AUTO SCHEDULE ALLOCATOR

Nichele Liong

Supervisor: Dr Vidya Sudarshan
Co-Supervisor: Dr Ku Cheng Yeaw

College of Computing and Data Science
2025

NANYANG TECHNOLOGICAL UNIVERSITY

CCDS24-0726

AUTO SCHEDULE ALLOCATOR

Submitted in Partial Fulfilment of the Requirements
for the Degree of Bachelor of Science in Mathematical and Computer Sciences (Double Major)
of the Nanyang Technological University

by

Nichele Liong

College of Computing and Data Science
2025

Abstract

The allocation of teaching duties to graduate students is a critical yet time-consuming administrative task in academic institutions. The current process at the College of Computing and Data Science (CCDS) at Nanyang Technological University (NTU) involves manual handling of graduate student preferences, lab schedules, and constraints via email and Excel sheets. This system is inefficient, error-prone and difficult to scale. Thus, to address these challenges, this project proposes the Graduate Student Lab Allocation System (GSLAS), a centralized web-based portal designed to automate and streamline the lab allocation process. The portal enables students to submit preferences in a standardized format and incorporates a flexible allocation algorithm that optimizes lab assignments based on constraints like student preferences and workload fairness. GSLAS provides a scalable solution to Graduate Student Lab Allocation, with potential for future enhancements such as Tutorial allocation and integration with faculty scheduling systems.

Acknowledgements

I would like to express my appreciation to my CCDS supervisor, Dr Vidya, and SPMS supervisor, Dr Ku, for their patience and guidance throughout this project.

In addition, I would also like to thank CCDS Academic Staff, Elaine Koh, for generously sharing her expertise on the existing allocation process and providing essential materials that grounded this project in real-world needs.

Lastly, I would like to thank my partner, friends and family for their unwavering support and belief in me throughout this process.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Background and Motivation

University instructor scheduling remains a complex and time-consuming administrative task that requires balancing multiple constraints and preferences. These scheduling tasks are “exhausting and time consuming” [1] when performed manually, particularly as the number of variables and constraints increases. Furthermore, traditional manual scheduling methods are inefficient, prone to human error [2], and fail to optimize instructor workload and course assignments effectively [3]. While extensive research has been conducted on faculty staff-level scheduling, resulting in numerous proposed solutions, smaller-scale internal departmental processes such as Graduate Student Teaching Allocation have received far less attention. This is likely due to their narrower scope and lower complexity compared to institution-wide scheduling. However, Graduate Student Teaching Allocation presents its own unique set of constraints and challenges and is equally critical to the smooth functioning of academic departments and thus warrants exploration to improve efficiency and reduce administrative burden.

Graduate Student Teaching Allocation typically involves assigning graduate students to both lab sessions and tutorials as part of their teaching duties. However, this project focuses specifically on lab allocation, as it is the first and more structured phase of the process, handled by a dedicated administrator.

To address these challenges, this project proposes the development of the Graduate Student Lab Allocation System (GSLAS), a web-based portal designed to streamline and automate the lab allocation process for graduate students.

1.2 Existing System

The current manual workflow at CCDS for Graduate Student Teaching Allocation presents several challenges.

1.2.1 Communication Management

The process involves communication with over 100 graduate students for their teaching preferences and constraints via emails. This large volume of emails could result in information fragmentation and potential for oversight.

1.2.2 Data Collection and Consolidation

Administrative staff is required to manually download and compile information from numerous individual spreadsheets per semester into a master allocation document. This process is highly time-consuming, tedious, and susceptible to data transfer errors.

1.2.3 Lack of Input Validation

Despite clear instructions being given for the format of indicating their teaching preferences, graduate students often fail to adhere to them. This lack of standardization makes it difficult for the administrator to process the data efficiently and accurately.

1.2.4 Scheduling Complexity

The allocation process must account for multiple simultaneous constraints such as students' lab load preferences, course preferences and timing restrictions. Ensuring that there are no clashes or over-allocations is a complex task.

1.2.5 Limited Scalability

The current system is not scalable, as the administrative workload increases significantly with the number of students and lab sessions.

1.3 Project Objectives and Scope

The primary objective of this project is to reduce manual effort, improve accuracy, and enhance the overall efficiency of the allocation process that occurs every semester. The portal will address the challenges of the existing system by providing the following key features:

1. **Centralized Data Management**: All lab information and student preferences will be submitted and stored in a single platform, minimising the need for manual data compilation.
2. **Input Validation**: The portal will enforce standardized input formats, ensuring data consistency.
3. **Automated Allocation Algorithm:** The portal will incorporate an allocation algorithm that considers various constraints; including availability, workload balance, and student preferences, to generate optimal assignments while minimizing conflicts.
4. **Administrative Flexibility:** The administrator will have the ability to assign weights to different parameters, providing flexibility in how allocations are prioritized. In addition, the system allows for various modes of allocation: manual with validation checks, semi-automated or fully automated. This allows administrators to adapt the system to their specific needs.

By implementing GSLAS, the administrative workload associated with teaching allocations will be significantly reduced, while fairness and accuracy in assignments will be improved. The structured approach not only benefits the administrators, but also provides students with a more transparent and reliable process for managing their teaching duties.

1.4 Report Organization

In the next chapter, the technologies used for this project will be discussed. This includes the framework and programming languages used for web development.

In Chapter 3, the database management system for the Graduate Student Lab Allocation System (GSLAS) will be discussed. This includes the choice of database and the database design.

In Chapter 4, the business process diagram will be presented to highlight the key actions, flows, and interactions.

In Chapter 5, the User Authentication and Profile Management Subsystem will be discussed. This includes an overview, its functional requirements, use cases, and implementation.

In Chapter 6, the Course and Lab Management Subsystem will be discussed. This includes an overview, its functional requirements, use cases, and implementation.

In Chapter 7, the Special Requests and Preferences Management Subsystem will be discussed. This includes an overview, its functional requirements, use cases, and implementation.

In Chapter 8, the Lab Allocation Management Subsystem will be discussed. This includes an overview, its functional requirements, use cases, and implementation.

In the final Chapter 9, the limitations of this project will be discussed, followed by recommendations for future enhancements and scalability.

2 Technologies for the Project

2.1 Django Framework

Django was chosen as the primary web framework for this project due to its robustness, scalability and ease of use. As a high-level Python framework, Django follows the “batteries-included” philosophy, providing built-in features such as authentication, URL routing, and database management, which significantly reduced development time. One of its key advantages is the Object-Relational Mapping (ORM) framework, which simplifies database interactions by allowing developers to work with databases without writing complex SQL queries. Additionally, Django’s built-in admin interface provided a seamless way to manage and view data during development and testing. Django’s structured project organization of grouping functionality into clear components like URLS, views, forms, and models made it easy to learn and maintain a clean codebase. [4][5]

2.2 Python Backend

Python was chosen as the backend programming language due to its simplicity, readability, and versatility. Since Django, the chosen web framework, is built on Python, the language naturally aligned with the project’s requirements, ensuring seamless integration and efficient development.

2.3 Frontend

The frontend of the GSLAS was developed using HTML, CSS and minimal JavaScript for dynamic functionality. HTML was used for structuring the web pages, while CSS and Bootstrap 5 were employed for styling and ensuring a responsive, user-friendly interface. Bootstrap 5 was particularly beneficial as it provided pre-designed components, reducing the need for custom CSS. JavaScript was used sparingly for basic dynamic interactions, such as form dropdown menus and tabbed tables. [6]

2.4 GitHub

GitHub served as the primary tool for version control and project management throughout the development process. Its intuitive interface and powerful features such as issue tracking made it easy to manage code changes, even as a solo developer.

3 Database Management System

3.1 Choice of Database

MySQL was chosen as the database management system for this project due to its reliability, scalability, and widespread adoption. As an open-source relational database, MySQL is well-suited for handling structured data. Its robust performance and seamless integration with Django's ORM framework made it a natural choice for this project. One of MySQL's key advantages is MySQL Workbench, which provides a user-friendly interface for managing databases. It proved to be immensely helpful in running scripts, viewing table structures and data types, and searching through records. These features significantly streamlined development and debugging. Given my prior experience with MySQL and its compatibility with Django, it was the most practical and efficient choice for this project. [7]

3.2 Database Design

3.2.1 Database Design Overview

The database design for the GSLAS was carefully structured to support the system's core functionalities while ensuring scalability, data integrity, and efficient querying.

Course and Lab
These tables store detailed information about courses and their associated lab sessions. This separation allows for flexibility in managing labs independently of courses, which is crucial for handling varying lab schedules and constraints.

Student_past_assignments
This table stores historical teaching assignments that have been indicated by the student for extra consideration during the allocation process.

SpecialRequest
This table captures special requests from students such as requests to lock a certain course's labs to them as agreed with the faculty, unavailable slots and maximum teaching days a week.

TeachingPreference
This table records students' course preferences and rankings.

AllocationWeights
This table allows the administrator to customize the allocation algorithm by adjusting weights for factors like workload distribution and course variety.

Assignment
This table stores the final lab assignments, linking students to specific lab sessions.

3.2.2 Entity-Relationship Diagram

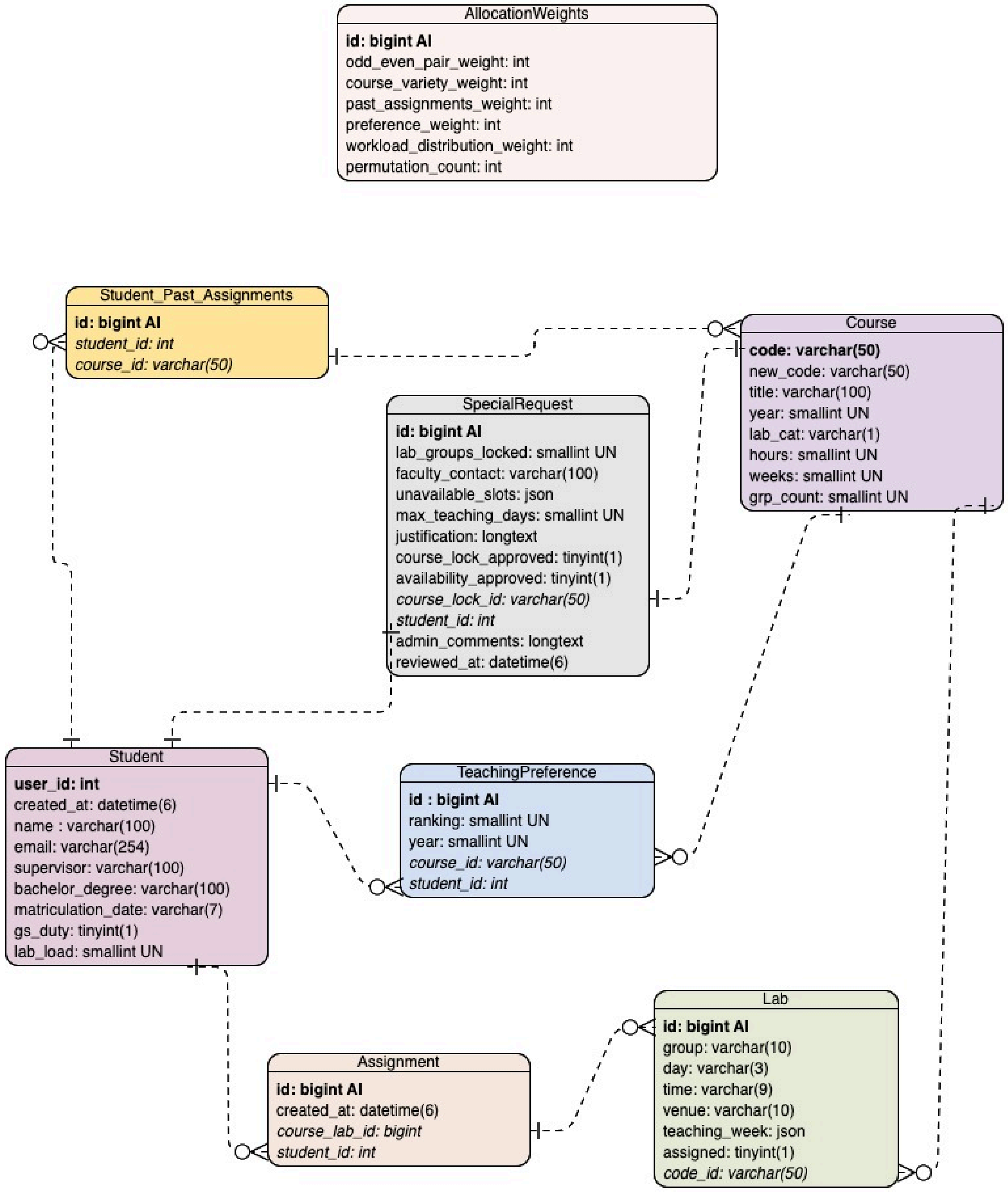


Figure 1: Entity-Relationship Diagram for GSLAS

4 Workflow

4.1. Business Process Diagram

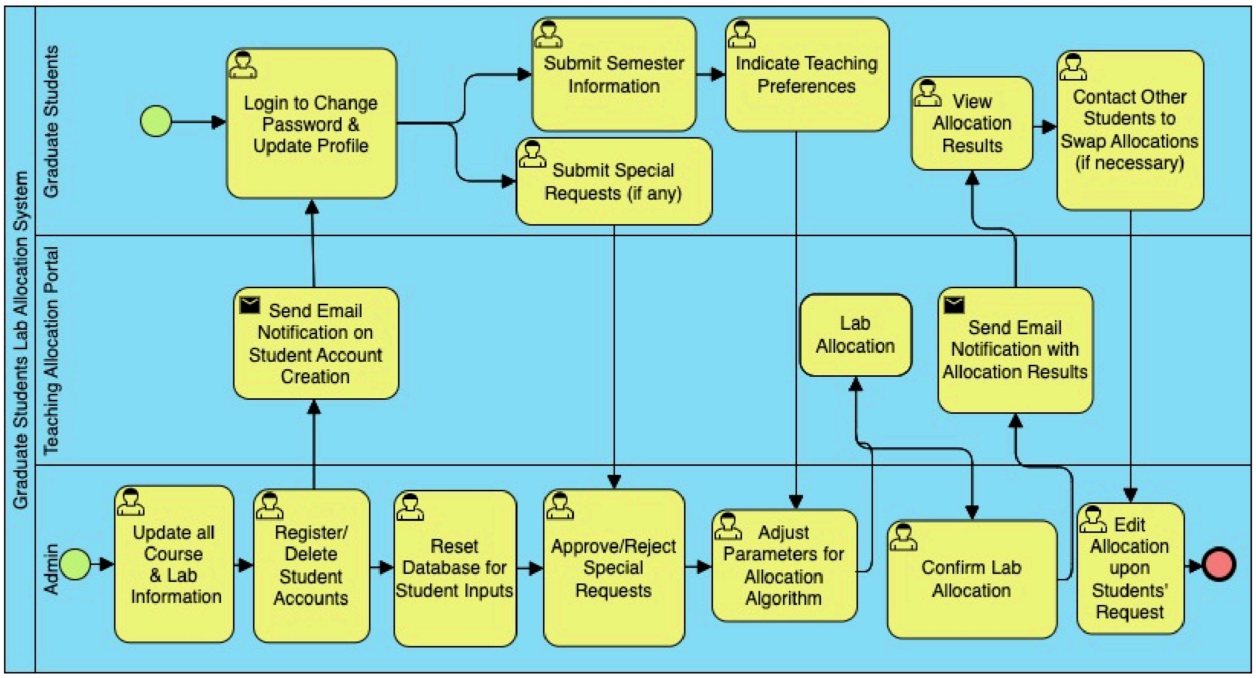

Graduate Students Lab Allocation System
Graduate Students
Teaching Allocation Portal
Admin
Login to Change Password & Update Profile
Submit Semester Information
Indicate Teaching Preferences
Submit Special Requests (if any)
View Allocation Results
Contact Other Students to Swap Allocations (if necessary)
Send Email Notification on Student Account Creation
Lab Allocation
Send Email Notification with Allocation Results
Update all Course & Lab Information
Register/ Delete Student Accounts
Reset Database for Student Inputs
Approve/Reject Special Requests
Adjust Parameters for Allocation Algorithm
Confirm Lab Allocation
Edit Allocation upon Students' Request

Figure 2: Business Process Diagram for GSLAS

The GSLAS follows a clear and structured workflow that separates the responsibilities of administrators and students. This workflow is best represented using a Business Process Diagram (BPD), as it emphasizes the user-centric flow of tasks and the interactions between different roles (admin and students). This approach is particularly suitable for GSLAS, as the system has a distinct separation between the admin and student portions of the portal.

4.2 Workflow Details

The lab allocation process begins about 2 months before the start of each semester and involves the following steps:

1. Admin Uploads Course and Lab Information
The admin updates the portal with accurate course and lab details for the upcoming semester, ensuring all information is current and correct.

2. Admin Manages Student Accounts
The admin registers new graduate students or deletes accounts for those who have graduated. During registration, the admin provides the student's first name, last name, school email address, and a username (derived from the email). The system auto-generates a random password, which

is saved in the database, and sends an email to the student with their login credentials. Students are instructed to log in and update their password and profile.

3. Admin Clears Previous Records and Notifies Students
The admin clears all records related to special requests, teaching preferences, assignments, and past assignments from the previous semester. A mass email is sent to students, notifying them to log in and submit their semester information, teaching preferences, and special requests before the deadline.

4. Students Submit Information and Preferences
Students log in to the portal to submit their semester information, indicate their teaching preferences, and submit special requests (if any).

5. Admin Reviews Special Requests
The admin reviews and chooses to approve or reject special requests. If a request is rejected, the student is notified via email and given the opportunity to resubmit with additional justification. This cycle repeats until all requests are resolved.

6. Admin Performs Lab Allocation
The admin accesses the allocation dashboard to allocate labs. The admin can adjust the algorithm's parameters (e.g. workload distribution, course variety) to prioritize specific constraints. They can choose to:

- Perform the manual allocation first, then let the system auto-fill the remaining labs
- Run the auto-allocation algorithm first, then manually edit the results with built-in validation checks

7. Admin Finalizes and Communicates Allocations
Once the admin is satisfied with the allocation, they send a mass email to students with a CSV file of the allocations. Students are also instructed to view their allocations in the portal.

8. Students Review and Swap Allocations
Students view their allocated labs and can choose to use the portal's "Contact Other Students" function to send an email to other students to negotiate swaps. After agreeing on a swap, students can notify the admin, who can update the allocations accordingly.

5 User Authentication and Profile Management Subsystem

5.1 Overview

The User Authentication and Profile Management Subsystem is responsible for managing user access and profile information within the Graduate Student Lab Allocation System (GSLAS). It ensures secure login and logout functionality, enforces role-based access control, and allows students to view and edit their profiles. The subsystem also includes administrative functions for registering and deleting student accounts. By separating the responsibilities of administrators and students, this subsystem ensures that users can only access features relevant to their roles.

5.2 Functional Requirements

1. The system must ensure that all user inputs are validated to prevent null or invalid data from being processed.

   1.1. The system must validate email addresses to ensure they are in the correct format.

   1.2. The system must ensure that usernames are unique during student registration.

2. The system must enforce password security requirements for student accounts.

   2.1. The system must require passwords to be at least 8 characters long.

   2.2. The system must ensure passwords are not entirely numeric.

   2.3. The system must ensure passwords are not a commonly used password.

   2.4. The system must ensure passwords are not too similar to the user's other personal information

   2.5. The system must ensure during password changes, the current password entered by the user matches the one stored in the database.

   2.6. The system must ensure that during password changes, the new password fields must match each other.

3. The system must ensure that the random password generated for new student accounts meets the same security requirements as user-created passwords.
4. The system must ensure that all new user accounts are being assigned to the backend group "Student".
5. The system must ensure that all inputs for matriculation date are in the format: mm/yyyy.

5.3 Use Case Diagram

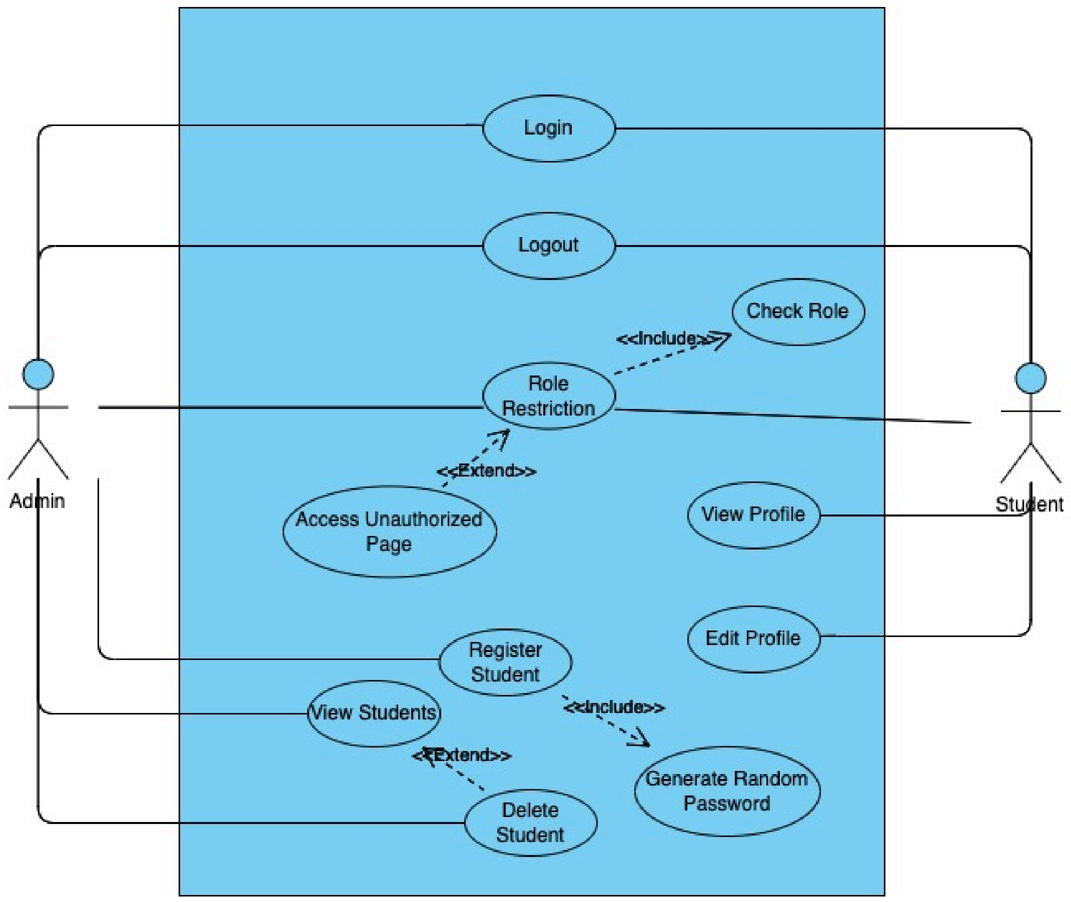


Figure 3: User Authentication and Profile Management Subsystem

5.4 Use Case Description

5.4.1 Login

Table 1: 'Login' Use Case Description

Use Case ID	UC-01
Use Case Name	Login
Actor(s)	Admin, Student
Description	Allows users to log in to the system using their username and password.

Entry Condition(s)	User is on the home page and not logged in.
Exit Condition(s)	User is redirected to their respective views (admin or student).
Flow of Events	1. User enters their username and password on the login form. 2. System authenticates the user. 3. If authentication is successful, the user is redirected to their view. 4. System displays a message: *"You Have Been Logged In!"*
Alternative Flows	AF-S3: Invalid username or password 1. System displays a message: *"There was An Error Logging In."* 2. System returns to Step 1.
Exceptions	N/A
Includes	N/A
Extends	N/A

5.4.2 Logout

Table 2: Logout Use Case Description

Use Case ID	UC-02
Use Case Name	Logout
Actor(s)	Admin, Student
Description	Allows users to log out of the system.
Entry Condition(s)	User is logged in.
Exit Condition(s)	User is redirected to the home login page.
Flow of Events	1. User clicks the logout button in the navbar. 2. System logs out the user. 3. User is redirected to the login page. 4. System displays a message: *"You Have Been Logged Out."*
Alternative Flows	N/A
Exceptions	N/A

Includes	N/A
Extends	N/A

5.4.3 Role Restriction

Table 3: Role Restriction Use Case Description

Use Case ID	UC-03
Use Case Name	Role Restriction
Actor(s)	Admin, Student
Description	Restricts access to pages based on user roles.
Entry Condition(s)	User attempts to access a restricted page.
Exit Condition(s)	User is either granted access or redirected to the unauthorized page.
Flow of Events	1. System checks the User's role. (Admin or Student) 2. If the User has the required role, access is granted. 3. If the User does not have the required role, they are redirected to the unauthorized page.
Alternative Flows	N/A
Exceptions	N/A
Includes	UC-04 (Check Role)
Extends	N/A

5.4.4 Check Role

Table 4: Check Role Use Case Description

Use Case ID	UC-04
Use Case Name	Check Role
Actor(s)	Admin, Student
Description	Verifies the User's role before granting access to a page.
Entry Condition(s)	User attempts to access a restricted page.

Exit Condition(s)	User’s role is verified.
Flow of Events	1. System checks if the User belongs to the required group (admin or student). 2. If the User belongs to the group, access is granted. 3. If not, access is denied.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

5.4.5 Access Unauthorized Page

Table 5: Access Unauthorized Page Use Case Description

Use Case ID	UC-05
Use Case Name	Access Unauthorized Page
Actor(s)	Admin, Student
Description	Redirects unauthorized Users to a custom error page.
Entry Condition(s)	User does not have the required role to access a page.
Exit Condition(s)	User is redirected to the unauthorized page.
Flow of Events	1. System detects that the User does not have the required role. 2. User is redirected to the Unauthorized page.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	UC-03 (Role Restriction)

5.4.6 Register Student

Table 6: Register Student Use Case Description

Use Case ID	UC-06

Use Case Name	Register Student
Actor(s)	Admin
Description	Allows the admin to register new student accounts.
Entry Condition(s)	Admin is logged in and on the register student page.
Exit Condition(s)	Student account is created, and an email is sent to the student.
Flow of Events	1. Admin fills out the registration form with the student's Username, First Name, Last Name, and Email. 2. System generates a random password and creates the student account. 3. System sends an email to the student with their login credentials. 4. System displays a success message: *"Student successfully registered. An email has been sent out to the student to notify them to setup their account."*
Alternative Flows	AF-S2: Existing username or Invalid email address 1. Display error message 2. System returns to Step 1.
Exceptions	N/A
Includes	UC-06 (Generate Random Password)
Extends	N/A

5.4.7 Generate Random Password

Table 7: Generate Random Password Use Case Description

Use Case ID	UC-07
Use Case Name	Generate Random Password
Actor(s)	System
Description	Automatically generates a random password for new student accounts.
Entry Condition(s)	Admin is registering a new student.
Exit Condition(s)	Random password is generated and saved.

Flow of Events	1. System generates a random password of length 8 using a combination of letters, numbers and symbols. 2. Password is saved to the database.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

5.4.8 View Students

Table 8: View Students Use Case Description

Use Case ID	UC-08
Use Case Name	View Students
Actor(s)	Admin
Description	Allows the admin to view a list of all registered students.
Entry Condition(s)	Admin is logged in and navigates to the “Students Overview” tab in navbar.
Exit Condition(s)	Admin is displayed the list of students, and the page is updated if a student is deleted.
Flow of Events	1. Admin navigates to the “Students Overview” page. 2. System retrieves the list of all registered students from the database. 3. System displays the list of students and their details (Name, Email, Supervisor, Bachelor Degree, Matriculation Date), along with a delete button for each student.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

5.4.9 Delete Student

Table 9: Delete Student Use Case Description

Use Case ID	UC-09
Use Case Name	Delete Student
Actor(s)	Admin
Description	Allows the admin to delete a student account from the system.
Entry Condition(s)	Admin is on the "Students Overview" page and clicks the delete button for a specific student.
Exit Condition(s)	Student account is deleted, and the list of students is updated.
Flow of Events	1. Admin clicks the delete button for a specific student on the "Students Overview" page. 2. System prompts the admin to confirm the deletion. 3. If the admin confirms, the system deletes the student account and all associated records from the database. 4. System updates the list of students displayed on the page. 5. System displays a success message: *"Student Record deleted successfully."*
Alternative Flows	AF-S3: Admin cancels the deletion 1. No changes are made to the database and the list remains unchanged. 2. System returns to Step 1.
Exceptions	N/A
Includes	N/A
Extends	UC-08 (View Students)

5.4.10 View Profile

Table 10: View Profile Use Case Description

Use Case ID	UC-10
Use Case Name	View Profile
Actor(s)	Student
Description	Allows students to view their profile information.

Entry Condition(s)	Student is logged in and has clicked on the “Profile & Settings” tab in the navbar.
Exit Condition(s)	Profile Information is displayed.
Flow of Events	1. System retrieves the student’s profile information from the database. 2. Profile information is displayed on the page.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

5.4.11 Edit Profile

Table 11: Edit Profile Use Case Description

Use Case ID	UC-11
Use Case Name	Edit Profile
Actor(s)	Student
Description	Allows students to edit their profile information and change their password.
Entry Condition(s)	Admin is on the “Profile & Settings” page and clicks the “Edit Profile” button.
Exit Condition(s)	Profile information is updated, and a success message is displayed.
Flow of Events	1. Student navigates to the “Edit Profile” page. 2. System displays the student’s current profile information and a password change form. 3. Student fills up the Change Password form or Profile Information form. 4. System validates the inputs. 5. If the inputs are valid, the system updates the profile or password in the database. 6. System displays a success message to the Student.
Alternative Flows	AF-S4: Invalid Input

	1. System displays an error message and prompts the student to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	UC-10 (View Students)

5.5 Implementation

User Authentication in this project is largely handled by Django's built-in authentication system and custom role-based access control.

5.5.1 User Authentication

The home view handles login functionality, authenticating users and redirecting them to their respective views.

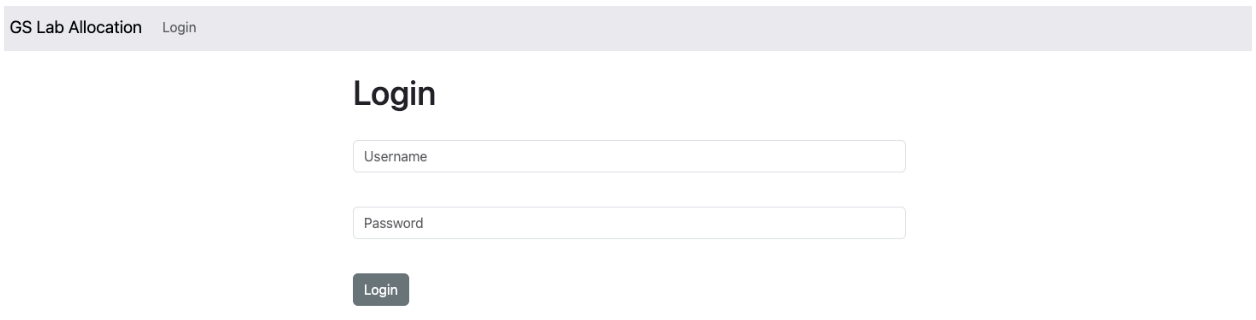
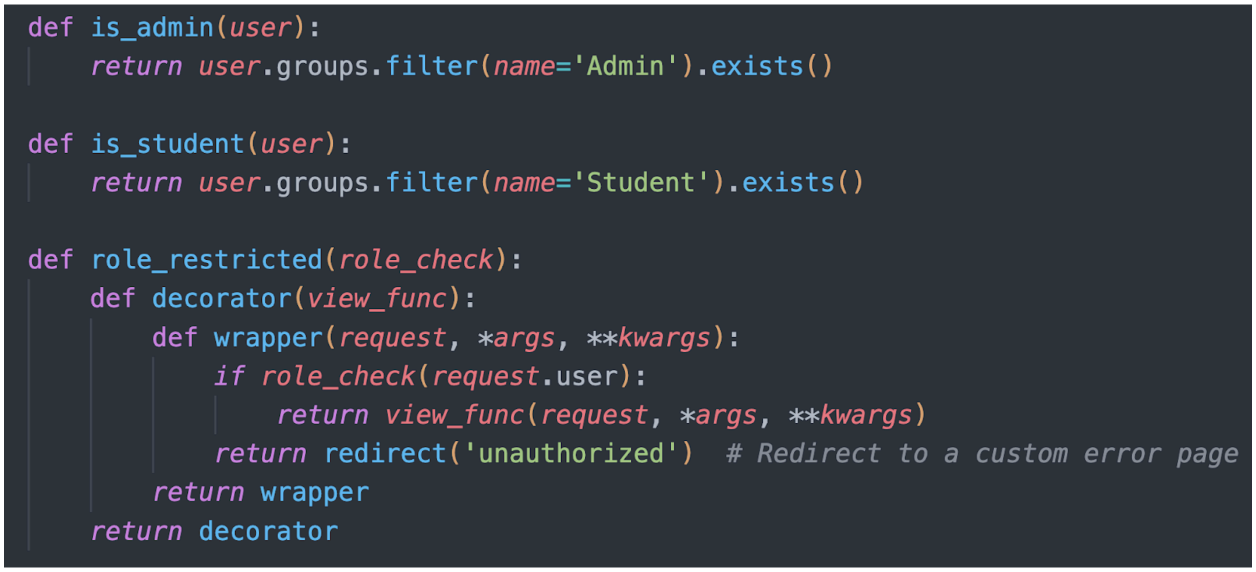


Figure 4: Login Page for GSLAS

5.5.2 Access Control

As GSLAS has a distinct separation between the admin and student portions of the portal, we make use of a custom decorator to ensure that only users with the correct role can access specific views.



```
def is_admin(user):
    return user.groups.filter(name='Admin').exists()

def is_student(user):
    return user.groups.filter(name='Student').exists()

def role_restricted(role_check):
    def decorator(view_func):
        def wrapper(request, *args, **kwargs):
            if role_check(request.user):
                return view_func(request, *args, **kwargs)
            return redirect('unauthorized')  # Redirect to a custom error page
        return wrapper
    return decorator
```

Figure 5: Implementation of role_restricted custom decorator

The system will then display different tabs on the navigation bar (NavBar) for each user group. Dropdown menus are utilized to keep the main NavBar view neat and concise.

5.5.2.1 Admin NavBar

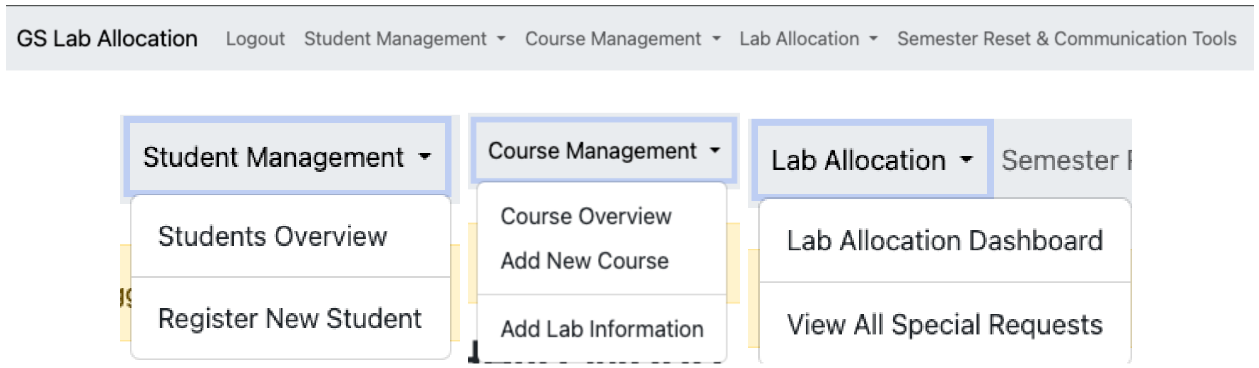


Figure 6: Admin NavBar

5.5.2.2 Student NavBar

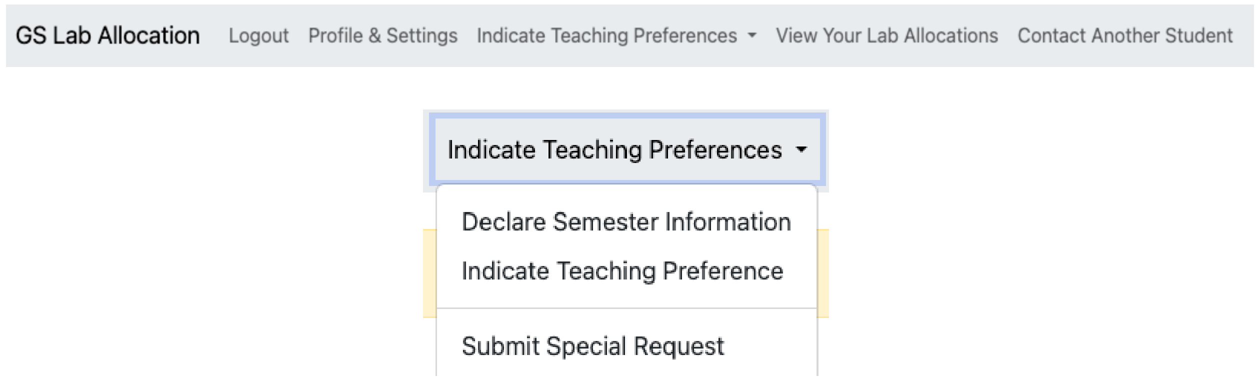
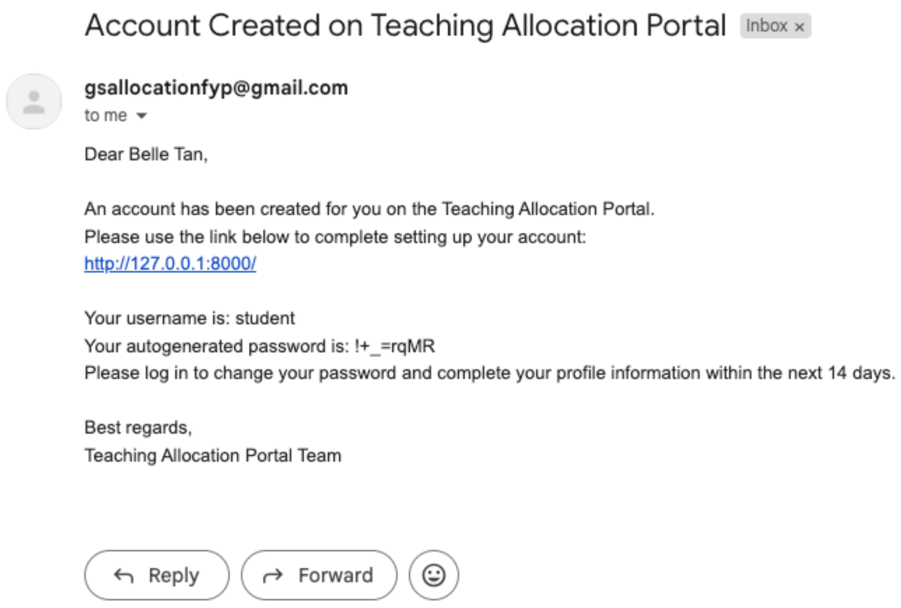

GS Lab Allocation
Logout
Profile & Settings
Indicate Teaching Preferences
View Your Lab Allocations
Contact Another Student
Indicate Teaching Preferences
Declare Semester Information
Indicate Teaching Preference
Submit Special Request

Figure 7: Student NavBar

5.5.3 Register User

When there is a new graduate student, the Admin is able to register the new user account with the student's username, name and email address. The password is an auto-generated random password, which will be sent to the newly registered student's email address.


Account Created on Teaching Allocation Portal

gsallocationfyp@gmail.com

Dear Belle Tan,

An account has been created for you on the Teaching Allocation Portal.
Please use the link below to complete setting up your account:
http://127.0.0.1:8000/

Your username is: student
Your autogenerated password is: !+_=rqMR
Please log in to change your password and complete your profile information within the next 14 days.

Best regards,
Teaching Allocation Portal Team

Figure 8: Email Sent to Student Upon Account Creation

6 Course and Lab Management Subsystem

6.1 Overview

The Course and Lab Management Subsystem is responsible for managing course and lab information within the Graduate Student Lab Allocation System (GSLAS), and can only be accessed by users with 'Admin' permissions. It allows administrators to view, add, edit, and delete courses, as well as add and manage lab sessions associated with each course. This subsystem ensures that all course and lab data is accurate and up-to-date, which is critical for the lab allocation process.

6.2 Functional Requirements

1. The system must ensure that all course codes are unique to prevent duplication in the database.
2. The system must derive new_code from the code column when a new course record is created.
3. The system must automatically update the course's group count wherever lab sessions are added, edited, or deleted.
   3.1. The system must override the grp_count field if the manual override field has been filled in by the admin.
4. The system must validate course details to ensure they meet the following criteria:
   4.1. Year: must be an integer between 1 and 3
   4.2. Lab Category: must be one of the predefined choices (C, D)
5. The system must validate lab sessions details to ensure they meet the following criteria:
   5.1. Day: must be one of the predefined choices (MON, TUE, WED, THU, FRI)
   5.2. Time: must be in the format HHMM-HHMM
   5.3. Teaching Week: must be a comma-separated list of integers, or left blank
6. The system must validate the presence of all required columns in the Excel file for add_labs.
7. The system must ensure that the sheet_name in the Excel file for add_labs matches a course in the database.

6.3 Use Case Diagram

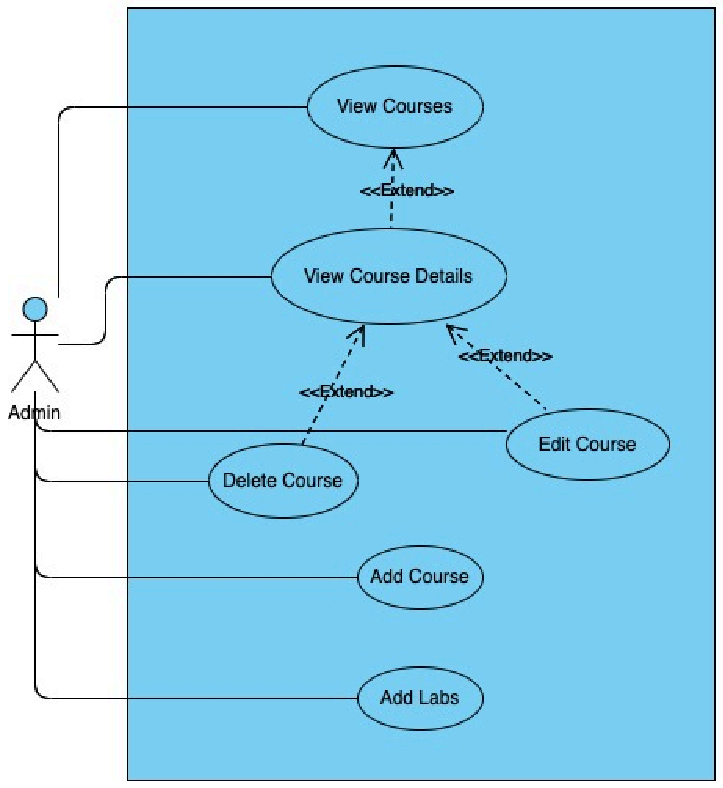


Figure 9: Use Case Diagram for Course and Lab Management Subsystem

6.4 Use Case Description

6.4.1 View Courses

Table 12: View Courses Use Case Description

Use Case ID	UC-12
Use Case Name	View Courses
Actor(s)	Admin
Description	Allows the admin to view a list of all courses.
Entry Condition(s)	Admin is logged in and navigates to the “Course Overview” page.
Exit Condition(s)	Admin is displayed the list of courses and their details.
Flow of Events	1. Admin navigates to the “Course Overview” tab in the navbar.

	2. System retrieves the list of all courses from the database. 3. System displays the list of courses and their details (Course Code, Title, Year, Lab Category, Hours, Weeks, Total Groups)
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

6.4.2 View Course Details

Table 13: View Course Details Use Case Description

Use Case ID	UC-13
Use Case Name	View Course Details
Actor(s)	Admin
Description	Allows the admin to view detailed information about a specific course, including its associated lab sessions.
Entry Condition(s)	Admin is logged in and selects a course from the “Course Overview” page.
Exit Condition(s)	Admin is displayed the course details and associated lab sessions.
Flow of Events	1. Admin selects a course code from the “Course Overview” page. 2. System retrieves the course details and associated lab sessions from the database. 3. System displays the course details and lab sessions on the “Course Details” page.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	UC-12 (View Courses)

6.4.3 Delete Course

Table 14: Delete Course Use Case Description

Use Case ID	UC-14
Use Case Name	Delete Course
Actor(s)	Admin
Description	Allows the admin to delete a course and all its associated lab sessions.
Entry Condition(s)	Admin is on the “Course Details” page and clicks the delete button.
Exit Condition(s)	Course and associated lab sessions are deleted, and the admin is redirected to the “Course Overview” page.
Flow of Events	1. Admin clicks the delete button for a specific course on the “Course Details” page. 2. The system deletes the course and all associated lab sessions from the database. 3. System redirects the admin to the “Course Overview” page and displays a success message.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	UC-13 (View Course Details)

6.4.4 Edit Course

Table 15: Edit Course Use Case Description

Use Case ID	UC-15
Use Case Name	Edit Course
Actor(s)	Admin
Description	Allows the admin to edit the details of a course and manage its associated lab sessions.
Entry Condition(s)	Admin is on the “Course Details” page and clicks the edit button.

Exit Condition(s)	Course and associated lab sessions are updated, and the admin is redirected to the “Course Overview” page.
Flow of Events	1. Admin clicks the edit button for a specific course on the “Course Details” page. 2. System retrieves the course details and associated lab sessions from the database and populates the edit form. 3. Admin updates the course details and/or lab sessions. 4. Admin submits the form. 5. System validates the inputs and updates the course and lab sessions in the database. 6. System redirects the admin to the “Course Overview” page and displays a success message.
Alternative Flows	AF-S5: Invalid Input 1. System displays an error message and prompts the admin to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	UC-13 (View Course Details)

6.4.5 Add Course

Table 16: Add Course Use Case Description

Use Case ID	UC-16
Use Case Name	Add Course
Actor(s)	Admin
Description	Allows the admin to add a new course to the database.
Entry Condition(s)	Admin is logged in and navigates to the “Add New Course” page either from the navbar or from the “Course Overview” page.
Exit Condition(s)	New course is added to the database, and the admin is redirected to the “Course Overview” page.
Flow of Events	1. Admin navigates to the “Add New Course” page. 2. Admin fills out the course details in the form. 3. Admin submits the form.

	4. System validates the inputs and adds the new course to the database. 5. System redirects the admin to the updated "Course Overview" page and displays a success message.
Alternative Flows	AF-S4: Invalid Input 1. System displays an error message and prompts the admin to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	N/A

6.4.6 Add Labs

Table 17: Add Labs Use Case Description

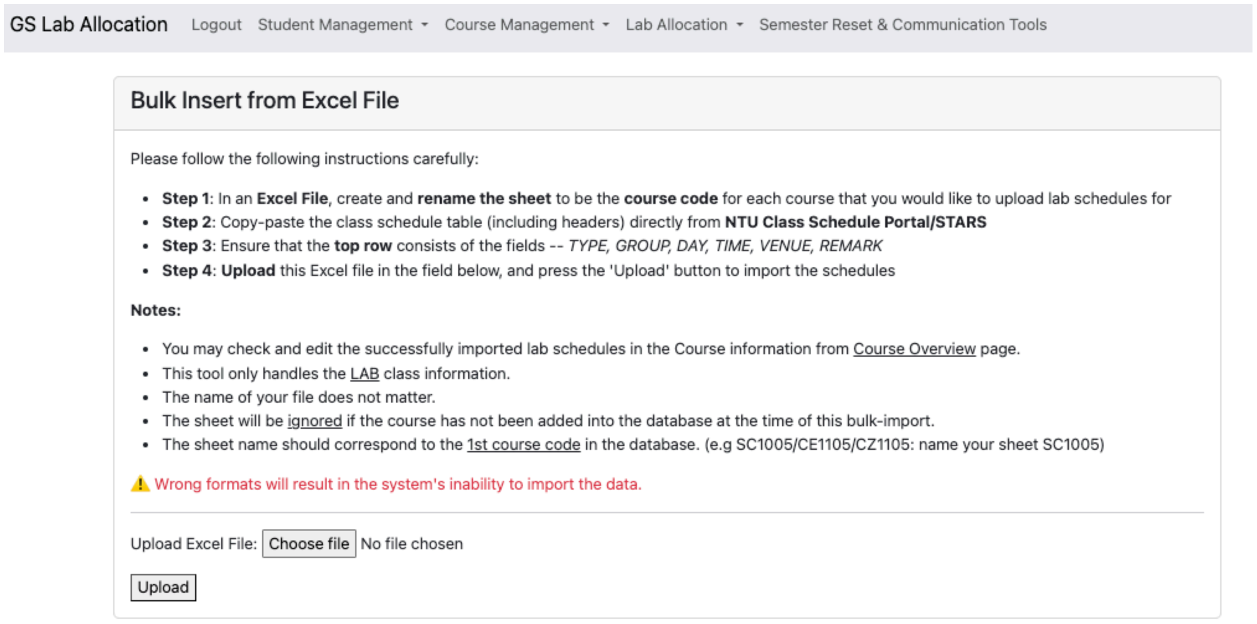
Use Case ID	UC-17
Use Case Name	Add Labs
Actor(s)	Admin
Description	Allows the admin to add lab sessions to a course by uploading an Excel file.
Entry Condition(s)	Admin is logged in and navigates to the "Add Lab Information" page from the NavBar.
Exit Condition(s)	Lab sessions are added to the database, and the admin is redirected to the "Course Overview" page.
Flow of Events	1. Admin navigates to the "Add Lab Information" page. 2. Admin uploads an Excel file according to the instructions for formatting displayed on the page. 3. Admin submits the form. 4. System validates the inputs and adds the lab sessions to the database. 5. System updates the course's group count based on the new lab sessions. 6. System redirects the admin to the "Course Overview" page and displays a success message.
Alternative Flows	AF-S4: Invalid Input

	1. System displays an error message and prompts the admin to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	N/A

6.5 Implementation

6.5.1 Add Labs

The Add Labs functionality is accessible through the “Add Lab Information” tab within the NavBar under “Course Management”. It allows administrators to bulk-upload lab sessions for multiple courses using an Excel file.



Bulk Insert from Excel File

Please follow the following instructions carefully:

- **Step 1**: In an **Excel File**, create and **rename the sheet** to be the **course code** for each course that you would like to upload lab schedules for
- **Step 2**: Copy-paste the class schedule table (including headers) directly from **NTU Class Schedule Portal/STARS**
- **Step 3**: Ensure that the **top row** consists of the fields -- *TYPE, GROUP, DAY, TIME, VENUE, REMARK*
- **Step 4**: **Upload** this Excel file in the field below, and press the 'Upload' button to import the schedules

Notes:

- You may check and edit the successfully imported lab schedules in the Course information from Course Overview page.
- This tool only handles the LAB class information.
- The name of your file does not matter.
- The sheet will be ignored if the course has not been added into the database at the time of this bulk-import.
- The sheet name should correspond to the 1st course code in the database. (e.g SC1005/CE1105/CZ1105: name your sheet SC1005)

⚠️ Wrong formats will result in the system's inability to import the data.

Upload Excel File: Choose file No file chosen

Upload

Figure 10: Add Labs page

Clear instructions are given on the required format and steps for uploading the file to import Lab information.[1]

6.5.1.1 File Upload and Validation

The admin uploads an Excel file through the *add_labs.html* form. The file is validated to ensure it contains the required columns: TYPE, GROUP, DAY, TIME, VENUE, and REMARK. The system checks if the sheet names in the Excel file match existing course codes in the database. Sheets with non-matching course codes are ignored and an error message will be displayed.

6.5.1.2 Data Processing

For each valid sheet, the system filters rows where TYPE is “LAB” and extracts the relevant lab details (group, day, time, venue, and teaching weeks). The teaching weeks are parsed from the REMARK column using the *parse_teaching_weeks* function, which converts the string into a list of integers.

[1] Refer to Appendix for example input and output

6.5.1.3 Database Update

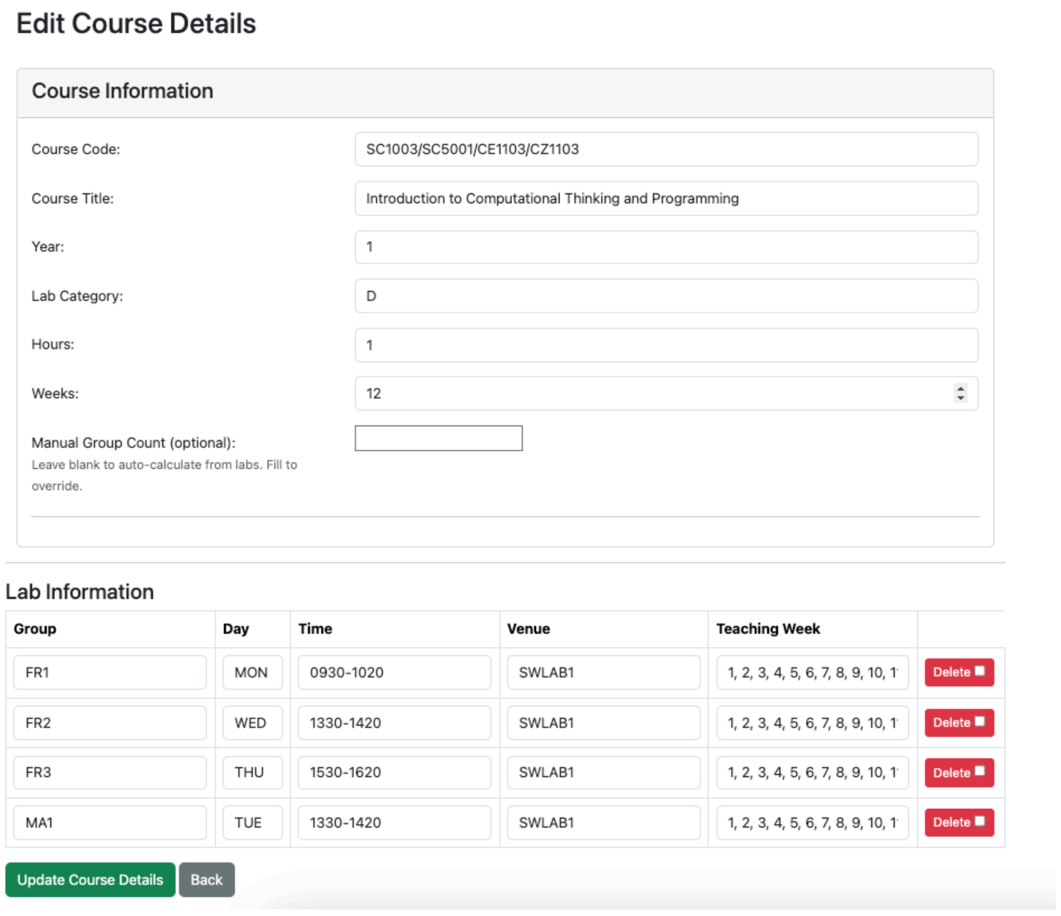
The system deletes all existing lab sessions for the courses being updated to ensure no duplicate or outdated records remain. The new lab sessions are bulk-inserted into the database for efficiency. The system updates the *grp_count* field for each course based on the number of unique lab groups.

6.5.1.4 User Feedback

If the upload is successful, the system displays a success message with the number of lab sessions added and the courses affected. If there are errors, the system displays an error message and prompts the admin to correct the file.

6.5.2 Edit Course

The Edit Course functionality can be accessed by clicking on the "Edit" button in the "Course Details" page of each course. It allows administrators to update course details and manage associated lab sessions. Here, they are allowed to manually override the *grp_count* field that has been auto-populated by the system in *add_labs*. Administrators can also select labs that they wish to delete.



Edit Course Details

Course Information

Course Code:	SC1003/SC5001/CE1103/CZ1103
Course Title:	Introduction to Computational Thinking and Programming
Year:	1
Lab Category:	D
Hours:	1
Weeks:	12
Manual Group Count (optional): Leave blank to auto-calculate from labs. Fill to override.	

Lab Information

Group	Day	Time	Venue	Teaching Week	
FR1	MON	0930-1020	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1	Delete
FR2	WED	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1	Delete
FR3	THU	1530-1620	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1	Delete
MA1	TUE	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1	Delete

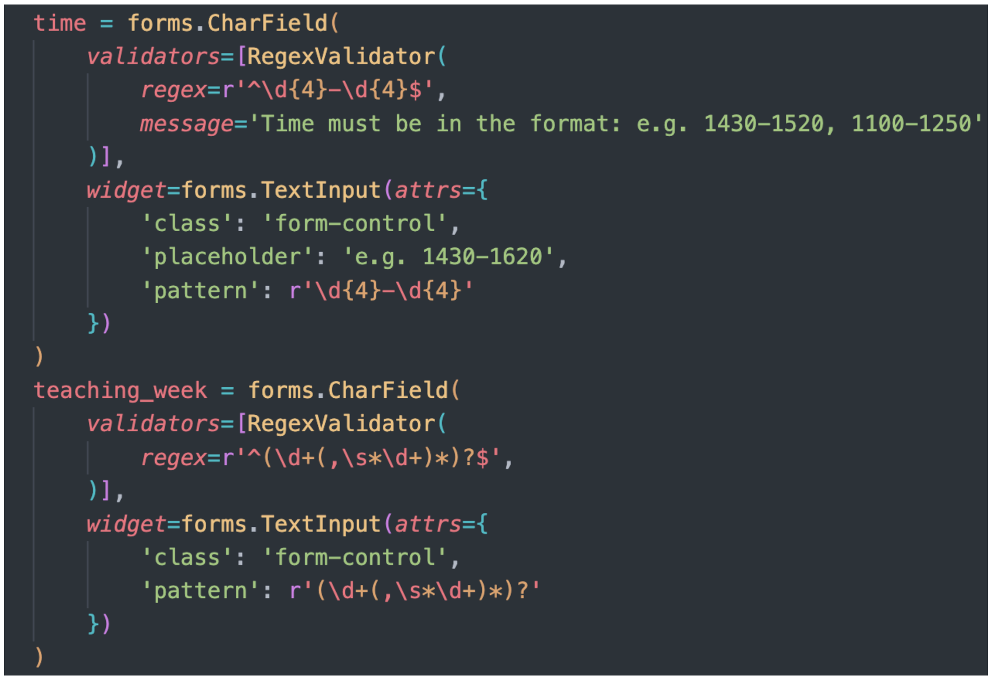
Figure 11: Edit course page

6.5.2.1 Form Initialization

The current course details and associated lab sessions are retrieved from the database and populated in the fields of the edit course form.

6.5.2.2 Form Submission and Validation

When the admin submits the form, the system validates both the course details and the lab sessions. The *time* and *teaching_week* fields use Regular Expression (RegEx) to validate the user input.



```
time = forms.CharField(
    validators=[RegexValidator(
        regex=r'^\d{4}-\d{4}$',
        message='Time must be in the format: e.g. 1430-1520, 1100-1250'
    )],
    widget=forms.TextInput(attrs={
        'class': 'form-control',
        'placeholder': 'e.g. 1430-1620',
        'pattern': r'\d{4}-\d{4}'
    })
)
teaching_week = forms.CharField(
    validators=[RegexValidator(
        regex=r'^(\d+(,\s*\d+)*)?$',
    )],
    widget=forms.TextInput(attrs={
        'class': 'form-control',
        'pattern': r'(\d+(,\s*\d+)*)?'
    })
)
```

Figure 12: RegEx validation for time and teaching_week

The RegExValidator enforces the following rules:

Time

- The input must be in the format HHMM-HHMM, where:
    - HHMM represents the start time
    - HHMM represents the end time

Teaching Week

- The input must consist of integers separated by commas (e.g. 1,2,3)
- Spaces around commas are allowed (e.g. 1, 2, 3)
- The input can be empty

6.5.2.3 Database Update

If the forms are valid, the system updates the course details and lab sessions in the database. The system checks if the admin has provided a manual override for the grp_count field. If not, it automatically calculates the group count based on the number of unique lab groups[2] at the point of form submission.

6.5.2.4 User Feedback

If the upload is successful, the system displays a success message and redirects the admin to the "Course Overview" page. If there are errors, the system displays an error message and prompts the admin to correct the file.

[2] Unique "Group" field

7 Special Requests and Preferences Management Subsystem

7.1 Overview

The Special Requests and Preferences Management Subsystem allows students to submit their semester information, teaching preferences, and special requests, while enabling administrators to review and approve or reject these requests. This subsystem ensures that student-specific constraints and preferences are captured and considered during the lab allocation process. It also provides a structured workflow for admins to manage special requests efficiently.

7.2 Functional Requirements

1. The system must ensure that the student can only have one declaration of semester information at a time. (keep latest)
2. The system must ensure that the student can only have one set of teaching preferences at a time. (keep latest)
3. The system must ensure that the student can only have one active special request at a time. (keep latest)
4. The system must validate that students rank at least 3 courses per year when submitting teaching preferences.
5. The system must enforce that students cannot assign the same ranking to more than 2 courses.
6. The system must ensure that admins can review special requests and provide comments or justification for their decisions.
7. The system must automatically send an email to students if any part of their special request is rejected, prompting them to resubmit with better justification.
8. The system must ensure that students who opt out of graduate student teaching duty (gs_duty=False) have their lab_load set to 0.
9. The system must ensure that the default lab_load is set to 4.
10. The system must ensure that if the max_teaching_days or unavailable_slots have been filled in, the justification field must be filled in as well.
11. The system must ensure that if a student has selected a course to lock, the lab_groups_locked field and faculty_contact field must be filled in as well.
12. The system must ensure that only a maximum of 8 unavailable time slots are selected.
13. The system must prevent database reset unless the admin enters "CONFIRM."
14. The system must ensure emails are only sent if both subject and message fields are filled.
15. The system must allow admin to view the list of students before selecting recipients for targeted emails.
16. The system must allow students to view the list of current courses before selecting past teaching assignments.

7.3 Use Case Diagram

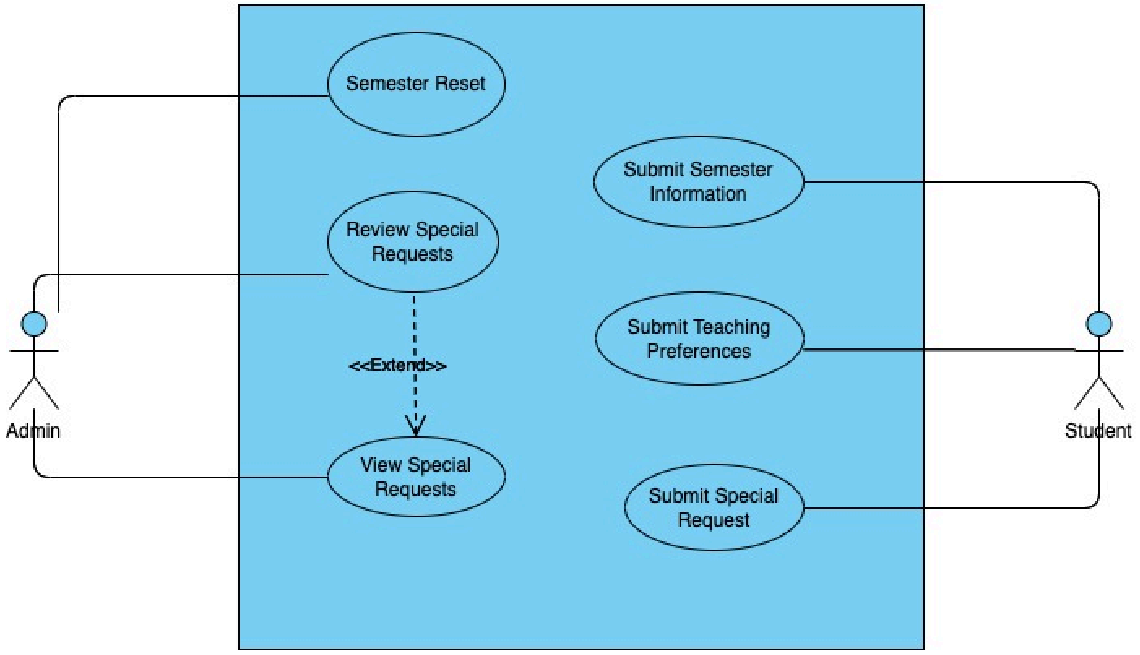


Figure 13: Use Case Diagram for Special Requests and Preferences Management Subsystem

7.4 Use Case Description

7.4.1 Submit Semester Information

Table 18: Submit Semester Information Use Case Description

Use Case ID	UC-18
Use Case Name	Submit Semester Information
Actor(s)	Student
Description	Allows students to submit their semester information, including whether they wish to participate in the lab allocation exercise (gs_duty), desired lab load and their past teaching assignments.
Entry Condition(s)	Student is logged in and navigates to the "Semester Information" page.
Exit Condition(s)	Semester information is saved, and the student is redirected to the next step (Teaching Preferences or Home).
Flow of Events	1. Student navigates to the "Semester Information" page.

	2. System retrieves the student’s current information (excluding past_assigments) and displays it in the form. 3. Student updates their semester information and selects as many past teaching assignments as they wish. 4. Student submits the form. 5. System validates the inputs and saves the information. 6. If the student opts out of gs_duty, they are redirected to the home page. Otherwise, they are redirected to the Teaching Preferences page.
Alternative Flows	AF-S5: Invalid Input 1. System displays an error message and prompts the student to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	N/A

7.4.2 Submit Teaching Preferences

Table 19: Submit Teaching Preferences Use Case Description

Use Case ID	UC-19
Use Case Name	Submit Teaching Preferences
Actor(s)	Student
Description	Allows students to rank their preferred courses for teaching assignments.
Entry Condition(s)	Student is logged in and navigates to the “Teaching Preferences” page, either from NavBar or after submitting “Semester Information”.
Exit Condition(s)	Teaching preferences are saved, and the student is redirected to the home page.
Flow of Events	1. Student navigates to the “Teaching Preferences” page. 2. System retrieves the list of available courses and displays them in the form. 3. Student ranks their preferred courses according to the instructions specified. 4. Student submits the form.

	5. System validates the inputs, ensuring that: ○ At least 3 courses per year are ranked. ○ Rankings are integer numbers from 1 to 8. ○ No ranking is assigned to more than 2 courses. 6. If the inputs are valid, the system saves the preferences and displays a success message.
Alternative Flows	AF-S5: Invalid Input 1. System displays an error message and prompts the student to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	N/A

7.4.3 Submit Special Request

Table 20: Submit Special Request Use Case Description

Use Case ID	UC-20
Use Case Name	Submit Special Request
Actor(s)	Student
Description	Allows students to submit special requests, such as unavailable time slots, maximum teaching days, or course lock requests.
Entry Condition(s)	Student is logged in and navigates to the “Special Request” page from NavBar.
Exit Condition(s)	Special request is saved, and the student is redirected to the home page.
Flow of Events	1. Student navigates to the “Special Request” page. 2. System retrieves the student’s current special request (if any) and displays it in the form. 3. Student updates their special request details (e.g., unavailable slots, course lock, maximum teaching days). 4. Student submits the form. 5. System validates the inputs, ensuring that: ○ All required fields are filled out. 6. If the inputs are valid, the system saves the special request and displays a success message.

Alternative Flows	AF-S5: Invalid Input 1. System displays an error message and prompts the student to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	N/A

7.4.4 View Special Requests

Table 21: View Special Requests Use Case Description

Use Case ID	UC-21
Use Case Name	View Special Requests
Actor(s)	Admin
Description	Allows admins to view a list of all special requests submitted by students.
Entry Condition(s)	Admin is logged in and navigates to the “View Special Requests” page.
Exit Condition(s)	Admin is displayed the list of special requests.
Flow of Events	1. Admin navigates to the “View Special Requests” page. 2. System retrieves all special requests submitted by students from the database. 3. System displays the list of special requests, including student details, request status and an action button to review the corresponding special request.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

7.4.5 Review Special Request

Table 22: Review Special Request Use Case Description

Use Case ID	UC-22
Use Case Name	Review Special Request
Actor(s)	Admin
Description	Allows Admin to review and approve or reject a specific special request.
Entry Condition(s)	Admin is on the “View Special Requests” page and selects a request to review.
Exit Condition(s)	Special request is updated, and the admin is redirected to the “View Special Requests” page.
Flow of Events	1. Admin selects a special request to review and clicks the action button on the “View Special Requests” page. 2. System retrieves the request details and displays them in the review form. 3. Admin toggles the fields to approve/reject course lock request and availability request separately. Admin can provide comments if necessary. 4. Admin submits the form. 5. System updates the request and logs the review timestamp. 6. If any part of the request is rejected, the system sends an email to the student with the review outcome and instructions to resubmit. 7. System redirects the admin to the “View Special Requests” page and displays a success message.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	UC-21 (View Special Requests)

7.4.6 Semester Reset

Table 23: Semester Reset Use Case Description

Use Case ID	UC-23

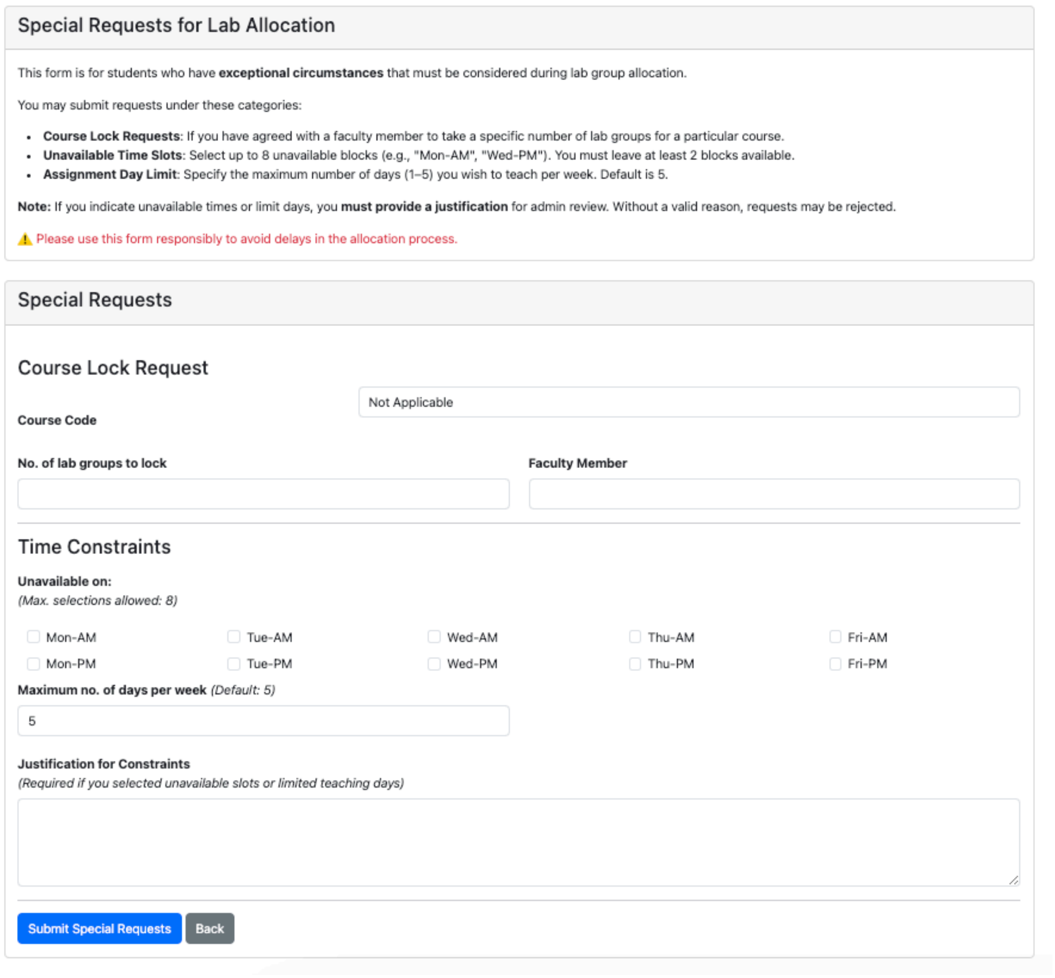
Use Case Name	Semester Reset
Actor(s)	Admin
Description	Allows the admin to prepare the system for a new semester by sending emails to students and resetting the database.
Entry Condition(s)	Admin is logged in and navigates to the “Semester Reset” page.
Exit Condition(s)	Emails are sent, and the database is reset (if confirmed).
Flow of Events	1. Admin navigates to the “Semester Reset” page. 2. Admin performs one of the following actions: 2.1. Send Email to All Students 2.1.1. Admin enters a subject and message. 2.1.2. Admin clicks “Send to all Students.” 2.1.3. System sends the email to all students and displays a success message. 2.2. Send Email to selected Students 2.2.1. Admin enters a subject and message. 2.2.2. Admin selects specific students from a dropdown list. 2.2.3. Admin clicks “Send to Selected Students”. 2.2.4. System sends the email to the selected students and displays a success message. 2.3. Reset Database 2.3.1. Admin types “CONFIRM” in the confirmation field. 2.3.2. Admin clicks “Reset Database”. 2.3.3. System deletes all records from the SpecialRequest, TeachingPreference, and Assignment tables, and clears past assignments from all students. 2.3.4. System displays a success message.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

7.5 Implementation

In this section, we expand more on the details for submitting and reviewing special requests, and submitting teaching preferences.[3]

7.5.1 Submit Special Request

The Submit Special Request functionality allows students to submit requests for course locks, unavailable time slots, and maximum teaching days.



Special Requests for Lab Allocation

This form is for students who have **exceptional circumstances** that must be considered during lab group allocation.

You may submit requests under these categories:

- **Course Lock Requests**: If you have agreed with a faculty member to take a specific number of lab groups for a particular course.
- **Unavailable Time Slots**: Select up to 8 unavailable blocks (e.g., "Mon-AM", "Wed-PM"). You must leave at least 2 blocks available.
- **Assignment Day Limit**: Specify the maximum number of days (1–5) you wish to teach per week. Default is 5.

Note: If you indicate unavailable times or limit days, you **must provide a justification** for admin review. Without a valid reason, requests may be rejected.

⚠ Please use this form responsibly to avoid delays in the allocation process.

Special Requests

Course Lock Request

Course Code: Not Applicable

No. of lab groups to lock

Faculty Member

Time Constraints

Unavailable on:
(Max. selections allowed: 8)

☐ Mon-AM	☐ Tue-AM	☐ Wed-AM	☐ Thu-AM	☐ Fri-AM
☐ Mon-PM	☐ Tue-PM	☐ Wed-PM	☐ Thu-PM	☐ Fri-PM

Maximum no. of days per week *(Default: 5)*: 5

Justification for Constraints
(Required if you selected unavailable slots or limited teaching days)

Figure 14: Submit Special Request page

Course Lock

If a course has been selected for locking, the system requires the student to specify the number of lab groups to lock and provide the name of the faculty member that they have arrived at this course lock agreement with.

Unavailable Slots

Students can select up to 8 unavailable time slots. If more than 8 slots are selected, the system displays an error.

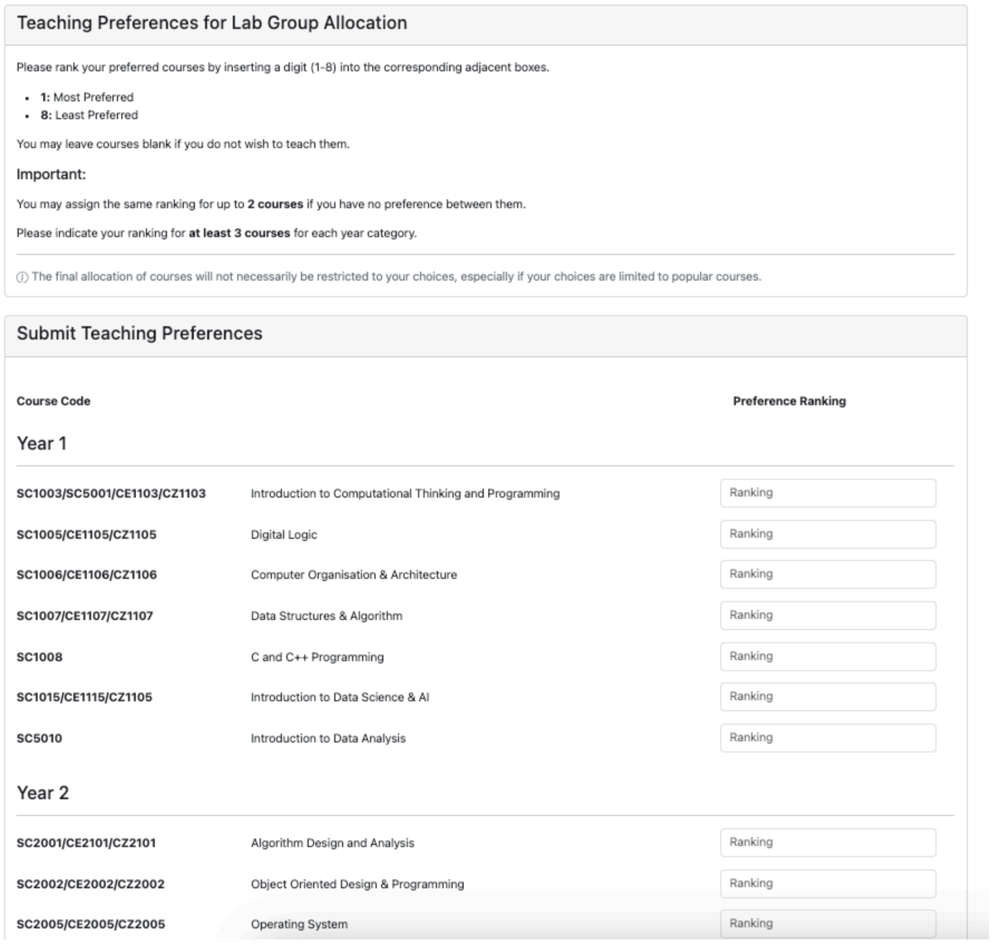
[3] Refer to Appendix for screen capture of submit semester information and view special requests.

Maximum Teaching Days
The default is 5 days (Mon-Fri), but students can reduce this to a minimum of 1 day if necessary.

Justification
This field is compulsory if the student has specified any unavailable slots or has limited their teaching days below the default of 5 days. Otherwise, the form is invalid. This is to ensure proper use of the form.

7.5.2 Submit Teaching Preferences

The Submit Teaching Preferences functionality allows students to rank their preferred courses for teaching assignments.



Teaching Preferences for Lab Group Allocation

Please rank your preferred courses by inserting a digit (1-8) into the corresponding adjacent boxes.

- **1:** Most Preferred
- **8:** Least Preferred

You may leave courses blank if you do not wish to teach them.

Important:

You may assign the same ranking for up to **2 courses** if you have no preference between them.

Please indicate your ranking for **at least 3 courses** for each year category.

ⓘ The final allocation of courses will not necessarily be restricted to your choices, especially if your choices are limited to popular courses.

Submit Teaching Preferences

Course Code		Preference Ranking
Year 1		
SC1003/SC5001/CE1103/CZ1103	Introduction to Computational Thinking and Programming	Ranking
SC1005/CE1105/CZ1105	Digital Logic	Ranking
SC1006/CE1106/CZ1106	Computer Organisation & Architecture	Ranking
SC1007/CE1107/CZ1107	Data Structures & Algorithm	Ranking
SC1008	C and C++ Programming	Ranking
SC1015/CE1115/CZ1105	Introduction to Data Science & AI	Ranking
SC5010	Introduction to Data Analysis	Ranking
Year 2		
SC2001/CE2101/CZ2101	Algorithm Design and Analysis	Ranking
SC2002/CE2002/CZ2002	Object Oriented Design & Programming	Ranking
SC2005/CE2005/CZ2005	Operating System	Ranking

Figure 15: Submit Teaching Preferences page

7.5.2.1 Form Structure

The form dynamically generates ranking fields for each course, grouped by year (1, 2, or 3). Students can rank courses from 1 (most preferred) to 8 (least preferred) or leave them blank if they do not wish to teach the course.

7.5.2.2 Validation Logic

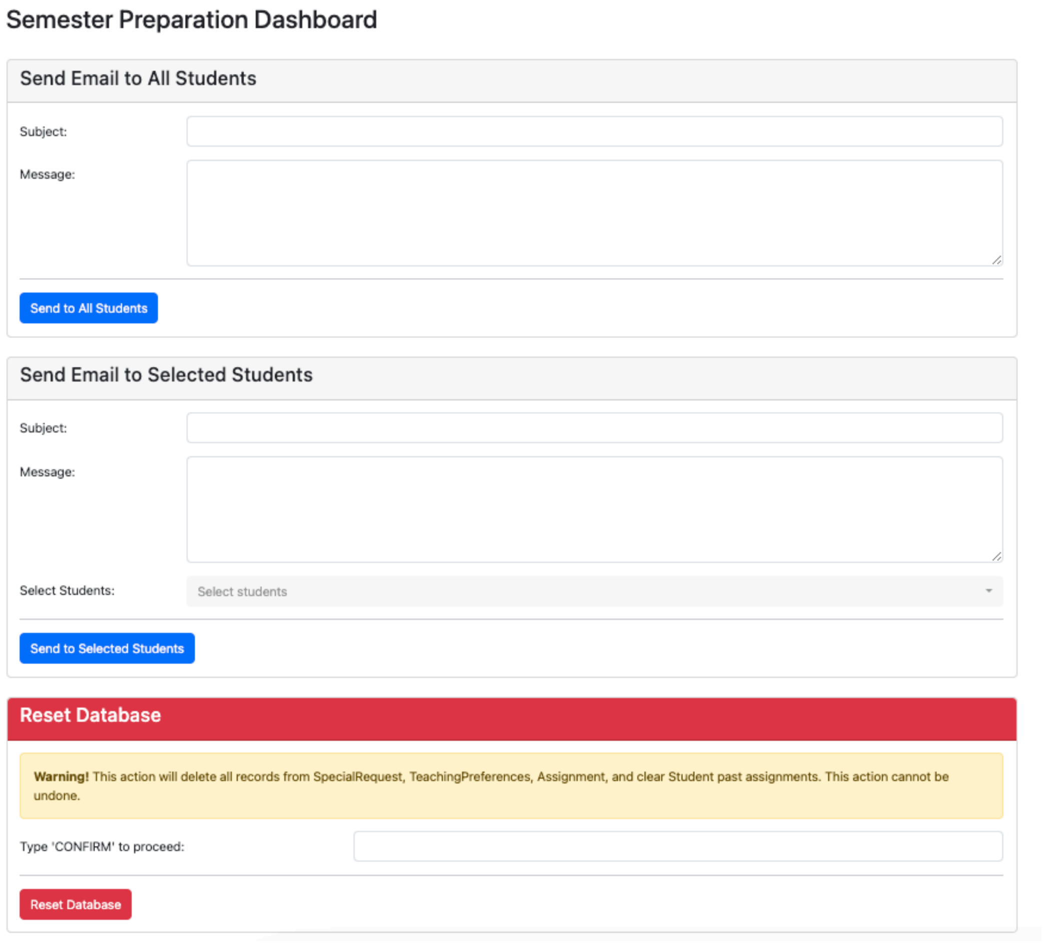
Students must rank at least 3 courses per year and cannot assign the same ranking to more than 2 courses. If any of these rules are violated, the system displays an error.

7.5.2.3 Database Update

If the form is valid, the system deletes the student's existing preferences and saves the new rankings to the database.

7.5.3 Semester Reset

The Semester Reset functionality allows admins to prepare the system for a new semester by sending emails to students and resetting parts of the database that contain student inputs for each semester.


Semester Preparation Dashboard

Send Email to All Students

Subject:

Message:

Send Email to Selected Students

Subject:

Message:

Select Students: Select students

Reset Database

Warning! This action will delete all records from SpecialRequest, TeachingPreferences, Assignment, and clear Student past assignments. This action cannot be undone.

Type 'CONFIRM' to proceed:

Figure 16: Semester Reset page

7.5.3.1 Send Email to all Students

This function allows the admin to enter a subject and message in the form, and then send the email to all registered students in the system's database.

7.5.3.2 Send Email to Selected Students

This function allows the admin to enter a subject and message in the form , and select multiple students from a dropdown list. When the “Send to Selected Students” button is clicked, the system sends the email only to the selected students.

7.5.3.3 Reset Database

This function allows the admin to delete all student records pertaining to the previous semester’s lab allocation cycle, in preparation for the next lab allocation cycle. These include SpecialRequest, TeachingPreference, and Assignment tables, as well as past_assignments fields from all students. As this is an irreversible function, the admin has to type “CONFIRM” in the confirmation field and click the “Reset Database” button. This is to ensure that there is no accidental deletion of the database records

8 Lab Allocation Management Subsystem

8.1 Overview

The Lab Allocation Management Subsystem is the core component of the system that handles the automated assignment of students to lab sessions while considering multiple constraints and optimization criteria. This subsystem provides administrators with tools to run the allocation algorithm, view results in different formats, manually adjust assignments, and notify students of their allocations. For students, it offers visibility into their assigned labs and the ability to contact peers about their allocations. The system employs a penalty-based optimization algorithm that considers teaching preferences, past assignments, workload distribution, and various constraints to produce fair and efficient lab allocations.

8.2 Functional Requirements

1. The system must ensure that lab groups (all sessions of the same lab) are assigned to the same student.
2. The system must validate admin manual assignments against student constraints before saving.
3. The system must calculate and display real-time statistics about allocation completeness and quality.
4. The system must allow administrators to adjust optimization weights that influence the allocation algorithm.
5. The system must implement multiple optimization criteria (odd/even pairing, preference ranking, past assignments, workload distribution) with configurable weights.
6. The system must run multiple permutations of the allocation algorithm to find the solution with the lowest penalty score, where the number of permutations are specified by the admin.
7. The system must handle course lock requests by assigning specified labs to students before general allocation.
8. The system must prevent time clashes when assigning labs to students; considering teaching week, day and time.
9. The system must ensure students with higher lab loads are prioritized in the allocation process.

8.3 Use Case Diagram

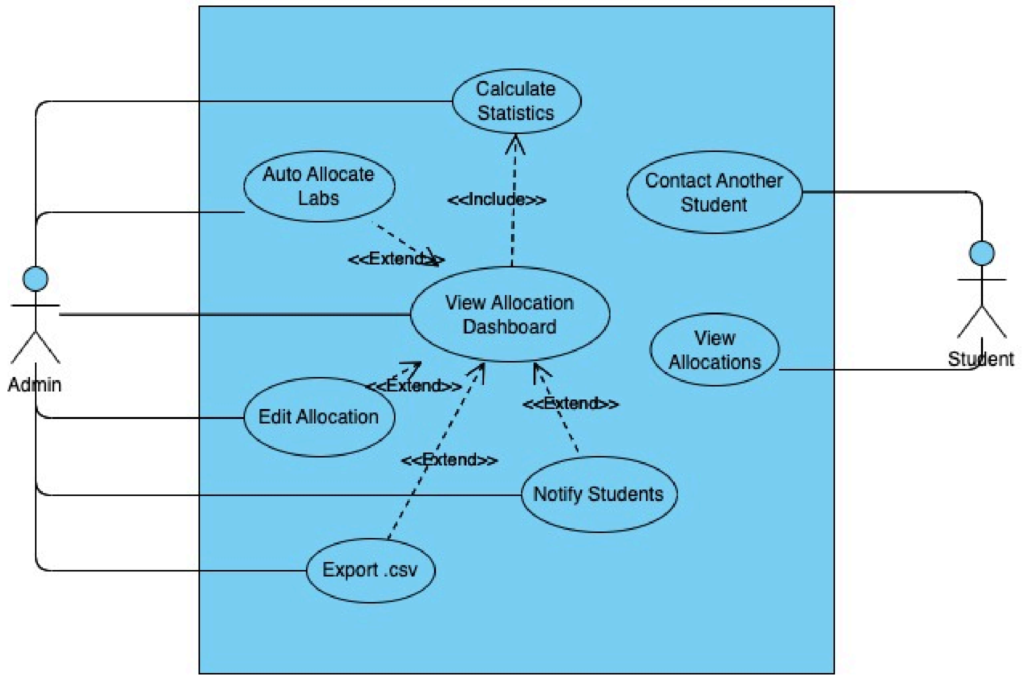


Figure 17: Use Case Diagram for Lab Allocation Management Subsystem

8.4 Use Case Description

8.4.1 View Allocation Dashboard

Table 24: View Allocation Dashboard Use Case Description

Use Case ID	UC-24
Use Case Name	View Allocation Dashboard
Actor(s)	Admin
Description	Allows the admin to view the allocation dashboard with statistics and allocation results.
Entry Condition(s)	Admin is logged in and navigates to the “Allocation Dashboard” page.
Exit Condition(s)	Admin views the dashboard with allocation statistics and results.
Flow of Events	1. System retrieves current allocation statistics. 2. System displays summary statistics cards.

	3. System displays allocation weights and permutation count form. 4. System displays allocation options (manual edit, clear assignments, auto-allocate) 5. System displays edit allocation and export CSV buttons 6. System displays allocation results in tabbed view (student view and course view) 7. Admin can interact with any of these components.
Alternative Flows	N/A
Exceptions	N/A
Includes	UC-25 (Calculate Statistics)
Extends	N/A

8.4.2 Calculate Statistics

Table 25: Calculate Statistics Use Case Description

Use Case ID	UC-25
Use Case Name	Calculate Statistics
Actor(s)	System
Description	System calculates and updates allocation statistics.
Entry Condition(s)	Allocation dashboard is loaded or allocations are updated.
Exit Condition(s)	Current statistics are available for display.
Flow of Events	1. System queries database for total students with gs_duty=True. 2. System counts students who have met their lab load requirement. 3. System counts total courses and labs in the system. 4. System calculates percentage of labs assigned. 5. System returns these statistics for display.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

8.4.3 Auto Allocate Labs

Table 26: Auto Allocate Labs Use Case Description

Use Case ID	UC-26
Use Case Name	Auto Allocate Labs
Actor(s)	Admin
Description	Runs the automatic allocation algorithm to assign students to labs.
Entry Condition(s)	Admin is on the allocation dashboard and clicks “Run Automatic Allocation”.
Exit Condition(s)	Allocation is completed and results are displayed.
Flow of Events	1. Admin selects whether to clear existing assignments. 2. Admin clicks the "Run Automatic Allocation" button. 3. System executes the allocation algorithm with current weights. 4. System runs multiple permutations to find the best solution, where the permutation count is decided by Admin. 5. System saves the best allocation (minimum penalty score) to the database. 6. System displays the penalty score along with other allocation statistics. 7. System displays a success message with the penalty score of the best allocation found.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	UC-24 (View Allocation Dashboard)

8.4.4 Edit Allocation

Table 27: Edit Allocation Use Case Description

Use Case ID	UC-27
Use Case Name	Edit Allocation
Actor(s)	Admin

Description	Allows admin to manually edit lab assignments.
Entry Condition(s)	Admin navigates to the edit allocation page, either by clicking the “Edit Allocations” button in the “Allocation Results” table or “Go to Manual Allocation” button in the “Allocation Options” card.
Exit Condition(s)	Manual edits are saved to the database.
Flow of Events	1. Admin selects a course from the dropdown. 2. System displays all lab groups for the selected course. 3. For each lab group, the system shows the list of all labs, along with the student that is being assigned to it currently. 4. Admin can change assignments using dropdowns or mark for deletion. 5. Admin clicks “Save Changes”. 6. System validates each assignment against constraints. 7. System saves valid assignments to database. 8. System redirects to dashboard with success message.
Alternative Flows	AF-S6: Invalid Input 1. System displays an error message and prompts the admin to correct the input. 2. System returns to Step 2.
Exceptions	N/A
Includes	N/A
Extends	UC-24 (View Allocation Dashboard)

8.4.5 Export CSV

Table 28: Export CSV Use Case Description

Use Case ID	UC-28
Use Case Name	Export CSV
Actor(s)	Admin
Description	Exports current allocations to a CSV file.
Entry Condition(s)	Admin clicks “Export Allocations” button.
Exit Condition(s)	CSV file is downloaded to Admin’s device.
Flow of Events	1. System queries database for all courses and their labs.

	2. For each lab, system includes assignment information if exists. 3. System formats data into CSV structure. 4. System generates HTTP response with CSV file. 5. Browser downloads the file.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	UC-24 (View Allocation Dashboard)

8.4.6 Notify Students

Table 29: Notify Students Use Case Description

Use Case ID	UC-29
Use Case Name	Notify Students
Actor(s)	Admin
Description	Sends allocation notifications to all students via email.
Entry Condition(s)	Admin clicks the “Confirm Allocations & Notify Students” button.
Exit Condition(s)	Emails are sent to all students with their allocations.
Flow of Events	1. Admin clicks "Confirm Allocations & Notify Students". 2. System shows confirmation dialog. 3. Admin confirms action. 4. System generates CSV of all allocations. 5. System composes email message with CSV attachment. 6. System sends email to all students involved in the current allocation exercise (gs_duty=True) 7. System shows success message with count of emails sent.
Alternative Flows	N/A
Exceptions	If email sending fails, system shows error message.
Includes	N/A
Extends	UC-24 (View Allocation Dashboard)

8.4.7 View Allocations

Table 30: View Allocation Use Case Description

Use Case ID	UC-30
Use Case Name	View Allocations
Actor(s)	Student
Description	Allows Student to view their assigned labs and all student allocations.
Entry Condition(s)	Student is logged in and navigates to the allocations page.
Exit Condition(s)	Student views their allocations or all student allocations.
Flow of Events	1. System retrieves logged-in student's assignments for “Your Assignments” tab. 2. System retrieves all student assignments for "All Student Assignments" tab. 3. System displays assignments in tabbed interface. 4. Student can switch between "Your Assignments" and "All Student Assignments" tabs.
Alternative Flows	N/A
Exceptions	N/A
Includes	N/A
Extends	N/A

8.4.8 Contact Another Student

Table 31: Contact Another Student Use Case Description

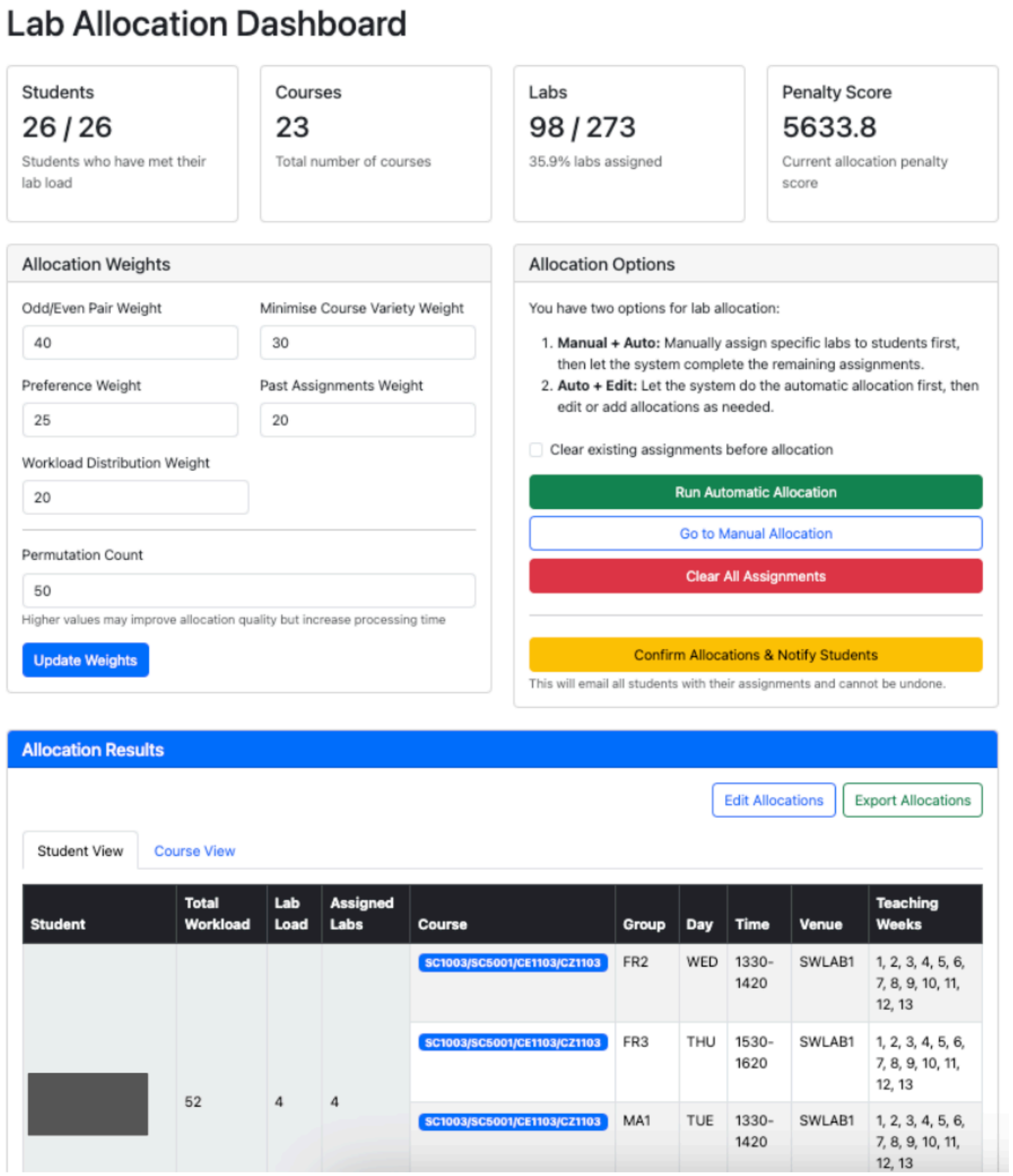
Use Case ID	UC-31
Use Case Name	Contact Another Student
Actor(s)	Student
Description	Allows Student to send a message to another student through the system.
Entry Condition(s)	Student is logged in and navigates to the contact page.
Exit Condition(s)	Message is sent to recipient’s email.

Flow of Events	1. System displays form with recipient dropdown and message field. 2. Student selects recipient and composes message. 3. Student submits form. 4. System sends email to recipient with sender's message and contact info (email address). 5. System shows success message. 6. System redirects to “View Allocations” page.
Alternative Flows	N/A
Exceptions	If email sending fails, system shows error message.
Includes	N/A
Extends	N/A

8.5 Implementation

8.5.1 View Allocation Dashboard

The allocation dashboard serves as the central hub for administrators to manage and view lab allocations.



Lab Allocation Dashboard

Students
26 / 26
Students who have met their lab load

Courses
23
Total number of courses

Labs
98 / 273
35.9% labs assigned

Penalty Score
5633.8
Current allocation penalty score

Allocation Weights

Odd/Even Pair Weight: 40	Minimise Course Variety Weight: 30
Preference Weight: 25	Past Assignments Weight: 20
Workload Distribution Weight: 20	

Permutation Count: 50

Higher values may improve allocation quality but increase processing time

Allocation Options

You have two options for lab allocation:

1. **Manual + Auto:** Manually assign specific labs to students first, then let the system complete the remaining assignments.
2. **Auto + Edit:** Let the system do the automatic allocation first, then edit or add allocations as needed.

☐ Clear existing assignments before allocation

This will email all students with their assignments and cannot be undone.

Allocation Results

Student	Total Workload	Lab Load	Assigned Labs	Course	Group	Day	Time	Venue	Teaching Weeks
[redacted]	52	4	4	SC1003/SC5001/CE1103/CZ1103	FR2	WED	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
				SC1003/SC5001/CE1103/CZ1103	FR3	THU	1530-1620	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
				SC1003/SC5001/CE1103/CZ1103	MA1	TUE	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

Figure 18: Lab Allocation Dashboard

8.5.1.1 Statistics Summary Cards

Displays real-time metrics about the allocation status:

- Students who have met lab load, out of the total number of students
- Total courses in the system
- Labs assigned, out of the total number of labs with the percentage
- Current penalty score (only displayed directly after the auto allocation has completed)

8.5.1.2 Allocation Parameters Configuration

Allows adjustment of optimization weights and number of permutations that influence the allocation algorithm.[4]

- Permutation Count
- Odd/Even Pair Weight
- Minimize Course Variety Weight
- Preference Weight
- Past Assignments Weight
- Workload Distribution Weight

8.5.1.3 Allocation Options

Provides different allocation strategies:

- Manual + Auto: Admin can manually edit assignments first, then auto-complete.
- Auto + Edit: Full auto-allocation first, then manual edits.
- Clear All Assignments: Resets all assignments

8.5.1.4 Allocation Results

Tabbed table interface, showing:

- Student view: all assignments grouped by student
- Course view: all assignments grouped by course

[4] All Allocation Parameters will be discussed in more detail in the next section.

8.5.2 Auto Allocation Algorithm

The core allocation algorithm follows this high-level flow:

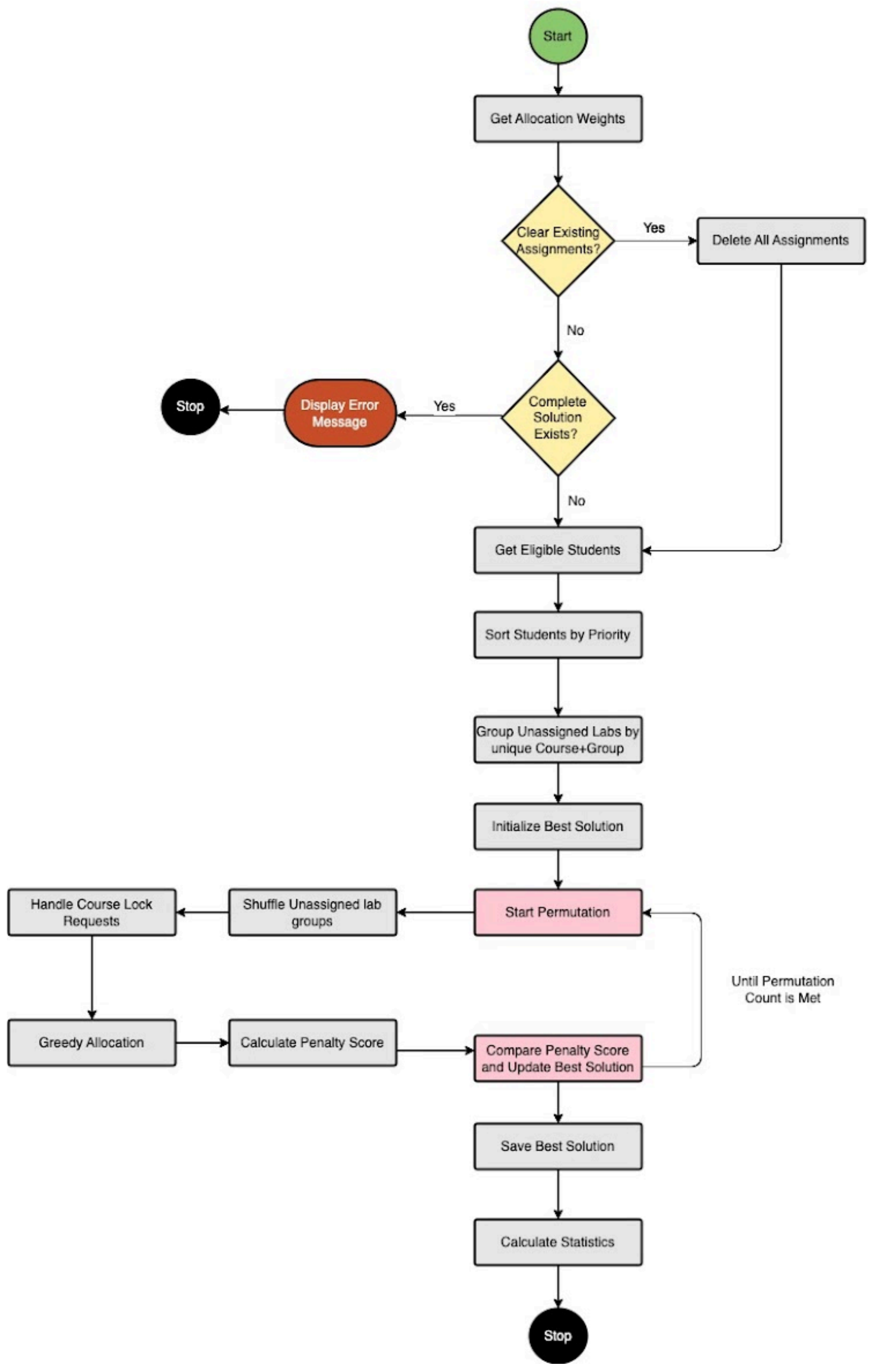

Start
Get Allocation Weights
Clear Existing Assignments?
Yes
Delete All Assignments
No
Complete Solution Exists?
Yes
Display Error Message
Stop
No
Get Eligible Students
Sort Students by Priority
Group Unassigned Labs by unique Course+Group
Initialize Best Solution
Start Permutation
Shuffle Unassigned lab groups
Handle Course Lock Requests
Greedy Allocation
Calculate Penalty Score
Compare Penalty Score and Update Best Solution
Until Permutation Count is Met
Save Best Solution
Calculate Statistics
Stop

Figure 19: High Level Flow Diagram for Allocation Algorithm for GSLAS

8.5.2.1 Student and Lab Groups Initialization

Sort all students that have indicated to be included in this allocation exercise by priority; those with availability constraints and higher lab loads come first.

For every permutation (until permutation count is met), shuffle the list of unique lab groups.

8.5.2.2 Course Lock Handling

Approved Course Lock Requests take priority in this algorithm. For students with approved course lock requests, until they meet their requested number of labs to lock:

1. Try to assign odd/even week pairs first
2. Try to assign the remaining lab(s) from the same course
3. Ensure assignments do not violate constraints

8.5.2.3 Greedy Allocation

For each student (in priority order), until they meet their lab load:

1. Try to assign labs from courses they have already been assigned
2. Try to assign odd/even week pairs[5] first
3. Try to assign based on teaching preferences (ranking 1 first)
   3.1 If Student has indicated the course in past_assignments, assign from that course first.
4. Finally, assign any remaining labs that fit constraints

For each assignment attempt:

1. Check for time clashes
2. Verify max teaching days not exceeded
3. Ensure unavailable slots are avoided
4. Confirm lab load not exceeded

[5] Odd/Even Week Pairs refer to pairs of lab groups that occur on the same day and time for the same course, with the only difference between them being the teaching week, whereby one happens on Odd Weeks only, and the other happens on Even Weeks only.

8.5.2.4 Penalty Score Calculation

The Allocation Weights input by the Admin are used in the calculation of penalty score for each permutation.

Table 32: Penalty Score Calculation

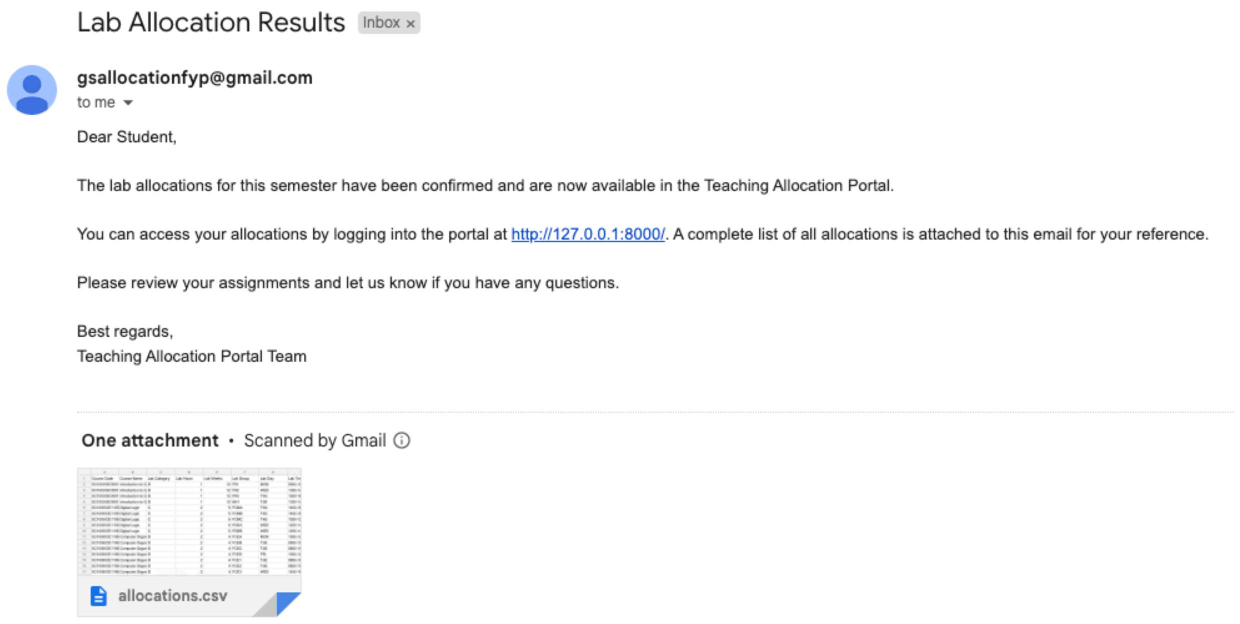
Factor	Default Weights	Calculation
Odd/Even Penalty	40	Count of odd/even week pairs assigned to different students
Course Variety Penalty	30	Sum for each student of (number of unique courses - 1)[6]
Preference Penalty	25	Sum of (preference rankings - 1) for assigned courses (double penalty for courses not in preferences)[7]
Workload Distribution Penalty	15	Standard deviation of (workload/lab_load) ratios across students
Past Assignments bonus	20	Count of past courses that were assigned (subtract from penalty)

Using default weights,
Final Penalty Score = (OddEvenPenalty x 40) +
(CourseVarietyPenalty x 30) +
(Preference Penalty x 25) +
(WorkloadPenalty x 15) -
(PastAssignmentsBonus x 20)

[6] No Penalty if student has only 1 course (best scenario)
[7] Preference rankings range from 1-8. Rank 1 will incur 0 penalty points, rank 8 will incur 7 penalty points. If course assigned is not in Student's preferences, incur 16 penalty points.

8.5.3 Confirm and Notify Students

This functionality allows administrators to send out the allocation results to all students with one click. The allocation results will be exported to CSV[8] and attached to a mass email that sends out to all students that have indicated to be included in this allocation exercise.



Lab Allocation Results

gsallocationfyp@gmail.com
to me

Dear Student,

The lab allocations for this semester have been confirmed and are now available in the Teaching Allocation Portal.

You can access your allocations by logging into the portal at http://127.0.0.1:8000/. A complete list of all allocations is attached to this email for your reference.

Please review your assignments and let us know if you have any questions.

Best regards,
Teaching Allocation Portal Team

One attachment • Scanned by Gmail

allocations.csv

Figure 20: Lab Allocation Results Email

[8] Refer to Appendix for example of output CSV

9 Conclusion

In this project, the Graduate Student Lab Allocation System (GSLAS) was successfully developed to address the inefficiencies of manual lab allocation in academic departments. The system provides a structured, automated approach to assigning graduate students to lab sessions while considering multiple constraints, including teaching preferences, workload balance, and scheduling conflicts.

The objectives outlined in Chapter 1.3 have been met:

- **Centralized data management** eliminates the need for fragmented email communication and manual spreadsheet consolidation.
- **Input validation** ensures standardized submissions, reducing errors in preference declarations.
- **Automated allocation algorithm** optimizes assignments based on configurable weights, improving fairness and efficiency.
- **Administrative flexibility** allows both manual adjustments and automated allocation modes, adapting to different departmental needs.

This report has detailed the system's design and implementation, covering database structure, business processes, and subsystem functionalities.

9.1 Challenges Faced

9.1.1 Full-Stack Development Learning Curve

As my first full-stack project, GSLAS required integrating frontend (HTML/CSS/Bootstrap), backend (Django/Python), and database (MySQL) components. While Django's "batteries-included" philosophy streamlined development, debugging interactions between layers—such as form validations and dynamic UI updates—proved time-consuming. Extensive reliance on documentation and iterative testing was necessary to resolve multiple issues.

9.1.2 Algorithm Optimization

The allocation algorithm's multi-criteria optimization demanded rigorous testing. Early versions produced suboptimal assignments and long processing times, requiring adjustments to the scoring system and permutation logic. While the final implementation achieves functional results, further refinements could enhance its efficiency for larger datasets.

9.1.3 Time Constraints

Despite initial plans for early development, delays in finalizing requirements and administrative approvals compressed the implementation timeline. This limited opportunities for extensive user testing and forced prioritization of core features over secondary enhancements.

9.2 Limitations

9.2.1 Scalability Testing

While GSLAS handles the current department’s student and lab volumes, its performance under significantly larger workloads remains untested.

In addition, GSLAS currently only handles lab allocation, and does not integrate a closely-related administrative task of assigning tutorials to graduate students as well. Future efforts can be made to incorporate this higher complexity problem as well.

9.2.2 Integration with Institutional Systems

GSLAS operates as a standalone system that requires many manual inputs. Future integration with university timetabling or student record systems could automate data synchronization and reduce manual entry.

9.2.3 Rigid Input Requirements

The system’s automated nature necessitates strict formatting rules for user inputs. While these constraints ensure error-free algorithm processing, they create a steep learning curve for first-time users. Administrative guides and form labels currently require extensive explanations (see Figure 10), which may overwhelm users accustomed to flexible manual systems. This trade-off between automation robustness and user-friendliness remains an ongoing challenge.

9.3 Future Enhancements

9.3.1 Advanced Conflict Resolution

Introducing a visual conflict detector (e.g. a calendar-style overlay) would help administrators spot and resolve scheduling clashes more intuitively.

9.3.2 Student Self-Swapping

Allowing students to request swaps for assigned labs (subject to admin approval) could reduce post-allocation adjustments while maintaining control.

9.3.3 Predictive Analytics

Machine learning could analyze historical allocation data to predict future demand for specific courses or time slots, further optimizing the algorithm.

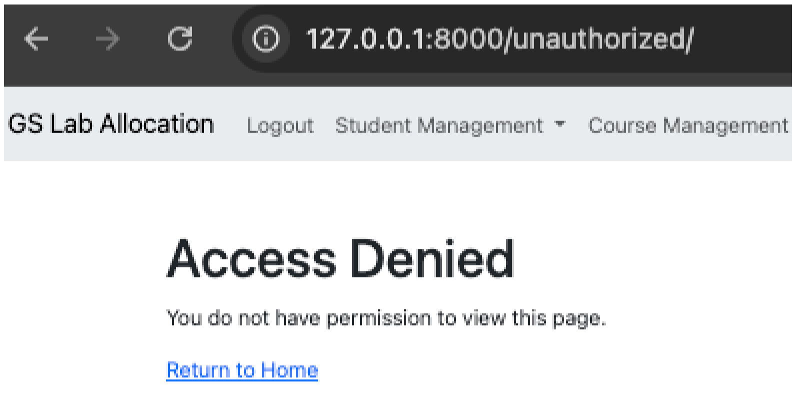
9.3.4 Expanded Role-Based Features

Adding instructor portals for lab supervisors to provide feedback on assigned students would close the loop between allocation and teaching quality assessment.

References

[1] Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86, 43-59.
[2] Perelstein, E. *et al.* (2016) ‘Automation improves schedule quality and increases scheduling efficiency for residents’, *Journal of Graduate Medical Education*, 8(1), pp. 45–49. doi:10.4300/jgme-d-15-00154.1.
[3] Diallo, F.P. and Tudose, C. (2024) ‘Optimizing the scheduling of teaching activities in a faculty’, *Applied Sciences*, 14(20), p. 9554. doi:10.3390/app14209554.
[4] Django Software Foundation. (2023). Django Documentation. Retrieved from https://docs.djangoproject.com
[5] GeeksforGeeks. (2023). Differences Between Django vs Flask. Retrieved from https://www.geeksforgeeks.org/differences-between-django-vs-flask/
[6] Bootstrap Team. (2023). Bootstrap Documentation. Retrieved from https://getbootstrap.com/docs/5.0/getting-started/introduction/
[7] MySQL. (2023). MySQL Workbench Documentation. Retrieved from https://dev.mysql.com/doc/workbench/en/

Appendix



127.0.0.1:8000/unauthorized/

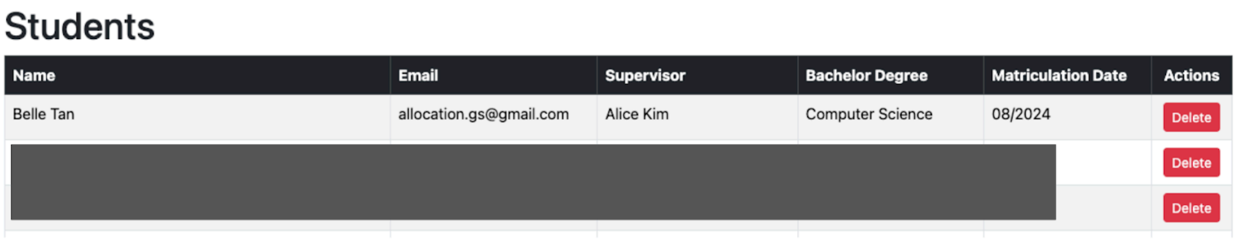
GS Lab Allocation Logout Student Management Course Management

Access Denied

You do not have permission to view this page.

Return to Home

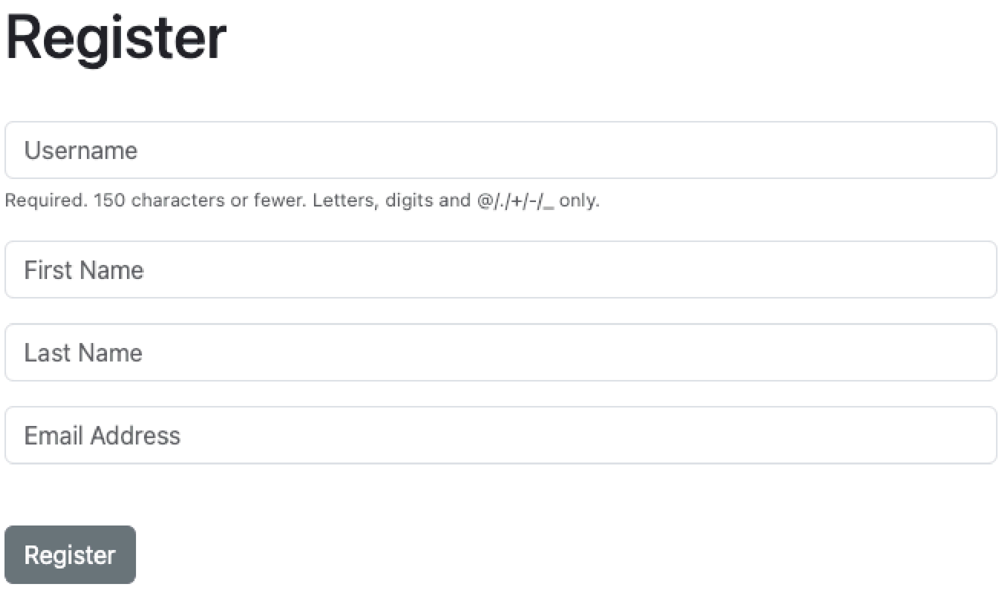
Figure 21: Unauthorized page



Students

Name	Email	Supervisor	Bachelor Degree	Matriculation Date	Actions
Belle Tan	allocation.gs@gmail.com	Alice Kim	Computer Science	08/2024	Delete
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Delete
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Delete

Figure 22: View Students page



Register

Username

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

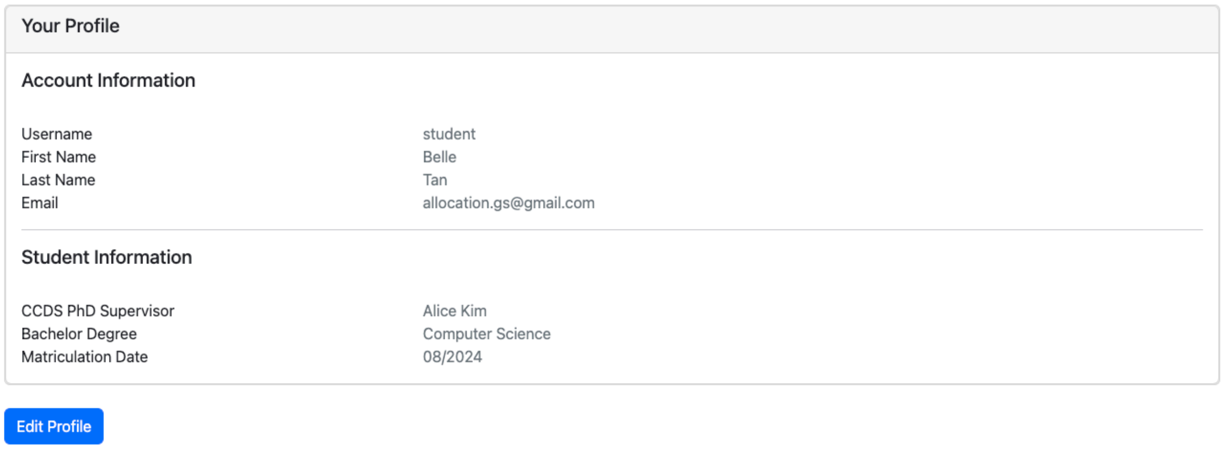
First Name

Last Name

Email Address

Register

Figure 23: Register User page


Your Profile

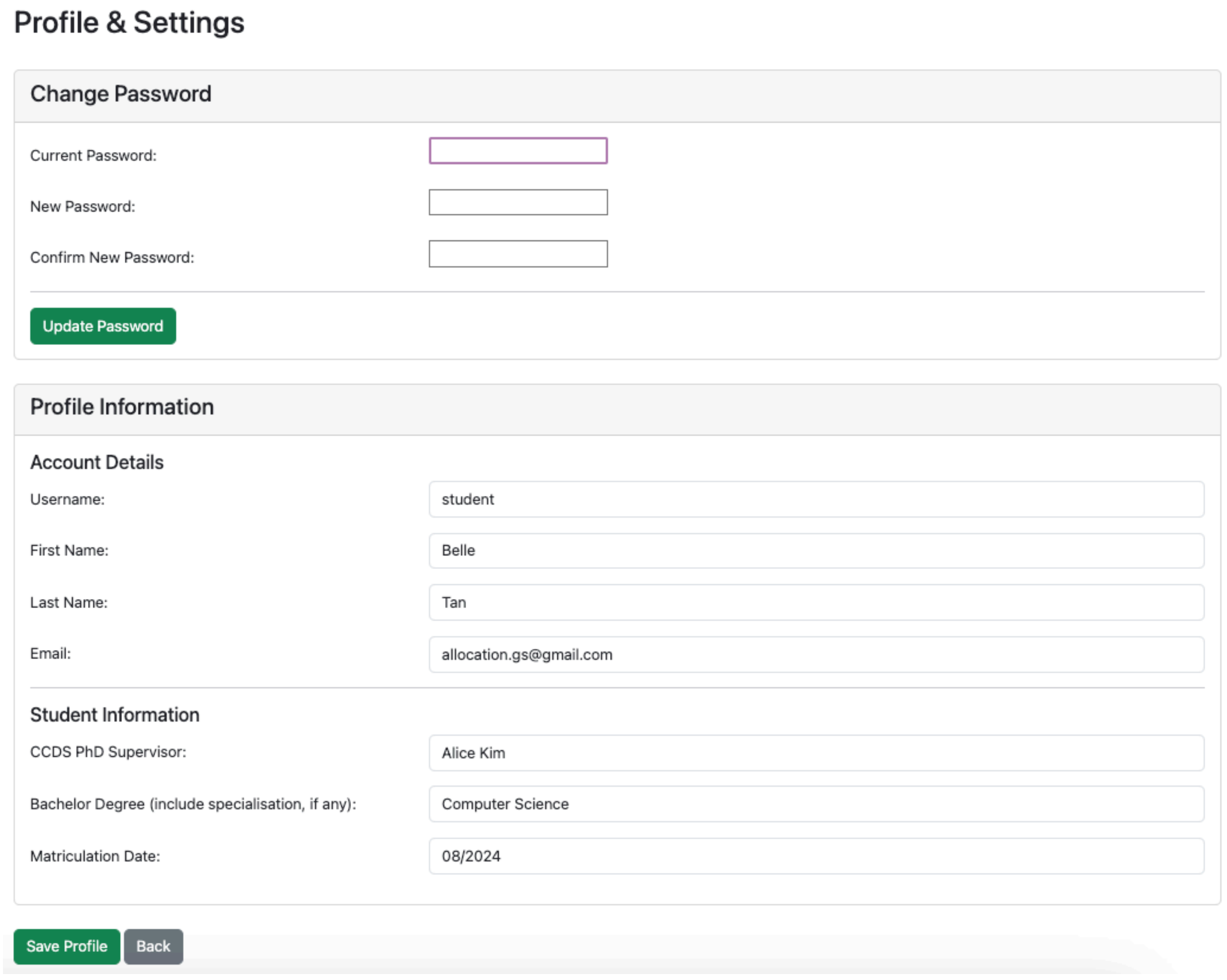
Account Information

Username	student
First Name	Belle
Last Name	Tan
Email	allocation.gs@gmail.com

Student Information

CCDS PhD Supervisor	Alice Kim
Bachelor Degree	Computer Science
Matriculation Date	08/2024

Figure 24: View Profile page


Profile & Settings

Change Password

Current Password:

New Password:

Confirm New Password:

Profile Information

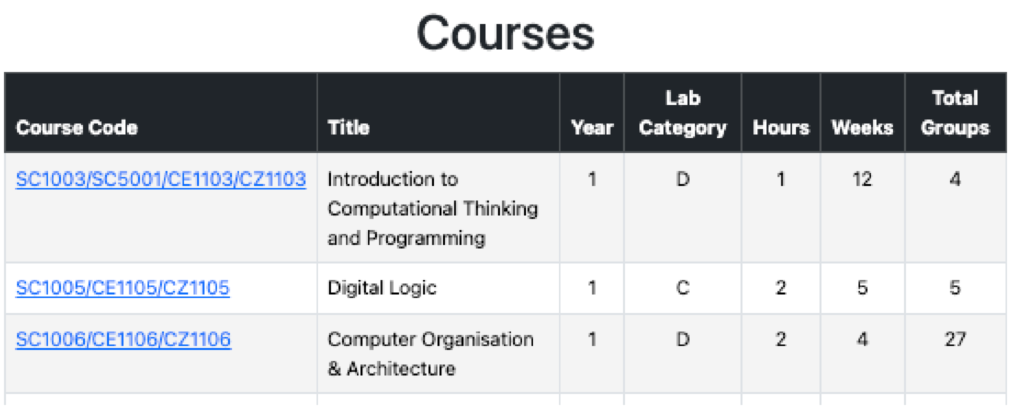
Account Details

Username:	student
First Name:	Belle
Last Name:	Tan
Email:	allocation.gs@gmail.com

Student Information

CCDS PhD Supervisor:	Alice Kim
Bachelor Degree (include specialisation, if any):	Computer Science
Matriculation Date:	08/2024

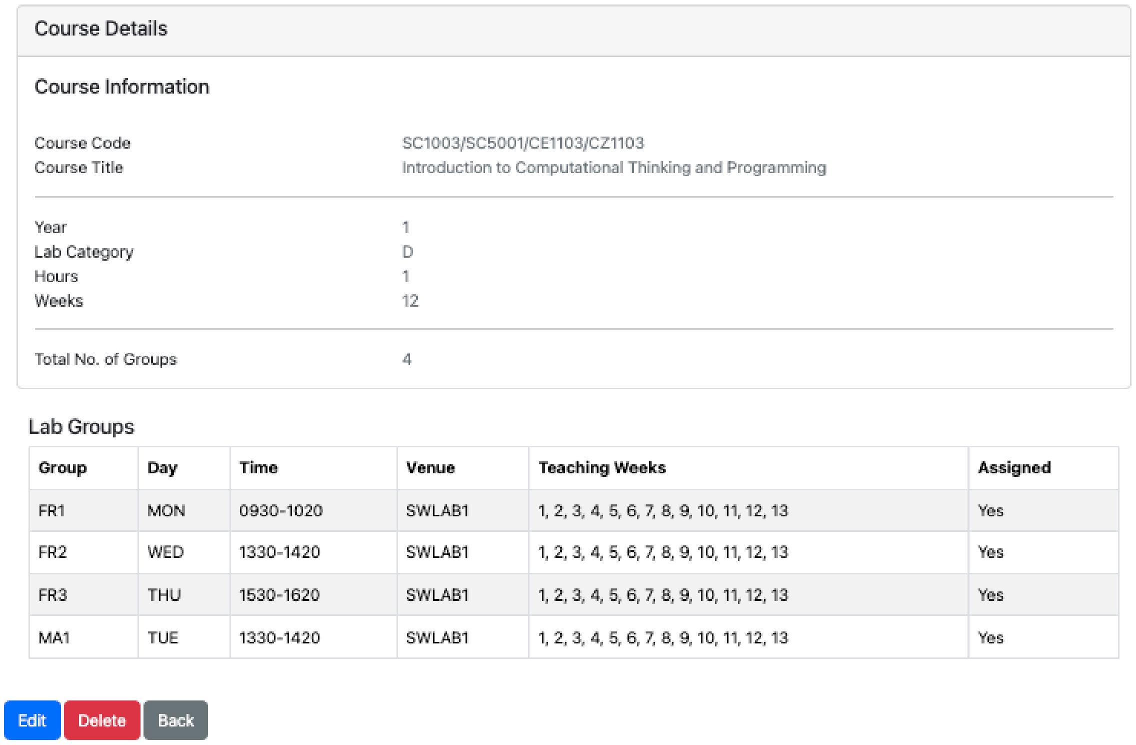
Figure 25: Edit Profile Page



Courses

Course Code	Title	Year	Lab Category	Hours	Weeks	Total Groups
SC1003/SC5001/CE1103/CZ1103	Introduction to Computational Thinking and Programming	1	D	1	12	4
SC1005/CE1105/CZ1105	Digital Logic	1	C	2	5	5
SC1006/CE1106/CZ1106	Computer Organisation & Architecture	1	D	2	4	27

Figure 26: View Courses page



Course Details

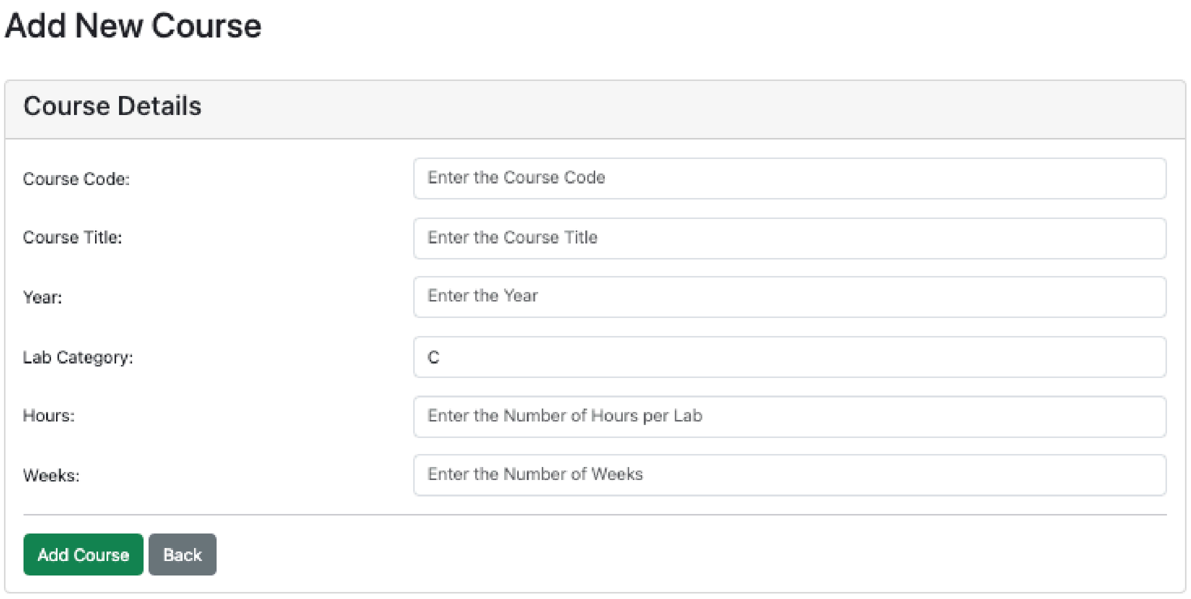
Course Information

Course Code	SC1003/SC5001/CE1103/CZ1103
Course Title	Introduction to Computational Thinking and Programming
Year	1
Lab Category	D
Hours	1
Weeks	12
Total No. of Groups	4

Lab Groups

Group	Day	Time	Venue	Teaching Weeks	Assigned
FR1	MON	0930-1020	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	Yes
FR2	WED	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	Yes
FR3	THU	1530-1620	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	Yes
MA1	TUE	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	Yes

Figure 27: View Course Details page

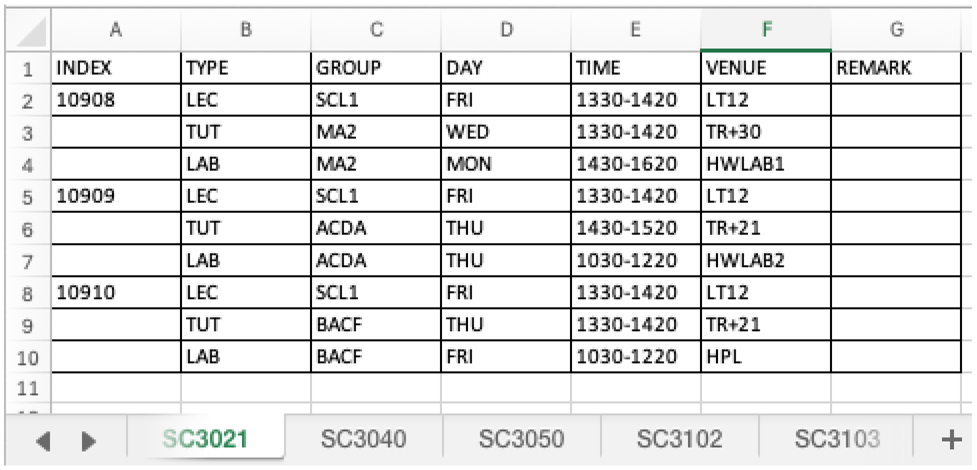


Add New Course

Course Details

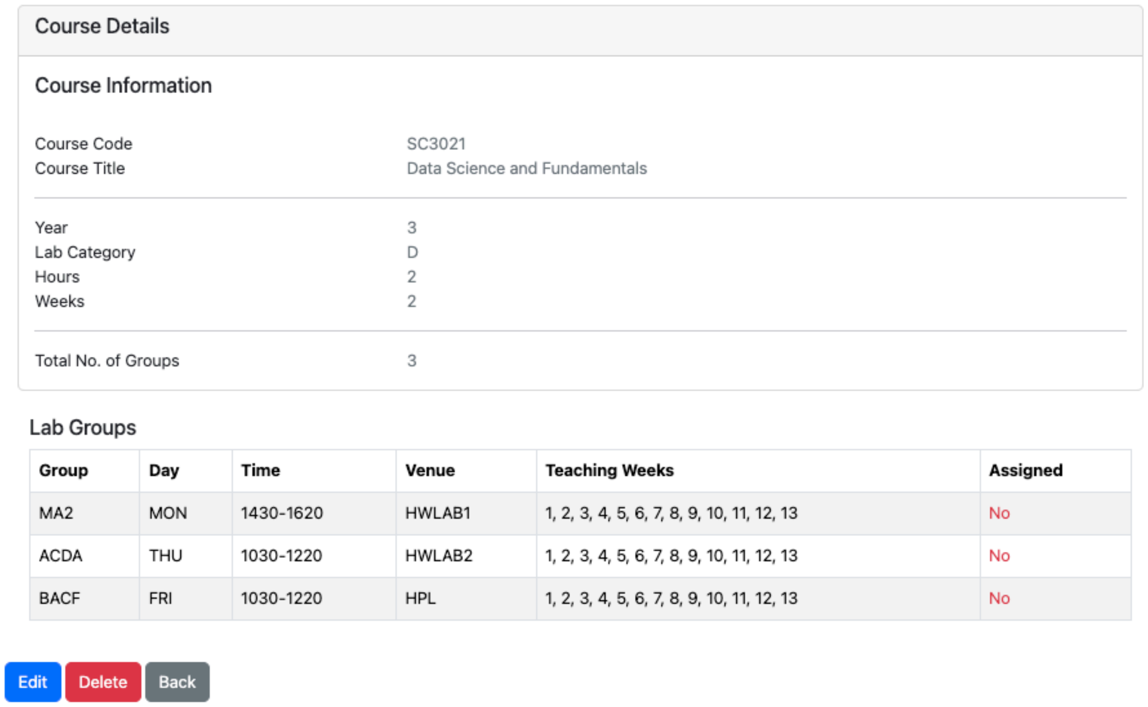
Course Code:	Enter the Course Code
Course Title:	Enter the Course Title
Year:	Enter the Year
Lab Category:	C
Hours:	Enter the Number of Hours per Lab
Weeks:	Enter the Number of Weeks

Figure 28: Add Course page



INDEX	TYPE	GROUP	DAY	TIME	VENUE	REMARK
10908	LEC	SCL1	FRI	1330-1420	LT12	
	TUT	MA2	WED	1330-1420	TR+30	
	LAB	MA2	MON	1430-1620	HWLAB1	
10909	LEC	SCL1	FRI	1330-1420	LT12	
	TUT	ACDA	THU	1430-1520	TR+21	
	LAB	ACDA	THU	1030-1220	HWLAB2	
10910	LEC	SCL1	FRI	1330-1420	LT12	
	TUT	BACF	THU	1330-1420	TR+21	
	LAB	BACF	FRI	1030-1220	HPL	

Figure 29: Input table for Add Labs



Course Details

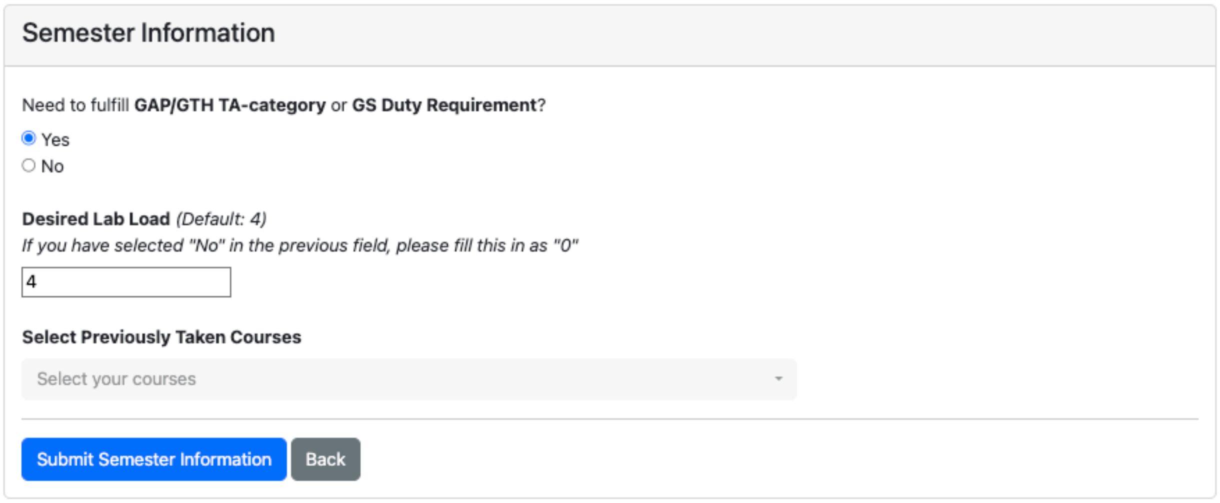
Course Information

Course Code	SC3021
Course Title	Data Science and Fundamentals
Year	3
Lab Category	D
Hours	2
Weeks	2
Total No. of Groups	3

Lab Groups

Group	Day	Time	Venue	Teaching Weeks	Assigned
MA2	MON	1430-1620	HWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	No
ACDA	THU	1030-1220	HWLAB2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	No
BACF	FRI	1030-1220	HPL	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	No

Figure 30: Output table for Add Labs



Semester Information

Need to fulfill **GAP/GTH TA-category** or **GS Duty Requirement**?

[✓ Yes] No

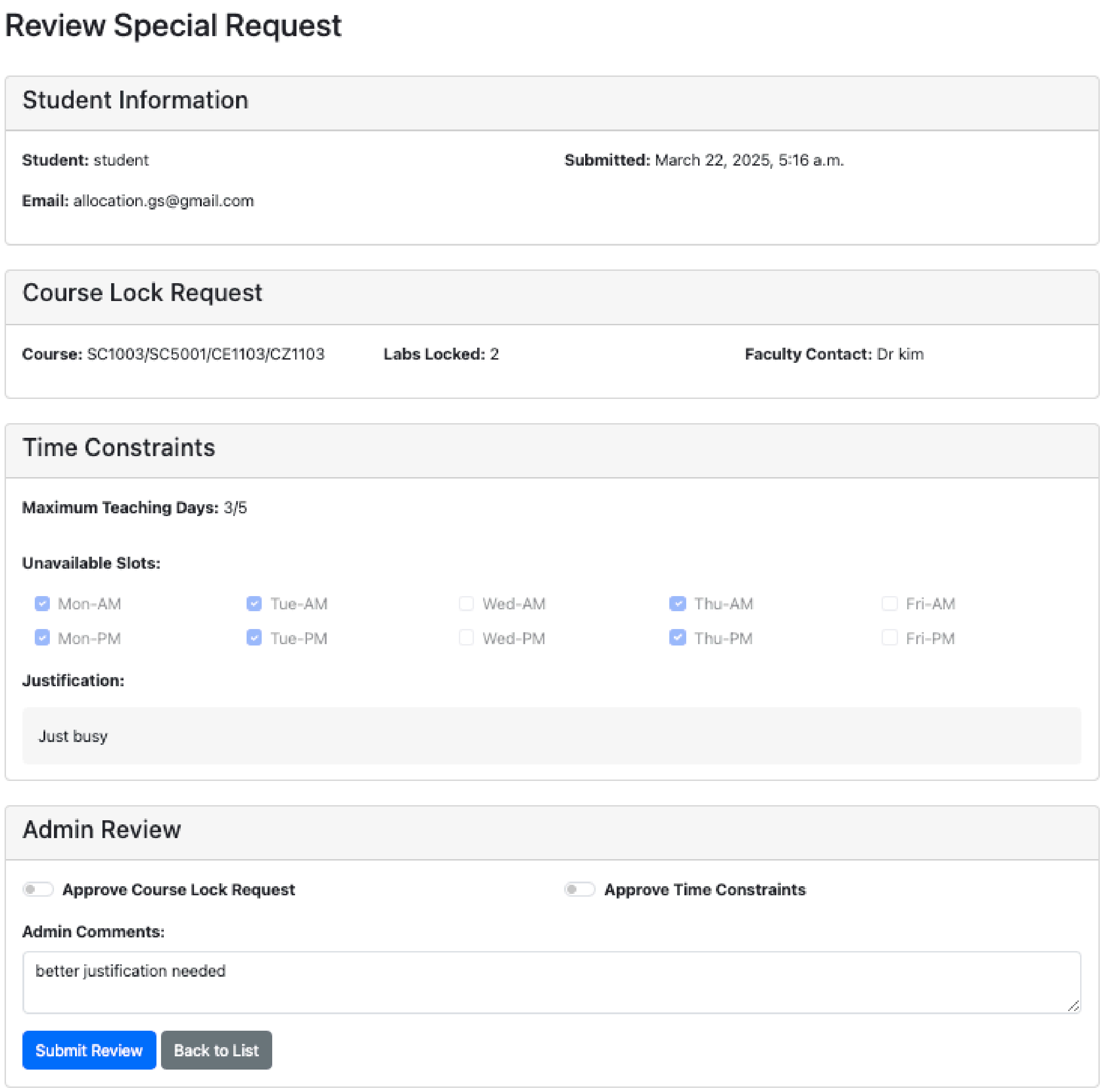
Desired Lab Load *(Default: 4)*
If you have selected "No" in the previous field, please fill this in as "0"

4

Select Previously Taken Courses

Select your courses

Figure 31: Submit Semester Information Page



Review Special Request

Student Information

Student: student

Submitted: March 22, 2025, 5:16 a.m.

Email: allocation.gs@gmail.com

Course Lock Request

Course: SC1003/SC5001/CE1103/CZ1103

Labs Locked: 2

Faculty Contact: Dr kim

Time Constraints

Maximum Teaching Days: 3/5

Unavailable Slots:

✓ Mon-AM	✓ Tue-AM	☐ Wed-AM	✓ Thu-AM	☐ Fri-AM
✓ Mon-PM	✓ Tue-PM	☐ Wed-PM	✓ Thu-PM	☐ Fri-PM

Justification:

Just busy

Admin Review

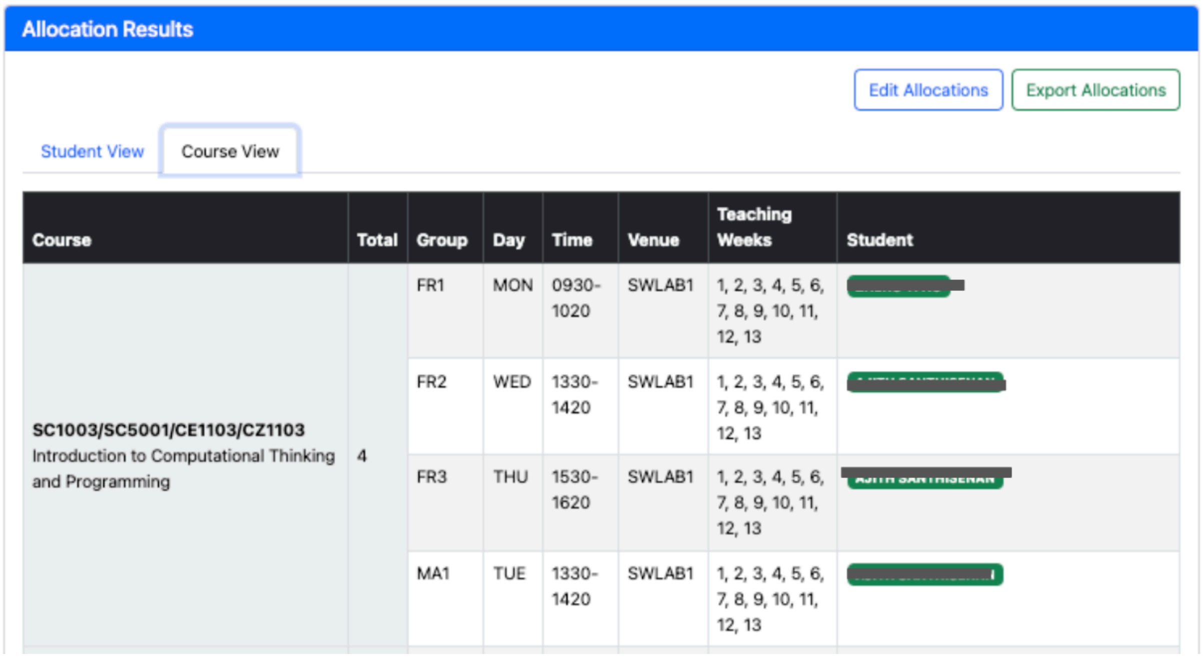
☐ **Approve Course Lock Request**

☐ **Approve Time Constraints**

Admin Comments:

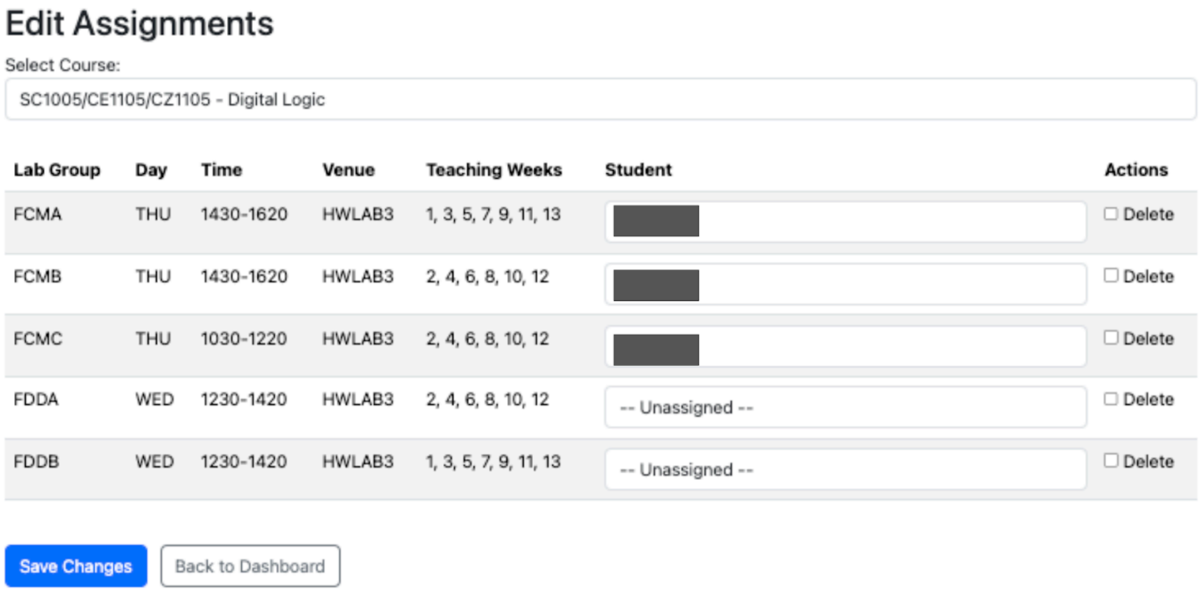
better justification needed

Figure 32: Review Special Request Page


Allocation Results

Course	Total	Group	Day	Time	Venue	Teaching Weeks	Student
SC1003/SC5001/CE1103/CZ1103 Introduction to Computational Thinking and Programming	4	FR1	MON	0930-1020	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	[redacted]
		FR2	WED	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	[redacted]
		FR3	THU	1530-1620	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	[redacted]
		MA1	TUE	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	[redacted]

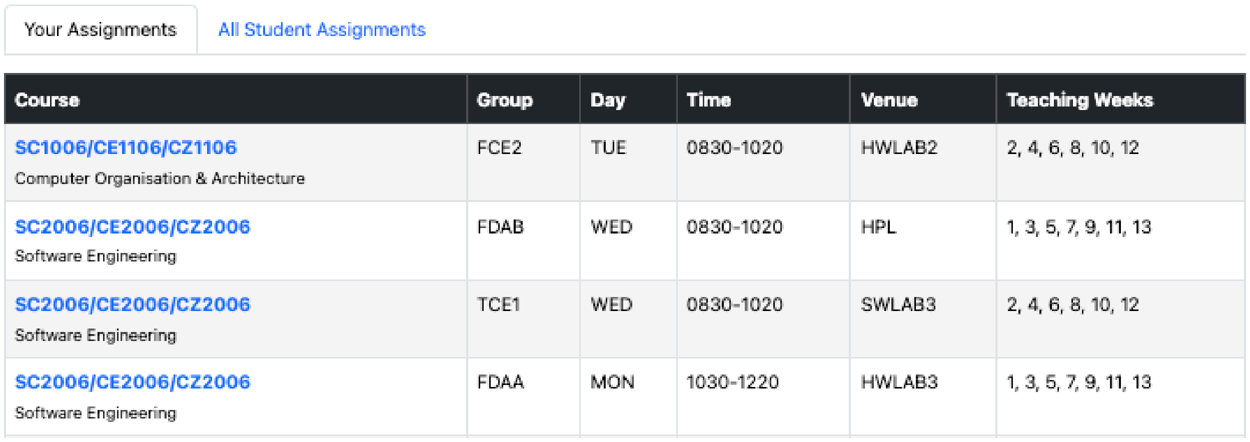
Figure 33: Course View for Allocation Dashboard


Edit Assignments

Select Course: SC1005/CE1105/CZ1105 - Digital Logic

Lab Group	Day	Time	Venue	Teaching Weeks	Student	Actions
FCMA	THU	1430-1620	HWLAB3	1, 3, 5, 7, 9, 11, 13	[redacted]	☐ Delete
FCMB	THU	1430-1620	HWLAB3	2, 4, 6, 8, 10, 12	[redacted]	☐ Delete
FCMC	THU	1030-1220	HWLAB3	2, 4, 6, 8, 10, 12	[redacted]	☐ Delete
FDDA	WED	1230-1420	HWLAB3	2, 4, 6, 8, 10, 12	-- Unassigned --	☐ Delete
FDDB	WED	1230-1420	HWLAB3	1, 3, 5, 7, 9, 11, 13	-- Unassigned --	☐ Delete

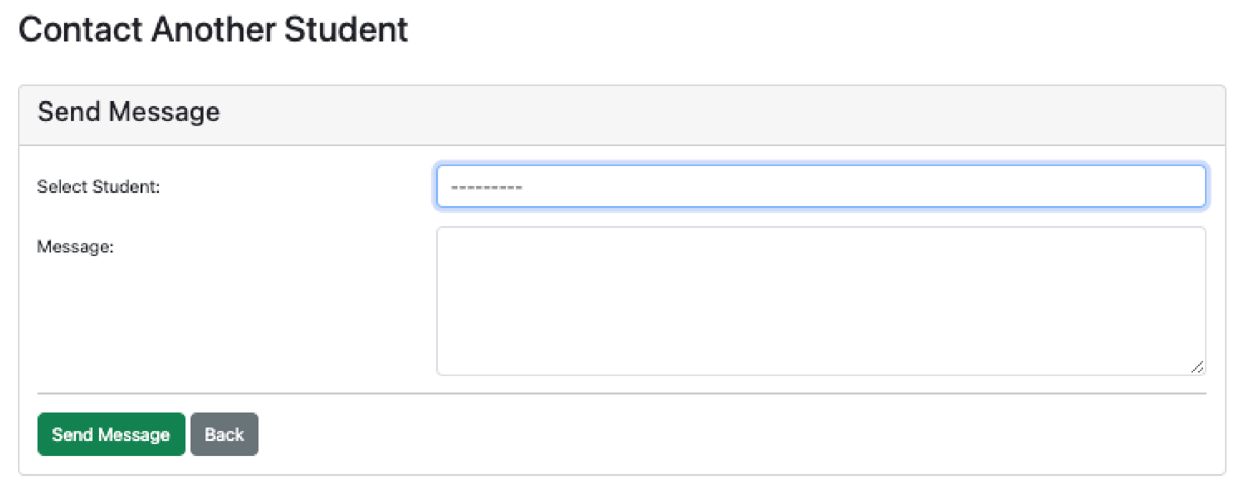
Figure 34: Edit Allocation Page



Your Assignments | All Student Assignments

Course	Group	Day	Time	Venue	Teaching Weeks
SC1006/CE1106/CZ1106 Computer Organisation & Architecture	FCE2	TUE	0830-1020	HWLAB2	2, 4, 6, 8, 10, 12
SC2006/CE2006/CZ2006 Software Engineering	FDAB	WED	0830-1020	HPL	1, 3, 5, 7, 9, 11, 13
SC2006/CE2006/CZ2006 Software Engineering	TCE1	WED	0830-1020	SWLAB3	2, 4, 6, 8, 10, 12
SC2006/CE2006/CZ2006 Software Engineering	FDAA	MON	1030-1220	HWLAB3	1, 3, 5, 7, 9, 11, 13

Figure 35: View Allocations Page



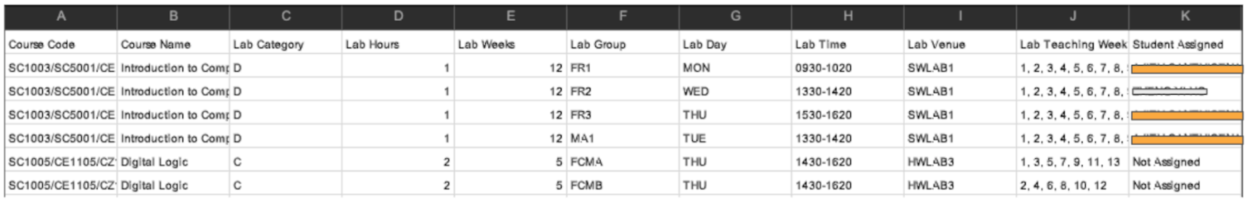
Contact Another Student

Send Message

Select Student: ---------

Message:

Figure 36: Contact Another Student Page



A	B	C	D	E	F	G	H	I	J	K
Course Code	Course Name	Lab Category	Lab Hours	Lab Weeks	Lab Group	Lab Day	Lab Time	Lab Venue	Lab Teaching Week	Student Assigned
SC1003/SC5001/CE	Introduction to Comp	D	1	12	FR1	MON	0930-1020	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8,	[redacted]
SC1003/SC5001/CE	Introduction to Comp	D	1	12	FR2	WED	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8,	[redacted]
SC1003/SC5001/CE	Introduction to Comp	D	1	12	FR3	THU	1530-1620	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8,	[redacted]
SC1003/SC5001/CE	Introduction to Comp	D	1	12	MA1	TUE	1330-1420	SWLAB1	1, 2, 3, 4, 5, 6, 7, 8,	[redacted]
SC1005/CE1105/CZ	Digital Logic	C	2	5	FCMA	THU	1430-1620	HWLAB3	1, 3, 5, 7, 9, 11, 13	Not Assigned
SC1005/CE1105/CZ	Digital Logic	C	2	5	FCMB	THU	1430-1620	HWLAB3	2, 4, 6, 8, 10, 12	Not Assigned

Figure 37: CSV file output