?? null;
if ($action === 'search_users') {
    searchUsers();
} elseif ($action === 'get_conversation') {
    getConversation();
} elseif ($action === 'send_message') {
    sendMessage();
} elseif ($action === 'get_recent_conversations') {
    get_recent_conversations();
}
function searchUsers()

{
    global $pdo;
    $query = $_GET['query'] ?? '';
    $query = "%$query%";
    $results = [];
    $tables = [
        'admin' => 'CONCAT(first_name, " ", last_name) AS name, id, "admin" AS type,
first_name, last_name',  // Add first_name and last_name here for WHERE clause
        'students' => 'fullName AS name, id, "student" AS type, fullName',  // Add
fullName for WHERE clause
        'staff_accounts' => 'fullName AS name, id, "staff" AS type, fullName',  // Add
fullName for WHERE clause
    ];
    foreach ($tables as $table => $fields) {
        // Update WHERE clause to use actual columns for filtering
        if ($table == 'admin') {
            $stmt = $pdo->prepare("SELECT $fields FROM $table WHERE CONCAT(first_name, '
', last_name) LIKE :query");
        } else {
            $stmt = $pdo->prepare("SELECT $fields FROM $table WHERE fullName LIKE
:query");
        }
        $stmt->bindParam(':query', $query);
        $stmt->execute();
        $results = array_merge($results, $stmt->fetchAll(PDO::FETCH_ASSOC));
    }
    echo json_encode(['success' => true, 'data' => $results]);
    exit;
}
function getConversation()
{    if (!isset($_SESSION['user_id']) || !isset($_SESSION['user_type'])) {
```

```
        echo json_encode(['success' => false, 'message' => 'User is not logged in.']);
        exit;
    }
    global $pdo;
    $userId = (int) $_SESSION['user_id'];
    $userType = trim($_SESSION['user_type']);
    $receiverId = isset($_GET['receiver_id']) ? (int) $_GET['receiver_id'] : 0;
    $receiverType = isset($_GET['receiver_type']) ? trim($_GET['receiver_type']) : '';
    error_log("Session - user_id: $userId, user_type: $userType");
    error_log("GET - receiver_id: $receiverId, receiver_type: $receiverType");
    if (!$userId || !$userType || !$receiverId || !$receiverType) {
        error_log("Missing parameters: userId=$userId, userType=$userType,
receiverId=$receiverId, receiverType=$receiverType");
        echo json_encode(['success' => false, 'message' => 'Invalid or missing
parameters.']);
        exit;
    }
    try {
        $sql = "
            SELECT sender_id, sender_type, receiver_id, receiver_type, message, timestamp
            FROM messages
            WHERE
                (sender_id = '$userId' AND sender_type = '$userType' AND receiver_id =
'$receiverId' AND receiver_type = '$receiverType')
                OR
                (sender_id = '$receiverId' AND sender_type = '$receiverType' AND
receiver_id =  '$userId' AND receiver_type = '$userType')
            ORDER BY timestamp ASC
        ";
        $stmt = $pdo->prepare($sql);
        error_log("Executing query with user_id=$userId, user_type=$userType,
receiver_id=$receiverId, receiver_type=$receiverType");
        $stmt->execute();
        $messages = $stmt->fetchAll(PDO::FETCH_ASSOC);
        if (empty($messages)) {
            echo json_encode(['success' => true, 'message' => 'No messages found.',
'data' => []]);
        } else {
            echo json_encode(['success' => true, 'data' => $messages]);
        }
    } catch (PDOException $e) {
        error_log("Error executing query: " . $e->getMessage());
        echo json_encode(['success' => false, 'message' => 'Error fetching
messagesss.']);
    }
}
function getRecentConversations()
{
    // Check if the user is logged in
    if (!isset($_SESSION['user_id']) || !isset($_SESSION['user_type'])) {
```

```
        echo json_encode(['success' => false, 'message' => 'User is not logged in.']);
        exit;
    }
    global $pdo;
    $userId = $_SESSION['user_id'];
    $userType = $_SESSION['user_type'];
    error_log("User ID: $userId, User Type: $userType");
    if (!isset($_GET['user_id'], $_GET['user_type'])) {
        echo json_encode(['success' => false, 'message' => 'Missing required
parameters.']);
        exit;
    }
    $receiverId = $_GET['user_id'];
    $receiverType = $_GET['user_type'];
    $query = "
        SELECT sender_id, sender_type, message, timestamp
        FROM messages
        WHERE
            (
                (sender_id = :currentUserId AND sender_type = :currentUserType AND
receiver_id = :receiverId AND receiver_type = :receiverType)
                OR
                (sender_id = :receiverId AND sender_type = :receiverType AND receiver_id
= :currentUserId AND receiver_type = :currentUserType)
            )
        ORDER BY timestamp ASC
    ";

    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':currentUserId', $userId, PDO::PARAM_INT);
    $stmt->bindParam(':currentUserType', $userType, PDO::PARAM_STR);
    $stmt->bindParam(':receiverId', $receiverId, PDO::PARAM_INT);
    $stmt->bindParam(':receiverType', $receiverType, PDO::PARAM_STR);
    $stmt->execute();
    $messages = $stmt->fetchAll(PDO::FETCH_ASSOC);
    error_log("Messages fetched: " . print_r($messages, true));
    if (empty($messages)) {
        error_log('No messages found for the conversation.');
        echo json_encode(['success' => true, 'data' => []]);
        return;
    }
    echo json_encode(['success' => true, 'data' => $messages]);
}

function get_recent_conversations()
{
    if (!isset($_SESSION['user_id']) || !isset($_SESSION['user_type'])) {
        echo json_encode(['success' => false, 'message' => 'User is not logged in.']);
        exit;
    }
```

```
    global $pdo;
    try {
        $userId = $_SESSION['user_id'];
        $userType = $_SESSION['user_type'];
        $query = "
        SELECT
            sender_id,
            sender_type,
            sender_name,
            receiver_id,
            receiver_type,
            receiver_name,
            message,
            timestamp,
            CASE
                WHEN sender_id = $userId AND sender_type = '$userType' THEN receiver_name
                ELSE sender_name
            END AS conversation_name
        FROM messages
        WHERE (
                (sender_id = $userId AND sender_type = '$userType')
             OR (receiver_id = $userId AND receiver_type = '$userType')
              )
        AND NOT (sender_id = receiver_id AND sender_type = receiver_type)
        GROUP BY
            LEAST(sender_id, receiver_id), GREATEST(sender_id, receiver_id)
        ORDER BY timestamp DESC
        LIMIT 10
        ";
        $stmt = $pdo->prepare($query);
        $stmt->execute();
        $conversations = $stmt->fetchAll(PDO::FETCH_ASSOC);
        error_log("Fetched conversations: " . print_r($conversations, true));
        if ($conversations) {
            echo json_encode(['success' => true, 'data' => $conversations]);
        } else {
            echo json_encode(['success' => false, 'message' => 'No conversations
found.']);
        }
    } catch (PDOException $e) {
        error_log("Error fetching conversations: " . $e->getMessage());
        echo json_encode(['success' => false, 'message' => 'Database error: ' . $e-
>getMessage()]);
    }
}
function sendMessage()
{    if (!isset($_SESSION['user_id'], $_SESSION['user_type'], $_SESSION['name'])) {
        echo json_encode(['success' => false, 'message' => 'User is not logged in.']);
        exit;
    }
```

```
    $data = json_decode(file_get_contents('php://input'), true);
    if (!$data || !isset($data['message'], $data['receiver_id'], $data['receiver_type'],
$data['receiver_name'])) {
        echo json_encode(['success' => false, 'message' => 'Missing required fields.']);
        exit;
    }
    $message = trim($data['message']);
    $receiverId = $data['receiver_id'];
    $receiverType = $data['receiver_type'];
    $receiverName = trim($data['receiver_name']);
    $senderId = $_SESSION['user_id'];
    $senderType = $_SESSION['user_type'];
    $senderName = $_SESSION['name'];
    if (empty($message)) {
        echo json_encode(['success' => false, 'message' => 'Message cannot be empty.']);
        exit;
    }
    global $pdo;
    try {
        // Prepare the SQL statement
        $stmt = $pdo->prepare("
            INSERT INTO messages (sender_id, sender_type, sender_name, receiver_id,
receiver_type, receiver_name, message)
            VALUES (:sender_id, :sender_type, :sender_name, :receiver_id, :receiver_type,
:receiver_name, :message)
        ");
        $stmt->bindParam(':sender_id', $senderId, PDO::PARAM_INT);
        $stmt->bindParam(':sender_type', $senderType, PDO::PARAM_STR);
        $stmt->bindParam(':sender_name', $senderName, PDO::PARAM_STR);
        $stmt->bindParam(':receiver_id', $receiverId, PDO::PARAM_INT);
        $stmt->bindParam(':receiver_type', $receiverType, PDO::PARAM_STR);
        $stmt->bindParam(':receiver_name', $receiverName, PDO::PARAM_STR);
        $stmt->bindParam(':message', $message, PDO::PARAM_STR);
        if ($stmt->execute()) {
            echo json_encode(['success' => true]);
        } else {
            echo json_encode(['success' => false, 'message' => 'Failed to send
message.']);
        }
    } catch (PDOException $e) {
        error_log("Error sending message: " . $e->getMessage());
        echo json_encode(['success' => false, 'message' => 'Database error: ' . $e-
>getMessage()]);
    }
}
```

Source Code for Chatting

Grading

```php
<?php
                                function calculateAttendance($class_id, $student_id,
$pdo)
                                {
                                    try {
                                        $stmt = $pdo->prepare("
            SELECT cm.id AS meeting_id,
                   COUNT(a.id) AS total_attendance,
                   SUM(CASE WHEN a.status = 'present' THEN 1
                            WHEN a.status = 'late' THEN 0.5 ELSE 0 END) AS total_score
            FROM classes_meetings cm
            LEFT JOIN attendance a
            ON cm.id = a.meeting_id
            AND a.class_id = :class_id1
            AND a.student_id = :student_id
            WHERE cm.class_id = :class_id2
            GROUP BY cm.id
        ");
                                        $stmt->bindParam(':class_id1', $class_id,
PDO::PARAM_INT);
                                        $stmt->bindParam(':student_id', $student_id,
PDO::PARAM_INT);
                                        $stmt->bindParam(':class_id2', $class_id,
PDO::PARAM_INT);
                                        $stmt->execute();
                                        $results = $stmt->fetchAll(PDO::FETCH_ASSOC);

                                        $totalMeetings = count($results);
                                        $totalScore = 0;
                                        foreach ($results as $row) {
                                            $totalScore += $row['total_score'];
                                        }

                                        $attendancePercentage = $totalMeetings > 0 ?
($totalScore / $totalMeetings) * 100 : 0;
                                        return $attendancePercentage;
                                    } catch (PDOException $e) {
                                        echo "<p class='error'>Error: " .
htmlspecialchars($e->getMessage()) . "</p>";
                                        return 0;
                                    }
                                }
                                $class_id = isset($_GET['id']) ? (int) $_GET['id'] :
null;

                                if ($class_id) {
                                    try {
                                        $stmt = $pdo->prepare("SELECT * FROM classes
WHERE id = :class_id");
```

```
                                    $stmt->bindParam(':class_id', $class_id,
PDO::PARAM_INT);
                                    $stmt->execute();
                                    $class = $stmt->fetch();

                                    if ($class) {
                                        $subject = $class['subject'];
                                        $stmt2 = $pdo->prepare("SELECT DISTINCT type,
id FROM classes WHERE subject = :subject");
                                        $stmt2->bindParam(':subject', $subject,
PDO::PARAM_STR);
                                        $stmt2->execute();
                                        $types = $stmt2->fetchAll(PDO::FETCH_ASSOC);
                                        $hasLecture = false;
                                        $hasLaboratory = false;
                                        foreach ($types as $typeRow) {
                                            if (strcasecmp($typeRow['type'],
'lecture') == 0) {
                                                $hasLecture = true;
                                            }
                                            if (strcasecmp($typeRow['type'],
'laboratory') == 0) {
                                                $hasLaboratory = true;
                                            }
                                        }
                                        $lectureWeight = $hasLaboratory ? 60 : 60;
                                        $labWeight = $hasLaboratory ? 40 : 40;
                                        function calculateTermGrades($class_id,
$term, $lectureWeight, $labWeight, $pdo, $hasLecture, $hasLaboratory)
                                        {
                                            $stmt = $pdo->prepare("
        SELECT a.id AS activity_id, a.type, a.max_points, s.student_id, s.score
        FROM activities a
        LEFT JOIN activity_submissions s ON a.id = s.activity_id
        WHERE a.class_id = :class_id AND a.term = :term
    ");
                                            $stmt->bindParam(':class_id', $class_id,
PDO::PARAM_INT);
                                            $stmt->bindParam(':term', $term,
PDO::PARAM_STR);
                                            $stmt->execute();
                                            $activities = $stmt-
>fetchAll(PDO::FETCH_ASSOC);
                                            $grades = [];
                                            foreach ($activities as $activity) {
                                                $student_id =
$activity['student_id'];
                                                $type = $activity['type'];
                                                $score = $activity['score'] ?? 0;
```

```
                                                $max_points =
$activity['max_points'];

                                                if (!isset($grades[$student_id])) {
                                                    $grades[$student_id] = [
                                                        'lecture' => [
                                                            'exam' => [],
                                                            'quiz' => [],
                                                            'activity' => [],
                                                        ],
                                                        'laboratory' => [
                                                            'exam' => [],
                                                            'exercise' => [],
                                                            'activity' => [],
                                                        ]
                                                    ];
                                                }
                                                $percentageScore = $score /
$max_points * 100;

                                                if (strcasecmp($type, 'exam') == 0) {
                                                    $grades[$student_id]['lecture']['
exam'][] = $percentageScore;

                                                } elseif (strcasecmp($type, 'quiz')
== 0) {

                                                    $grades[$student_id]['lecture']['
quiz'][] = $percentageScore;

                                                } elseif (strcasecmp($type,
'activity') == 0) {

                                                    $grades[$student_id]['lecture']['
activity'][] = $percentageScore;

                                                }
                                            }
                                            $termGrades = [];
                                            foreach ($grades as $student_id =>
$studentScores) {

                                                $lectureExamAvg =
!empty($studentScores['lecture']['exam']) ? array_sum($studentScores['lecture']['exam'])
/ count($studentScores['lecture']['exam']) : 0;
                                                $lectureQuizAvg =
!empty($studentScores['lecture']['quiz']) ? array_sum($studentScores['lecture']['quiz'])
/ count($studentScores['lecture']['quiz']) : 0;
                                                $lectureActivityAvg =

!empty($studentScores['lecture']['activity']) ?
array_sum($studentScores['lecture']['activity']) /
count($studentScores['lecture']['activity']) : 0;

                                                $attendancePercentage =
calculateAttendance($class_id, $student_id, $pdo);
                                                $attendanceScore =
($attendancePercentage / 100) * 30
```

```
                                                                                $lect
ureGrade = ($lectureExamAvg * 0.4) + ($lectureQuizAvg * 0.3) + ($lectureActivityAvg *
0.3) + $attendanceScore;
                                                    $numericalRating =
convertToNumericalRating($lectureGrade, $student_id, $class_id, $pdo);
                                                    $termGrades[$student_id] = [
                                                        'percentage' =>
round($lectureGrade, 2),
                                                        'numerical_rating' =>
$numericalRating
                                                    ];
                                                }
                                                return $termGrades;
                                            }
                                            function
convertToNumericalRating($percentage, $student_id, $class_id, $pdo)
                                            {
                                                $stmt = $pdo->prepare("SELECT
midterm_grade, final_grade FROM student_grades WHERE student_id = :student_id AND
class_id = :class_id");
                                                $stmt->bindParam(':student_id',
$student_id, PDO::PARAM_INT);
                                                $stmt->bindParam(':class_id', $class_id,
PDO::PARAM_INT);
                                                $stmt->execute();
                                                $grades = $stmt-
>fetch(PDO::FETCH_ASSOC);
                                                if ($grades) {
                                                    if ($grades['midterm_grade'] == 'UW'
|| $grades['final_grade'] == 'UW') {
                                                        return 'UW';
                                                    } elseif ($grades['midterm_grade'] ==
'INC' || $grades['final_grade'] == 'INC') {
                                                        return 'INC';
                                                    } elseif ($grades['midterm_grade'] ==
'AW' || $grades['final_grade'] == 'AW') {
                                                        return 'AW';
                                                    }
                                                }
                                                if ($percentage >= 99)
                                                    return 1.0;
                                                if ($percentage >= 95)
                                                    return 1.25;
                                                if ($percentage >= 90)
                                                    return 1.5;
                                                if ($percentage >= 85)
                                                    return 1.75;
                                                if ($percentage >= 80)
                                                    return 2.0;
                                                if ($percentage >= 75)
```

```
                                                return 2.25;
                                            if ($percentage >= 70)
                                                return 2.5;
                                            if ($percentage >= 65)
                                                return 2.75;
                                            if ($percentage >= 60)
                                                return 3.0;
                                            return 5.0;
                                        }
                                        $midtermGrades =
calculateTermGrades($class_id, 'midterm', $lectureWeight, $labWeight, $pdo, $hasLecture,
$hasLaboratory);
                                        $finalGrades = calculateTermGrades($class_id,
'final', $lectureWeight, $labWeight, $pdo, $hasLecture, $hasLaboratory);
                                        $overallGrades = [];
                                        foreach ($midtermGrades as $student_id =>
$midtermGrade) {
                                            $finalGrade = $finalGrades[$student_id]
?? 0;

                                            $overallPercentage =
round(($midtermGrade['percentage'] * 0.4) + ($finalGrade['percentage'] * 0.6), 2);
                                            $overallRating =
convertToNumericalRating($overallPercentage, $student_id, $class_id, $pdo);

                                            $overallGrades[$student_id] = [
                                                'percentage' => $overallPercentage,
                                                'numerical_rating' => $overallRating
                                            ];
                                        }
                                    } else {
                                        echo "<p class='error'>Class not found.</p>";
                                    }
                                } catch (PDOException $e) {
                                    echo "<p class='error'>Error: " .
htmlspecialchars($e->getMessage()) . "</p>";
                                }
                            } else {
                                echo "<p class='error'>No class selected.</p>";
                            }
                            ?>
```

Source Code for Grading

Attendance

```
  <?php
    exit();
}
$studentId = $_SESSION['student_id'];
$classId = $_GET['class_id'] ?? null;
```

```php
$date = $_GET['date'] ?? null;
$startTime = $_GET['start_time'] ?? null;
$endTime = $_GET['end_time'] ?? null;
$meetingId = $_GET['meetingId'] ?? null;
if ($classId && $date && $startTime && $endTime) {
    markAttendance($studentId, $classId, $date, $startTime, $endTime, $meetingId); //
} else {
    $_SESSION['status_message'] = "Invalid attendance data.";
    header('Location: ../students/attendance_status.php');
    exit();
}

function markAttendance($studentId, $classId, $date, $startTime, $endTime, $meetingId) {
    global $pdo; // Ensure you use your PDO connection
    $currentTime = date('h:i A');
    $currentTimestamp = strtotime($currentTime);
    $startTimeStamp = strtotime($date . ' ' . $startTime);
    $endTimeStamp = strtotime($date . ' ' . $endTime);

    $stmt = $pdo->prepare("SELECT status FROM attendance WHERE student_id = :student_id
AND class_id = :class_id AND date = :date AND meeting_id = :meeting_id");
    $stmt->execute([
        ':student_id' => $studentId,
        ':class_id' => $classId,
        ':date' => $date,
        ':meeting_id' => $meetingId,
    ]);

    if ($stmt->rowCount() > 0) {
        $existingRecord = $stmt->fetch(PDO::FETCH_ASSOC);
        $_SESSION['status_message'] = "Attendance already recorded as: " .
htmlspecialchars($existingRecord['status']);
        header('Location: ../students/attendance_status.php');
        exit();
    }
    $status = 'absent';
    if ($currentTimestamp >= $startTimeStamp && $currentTimestamp <= ($startTimeStamp +
(5 * 60))) {
        $status = 'present'; // Present if within 5 minutes after start time
    } elseif ($currentTimestamp > ($startTimeStamp + (5 * 60)) && $currentTimestamp <=
$endTimeStamp) {
        $status = 'late';
    } elseif ($currentTimestamp > $endTimeStamp) {
        $status = 'absent';
    }
    $stmt = $pdo->prepare("INSERT INTO attendance (student_id, meeting_id, class_id,
date, status) VALUES (:student_id, :meeting_id, :class_id, :date, :status)");
    $stmt->execute([
        ':student_id' => $studentId,
        ':meeting_id' => $meetingId,
```

```
        ':class_id' => $classId,
        ':date' => $date,
        ':status' => $status,
    ]);
    $_SESSION['status_message'] = "Attendance marked successfully as: " .
htmlspecialchars($status);
    header('Location: ../students/attendance_status.php');
    exit();
}
?>
```

Source Code for Attendance

\

APPENDIX C

APPENDIX C: TESTING

Unit Testing	
Project Number:	1.0
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Type:	Unit Testing
Test Description:	Unit testing is a method of testing individual units or components of a software application.
Date:	From December 02, 2024 to December 08, 2024
App Version No:	N/A
Total Duration:	7 days
Total no. of passed test cases:	**6**
Total no. of failed test cases:	**0**
Total no. of tested test cases:	**0**
Total no. of test cases:	**6**

STATUS	
% Passed Test Cases	100%
% Failed Test Cases	0%
% Tested Test Cases	100%

Integration Testing	
Project Number:	1.0
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Type:	Integration Testing
Test Description:	Integration testing is a method of testing how different units or components of a software application interact with each other.

Date:	**December 03, 2024**
App Version No:	N/A
Total Duration:	7
Total no. of passed test cases:	6
Total no. of failed test cases:	1
Total no. of tested test cases:	0
Total no. of test cases:	6

STATUS	
% Passed Test Cases	90%
% Tested Test Cases	10

System Testing	
Project Number:	1.0
Project Name:	Student Management System: Enhanced Grade Tracking, Attendance Management and Advanced Lecture Integration.
Test Type:	System Testing
Test Description:	System testing tests all the processes of the system.

Date:	December 04, 2024
App Version No:	**1.0**
Total Duration:	**7**
Total no. of passed test cases:	**8**
Total no. of failed test cases:	**2**
Total no. of tested test cases:	**8**
Total no. of test cases:	**6**

STATUS	
% Passed Test Cases	80%
% Tested Test Cases	20%

APPENDIX D: EVALUATION FORM

Introduction

Dear Participants,

Thank you for participating in the evaluation of the Student Management System as part of out Capstone Project. Your Feedback is crucial for enhancing the system and ensuring it meets user expectations. We value your time and insights, which will greatly contribute to refining the system.

This evaluation uses a 4-point raising scale to capture your experience with various aspects of the system. Please choose the response that best reflects your experience. All responses will be kept confidential Thank you. Kindly return the completed form to the system proponents.

Part I. Graphical User Interface	**Ratings**			
Questions	**1 (Strongly Disagree)**	**2 (Disagree)**	**3 (Agree)**	**4 (Strongly Agree)**
The design elements (e.g., buttons, icons) are consistent throughout the interface.				
The font size and style used in the GUI are clear and readable.				
Overall, I find the interface easy to use.				
The design of the system is visually appealing				
Font style and size is easy to read				
The colors used in the interface were pleasing and appropriate.				
Buttons and icons are easy to identify.				
The layout of the pages is consistent and well-structured.				
The onboarding process (e.g., log in, sign up) is straightforward.				
The interface provides a clear and logical workflow for completing tasks.				

Comments:

_____

_____

_____

_____

Instruction: Please indicate your response by checking the corresponding rating number.

Part II. Usability	**Ratings**			
Questions	**1 (Strongly Disagree)**	**2 (Disagree)**	**3 (Agree)**	**4 (Strongly Agree)**
The SMS is easy to learn.				
Error messages within the system are clear and informative.				
Overall, I am satisfied with the usability of the SMS.				
The system loads quickly when first accessed.				
The system responds quickly when interacting with the features.				
There were no delays in completing tasks.				
The system runs smoothly without any crashes.				
The system performs consistently well under normal use.				
The system is stable even with multiple users.				
The interface provides a clear and logical workflow for completing tasks.				
The navigation between different features (e.g., grades, attendance) is intuitive and efficient.				

Comments:

_____

_____

_____

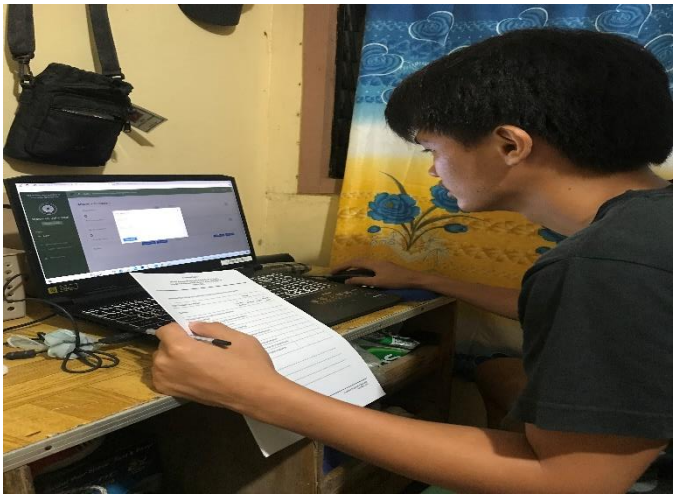
Instruction: Please indicate your response by checking the corresponding rating number.

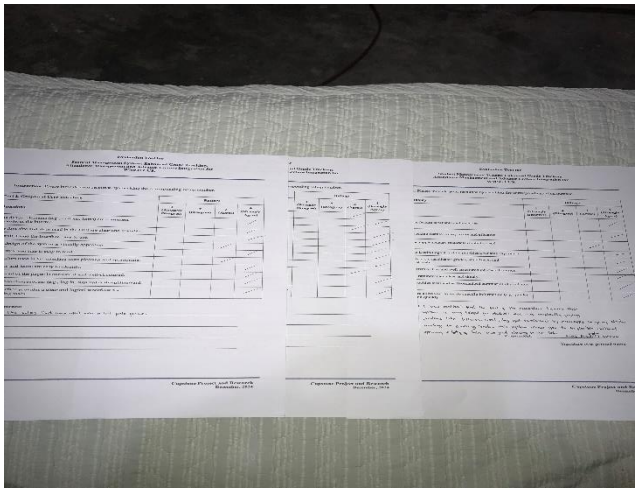
Part III. Functionality	**Ratings**			
Questions	**1 (Strongly Disagree)**	**2 (Disagree)**	**3 (Agree)**	**4 (Strongly Agree)**
The grade tracking feature is reliable and accurate.				
The attendance tracking feature is easy to use and effective.				
The repository for supplementary materials is intuitive and comprehensive.				
The User Account Login/Logout feature provides a reliable experience.				
The system effectively consolidates grades, attendance, and supplementary materials				
The Dashboard offers a clear and well-structured interface for users.				
Notifications and reminders are clear and timely.				
The dashboard provides clear and well-organized summaries of academic data.				
The search feature allows me to locate specific information (e.g., grades, subjects) easily and quickly.				

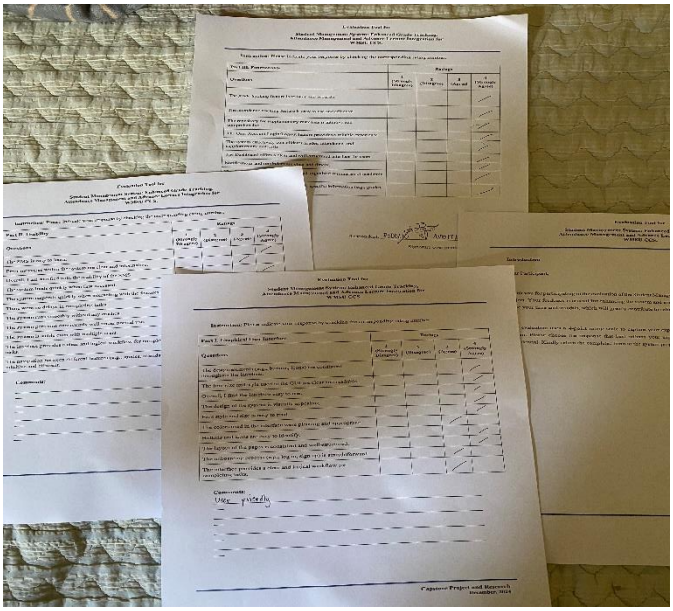
Comments:

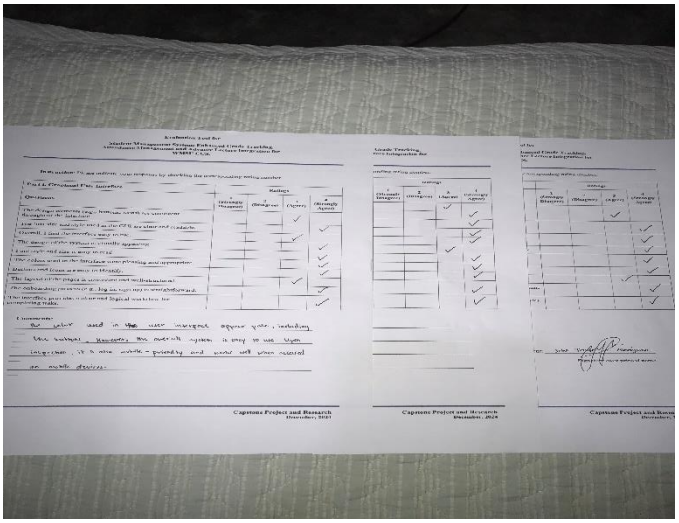
Respondent: _____

Signature over printed name









APPENDIX E

GANTT CHART

Task ID	Task Name	Duration (Days)	July				August				September				October To December			
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
	Requirement Gathering	10																
	Title Planning																	
	Consultation																	
I	Chapter 1: Introduction	5																
1.1	Project Context	4																
1.2	Purpose & Description	4																
1.3	Objectives	3																
1.4	Scope and Limitation	5																
1.5	Definition of Terms	2																

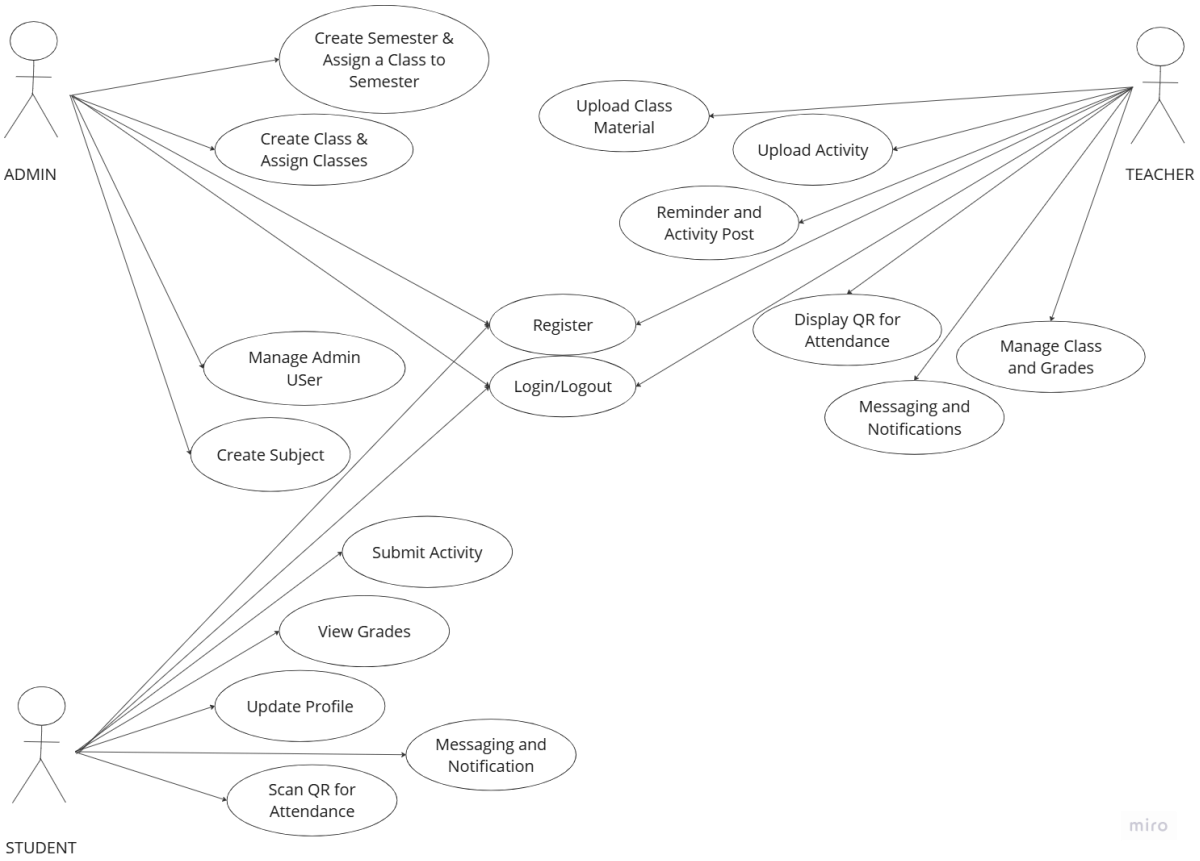
II	Chapter 2: Review Related Literature and Systems	8																
2.1	Foreign and Local Related Literature/ Studies	4																
2.2	Foreign and Local Related Systems	4																
2.3	Table of Comparison	1																
2.4	Synthesis	1																
III	Chapter 3: Technical Background	10																
3.1	The Technicality of The Project	5																
3.2	Details Of Technologies to Be Used	5																

3.3	How The Project Will Work	3																
IV	Chapter 4: Methodology	20																
4.1	Requirements Analysis	5																
4.2	Requirements Documentation	5																
4.3	Design Of Software, Systems, Product, and/or Process	11																
	Use Case Analysis																	
	Data Flow Diagram																	
	Entity-Relationship Diagram																	

4.4	Development And Testing Plan	60																
	Project Work Plan																	
	Project Teamwork																	
4.5	Description Prototype	14																
4.6	Implementation Plan	14																

APPENDIX F

APPENDIX F: USE CASE



Use Case #1: Log-in

User: Administrator, Teacher and Student

Description: The system must permit user to access the system.

Fit Criterion: The user must input the given username/email and password, and it should match the credentials to access the assigned module after log-in.

Use case scripts:

6. The log-in form is displayed.
7. The user enters a valid username/email and password
8. Upon clicking the “log-in” button, the system checks the entered credentials.
9. If valid, the user is granted access, displaying the message: “Login Succesful!”
10. If invalid, the system returns: “Account login error!”.

Use Case #2: Register Account

User: Administrator, Teacher and Student

Description: This allows the users to create an account in the system by providing necessary information.

Fit Criterion: The system verifies that all required fields are filled and that the username/email is unique before allowing registration.

Use case scripts:

6. The registration form is displayed.
7. The user inputs personal details
8. The system validates the inputs, checking for completeness and uniqueness of the username.
9. If all checks pass, the system registers the user and display: “Admin Addition successful!”
10. If there are errors like duplicate username, the system displays an appropriate error message.

Use Case #3: Upload Class Material User: Teacher

Description: This allows teachers to upload educational materials for their assigned classes.

Fit Criterion: The uploaded files should be linked to the correct class and made available to the students in that class

Use case scripts:

6. The teacher navigates to the “Upload Material” section.
7. The teacher selects the class and uploads relevant files.
8. The system verifies the file type and uploads it to the server
9. If successful, a message displays: “Material uploaded successfully.”
10. If an error occurs, the system display an error message and prompts the user to try again.

Use Case #4: Submit Activity User: Student

Description: This allows students to submit activities uploaded by the teacher.

Fit Criterion: The submitted activity is saved in the system and linked to the correct activity and student

Use case scripts:

1. The student navigates to the "Submit Activity" section.
2. The student selects the assignment and uploads the completed work.
3. The system checks the file type and saves the submission.
4. If successful, a message displays: "Activity Submitted Successfully"
5. If the upload fails, the system provides an error message.

Use Case #5: View Grades User: Student

Description: This allows students to view their grades for various assignments and overall class performance.

Fit Criterion: The system retrieves the student's grades from the database and displays them in an easy-to-read format.

Use case scripts:

6. The student selects the "Grade" option.
7. The system fetches the grades for the current semester from the database.
8. The grades are displayed, showing individual activity grades and overall scores.
9. If there are no grades available, the system displays a message: "Grades Not Yet Available."