PUP TAGUIG FACULTY LOADING AND SCHEDULING SYSTEM

A Capstone Project

Presented to

Polytechnic University of the Philippines – Taguig Campus

In Partial Fulfillment of the

Capstone Project 2

First Semester A.Y. 2024-25

Bachelor of Science in Information Technology

By

Martinez, Emmanuel Q.

Malaluan, Kyla Rica C.

Naoe, Adrian B.

Rasquero, Via Clariz A.

JULY 2024

CERTIFICATION OF ORIGINALITY

This is to certify that the research work presented in this capstone project, **DEVELOPMENT OF THE PUP TAGUIG FACULTY LOADING AND SCHEDULING SYSTEM** for the degree Bachelor of Science in Information Technology at the Polytechnic University of the Philippines embodies the result of original and scholarly work carried out by the undersigned. This capstone project does not contain words or ideas taken from published sources or written works that have been accepted as the basis for the award of a degree from any other higher education institution, except where proper referencing and acknowledgement were made.

EMMANUEL Q. MARTINEZ
Researcher

KYLA RICA C. MALALUAN
Researcher

ADRIAN B. NAOE
Researcher

VIA CLARIZ A. RASQUERO
Researcher

_____
Date Signed

CHAPTER 1

THE PROBLEM AND ITS BACKGROUND

1. Introduction

In today's rapidly evolving academic landscape, efficient faculty resource management and course scheduling are essential for maintaining a dynamic learning environment. The Polytechnic University of the Philippines (PUP) Taguig, with its strong legacy and commitment to excellence, has continually worked to optimize its administrative processes to meet the growing demands of its diverse student body.

The PUP Taguig Faculty Loading and Scheduling System is a vital component in the university's operations, originating in an era dominated by manual methods of academic resource allocation. Initially developed to address the need for a more organized approach to faculty assignments, it has supported the university's growth over the years.

However, as academic programs expanded, accreditation standards evolved, and new pedagogical approaches emerged, the system's limitations became increasingly evident. What once served as a functional solution has struggled to keep pace with the demands of a modern educational environment. Manual entry processes, inflexibility in adapting to new curricula, and a dated user interface have led to inefficiencies in faculty resource management and scheduling.

These challenges are not unique to PUP Taguig but reflect a wider trend among educational institutions globally. The difficulties of adapting traditional

systems to contemporary educational needs highlight the need for innovative solutions that integrate technological advancements with the complexities of academic administration.

As academic programs diversified, accreditation standards evolved, and technology became an integral part of education, the limitations of the existing system became increasingly apparent. Manual entry processes, lack of adaptability to new curricula, and a user interface that struggled to meet modern usability standards have cumulatively contributed to inefficiencies in faculty resource allocation and scheduling.

The researchers aim to explore the historical development of the PUP Taguig Faculty Loading and Scheduling System in relation to the current demands of educational administration. By understanding its origins, the researchers can gain a deeper perspective on the factors that have shaped its current structure. The challenges of the past will help identify opportunities for improvement—focusing on enhancing the system through the integration of evolving curricula, technological advancements, and user-centered design principles.

As the researchers delve into the existing PUP Taguig Faculty Loading and Scheduling System, their intent is not only to address existing deficiencies but also to contribute to the enhancement and progress that will propel PUP's academic administration into a future defined by efficiency, adaptability, and a commitment to providing an optimal learning experience for both faculty and students.

1.1 Overview of the Current System Process

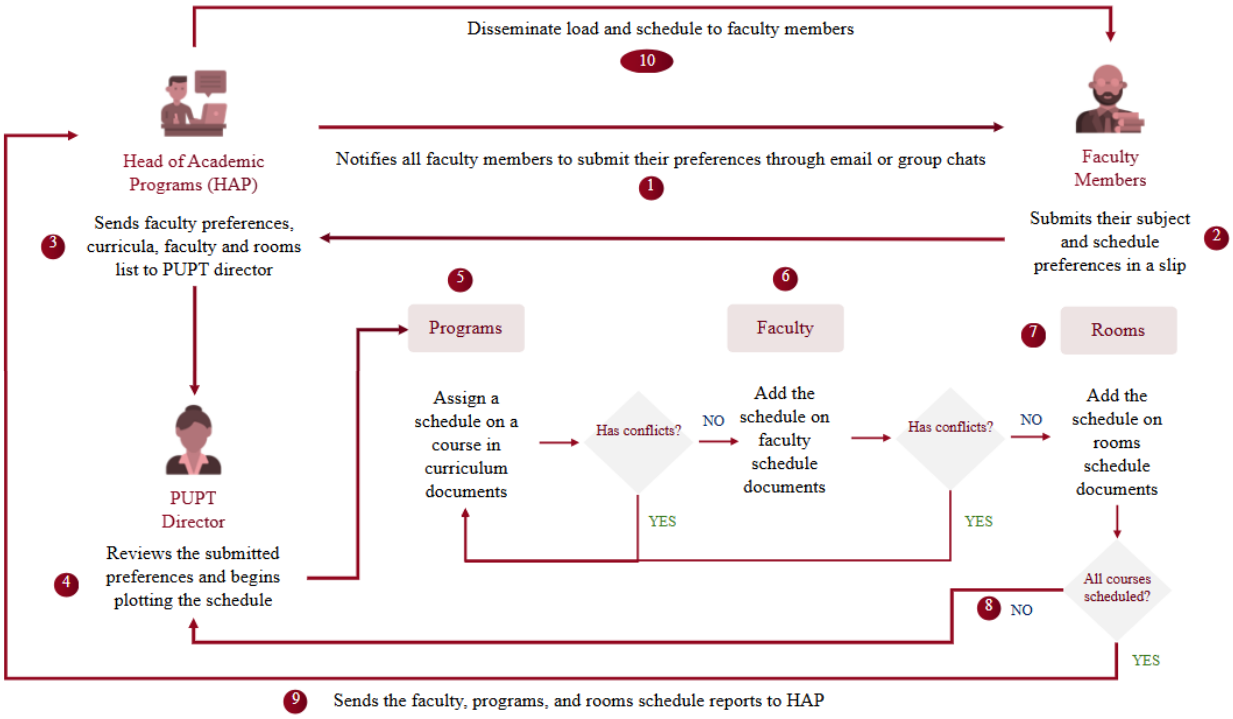


Figure 1. The current process flow of the PUPT Faculty Loading and Scheduling

1.2 Background of the Organization

1.2.1. Organizational Hierarchy

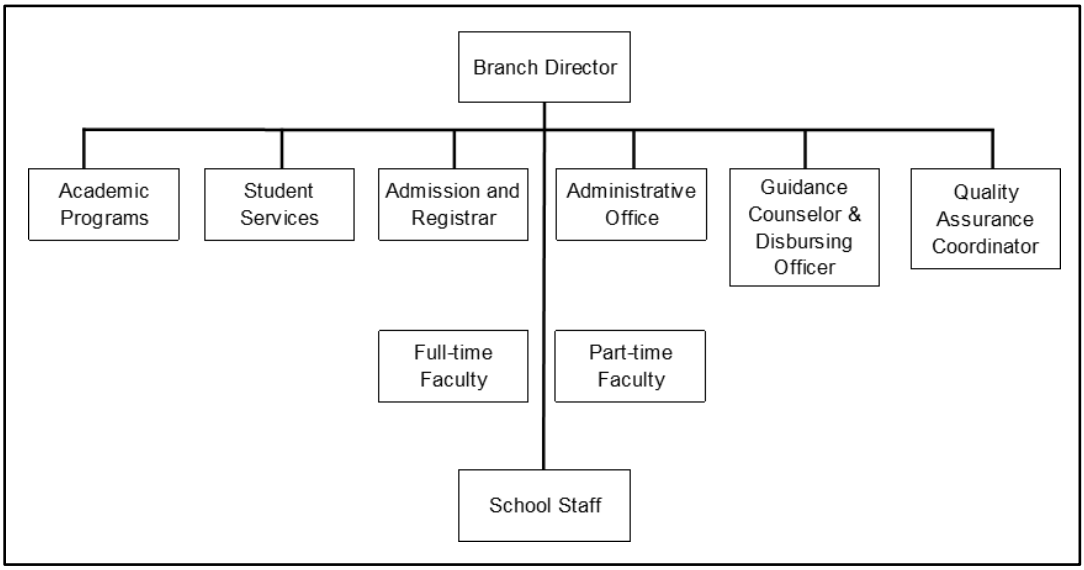


Figure 2. Organizational Hierarchy Overview at PUP Taguig

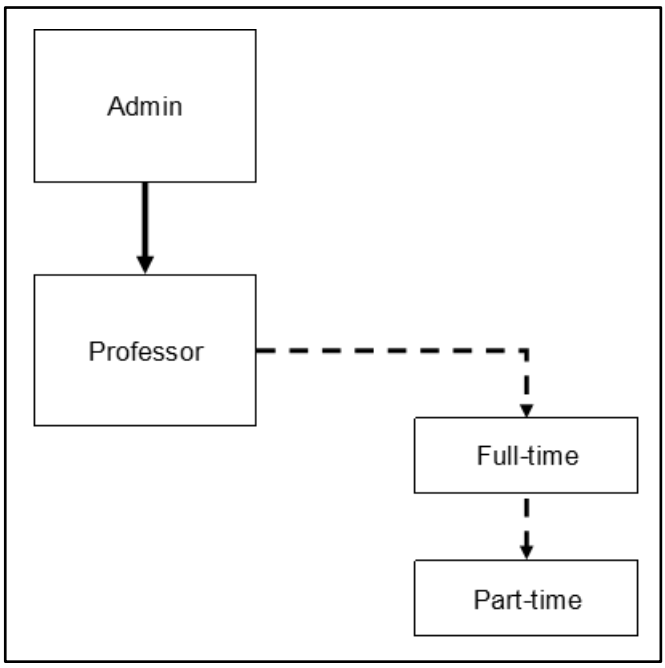


Figure 3. The Flow of Assigning Faculty Roles in PUP Taguig FLS System

1.2.2. Policies and Procedures

The following are the policies and procedures of PUP Taguig in terms of faculty loading:

- **Classification of Faculty Members**

Faculty Classification	Teaching Load (Units)	Details
Full-Time Faculty	15 units (Regular)	Faculty members who are permanently employed and typically have a full load of teaching and other academic duties.
Part-Time Faculty	12 units	Faculty members employed on a short-term or semestral basis, typically teaching fewer hours than full-time faculty.
Permanent Faculty (Designee)	6 units	Faculty who hold permanent positions but may have administrative duties, which may reduce their teaching load.

Table 1. Faculty Classifications and Teaching Loads

The 2024 Faculty Manual outlines the teaching load and units allowed for different types of faculty members at the Polytechnic University of the Philippines. Below is an analysis of the different types of faculty and the units allowed for each:

1. **Full-Time Faculty**
   - **Teaching Load**: Full-time faculty members are assigned a **standard teaching load of 15 units** per semester. This includes **15 hours of**

teaching per week, along with time dedicated to **research, extension, and consultation duties.**

- **Additional Teaching:** Full-time faculty may also take on additional **12 hours per week** in a **part-time capacity**, still under their full-time employment status. These extra hours are included within the **40-hour workweek limit**, which includes teaching, research, and other duties.
- **Temporary Substitution (TS):** If full-time faculty wish to exceed their regular 15 hours plus 12 part-time hours (i.e., beyond 27 hours of teaching), they can take on up to **12 additional hours** of teaching, but this extra time is considered **Temporary Substitution (TS)**. These hours are included in their total workweek, and **the total workweek cannot exceed 40 hours per week**. This includes teaching, research, extension, and other duties.

2. **Part-Time Faculty**
    - **Teaching Load:** Part-time faculty members can teach a **maximum of 12 units** per semester. If their teaching load exceeds this, any additional hours are considered **Temporary Substitution (TS)**.
    - **Appointment:** Part-time faculty members are typically appointed on a **semestral basis**, depending on the available teaching load for each semester. They are generally not involved in other university duties outside their assigned teaching hours.

3. **Permanent Faculty (Designee)**
    - **Teaching Load:** Permanent faculty who hold administrative roles (e.g., Deans, Directors, Associate Deans, Chairpersons, Faculty Assistants, etc.) are assigned **6 hours of teaching per week**. These faculty members have reduced teaching loads due to their administrative responsibilities, allowing them to focus on their administrative duties while still maintaining teaching roles.

- **Excess Hours:** If their teaching exceeds **6 hours**, any additional hours are treated as part-time teaching. These extra hours are typically scheduled between **7:30 PM and 9:00 PM**, ensuring that administrative duties are still prioritized.

4. **Special Designations**
    - Faculty members involved in **extension services** or other **special designations** (e.g., research assignments, specialized teaching responsibilities) may be eligible for a **reduction in their regular teaching load**, which may be up to **6 units**. This reduction allows faculty to accommodate their specialized roles while maintaining their primary teaching duties. Faculty members may still handle **part-time teaching loads**, but they are capped at 12 units.

Factors in selecting teaching load and assigning schedules

Faculty availability:

- This considers the preferred schedules submitted by the faculty members and their preferred subjects

Faculty academic qualifications:

- The faculty meets the minimum requirements of educational level for which he/she is being hired.

Faculty Evaluation:

- Refers to the result of the semestral evaluation from the students. If a faculty member has a three (3) consecutive evaluation grade below 2.0 (“fair”), their maximum load will be reduced

Room availability

- Refers to the available regular and laboratory rooms

Course availability

- Refers to the availability of offered courses under a certain program

Program availability

- Refers to the availability of offered programs in a school year

Administrative Roles

- If a faculty member is a branch official or designee, their teaching load is reduced compared to the full-time/regular faculty

1.3 Statement of the Problem

The current PUP Faculty Loading and Scheduling System face inefficiencies in adapting to the evolving educational landscape. Manual processes and limitations in system capabilities hinder the optimal allocation of faculty resources and the seamless integration of new curricula. These challenges contribute to scheduling conflicts, reduced system performance, and a less-than-optimal user experience. Addressing these issues is critical to fostering an effective and adaptive academic environment.

The specific challenges in the Faculty Loading and Scheduling project are as follows:

- *Inefficient and Time-Consuming Manual Processes:* The current process requires excessive paperwork and repeated data entry as each assignment must be recorded multiple times across different piles of information (faculty, rooms, courses, and programs) which leads to substantial administrative workload burden.

- *High Error Rate and Inconsistencies:* The manual handling of scheduling tasks often results high error rates in assigning faculty, rooms, and time

slots, leading to scheduling conflicts, missing schedules, and inconsistencies across curricula impacting both faculty and students.

- *Failure of Existing Online Solutions:* Previous attempts to digitize the scheduling process have failed due to outdated UI/UX and critical functional flaws such as inability to handle multiple curricula, resulting in lack of adoption and continued dependence on inefficient manual methods.

- *Lack of Integrated Data Management:* The absence of a centralized platform for managing faculty preferences, curriculum requirements, program details, and room allocations leads to fragmented data handling, making it difficult to generate coherent and optimized schedules.

1.4 Theoretical Framework

The development of the enhanced PUP Taguig Faculty Loading and Scheduling System (FLSS) is grounded in the theoretical concepts established in the study conducted by Raymundo et al. (2016) titled *Development of an Automated Faculty Loading, Room Utilization, Subject, and Student Scheduling System (AFLRUS4)* for Bulacan Polytechnic College. This study serves as a primary framework, offering insights and evidence that underpin the FLSS's focus on automation, efficiency, and data management. Additionally, the ISO/IEC 25010:2011 standards for software quality and the Open Web Application Security Project (OWASP) framework are integrated to provide a comprehensive theoretical foundation, particularly addressing system reliability and security.

The findings of Raymundo et al. provide tangible evidence of the positive impact of automation on scheduling processes within academic institutions. The AFLRUS4 system demonstrated the capability to streamline intricate scheduling

tasks, reducing manual effort and minimizing the likelihood of errors. This aligns seamlessly with the theoretical concept of efficiency embedded in the FLSS framework. The study illustrates how automation expedites scheduling procedures, contributing to more accurate and error-free allocation of faculty resources. Furthermore, Raymundo et al. emphasize the importance of flexibility in academic environments, which are dynamic and subject to changes in faculty availability, student enrollment, and course requirements. The AFLRUS4 system's adaptability underscores the need for a flexible FLSS that can adjust to the evolving demands of academic operations, a principle that is central to the theoretical framework of this study.

In addition to the AFLRUS4 framework, the ISO/IEC 25010:2011 standards provide essential guidelines for ensuring system quality and functionality. These standards emphasize functional suitability, usability, reliability, and scalability, ensuring that the FLSS meets academic requirements while maintaining high performance and accommodating future growth. Meanwhile, the OWASP framework addresses the security dimensions of the FLSS by guiding the implementation of secure authentication, data encryption, and robust protection against vulnerabilities. These measures ensure the confidentiality, integrity, and availability of sensitive academic data.

Together, the AFLRUS4 study, ISO standards, and OWASP framework form the theoretical backbone of the FLSS. These integrated frameworks ensure that the system is efficient, flexible, secure, and reliable, allowing it to meet the dynamic needs of academic institutions while adhering to high standards of quality and security.

1.5 Conceptual Framework

The *Optimizing Faculty Workload Management: A Revamped Approach for PUP Taguig's Faculty Loading and Scheduling System* aims to improve the overall performance of this system for all users. This study is inspired by the Input-Process-Output (IPO) model for enhancing the scheduling and loading system. It enables users to add new curricula, access availability schedules, store data in the database, and experience a seamlessly user-friendly interface while using this system.

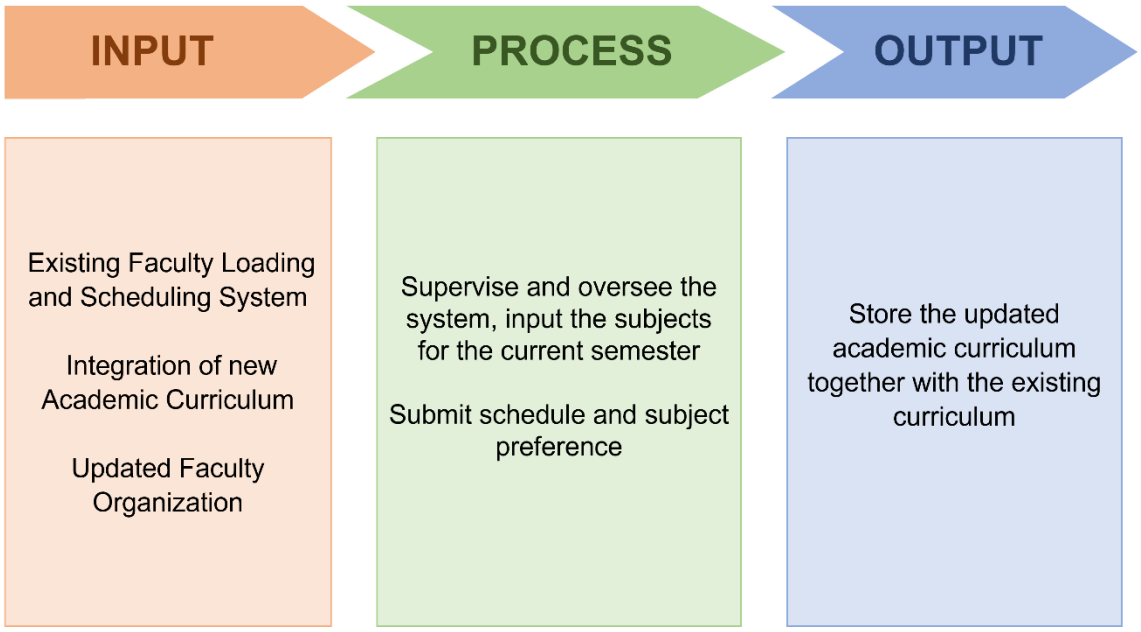


Figure 4. Input-Process-Output Model of the PUPT Faculty Loading and Scheduling System

1.6 Objectives

General Objectives

The overarching goal of this project is to optimize the management of faculty workload and class scheduling at PUP Taguig. By introducing a more comprehensive and innovative approach, the enhanced system aims to provide

administrators with a powerful tool to navigate the intricacies of faculty workload and scheduling.

Specific Objectives

In pursuit of optimizing the Faculty Loading and Scheduling System, the project is set to achieve key milestones, as outlined in the following objectives:

- *Streamlined Scheduling Process:* To develop a web-based app that streamlines the faculty loading and scheduling process, eliminating excessive paperwork and repeated data entry across different categories, thereby decreasing the administrative workload and improving efficiency.

- *Error Reduction and Conflict Prevention:* To implement conflict detection features within the system to eliminate errors associated with manual scheduling to ensure that faculty assignments, room allocation, and schedules are accurate, reliable, and free of conflicts.

- *Modern, User-Friendly System Design:* To design a modern, user-friendly, and fully functional web-based scheduling system that addresses the limitations of previous systems, including the inability to handle multiple curricula, encouraging widespread adoption for faculty and administrators.

- *Integrated Data Management Platform:* To establish an integrated data management platform that centralizes faculty preferences, curriculum requirements, program details, and room allocations, facilitating seamless data handling and enabling the generation of optimized and coherent schedules.

1.7 Scope and Delimitation

This study focuses on implementing specific key enhancements to the existing PUP Taguig Faculty Loading and Scheduling System. These enhancements include the integration of the new curriculum, migration to the LARAVEL framework, and a redesign of the user interface for improved usability and accessibility. Academic stakeholders such as Academic Administrators, Curriculum Developers, and Faculty Members who interact with the system will be primary involved. Their experiences and feedback are crucial for evaluating the effectiveness of the enhancements. The scope also includes the measurement and analysis of system performance metrics, such as response time, resource utilization, and overall efficiency. These metrics will provide quantitative insights into the impact of the enhancements.

On the other hand, this study is limited to the Faculty Loading and Scheduling System within PUP Taguig, specifically focusing on improvements related to faculty load assignment, scheduling processes, user interface design, and conflict detection and validation. The study aims to enhance the allocation and distribution of teaching loads, improve the automation and efficiency of scheduling classes, rooms, and programs, and redesign the system interface to ensure it is intuitive, user-friendly, and accessible. Additionally, a conflict detection feature will be implemented to automatically resolve scheduling conflicts involving rooms, programs, and faculty availability. This study does not extend to other systems or departments within PUP Taguig nor any other PUP branches. Integration with external systems beyond the scope of the faculty loading and scheduling processes will not be addressed in this research. While security features will be implemented and evaluated in the enhanced system, the study does not involve ethical hacking or extensive security testing beyond the scope of system evaluation.

The study does not extend to a long-term impact assessment of the enhanced system beyond the immediate post-implementation phase. The focus is on the initial evaluation and user feedback. Furthermore, this study also acknowledges that certain technical limitations may exist, such as hardware constraints or compatibility issues with the legacy system. These constraints may influence the implementation of certain enhancements.

1.8 Significance of the Study

A system capable of seamlessly and efficiently loading and scheduling is crucial for professors. Having such a system implemented in PUP-Taguig holds significant importance for the institution's performance and operation. Specifically, the following will benefit from the enhancement of the PUP Taguig Faculty Loading and Scheduling System:

Academic Administrators: Academic administrators will experience streamlined decision-making processes for faculty workload allocation, resulting in a balanced and optimized distribution of teaching assignments.

Curriculum Developers: Curriculum developers will benefit from an enhanced system capable of seamlessly integrating new curricula. This ensures that course offerings align with emerging academic standards, providing a more relevant and engaging learning experience for students.

Faculty Members: Faculty members will enjoy an improved user experience and increased efficiency in managing their teaching assignments. The redesigned user interface and enhanced functionalities contribute to higher satisfaction and a more user-friendly interaction with the system.

Students: Students will indirectly benefit from the optimized faculty loading and scheduling system. A well-organized and efficiently managed system

contributes to a smoother academic environment, potentially reducing scheduling conflicts and ensuring a more consistent learning experience.

Accreditation Bodies: Accreditation bodies responsible for evaluating the academic quality of the institution may benefit indirectly from a system that ensures compliance with accreditation standards through efficient faculty assignment and curriculum integration.

Information Technology (IT) Personnel: IT personnel responsible for system maintenance will benefit from the migration to the LARAVEL framework, which provides a more modern, scalable, and maintainable technological infrastructure. This ensures easier maintenance and future adaptability.

Future Researchers and Developers: The findings and best practices established in this study serve as a valuable resource for future researchers and developers in the field of academic administration. The study contributes to the body of knowledge on the integration of research findings into the technological enhancement of educational systems.

1.9 Definition of Terms

Faculty Load: The total amount of teaching hours, courses, or credits assigned to a faculty member within a given period. It includes classroom teaching, preparation time, and other academic responsibilities.

Class Schedule: A timetable that outlines the specific days and times that courses are offered, including room assignments, faculty, and student enrollment.

Course Allocation: The process of assigning faculty members to specific courses, ensuring a balanced distribution of teaching load, expertise, and availability.

Units: The credit value assigned to a course, typically reflecting the number of hours a student is expected to spend in class or studying. Faculty load is often determined by the number of units a course carries.

Preferences: The specific requests or priorities of faculty members regarding their teaching schedule, including preferred time slots, days, courses, or teaching locations. These are considered during the scheduling process.

Academic Year: The defined period during which an academic institution conducts its educational programs, usually consisting of two semesters or terms. It outlines the schedule for courses, holidays, and faculty assignments.

Curricula: The structured set of courses and academic content offered by a department or institution, detailing the subjects, learning outcomes, and required courses for a particular program or degree.

CHAPTER 2

REVIEW OF RELATED LITERATURE AND STUDIES

2.1 Introduction

This chapter aims to provide a comprehensive understanding of the existing relevant literature and studies related to the development and implementation of faculty loading and scheduling systems in the context of higher educational institutions. The following studies showcase the efficient distribution of faculty responsibilities, and the creation of optimal scheduling systems are essential components in fostering a conducive academic environment. This review will lay the groundwork for the rationale, theoretical framework, and methodologies employed for this study.

2.2 Faculty Workload Optimization and Scheduling Systems

The study of Abed (2021) titled *Development Of A Faculty Workload Management System Using Optimization Algorithms* addresses the prevalent concern of faculty workload imbalance in Jordan Universities by proposing a novel Faculty Workload Management System (FWMS) utilizing optimization algorithms. Recognizing the inherent efficiencies of manual scheduling, the FWMS leverages genetic algorithms to meticulously assign courses to faculty members, considering their expertise, preferences, and pre-defined workload limits. This innovative approach demonstrably reduced the number of overloaded faculty members, contributing to improved faculty well-being and potentially enhanced student learning outcomes. Abed's study emphasizes the critical roleof accurate data and user acceptance in ensuring the successful implementation and sustained efficacy of such systems within academic institutions.

In their research work titled *ClassSchedule: A Web-Based Application for School Class Scheduling with Real-Time Lazy Loading*, Nuengwong et al. (2018) introduced an innovative method to address the complexities associated with class scheduling in educational institutions. The study delves into the intricate management of resources, encompassing teachers, students, subjects, classrooms, and periods, and highlights the inherent limitations of traditional manual scheduling methods employing pens and paper. The authors underscore the insufficiencies of existing software and algorithms in meeting the diverse demands imposed by distinct departments and overarching school policies. In response to these challenges, they developed *ClassSchedule*—a sophisticated web-based application utilizing technologies such as PHP, HTML, CSS, AJAX, JavaScript, and MySQL. This advanced tool represents a departure from conventional approaches, streamlining the scheduling process and providing educators with the capability to schedule classes and access real-time data seamlessly. Key features of *ClassSchedule* include the facilitation of collaborative scheduling among multiple department representatives, allowing for simultaneous editing of scheduling data and real-time assessment of resource collisions.

Meanwhile, the Development of Online-Class Scheduling Management System Conducted by the Case Study of Department of Social Science: Faculty of Humanities and Social Sciences Suan Sunandha Rajabhat University" by Wipada Chaiwchan and Patcharee Klinhom, explores the intricacies of implementing an online-class scheduling management system. The study places a primary emphasis on enhancing the efficiency and convenience of class scheduling procedures for educators and students alike. Executed as a web-based application, the system is meticulously developed using PHP as the programming language and MySQL as the designated database management system. The research delves into the nuanced nature of class scheduling, taking into account factors such as course quantities, student volumes, and the specific physical attributes of individual

classrooms. The scholarly discourse unfolds by highlighting the transformative role of information technology in shaping educational paradigms and facilitating communication. The authors illustrate how technology, particularly online scheduling systems, streamlines class schedule management, benefiting both faculty and students. The study further investigates the application of genetic algorithms in solving complex scheduling challenges, showcasing their effectiveness in reducing complexity, saving labor time, and minimizing costs. Methodologically, the research employs a comprehensive evaluation strategy, incorporating Black-Box testing and obtaining feedback from both experts and general users. The positive outcomes of the evaluation affirm the system's high satisfaction levels among users, indicating its potential for effective implementation in authentic educational settings. In summary, the study underscores the profound significance of technologically-driven solutions, exemplified by the online-class scheduling management system, in improving educational processes and meeting user requirements.

Faculty members are at the forefront in every learning institution. They teach and nurture the minds of students. The tasks of faculty members are not only limited to teaching; they check the large volume of test papers, which add burden to their work. This is due to the reason that faculty members handle as much as fifteen teaching loads every semester. The more teaching loads, the higher the salary they received (Etcuban, 2013) This study emphasizes the pivotal role of faculty members within educational institutions. These professionals are responsible for educating and guiding students, and their duties extend beyond teaching. They also have to manage a significant volume of test papers, which contributes to their workload. This added burden is notably high because faculty members typically handle up to fifteen teaching assignments per semester. Interestingly, their compensation often correlates with the number of teaching assignments they undertake, meaning that higher teaching loads lead to increased salary, as mentioned by Etcuban (2013).

2.3 Historical Perspectives on Manual Scheduling

Miranda (2010) explained that a university, historically, subjects were scheduled its courses and assigned classrooms manually by trial and error, a slow and tedious process with average execution times that had ballooned to four weeks. In the article entitled Solving the Problem in timetabling (2010), typically, a manual solution requires expert attention and can take many weeks for large educational establishments. Moreover, because of the problem complexity, planners are not always able to make the best decisions, building schedules that are inconsistent with teaching requirements and do not satisfy all teachers' needs. Miranda's (2010) study reveals the historical reliance of universities on manual methods for scheduling courses and assigning classrooms, a slow and laborious process that involved trial and error, taking an average of four weeks for execution. This manual approach, as detailed in

"Solving the Problem in Timetabling" (2010), demanded expert attention and often consumed numerous weeks, particularly in larger educational institutions. However, due to the intricate nature of the task, planners encountered difficulties in making optimal decisions. Consequently, the resultant schedules frequently fell short in meeting teaching requirements and satisfying the diverse needs of teachers, highlighting inconsistencies and inefficiencies within the scheduling system in educational settings.

2.4 University Scheduling Challenges: A Philippine Context

Baccay & Cabahug (2019) address the multifaceted challenges of university scheduling in the Philippines. Their Class Scheduling and Faculty Loading System (CSFLS) utilizes a modified particle swarm optimization algorithm to generate efficient schedules, considering course requirements, room availability, faculty

preferences, and minimizing conflicts. The study extends its optimization scope beyond workload management to include classroom utilization, showcasing its potential for broader resource optimization within universities. While recommending further testing and user training for enhanced adoption and user satisfaction, Baccay & Cabahug's work demonstrates the effectiveness of optimization algorithms in addressing complex challenges in university scheduling.

In a study conducted by Botangen (2016), Central Luzon State University struggles with challenges in their manual scheduling system, encompassing the laborious task of organizing classes, recurrent delays in finalizing schedules, and errors leading to conflicts. Face-to-face discussions among registrars are essential for conflict resolution yet hindered by geographical distances and varying priorities among units. To tackle these hurdles, the university adopted a web-based Class-Scheduling system, integrating technologies like Apache, PHP/HTML/JavaScript, and MySQL, fostering collaboration among dispersed registrars. This innovative system operates through block, room, and faculty scheduling categories, streamlining outcomes to create a cohesive schedule. Despite challenges like delays and conflicts, the system's integration of compatible outcomes ensures a comprehensive and optimized scheduling solution.

According to Louie et al. (2018), in their paper titled "Predictive Analytics Implementing Genetic Algorithm on Class Scheduling System," concerns related to faculty course-loading and student accommodation at Trimex Colleges in Laguna, Philippines are addressed. With a notable increase in student population, there is a need for an efficient class scheduling and faculty course-loading system. The primary objective of this study is to develop a system capable of swiftly generating class schedules and faculty course-loading while predicting the demand for course sections based on data from previous terms. Their study introduces an inclusive

system that integrates predictive analytics and genetic algorithms to address challenges associated with class scheduling and faculty course loading at Trimex Colleges. The system's efficiency lies in its ability to swiftly generate class schedules and faculty course loading, resulting in substantial time and resource savings. Furthermore, the system exhibits predictive capabilities, anticipating the need for course sections by leveraging insights derived from historical data. The results of the software evaluation attest to the system's success in achieving the outlined objectives, positioning it as a valuable tool for Trimex Colleges.

In an article by Ortega et. al (2015) titled, *Online Class Scheduling and Faculty Loading System within a Decision Support Framework*, they discussed the challenges faced by academic department heads in creating course schedules to accommodate student enrollment needs. The study focuses on developing a class scheduling and loading system using a knowledge-based approach, heuristic functions, and rule sets within a decision support framework. The goal is to optimize available resources, such as classrooms and faculty, and enhance the efficiency of the scheduling process. The research identifies key processes in creating course schedules, including regular offerings, checking teacher availability, and allocating breaks.

Constraints, such as room availability, faculty preferences, and workload considerations, are highlighted. The study emphasizes the importance of aligning faculty members with their qualifications and preferences, contributing to the overall effectiveness of the scheduling process. The article introduces an online system prototype that allows for data entry, room availability checks, and faculty assignment validation, aiming to streamline the scheduling and loading procedures. Overall, the research concludes that the developed online system is feasible and recommends continuous improvement and maintenance for enhanced functionality and

adaptability to evolving constraints and requirements. The translated outcome of the study is an operational web system used by the University of Cebu-Banilad Campus for class scheduling and faculty loading, contributing to better resource management and overall efficiency.

In a study conducted by Jalandoon et. al (2023) titled *Development of Faculty Loading and Credential Management for STI College Alabang,* they tackled the challenges in manually creating schedules for each room and section, and manually assigning loads to faculty. They assessed the level of efficiency and effectiveness of a developed web-based online system in terms of functionality, efficiency, compatibility, usability, reliability, security, and maintainability. They also looked into the satisfaction of end users based off system performance in terms of speed, reliability, functionality, and usability. Their study found out that the developed system meets the mentioned standards and increased the efficiency in managing teaching load in the said university.

2.5 Synthesis and Relevance to the Current Study

The reviewed literature and studies present a comprehensive landscape of faculty workload optimization and scheduling systems in the higher education context, each offering unique insights and approaches. Drawing upon these studies is essential to establish a foundation for the rationale, theoretical framework, and methodologies applied in the current study titled "Optimizing Faculty Workload: Enhancing the PUP Taguig Faculty Loading and Scheduling System." The synthesis of these works provides valuable perspectives on the challenges faced by academic institutions and the innovative solutions devised to address them.

In examining Abed's (2021) Faculty Workload Management System (FWMS) and Nuengwong et al.'s (2018) ClassSchedule, the integration of optimization algorithms emerges as a crucial component in achieving efficient faculty assignments and streamlined class scheduling. These studies collectively advocate for the adoption of genetic algorithms to meticulously match faculty expertise, preferences, and workload limits, resulting in improved well-being for faculty members and potentially enhanced student learning outcomes.

The exploration of technology-driven solutions by Chaiwchan and Klinhom (2014) and Ortega et al. (2015) adds another layer to the synthesis. The implementation of web-based scheduling systems showcases the transformative impact of technology on educational processes. Leveraging PHP, HTML, CSS, AJAX, JavaScript, and MySQL, these systems not only streamline class schedule management but also provide real-time data access and collaborative scheduling features. The technological emphasis resonates with the contemporary landscape of educational technologies, aligning with the overarching goal of our study to enhance the PUP Taguig Faculty Loading and Scheduling System.

Miranda's (2010) historical perspective on manual scheduling serves as a crucial reminder of the challenges faced by universities in the past. The slow and error-prone nature of manual processes, taking weeks for execution, emphasizes the need for modern, efficient solutions. Our study acknowledges this historical context and endeavors to propel the PUP Taguig Faculty Loading and Scheduling System into a new era of effectiveness.

Addressing challenges specific to the Philippine context, Baccay & Cabahug's (2019) CSFLS, Botangen's (2016) case study on Central Luzon State

University, and Louie et al.'s (2018) predictive analytics approach for Trimex Colleges underscore the multifaceted issues faced by Philippine universities. These challenges range from laborious manual scheduling systems and recurrent delays to the need for predictive analytics in accommodating a growing student population. Our study, focusing on PUP Taguig, aligns with these contextual challenges and endeavors to provide tailored solutions that consider the unique dynamics of the Philippine higher education landscape.

The synthesis of these diverse studies reinforces the significance of our research objectives in optimizing the PUP Taguig Faculty Loading and Scheduling System. By integrating optimization algorithms, adopting technology-driven solutions, acknowledging historical challenges, and addressing specific Philippine contextual issues, our study aims to enhance the existing system. The infusion of a new curriculum, migration to a new framework, and revamping the user interface align with the evolving landscape of educational technologies and best practices identified in the literature. The insights gained from these studies will guide the implementation and evaluation phases of our proposed enhancements, ensuring a more robust and efficient faculty workload optimization and scheduling system at PUP Taguig.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter outlines the systematic approach undertaken to conduct the research, providing a comprehensive overview of the research design, data sources, construction of research instruments, data generation procedures, ethical considerations, and the proposed structure of the system under investigation.

3.1 Research Design

This study adopts a quantitative research design, focusing on the systematic collection and analysis of numerical data to assess the impact of enhancements to the Polytechnic University of the Philippines Taguig Faculty Loading and Scheduling System. The quantitative research design emphasizes objective measurement, statistical analysis, and structured data collection methods to evaluate specific metrics related to system performance, efficiency, and user satisfaction.

3.1.1 Quantitative Measures:

The research design involves the quantification of key variables through structured surveys and questionnaires. Participants, including academic administrators and faculty members, will provide numerical responses using Likert scales and closed-ended questions. The survey instruments are designed to capture quantitative data on user satisfaction, perceived system efficiency, and overall feedback on the enhanced features.

3.1.2 System Performance Metrics: (REVISED)

Objective metrics related to system performance, response time, and resource utilization will be automatically collected by the enhanced Faculty Loading and Scheduling System through built-in monitoring tools. System performance will be assessed by evaluating the overall efficiency of the system in executing faculty load assignments, scheduling tasks, and processing user inputs. Key indicators, such as the number of successful transactions processed per minute and the system's ability to handle concurrent user requests without significant slowdowns, will be tracked. Response time will be measured by recording the time it takes for the system to respond to user actions, such as loading schedules, submitting faculty load assignments, or generating reports. This will be tracked in milliseconds or seconds, with measurements taken during typical and peak usage scenarios to evaluate if the system meets acceptable performance standards. Resource utilization will focus on monitoring the system's use of computational resources, such as CPU, memory, and storage, during typical operations like faculty load assignments and scheduling processes. This will be done through system logs and resource monitoring tools, helping assess whether the system is optimizing hardware usage and functioning within acceptable limits. The data generated by the system, including log files, analytics, and real-time performance reports, will serve as the primary sources for quantitative analysis. These metrics will provide critical insights into the impact of the system enhancements on overall system performance, operational efficiency, and resource management.

3.2 Research Instrument

In this study, the research instrument is crafted in accordance with the ISO 25010 standard, a comprehensive framework for evaluating software product quality and system performance. ISO 25010 outlines a set of quality characteristics and sub-characteristics crucial for assessing the efficiency, usability, and reliability of

software systems. Specifically, the research instrument employed in this study takes the form of structured surveys and questionnaires. These instruments are designed to align with internationally recognized criteria for software quality, as outlined in the ISO standard, ensuring a systematic and rigorous evaluation of the targeted aspects of software performance.

3.2.1 ISO 25010 Quality Characteristics:

The research instrument will focus on the following ISO 25010 quality characteristics, tailored to the specific context of the enhanced Polytechnic University of the Philippines (PUP) Faculty Loading and Scheduling System:

1. **Performance Efficiency:**

- *Aspect of Focus:* Evaluation of system performance metrics, including response time and resource utilization.
- *Instrumentation:* Automated tracking of response time, system-generated metrics related to efficiency, and participant feedback on perceived system speed and responsiveness.

2. **Usability:**

- *Aspect of Focus:* Assessment of user interface design, accessibility, and overall user experience.
- *Instrumentation:* Structured surveys and questionnaires based on ISO 25010 sub-characteristics such as learnability, operability, and user satisfaction. These instruments will gather quantitative data on user perceptions of the system's usability.

3. **Reliability:**

- *Aspect of Focus:* Evaluation of system stability, accuracy, and error handling.

- *Instrumentation:* Automated collection of system-generated data on error rates and reliability. Surveys will also include questions related to users' experiences with system reliability.

3.2.2 Data Collection Methods:

The research instrument employs a combination of automated system metrics and participant responses through surveys and questionnaires. The system will automatically generate data related to performance and reliability, while structured surveys will capture user perceptions aligned with the ISO 25010 usability characteristics.

3.2.3 Structured Surveys and Questionnaires:

Structured surveys and questionnaires will be designed based on ISO 25010 guidelines to gather quantitative data from participants. These instruments will include Likert scales and closed-ended questions, enabling participants to provide numerical feedback on specific aspects of system performance, usability, and reliability.

By adopting ISO 25010 as the foundation for the research instrument, this study ensures a systematic and internationally recognized approach to assessing the enhanced PUP Faculty Loading and Scheduling System. The chosen quality characteristics guide the instrument development, facilitating a comprehensive evaluation of the software product's performance and user satisfaction.

3.3 Data Gathering Procedure

The researchers employed a comprehensive data gathering approach, primarily utilizing structured interview with key stakeholders, specifically the school

director. These interview, guided by a specific protocol, explored participant past experiences, challenges, and expectations regarding to the previous faculty loading and scheduling system. In-depth interviews with the school director, the primary end-user, and the system developer provided valuable insights into user perspectives and technical aspects. Additionally, purposive sampling was applied to carefully select faculty members involved with the system, ensuring diverse perspectives. This multifaceted data collection strategy, combining structured interviews with strategic ones and purposive sampling, allowed for a thorough understanding of the faculty loading and scheduling system at PUP Taguig, informing the proposed optimized approach.

3.3.1 Sampling Method

The study employs a purposive sampling method to select participants who possess specific expertise and roles within the academic community of the Polytechnic University of the Philippines (PUP). Purposive sampling is a deliberate and non-random method chosen to target individuals with in-depth knowledge and experience relevant to the faculty loading and scheduling system. A total of 20 participants will be selected, including Campus Director, Head of Academic Programs Staff, and Faculty Members who regularly interact with the system. This sample size is chosen to ensure that the selected participants have sufficient experience and insight into the system's functionality and the enhancements being implemented.

3.3.1.1 Rationale for Purposive Sampling

Expertise and Experience:

The primary objective of this research is to gain insights from individuals who hold key roles in the academic administration and faculty at PUP. Academic

administrators, responsible for overseeing and managing faculty resources, and faculty members, who directly engage with the faculty loading and scheduling system, possess critical expertise and experience relevant to our study. Purposive sampling allows us to intentionally target individuals with specific knowledge about the existing system and, if applicable, those who will interact with the enhanced system.

Targeted Perspectives:

By selecting participants purposefully, we aim to capture a diverse range of perspectives within the academic community. This includes individuals from different departments, varying levels of experience, and distinct roles within academic administration and faculty. The targeted perspectives contribute to a more comprehensive understanding of the faculty loading and scheduling system, ensuring that insights are reflective of the diverse nature of the academic community at PUP.

Overall, purposive sampling serves as a strategic approach to ensure that our study benefits from the rich insights of individuals with specific roles and experiences within the academic community at PUP. While acknowledging the intentional selection process, the study aims for diversity within the chosen participant group to enhance the robustness and generalizability of our findings.

3.4 Ethical Considerations

Prior to conducting interviews, the researchers obtained explicit consent from the school director, our primary client, seeking permission to engage in the study. The consent process included a detailed explanation of the study's purpose, procedures, and the confidential nature of the information to be gathered. An agreement was established with the school director, affirming that all details

obtained during the research process would be treated with the utmost confidentiality and would solely remain within the research team. This commitment aligns with the principles of ethical conduct and the requirements of the Republic Act No. 10173, known as the Data Privacy Act. Furthermore, the researchers ensured that the personal information of all participants, particularly faculty members, was handled with strict confidentiality to protect their privacy and adhere to ethical standards.

3.5 System Architecture

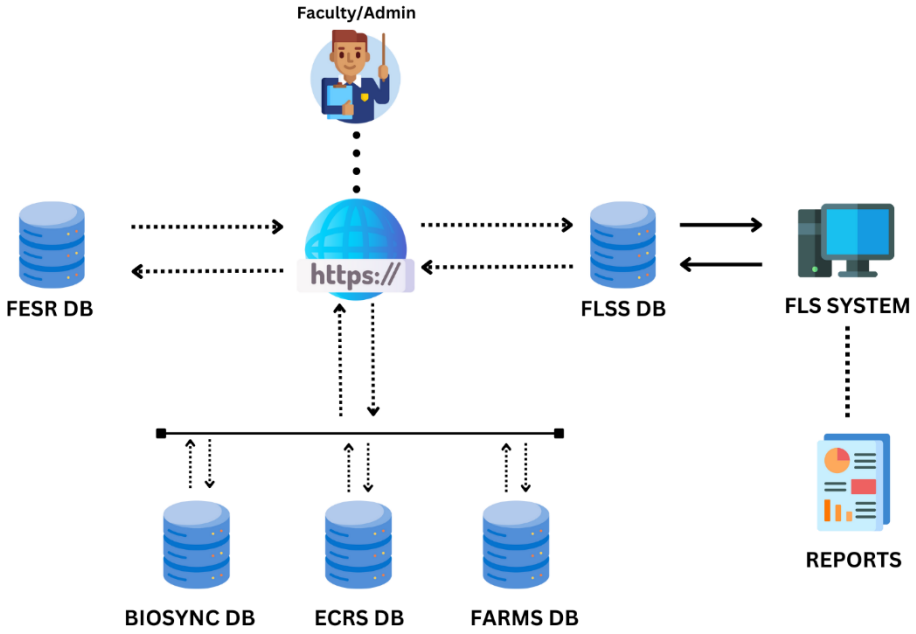


Figure 5. System Architecture

The illustration highlights the interaction among faculty members, administrators, and the Faculty Loading and Scheduling System (FLSS). The administrator manages the system via a computer interface, while the FLSS utilizes data stored in various databases to ensure accurate registration of information. Faculty loading details and schedules for each academic year are gathered from the ECRS, FARMS, and BioSync databases, ensuring the availability of up-to-date

information. Additionally, the PUP Faculty Data Management and Evaluation System with Research Repository (FESR) supplies faculty credentials to the FLSS, facilitating login processes. Faculty members can access reports and key details through the system, streamlining operational efficiency and user experience.

INTEGRATION WITH FESR

The FESR OAuth implementation provides a secure and standardized authentication mechanism for integrating with the Faculty Load Scheduling System (FLSS). Leveraging the OAuth 2.0 protocol, it ensures secure user authentication, authorization, and seamless data exchange. The integration supports real-time faculty data synchronization and prioritizes security and scalability through tokenized communication and robust consent management.

FESR O-AUTH TECHNICAL OVERVIEW

1. **Authentication Flow**:
    - The process begins when a faculty user initiates login via the FLSS Frontend, which redirects them to the FESR login page. The redirection includes the client_id, redirect_uri, and state parameters, ensuring secure and traceable communication.
    - After the user enters valid credentials, the FESR Backend authenticates the user and generates an authorization code, which is returned to the FLSS Backend.
2. **OAuth Token Exchange**:
    - The FLSS Backend exchanges the authorization code for an access token by sending a request to the FESR Backend, including the client_secret for secure verification.
    - Upon successful verification, the FESR Backend returns an access token along with the faculty user's data.

3. **Data Synchronization**:
    - The faculty data retrieved during the token exchange is processed by the FLSS Backend and stored in the system. A session token is generated to enable user access to the FLSS system.
    - To maintain data consistency, subsequent updates to faculty information in the FESR are sent to the FLSS via webhooks. These webhooks are signed with a shared secret to ensure integrity and authenticity.
4. **Security and Scalability**:
    - OAuth 2.0 ensures secure authentication by isolating access token handling and relying on encrypted communication channels.
    - The shared secret used in webhook signatures adds an additional layer of security by validating the origin and content of webhook requests.
    - The modular architecture supports scalability, allowing new features or systems to integrate without significant redesign.

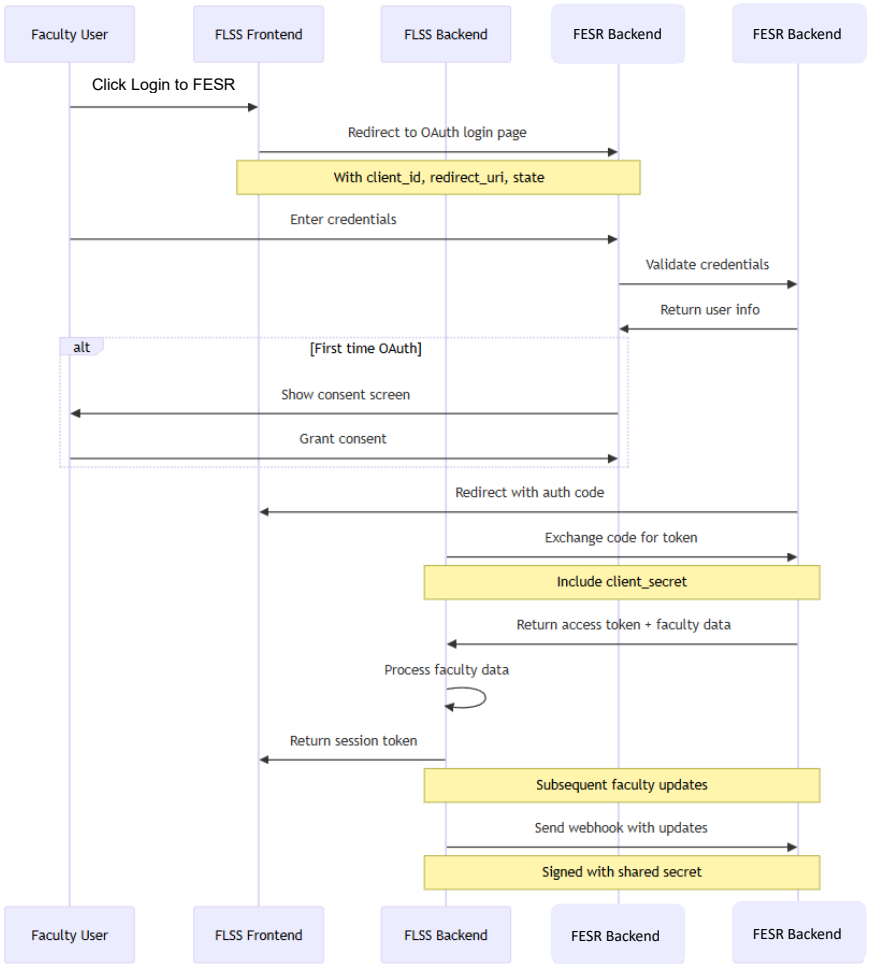


Figure 6. OAuth 2.0 based authentication and data retrieval process between FLSS and FESR

The figure outlines the OAuth 2.0-based authentication and data retrieval process between the Faculty Loading and Scheduling System (FLSS) and the Faculty Data Management and Evaluation System with Research Repository (FESR). Faculty users authenticate through the FESR login page, granting access to FLSS via authorization code exchange. Faculty data is retrieved securely using access tokens, while ongoing updates are handled via signed webhooks to maintain consistency between systems.

OAUTH WITH FESR FEATURES

1. **Authorization Code Grant**: Secure OAuth 2.0 flow for faculty authentication.
2. **CSRF Protection**: State parameter implementation and validation to prevent attacks.
3. **JWT-based Tokens**: Secure information exchange with configurable token expiration.
4. **Secure Client Authentication**: Client ID and secret verification for trusted communication.
5. **Rate Limiting**: Throttles OAuth endpoint requests to prevent abuse.
6. **Consent Management**: User consent handling with clear permission scopes for initial access.

DETAILED BREAKDOWN OF FESR SEQUENCE DIAGRAM

Step 1: User Initiates Login

- The faculty user clicks on the "Login with FESR" button in the FLSS Frontend.
- This action redirects the user to the OAuth login page of the FESR system.

Step 2: OAuth Authorization Request

- The FLSS Backend sends a request to the FESR Frontend with key parameters:
    - client_id: Identifies the FLSS application.
    - redirect_uri: Specifies where the FESR should redirect after authentication.

- state: Ensures request integrity.

Step 3: User Credential Validation

- The user enters their credentials (username and password) on the FESR Frontend.
- The FESR Backend validates the credentials and returns user information to the FESR Frontend.

Step 4: Consent Handling (First-Time OAuth Login)

- If this is the user's first login through OAuth, the system displays a consent screen.
- The user grants permission for FLSS to access specific FESR data.

Step 5: Authorization Code Exchange

- Upon successful authentication, the FESR redirects the user back to the FLSS Backend with an authorization code.
- The FLSS Backend exchanges this authorization code with the FESR Backend for an access token and faculty data.

Step 6: Access Token and Data Retrieval

- The FLSS Backend includes the client_secret (a secure key) in its token exchange request.
- The FESR Backend validates the request and returns an access token along with faculty data.

Step 7: Data Processing and Session Creation

- The FLSS Backend processes the retrieved faculty data.

- A session token is generated and returned to the FLSS Frontend, allowing the user to access the system.

Step 8: Subsequent Faculty Data Updates

- For ongoing synchronization, the FESR Backend sends updates (e.g., faculty changes) to the FLSS Backend via webhooks.
- These updates are signed with a shared secret to ensure security and integrity.

FESR WEBHOOKS TECHNICAL OVERVIEW

Webhooks are a crucial component of the integration between the Faculty Loading and Scheduling System (FLSS) and the Faculty Data Management and Evaluation System with Research Repository (FESR), enabling real-time, event-driven communication. When faculty data is updated in FLSS, the system generates webhook requests that are securely sent to the FESR for synchronization. These webhooks are designed to ensure reliability, security, and data integrity while supporting scalable and efficient data synchronization.

1. **Event-Driven Architecture**
   - Webhooks are triggered whenever faculty-related events (e.g., status changes or updates) occur in the FLS System, eliminating the need for periodic polling.
2. **Secure Transmission**
   - Each webhook request is signed with an HMAC (Hash-based Message Authentication Code) using a shared secret. This signature is included in the headers to authenticate the source and ensure the integrity of the payload during transmission.
3. **Idempotency and Reliability**

- Webhook requests include a unique webhook_id to ensure idempotency, preventing duplicate processing in FESR. Retry mechanisms, such as exponential backoff, are implemented to handle transient failures and ensure reliable delivery.

4. **Response Validations**
   - The FESR verifies the HMAC signature and processes the webhook only if the signature and webhook_id are valid. Responses such as 200 (OK), 401 (Unauthorized), and 409 (Conflict) provide feedback on the processing status.
5. **Scalability**
   - The architecture supports concurrent webhook processing, allowing multiple updates to be synchronized efficiently without impacting system performance.
6. **Error Handling and Recovery**:
   - In cases of server errors (500 status), the FLSS schedules retries with exponential backoff, ensuring reliable delivery without overwhelming the FESR.

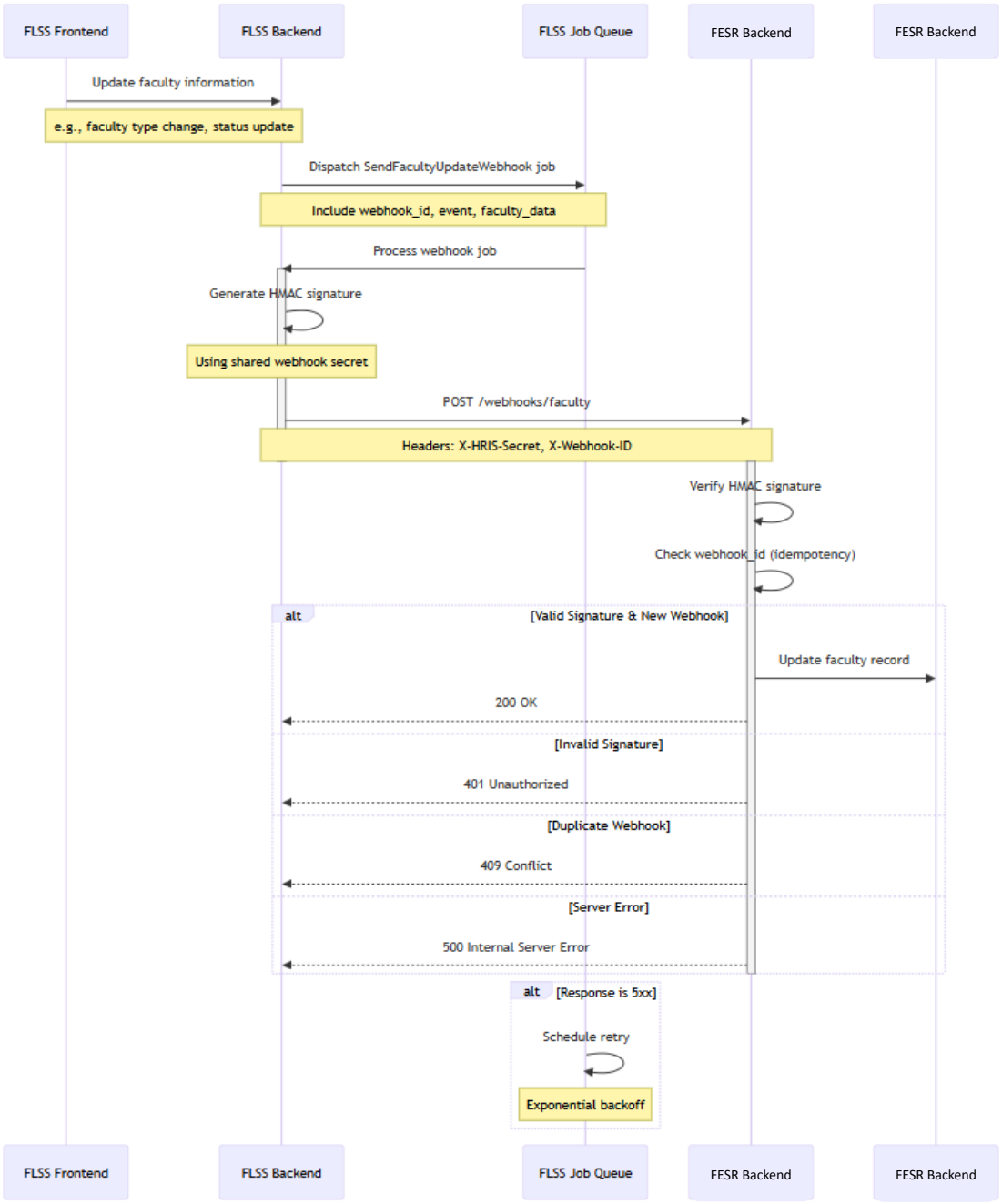


Figure 7. Process for handling faculty data updates via webhooks between FLSS and FESR

The figure illustrates the process for handling faculty data updates via webhooks between FLSS and FESR. Updates are dispatched from FLSS Backend to FESR Backend through a job queue. Security is maintained by generating HMAC signatures and verifying webhook integrity. Retry mechanisms ensure reliability in case of delivery failures.

DETAILED BREAKDOWN OF FESR WEBHOOKS SEQUENCE DIAGRAM

Step 1: Update Faculty Information:

- Updates (e.g., faculty type changes) in FLSS are dispatched as jobs to the FLSS Job Queue.

Step 2: Webhook Job Processing:

- FLSS Backend generates a webhook payload including webhook_id, event type, and faculty data.

Step 3: Security Signature:

- A secure HMAC signature is generated using a shared webhook secret for secure transmission.

Step 4: Send Webhook to FESR:

- FLSS sends the webhook to FESR Backend with headers (X-FESR-Secret, X-Webhook-ID) for signature verification.

Step 5: Verify and Update:

- FESR Backend verifies the HMAC signature and checks webhook ID for idempotency. If valid, it updates the faculty record in the FESR database.

Response Handling:

- **200 OK**: Update successful.

- **401 Unauthorized**: Invalid signature or unauthorized request.
- **409 Conflict**: Duplicate webhook detected.
- **500 Server Error**: Retry scheduled using exponential backoff.

Step7: Retry Mechanism:

- Failed webhooks (500 error) are retried automatically to ensure successful delivery and processing.

INTEGRATION WITH ECRS

The integration between the Faculty Load Scheduling System (FLSS) and the E- Class Record System (ECRS) leverages a RESTful API to securely and efficiently deliver faculty schedule data. This integration ensures that schedules are always up-to-date in ECRS through real-time data retrieval and structured JSON responses. Robust security mechanisms, efficient data queries, and clear response structures enable scalable and reliable data synchronization.

ECRS API CALL TECHNICAL OVERVIEW

1. **Real-Time Data Delivery**
    - Faculty schedules are retrieved from FLSS whenever ECRS requests them, ensuring the data remains current without the need for manual synchronization.
2. **Secure Authentication**
    - HMAC (Hash-based Message Authentication Code) middleware validates each request using a shared secret. This prevents unauthorized access and ensures that the source of the request is authenticated.
3. **Optimized Data Retrieval**

- Queries are designed to fetch active semester schedules with related faculty, course, and room details using efficient joins and subqueries, minimizing database load.

4. **Structured Data Response**
   - Data is grouped and formatted into a JSON response, including academic year details, semester information, and faculty schedules with all relevant details (e.g., day, time, room, course).
5. **Error Handling**
   - Clear and specific error responses (e.g., 404 for missing data, 401 for invalid HMAC, and 500 for internal errors) guide ECRS in handling potential issues effectively.
6. **Scalability**
   - The architecture supports high volumes of requests from ECRS, ensuring consistent performance through asynchronous request handling and efficient database operations.

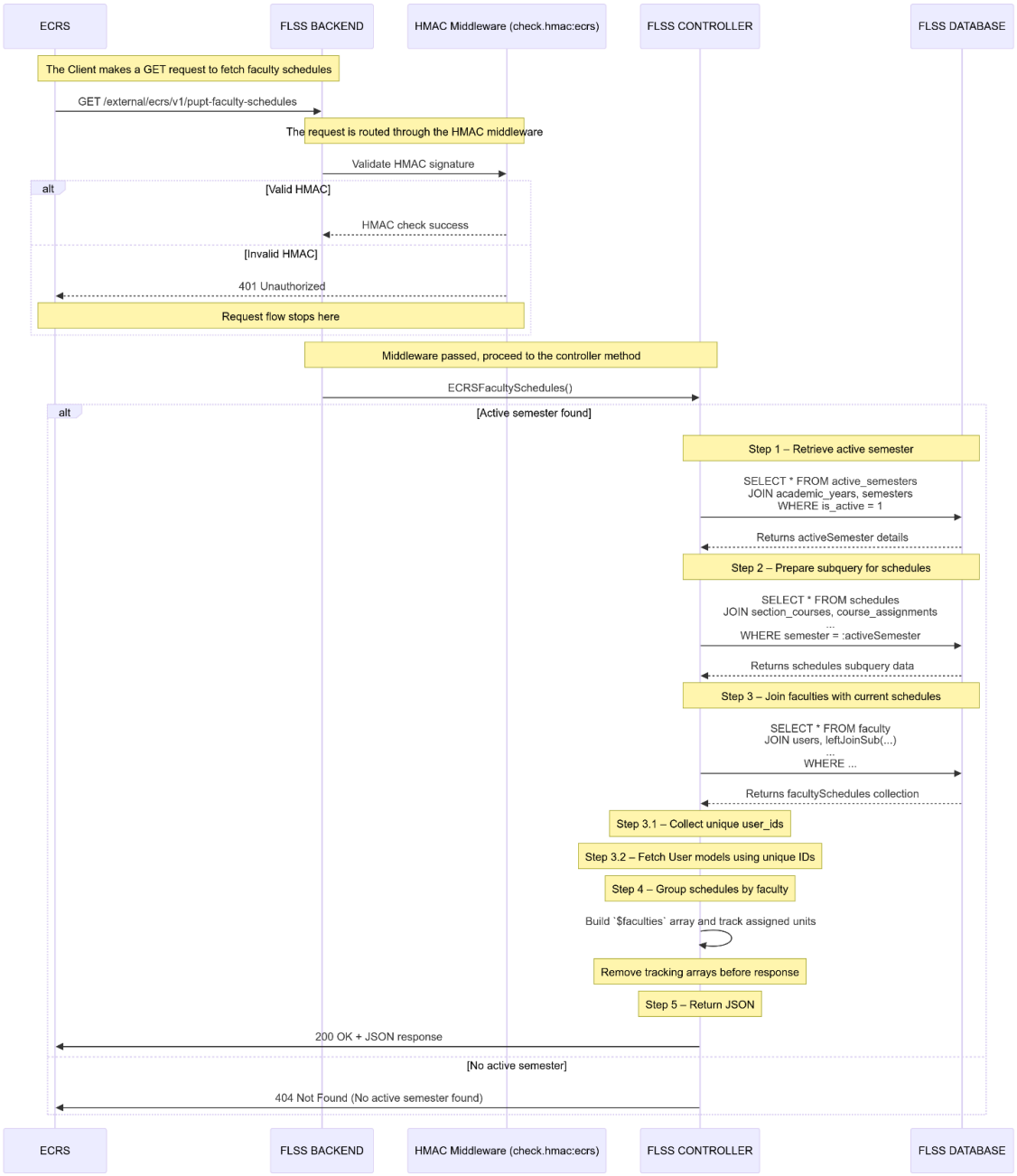


Figure 8. Secure process for handling faculty schedule requests between the ECRS client and the FLSS backend.

The figure outlines the secure process for handling faculty schedule requests between the ECRS client and the FLSS backend. When the client initiates a GET request, the HMAC middleware validates the request for security. Once validated, the controller processes the request by retrieving the active semester, schedules, and faculty data from the database, structuring the information by faculty. The system then responds with a JSON payload containing the requested schedules or a 404 error if no active semester is found. Robust error handling ensures the process gracefully manages invalid requests or missing data, maintaining reliability and accuracy throughout.

DETAILED BREAKDOWN OF ECRS WEBHOOKS SEQUENCE DIAGRAM

Step 1: Client Request

- The ECRS client initiates a **GET** request to the FLSS backend endpoint /external/ecrs/v1/pupt-faculty-schedules to retrieve faculty schedule data for the current active semester.

Step 2: Middleware Validation

- The request is passed through the **HMAC Middleware** (check.hmac:ecrs) to validate its integrity and authenticity using a secure HMAC signature.
- If the HMAC signature is invalid, the middleware immediately returns a **401 Unauthorized** response.

Step 3: Retrieve Active Semester

- Upon successful middleware validation, the controller retrieves the current active semester details (e.g., academic year and semester) from the database.
- If no active semester is found, the system returns a **404 Not Found** response.

Step 4: Query Faculty Schedules

- Using the active semester information, the controller prepares a subquery to fetch schedules for faculty during the current academic term.
- The query retrieves related data such as faculty details, course assignments, rooms, and program information.

Step 5: Data Structuring

- The controller processes the retrieved data, grouping schedules by faculty.
- Each faculty's schedule includes details such as day, time, room, program, and course-specific information (e.g., course title, units, lecture/lab hours).

Step 6: JSON Response Generation

- The structured data is formatted into a JSON response. If the active semester and schedules are available, the system responds with a **200 OK** status and the faculty schedule data.
- If no schedules or active semester data exist, a **404 Not Found** response is returned.

Step 7: Error Handling

- **401 Unauthorized**: If the HMAC signature is invalid, the middleware prevents further processing and returns an error.
- **404 Not Found**: If no active semester or schedule data is found, a "not found" response is provided.
- **500 Server Error**: If any unexpected issue occurs, an appropriate error message is logged for troubleshooting.

Step 8: Retry Mechanism (Optional)

- If there are failures during database access or JSON response generation, retries can be implemented (if necessary) to ensure the API call completes successfully.

INTEGRATION WITH FARMS

The integration between the Faculty Academic Requirements Management System (FARMS) and the Faculty Load Scheduling System (FLSS) ensures secure and efficient delivery of course schedule data through a RESTful API. This integration leverages real-time data retrieval, robust security measures, and structured JSON responses to maintain data accuracy and synchronization. Scalable architecture and detailed error handling make this integration reliable and effective.

FARMS API CALL TECHNICAL OVERVIEW

1. **Real-Time Data Delivery**
    - Course schedules are retrieved dynamically from the FLSS backend whenever FARMS requests them. This eliminates the need for manual synchronization, ensuring schedules are always accurate and up-to-date.
2. **Secure Authentication**
    - HMAC (Hash-based Message Authentication Code) middleware authenticates requests using a shared secret, preventing unauthorized access and verifying request integrity.
3. **Optimized Data Retrieval**
    - Efficient queries fetch active semester schedules, joining faculty, courses, and additional details. This reduces database load while ensuring accurate data delivery.
4. **Structured Data Response**
    - Course schedule data is formatted into a JSON response, grouped by faculty. Each schedule includes details such as academic year, semester, day, time, room, and course-specific information.

5. **Error Handling**
    - Specific error responses guide FARMS in managing potential issues:
        - 404: No active semester or schedules found.
        - 401: Invalid HMAC signature.
        - 500: Internal server errors.
6. **Scalability**
    - The system handles high request volumes efficiently using asynchronous request processing and optimized database operations.

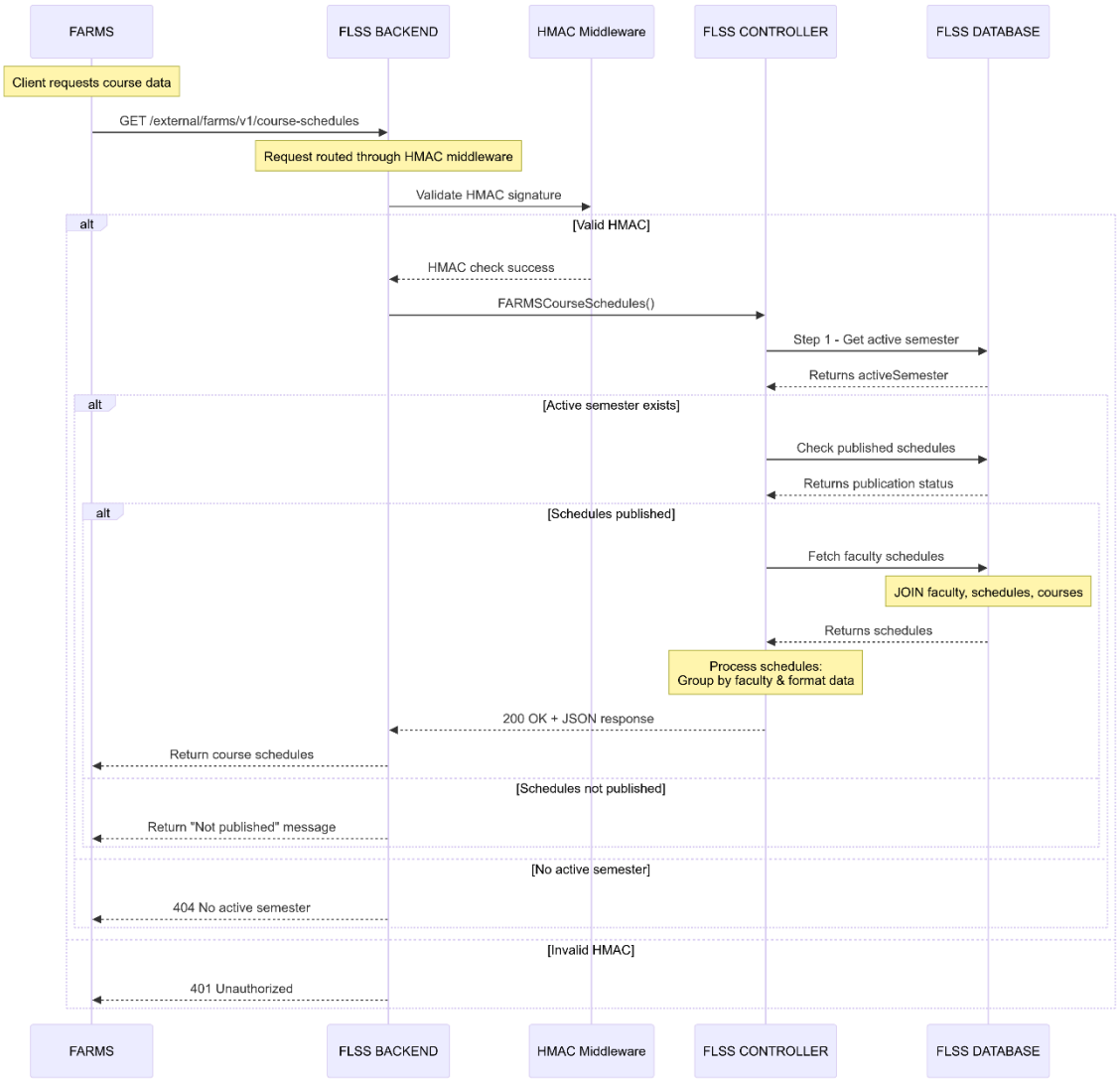


Figure 9. Secure process for handling course schedule requests between FARMS and the FLSS backend

The figure illustrates the secure process of handling course schedule requests. FARMS initiates a GET request, which is validated through HMAC middleware. Once verified, the request is processed by retrieving active semester data and querying faculty schedules from the database. The system structures the data into a JSON payload grouped by faculty or returns specific error responses for missing or invalid data.

DETAILED BREAKDOWN OF FARMS API SEQUENCE DIAGRAM

Step 1: Client Request

- FARMS initiates a GET request to the endpoint /external/farms/v1/course-schedules to retrieve schedule data for the current semester.

Step 2: Middleware Validation

- The HMAC middleware validates the request using a secure HMAC signature.
    - If invalid, the middleware immediately returns a 401 Unauthorized response.

Step 3: Retrieve Active Semester

- Upon successful validation, the FLSS controller retrieves the active semester details (academic year and semester).
    - If no active semester exists, a 404 Not Found response is returned.

Step 4: Query Faculty Schedules

- The controller queries the database for faculty schedules, courses, and related details for the active semester.

Step 5: Data Structuring

- The system processes and groups schedules by faculty. Each faculty schedule includes:
    - Day, time, room, course details, and other relevant information.

Step 6: JSON Response Generation

- A JSON payload is returned to FARMS, containing:
    - 200 OK: Schedules for the active semester.
    - 404 Not Found: No schedules available.

Step 7: Error Handling

- The system handles potential errors:
  - 401 Unauthorized: Invalid HMAC signature.
  - 404 Not Found: Missing active semester or schedule data.
  - 500 Server Error: Internal processing issues logged for

3.5.1 System Flowchart

Figure 11. User Login Module

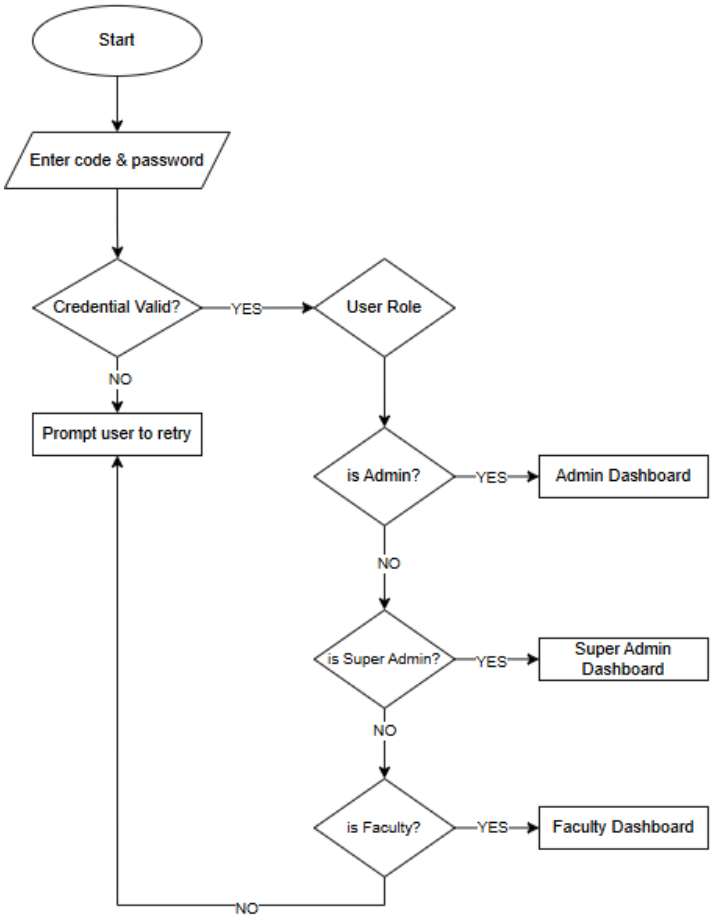


Figure 11 depicts the process for logging into the system. The login process begins when a user inputs their unique code and password. The system validates these credentials, ensuring they match the stored information. If the credentials are invalid, the system prompts the user to retry. Upon successful validation, the system checks the user's role. Based on the user's role, they are directed to the appropriate dashboard. An admin user is given access to the Admin Dashboard, while a super admin is redirected to the Super Admin Dashboard. Faculty members will be directed to faculty dashboard. If the user does not fit any of these categories, the login attempt is denied.

Figure 12. Submit Preferences in the Faculty Module

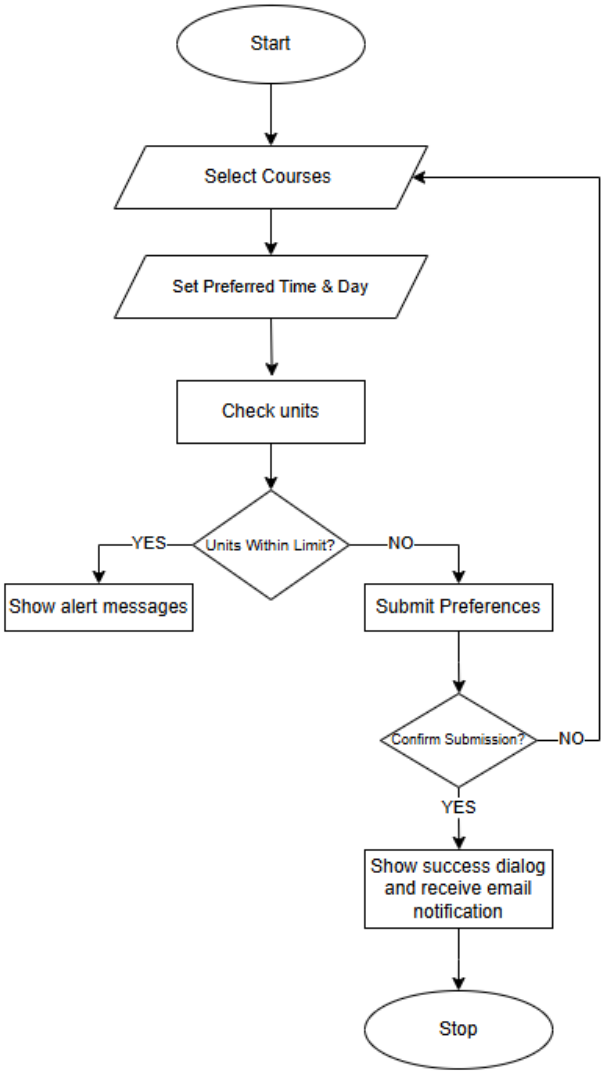


Figure 12 illustrates the process of selecting and submitting course preferences. The user starts by choosing courses and setting preferred times and days. The system checks if the total number of units is within the allowed limit. If the units exceed the limit, an alert is shown; otherwise, the user submits their preferences. After confirming submission, a success message is displayed, and an email notification is sent, completing the process.

Figure 13. Scheduling and Loading in the Faculty Module

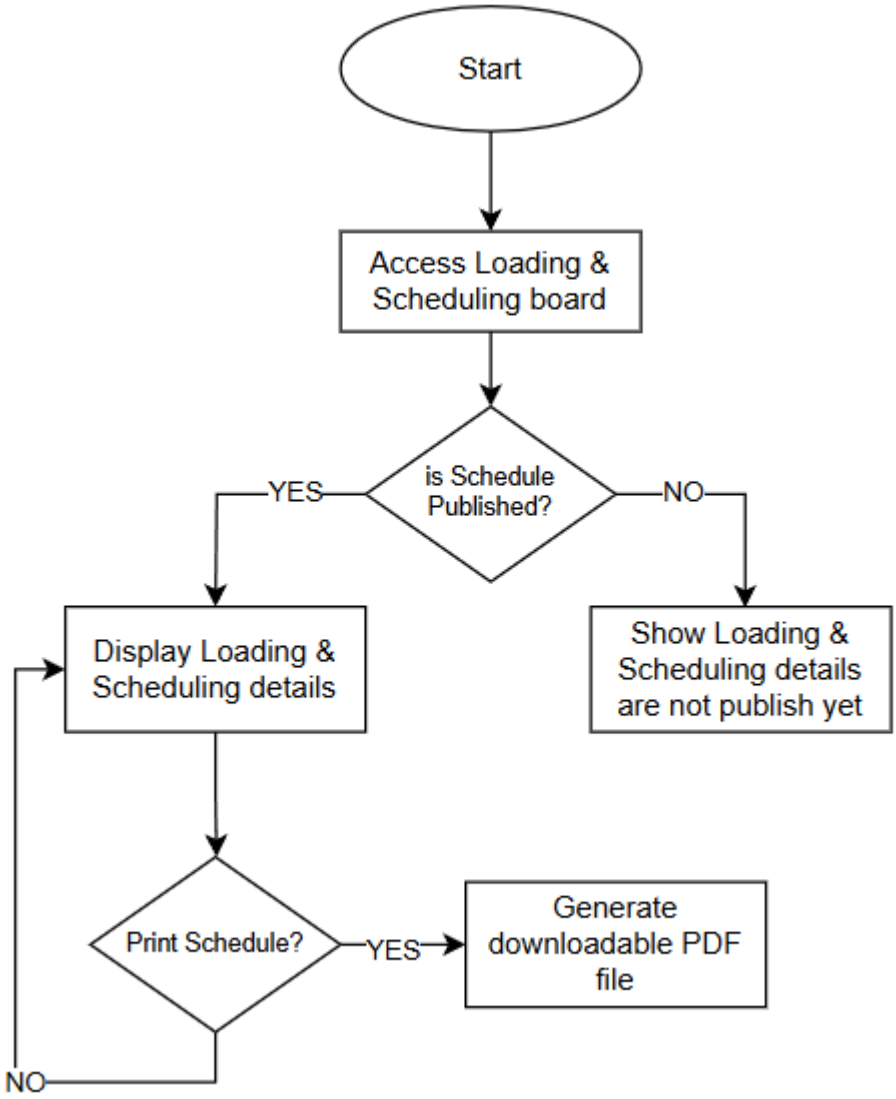


Figure 13 illustrates the process for accessing, viewing, and printing loading and scheduling details. The user begins by accessing the loading and scheduling board. The system checks if the schedule has been published. If it is published, the loading and scheduling details are displayed. If not, the system informs the user that the details have not yet been published. After displaying the details, the user is given the option to print the schedule. If the user chooses to print, the system generates a downloadable PDF file. If printing is not selected, the process ends.

Figure 14. Viewing and Generating Faculty Information in the Super Admin Module

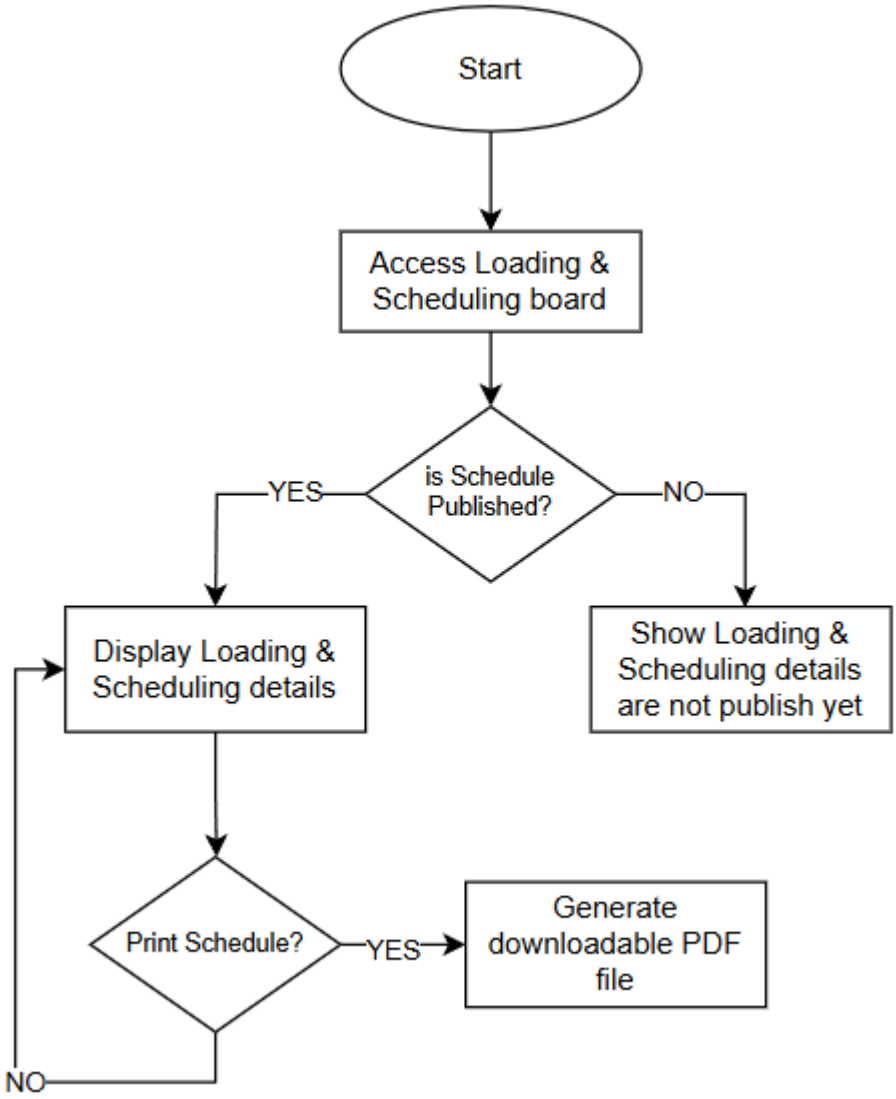


Figure 14 illustrates the process of viewing and generating a PDF for faculty details. The process starts when the user accesses the faculty management board. If the user chooses to generate a PDF, the system displays a dialog box containing the faculty details for review. If the user opts to print the PDF, the system generates a downloadable PDF file. If the user decides not to print, the process loops back to the review stage or stops without generating the file. The process concludes once the PDF is generated or the decision to not print is finalized.

Figure 15. Adding Faculty Information in the Super Admin Module

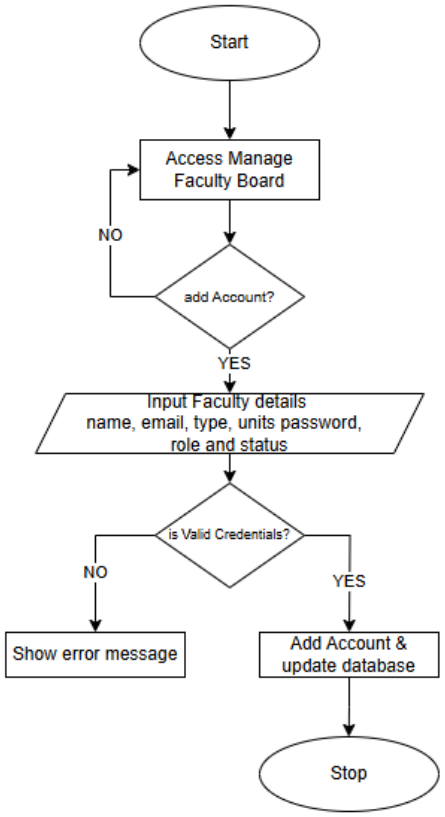


Figure 15 illustrates the process for adding a faculty account to the system. The user begins by accessing the faculty management board. If the user chooses to add an account, they input the faculty details, including name, email, type, units, password, role, and status. The system then checks the validity of the entered credentials. If the credentials are invalid, an error message is shown. If the credentials are valid, the account is added to the database, and the system is updated. The process then concludes.

Figure 16. Editing Faculty Account in the Super Admin Module

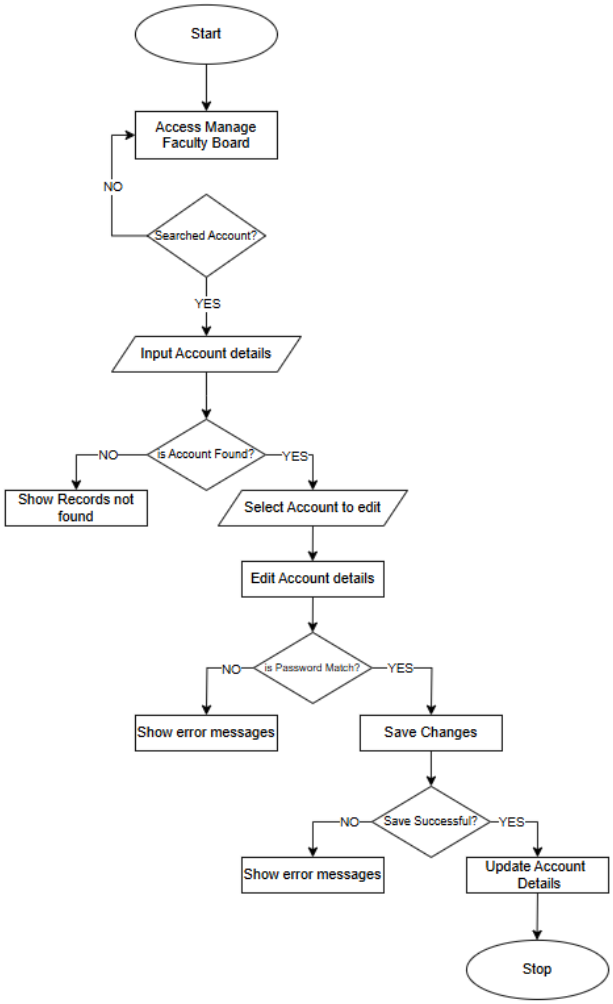


Figure 16 illustrates the process for managing faculty accounts. The user begins by accessing the faculty management board. If an account search is initiated, the user inputs the account details. If the account is found, the user selects it for editing. If the account is not found, the system shows an error message. Once the account is selected, the user proceeds to edit the account details. The system verifies if the password matches. If it does not, an error message is shown. If the password matches, changes are saved, and the system checks if the save was successful. If successful, the account details are updated, otherwise, an error message is displayed, and the process concludes.

Figure 17. Deleting Faculty Account in the Super Admin Module

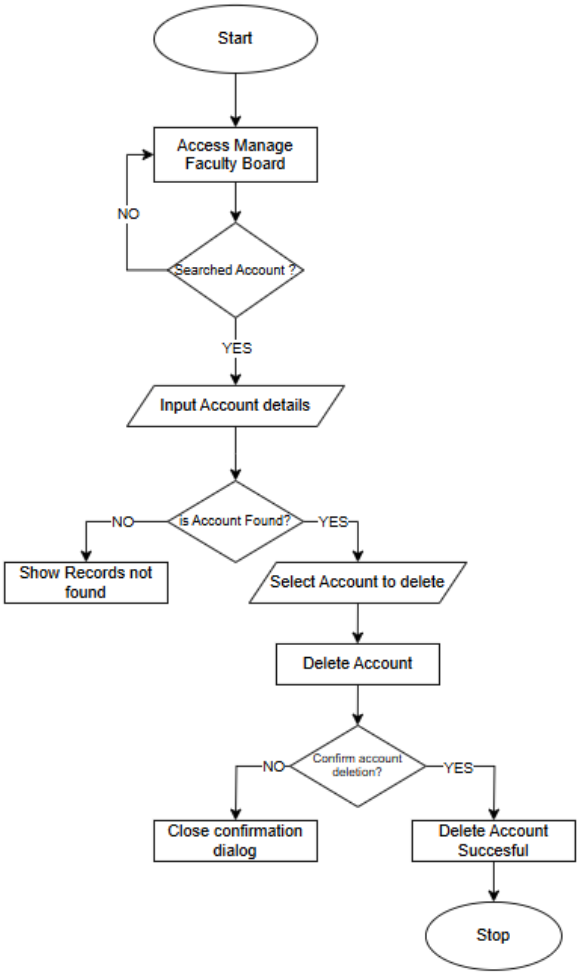


Figure 17 illustrates the process for deleting a faculty account. The user starts by accessing the faculty management board and searches for the account they want to delete. If no account is found, the system shows a "Records not found" message. If the account is found, the user inputs the account details and selects it for deletion. The system then checks if the faculty member is currently active in the current academic year (A.Y.), with any teaching load or schedule. If the faculty member is inactive, the account is deleted. However, if the faculty is active, the system displays a message indicating that the account cannot be deleted while active. The process then concludes.

Figure 18. Viewing and Generating Room Report in the Super Admin Module

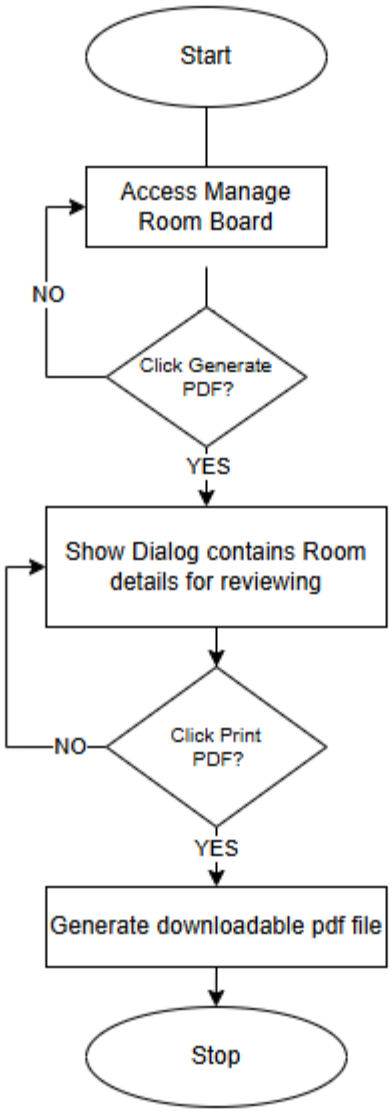


Figure 18 illustrates the process for generating and reviewing room details in PDF format. The user begins by accessing the room management board. If the user opts to generate a PDF, the system displays a dialog containing the room details for review. The user can then decide whether to print the PDF. If the print option is selected, the system generates a downloadable PDF file. If not, the process returns to the review stage. Once the PDF is generated, the process ends.

Figure 19. Adding Room in the Super Admin Module

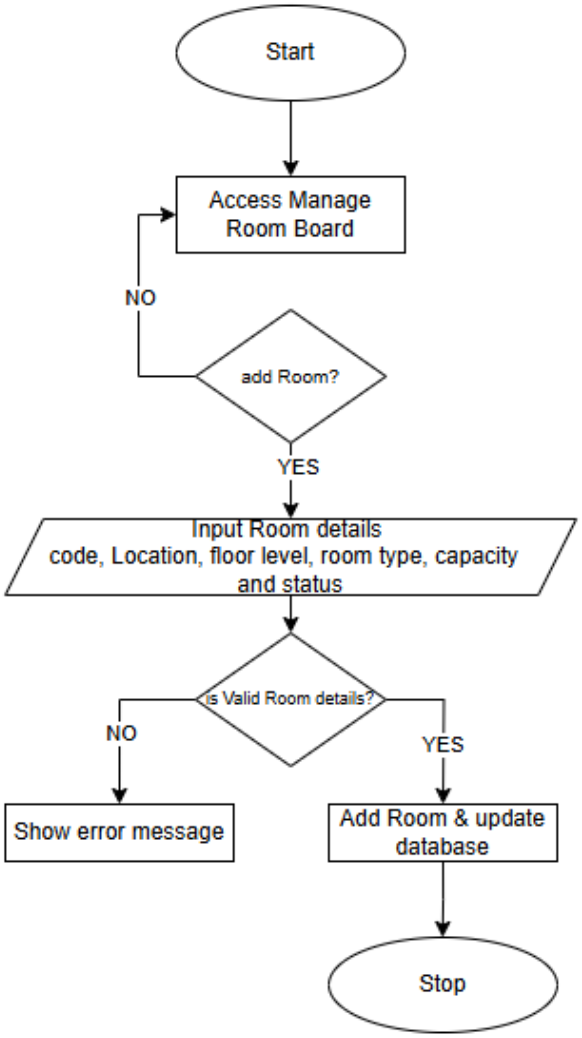


Figure 19 illustrates the process for adding a room to the system. The user begins by accessing the room management board. If the user decides to add a room, they input the room details, including the code, location, floor level, room type, capacity, and status. The system then checks if the room details are valid. If the details are invalid, an error message is shown. If the details are valid, the room is added to the system, and the database is updated. The process concludes once the room is successfully added or if an error is encountered.

Figure 20. Editing Room details in the Super Admin Module

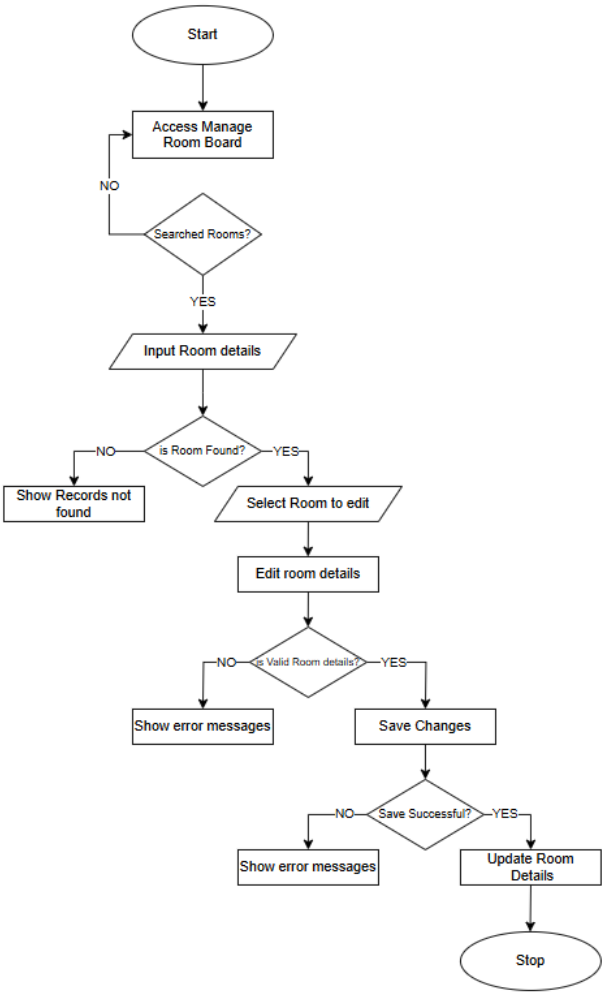


Figure 20 illustrates the process for searching and editing room details. The process starts when the user accesses the room management board. If the user searches for a room, they input the room details. If the room is found, the user selects the room for editing; otherwise, the system displays a "Records not found" message. Once the room is selected, the user can proceed to edit the room details. The system then validates the details entered. If the details are invalid, an error message is shown. If valid, the changes are saved, and the system checks if the save was successful. If the save is unsuccessful, another error message is shown. If successful, the room details are updated, and the process concludes.

Figure 21. Deleting Room details in the Super Admin Module

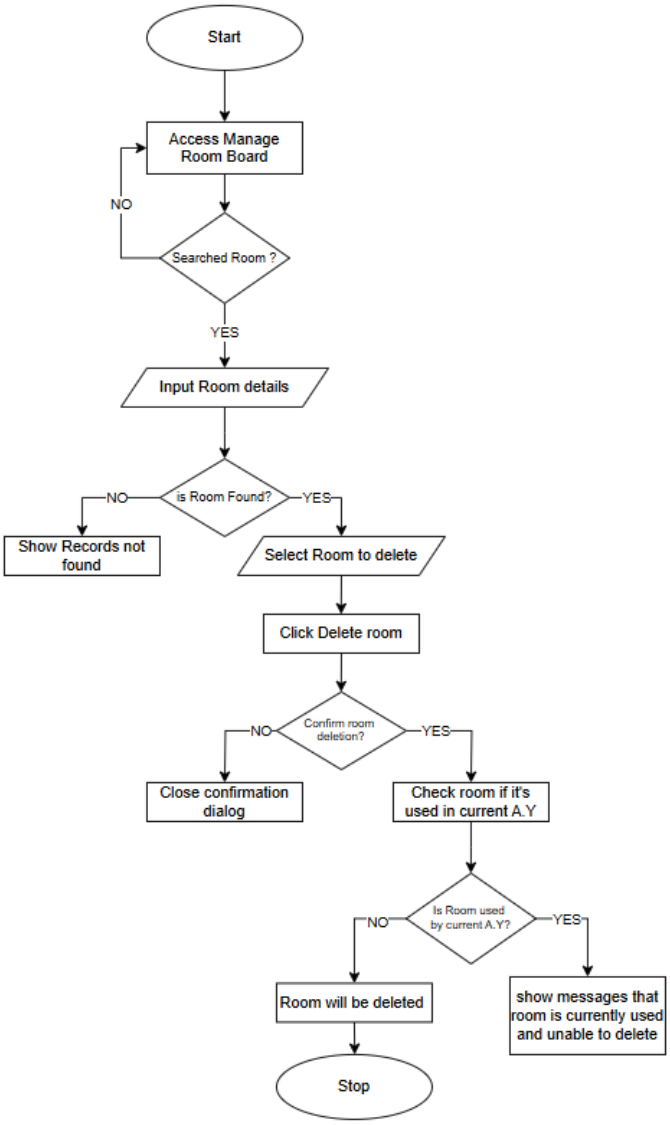


Figure 21 shows the process for deleting a room. The user searches for a room in the room management board. If the room is found, they select it and click delete. The system checks if the room is in use during the current academic year. If it's in use, a message appears saying the room cannot be deleted. If not, the room is deleted. If the room is not found or the deletion is canceled, the process stops.

Figure 22. Viewing and Generating Program Report in the Super Admin Module

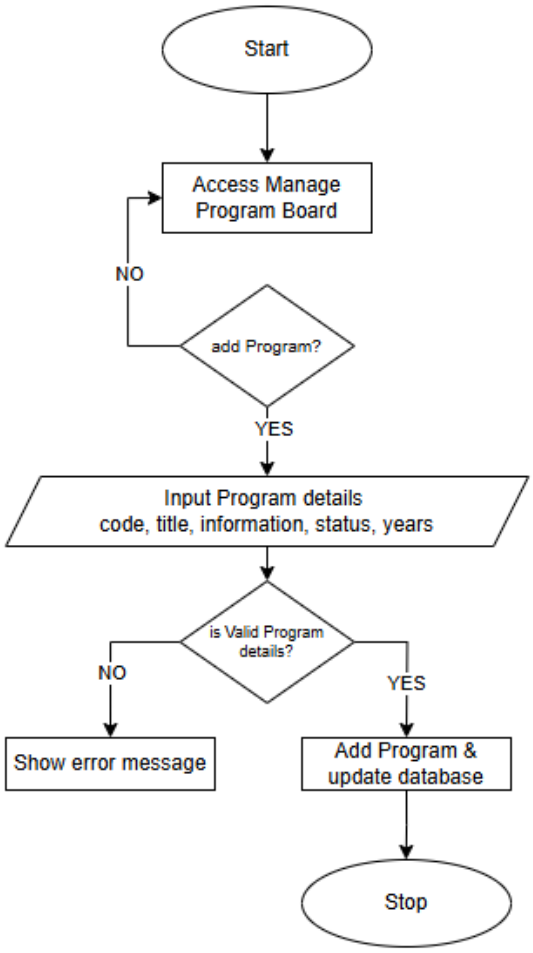


Figure 22 illustrates the process for adding a program to the system. The user begins by accessing the program management board. If they choose to add a program, they input the program details, including the code, title, information, status, and years. The system checks if the program details are valid. If the details are invalid, an error message is shown. If the details are valid, the program is added, and the database is updated. The process then concludes.

Figure 23. Adding Program in the Super Admin Module

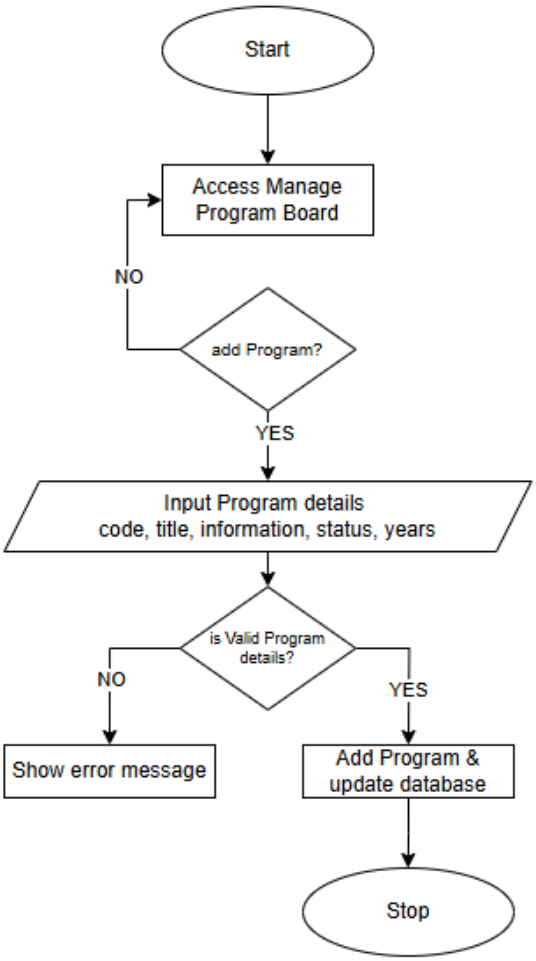


Figure 23 illustrates the process of adding a new program to the system. The user accesses the program management board and chooses whether to add a new program. If they proceed, they input the program details such as the code, title, information, status, and duration (years). The system then validates the details entered. If the details are invalid, an error message is displayed. If the details are valid, the program is added to the system, and the database is updated, concluding the process.

Figure 24. Editing Program Details in the Super Admin Module

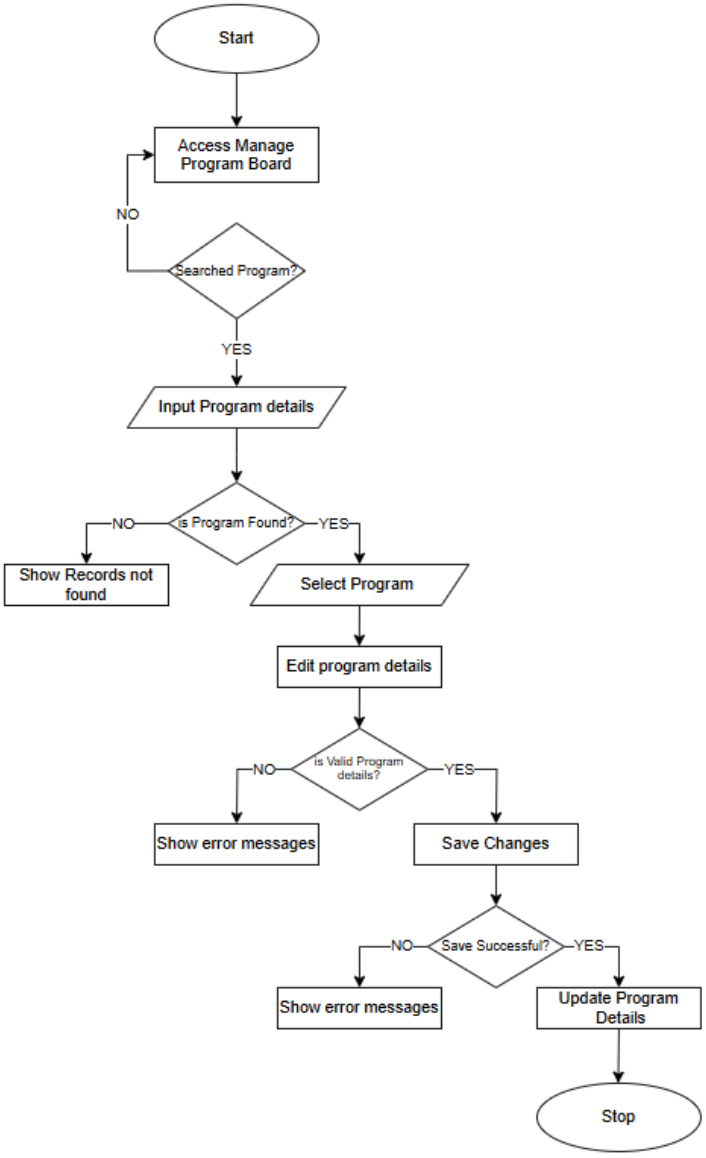


Figure 24 shows the process of searching and editing a program. The user accesses the program management board and searches for a program. If found, they select and edit it; if not, a "Records not found" message is shown. After editing, the system checks if the changes are valid. If valid, the changes are saved, and the program details are updated. If not, an error message is displayed. The process then ends.

Figure 25. Deleting Program in the Super Admin Module

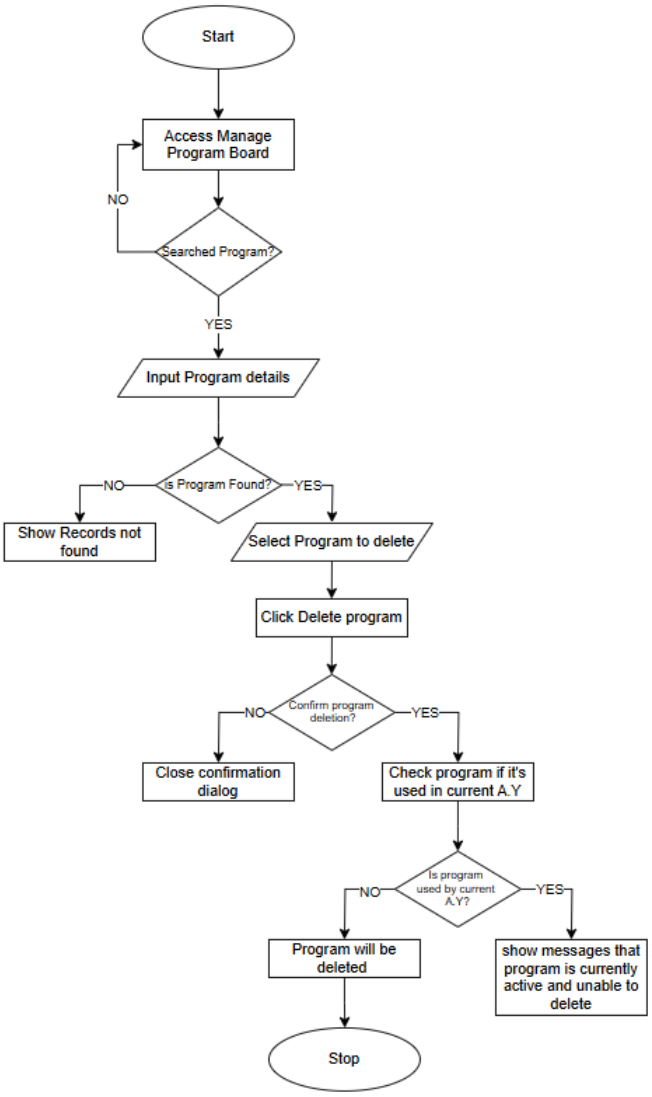


Figure 25 shows the process for deleting a program from the system. The user starts by accessing the program management board and searching for the program. If the program is found, it is selected for deletion; if not, a "Records not found" message appears. After selecting the program, the user confirms the deletion. The system then checks if the program is currently active in the current academic year. If the program is active, a message is shown indicating it cannot be deleted. If the program is inactive, it is deleted, and the process concludes.

Figure 26. Viewing and Generating Curriculum Details in the Super Admin Module

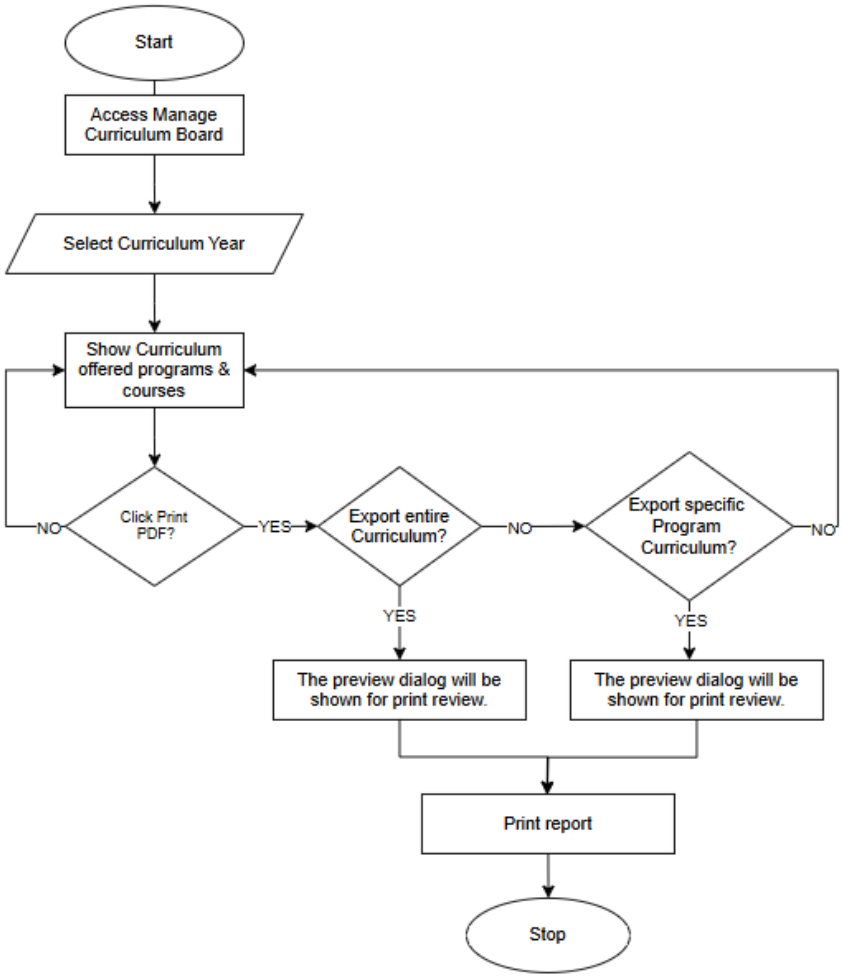


Figure 26 shows the process for exporting the curriculum program report. The process starts when the user accesses the "Manage Curriculum Board" and selects the desired curriculum year. Once the year is selected, the system displays the offered programs and courses. The user is then prompted with the option to print the curriculum as a PDF. If the user selects to print, the system will check whether to export the entire curriculum or just a specific program's curriculum. Depending on the selection, a preview dialog is shown for print review. After the review, the report is printed, and the process concludes. If the user chooses not to print, the process terminates without further action.

Figure 27. Adding a Curriculum in the Super Admin Module

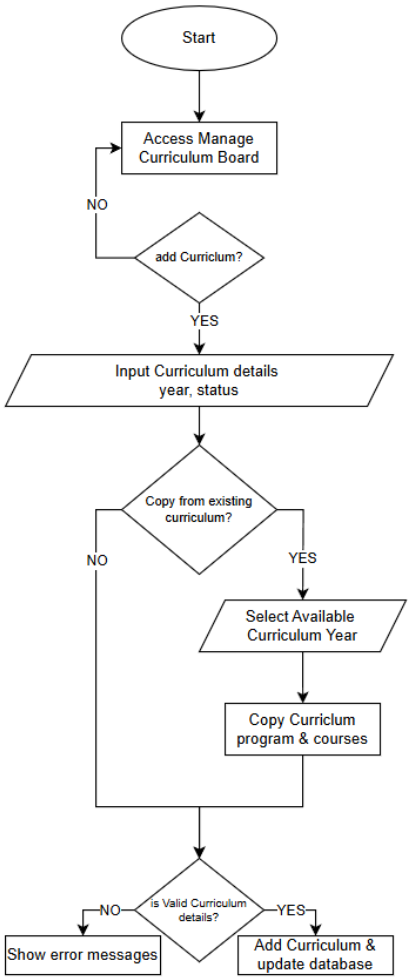


Figure 27 shows the process for adding a curriculum in the system. The process begins when the user accesses the "Manage Curriculum Board" and decides whether to add a curriculum. If the user chooses to proceed, they are required to input curriculum details, such as the year and status. The system then asks if the user wants to copy the curriculum from an existing one. If the user selects "Yes," they must choose the available curriculum year to copy the programs and courses. After copying, or if no existing curriculum is copied, the system checks if the curriculum details are valid. If the details are valid, the curriculum is added to the system, and the database is updated. If the details are invalid, the user is shown error messages, and the process concludes without adding the curriculum.

Figure 28. Editing a Curriculum in the Super Admin Module

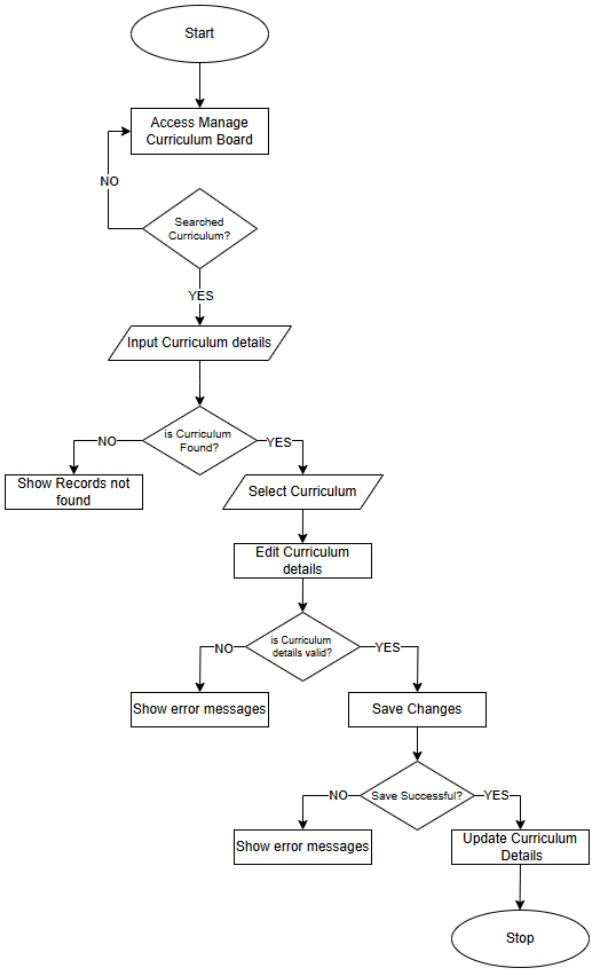


Figure 28 shows the process for editing a curriculum in the system. The process starts when the user accesses the "Manage Curriculum Board" and decides whether to search for a curriculum. If the user chooses to search, they are prompted to input the curriculum details. If the curriculum is found, the user selects it for editing; otherwise, a "Records not found" message is displayed. Once the curriculum is selected, the user can edit the curriculum details. After making changes, the system verifies if the entered details are valid. If the details are not valid, error messages are shown, and the process cannot proceed. If the details are valid, the user saves the changes. The system then checks if the save was successful. If not, an error message is displayed. If the save is successful, the curriculum details are updated, and the process concludes.

Figure 29. Deleting a Curriculum in the Super Admin Module

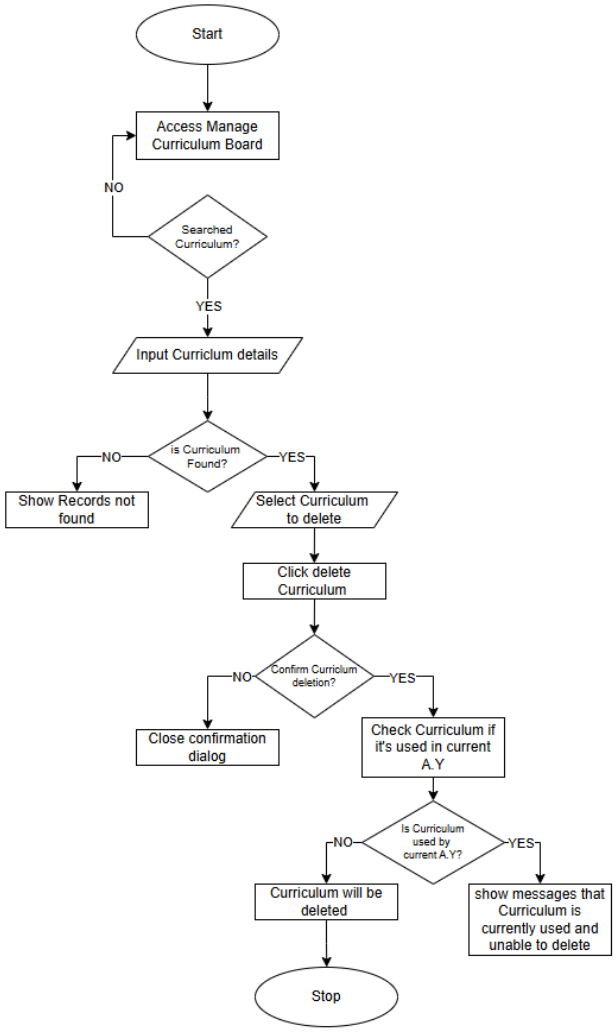


Figure 29 shows the process for deleting a curriculum from the system. The process starts with the user accessing the "Manage Curriculum Board" and deciding whether to search for a curriculum. If the user chooses to search, they are prompted to input the curriculum details. If the curriculum is found, the user selects it for deletion; otherwise, a "Records not found" message is displayed. After selecting the curriculum, the user clicks to delete it, and a confirmation dialog appears. If the user cancels the confirmation, the process stops. If the deletion is confirmed, the system checks if the curriculum is currently in use in the current academic year. If the curriculum is being used, a message is shown indicating that it cannot be deleted. If it is not in use, the curriculum is deleted, and the process ends.

Figure 30. Assigning a Course to a Curriculum in the Super Admin Module

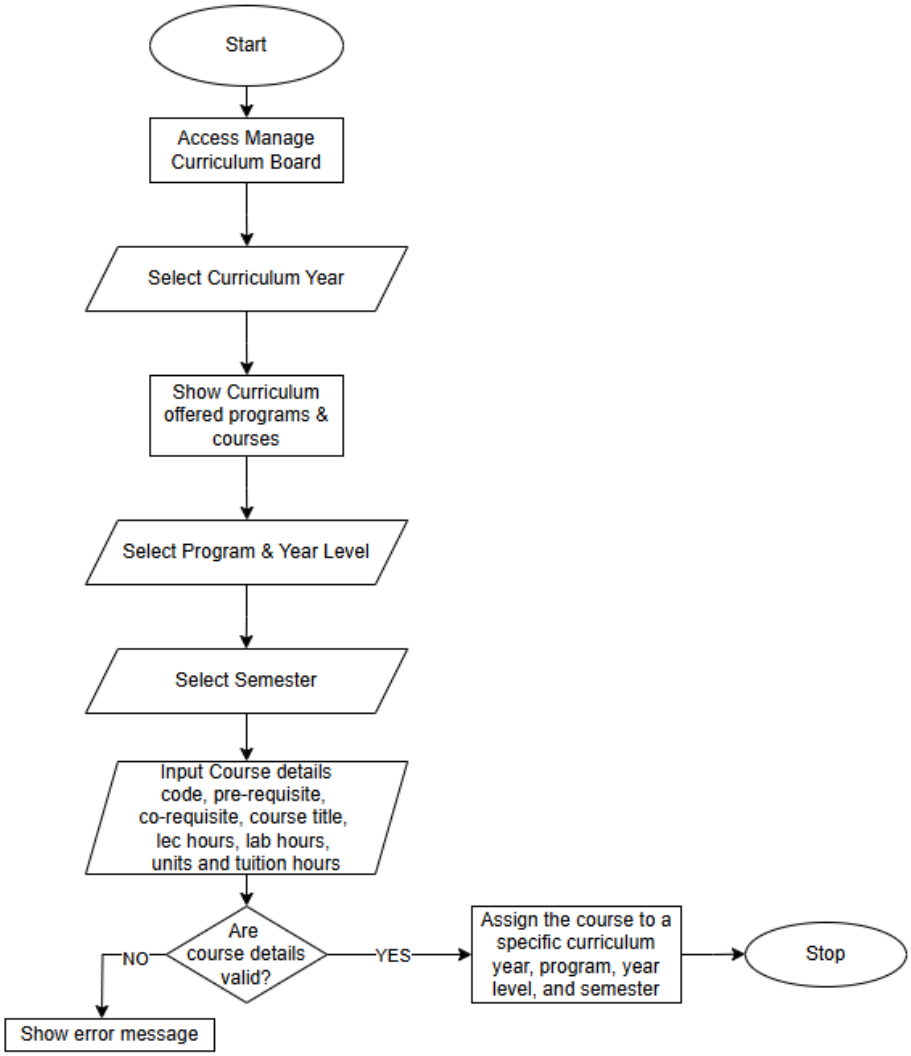


Figure 30 shows the process for assigning a course to a curriculum. The user starts by accessing the "Manage Curriculum Board" and selecting the curriculum year. After viewing the offered programs and courses, the user selects the program, year level, and semester. They then input course details, such as the course code, prerequisites, hours, and units. The system checks if the details are valid. If invalid, an error message is shown. If valid, the course is assigned to the selected curriculum, program, and semester, completing the process.

Figure 31. Editing Course Details in the Super Admin Module

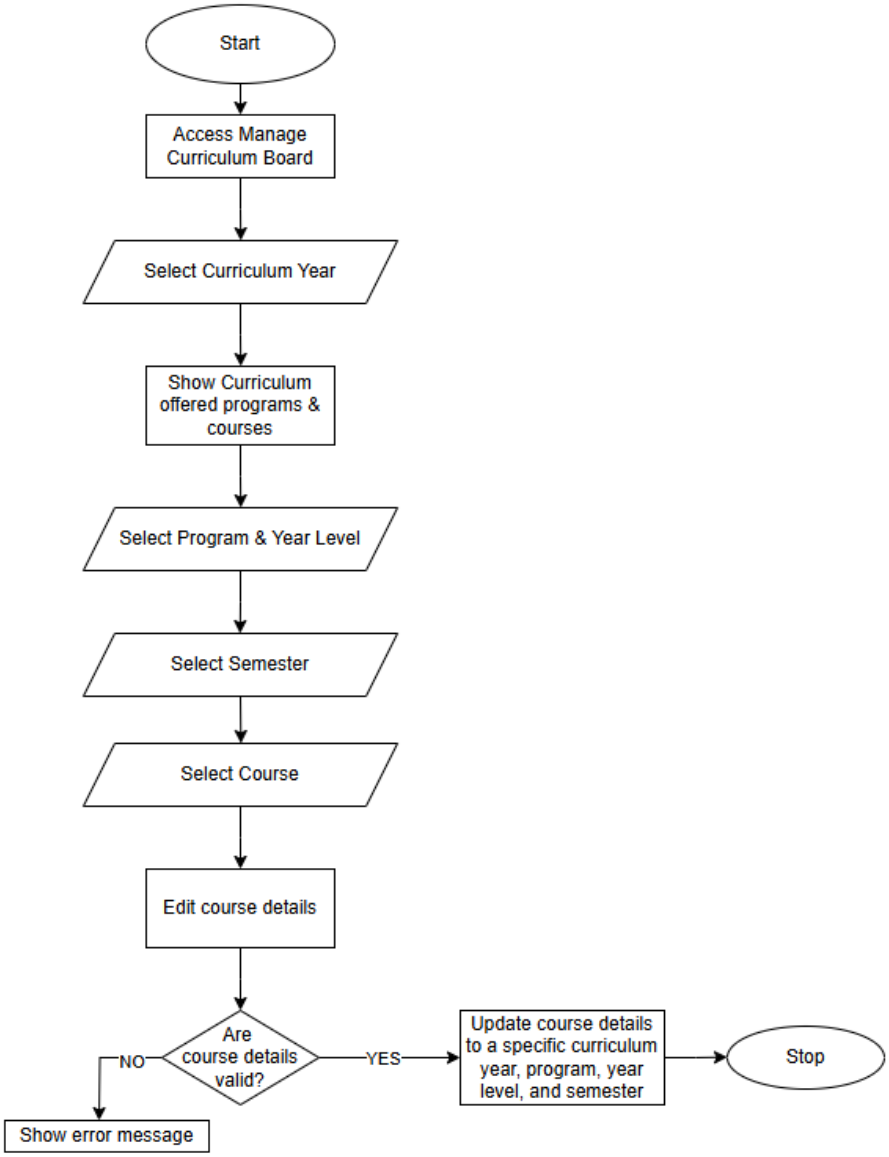


Figure 31 shows the process for editing course details in the system. The user begins by accessing the "Manage Curriculum Board," selecting the curriculum year, and viewing the offered programs and courses. They then choose the program, year level, semester, and specific course. After selecting the course, the user edits the course details. The system checks if the details are valid. If invalid, an error message is shown. If valid, the course details are updated, and the process ends.

Figure 32. Deleting a Course in the Super Admin Module

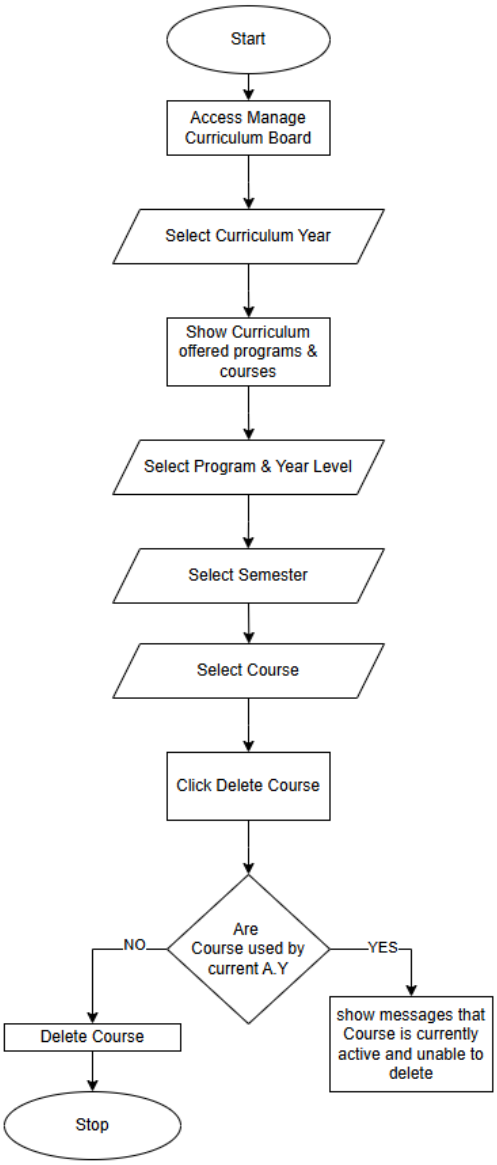


Figure 32 shows the process for deleting a course from the system. The user begins by accessing the "Manage Curriculum Board," selecting the curriculum year, and viewing the offered programs and courses. The user then selects the program, year level, semester, and the specific course. After selecting the course, the user clicks to delete it. The system checks if the course is currently being used in the current academic year. If the course is in use, a message is displayed indicating it cannot be deleted. If the course is not in use, it is deleted, and the process ends.

Figure 33. Managing Programs in Specific Curriculum Year in the Super Admin Module

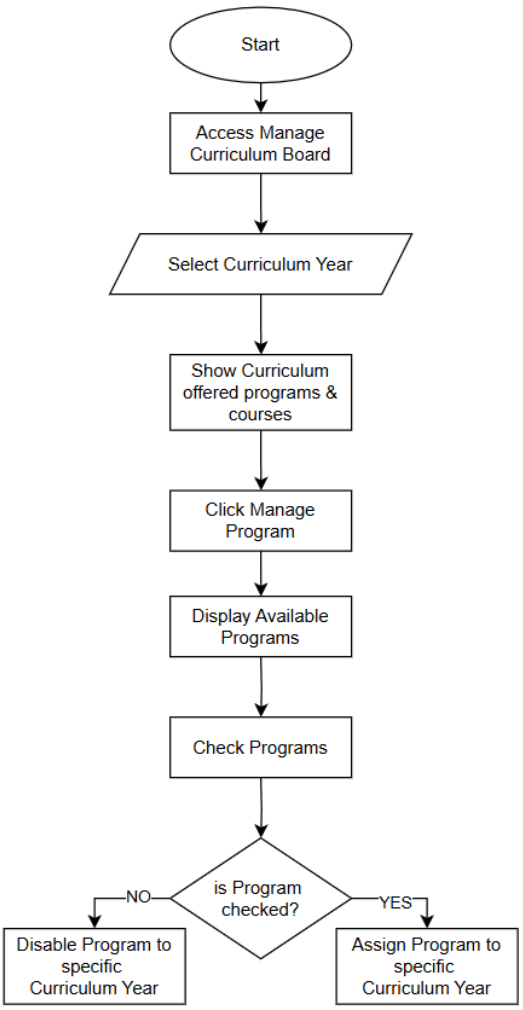


Figure 33 shows the process for managing programs in the system. The user begins by accessing the "Manage Curriculum Board" and selecting the curriculum year. The system then displays the offered programs and courses for that curriculum year. The user clicks on "Manage Program" to view the available programs. The programs are then checked to determine if they are selected. If a program is checked, it is assigned to the specific curriculum year. If it is not checked, the program is disabled for that curriculum year, and the process ends.

Figure 34. Adding Academic Years in the Admin Module

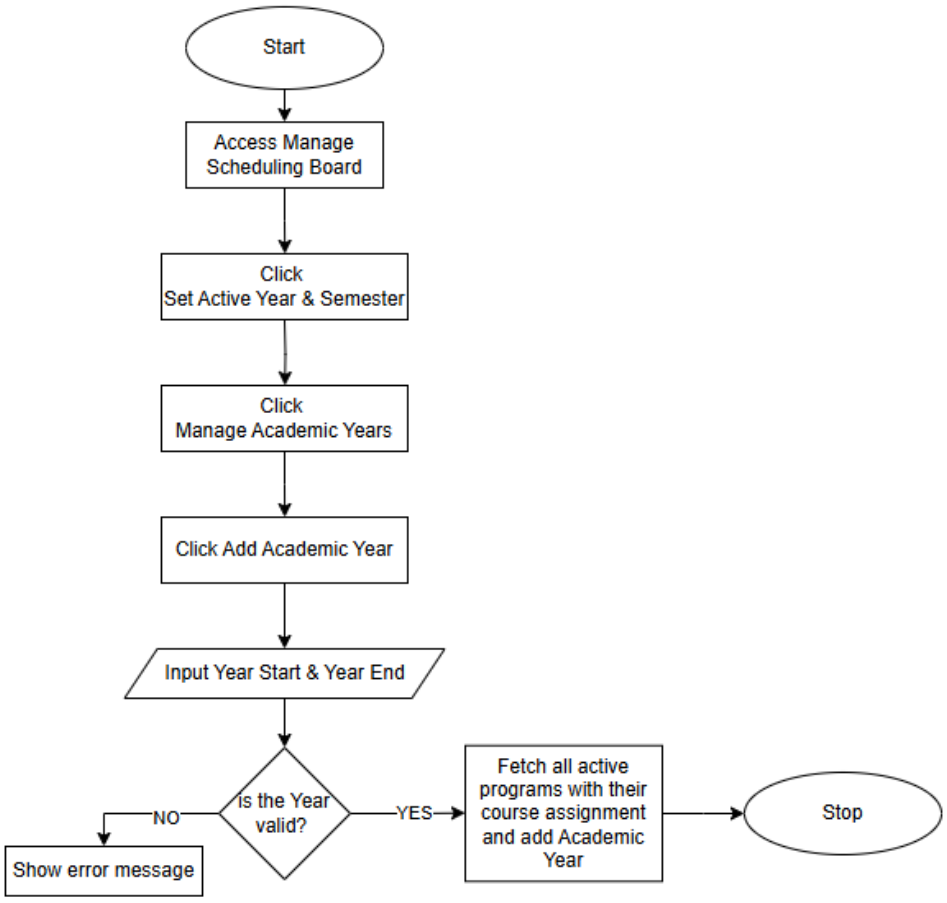


Figure 34 shows the process for managing academic years in the scheduling board. The user starts by accessing the "Manage Scheduling Board" and clicking on "Set Active Year & Semester." Then, the user navigates to "Manage Academic Years" and selects "Add Academic Year." After that, the user inputs the start and end years for the new academic year. The system checks if the year is valid. If not, an error message is displayed. If the year is valid, the system fetches all active programs with their course assignments and adds the new academic year, completing the process.

Figure 35. Managing Year Levels Program Curriculum in the Admin Academic Year Board

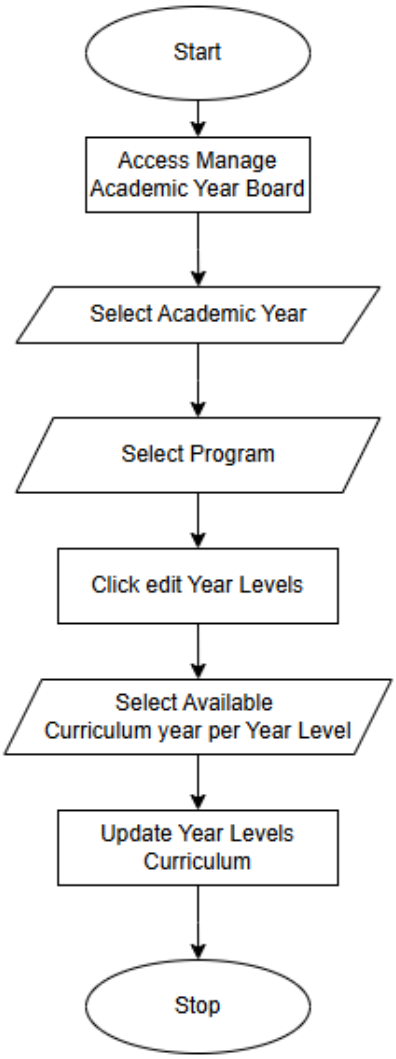


Figure 35 shows the process for updating year levels in the system. The user begins by accessing the "Manage Academic Year Board" and selecting the relevant academic year. Next, the user selects the program and clicks to edit the year levels. After that, the user selects the available curriculum year for each year level. Finally, the curriculum for the year levels is updated, and the process is complete.

Figure 36. Managing Year Levels Program Sections in the Admin Academic Year Board

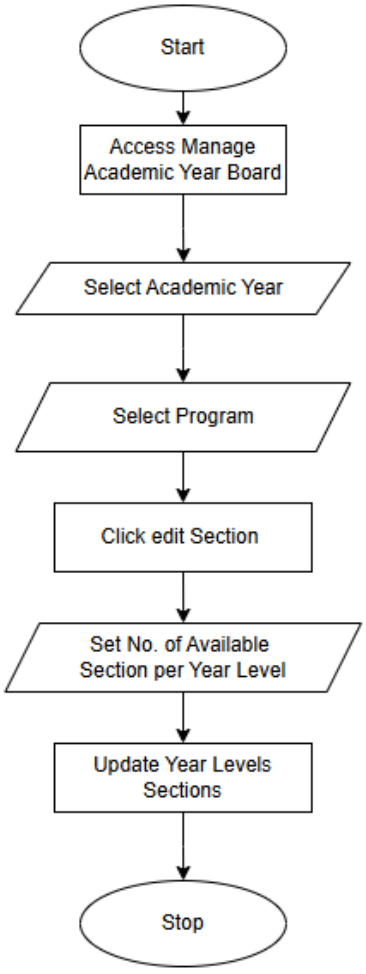


Figure 36 shows the process for updating sections in the system. The user begins by accessing the "Manage Academic Year Board" and selecting the academic year. After selecting the program, the user clicks to edit the sections. The next step is to set the number of available sections for each year level. Finally, the sections for the year levels are updated, and the process is complete.

Figure 37. Deleting a Specific Program in the Selected Academic Year in Admin Academic Year Board

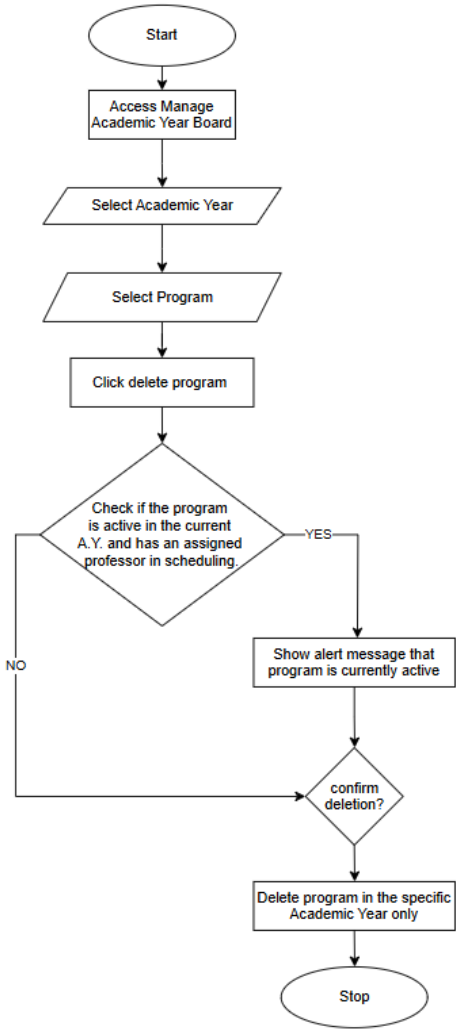


Figure 37 shows the process for deleting a program in the system. The user starts by accessing the "Manage Academic Year Board" and selecting the academic year and program. After selecting the program, the user clicks on "Delete Program." The system then checks if the program is active in the current academic year and has an assigned professor in scheduling. If the program is active, an alert message is displayed, indicating that the program cannot be deleted. If not, the system asks for confirmation to proceed with deletion. Upon confirmation, the program is deleted for the specific academic year, and the process concludes.

Figure 38. Deleting a Academic Year in the in Admin Academic Year Board

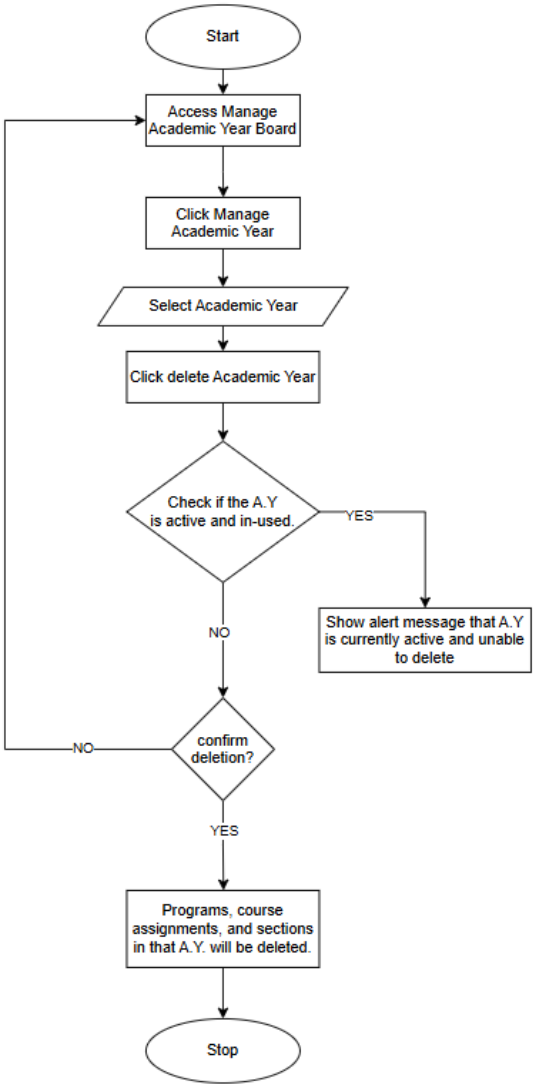


Figure 38 shows the process for deleting an academic year in the system. The user starts by accessing the "Manage Academic Year Board" and selecting "Manage Academic Year." After selecting the desired academic year, the user clicks "Delete Academic Year." The system then checks if the academic year is currently active and in use. If it is active, an alert message is shown, indicating that the academic year cannot be deleted. If the academic year is not in use, the system asks for confirmation to proceed with deletion. Upon confirmation, all programs, course assignments, and sections associated with that academic year will be deleted, and the process concludes.

Figure 39. Set Active Semester in the Admin Scheduling Module

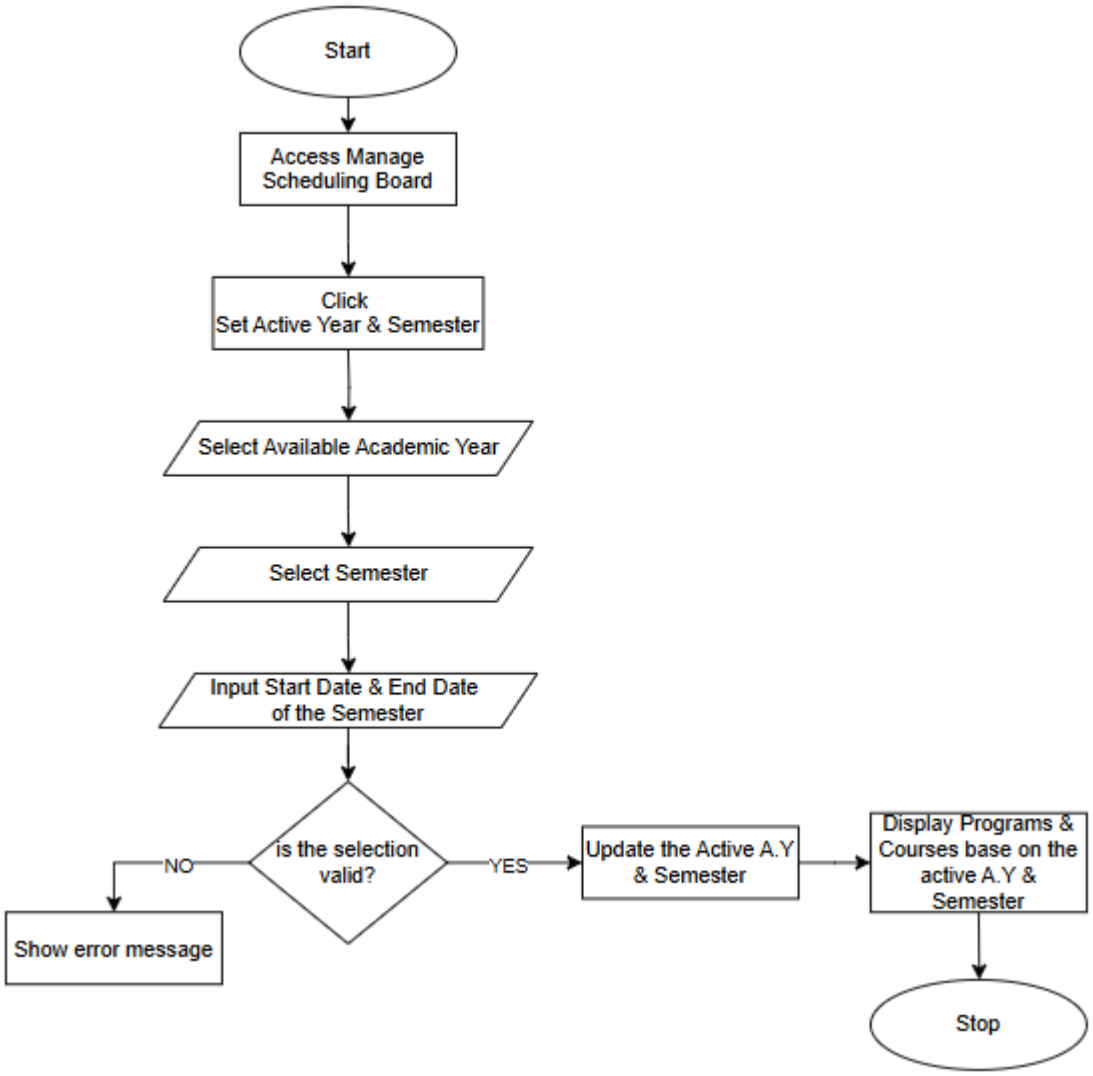


Figure 39 shows the process for setting the active academic year and semester in the system. The user begins by accessing the "Manage Scheduling Board" and selecting "Set Active Year & Semester." The next step is to choose an available academic year and semester. After selecting the semester, the user inputs the start and end dates for the semester. The system then checks if the selection is valid. If the selection is invalid, an error message is displayed. If valid, the active academic year and semester are updated, and the system displays programs and courses based on the newly set active year and semester, completing the process.

Figure 40. Viewing and Generating Faculty Preferences in the Admin Module

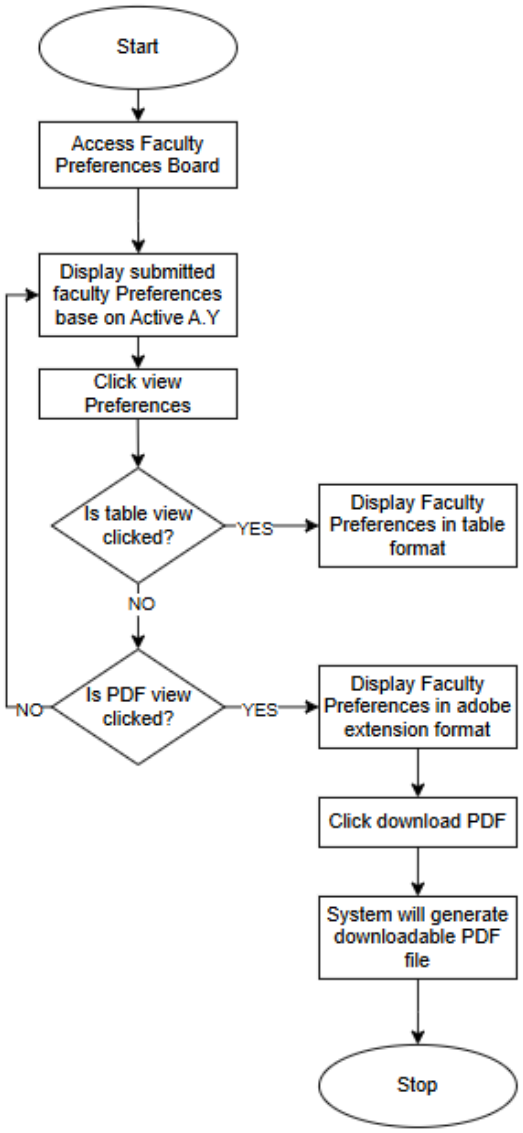


Figure 40 illustrates the process for viewing and generating faculty preferences in the system. The user starts by accessing the "Faculty Preferences Board" and displaying the submitted preferences based on the active academic year. The user can then click "View Preferences" to proceed. If the user selects the table view, the faculty preferences are displayed in a table format. If the PDF view is selected, the preferences are displayed in Adobe format. The user can then click "Download PDF," and the system will generate a downloadable PDF file, concluding the process.

Figure 41. Exporting All Submitted Faculty Preferences

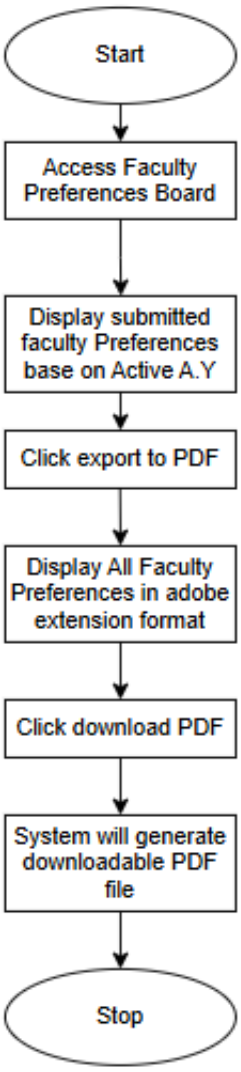
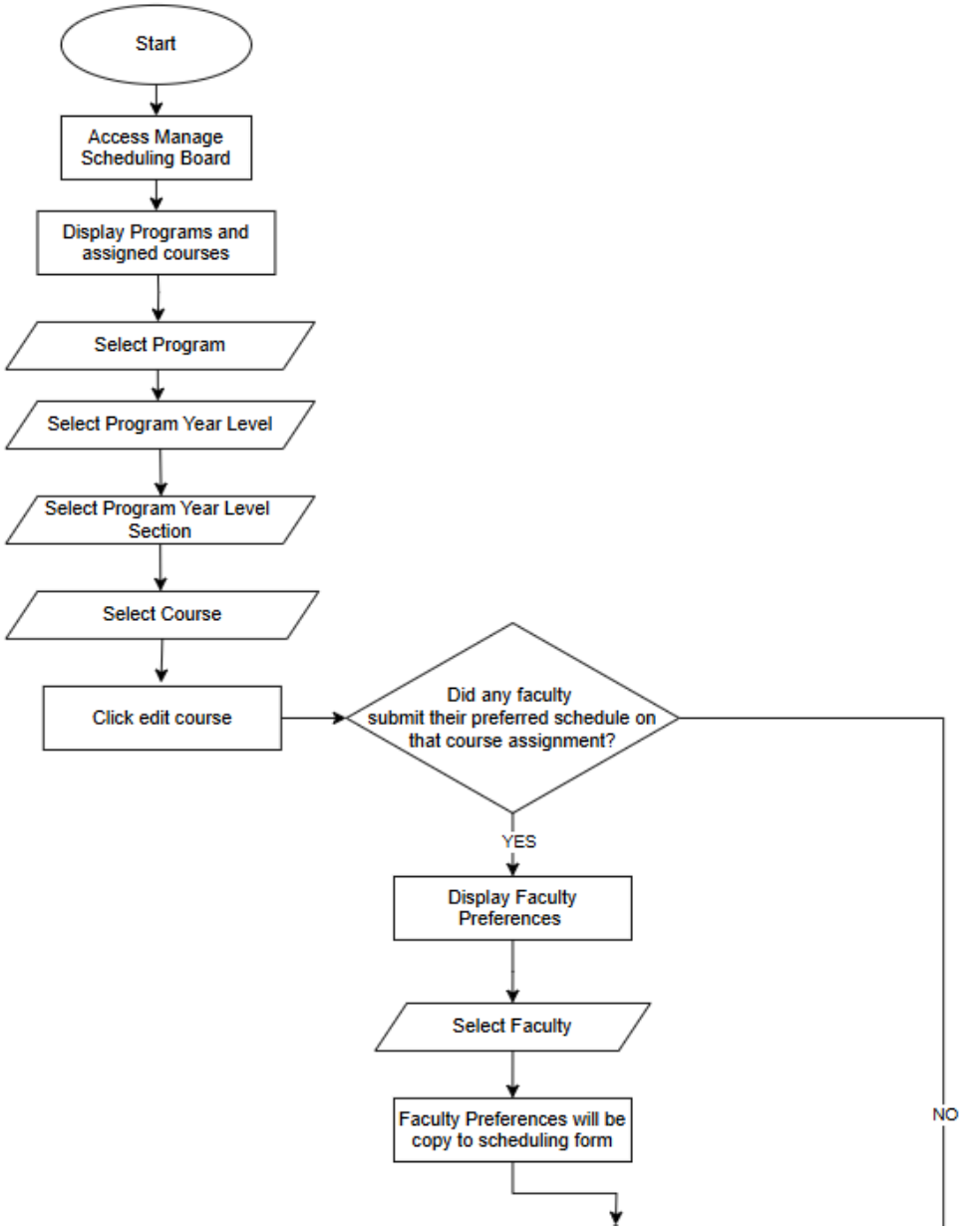


Figure 41 illustrates the process for exporting all submitted faculty preferences. The user begins by accessing the "Faculty Preferences Board" and displaying the submitted preferences based on the active academic year. The user then clicks "Export to PDF," and the system displays all faculty preferences in Adobe format. Afterward, the user clicks "Download PDF," and the system generates a downloadable PDF file, concluding the process.

Figure 42. Scheduling Faculty to specific Course Assignment in the Admin Scheduling Module



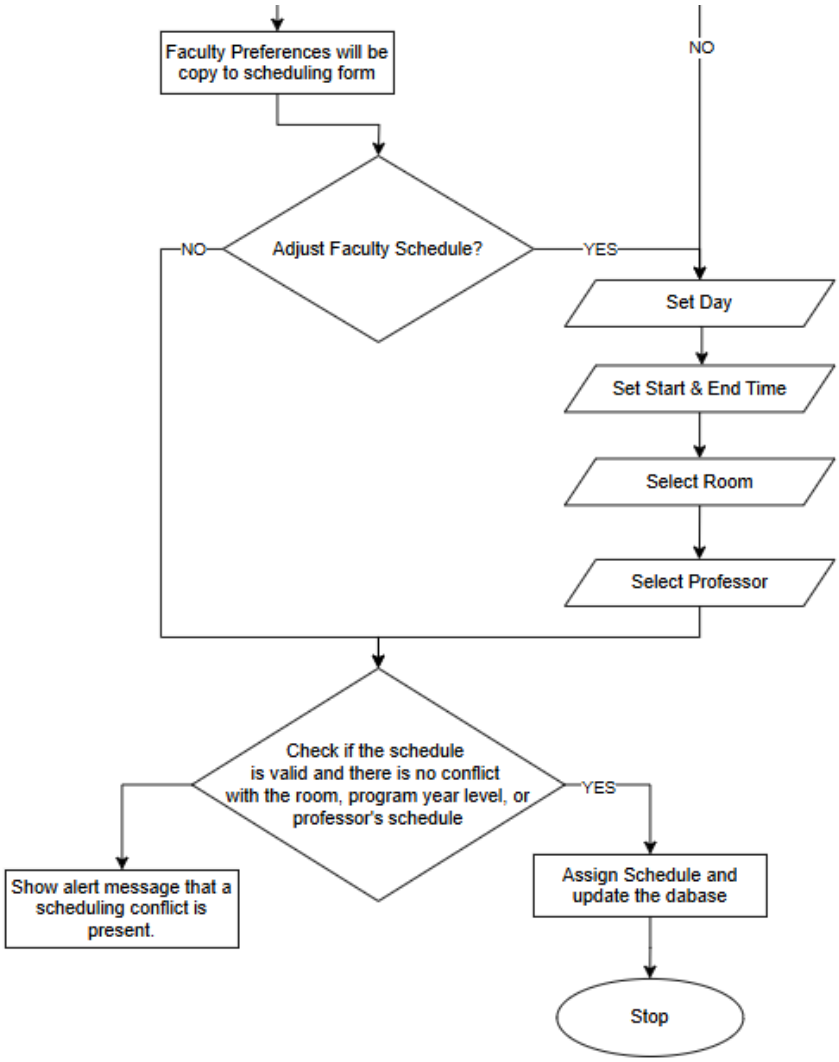


Figure 42 shows the process for scheduling of course assignment to specific program with specific year levels and sections. The user begins by accessing the "Manage Scheduling Board" and selecting a program, year level, section, and course. If a faculty member has submitted a preferred schedule for the course, the system displays those preferences. The user then selects the faculty and copies the preferences to the scheduling form. If needed, the user can adjust the schedule by setting the day, start and end times, room, and professor. The system then checks if the schedule is valid and free of conflicts with the room, program year level, or professor's schedule. If a conflict exists, an alert is shown. If the schedule is valid, it is assigned, and the system updates the database, completing the process.

Figure 43. Exporting Faculty Assigned Load and Schedule in Admin Module

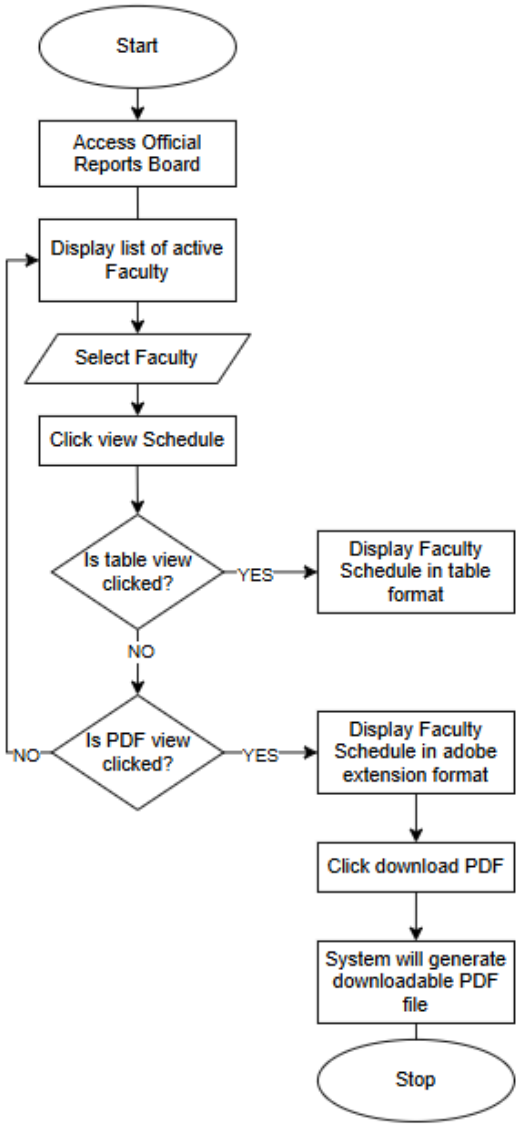


Figure 43 illustrates the process for exporting a faculty member's assigned load and schedule. The user starts by accessing the "Official Reports Board" and displaying the list of active faculty. After selecting a faculty member, the user clicks "View Schedule." If the table view is selected, the faculty schedule is displayed in table format. If the PDF view is selected, the schedule is displayed in Adobe format. The user can then click "Download PDF," and the system generates a downloadable PDF file, concluding the process.

Figure 44. Exporting Program Assigned Load and Schedule in Admin Module

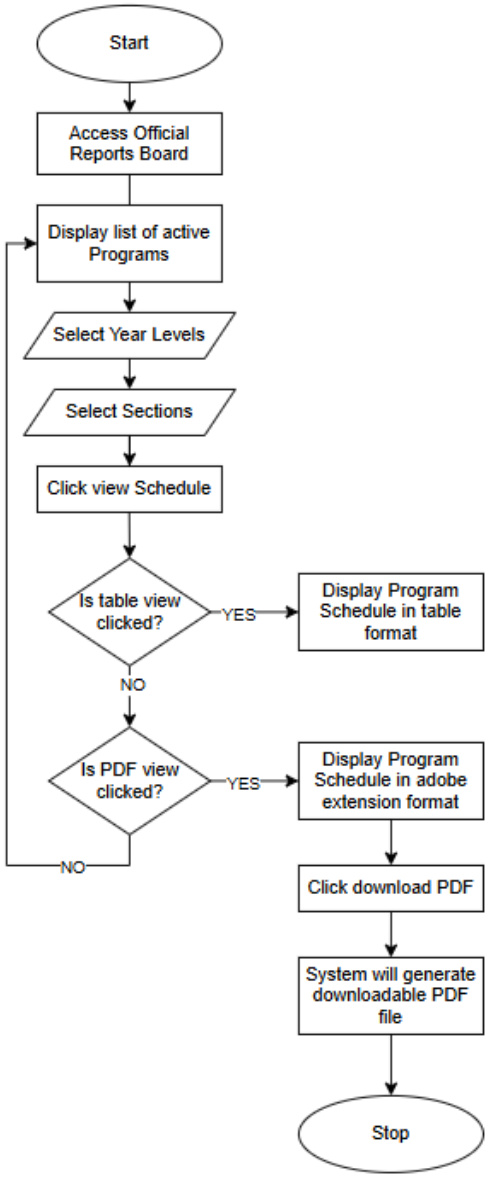


Figure 44 illustrates the process for exporting the schedule of a program. The user starts by accessing the "Official Reports Board" and displaying a list of active programs. The user then selects the year levels and sections before clicking "View Schedule." If the table view is selected, the program schedule is displayed in table format. If the PDF view is selected, the schedule is displayed in Adobe format. The user can then click "Download PDF," and the system generates a downloadable PDF file, concluding the process.

Figure 45. Exporting Room Assigned Load and Schedule in Admin Module

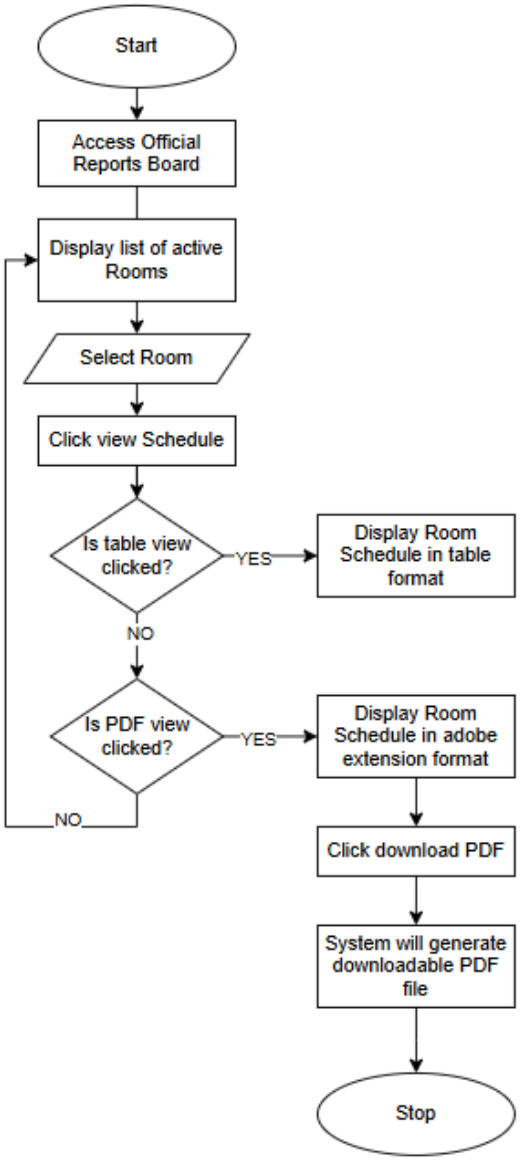


Figure 45 illustrates the process for exporting the schedule of a room. The user begins by accessing the "Official Reports Board" and displaying a list of active rooms. The user selects a room and clicks "View Schedule." If the table view is selected, the room schedule is displayed in table format. If the PDF view is selected, the schedule is displayed in Adobe format. The user can then click "Download PDF," and the system generates a downloadable PDF file, completing the process.

Figure 46. Publishing Assigned Load and Schedule of Faculty in Admin Reports Module

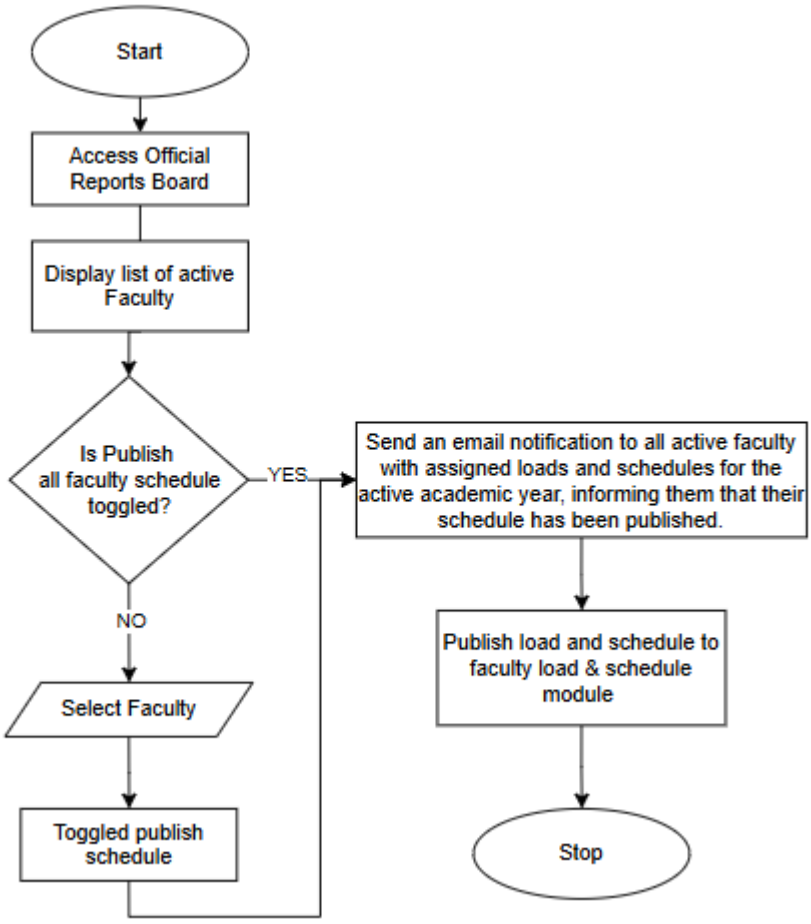


Figure 46 illustrates the process for publishing the load and schedule of faculty. The user begins by accessing the "Official Reports Board" and displaying a list of active faculty. If the option to "Publish All Faculty Schedules" is toggled, an email notification is sent to all active faculty with assigned loads and schedules for the active academic year, informing them that their schedule has been published. The system then publishes the load and schedule to the "Faculty Load & Schedule Module." If the "Publish All" option is not toggled, the user selects the individual faculty member, toggles the "Publish Schedule" option, and the schedule is published for that specific faculty member.

3.5.2 Database Design

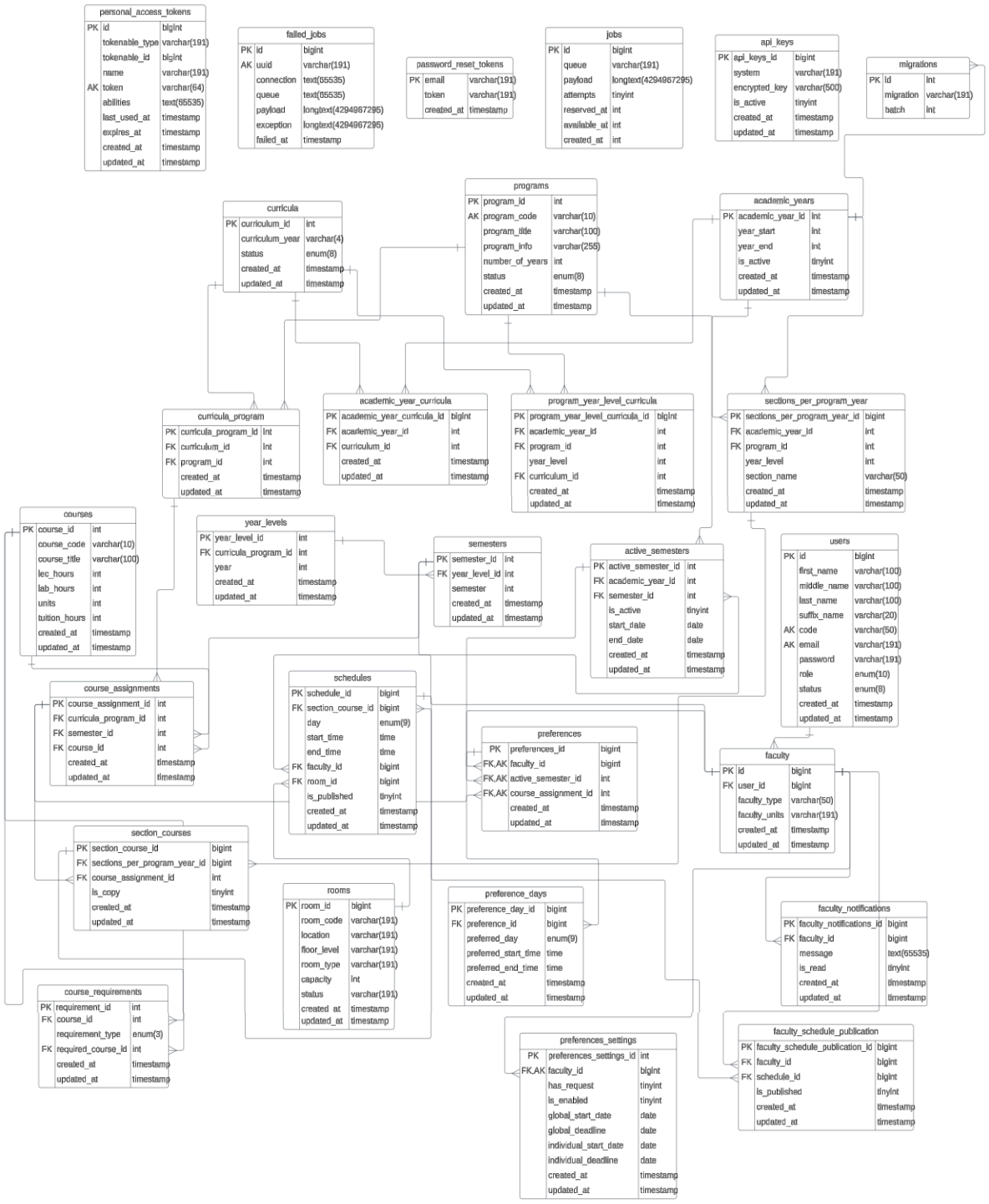


Figure 47. The new database design

3.5.3 Database Dictionary

users			
Field Name	**Data Type**	**Constraint**	**Description**
id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for each flss user.
last_name	varchar(191)	NOT NULL	Last name of the user
first_name	varchar(191)	DEFAULT NULL	First name of the user
middle_name	varchar(191)	DEFAULT NULL	Middle name of the user
suffix name	varchar(191)	DEFAULT NULL	Suffix name of the user (e.g., Jr., Sr.).
code	varchar(191)	NOT NULL, UNIQUE	Unique code/label for the user.
email	varchar(191)	NOT NULL, UNIQUE	Email address of the user.
password	varchar(191)	NOT NULL	Hashed password for user authentication.
role	Enum ('faculty','admin', 'superadmin')	NOT NULL	Role of the user within the system.
status	num ('Active','Inactive')	NOT NULL	Current status of the user account.

created_at	timestamp	NULL DEFAULT NULL	Timestamp when the user was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the user information was last updated

Table 2. Data Dictionary – users

programs			
Field Name	**Data Type**	**Constraint**	**Description**
program_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each program.
program_code	varchar(10)	NOT NULL, UNIQUE	Unique code representing the program (e.g., BSIT).
program_title	varchar(100)	NOT NULL	Official title of the program.
program_info	varchar(255)	NOT NULL	Brief description or information about the program.
number_of_years	int	NOT NULL	Duration of the program in years.
status	enum ('Active','Inactive')	NOT NULL	Current status of the program.

created_at	timestamp	NULL DEFAULT NULL	Timestamp when the program was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the program information was updated.

Table 3. Data Dictionary – programs

curricula			
Field Name	**Data Type**	**Constraint**	**Description**
curriculum_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each curriculum version.
curriculum_year	varchar(4)	NOT NULL, UNIQUE	Year associated with the curriculum (e.g., "2024").
status	enum ('Active','Inactive')	NOT NULL	Current status of the curriculum.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the curriculum was created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the curriculum was last updated.

Table 4. Data Dictionary – curricula

academic_years			
Field Name	**Data Type**	**Constraint**	**Description**
academic_year_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each program.
year_start	int(11)	NOT NULL	Starting year (e.g., 2025).
year_end	int(11)	NOT NULL	Ending year (e.g., 2026)
is_active	inyint(1)	NOT NULL DEFAULT 0	Indicates if this academic year is current.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the academic year was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the academic year was updated last.

Table 5. Data Dictionary – academic_years

courses			
Field Name	**Data Type**	**Constraint**	**Description**

course_id	Int(10) UNSIGNED	PRIMARY KEY,	Unique identifier for each curriculum version.
course_code	varchar(10)	NOT NULL, UNIQUE	Year associated with the curriculum (e.g., "2024").
course_title	varchar(100)	NOT NULL	Current status of the curriculum.
lec_hours	int(11)	NOT NULL	Number of lecture hours
lab_hours	int(11)	NOT NULL	Number of laboratory hours
units	int(11)	NOT NULL	Timestamp when the curriculum was created
tuition_hours	int(11)	NOT NULL	Used for tuition/fee calculation
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the course was created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the course information was updated

Table 6. Data Dictionary – courses

rooms			
Field Name	**Data Type**	**Constraint**	**Description**
room_id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for a room.
building_id	bigint UNSIGNED	NOT NULL, FOREGIN KEY	Foreign key referencing building
room_code	varchar(191)	NOT NULL	Encrypted form of the API key.
floor_level	varchar(191)	NOT NULL	Indicates Floor level of the room
room_type_id	bigint	NULL	Foreign key referencing room type
capacity	int	NOT NULL	Capacity of the room
status	varchar(191)	NOT NULL	Current status of the room
created_at	timestamp	NULL DEFAULT NULL	Timestamp of when the room was created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp of the last update

Table 7. Data Dictionary – rooms

room_type			
Field Name	**Data Type**	**Constraint**	**Description**
room_type_id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for room type
type_name	varchar(191)	NOT NULL	Name of the room type
created_at	timestamp	NULL DEFAULT NULL	Timestamp of when the room was created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp of the last update

Table 8. Data Dictionary – room_type

building			
Field Name	**Data Type**	**Constraint**	**Description**
building_id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for the building
building_name	varchar(191)	NOT NULL	Name of the building
floor_levels	int	NOT NULL	Number of floors in the building
created_at	timestamp	NULL DEFAULT NULL	Timestamp of when the building record was created

Table 9. Data Dictionary – building

curricula_program			
Field Name	**Data Type**	**Constraint**	**Description**
curricula_ program_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each curriculum-program link.
curriculum_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated curriculum.
program_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated program.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the link was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the link was updated.

Table 10. Data Dictionary – curricula_programs

year_levels			
Field Name	**Data Type**	**Constraint**	**Description**

year_level_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each curriculum-program link.
curricula_ program_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated curriculum-program link.
year	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Numerical representation of the year level (e.g., 1, 2, 3, 4).
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the year level was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the year level was updated.

Table 11. Data Dictionary – year_levels

semesters			
Field Name	**Data Type**	**Constraint**	**Description**

semester_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each curriculum-program link.
year_ **level_id**	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated curriculum-program link.
semester	int	DEFAULT NULL, FOREIGN KEY	Numerical representation of the year level (e.g., 1 = 1st sem, 2 = 2nd sem, 3rd summer,).
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the semester was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the semester was updated.

Table 12. Data Dictionary – semesters

course_assignments			
Field Name	**Data Type**	**Constraint**	**Description**

course_ assignment_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each course assignment.
curricula_ program_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated curriculum-program link.
semester_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated semester.
course_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the assigned course.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the course assignment was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the course assignment was updated.

Table 13. Data Dictionary – course_assignment

course_requirements			
Field Name	**Data Type**	**Constraint**	**Description**
requirement_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each course requirement.
course_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the course that has the requirement.
requirement_type	enum ('pre','co')	NOT NULL	Type of requirement: 'pre' for prerequisite, 'co' for corequisite.
required_course_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the course that is required.

created_at	timestamp	NULL DEFAULT NULL	Timestamp when the requirement was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the requirement was updated.

Table 14. Data Dictionary – course_requirements

sections_per_program_year			
Field Name	**Data Type**	**Constraint**	**Description**
section_per_ _program_ year_id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for each section.
academic _year_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated academic year.
program_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated program.

year_level	int	NOT NULL	Numerical representation of the year level (e.g., 1, 2, 3, 4).
section_name	varchar(50)	NOT NULL	identifier for the section (e.g., "1", "2").
created_at	timestamp	NULL DEFAULT NULL	Timestamp when section per program was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when section per program was updated.

Table 15. Data Dictionary – sections_per_program_year

section_courses			
Field Name	**Data Type**	**Constraint**	**Description**

section_course_id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for each section-course association.
section_per_program_year_id	bigint UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated section.
course_assignment_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated course assignment.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the association was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the association was last updated.

Table 16. Data Dictionary – section_courses

faculty			
Field Name	**Data Type**	**Constraint**	**Description**
id	bigint(20) UNSIGNED	PRIMARY KEY	Unique identifier for each faculty member.
user_id	bigint(20) UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated user in the users table
hris_user_id	bigint(20) UNSIGNED	UNIQUE, DEFAULT NULL	HRIS user ID for syncing external system
faculty_type	varchar(50)	NOT NULL	Type of faculty (e.g., "Full-time", "Part-time")
faculty_units	varchar(191)	NOT NULL	Number of teaching units assigned to the faculty member.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when faculty was created.

updated_at	timestamp	NULL DEFAULT NULL	Timestamp when faculty was updated.

Table 17. Data Dictionary – faculty

schedules			
Field Name	**Data Type**	**Constraint**	**Description**
schedule_id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for each schedule entry.
section_ course_id	varchar(50)	NOT NULL, FOREIGN KEY	Reference to the associated section-course.
day	enum ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',	NOT NULL, UNIQUE	Day of the week when the course is scheduled.

	'Saturday', 'Sunday')		
start_time	time	DEFAULT NULL	Start time of the scheduled course
end_time	time	DEFAULT NULL	End time of the scheduled course.
faculty_id	bigint UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the faculty assigned to the course
room_id	bigint UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the room where the course is held
is_published	tinyint	NOT NULL, DEFAULT '0'	Indicates whether the schedule is published (1) or not (0)
created_at	timestamp	NULL DEFAULT	Timestamp when schedule

		NULL	was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when schedule was updated.

Table 18. Data Dictionary – schedules

active_semesters			
Field Name	**Data Type**	**Constraint**	**Description**
active_ semester_id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each active semester entry.
academic_year _id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated academic year.

semester_id	int UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated semester.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the active semester was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the active semester was last updated.

Table 19. Data Dictionary – active_semester

preferences			
Field Name	**Data Type**	**Constraint**	**Description**
preferences_id	bigint UNSIGNED)	PRIMARY KEY,	Unique identifier for each preference record.
faculty_id	bigint UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated faculty member.

active_semester_id	int UNSIGNED	NOT NULL, UNIQUE	Reference to the active semester context.
course_assignment_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the course assignment related to the preference.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when preference was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when preference was updated.

Table 20. Data Dictionary – preferences

preferences_days			
Field Name	**Data Type**	**Constraint**	**Description**
preferences_day_id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for the preference day entry.
preferred_day	enum('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday','Saturday', 'Sunday')	NOT NULL	Day of the week requested
preferred_start_time	time	NOT NULL	Start time for this preference.
preferred_end_time	time	NOT NULL	End time for this preference
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the setting was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when preference was updated.

Table 21. Data Dictionary – preferences_days

preferences_setting			
Field Name	**Data Type**	**Constraint**	**Description**
preferences_settings_id	int UNSIGNED	PRIMARY KEY	Unique identifier for each preferences setting.
faculty_id	bigint UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated faculty member.
has_request	tinyint	NOT NULL, DEFAULT '0'	1 for request made for preference, 0 for no request.
is_enabled	tinyint	NOT NULL, DEFAULT '0'	Indicates if preferences are enabled (1) or disabled (0).
global_start_date	date	DEFAULT NULL	Global start date for preference setting
global_deadline	date	DEFAULT NULL	Global deadline of preference for all users, if applicable.

individual_start_date	date	DEFAULT NULL	Individual start date for this faculty
individual_deadline	date	NULL	Specific submission preference deadline for individual users.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the setting was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the setting was updated.

Table 22. Data Dictionary – preferences_setting

faculty_schedule_publication			
Field Name	**Data Type**	**Constraint**	**Description**
faculty_schedule_publication_id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for each publication record.
faculty_id	bigint UNSIGNED	DEFAULT NULL, FOREIGN KEY	Reference to the associated faculty member.

schedule_id	bigint UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated schedule.
is_published	tinyint(1)	NOT NULL, DEFAULT '0'	Indicates if the schedule is published (1) or not (0).
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the publication record was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the publication record was last updated.

Table 23. Data Dictionary – faculty_schedule_publication

academic_year_curricula			
Field Name	**Data Type**	**Constraint**	**Description**
academic_ year_curricula _id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for each academic year-curriculum link.

academic_ year_id	int UNSIGNED	NOT NULL, FOREIG N KEY	Reference to the associated academic year.
curriculum_id	int UNSIGNED	NOT NULL, FOREIGN KEY	Reference to the associated curriculum.
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the link record was created.
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the link record was last updated.

Table 24. Data Dictionary – academic_year_curricula

program_year_level_curricula			
Field Name	**Data Type**	**Constraint**	**Description**
program_year_level_curricula_id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for the record
academic_year_id	int UNSIGNED	NOT NULL, FOREIGN KEY	References the academic year
program_id	int UNSIGNED	NOT NULL, FOREIGN KEY	References the program
year_level	int(11)	NOT NULL	Numeric year level
curriculum_id	int(10) UNSIGNED	NOT NULL, FOREIGN KEY	References the curriculum
created_at	timestamp	NULL DEFAULT NULL	Timestamp of record creation created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp of record creation updated

Table 25. Data Dictionary – program_year_level_curricula

faculty_notifications			
Field Name	**Data Type**	**Constraint**	**Description**
faculty_ notifications_id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for a notification.
faculty_id	bigint UNSIGNED	NOT NULL	References the faculty record
message	TEXT	NOT NULL	Notification text/message
is_read	TINYINT(1)	NOT NULL, DEFAULT 0	Flag indicating if the notification is read.
created_at	timestamp	NULL DEFAULT NULL	Timestamp of record creation created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp of record creation updated

Table 26. Data Dictionary – faculty_notifications

api_keys			
Field Name	**Data Type**	**Constraint**	**Description**

api_keys_id	bigint UNSIGNED	PRIMARY KEY	Unique identifier for the API key record.
system	varchar(191)	NOT NULL	Name or identifier of the external system
encrypted_key	varchar(500)	NOT NULL	Encrypted form of the API key.
is_active	tinyint	NOT NULL, DEFAULT 1	Indicates whether the key is active
created_at	timestamp	NULL DEFAULT NULL	Timestamp of record creation created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp of record creation updated

Table 27. Data Dictionary – api_keys

migrations			
Field Name	**Data Type**	**Constraint**	**Description**
id	int UNSIGNED	PRIMARY KEY,	Unique identifier for each migration record.

migration	varchar(191)	NOT NULL	Name of the migration file (e.g., "create_users_table").
batch	int	NOT NULL	Batch number indicating the order of migrations.

Table 28. Data Dictionary – migrations

personal_access_tokens			
Field Name	**Data Type**	**Constraint**	**Description**
id	bigint UNSIGNED	PRIMARY KEY,	Unique identifier for each personal access token.
tokenable_type	varchar(191)	NOT NULL	Type of the model that owns the token (e.g., "App\Models\User")

tokenable_id	bigint UNSIGNED	NOT NULL, FOREIGN KEY	ID of the model that owns the token.
name	varchar(191)	NOT NULL, UNIQUE	Indicates if the schedule is published (1) or not (0).
abilities	text	NULL	JSON array specifying the abilities granted by the token.
last_used_at	timestamp	NULL DEFAULT NULL	Timestamp when the token was last used..
expires_at	timestamp	NULL DEFAULT NULL	Timestamp when the token expires
created_at	timestamp	NULL DEFAULT NULL	Timestamp when the token was created
updated_at	timestamp	NULL DEFAULT NULL	Timestamp when the token was last updated

Table 29. Data Dictionary – faculty_schedule_publication

3.6 Functional Specifications

This section includes the system boundaries of the proposed system which are shown through a use case diagram.

1.6.1 System Boundaries

The PUP-Taguig will employ the Faculty Loading and Scheduling system, utilizing data and processes compiled by the group. This system is designed to handle administrative tasks and manage details and transactions for both administrators and professors. Use-case diagram was a vital tool in helping developers define the requirements for the system. It provides insight into how each subsystem communicates and outlines the necessary flow of transactions.

Figure 48. Use Case Diagram for Administrator and Faculty users of the enhanced PUP Taguig Faculty Loading and Scheduling System

1.6.2 Use Case Report

Use Case ID:	UC-1
Use Case Name:	Manage Curriculum

Actor/ User:	Super Admin
Description:	The Super Admin is responsible for managing the curriculum, which includes adding new curricula and copying existing ones. Copying a curriculum will automatically copy associated course assignments and active programs.
Trigger:	1. Upon clicking “Add New Curriculum” in the Curriculum Management interface. 2.Upon selecting the “Copy Curriculum” option in the Curriculum List interface.
Pre-Conditions:	1. The user must be logged in with a Super Admin role. 2. The curriculum being added or copied must adhere to system standards (e.g., course details, programs, etc.).
Post-conditions:	1. The system will successfully add a new curriculum to the curriculum list. 2. If a curriculum is copied, the associated course assignments and active programs will also be copied. 3. The new or copied curriculum will be saved in the system and available for future edits or viewing by the Super Admin or other authorized users..
Normal Flow:	1. The Super Admin logs into the system with valid credentials. 2. Upon successful login, the Curriculum Management interface appears. 3. The Super Admin clicks the “Add New Curriculum” button. 4. The system prompts the Super Admin to enter the necessary curriculum details (e.g., curriculum name, description, course requirements, etc.). 5. The Super Admin enters the required details and clicks “Save”.

	6. The system confirms that the new curriculum has been successfully added. 7. Alternatively, the Super Admin may click the "Copy Curriculum" button next to an existing curriculum. 8. The system prompts the Super Admin to confirm the curriculum to copy. 9. The Super Admin confirms and clicks "Copy". 10. The system copies the selected curriculum, including all associated courses and active programs. 11. The system confirms that the curriculum has been successfully copied and added to the list.
Alternative Flow:	1. If the Super Admin clicks "Cancel" during the Add or Copy process, the action is aborted, and no changes are made to the curriculum list. 2. If any required fields (e.g., curriculum name) are left blank, the system will display an error message indicating the missing information and will not allow the Super Admin to save the curriculum until all fields are completed.
Exceptions:	1. If the Super Admin tries to add a curriculum with invalid or duplicate data, an error message will be displayed, and the system will not allow the curriculum to be saved. 2. If the Super Admin exceeds the allowed session time, a timeout error will be displayed, and they will need to log in again to proceed. 3. If there is an issue copying the curriculum (e.g., associated programs or courses cannot be copied), an error message will appear explaining the issue. The curriculum copy process will not complete until the issue is resolved.
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Whenever the Super Admin needs to create or copy a curriculum, especially during the academic year start or curriculum review periods.
Assumptions:	1. The system has predefined course

	assignments and active programs that can be copied when the curriculum is copied. 2. The system ensures that there are no duplicates or conflicts when copying curricula.
Notes and Assumption:	1. Curriculum details must be entered correctly to avoid errors in the curriculum list. 2. Copying a curriculum will only copy active courses and programs associated with it at the time of copying.

Table 25. Use Case Report – Curriculum Management

Use Case ID:	UC-2
Use Case Name:	Course Assignment Management

Actor/ User:	Super Admin
Description:	The Super Admin can assign specific courses to programs, set specific year levels and semesters, and generate reports for the course assignments. This module allows for detailed mapping of courses to academic programs and generates insights through reports.
Trigger:	1. Upon clicking the “Add Course Assignment” button in the Course Assignment Management interface. 2. Upon selecting “Generate Report” after course assignments have been added.
Pre-Conditions:	1. The user must be logged in as a Super Admin with access to course assignment functionalities. 2. The program to which courses will be assigned must already exist in the system. 3. The course to be assigned must already be available in the course catalog. 4. The system must have defined year levels and semesters available for assignment.
Post-conditions:	1. The system will assign the selected course to the chosen program, year level, and semester.

	2. The course assignment will be stored in the system and linked to the relevant program and semester. 3. The system will generate a report detailing the course assignments, including program names, courses, year levels, and semesters.
Normal Flow:	1. The Super Admin logs into the system. 2. After logging in, the Course Assignment Management interface appears. 3. The Super Admin clicks the “Add Course Assignment” button. 4. The system displays a form with fields to select a program, course, year level, and semester. 5. The Super Admin selects the program, course, year level, and semester from the dropdown menus. 6. After filling in the necessary fields, the Super Admin clicks “Save” to assign the course. 7. The system confirms the course has been successfully assigned to the program, year level, and semester. 8. Alternatively, after assigning courses, the Super Admin can click “Generate Report” to produce a report on the course assignments. 9. The system generates the report, displaying details of all course assignments by program, year level, and semester.
Alternative Flow:	1. If the Super Admin cancels the assignment, the action is aborted, and no course is assigned. 2. If required fields (e.g., course or program) are missing, the system will show an error message and will not allow the assignment to proceed until all fields are filled.
Exceptions:	1. If Super Admin selects an invalid or nonexistent program or course, the system will show an error message and prevent the course assignment. 2. If the system detects a conflict in year levels or semesters (e.g., multiple courses assigned to the same semester), an error will be shown, and the Super Admin will need to resolve the conflict before proceeding.
Includes:	(TO BE UPDATE)

Priority:	High
Frequency of Use:	Whenever the Super Admin needs to assign courses to programs and generate reports.
Assumptions:	1. The system has a valid course catalog and program list. 2. The year levels and semesters are pre-configured in the system for assignment. 3. The report generation feature accurately reflects the course assignments stored in the system.
Notes and Assumption:	1. Course assignments must be done correctly to avoid conflicts with other curriculum assignments. 2. Reports should be easy to interpret and provide clear details about the assigned courses and programs.

Table 26. Use Case Report – Course Assignment Management Module

Use Case ID:	UC-3
Use Case Name:	Program Management

Actor/ User:	Super Admin
Description:	The Super Admin can add programs offered by the school, mark programs as active, and manage the list of programs. Active programs will be fetched during curriculum creation. Super Admin can also generate reports detailing the programs in the system.
Trigger:	1. Upon clicking the “Add New Program” button in the Program Management interface. 2. Upon selecting “Generate Program Report” after adding or managing programs.
Pre-Conditions:	1. The user must be logged in as a Super Admin with access to program management functionalities. 2. The program being added must include necessary details (e.g., program name, description, academic level). 3. The program should be marked as active if it

	is to be included during curriculum creation.
Post-conditions:	1. The program will be successfully added to the system and saved with the correct details (name, description, etc.). 2. Active programs will be fetched during the curriculum creation process. 3. The system will generate a report detailing the active and inactive programs, including their attributes and status.
Normal Flow:	1. The Super Admin logs into the system. 2. Upon logging in, the Program Management interface is displayed. 3. The Super Admin clicks the “Add New Program” button. 4. The system prompts Super Admin to fill in the required fields (e.g., program name, description, academic level, active/inactive status). 5. The Super Admin fills in the required details and marks the program as Active or Inactive as needed. 6. After filling in the details, the Super Admin clicks “Save” to add the program to the system. 7. The system confirms that the program has been successfully added. 8. Alternatively, after managing the programs, the Super Admin can click “Generate Program Report” to produce a report of all programs in the system. 9. The system generates the report, displaying the list of programs, their status (active/inactive), and other relevant details.
Alternative Flow:	None
Exceptions:	1. If Super Admin tries to add a program with invalid or duplicate details, the system will show an error message and prevent the program from being saved. 2. If the system detects that the program has a conflicting status (e.g., both active and inactive), an error will be shown, and the Super Admin will need to resolve the issue before proceeding.
Includes:	(TO BE UPDATE)
Priority:	High

Frequency of Use:	Whenever the Super Admin needs to add, update, or manage programs, or generate reports on program details.
Assumptions:	1. The system has predefined fields for program attributes (name, description, status, etc.). 2. Only active programs will be included during curriculum creation. 3. The report generation accurately reflects the active and inactive programs stored in the system.
Notes and Assumption:	1. The program details must be correct to ensure that active programs are properly fetched for curriculum creation. 2. The report should clearly differentiate between active and inactive programs, along with any other program details the Super Admin requires.

Table 27. Use Case Report – Program Management Module

Use Case ID:	UC-4
Use Case Name:	Room Management

Actor/ User:	Super Admin
Description:	This module allows the Super Admin to add, edit, and remove room details, and generate reports on room availability and usage.
Trigger:	1. Upon clicking the “Add New Room” button in the Room Management interface. 2. Upon selecting “Generate Room Report” after managing room details.
Pre-Conditions:	1. The user must be logged in as a Super Admin with access to room management functionalities. 2. Rooms should be assigned with the necessary details (e.g., room number, capacity, and location).
Post-conditions:	1. New rooms will be successfully created and saved in the system. 2. Existing room details can be updated or deleted.

	3. A report will be generated that details room availability, capacity, and other related information.
Normal Flow:	1. The Super Admin logs into the system. 2. Upon logging in, the Room Management interface is displayed. 3. The Super Admin clicks the “Add New Room” button. 4. The system prompts the Super Admin to enter necessary room details (e.g., room number, building, capacity, and status). 5. The Super Admin enters the required details and clicks “Save” to add the room to the system. 6. The system confirms the room has been successfully created and added. 7. The Super Admin can select an existing room from the list to edit or delete. 8. To edit a room, the Super Admin clicks the “Edit” button, modifies the room details, and clicks “Save”. 9. To delete a room, the Super Admin clicks “Delete” and confirms the action. 10. After managing rooms, the Super Admin can click “Generate Room Report” to produce a report of all rooms. The system generates the report, displaying room numbers, capacities, locations, and status.
Alternative Flow:	1. If the Super Admin cancels the action during room creation or editing, no changes will be made, and the system will return to the main Room Management interface. 2. If required fields (e.g., room number or capacity) are left blank, the system will show an error message and prevent the action from proceeding.
Exceptions:	1. If the Super Admin tries to add a room with duplicate or invalid details (e.g., room number already exists), the system will display an error message and prevent the room from being added.
Includes:	(TO BE UPDATE)
Priority:	High

Frequency of Use:	Whenever the Super Admin needs to add, update, delete, or generate reports on room details.
Assumptions:	1. Room details should be correctly entered to avoid conflicts or duplication. 2. The system provides valid data fields for room management, such as room number, capacity, and location. 3. Report generation should include all room details, including usage and availability where applicable.
Notes and Assumption:	1. Room management functionalities must be accurately performed to ensure that rooms are available for scheduling and usage. 2. The report should display a comprehensive list of rooms with their details (e.g., availability, capacity, location).

Table 28. Use Case Report – Room Management Module

Use Case ID:	UC-5
Use Case Name:	Admin Overview

Actor/ User:	Admin
Description:	The Admin can perform quick actions such as enabling faculty to submit preferences within a set date range, publishing faculty schedules once available, and generating reports about rooms, programs, and faculty schedules.
Trigger:	1. Upon clicking the “Enable Faculty to Submit Preferences” button in the Admin Overview interface. 2. Upon clicking the “Publish Faculty Schedule” button once the faculty schedules are ready. 3. Upon selecting “Generate Report” to view reports related to room assignments, program offerings, or faculty schedules.
Pre-Conditions:	1. The user must be logged in as an Admin with access to the overview and management functions. 2. The faculty schedules, room assignments, and

	program details should already exist in the system to be included in the reports.
Post-conditions:	1. Faculty preferences will be enabled for submission within the specified date range. 2. The faculty schedule will be published and made available to the faculty. 3. A report will be generated detailing the room assignments, program offerings, or faculty schedules.
Normal Flow:	1. The Admin logs into the system. 2. Upon logging in, the Admin Overview dashboard is displayed. 3. The Admin clicks the “Enable Faculty to Submit Preferences” button. 4. The system prompts the Admin to specify the start and end date for faculty preferences submission. 5. The Admin sets the required date range and clicks “Enable”. 6. The system confirms that the faculty preference submission window is now open for the specified dates. 7. Once the faculty schedules are ready, the Admin clicks the “Publish Faculty Schedule” button. 8. The system prompts the Admin to confirm that the schedule is ready for publication. 9. The Admin confirms and clicks “Publish”. 10. The system confirms that the faculty schedule has been published. 11. The Admin clicks “Generate Report” to produce a report. 12. The system generates the requested report, displaying details of room assignments, program offerings, or faculty schedules.
Alternative Flow:	1. If the Admin cancels the action during the faculty preference submission process, no changes will be made, and the system will return to the main overview interface. 2. If the Admin attempts to publish a faculty schedule before it is ready, the system will show an error message and prevent the schedule from being published. 3. If required fields for generating the report

	(e.g., selection of room, program, or schedule) are not provided, the system will show an error message and prompt the Admin to make a valid selection.
Exceptions:	1. If the Admin tries to enable faculty preferences outside the allowed submission dates, the system will show an error message indicating the submission period is not valid. 2. If the Admin attempts to publish the faculty schedule without confirming that all details are correct, the system will show an error and prevent publishing until confirmed. 3. If the system encounters an issue generating the report (e.g., missing data or incorrect program/room/faculty assignments), the Admin will be notified that the report could not be generated.
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Whenever the Admin needs to enable faculty preference submissions, publish faculty schedules, or generate reports.
Assumptions:	1. The system has valid date ranges for faculty preferences and faculty schedules. 2. The reports generated should provide clear and actionable insights into room assignments, programs, and faculty schedules. 3. The system should be able to track the status of faculty preferences and faculty schedules.
Notes and Assumption:	1. Admin actions should be executed in a timely manner to avoid delays in faculty preference submissions and schedule publishing. 2. Report generation should include all necessary details for decision-making regarding room assignments, programs, and faculty schedules.

Table 29. Use Case Report- Admin Overview Module

Use Case ID:	UC-6
Use Case Name:	Faculty Preference

Actor/ User:	Admin
Description:	The Admin can view, print, and manage faculty preferences. The Admin has control over whether faculty can submit preferences via a toggle button. The toggle can be set universally (for all faculty) or individually (for specific faculty members). Admin can also enable faculty to resubmit preferences if a change request is made.
Trigger:	1. Upon accessing the Faculty Preference Management page from the Admin dashboard. 2. Upon toggling the “Enable Faculty to Submit Preferences” button (either universally or individually). 3. Upon viewing or printing submitted faculty preferences.
Pre-Conditions:	1. The Admin must be logged in and have proper access to manage faculty preferences. 2. The faculty members must have submitted preferences within the allowed date range when the toggle is on. 3. The preference submission toggle must be accessible for faculty to submit or edit preferences.
Post-conditions:	1. The Admin can view submitted preferences for all or specific faculty members. 2. The Admin can enable or disable the preference submission toggle for individual faculty members or for all faculty members. 3. The Admin can approve change requests by enabling the submission toggle for individual faculty members, allowing them to resubmit their preferences.
Normal Flow:	1. The Admin logs into the system and accesses the Faculty Preference Management page. 2. The system displays a list of active faculty and their current preference submission status (enabled/disabled). 3. The Admin can toggle the submission status for individual faculty or toggle it universally for all faculty members. • If the toggle is ON, faculty members can submit or edit their preferences. • If the toggle is OFF, faculty members

	cannot submit preferences. 4. The Admin views the submitted preferences and can print them as needed. 5. If a faculty member requests to change their preferences, the Admin enables the toggle for that specific faculty member, allowing them to resubmit their preferences. 6. Once the toggle is enabled, the faculty member can update and submit their preferences again. 7. The Admin confirms the resubmission and change request.
Alternative Flow:	1. The Admin may choose to toggle the preference submission status for individual faculty members, enabling or disabling the ability for specific faculty to submit or change their preferences. 2. The Admin may also choose to print the preferences of any or all faculty by selecting the “Print Preferences” option in the interface.
Exceptions:	1. If the Admin attempts to submit preferences while the toggle is off, the system will show a message indicating that preferences cannot be submitted at this time. 2. If a faculty member needs to change their preferences after the submission period is closed, the Admin must manually enable the toggle to allow them to resubmit their preferences
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Whenever the Admin needs to enable faculty preference submissions, or generate reports.
Assumptions:	1. The toggle will be used to manage submission periods effectively. 2. Faculty members’ preferences can be accessed, printed, or resubmitted if required.
Notes and Assumption:	1. The Admin has the ability to manage preferences globally (for all faculty) or on an individual basis. 2. Faculty preferences can be printed for record-keeping or review purposes.

Table 30. Use Case Report- Admin Faculty Preference Module

Use Case ID:	UC-7
Use Case Name:	Manage Academic Year

Actor/ User:	Admin
Description:	The Admin can add a new academic year, assign curricula to different year levels within programs, and manage sections and course assignments. The system supports the assignment of multiple curricula within a single academic year, allowing for variations in course unit hours, course codes, and other specifics between different year levels.
Trigger:	1. Upon accessing the Manage Academic Year page from the Admin dashboard. 2. Upon clicking the “Add New Academic Year” button to create a new academic year. 3. Upon assigning curricula to specific year levels within a program.
Pre-Conditions:	1. The Admin must be logged in and have access to the Manage Academic Year functionality. 2. Active programs must already be available in the system to which curricula can be assigned. 3. Curriculum information must be available for assignment to each year level.
Post-conditions:	1. A new academic year will be added to the system. 2. Curricula will be successfully assigned to each year level within a specific program. 3. Sections and course assignments will be properly managed to accommodate different curricula for the same course across multiple year levels.
Normal Flow:	1. The Admin logs into the system and accesses the Manage Academic Year page. 2. The Admin clicks the “Add New Academic Year” button. 3. The system prompts the Admin to enter the academic year details (e.g., start and end dates, academic year name). 4. The Admin enters the necessary information

	and clicks “Save”. 5. The system creates the new academic year and displays a list of active programs for the Admin to assign curricula. 6. The Admin selects a program and assigns specific curricula to each year level (e.g., Year 1, Year 2, Year 3, etc.). • The Admin can assign different curricula for the same course if required (e.g., different course codes, unit hours, or prerequisites). 7. The Admin reviews and confirms the assignments of curricula to each year level. 8. The system saves the changes and updates the academic year structure with the assigned curricula.
Alternative Flow:	1. If the Admin needs to assign different curricula for the same course across multiple year levels (due to different unit hours or course codes), they select the appropriate curriculum for each year level during the assignment process. 2. If the Admin needs to adjust course assignments or curriculum structure after the initial assignment, they can edit the curricula by selecting the appropriate program and year level, then modifying the curriculum details.
Exceptions:	1. If the Admin tries to assign a curriculum that is not available or compatible with the selected program/year level, the system will show an error message indicating that the assignment cannot be completed. 2. If the Admin forgets to assign a curriculum to a particular year level, the system will prompt the Admin to complete the assignment before saving.
Includes:	(TO BE UPDATE)
Priority:	High

Frequency of Use:	Whenever the Admin needs to add new academic year and adjust curricula in each year level
Assumptions:	The system can handle multiple curricula for the same course across different year levels in a single academic year.
Notes and Assumption:	1. The Admin has full control over assigning curricula, sections, and course assignments for each academic year. 2. The system should allow flexibility to accommodate differences in course content and structure for various year levels.

Table 31. Use Case Report- Academic Year Management Module

Use Case ID:	UC-8
Use Case Name:	Set Active Academic Year and Semester

Actor/ User:	Admin
Description:	The Admin can set the active academic year and semester for scheduling purposes. Once the active academic year and semester are set, the system will update course assignments and allow the Admin to assign schedules to specific courses within active programs and their corresponding year levels. This ensures that the right curriculum and course assignments are used in the scheduling board.
Trigger:	1. Upon accessing the Manage Academic Year page from the Admin Scheduling dashboard. 2. Upon clicking the “Set Active Academic Year” button for selecting the academic year and semester.
Pre-Conditions:	1. The Admin must be logged in and have access to the Manage Academic Year functionality. 2. The active academic year and semester must be properly configured in the system. 3. Active programs and curricula for each year level must be available.
Post-conditions:	1. The selected academic year and semester

	will be marked as active. 2. The course assignments for each active program and year level will be updated. 3. The Admin can now assign specific schedules to each course in the active program for the corresponding year level and semester.
Normal Flow:	1. The Admin logs into the system and accesses the Manage Academic Year page. 2. The Admin selects the “Set Active Academic Year” button from the interface. 3. The system displays a list of available academic years and semesters. 4. The Admin selects the appropriate academic year and semester to set as active. 5. The system updates the status of the selected academic year and semester as active. 6. The Admin then navigates to the Scheduling Board. • The Scheduling Board will now display the active academic year and semester, along with the corresponding programs and year levels. 7. The Admin assigns specific courses to their respective schedules within the active academic year, program, and semester. • The Admin selects the program and year level, and then assigns the appropriate course schedule for each course within the program. 8. The system ensures that the correct course assignments are linked to the active academic year and semester, reflecting the program’s year level and curriculum.
Alternative Flow:	1. If the Admin needs to modify the active academic year or semester, they can select a different year or semester and repeat the process. 2. If the Admin wants to update course assignments after setting the active year and semester, they can navigate to the Scheduling Board and modify the schedules

	accordingly.
Exceptions:	1. If the Admin tries to set an academic year or semester that is not yet configured in the system, the system will show an error message indicating that the selection is invalid.
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Whenever the Admin needs to set and start academic year and when preparing for course scheduling.
Assumptions:	1. The system allows flexibility in assigning schedules based on the academic year, semester, program, and year level. 2. The Admin has sufficient access rights to make changes to the active academic year and semester.
Notes and Assumption:	1. The Admin must ensure that the active academic year and semester are correctly configured before proceeding with course schedule assignments. 2. Course assignments should be updated in real-time on the Scheduling Board once the active academic year and semester are set.

Table 32. Use Case Report- Set Active Academic Year and Semester

Use Case ID:	UC-9
Use Case Name:	Scheduling Module

Actor/ User:	Admin
Description:	The Admin can perform scheduling for courses once the active academic year and semester are set. The Admin can view the available course assignments for specific programs and year levels, assign faculty to courses based on availability and preferences, and ensure there are no conflicts in scheduling (e.g., room, faculty, and program conflicts). The system automatically fetches the active semester and displays relevant course assignments.
	1. Upon setting the active academic year and

Trigger:	semester (refer to UC-: Set Active Academic Year and Semester). 2. Upon accessing the Scheduling Board or Schedule Management page.
Pre-Conditions:	1. The active academic year and semester must be set in the system. 2. Course assignments for the active semester and academic year must be available for specific programs and year levels. 3. Faculty preferences must be submitted and available in the system. 4. The Admin must have access to the Scheduling Board for performing scheduling tasks.
Post-conditions:	1. The Admin can assign faculty to courses based on availability, preferences, and the current schedule. 2. The system will display real-time information on room occupancy, faculty availability, and potential conflicts. 3. The scheduling algorithm will detect conflicts and notify the Admin if there are issues with room availability, faculty scheduling, or course assignments.
Normal Flow:	1. The Admin logs into the system and navigates to the Scheduling Board or Schedule Management page. 2. The system automatically fetches and displays the active semester and the corresponding course assignments for the selected program and year level. 3. The Admin selects a specific course assignment from the displayed list. 4. The system displays the available faculty for the selected course along with their preferences and availability (if faculty preferences have been submitted). 5. The Admin selects a faculty member to assign to the course. • If the faculty member is unavailable, the system will show a notification indicating the conflict. 6. The Admin assigns a room to the selected course.

	• The system will check room availability for the selected time and notify the Admin if there is a conflict. 7. The Admin can review all course assignments for the active semester, program, and year level. 8. The system runs a conflict detection algorithm to ensure there are no issues with the scheduling. • The system checks if the room is occupied, if the faculty member is available, and if any program has a class scheduled at the same time. • If a conflict is detected (e.g., overlapping course schedules, room conflicts, or faculty availability), the system will notify the Admin and provide possible solutions or alternative times. 9. The Admin confirms the schedule and finalizes the assignment. 10. The system saves the schedule and updates the scheduling board with the confirmed course assignments.
Alternative Flow:	1. If the Admin needs to modify the active academic year or semester, they can select a different year or semester and repeat the process. 2. If the Admin wants to update course assignments after setting the active year and semester, they can navigate to the Scheduling Board and modify the schedules accordingly.
Exceptions:	1. If the Admin tries to set an academic year or semester that is not yet configured in the system, the system will show an error message indicating that the selection is invalid.
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Frequently, especially during the preparation phase of each semester or academic year.

Assumptions:	1. The system can fetch the active semester and academic year data automatically. 2. Faculty preferences are properly submitted and available for scheduling. 3. The system's conflict detection algorithm will be able to detect common issues such as room conflicts, overlapping schedules, and faculty availability.
Notes and Assumption:	1. The Admin has the ability to manually override the system's conflict detection if needed, with appropriate notifications for any manual changes. 2. Faculty members' availability and preferences must be correctly inputted into the system for the scheduling algorithm to function properly.

Table 33. Use Case Report- Scheduling Module

Use Case ID:	UC-10
Use Case Name:	Official Report Management

Actor/ User:	Admin
Description:	The Admin can publish the official schedules for faculty members, generate reports for programs, and manage room utilization through the Official Report Module. The Admin can publish faculty schedules by toggling individual or universal settings, notify faculty via email, and print reports for faculty, programs, and rooms.
Trigger:	1. Upon accessing the Official Report Module from the Admin dashboard. 2. Upon clicking the Publish Schedule or Generate Report button for Faculty, Program, or Room sections.
Pre-Conditions:	1. The active academic year (A.Y.) and semester must be set in the system. 2. Faculty schedules must be finalized and available for publishing. 3. Program and room schedules must be configured and available for report generation.

	4. Faculty schedules must be assigned and ready for publishing.
Post-conditions:	1. The Admin can publish the faculty schedule for the active academic year and semester. 2. The Admin can generate and print program reports with assigned schedules for the active academic year. 3. Admin can generate and print room utilization reports for the active academic year and semester. 4. Faculty members will be notified via email about their published schedule.
Normal Flow:	1. The Admin logs into the system and accesses the Official Report Module from the dashboard. 2. The Admin selects one of the following sections: Faculty Report, Program Report, or Room Report. **Faculty Report** 1) The Admin selects the Publish Faculty Schedule button. 2) The system displays all the faculty members and their respective schedules for the active academic year and semester. 3) The Admin can choose to publish schedules either individually (for a single faculty) or universally (for all faculty members). 4) The Admin toggles the publish button for individual or universal publishing. 5) The system will send a notification email to the faculty members about their published schedule. 6) The Admin can choose to print a single faculty's schedule or all faculty schedules for the current academic year. **Program Report** 1) The Admin selects the Generate Program Schedule Report button. 2) The system fetches the program assignments and schedules for the active academic year.

	3) The Admin can view and print the program schedule report. **Room Report** 1) The Admin selects the Generate Room Utilization Report button. 2) The system fetches the room assignments and utilization data for the active academic year and semester. 3) The Admin can view and print the room utilization report. 3. The Admin confirms the publishing of schedules and reports, and the system processes the request. 4. The Faculty, Program, or Room Reports are generated based on the selected options.
Alternative Flow:	1. If the Admin encounters an issue while publishing faculty schedules (e.g., incomplete schedules), the system will notify the Admin and prompt them to complete or correct the schedule before publishing. 2. If the Admin does not want to publish schedules immediately, they can choose to generate reports first and then publish schedules at a later time.
Exceptions:	1. If the Admin cannot access the active academic year or semester data, the system will prompt them to set the active year and semester before proceeding.
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Whenever the Admin needs to set and start academic year and when preparing for course scheduling.
Assumptions:	1. The Faculty Schedule, Program Schedule, and Room Utilization data are up-to-date and ready for publishing. 2. The system is capable of sending email notifications to faculty members regarding their schedules. 3. The Admin has the necessary permissions to generate and publish reports.

Notes and Assumption:	1. The Admin can choose to publish faculty schedules in bulk (universal) or individually. 2. Faculty members will receive email notifications with their specific schedules after the Admin publishes them. 3. The system allows for the generation of printable reports for faculty, programs, and rooms in PDF or similar formats**.**

Table 34. Use Case Report- Official Reports Module

Use Case ID:	UC-11
Use Case Name:	Faculty Dashboard Overview

Actor/ User:	Faculty
Description:	The Faculty can access the Faculty Dashboard to view their set schedule on a calendar interface and receive notifications about upcoming events or changes to their schedule. The dashboard provides an overview of the faculty member's upcoming classes, assignments, and any important announcements or updates.
Trigger:	1. Upon logging into the system with their credentials. 2. Upon accessing the Faculty Dashboard page.
Pre-Conditions:	1. The faculty member must be a registered user in the system with an active role. 2. The faculty member's schedule must be set and available in the system. 3. Notifications about the faculty member's schedule and events must be enabled.
Post-conditions:	1. The faculty member can see the upcoming set schedule displayed on the calendar. 2. The faculty member receives relevant notifications about changes or updates to their schedule. 3. The faculty member can interact with the calendar to view detailed information about each scheduled class.
Normal Flow:	1. The faculty member logs in to the system.

	2. The system redirects them to the Faculty Dashboard. 3. The faculty member sees their schedule on a calendar for the current or upcoming weeks. 4. The calendar shows the date, time, course name, section, and room. 5. The faculty member can click on any event to view more details about the class (e.g., content, location, special instructions). 6. The faculty member can check the notification panel for important updates (e.g., schedule changes, assignments, announcements). 7. They can mark notifications as read or click to view more details and take action, if needed.
Alternative Flow:	1. If there are no updates or changes in the schedule, the notification panel will display a message saying, "No new notifications". 2. If the faculty member clicks on a date or class with no scheduled event, the system will display a message like "No classes scheduled for this day."
Exceptions:	1. If the faculty member does not have any notifications, the notification panel will remain empty or display "No new notifications."
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Frequently, daily or weekly, depending on the faculty member's teaching schedule.
Assumptions:	1. The faculty member's schedule is up-to-date and correctly assigned to the calendar interface. 2. Notifications are configured to be sent to the faculty member based on their set preferences and any schedule updates.
Notes and Assumption:	1. The calendar interface is intuitive and provides clear visuals for easy understanding of the faculty member's schedule. 2. Notifications are triggered when there are schedule changes or relevant messages from the Admin or other relevant sources.

Table 35. Use Case Report- Faculty Dashboard Overview Module

Use Case ID:	UC-12
Use Case Name:	Faculty Preference Submission

Actor/ User:	Faculty
Description:	The Faculty can submit their course preferences within a specified deadline set by the Admin. After the deadline passes, the faculty can only view their submitted preferences and can request changes if needed.
Trigger:	1. Upon logging into the system and accessing the Faculty Preference Submission page. 2. Upon reaching the set deadline for submitting preferences.
Pre-Conditions:	1. The Admin has set a submission deadline for faculty preferences. 2. The faculty member is logged in with an active role in the system.
Post-conditions:	1. The faculty member can submit or view their preferences based on the set deadline. 2. After the deadline, the faculty member can request changes to their preferences.
Normal Flow:	1. The faculty member logs into the system and navigates to the Faculty Preference Submission page. 2. The system displays the available courses and timeslots for submission. 3. The faculty member selects their preferred courses and timeslots and submits them. 4. If the submission is within the deadline, the system confirms the submission. 5. If the deadline has passed, the faculty member can only view their submitted preferences. 6. If needed, the faculty member can request to change their submitted preferences, and the Admin will review the change request.
Alternative Flow:	1. If the faculty member tries to submit

	preferences after the deadline, the system will show a message indicating that the submission period is closed. 2. The faculty member will only have the option to request changes to their preferences.
Exceptions:	1. If the faculty member submits preferences after the deadline, the system prevents further submissions but allows viewing of submitted preferences. 2. If no preferences are submitted before the deadline, the faculty member will see a message stating, "No preferences submitted."
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	At the start of each semester, whenever faculty are required to submit preferences.
Assumptions:	1. The **Admin** sets the deadline correctly and ensures it is visible to faculty before submission. 2. Faculty are aware of the deadline and can submit preferences within the allowed time.
Notes and Assumption:	1. The system will automatically close the submission interface when the deadline has passed. 2. Faculty members can only request changes to their preferences after the submission period ends.

Table 36. Use Case Report- Faculty Preference Submission Module

Use Case ID:	UC-13
Use Case Name:	Faculty Load and Schedule

Actor/ User:	Faculty
Description:	The Faculty can view their schedule once it is published by the Administrator. They can also print their schedule for a specific academic year.
Trigger:	1. Upon the Administrator publishing the faculty schedule. 2. Upon the faculty member accessing the Load

	and Schedule page in the system.
Pre-Conditions:	1. The faculty member is logged into the system. 2. The Administrator has published the faculty schedule. 3. The faculty member is assigned courses for the active academic year.
Post-conditions:	1. The faculty member can view their load schedule for the active academic year. 2. The faculty member can print their schedule.
Normal Flow:	1. The faculty member logs into the system and navigates to the Load and Schedule page. 2. The system shows the faculty member's published schedule for the current or specified academic year. 3. The faculty member can view the details of the schedule, including course names, timeslots, and rooms. 4. The faculty member can select the option to print the schedule. 5. The system generates a printable version of the schedule in a readable format.
Alternative Flow:	1. If the schedule has not been published yet, the system will show a message saying, "Schedule not available." 2. If the faculty member does not have a schedule for the academic year, the system will display a message saying, "No schedule assigned for this year."
Exceptions:	1. If the faculty member tries to print the schedule before it's published, the system will prevent printing and notify the user. 2. If the faculty member tries to view a schedule for an inactive academic year, the system will show an error message saying, "Schedule not available for this academic year."
Includes:	(TO BE UPDATE)
Priority:	High
Frequency of Use:	Regularly, once the schedule is published by the Administrator.
Assumptions:	1. The Administrator publishes the schedule in a timely manner.

	2. The faculty member has the correct permissions to view their schedule.
Notes and Assumption:	1. The faculty member's schedule is up-to-date and correctly assigned by the Administrator. 2. The system supports printing schedules in a clean, formatted layout.

Table 37. Use Case Report- Faculty Load and Schedule Module

3.7 Development Methodology

3.7.1 Process Model

With this project, researchers are using the Agile Methodology to develop the system's needs. The Agile Methodology assists the team in handling changing requirements and delivering high-quality products that meet or exceed our clients' expectations. The first step is the planning phase, during which researchers gather all the needed data related to the project, requirements from the client, and all the data and other elements necessary to start the project. The second phase involves designing, wherein researchers create plans based on the first phase. After designing, the third phase involves development based on the planning and designing phases from both previous stages. Testing occurs after the development phase and includes functional testing, user acceptance testing, and end-to-end testing. Releasing takes place after all the testing succeeds without errors or problems. During this phase, researchers implement the system and accept feedback from the client and target users, marking the beginning of the feedback phase. Using this model in developing a system helps researchers create a more effective and efficient solution for the client's main problems.

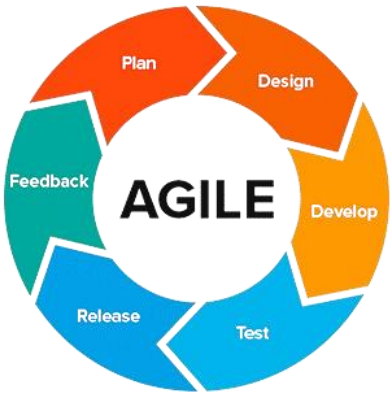


Figure 47. Process Model – Agile Development

3.7.2 Quality Plan

The objective of the quality management plan is to outline the procedures and activities required to ensure the system's quality is met and achieved. In the context of the PUPT Faculty Loading and Scheduling System project, the following steps will be taken to fulfill the client's needs and adhere to system requirements:

- *Review of Documents:* The aim is to verify if the document aligns with the system's functions and requirements. This involves reading the document, checking sources, and ensuring the accuracy of all involved data.

- *Code Review:* This step is intended to identify and rectify oversights made during the initial development phase. The purpose is to prevent the presence of unstable code that could negatively impact the system's processes, fostering continuous improvement.

- *System Testing:* Ensuring that the system meets the specified requirements outlined in the document is the focus of system testing. This step also aims to

confirm that the system operates smoothly as intended. Detailed test cases will be written and executed to assess each feature of the system.

For an accurate evaluation of the system to be developed, researchers will use a survey based on ISO 25010, the International Standard for Software and Data Quality. This standard serves as a guide for creating a high-quality software product.

Below is a list of characteristics to consider when developing quality software:

- Functionality
- Reliability
- Usability
- Efficiency
- Compatibility
- Security
- Maintainability

3.8 Validity and Reliability

The success and credibility of any research study hinge on the robustness of its data, requiring a meticulous examination of both validity and reliability. In this section, we delve into the strategies and measures implemented to ensure the trustworthiness of the study outcomes.

3.8.1 Validity

Internal Validity: To ensure the internal validity of this study, rigorous measures have been implemented to accurately measure the impact of enhancements to the Polytechnic University of the Philippines (PUP) Faculty Loading and Scheduling System. The following steps have been taken:

1. **Alignment with Research Objectives:**

- All data collection instruments, including surveys and automated system metrics, have been meticulously designed to align with the specific research objectives. This ensures that the gathered data directly contributes to addressing the research questions and testing the study hypotheses.

2. **ISO 25010 Framework:**

- The research instrument, particularly the surveys, is structured based on the ISO 25010 standard to evaluate software product quality. By aligning with internationally recognized criteria for software quality, the study enhances the internal validity of the collected data.

3. **Pilot Testing:**

- Prior to full-scale implementation, the research instruments underwent pilot testing with a small group of participants. Feedback from the pilot study facilitated refinement and ensured that the instruments effectively measure the intended constructs.

3.8.2 Reliability

Consistency and Stability: Reliability measures have been implemented to ensure the consistency and stability of the research findings. Key steps taken include:

1. **Structured Surveys:**

- The structured surveys employ established scales and standardized questions, contributing to the reliability of responses. Likert scales, for instance, offer a consistent format for participants to express their opinions.

2. **Automated System Metrics:**

- The use of automated system metrics, such as response time and error rates, ensures objective and consistent measurements of system performance. These metrics contribute to the overall reliability of the technical assessments.

3. **Pilot Testing and Calibration:**

- Pilot testing not only served as a validity check but also allowed for the calibration of the instruments to enhance their reliability. Adjustments were made based on the pilot study outcomes to ensure that the instruments consistently measure the intended variables.

3.8.3 Addressing Potential Bias

Mitigation Strategies: Recognizing the potential for bias in the research process, the study incorporates several strategies to minimize and address bias:

1. **Transparent Communication:**

- Participants are provided with clear and transparent communication regarding the purpose of the study, the voluntary nature of their participation, and the unbiased intent of the research.

2. **Diverse Participant Selection:**

- Purposive sampling is employed to intentionally select participants based on specific roles and expertise. However, efforts have been made to ensure diversity within the chosen group, minimizing the risk of bias in participant perspectives.

3. **Anonymous Data Collection:**

- Responses collected through surveys are anonymized to encourage honest and unbiased feedback. Participants are assured that their responses will be

kept confidential, fostering an environment conducive to open and unbiased input.

4. **Continuous Monitoring:**

- Throughout the data collection process, the research team actively monitors for potential sources of bias. Any identified issues are promptly addressed to maintain the integrity of the study.

By addressing potential sources of bias and implementing measures to enhance both validity and reliability, this study aims to produce robust and trustworthy findings that contribute meaningfully to the understanding of the enhanced PUP Faculty Loading and Scheduling System.

3.9 Development Tools and Technologies

In the pursuit of creating an effective and robust system, careful consideration was given to the selection of development tools and technologies. This section provides a general overview of the overarching technological framework employed in the creation of the PUP Taguig Faculty Loading and Scheduling System. The choices made in this realm were driven by a combination of factors, including project requirements, scalability, maintainability, and alignment with industry best practices.

3.9.1 Software Development Framework

The enhancement of the Polytechnic University of the Philippines (PUP) Faculty Loading and Scheduling System involves the adoption of both the Laravel and Angular frameworks. Laravel, a robust and widely-used PHP framework, is chosen for its versatility, scalability, and adherence to modern software development practices, providing a strong backend foundation. Meanwhile, Angular, a popular front-end framework developed by Google, is selected for its

ability to build dynamic, responsive, and interactive user interfaces. The combination of Laravel for the back-end and Angular for the front-end ensures a seamless integration of the new curriculum, offering a more efficient and user-friendly experience.

3.9.2 Database Management System

The study employs MySQL as the primary database management system for storing and retrieving data related to faculty loading, scheduling, and curriculum information. MySQL is recognized for its reliability, performance, and ease of integration with Laravel, making it a suitable choice for managing the system's database.

3.9.3 Front-End Development Technologies

The user interface (UI) redesign component of the study leverages cutting-edge front-end development technologies to enhance both visual aesthetics and user experience. These technologies are carefully selected to create a responsive, engaging, and intuitive interface. The key front-end development technologies employed in this study include:

1. **HTML5/SCSS:**
    - HTML5 provides the core structure of the user interface, while SCSS (Sassy CSS) is utilized for styling with enhanced capabilities over traditional CSS, offering better organization, reusability, and maintainability. This combination ensures flexibility and responsiveness for a seamless experience across devices.
2. **Angular 18 with Angular Material:**

- Angular 18 is used as the primary framework for building dynamic and robust front-end applications. It allows the development of responsive, scalable, and high-performance web applications. Angular Material, a UI component library for Angular, is integrated to create visually appealing and consistent user interfaces, adhering to modern design standards and enhancing user experience with pre-built, responsive UI components.

3. **JavaScript (TypeScript):**

   - JavaScript, specifically TypeScript (the recommended language for Angular), is employed to implement dynamic and interactive elements within the user interface. This ensures a modern, efficient, and user-friendly design.

4. **NPM (Node Package Manager):**

   - NPM, a modern library and package manager for JavaScript, is used to manage dependencies and streamline the development process. It provides access to a wide range of libraries and tools that help in optimizing performance and adding advanced functionality to the system.

3.9.4 Version Control and Collaboration

Git is employed as the version control system to manage the source code and facilitate collaborative development. The use of Git allows for efficient tracking of changes, collaboration among team members, and the ability to roll back to previous versions if needed.

3.9.5 Project Management

For effective project management and collaboration, the study utilizes Notion. Notion provides a comprehensive and flexible platform for project planning, task management, and collaboration, enabling the development team to organize work seamlessly and enhance communication.

3.9.6 Hosting and Deployment

Hostinger is selected as the hosting and deployment platform for the enhanced system. Hostinger offers reliable and cost-effective hosting solutions, ensuring the availability and accessibility of the system for end-users. The deployment process is streamlined through Hostinger's infrastructure, contributing to a smooth and efficient release of updates.

3.9.7 Security Measures

Security is a paramount consideration in the development process. The study incorporates industry-standard security practices and tools to ensure the protection of sensitive data and system integrity. Key security measures include:

1. **SSL/TLS Encryption:**
    - Secure Sockets Layer (SSL) and Transport Layer Security (TLS) encryption are implemented to secure data transmission between the user interface and the server, ensuring that all communication remains private and tamper-proof.

2. **Testing of Hosting Environment:**

   - The hosting environment undergoes rigorous security testing using OWASP ZAP (Open Web Application Security Project Zed Attack Proxy) to identify vulnerabilities, such as cross-site scripting (XSS) and SQL injection, ensuring the system is protected from common web application attacks.
   - Qualys, a leading vulnerability management tool, is used to conduct in-depth scans of the server and network environment to detect potential weaknesses and ensure compliance with security standards.

3.9.8 Testing Tools

A variety of testing tools are employed to ensure the system's quality and reliability. PHPUnit is used for unit testing in Laravel, while Laravel Dusk handles browser testing to simulate user interactions. Postman is utilized for API testing to verify secure and accurate data transmission. For front-end testing in Angular, Jasmine and Karma are used together, with Jasmine providing a behavior-driven framework and Karma automating test execution across browsers. These tools work together to ensure comprehensive testing of both the back-end and front-end components.

CHAPTER 4

RESULTS AND DISCUSSIONS

This chapter discusses the results of the testing, quality, and evaluation plan for the PUP Taguig Faculty Loading and Scheduling System. The research aims to develop and deploy a system that effectively manages faculty scheduling and loading processes. The researchers based the evaluation on the feedback gathered from the total number of users participated in the testing phase, which included faculty members, the Director of the School, and the Head of Academic Programs.

The user acceptance test questions were based on the ISO 25010 quality plan, which compromises the following criteria: Functional Suitability, Reliability, Performance Efficiency, Usability, Security, Compatibility, Maintainability, and Portability. These characteristics were evaluated by the respondents using a series of designed survey questions. To analyze the outcomes of the system performance during the testing phase, the researchers used the following approach for Central Tendency or Mean computation.

Central Tendency/Mean

The mean is the average of all the survey responses. It is calculated by adding up the scores given by respondents using the Likert Scale and dividing by the total number of responses. The formula for obtaining the mean is as follows:

Equation 1. Formula for Mean

$$mean = \frac{sum\ of data\ /\ responses\ gathered}{total\ number\ of\ responses\ gathered}$$

Likert Scale

NUMERICAL RATING	RANGE OF MEAN SCORE	INTERPRETATION
1	1.00 – 1.79	Strongly Disagree
2	1.80 – 2.59	Disagree
3	2.60 – 3.39	Neutral
4	3.40 – 4.19	Agree
5	4.20 – 5.00	Strongly Agree

Table 38. Likert Scale Numerical Rating and Interpretation

To assess the quality and performance of the PUP Taguig Faculty Loading and Scheduling System, test procedures were conducted using the ISO 25010 model. The test findings are as follows:

A. Functional Suitability – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the functional suitability testing.

A. FUNCTIONAL SUITABILITY			
Indicators		**Mean**	**Verbal Interpretation**
Completeness	**PUPT Faculty Loading and Scheduling System meets all specified task and user objective** *(Natutugunan ng PUPT Faculty Loading and*	4.58	Strongly Agree

	Scheduling System ang lahat ng tinutukoy na mga gawain at mga layunin ng gumagamit.)		
Correctness	**PUPT Faculty Loading and Scheduling System provides the correct results with the needed level of precision.** *(Nagbibigay ang PUPT Faculty Loading and Scheduling System ng tamang resulta sa bawat aksyon na ginagawa ng gumagamit.)*	4.41	Strongly Agree
Appropriateness	**PUPT Faculty Loading and Scheduling System facilitates the accomplishment of specified tasks and objectives.** *(Napapadali ng PUPT Faculty Loading and Scheduling System ang pagkamit ng bawat gawain ng gumagamit.)*	4.58	Strongly Agree

Overall Mean	**4.52**	**Strongly Agree**

Table 39. Functional Suitability Testing of the PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a very positive outlook on the Functional Suitability of the system. The survey results show that the respondents strongly agree on the Completeness (4.58), Correctness (4.41), and Appropriateness (4.58) of the system. This demonstrates that the system meets all its objectives, delivers accurate results, and facilitates the accomplishment of tasks effectively. Overall, the mean score for the system's Functional Suitability is 4.52, which corresponds to a "Strongly Agree" rating.

B. Performance Efficiency – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the functional suitability testing.

B. PERFORMANCE EFFICIENCY			
Indicators		**Mean**	**Verbal Interpretation**
Time Behavior	**When performing its functions, PUPT Faculty Loading and Scheduling System meets the requirements of the response, processing times.**	4.58	Strongly Agree

	(Kapag isinasagawa ang gawain, natutugunan ng PUPT Faculty Loading and Scheduling System ang mga kinakailangan sa bilis ng pagtugon at oras ng pagproseso.)		
Resource Utilization	**When performing its functions, PUPT Faculty Loading and Scheduling System meets the requirements of the amount types of resource used.** *(Kapag isinasagawa ang gawain, natutugunan ng PUPT Faculty Loading and Scheduling System ang mga kinakailangan ng mga halaga at uri ng mga mapagkukunan na gamit.)*	4.50	Strongly Agree
Capacity	**PUPT Faculty Loading and Scheduling System parameter**	4.66	Strongly Agree

	limit meets the requirements. *(Ang limitasyong parameter ng PUPT Faculty Loading and Scheduling System ay nakakatugon sa mga kinakailangan.)*		
Overall Mean		**4.58**	**Strongly Agree**

Table 40. Performance Efficiency Testing of the PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a very positive perspective on the system's Performance Efficiency. The survey results reveal that the respondents strongly agree on the Time Behavior (4.58), Resource Utilization (4.50), and Capacity (4.66) of the system. This highlights that the system performs tasks efficiently within the expected time frame, optimally utilizes resources, and handles the required capacity effectively. Overall, the mean score for the system's Performance Efficiency is 4.58, reflecting a "Strongly Agree" rating.

C. Compatibility – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the compatibility testing.

C. COMPATIBILITY		
Indicators	**Mean**	**Verbal Interpretation**

Co-existence	**PUPT Faculty Loading and Scheduling System can perform its required functions efficiently while sharing common environment and resources without detrimental impact.** *(Nagagawa pa rin ng maayos ng PUPT Faculty Loading and Scheduling System ang mga tungkulin nito kahit na may pagbabahagi ng nagaganap sa iba't ibang environment at mapagkukunan ng walang masamang epekto.)*	4.24	Strongly Agree
Interoperability	**PUPT Faculty Loading and Scheduling System can exchange information and use the information that has been exchanged.** *(May kakayahan ang PUPT Faculty Loading and Scheduling System*	4.5	Strongly Agree

	na makipag palit ng impormasyon at gamitin ang mga impormasyon pinagpalit.)		
Overall Mean		**4.37**	**Strongly Agree**

Table 41. Compatibility Testing of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a favorable perspective on the system's Compatibility. The survey results show that the respondents strongly agree on the Co-existence (4.24) and Interoperability (4.50) of the system. This demonstrates that the system seamlessly integrates and functions alongside other systems while maintaining effective communication and compatibility. Overall, the mean score for the system's Compatibility is 4.37, which corresponds to a "Strongly Agree" rating.

D. Usability – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the usability testing.

D. USABILITY			
Indicators		**Mean**	**Verbal Interpretation**
Appropriateness Recognizability	**Users can recognize whether the PUPT Faculty Loading and Scheduling System is**	4.92	Strongly Agree

	appropriate for their needs. *(Malalaman ng mga gumagamit kung ang PUPT Faculty Loading and Scheduling System ay angkop para sa kanilang mga pangangailangan. oras ng pagproseso.)*		
Learnability	**PUPT Faculty Loading and Scheduling System can be used by specific users to achieve specific goals of learning to use the application with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.** *(Ang PUPT Faculty Loading and Scheduling System ay maaaring gamitin ng mga partikular na user upang makamit ang*	4.83	Strongly Agree

	mga partikular na layunin ng pag-aaral na gamitin ang application ng may bisa, kahusayan sa kalayaan mula sa panganib at kaluguran sa isang partikular na konteksto ng paggamit.)		
Operability	**PUPT Faculty Loading and Scheduling System has attributes that make it easy to operate and control.** *(Ang PUPT Faculty Loading and Scheduling System ay may katangian na nagpapadali sa pagpapatakbo at pagkontrol.)*	3.5	Neutral
User Error Protection	**The PUPT Faculty Loading and Scheduling System protects users against making errors.** (Pinoprotektahan ng PUPT Faculty Loading	4.5	Strongly Agree

	and Scheduling System ang mga gumagamit laban sa paggawa ng mga pagkakamali.)		
User Interface Aesthetics	**The user interface of PUPT Faculty Loading and Scheduling System enables pleasing and satisfying interaction for the user.** *(Ang user interface ng PUPT Faculty Loading and Scheduling System ay nagbibigay daan sa kasiya-siya at kalugod lugod na pakikipag-ugnayan para sa user.)*	5.17	Strongly Agree
Accessibility	**PUPT Faculty Loading and Scheduling System can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.**	4.83	Strongly Agree

	(Ang PUPT Faculty Loading and Scheduling System ay maaaring gamitin ng mga taong may pinakamalawak na hanay ng mga katangian at kakayahan upang makamit ang tinutukoy na layunin sa isang tinutukoy na konteksto ng paggamit.)		
Overall Mean		**4.62**	**Strongly Agree**

Table 42. Usability Testing of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a highly favorable outlook on the system's Usability. The survey results reveal strong agreement on various aspects, including Appropriateness Recognizability (4.92), Learnability (4.83), User Error Protection (4.50), User Interface Aesthetics (5.17), and Accessibility (4.83). Although Operability received a slightly lower rating of 3.50, the system's overall usability was rated positively. These findings demonstrate that the system is generally user-friendly, visually appealing, and accessible, with minimal errors encountered by users. The overall mean score for the system's Usability is 4.62, reflecting a "Strongly Agree" rating.

E. Reliability – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the reliability testing.

E. RELIABILITY			
Indicators		**Mean**	**Verbal Interpretation**
Maturity	**PUPT Faculty Loading and Scheduling System meets the needs for reliability under normal operation.** *(Natutugunan ng PUPT Faculty Loading and Scheduling System ang mga pangangailangan para sa pagiging maaasahan sa ilalim ng normal na operasyon.)*	4.42	Strongly Agree
Availability	**PUPT Faculty Loading and Scheduling System is operational and accessible when required for use.** *(Ang PUPT Faculty Loading and Scheduling System ay gumagana at*	4.67	Strongly Agree

	nagagamit kapag kinakailangan gamitin.)		
Fault Tolerance	**PUPT Faculty Loading and Scheduling System operates as intended despite the presence of hardware or software faults.** *(Gumagana ang PUPT Faculty Loading and Scheduling System ayon sa layunin sa kabila ng pagkakaroon ng mga pagkakamali sa hardware o software.)*	4.42	Strongly Agree
Recoverability	**PUPT Faculty Loading and Scheduling System, in the event of an interruption or failure, can recover the data directly affected and re-establish the desired state of the system.** *(Ang PUPT Faculty Loading and Scheduling System, kaganapan ng pagkaantala o*	4.42	Strongly Agree

	pagkabigo, ay maaaring mabawi ang datos na direktang apektado at muling itatag ang nais na estado ng system.)		
Overall Mean		**4.48**	**Strongly Agree**

Table 43. Reliability Testing of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a positive perspective on the system's Reliability. The survey results show strong agreement on key aspects, including Maturity (4.42), Availability (4.67), Fault Tolerance (4.42), and Recoverability (4.42). These findings demonstrate that the system is dependable, consistently available, and capable of handling faults effectively while ensuring swift recovery. The overall mean score for the system's Reliability is 4.48, reflecting a "Strongly Agree" rating.

F. Security – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the security testing.

F. SECURITY			
Indicators		**Mean**	**Verbal Interpretation**
Confidentiality	**PUPT Faculty Loading and Scheduling System ensures that**	4.33	Strongly Agree

	data are accessible only to those authorized to have access. *(Tinitiyak ng PUPT Faculty Loading and Scheduling System na ang datos ay magagamit lamang ng mga awtorisadong tao.)*		
Integrity	**PUPT Faculty Loading and Scheduling System stops unauthorized users from accessing or changing computer programs or data.** *(Hinahadlangan ng PUPT Faculty Loading and Scheduling System ang hindi awtorisadong paggamit o pagbabago ng mga computer program o datos.)*	4.33	Strongly Agree
Non-Repudiation	**PUPT Faculty Loading and Scheduling System can be used to demonstrate its validity, preventing a**	4.33	Strongly Agree

	later rejection of the activities or occurrences. *(Ang PUPT Faculty Loading and Scheduling System ay maaaring gamitin upang ipakita ang pagka-epektibo nito at maiwasan ang rejection sa mga aktibidad o mga pangyayari.)*		
Overall Mean		**4.33**	**Strongly Agree**

Table 44. Security Testing of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a favorable outlook on the system's Security. The survey results show strong agreement on aspects such as Confidentiality (4.33), Integrity (4.33), and Non-repudiation (4.33). These findings highlight that the system effectively ensures the confidentiality of data, maintains data integrity, and provides accountability in operations. The overall mean score for the system's Security is 4.33, reflecting a "Strongly Agree" rating.

G. Maintainability – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the maintainability testing.

G. MAINTAINABILITY

Indicators		Mean	Verbal Interpretation
Modularity	**PUPT Faculty Loading and Scheduling System can be improved without producing faults or decreasing the quality of the result.** *(Ang PUPT Faculty Loading and Scheduling System ay maaaring mapabuti nang walang pagkakamali o pagbabawas sa kalidad ng resulta.)*	4.17	Agree
Modifiability	**PUPT Faculty Loading and Scheduling System can change one element while having little impact on the other elements.** *(Ang PUPT Faculty Loading and Scheduling System ay maaaring mag-bago ng isang elemento nang hindi*	4.67	Strongly Agree

	nakakaimpluwensya sa iba pang mga elemento sa system.)		
Overall Mean		**4.42**	**Strongly Agree**

Table 45. Maintainability Testing of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a positive perspective on the system's Maintainability. The survey results reveal strong agreement on aspects such as Modularity (4.17) and Modifiability (4.67). These findings demonstrate that the system is designed with well-structured components, making it easier to modify and maintain as needed. The overall mean score for the system's Maintainability is 4.42, reflecting a "Strongly Agree" rating.

H. Portability – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the portability testing.

H. PORTABILITY			
Indicators		**Mean**	**Verbal Interpretation**
Adaptability	**The PUPT Faculty Loading and Scheduling System can be moved to other environments.** (*Ang PUPT Faculty Loading and*	4.58	Strongly Agree

	Scheduling System ay maaaring ilipat sa ibang mga lugar.)		
Replaceability	**When performing its functions, PUPT Faculty Loading and Scheduling System meets the requirements of the amount types of resource used.** *(Kapag isinasagawa ang gawain, natutugunan ng PUPT Faculty Loading and Scheduling System ang mga kinakailangan ng mga halaga at uri ng mga mapagkukunan na gamit.)*	4.67	Strongly Agree
Overall Mean		**4.62**	**Strongly Agree**

Table 46. Portability Testing of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a highly favorable outlook on the system's Portability. The survey results show strong agreement on aspects such as Adaptability (4.58) and Replaceability (4.67). These findings demonstrate that the system is capable of adapting to different environments and can be replaced or updated with ease.

The overall mean score for the system's Portability is 4.62, reflecting a "Strongly Agree" rating.

I. Overall – A total of 12 faculty members, each with different roles, used the PUP Taguig Faculty Loading and Scheduling System and participated in the overall testing.

I. OVERALL		
Indicators	**Mean**	**Verbal Interpretation**
I am satisfied with the performance of the PUPT Faculty Loading and Scheduling System. *(Ako ay kuntento sa performance ng PUPT Faculty Loading and Scheduling System.)*	4.08	Agree
The PUPT Faculty Loading and Scheduling System helped eliminate problems that are caused by templated inputs and is a good replacement for the current system. (*Nakatulong ang PUPT Faculty Loading and Scheduling System na alisin ang mga problemang dulot ng mga naka-template na input at isang magandang pamalit para sa kasalukuyang sistema.)*	4.58	Strongly Agree
Overall Mean	**4.33**	**Strongly Agree**

Table 47. Overall Assessment of PUPT FLSS

The results gathered in the survey, as shown in the table above, indicate that the respondents had a positive overall perception of the PUPT Faculty

Loading and Scheduling System. The respondents agreed that they are satisfied with the system's performance, as reflected in the mean score of 4.08. Additionally, they strongly agreed that the system effectively eliminated problems caused by templated inputs and serves as a good replacement for the previous system, with a mean score of 4.58. The overall mean score for the system is 4.33, reflecting a "Strongly Agree" rating.

OVERALL RESULT		
Description	**Mean**	**Verbal Interpretation**
ISO 25010		
FUNCTIONAL SUSTAINABILITY	4.52	Strongly Agree
PERFORMANCE EFFICIENCY	4.58	Strongly Agree
COMPATIBILITY	4.37	Strongly Agree
USABILITY	4.62	Strongly Agree
RELIABILITY	4.48	Strongly Agree
SECURITY	4.38	Strongly Agree
MAINTAINABILITY	4.42	Strongly Agree
PORTABILITY	4.62	Strongly Agree
OVERALL MEAN	**4.50**	**Strongly Agree**

Table 48. Overall Result of the PUPT FLSS

In summary, the overall evaluation of the PUPT Faculty Loading and Scheduling System, based on ISO 25010 quality characteristics, indicates a highly positive assessment. The system achieved "Strongly Agree" ratings across all criteria, demonstrating exceptional performance in Functional Sustainability

(4.52), Performance Efficiency (4.58), Compatibility (4.37), Usability (4.62), Reliability (4.48), Security (4.38), Maintainability (4.42), and Portability (4.62). The system's ability to meet objectives, operate efficiently, integrate seamlessly, provide a user-friendly experience, ensure data security, and adapt to different environments highlights its effectiveness and dependability. With an overall mean score of 4.49, the system has proven to be a reliable and efficient replacement for the previous process.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

Summary of Findings

This study aimed to streamline the manual faculty loading and scheduling processes at the Polytechnic University of the Philippines Taguig Campus by implementing an automated Faculty Loading and Scheduling System (FLSS). The system has significantly improved efficiency, productivity, data accuracy, and accessibility in managing faculty resources and scheduling tasks. Feedback from 20 participants, including administrators, faculty, and superadmin, highlighted its positive impact on workflow and organization. The FLSS also reduced the time required to complete scheduling tasks, enabling administrators to focus on more strategic responsibilities. Its user-friendly interface ensures that even non-technical users can navigate the system with ease. Overall, the FLSS demonstrates the potential of automation to enhance resource management in academic institutions.

Testing Results

The researchers were able to determine the respondents' points of view and feedback about the Faculty Loading and Scheduling System- Taguig (FLSS) using the ISO 25010 Software Quality Characteristics as a framework. Functional sustainability received an average mean of 4.52, followed by performance efficiency with an average of 4.58, compatibility with 4.37, usability and portability with 4.62 each, reliability with 4.48, security with 4.38, and maintainability with 4.42. The overall assessment of the system yielded an average mean score of 4.33, indicating strong approval for the system's implementation. These results highlight the system's effectiveness and alignment with quality standards.

Conclusion

The evaluation of the Faculty Loading and Scheduling System (FLSS) demonstrates its effectiveness in streamlining academic operations and addressing the needs of its users. The system efficiently detects scheduling conflicts, allowing faculty members to resolve them promptly. It also enables faculty to set their preferences with ease and provides timely access to their schedules, enhancing productivity and satisfaction.

The system offers great potential for enhancement in the near future through the integration of emerging technologies. Features such as artificial intelligence and machine learning could be utilized to provide predictive analytics for schedule optimization and conflict resolution. Additionally, incorporating cloud-based platforms and mobile applications would ensure real-time access and greater flexibility for users. These advancements would further improve the system's functionality, user experience, and adaptability to evolving institutional requirements.

The FLSS has proven to be a significant enhancement to academic operations, ensuring an organized and conflict-free scheduling process while prioritizing user convenience. Its ability to streamline tasks, provide timely information, and support faculty preferences highlights its value as an essential tool for institutional management. Moving forward, continuous improvements and user feedback integration will further enhance its capabilities, ensuring its sustainability and adaptability to future requirements. Ultimately, the system demonstrates its potential to contribute to a more efficient and well-coordinated academic environment.

Recommendations:

REFERENCES

Abed, M. (2021). Development of a faculty workload management system using optimization algorithms. *International Journal of Scientific Research in Computer Science and Engineering,* 10*(*2), 327-333.

Baccay, J., & Cabahug, N. (2019). Class scheduling and faculty loading system using optimization algorithm. *International Journal of Advanced Trends in Computer Science and Engineering,* 9*(*4), 4243-4248.

Nuengwong, Tuaycharoen., Veerawat, Prodpran., Boonsong, Srithong. (2018). ClassSchedule: A web-based application for school class scheduling with real-time lazy loading. *IEEE International Conference on Business and Information Research (ICBIR)*, 210-214. doi: 10.1109/ICBIR.2018.8391194

Louie, F., Agustin, J. K. A., Losito, A. R. Mendoza, A. T., Factoriza, A. (2018). Predictive Analytics Implementing Genetic Algorithm on Class Scheduling System. *International Journal of Advanced Research in Computer Science,* 9(6), 18-22. doi: 10.26483/IJARCS.V9I6.6333

Botangen, K. A. (2016). Class-Scheduling System for the Central Luzon State University [Abridged]. Retrieved from ResearchGate website: (PDF) Class-Scheduling System for the Central Luzon State University

Miranda, J. (2010). eClasSkeduler: *A Course Scheduling System For The Executive Education Unit At The Universidad De Chile*. Retrieved on January 13, 2015 from http://goo.gl/6ljMn

Etcuban, J. O. (2013). Automated test generator for the faculty of the University of Cebu, Philippines. *IAMURE International Journal of Mathematics, Engineering & Technology*, 6, 30. Retrieved on March 2015 from http:// goo.gl/aLp1Ry

Ortega, E. P., & Brigoli, D. I. (2015). *Online Class Scheduling and Faculty Loading System within a Decision Support Framework*. JPAIR Institutional Research, 6(1), 61-77.

Chaiwchan, W., & Klinhom, P. (2014). *The Development of Online-Class Scheduling Management System Conducted by the Case Study of Department of Social Science: Faculty of Humanities and Social Sciences Suan Sunandha Rajabhat University.* International Journal of Computer and Information Engineering, 8(8), 2740-2744.