Group 17

Report

Postgraduate Student Management System

Table of Content

[Introduction 4](#_Toc183905323)

[Introduction 4](#_Toc183905324)

[Objectives of the System 4](#_Toc183905325)

[Scope of the Project 5](#_Toc183905326)

[Analysis 6](#_Toc183905327)

[Requirements Gathering Techniques 6](#_Toc183905328)

[Analyzing the Current system and Processes 7](#_Toc183905329)

[Student Registration Process 7](#_Toc183905330)

[Research progress tracking Process 7](#_Toc183905331)

[Reports/Thesis Evaluating Process 7](#_Toc183905332)

[Functional Requirements of the System 8](#_Toc183905333)

[User Management 8](#_Toc183905334)

[Student Enrollment 8](#_Toc183905335)

[Automated Email Notifications 8](#_Toc183905336)

[Role-Based Access Control 9](#_Toc183905337)

[Authentication and Security 9](#_Toc183905338)

[Non-functional Requirements of the system 9](#_Toc183905339)

[Accuracy 9](#_Toc183905340)

[Security 9](#_Toc183905341)

[User-friendliness 9](#_Toc183905342)

[Maintainability 10](#_Toc183905343)

[Interoperability 10](#_Toc183905344)

[Overview of Software Process Models 11](#_Toc183905345)

[Design 12](#_Toc183905346)

[How design started 12](#_Toc183905347)

[Implementing the System as a Web Based Application 12](#_Toc183905348)

[System analysis and Design 14](#_Toc183905349)

[Use case diagram 14](#_Toc183905350)

[EER diagram 18](#_Toc183905351)

[UML Class diagram 19](#_Toc183905352)

[Sequence diagram 20](#_Toc183905353)

[System design methodologies 24](#_Toc183905354)

[Database Design for the system 25](#_Toc183905355)

[Project Management and Version Control 27](#_Toc183905356)

[Implementation 30](#_Toc183905357)

[System Development Tools and Technologies 30](#_Toc183905358)

[Spring Boot 30](#_Toc183905359)

[Angular 30](#_Toc183905360)

[MySQL Workbench 30](#_Toc183905361)

[Major Code Structure 31](#_Toc183905362)

[User logging handling 31](#_Toc183905363)

[Session management 32](#_Toc183905364)

[Database Connection Session 33](#_Toc183905365)

[Module Structure of the System 34](#_Toc183905366)

[Testing and Bug fixing 38](#_Toc183905367)

[Software Testing vs Software Evaluation 38](#_Toc183905368)

[Software Testing Process 38](#_Toc183905369)

[Software Evaluation 39](#_Toc183905370)

[System Test Plan 39](#_Toc183905371)

[System Test Cases 41](#_Toc183905372)

[User Manual 44](#_Toc183905373)

[Conclusion 54](#_Toc183905374)

[Challenges faced 54](#_Toc183905375)

[Feedback 54](#_Toc183905376)

[Work Task Division and Individual Contribution 55](#_Toc183905377)

[Lessons learnt During the Course of the Project 57](#_Toc183905378)

[Declaration 57](#_Toc183905379)

[References 58](#_Toc183905380)

# Introduction

## Introduction

The **Postgraduate Students Management System (PGSMS)** is a comprehensive platform designed to efficiently manage the academic and administrative processes related to MPhil. and PhD. students in the Department of Computer Engineering at the University of Peradeniya. This system addresses the challenges associated with traditional, manual processes by introducing streamlined, automated workflows that enhance operational efficiency, improve communication, and ensure effective monitoring of students’ academic progress.

The system is tailored to meet the specific needs of postgraduate programs, focusing on the lifecycle of a student’s academic journey, from registration to graduation.

## Objectives of the System

The primary objective of the PGSMS is to simplify and automate the administrative and academic process associated with the postgraduate studies in the department of computer engineering. Key goals include:

1. Streamline Enrollment Processes: Enable students to submit their enrollment details online and facilitate efficient review by administrators.
2. Improving Academic Monitoring: Provide an easier way to manage student evaluations, track submissions, and assign examiners seamlessly.
3. Reminding Important Submissions: Enable reminders for the important submissions through features like automated email notifications.
4. Promoting Administrative Efficiency: Reduce manual effort in managing postgraduate activities and focus on strategic tasks.

## Scope of the Project

The scope of the PGSMS is to provide a centralized platform catering to the administrative and academic needs of postgraduate programs. It covers up:

* User Management
  + Admins can manage student and staff profiles.
  + Role-based functionalities ensure access is restricted to authorized users.
* Enrollment System
  + Students can submit their enroll details online.
  + Admins can review those details.
* Submission and Evaluation Management
  + Submission portals for students to upload half-year and year-end reports.
  + Examiners can provide detailed feedback and evaluations.
* Examiner Assignment:
  + Admins can assign examiners for student submissions.
  + The system checks and prevents duplicate or conflicting assignments.
* Communication System
  + Automated email notifications for submission status, reminders, and evaluation feedback.
  + Template for emails, customizable by the admin, ensures consistency.

# Analysis

## Requirements Gathering Techniques

Requirement gathering is a critical phase in the project, as it lays the foundation for what needs to be built. For our project, a combination of direct communication with stakeholders and document analysis was used, alongside knowledge of other standard techniques that could be applied in similar scenarios.

Face-to-face meetings and discussions with supervisors and lecturers played a key role in understanding the project’s scope and objectives. These interactions provided an opportunity to clarify expectations, ask questions, and gather insights directly from stakeholders. Such conversations are invaluable for gaining a comprehensive understanding of both the explicit and implicit requirements.

A detailed PDF document containing the project requirements was provided. This served as a formal reference point for the design and development of the system. Document analysis involved thoroughly reviewing this file to extract functional, non-functional, and technical requirements, ensuring alignment with the stakeholders’ vision.

Also, the previously used system named [CERPS](https://eng.pdn.ac.lk/cerpsnew/index.php) shows the requirements of the system clearly. By treating [CERPS](https://eng.pdn.ac.lk/cerpsnew/index.php) as both a benchmark and a learning opportunity, the new system was designed not only to meet existing requirements but also to anticipate future needs, ensuring its relevance and usability over time.

## Analyzing the Current system and Processes

### Student Registration Process

The current enrollment process happens manually. Students can download the Registration forms from the [CERPS](https://eng.pdn.ac.lk/cerpsnew/index.php) website and email them to the department. That’s the whole process of registration in the current system.

### Research progress tracking Process

Coordinator and Supervisors communicated through emails to track students’ progress. Students had to visit the [CERPS](https://eng.pdn.ac.lk/cerpsnew/index.php) website and download necessary documents like report templates and email them before the deadline.

This process placed the burden of organization and adherence to timelines largely on the students and supervisors, often leading to challenges in maintaining a clear and centralized record of submissions and feedback.

### Reports/Thesis Evaluating Process

This process was another manual and time-intensive task that posed challenges for coordinators. After collecting the reports or theses from students, coordinators were responsible for distributing these documents to examiners via email for evaluation. This process not only required significant effort to ensure that all submissions were accurately tracked and forwarded to the correct examiners but also introduced the risk of human errors, such as missed emails or incorrect attachments. Additionally, there was no centralized platform for examiners to provide feedback or for students to track the status of their evaluations, resulting in inefficiencies and delays in communication. The lack of automation and a unified system often made the evaluation process cumbersome and prone to inconsistencies.

## Functional Requirements of the System

Functional requirements define the functions that the system or its components are supposed to perform. They describe the intended behavior and outputs of the system based on various inputs.

The following section describes the functional requirements that have been identified during the system analysis and the detailed specifications of the functionalities that the users expect from the proposed system.

### User Management

* The system must allow admin to add users to the system.
* Each user must be assigned specific roles (ex: admin, student, examiner) to determine the access privileges.
* Students have to register using the online enrollment form.
* Enrolled students and newly added staff members must receive email notifications with the login credentials.

### Student Enrollment

* The system has to provide a user-friendly form for the students to submit personal details and academic qualifications.
* Admins must be able to review and enroll the students.
* The system must store student profiles with details.

Progress Monitoring

* Admins must be able to create evaluation links for half-yearly and yearly reviews.
* Students must upload progress reports and other required submissions through the given links.
* Examiners must be able to review submissions and provide feedback.
* The system must maintain a history of all submissions and feedback for each student.

Thesis and Publication Management

* Students must upload their thesis and other submissions.
* Supervisors and examiners must access these documents to provide feedback.
* Notifications must be sent to relevant stakeholders upon submission and feedback.

### Automated Email Notifications

* Admin must be able to create or edit email templates.
* The system must enable staff to send emails to students and other stakeholders using pre-defined templates.

### Role-Based Access Control

* Admins must have access to all system functionalities.
* Students must only access their profile, submissions and feedback they have received.
* Supervisors must track the progress of students under their guidance.
* Examiners must access assigned student submissions and provide feedback.

### Authentication and Security

* The system must restrict access to sensitive data based on user roles.
* Passwords must be encrypted before storage.

## Non-functional Requirements of the system

The definition for a non-functional requirement is that it essentially specifies how the system should behave and that it is a constraint upon the system’s behavior. Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements.

The following section explains the non-functional requirements that should be on when implementing the proposed Students Management System.

### Accuracy

Since the proposed system intends to deal with the students’ submissions and the evaluations of them, that information should be accurate and trustworthy.

### Security

Since the system contains students’ submissions and their evaluation results, securing such information will be one of the major non-functional requirements that should be focused. Only the authorized people should be able to access the system using usernames and passwords. A mechanism for encrypting the passwords should also be implemented in order to secure the user accounts that registered in the system. Proper access privileges should be defined for the different use roles.

### User-friendliness

The overall functionalities of the system should be implemented in a way that is easy to use for all the user levels who take benefits of the system.

### Maintainability

The codebase must follow clean coding practices and be well-documented to facilitate future updates. Modular architecture must ensure that individual components can be updated or replaced without affecting other parts of the codebase.

### Interoperability

The system must be compatible with standard browsers such as Google Chrome, Firefox and Microsoft Edge.

## Overview of Software Process Models

The decision to select the most suitable software process model for developing the intended software is usually considered as one of the most challenging decisions any software engineer has to make.

That is mainly because the success factor of the developed software mostly depends on the decision to select the most suitable software process model. Therefore, it’s required to perform a comprehensive analysis about the ‘Requirements of the project’ before taking the final decision on choosing among the structured methodologies and object-oriented methodologies for software development.

**Development Approach**

The project follows the Agile methodology, ensuring flexibility throughout the development process. The agile principle allowed the team to break the task into iterative sprints. In those sprints features were implemented, tested and refined. Regular meetings with the team and the supervisor ensured that requirements were addressed closely with the user needs.

**Architecture**

The system was built using a **monolithic architecture**, integrating all components into a single cohesive application. This design was chosen to simplify development, deployment, and maintenance, particularly given the system's small-to-medium scale requirements

# Design

## How design started

The initial focus was on understanding the needs of system users; students, administrators, supervisors, and examiners and identifying the challenges they face in their day-to-day operations. The design process was guided by the following principles:

Understanding User Roles:

We began by defining the key roles in the system. Each role has specific needs and responsibilities, which informed the functionality and user interface required. For example, students need to access their data effortlessly, while administrators require tools for managing profiles and research tasks.

Clarity and Accessibility:

The design aimed to make the system easy to navigate, ensuring that users could find what they needed without unnecessary complexity. This user-centric approach was critical to ensuring the system would be adopted effectively.

Collaboration and Feedback:

Input from supervisor-administration and team members-was integral in shaping the design. Initial drafts and mockups were shared to gather insights, which helped refine the structure and flow of the system.

## Implementing the System as a Web Based Application

The decision to implement the system as a web-based application was driven by its ability to offer accessibility and convenience.

Global Access:

A web-based application allows users to interact with the system from anywhere, as long as they have an internet connection. This aligns with the modern need for flexibility in accessing data and managing tasks remotely.

No Installation Hassle:

Unlike desktop software, which requires installation and compatibility checks, a web application only requires a browser. This reduces the burden on users and makes updates seamless.

Streamlined User Experience:

A web-based approach enables consistent experiences across devices. Whether users are accessing the system on a desktop, tablet, or smartphone, they can perform their tasks with ease.

Centralized Management:

By being web-based, the system allows centralized data storage and management. Administrators can update records, make changes, or introduce new features without requiring users to download updates or patches.

Also, as a **Web Application**, it offers several advantages over traditional websites, making them a preferred choice for interactive and dynamic user experiences. Unlike static websites, which primarily display information, web applications allow users to perform specific tasks, interact with data, and engage in real-time activities. This interactive capability makes web applications more versatile and user-friendly.

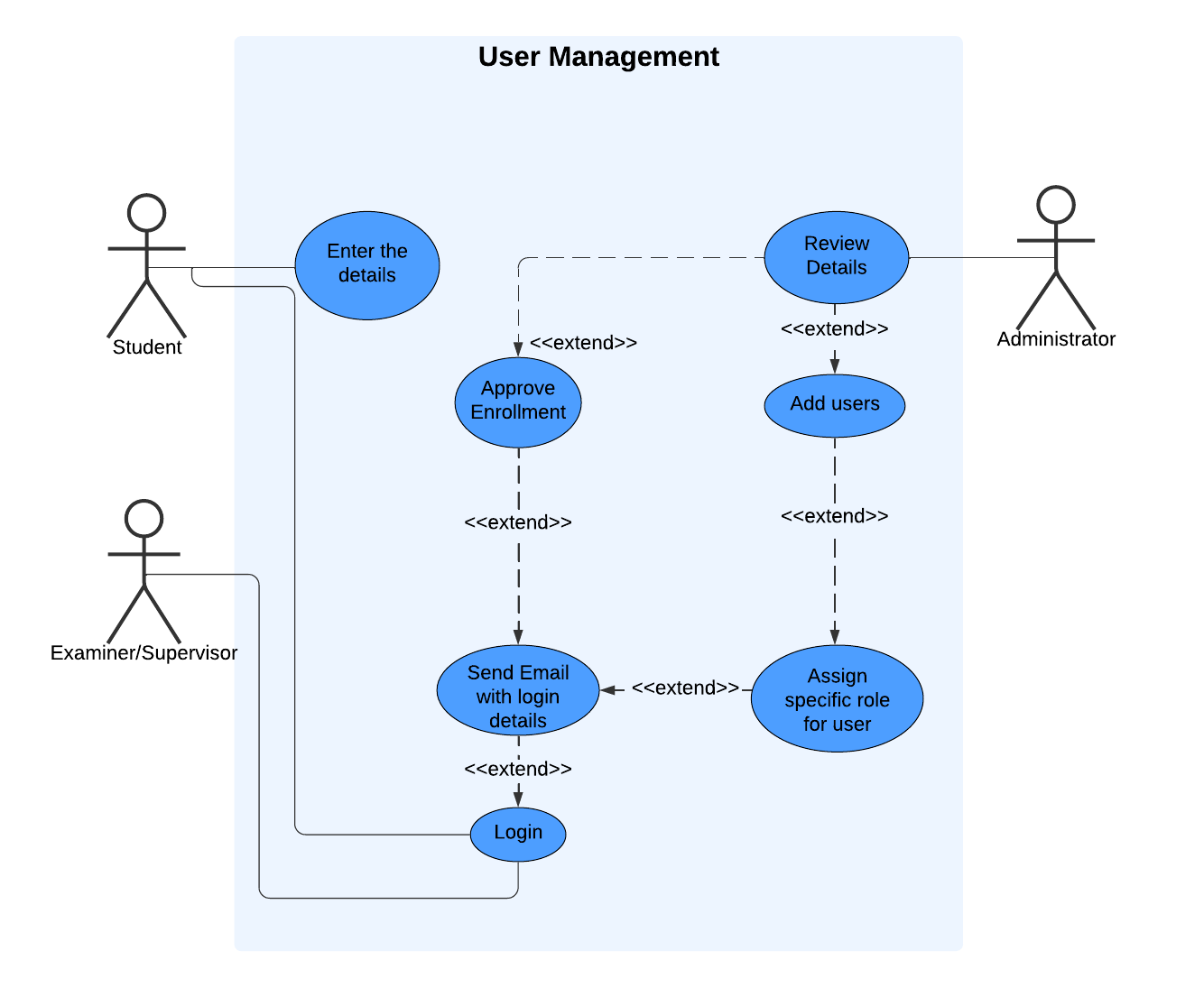
Additionally, web applications provide offline capabilities in some cases, allowing users to perform tasks without being constantly connected to the internet, unlike websites that depend entirely on an active connection. Web applications use advanced technologies like AJAX and APIs to enable real-time updates without refreshing the entire page, resulting in smoother and faster performance. Websites, on the other hand, usually require page reloads for content updates, leading to a less efficient user experience.

Hence the concept of web applications was chosen to implement the Postgraduate Student Management System according to the requirements.

## System analysis and Design

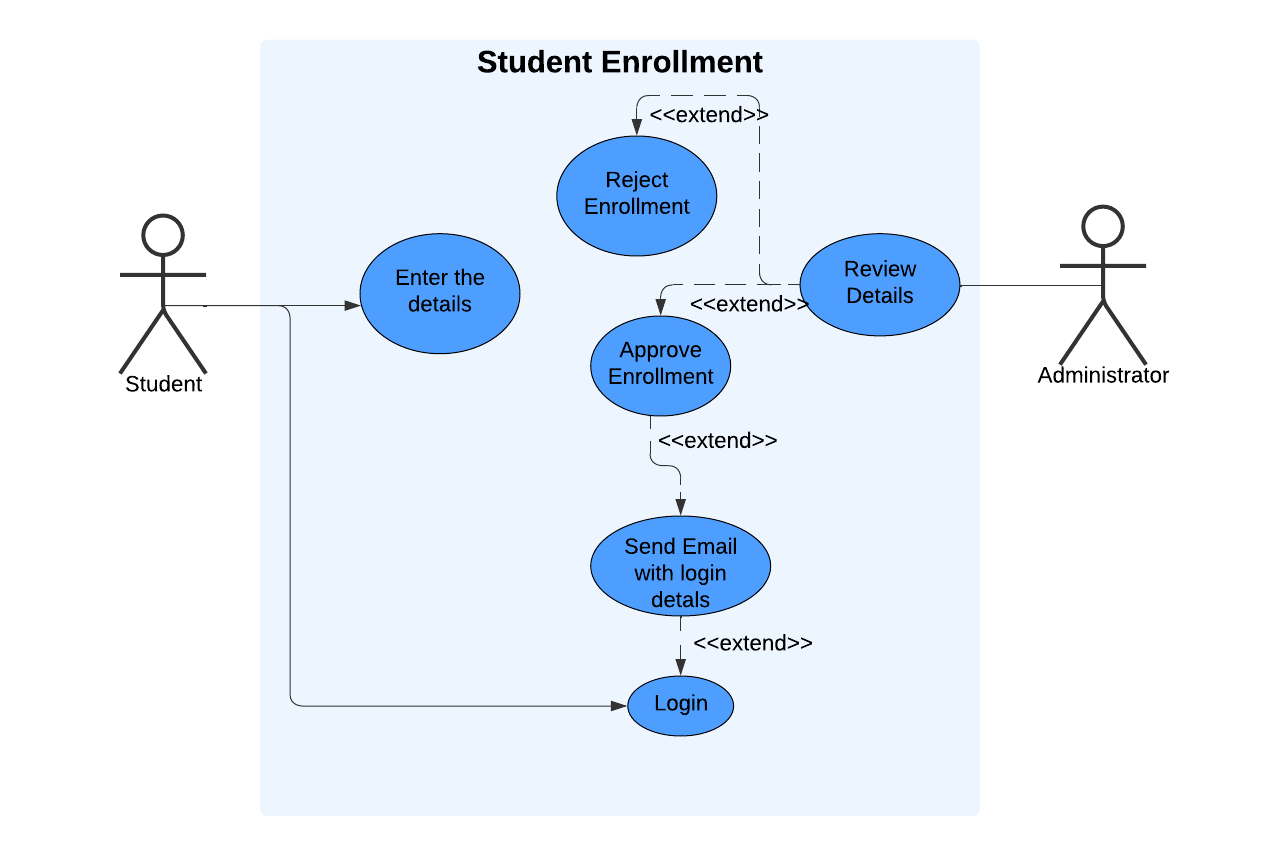
### Use case diagram

User Management



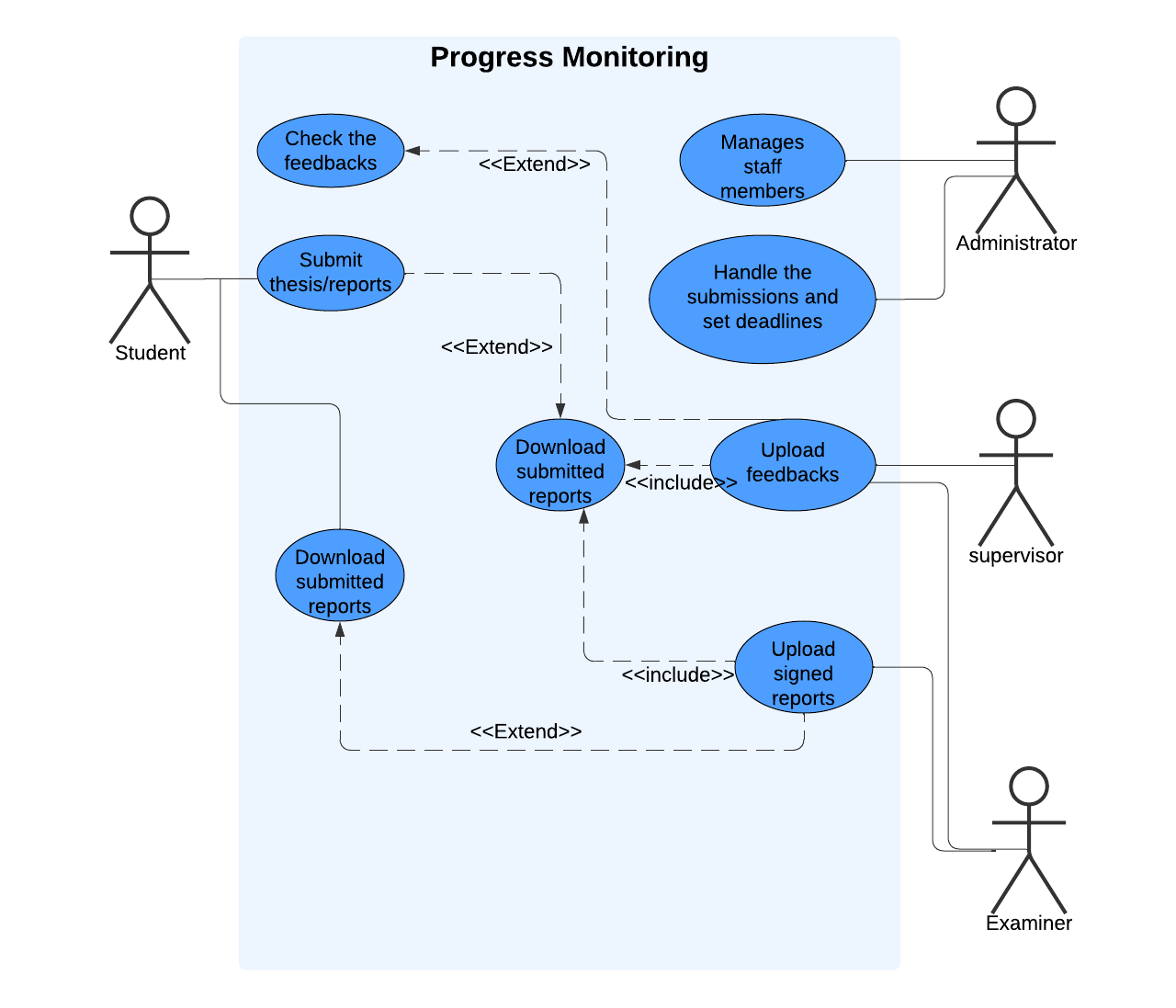
|  |  |
| --- | --- |
| Use case name | User Management |
| Actors | Student, Examiner, Supervisor, Administrator |
| Description | Students begin by submitting their details, which are reviewed by the administrator. Upon approval, the system automatically sends an email with login credentials to the student, allowing them to access the platform. Administrators can also add new users, such as examiners or supervisors, and assign them specific roles |
| Pre-conditions | * The administrator must have access to the system for reviewing and approving details. * The email system must be configured for sending notifications. |
| Post-conditions | * The student can log in to the system successfully. * Other users, if added, are assigned roles and notified accordingly. |

Student Enrollment



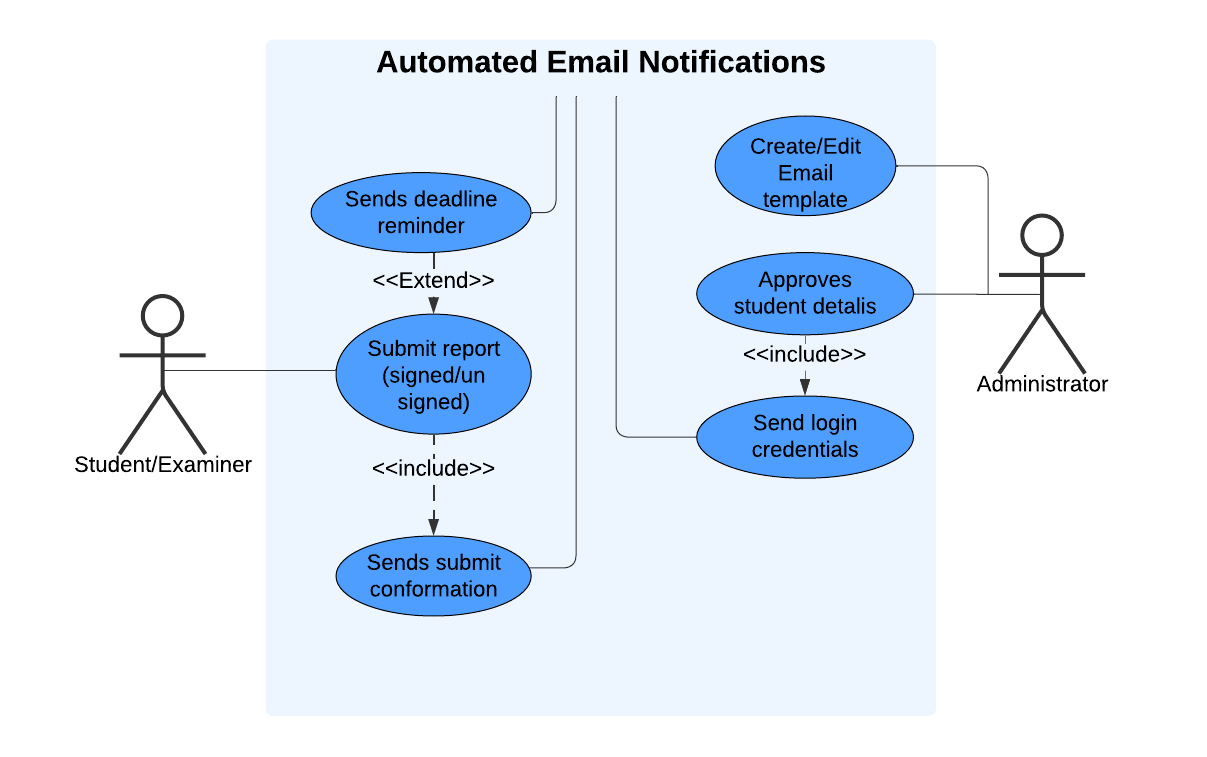
|  |  |
| --- | --- |
| Use case name | Student Enrollment |
| Actors | Student, Administrator |
| Description | Streamlines the process of registering students into the platform. This process ensures a controlled, automated  workflow for managing student registrations effectively |
| Pre-conditions | * The administrator must have access to the system for reviewing and approving details. * The system database must be ready to store student details and approval statuses. |
| Post-Conditions | * system maintains a complete record of all student enrollments for future reference |
| Flow of Control | * The student fills out their personal and enrollment details and submits them * administrator accesses the system to review the submitted information for completeness and accuracy. * If the details are valid, the administrator approves the enrollment. Otherwise, the submission is rejected * The system automatically sends the student an email containing their login credentials. * The student uses the credentials to log in and access the platform's features |

Progress Monitoring



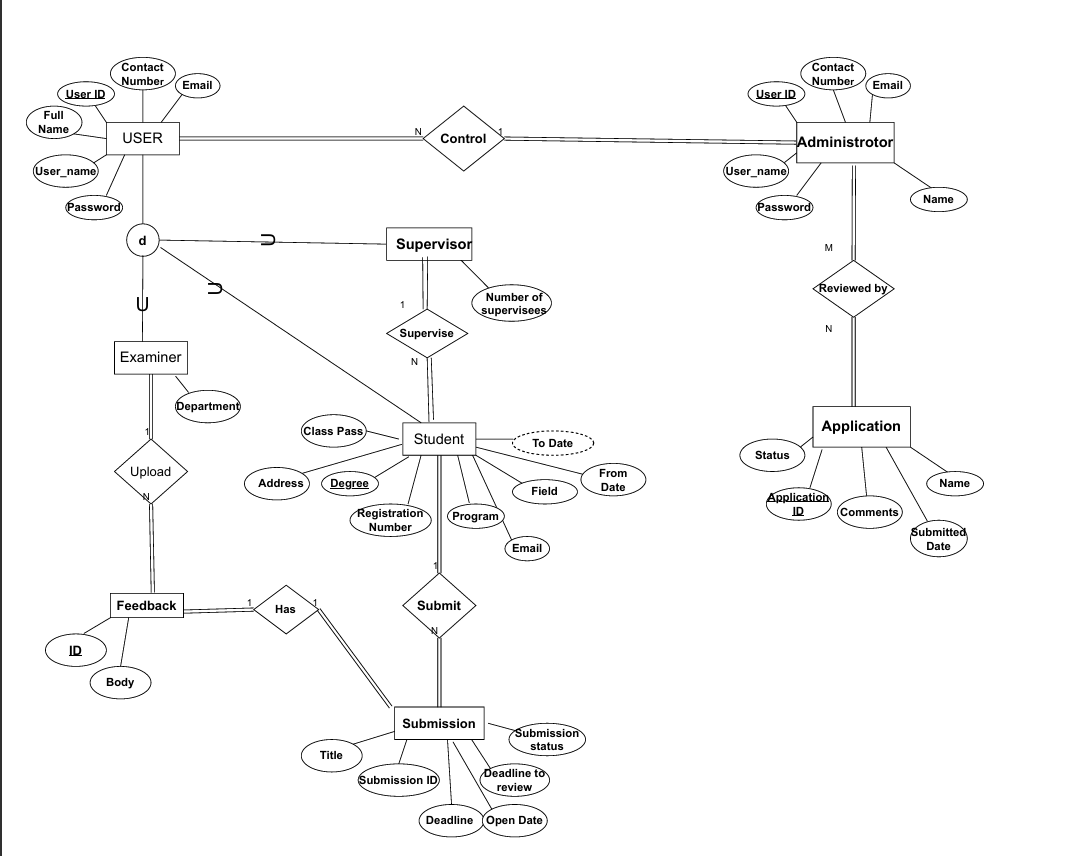
|  |  |
| --- | --- |
| Use case name | Progress Monitoring |
| Actors | Student, Administrator, Examiner, Supervisor |
| Description | Ensures an efficient workflow for tracking student progress |
| Pre-conditions | * The student, administrator, supervisor and examiner must have access to the system |
| Post-Conditions | * The system maintains a log of all submissions and reviews for tracking progress |
| Flow of Control | * The student uploads their report or progress update to the system. * supervisor, or examiner accesses the system to review the submitted reports. * The Examiner/Supervisor adds feedback or marks the submission and upload them. |

Automated Email Notifications

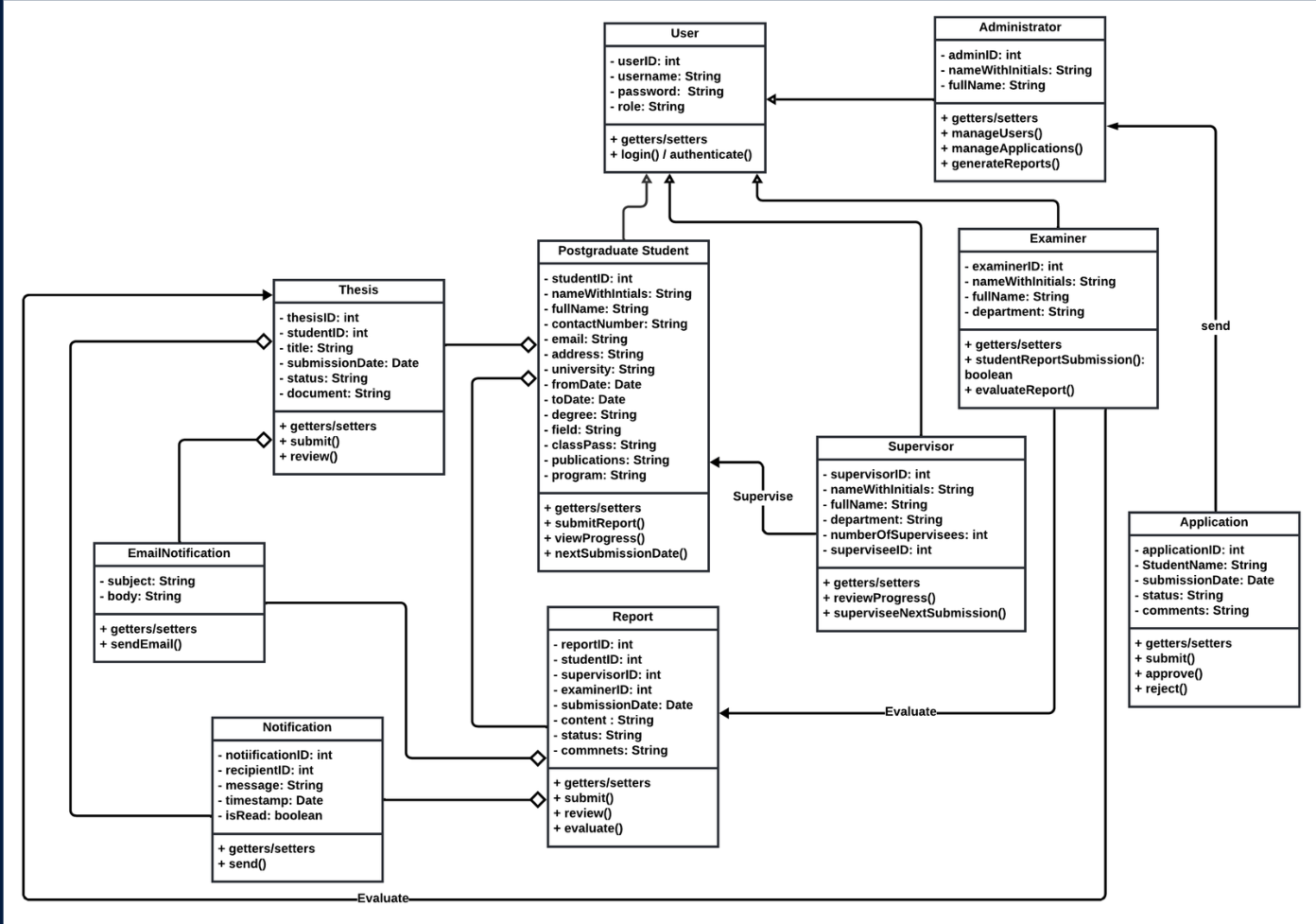


|  |  |
| --- | --- |
| Use case name | Automated Email Notifications |
| Actors | Student, Administrator, Examiner |
| Description | Ensures timely communication between stakeholders. When specific actions occur, such as enrollment approval, report submission, or approaching deadlines, the system automatically generates and sends email notifications to the relevant users |
| Pre-conditions | * The system must have email notification functionality configured and operational. * Predefined templates for various scenarios * Valid email addresses of recipients must be stored in the system * The triggering event must occur |
| Flow of Control | * An event such as enrollment approval, report submission, or an upcoming deadline triggers the system. * The system creates an automated email based on predefined templates corresponding to the triggered event * The system sends the email to the appropriate recipient(s) |

### EER diagram

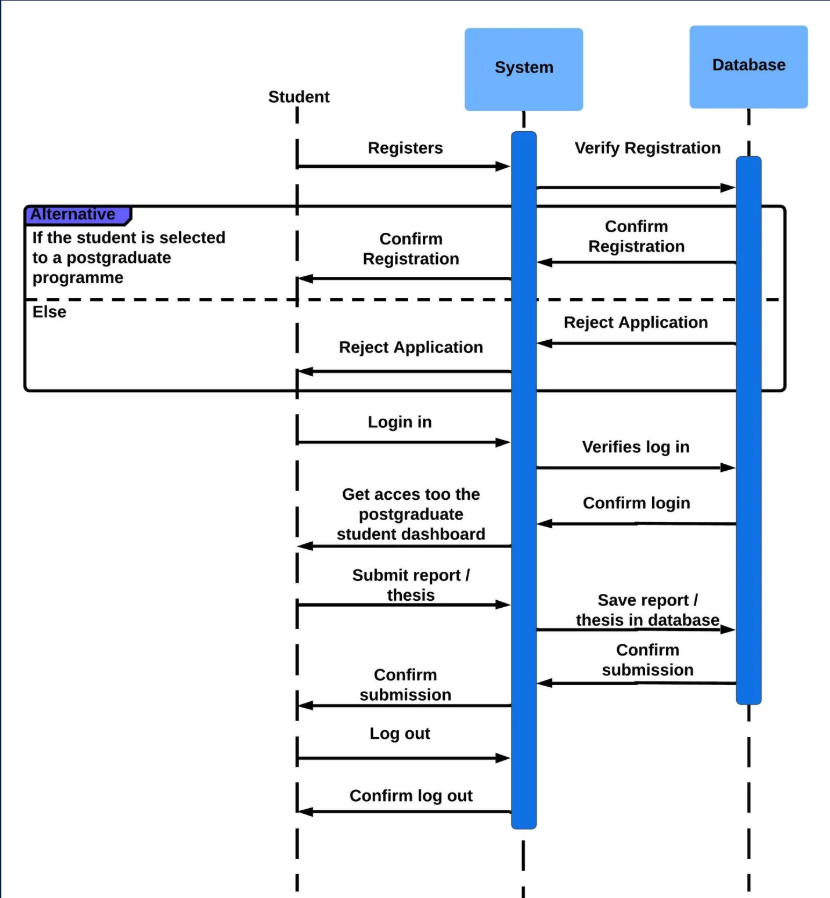


### UML Class diagram

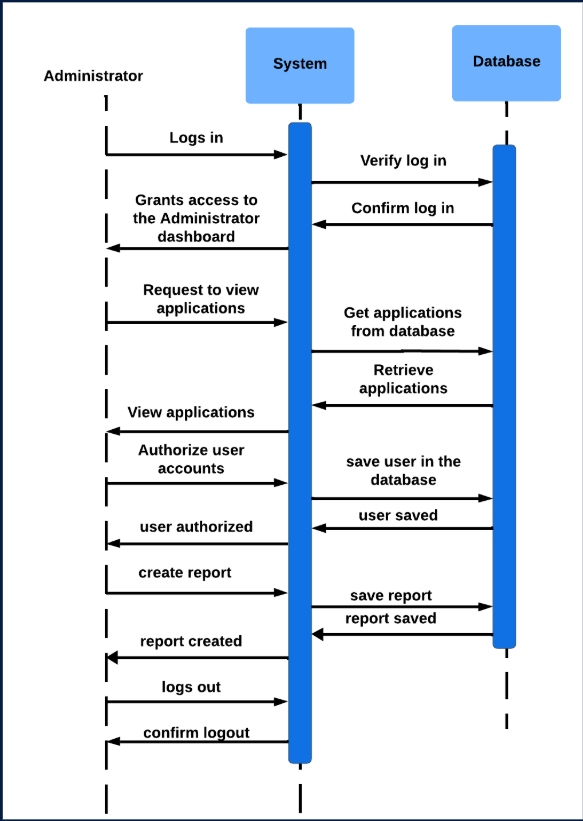


### Sequence diagram

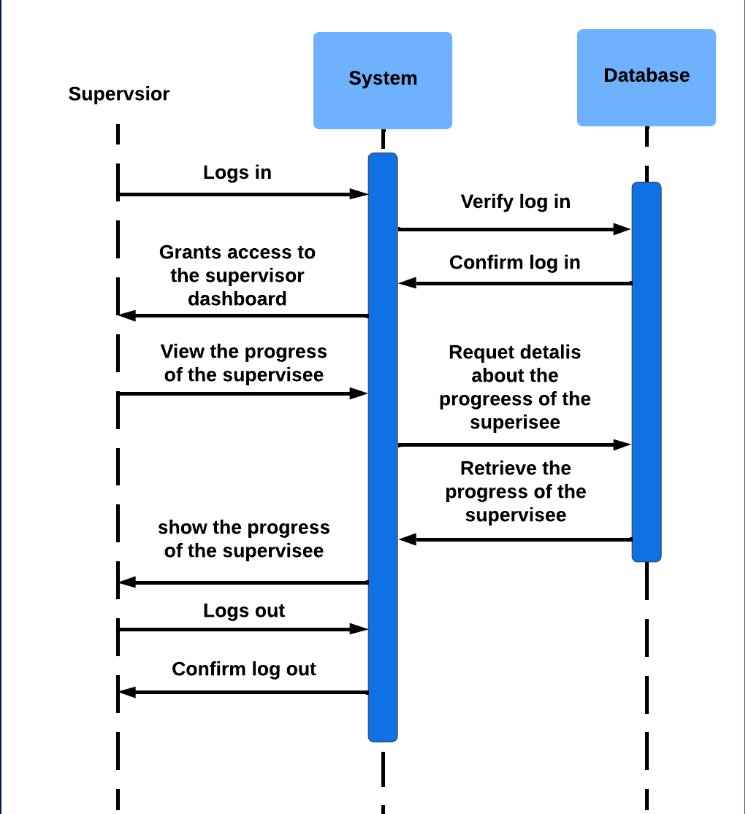
Student View



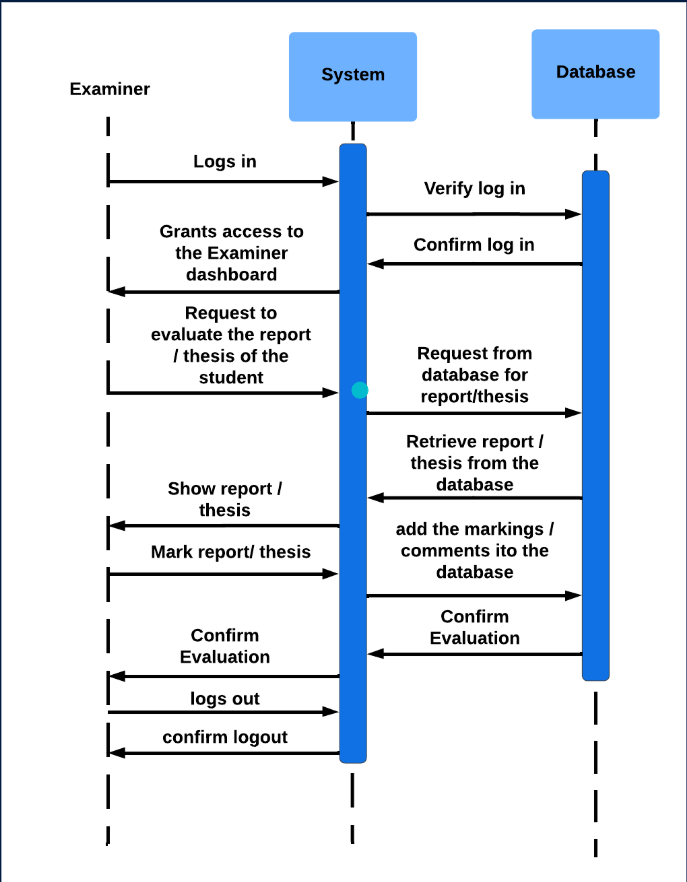
Administrator View



Supervisor View



Examiner View



## System design methodologies

**Frontend Design Methodology: Component-Based Architecture**

The front end of the system is designed using a component-based architecture, which focuses on breaking down the user interface into smaller, reusable, and self-contained units. Each component is responsible for a specific part of the user interface, such as a menu, a profile section, or a data table. These components work independently yet seamlessly integrate to form the entire application.

This architecture promotes modularity, making the system easier to maintain and scale. Each component manages its own logic, appearance, and behavior, ensuring that changes in one part of the interface do not inadvertently affect others. Components can also be reused across different parts of the system, reducing redundancy and development effort.

By using a hierarchical structure, components are organized into a parent-child relationship. Parent components define the overall layout and pass data or functionality to their child components. This structured approach enables efficient data flow and ensures consistency across the interface.

**Backend Design Methodology: Layered Architecture (MVC)**

The backend system is designed using a layered architecture, which organizes the application into distinct layers, each with a specific role and responsibility. This separation of concerns ensures that the system is well-structured and maintainable.

1. Presentation Layer:

This layer handles interactions with the frontend. It receives requests from the user interface, processes them, and sends back appropriate responses. By isolating this functionality, the presentation layer ensures the backend can adapt to different interfaces without significant changes.

1. Service Layer:

Acting as the intermediary, the service layer processes business logic and orchestrates the interaction between other layers. It handles tasks such as validation, decision-making, and invoking necessary operations. This layer ensures the backend adheres to the rules and workflows defined for the system.

1. Data Access Layer:

This layer manages communication with the database. It performs tasks such as querying, saving, and updating data. By keeping data operations separate from other logic, this layer ensures flexibility and security in managing the system’s information.

The layered approach provides clear boundaries between responsibilities, making the system easier to debug, test, and extend. Each layer can evolve independently, allowing developers to update business logic, user interfaces, or database structures without impacting other parts of the system.

## Database Design for the system

The database design of the system focuses on ensuring efficient data storage, retrieval, and management. A core principle adopted in the design is **normalization**, which organizes the data to eliminate redundancy and improve consistency.

**Normalization of the Database**

Normalization process comprises several stages and every stage has different goals as structuring the database into tables to reduce duplication and dependency issues. By dividing data into smaller, related tables and establishing clear relationships between them, normalization ensures:

* Data Integrity: Each piece of information is stored in a single place, preventing inconsistencies that arise from duplicate data.
* Efficient Storage: Reducing redundancy minimizes the amount of data stored, leading to better storage utilization.
* Simplified Maintenance: Changes to data can be made in one location and automatically reflected wherever that data is used, reducing the risk of errors.

The system’s database design follows multiple normalization forms:

* **First Normal Form (1NF)**: Ensures that all columns in a table contain atomic, indivisible values.
* **Second Normal Form (2NF)**: Eliminates partial dependencies by ensuring that all non-key attributes are fully dependent on the primary key.
* **Third Normal Form (3NF)**: Removes transitive dependencies, so non-key attributes are dependent only on the primary key.

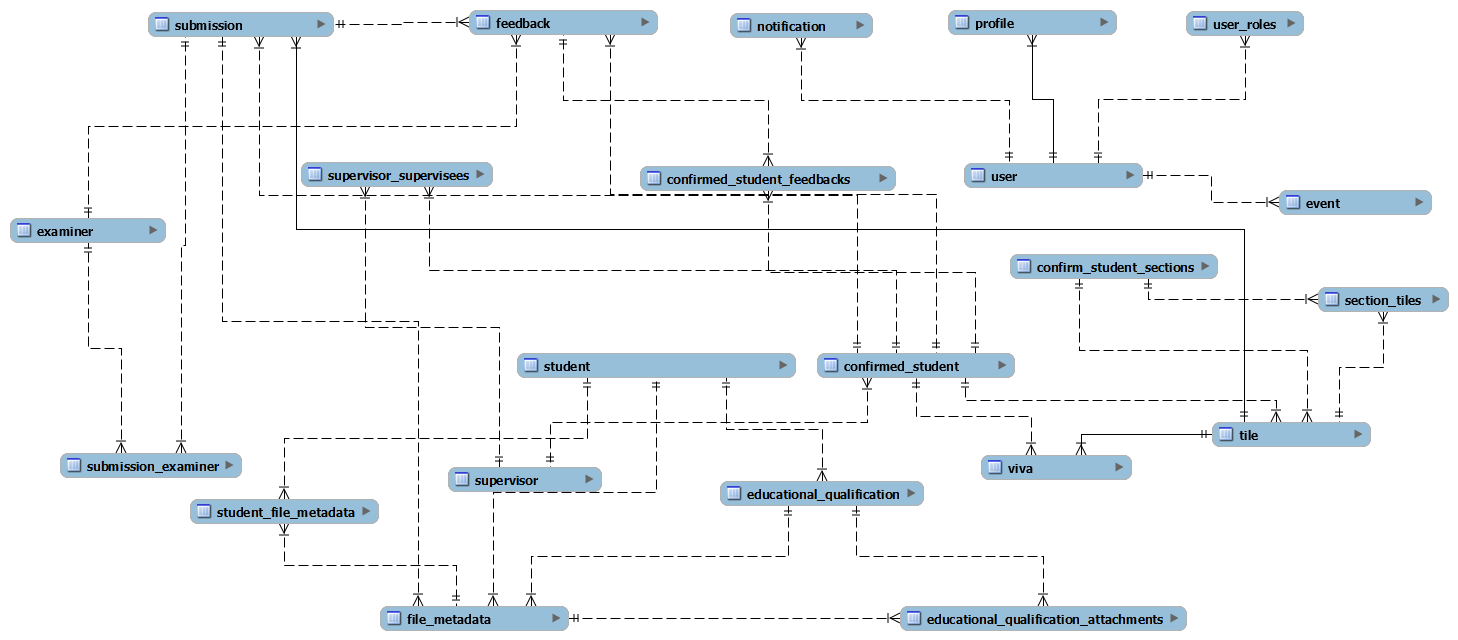
By adhering to these forms, the database is designed to be both efficient and adaptable to future changes.

**Backend Management of Queries**

The backend system handles database interactions using a well-structured approach that simplifies and streamlines query management:

1. *Abstraction Through Repository Layers*: Queries are managed via dedicated repository classes, abstracting direct database interactions. This ensures that the business logic remains separate from database logic, enhancing maintainability.
2. *Parameterized Queries*: To prevent vulnerabilities such as SQL injection, all queries are parameterized, ensuring that user inputs are safely handled before being passed to the database.
3. *Dynamic Query Building*: In scenarios where the system requires queries tailored to specific conditions, dynamic query builders are employed. These builders create custom queries at runtime while maintaining readability and security.
4. *Transactions for Data Integrity*: For operations involving multiple steps, transactions are used to ensure atomicity. If any step fails, the transaction is rolled back, maintaining the integrity of the database.

This robust database design and query management ensures that the system operates efficiently, securely, and reliably, meeting the needs of its users while being scalable for future enhancements.



## Project Management and Version Control

Project Management: process of planning, organizing, and executing tasks to achieve specific objectives within a set timeframe and budget.

* Defining Objectives
* Planning and scheduling
* Resource Management
* Risk Management
* Monitoring and Controlling

We mostly did the project management part through physical and online meetings. We discussed the problems we got in different stages of the project through meetings. Also, for requirement handling and new features introducing problems we met our Supervisor time to time. These meetings help a lot to make a good product.

Version Control: practice of tracking and managing changes to a project’s files over time.

It allows us to collaborate efficiently without overwriting each other’s work and provides a history of changes that can be used to debug issues or roll back to earlier versions.

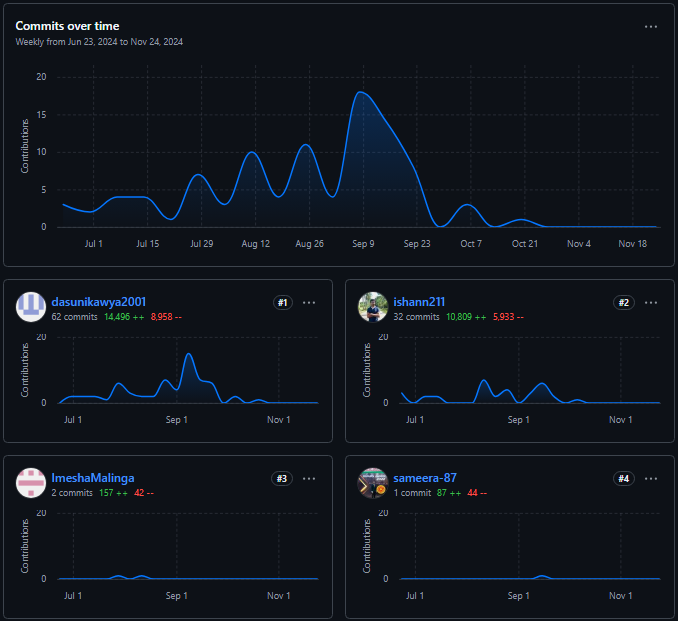
Key benefits we got,

* Collaboration
* Change Tracking
* Error Recovery
* Branching and Merging

We used version control system **Git** (platform GitHub) for our collaborative work in the system. We divide, create issues and share the code through GitHub to involve in the project effectively.

How we used Git can be shown as in the diagrams below.

Backend Development



Frontend Development

A screenshot of a computer

Description automatically generated

# Implementation

Implementation phase of PGSMS is the process of developing user requirements using the selected programming languages and tools according to the specifications. This is the process where functionalities expected by the system users become executable functions. This phase is executed progressively and carefully from one stage to the next by implementing functions that have been defined as several modules.

The following section describes the major code segments that have been used and the modular structure of the codebase.

## System Development Tools and Technologies

The PGSMS was developed using modern tools and technologies to ensure efficiency, scalability and ease of use.

### Spring Boot

Used for building the backend services, handling API requests, managing database operations, and implementing security features like JWT-based authentication.

### Angular

Used for creating the front-end, providing an interactive and user-friendly interface for various stakeholders. Ensures responsive and supports role-based access to different features.

### MySQL Workbench

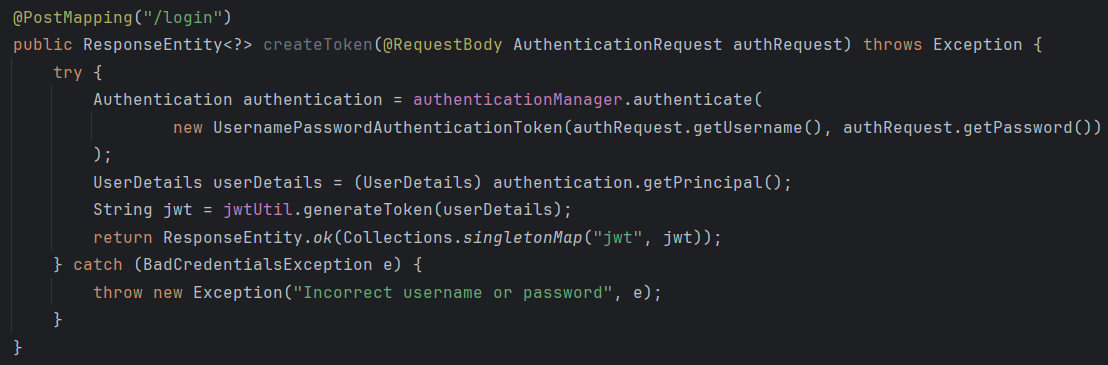
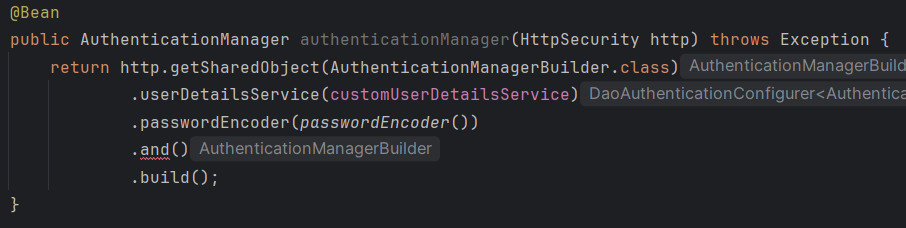
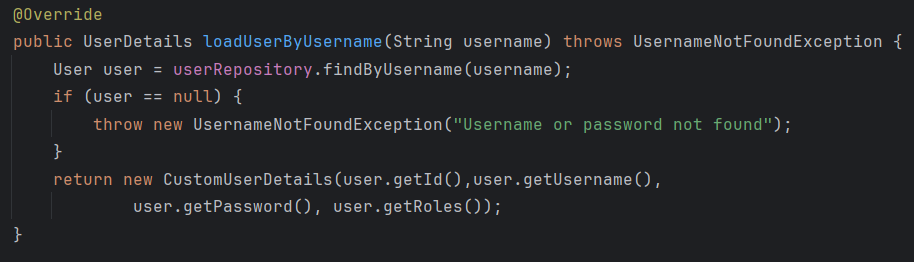
Utilize for designing and managing the database.

## Major Code Structure

This section describes all major code segments that have been implemented during the implementation phase to make the intended functionalities of the project a workable and productive solution.

### User logging handling

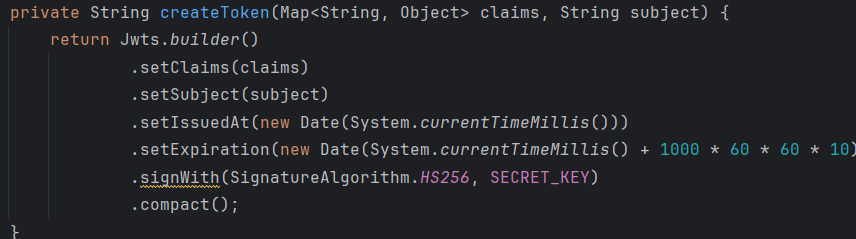
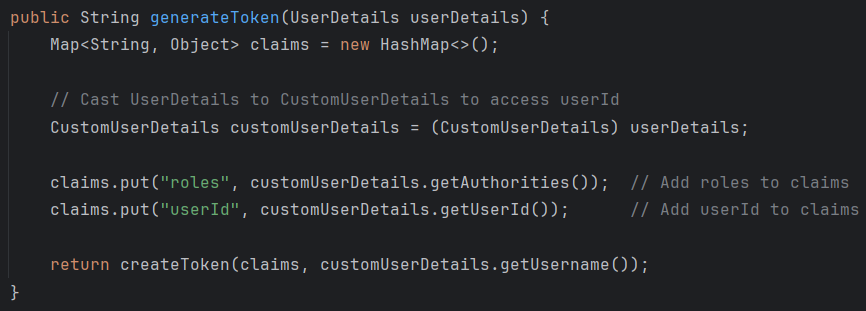
The code segment displayed below handles the user login accounts in the system.



The user login function was implemented using Spring Security in the backend. Upon login, the user’s credentials are validated against the database. If successful, a JWT token is generated and returned to the front end. The angular front end stores the token securely and includes it in headers for all subsequent API requests to ensure secure communication.

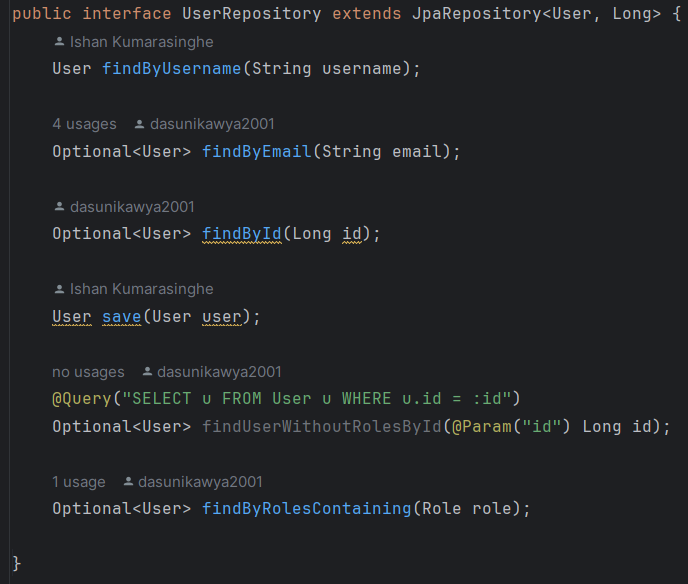
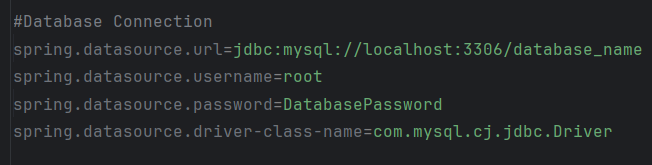
### Session management

Sessions are managed using JWT tokens.



Tokens include user information and expiration time to enhance security. The front end automatically logs users out when their token expires, ensuring secure access to the system.

### Database Connection Session

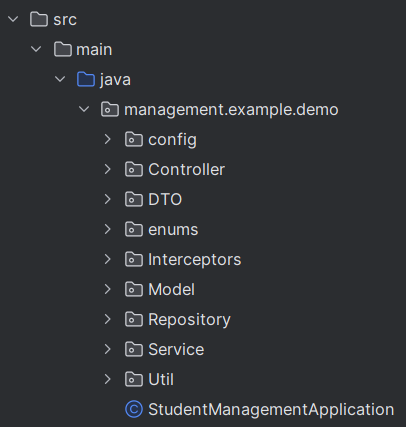


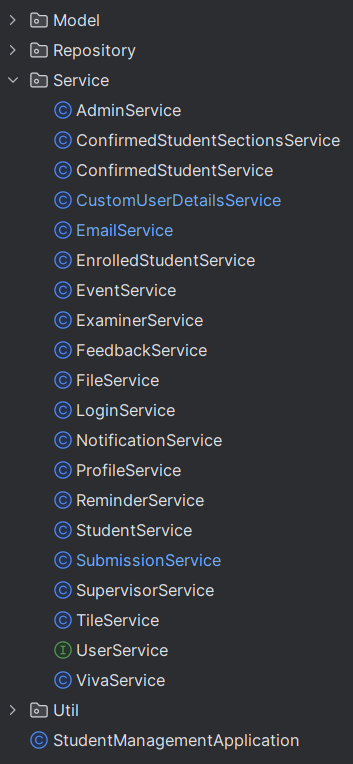
The system uses Spring Data JPA to connect to a MySQL database.

Predefined repository methods for common database operations like save(), findById(), and delete() have been used. Custom queries were used for complex data retrieval.

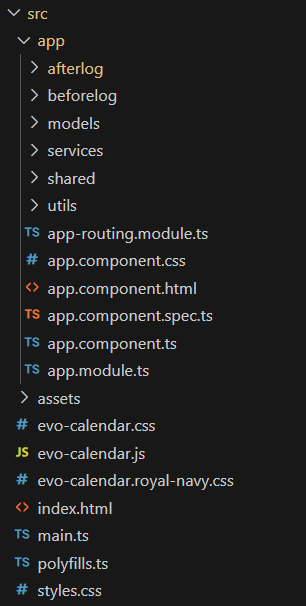
## Module Structure of the System

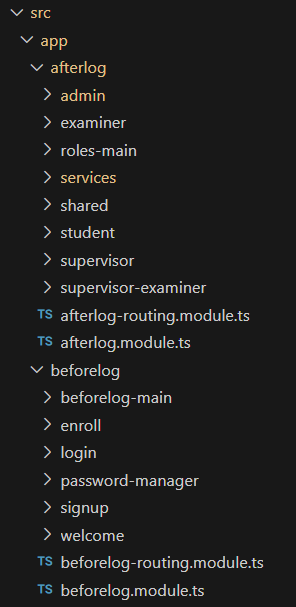
The following figure illustrates the module structure of the spring boot backend of the system.





The following figure displays the module structure of the Angular front end.





# Testing and Bug fixing

Software testing involves examining and checking software through different testing processes. The objectives of these processes can include:

* Verifying software completeness in regard to functional requirements.
* Identifying technical bugs/errors and ensuring the software is error-free.
* Assessing usability, performance, and security.

## Software Testing vs Software Evaluation

Software testing and Software evaluation are two major processes that should be clearly understood to produce a high-quality software system.

### Software Testing Process

The testing process aims to verify that the system performs according to its design specifications. Different techniques can be used for testing and the following are the widely used methods among them.

**Types of Testing**

* Unit Testing

In this level of testing, individual units or components of the system are tested. The purpose is to validate that each unit performs as designed.

* Integration Testing

The level of the testing process where individual units or components of the system are tested. The purpose is to expose faults in the interaction between the integrated units.

* System Testing

The level of the testing process where a complete, integrated system is tested. The purpose of this level is to evaluate the system’s compliance with the specified requirements.

* Acceptance Testing

The system is tested for acceptability to ensure the system is ready for delivery to users.

Unit testing was performed in parallel with each individual module development during the system implementation. Integration and system testing activities were carried out after completing the implementation of each use case.

### Software Evaluation

The objective of the software evaluation process is to validate that the implemented software system fulfills the requirements of the project.

## System Test Plan

A Test Plan can be described as a methodical and systematic approach which is used to verify and validate the quality of PGSMS. Defining a good test plan and a strategy can be very effective in discovering errors and bugs in the system. A properly defined and completed test plan can be used as documentary evidence to prove the quality of the software product.

|  |  |  |
| --- | --- | --- |
| Module Name | Functions to be Tested | Test Priority |
| User Login Module | Student Login Function | High |
| Staff Login Function | High |
| Forget Password Function | High |
| Student Enrollment Function | New student enrollment function | High |
| View students enroll data | Medium |
| Signing up students to the system | High |
| Staff Management Module | Add new staff member function | High |
| Search existing staff members | Medium |
| Assign Supervisor Module | Assign a supervisor to a student | High |
| Change the existing supervisor | High |
| Report Submission Module | Open a submission link | High |
| Assign a deadline | Medium |
| Submit a completed report | High |
| Submit with no attachment | Medium |
| Assign examiners to review the submissions | High |
| Email Template Module | Adding new templates | High |
| Editing existing templates | High |
| Use a template and send an email through the system | High |

## System Test Cases

**Test Cases for User Loging Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Steps for Testing** | **Expected Result and Actual Result** | **Status** |
| 1 | Input correct username Input correct password Click - Log In button | Successfully login into the system | Pass |
| 2 | Input correct username Input incorrect password Click - Log In button | Prompt “Incorrect username or password” | Pass |
| 3 | Input incorrect username Input correct password Click - Log In button | Prompt “Incorrect username or password” | Pass |
| 4 | Input incorrect username Input incorrect password Click - Log In button | Prompt “Incorrect username or password” | Pass |

**Test Cases for Student Enrollment Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Steps for Testing** | **Expected Result and Actual Result** | **Status** |
| 1 | Go to enroll option and click that button | “Enrollment Form” should display | Pass |
| 2 | Leave some fields empty and click submit button | Display error message |  |
| 3 | Leave the personal information documents fields empty and click submit button | Display error message |  |
| 4 | Input all the fields with correct pattern and click submit button | Display message “Enrolled Successfully.” | Pass |
| 5 | Go to “Enrolled Students” option | All enrolled students should be loaded. | Pass |

**Test Cases for Report Submission Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Steps for Testing** | **Expected Result and Actual Result** | **Status** |
| 1 | Submit without uploading the submission | The submit option does not display. | Pass |
| 2 | Upload a single file. | Display ‘files upload successfully.’ message. | Pass |
| 3 | Upload multiple files. | Display ‘files upload successfully.’ message. | Pass |
| 4 | Remove uploaded files. | Ask another time to remove the uploaded files. | Pass |

**Test Cases for Test Cases for Email Template Handling Module**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Steps for Testing** | **Expected Result and Actual Result** | **Status** |
| 1 | Adding new template | Display “Template added successfully.” | Pass |
| 2 | Editing existing template | Display “Template updated successfully.” | Pass |
| 3 | Send email using an existing template | Display “Emails sent successfully.” | Pass |

# User Manual

Postgraduate Student Management System User Manual

**Table of Contents**

1. Introduction
   1. What is the Student Management System
   2. User types
   3. Requirements for users
   4. Logging to the system and editing profile
      1. How do I register to the system?
      2. How do I log into the site?
      3. How do I change my password?
      4. How do I edit my profile?
      5. How do I view my profile?
2. Manual for Administrator
   1. Getting started
   2. Student registration
   3. Manage Students’ page
   4. Task creation and management
      1. How do I create a task?
      2. How do I grant access to students?
      3. How do I add content to a task?
      4. Content allowed for uploading
   5. Resources & activities
      1. Emails management
      2. Feedback forums
   6. How to analyses the student progress
3. Manual for Supervisors
   1. How to manage Student progress
4. Manual for Examiners
   1. How to examine a report
5. Manual for students
   1. How do I log after registration
   2. How do I contact a supervisor?
   3. How do I take part in a course?
   4. How do I submit a report/thesis?

**Introduction**

**What is the Student Management System**

The *Postgraduate Students Management System (PGSMS)* is a comprehensive platform designed to efficiently manage the academic and administrative processes related to MPhil. and PhD. students in the Department of Computer Engineering at the University of Peradeniya. This system addresses the challenges associated with traditional, manual processes by introducing streamlined, automated workflows that enhance operational efficiency, improve communication, and ensure effective monitoring of students’ academic progress.

The purpose of this document is to make potential users familiar with the system and its basic functions in order to facilitate proper and smooth use of these e-learning tools. This customized manual describes just essential functions and resources regularly which users need for optimal performance.

This manual was developed to explain how to use/operate PGSMS being “Administrator”, “Supervisor”, Examiner” and “student” user type.

**User types**

There are 5 types of PGSMS users:

* **Administrator**, who is responsible for system functionality and administrators access rights, keys, passwords, courses that are created, etc.
* **Supervisors**, also known as professors, instructors, lecturers or evaluators, who are responsible for supervising postgraduate students.
* **Examiners**, who is for evaluating of research reports/ thesis and other criteria.
* **Supervisor-Examiner**, who has responsibilities of both supervisors and examiners.
* **Students** who have access to the materials and activities made available on the platform.

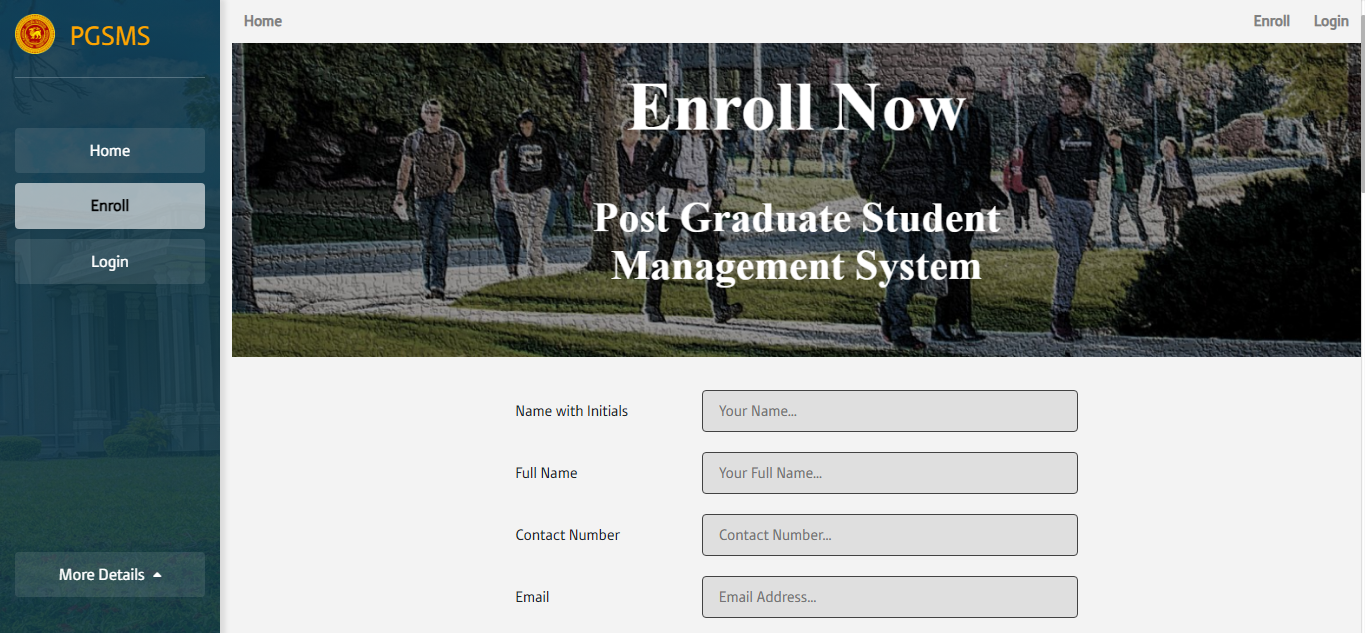
**Requirements for users**

* A PC with an internet browser
* A stable Internet connection

**Logging to the system and editing profile**

**How do I register for the system?**

In order to register, *students* need to fill in the enrollment form in the system.



After that, the Administrator will be concerned about your enrollment.

*Supervisors and Examiners* are assigned by the administrator, and they will receive a username and password after they are assigned.

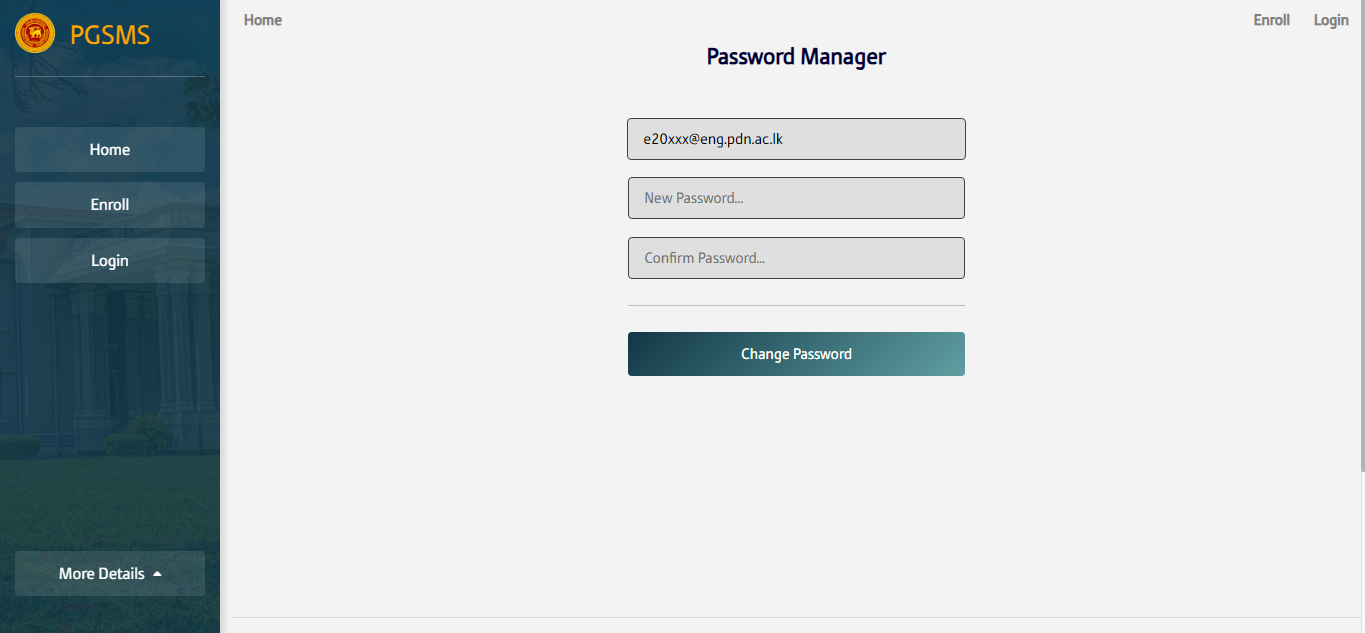
**How do I log into the site?**

A screenshot of a login box

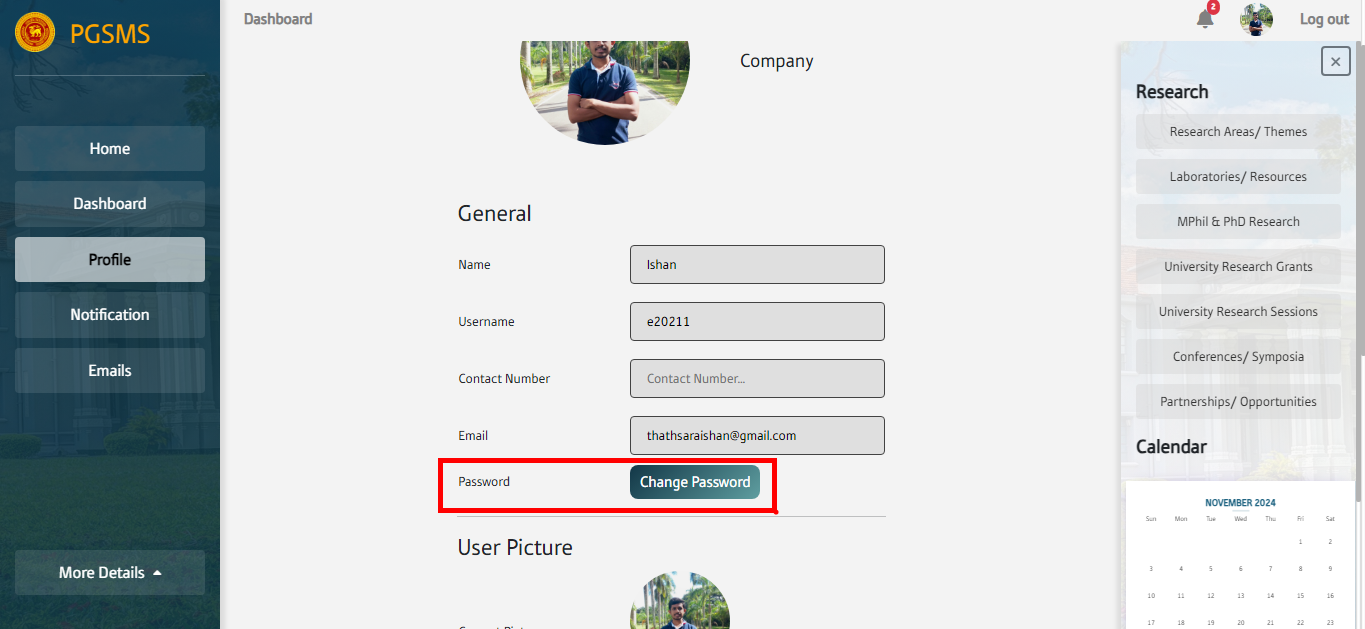
Description automatically generatedOnce the system administrator has informed you by email (Contains username and password) that you may log in to the site, open a web browser and access PGSMS through internet.

**How do I change my password?**

If you forgot the password and you can’t login to the system, then you can change the password using your email address.

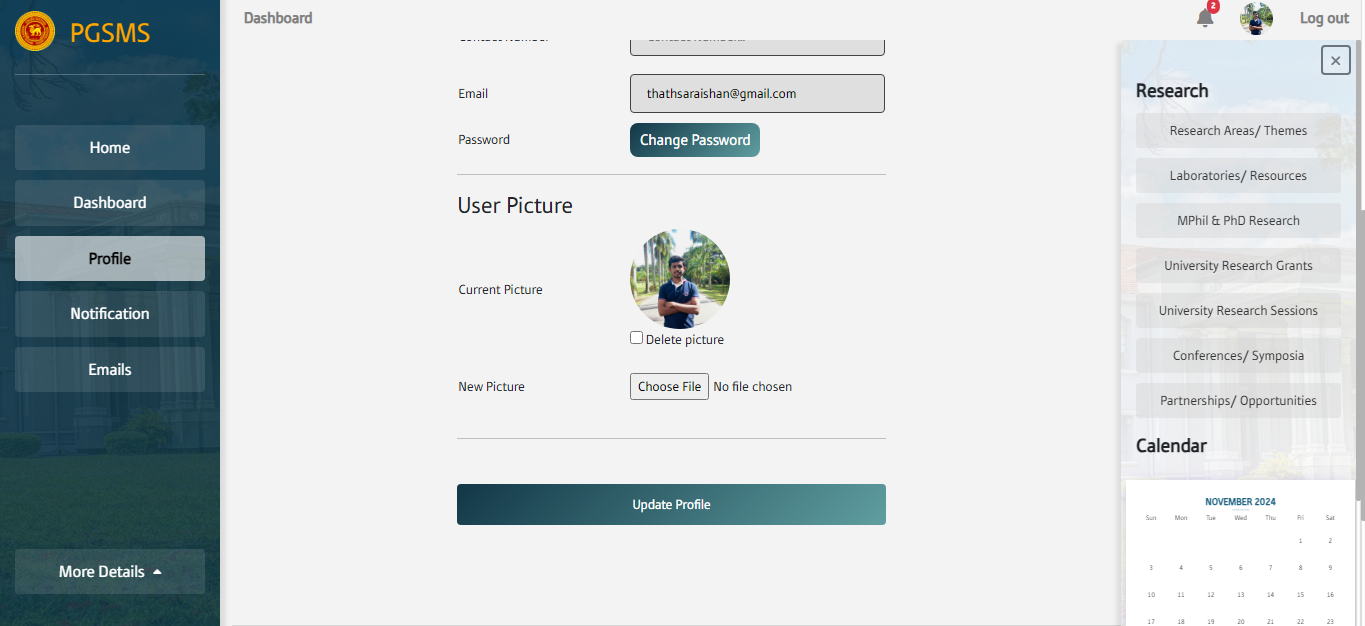


If you logged in and you want to change your password, you can go to the profile and change your password.



**How do I edit my profile?**

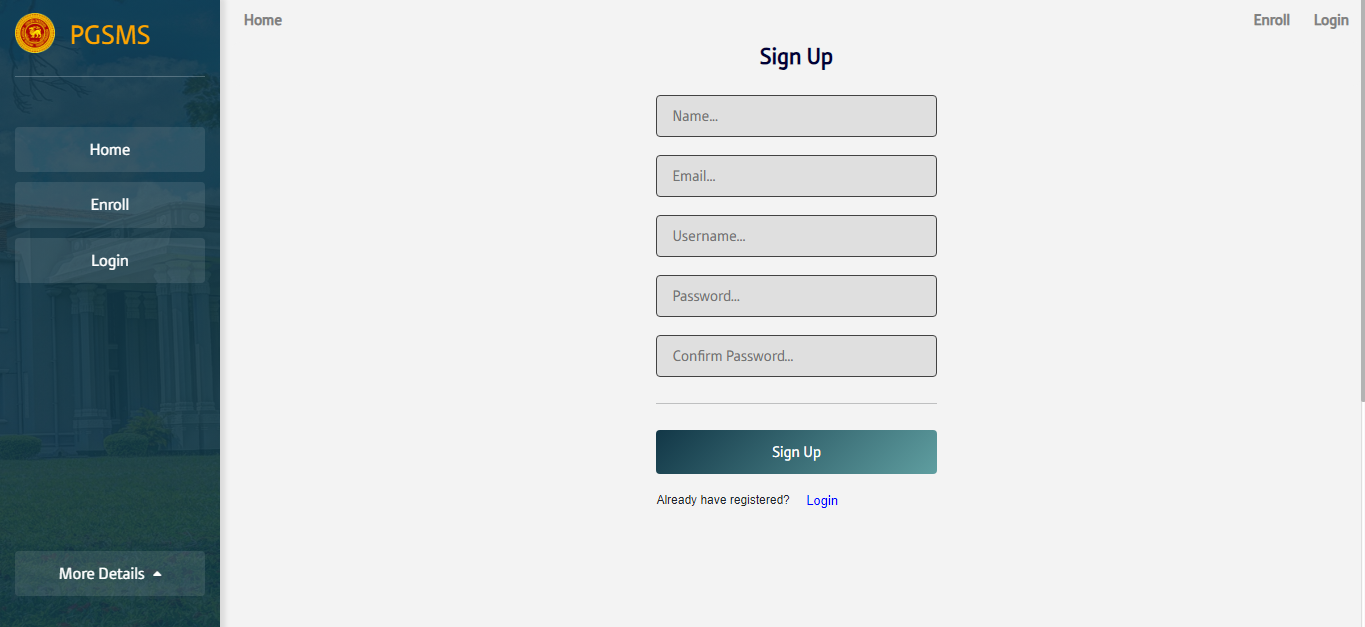
In the profile page you can edit your profile details like Name, email address, password, profile picture etc.



**Manual for Administrator**

**Getting started**

Administrators must signup to the system before login and manage the system.



**Student registration**

Administrators can see the Enrolled Students in the Enrolled Student page.

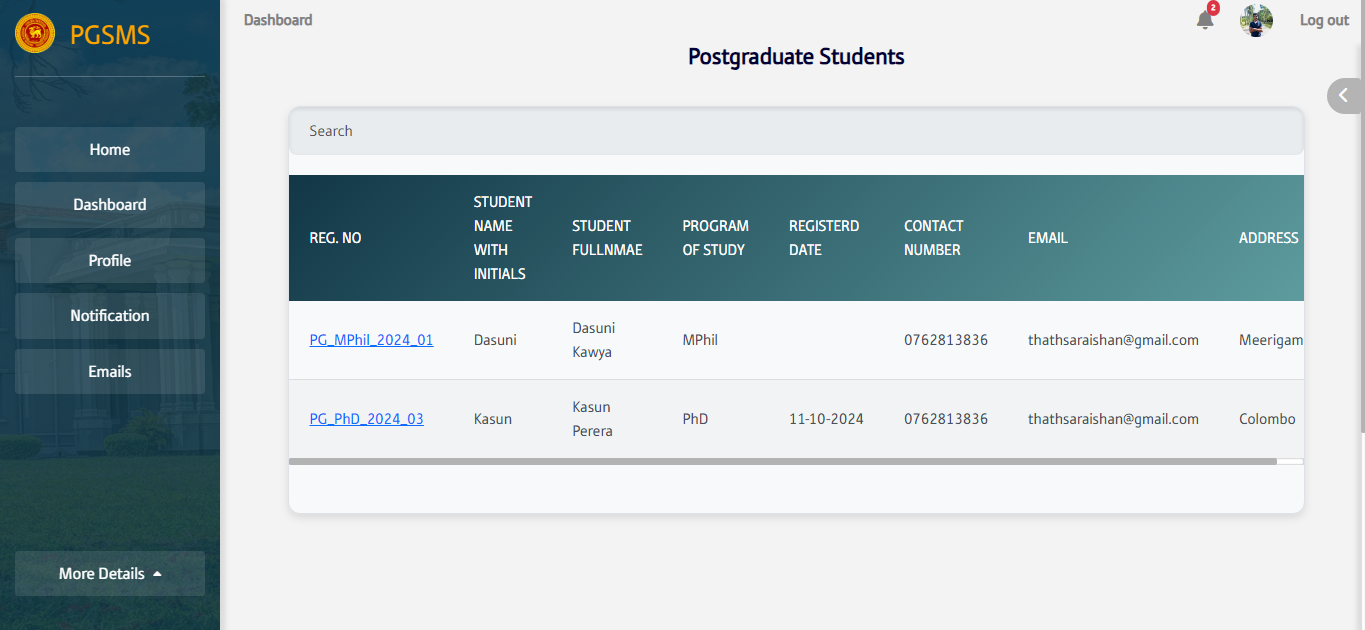


Administrators can see Students’ details and qualifications. After looking at them Administrators can change their details, assign a registration number directly and register them to the system.

Then, students will notify via an email with login credentials and then they can log into the system.

**Manage Students’ page**

All the registered students are on the Postgraduate Students page.

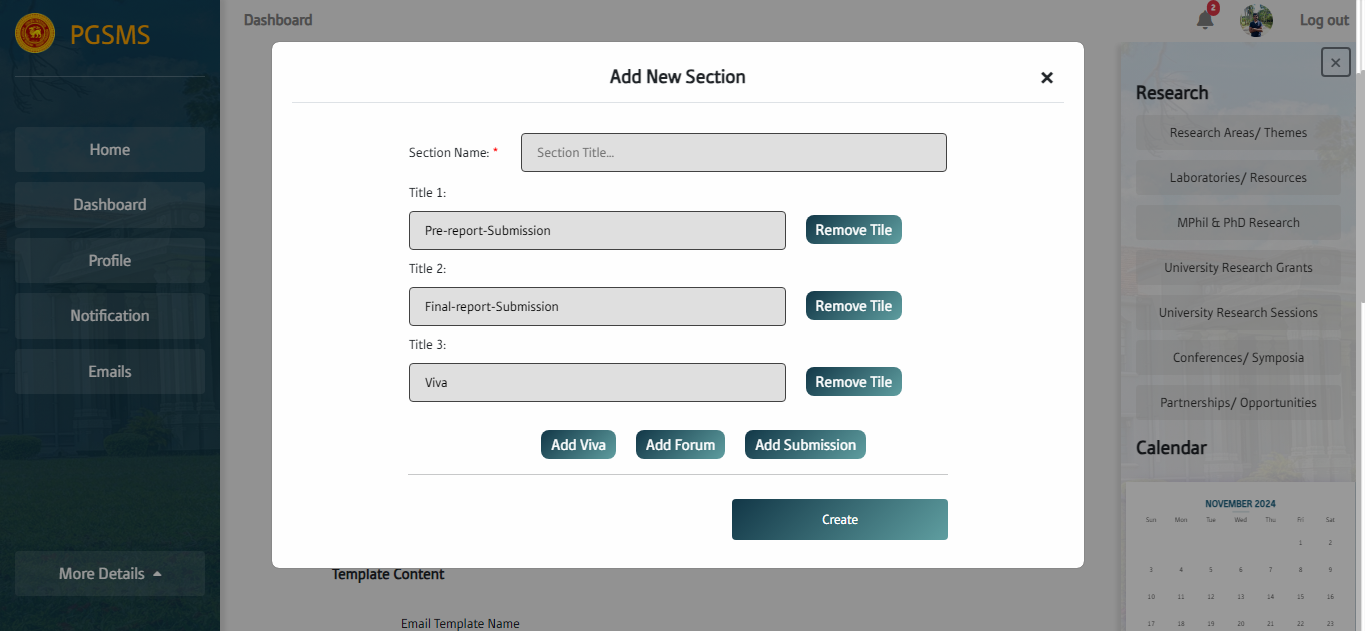


Clicking the Registration Number Administrator can view the profile of students and manage their research progress.

**Task creation and management**

**How do I create a task?**

Administrator can create a section with tasks assigned to a student. Tasks are managed for every student because they have different tasks according to their research.



A section can include tasks like Submissions, Vivas and Forums according to student’s academic year.

A screenshot of a calendar

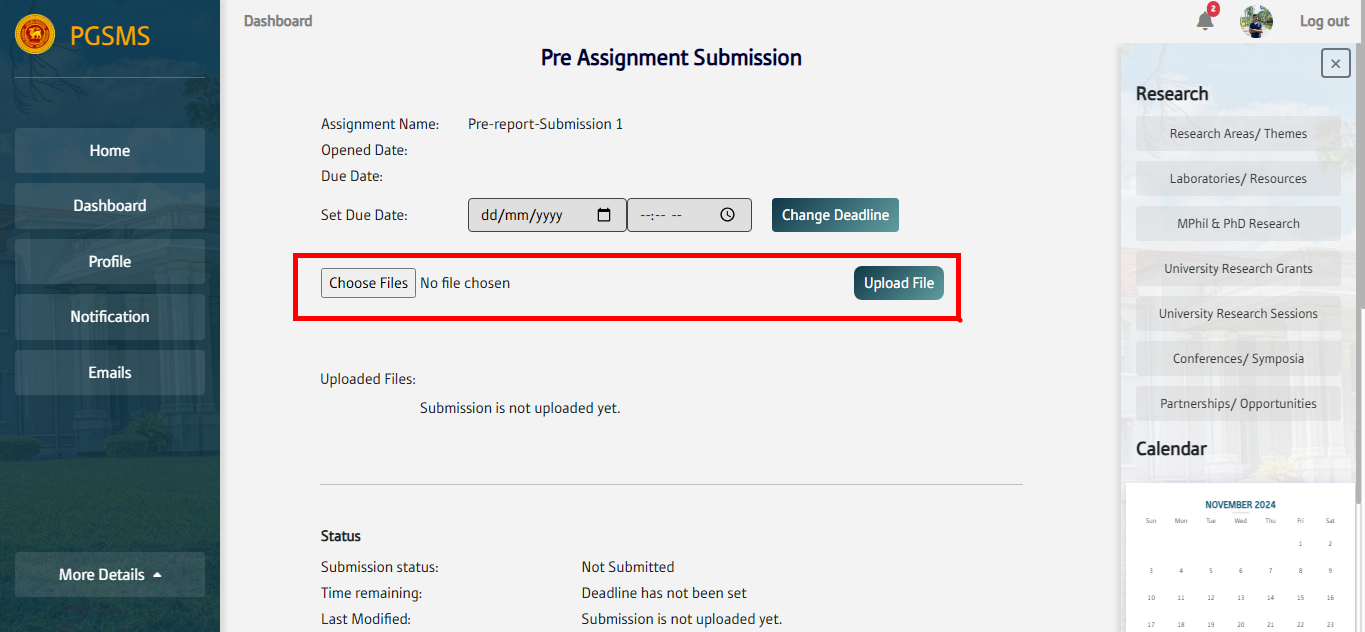
Description automatically generated

**How do I grant access to students?**

Students will see the sections after creating them in their academic profiles.

**How do I add content to a task?**

In a task, the administrator or supervisor can upload any files included about the task.



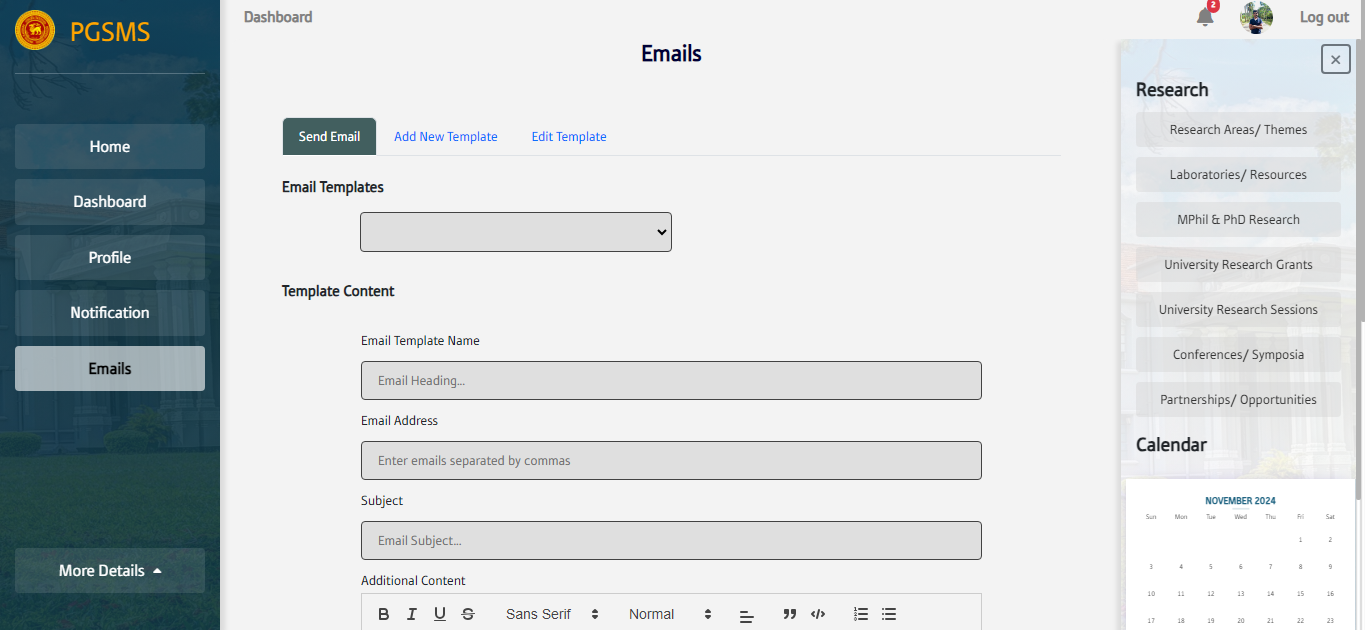
Also, the administrator can set a due date for submissions and track the progress of the task.

**Content allowed for uploading**

**Resources & activities**

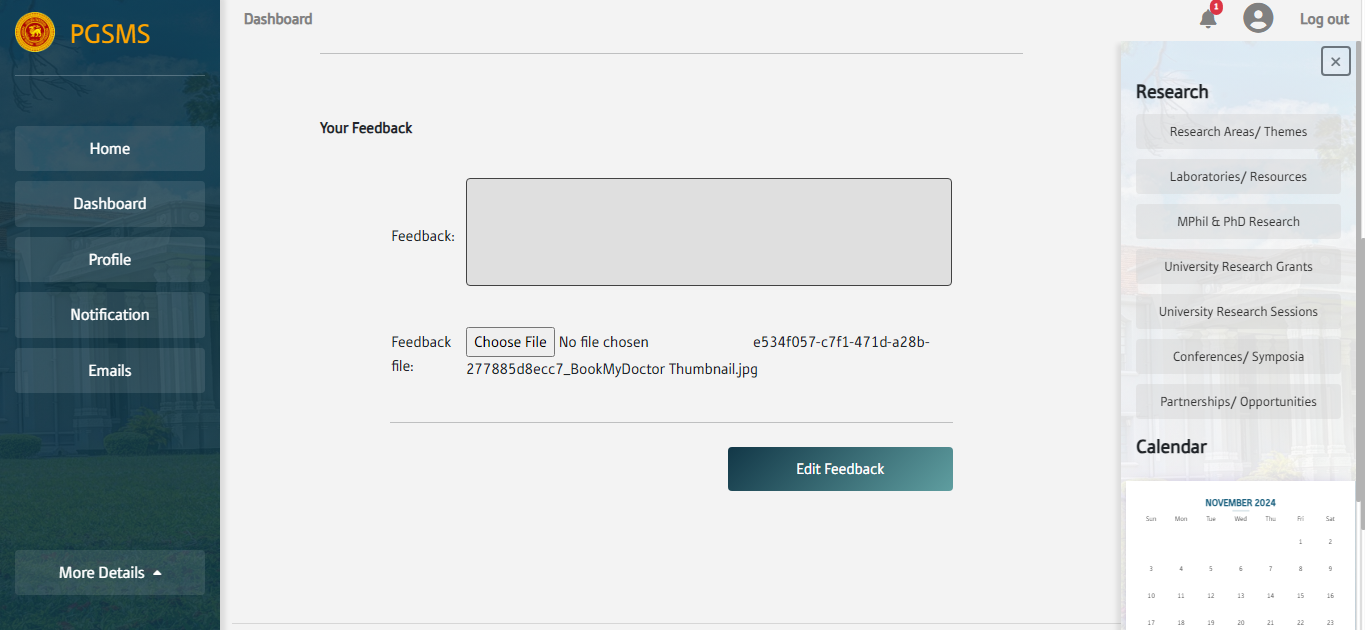
Students and upload their report to the system before the deadline. The whole process is managed by the administrator.

**Emails management**



The Administrator can manage the automated email system of the system and also send instant private emails to students via the system.

**Feedback forums**



Supervisors and examiners can give their Feedback to the reports that students have submitted.

**Manual for Supervisors**

**Manual for Examiners**

**Manual for students**

# Conclusion

## Challenges faced

During the development of the Postgraduate Student Management System (PGSMS), several challenges were encountered.

* Integrating the *Spring Boot backend* with the *Angular frontend* required careful handling of API communication and data consistency. The team overcame this by thoroughly testing API end points.
* Ensuring secure authentication and *role-based access* control with *JWT tokens*. The team addressed this by utilizing Spring Security to ensure strict access controls.
* Designing a *scalable database* schema that accommodated complex relationships, such as student submissions, assigning examiners, and examiner feedback, was another critical task. The team used tools like MySQL Workbench to visualize and optimize the database structure effectively.
* Implementing *email templates* and *file management* introduced complexities in maintaining a user-friendly interface and backend logic. This process was streamlined by incorporating reusable email template features and file services.

Time management was also a challenge, as balancing feature implementation, testing, and documentation required meticulous planning. Despite these hurdles, the project team successfully delivered a functional and robust system.

## Feedback

To enhance the system, valuable feedback and suggestions were received from the supervisors and lecturers throughout the development and implementation of the Post Graduate Student Management System (PGSMS). The supervisor as well as the customer Dr. Upul Jayasinghe praised the project as a great effort, recognizing the functionality and features implemented. Dr. Damayanthi suggested improving the email functionality by configuring email templates for better customization and usability. For that, the team has implemented a robust Email Template Service, allowing admins to create, edit, and manage email templates directly through the system.

Additionally, based on the user requirements, the system was enhanced to include features for editing student enrollment details by the admins during the enrollment process. The system was further upgraded to allow students to upload multiple educational qualifications when enrolling, rather than limited to one, providing more flexibility. These improvements, driven by constructive feedback, made the system more user-friendly and aligned with the department’s requirements.

## Work Task Division and Individual Contribution

Supervisor – Dr. Upul Jayasinghe

|  |  |
| --- | --- |
| E/20/197 | * Mainly worked as the backend developer. * Manage the user roles by defining, assigning them to users and restricting access to specific endpoints based on the user authorities. Configure Spring Security to secure those endpoints by role. * Implement JWT-based authentication, integrate JWT with the Angular frontend. * Secure the WebSocket connections by including JWT tokens as query parameters in the connection requests for real-time features. * Manage the database schema by connecting the entities through the foreign keys and relationships, ensuring data integrity. Implement the automated email service using Java Mail Service to generate and send emails based on the specific requirements of the system. * Implement the notification service to handle the notifications for each user based on their roles in the system. * Implement file download services and connect it with the Angular frontend to facilitate the enrollment process, ensuring admin can easily access necessary documents. * Integrate file upload features with the Angular frontend to support proper submission services, allowing students to upload files as part of their submissions. * Implement set deadlines functions for submissions and connect this feature with the Angular frontend, ensuring deadlines are enforced and communicated clearly to students. |
| E/20/211 | * Designed web pages based on provided wireframes. * Built responsive web pages using Bootstrap and jQuery. * Connected web pages to the database using PHP and Spring Boot. * Implemented the front-end using Angular to support role-based user management. * Created components and developed front-end web pages incrementally. * Designed front-end using Bootstrap, jQuery, and CSS for better aesthetics and usability. * Ensured front-end modules enhance security through backend integration. * Handled tokens in local storage for user authority management on the front end. * Developed the Student Research Page functionality (e.g., handling student submissions, feedback) with Spring Boot backend. * Built backend services for managing supervisor assignments by admins and integrated them with the front end. * Added functionality for the admin to add staff members, implemented backend services, and integrated them with the front end. * Developed backend functions for various admin tasks (e.g., assigning supervisors, managing staff members). * Managed multiple user roles effectively. * Designed and developed a mobile-friendly version of the website. * Created and managed submission types for various user needs. * Added tab sections for user profiles. * Built event creation and deletion features for the calendar. * Implemented session management to handle user inactivity timeouts. * Collaborated front-end and back-end seamlessly to create a cohesive and functional system. |
| E/20/212 | * Designed and developed (CRUD) functionality for key modules such as student management. * Designed and optimized database schemas in MySQL ensuring proper integration with the database. * Implemented session management to maintain user authentication and session timeout. * Developed an automated email notification system, allowing the system to send important events like student registration. * Designed and implemented a custom function to generate unique registration numbers for postgraduate students during their enrollment process. |
| E/20/242 | * Developed a secure and user-friendly feature that allows students to upload their reports or theses directly to the system * Designed and integrated a system enabling authorized users to securely download submitted files. |

## Lessons learnt During the Course of the Project

The second-year project paved the path to learning really valuable lessons. The project was a great opportunity to gain experience while carrying out all the activities in the Software Development Life Cycle that is developing a working software solution starting from the initial customer requirements. The following are the notable lessons learnt during the project.

This was a great opportunity to improve our technical knowledge in areas such as web development languages and database management techniques.

Another good lesson was managing pressure and stress while meeting various deadlines as well as when things are not going according to the initial plan.

How to plan and organize the work and how to carry out the tasks by following the guidelines given.

## Declaration

We hereby declare that the particulars and work presented in this report are true and accurate to the best of my understanding and knowledge.

|  |  |  |
| --- | --- | --- |
| E Number | Name | Signature |
| E/20/197 | Kawya A.H.D. |  |
| E/20/211 | Kumarasinghe K.W.I.T. |  |
| E/20/212 |  |  |
| E/20/242 | Malinga G.A.I. |  |

# References

1. Spring.io. (2024). *Sending Email :: Spring Boot*. [online] Available at: <https://docs.spring.io/spring-boot/reference/io/email.html> .
2. Ram (2023). *Registration and Login Authentication with Database using SpringBoot…!* [online] Medium. Available at: <https://ramakrishna-01.medium.com/registration-and-login-authentication-with-database-using-springboot-a47e3782658d> .
3. docs.spring.io. (n.d.). *1. Working with Spring Data Repositories*. [online] Available at: <https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html>.
4. Spring.io. (2019). *Spring Projects*. [online] Available at: <https://spring.io/projects/spring-security>.
5. Getting Started | Securing a Web Application. (n.d.). *Getting Started | Securing a Web Application*. [online] Available at: <https://spring.io/guides/gs/securing-web>.
6. Devdocs.io. (2024). *DevDocs — Angular documentation*. [online] Available at: <https://devdocs.io/angular/> .
7. Jscrambler. (2022). *Working with Angular Local Storage: How and When to Use It*. [online] Available at: <https://jscrambler.com/blog/working-with-angular-local-storage>

The End…