Department of Computer Science & Informatics (CSIS6809)

Personal Scheduling Assistant for Classes and Practicals

Technical Manual

Name:	Oratile Molongwana
Student no.:	2021578097
Project no.:	2025/16
Supervisor:	Prof. T. Stott
Co-Supervisor:	Mr J. Marais

NATURAL AND AGRICULTURAL SCIENCES UFS

Inspiring excellence, transforming lives through quality, impact, and care.

VISION130 Renew and Reimagine for 2034

UNIVERSITY OF THE FREE STATE UNIVERSITEIT VAN DIE VRYSTAAT YUNIVESITHI YA FREISTATA

UFS NATURAL AND AGRICULTURAL SCIENCES

Personal Scheduling Assistant for Classes and Practicals

CSIS6809 HONOURS PROJECT

Technical Manual

Submitted as partial requirement for the degree of CSIS Honours

Department of Computer Science and Informatics

Faculty of Natural and Agricultural Sciences

University of the Free State

Oratile Molongwana

2021578097

Supervisor: Prof T. Stott

Co-supervisor: Mr J. Marais

27 October 2025

Table of Contents

List of Figures

List of Tables

List of Abbreviations

UFS	University of the Free State
LMS	Learning Management System
ERP	Enterprise Resource Planning
AI	Artificial Intelligence
HR	Human Resources
DBMS	Database Management System
DB	Database
UI	User Interface

Glossary

Term	Definition
Admin	A system role responsible for managing users, sessions, and maintaining overall system integrity.
Backend	The server-side component of the system responsible for logic, processing, and database management.
Conflict Detection	A feature that identifies overlapping schedules involving students, lecturers, or demmies.
Database Schema	The structural design defining all tables, attributes, and relationships within the database.
Demmie	An academic assistant who supports lecturers during practical sessions or tutorials.
Evaluation	The process of assessing system usability, effectiveness, and performance through user testing.
Frontend	The user interface of the web system that allows lecturers, demmies, and admins to interact with system functions.
Hardware Requirements	The physical computing and networking resources necessary to host and operate the system efficiently.
Module	An academic subject or course component that includes lectures, practicals and assessments.
Module Session	A scheduled instance of teaching activity related to a specific module.
Microsoft SQL Server	The relational database management system used for secure and efficient data storage and querying.
Net Promoter Score (NPS)	A user satisfaction measure indicating how likely users are to recommend the system to others.

Term	Definition
Notification System	A module that delivers updates such as timetable changes, allocations or system alerts to users.
Prototype	A preliminary working model developed to evaluate system functionality and gather user feedback.
Scheduling Efficiency	The degree to which the system automates and optimizes session and resource allocation.
Session	A scheduled class, tutorial, or practical where students and demmies participate under lecturer supervision.
Software Requirements	The operating systems, libraries and applications required to run the system.
Usability	The ease with which end-users can navigate, understand and operate the system effectively.
Venue	A physical room, hall, or laboratory where a scheduled session is conducted.
Web Application	A browser-based software system enabling multi-role access and real-time scheduling interactions.

Chapter 1: Introduction

1.1 Introduction

As the number of students enrolling for tertiary education keeps growing exponentially (Cowling, statista, 2024), the demand for automated systems that increase productivity for both lecturers and students becomes increasingly evident. Across multiple institutions, students often receive personalized schedules that list all potential lecture and practical session times, regardless of their actual allocation. This then leads to unnecessary confusion about attendance requirements. Students end up attending way more classes than they actually need. This technical manual highlights how the mentioned problem is tackled.

1.2 Background to the study

Most existing systems either generate generic schedules for entire classes or only allow student-preference based session allocations. They often fail to take into account that students do not need to attend all the presented sessions, rather only the sessions they are allocated to. These existing systems rely heavily on manual interventions, which is not scalable for large student groups and can lead to inefficiencies in managing lecture and practical sessions. Needless to say, this presents a plethora of issues and often leads to uneven distribution venues, underutilized university resources and inconsistent student experiences.

To eliminate the above highlighted predicament, a solution that provides real-time conflict detection, student-specific timetables and flexible lecturer controls is urgently needed. The proposed project will allocate students to specific sessions based on a predetermined criteria defined by the lecturer. The system will offer various allocation strategies such as round-robin assignment or venue-priority filling, ensuring balanced student distribution. The students will then be sent a personalized timetable with only sessions that are relevant to the particular student improving the overall effectiveness of the personalized schedule and overall efficiency of the allocation process.

1.3 Problem statement and aims

As aforementioned, a solution that will have real-time conflict detection capabilities, a personalised-timetable generator and built-in manual controls is of urgent need. Solving this problem is crucial because it directly affects the academic experience for both students and staff. A lecturer-driven system that allows dynamic, rule-based student allocation would significantly improve the scheduling efficiency within a huge institution like the UFS, it would ensure balanced student occupation across all the available venues, reduce the administrative burden and enhance the clarity of student timetables. This in turn supports better learning outcomes, more effective use of institutional resources and streamlines the academic management process.

1.4 Scope of the Project

This project aims to design and prototype a smart, lecturer-driven timetable management system capable of:

- Automatically assigning students to repeated lecture and practical sessions using custom lecturer-defined rules.
- Generating personalized timetables for students that only include sessions they are required to attend.
- Dynamically detecting and resolving conflicts in schedules, including session overlaps, room limitations and late enrolments.
- Supporting manual overrides by lecturers or admins to accommodate special cases.
- Allowing lecturers to assign demmies to various venues based on availability and venue capacity.

The scope of this project will focus on creating the student allocation prototype with the above-mentioned functionalities. However, it is important to note that full LMS (blackboard) integration, attendance tracking and reporting, administrative interfaces or financial modules are beyond the scope of this project.

1.5 Limitations of the project

The system will not be fully integrated with the existing LMS platform which is blackboard in this case. However, it will simulate key functionalities of the platform to demonstrate the real-world application of the system. Moreover, the system will have limited access to real institutional data and will therefore make use of mock datasets for testing and evaluation.

1.5.1 Performance limitations

The system must be capable of handling concurrent usage as it will be expected to support at least 250 simultaneous users. This implies that the underlying architecture must be optimized for high performance and scalability.

It is also worth noting that ideally the system will be designed to be accessible from any device with an internet connection. It will be web-based, requiring compatibility across multiple devices and operating systems, as a result it must be designed to function efficiently on various hardware configurations including desktops, laptops, tablets, etc. Ensuring responsiveness and adaptability across different screen sizes and input methods will be a critical consideration during the development process.

1.6 Project Schedule

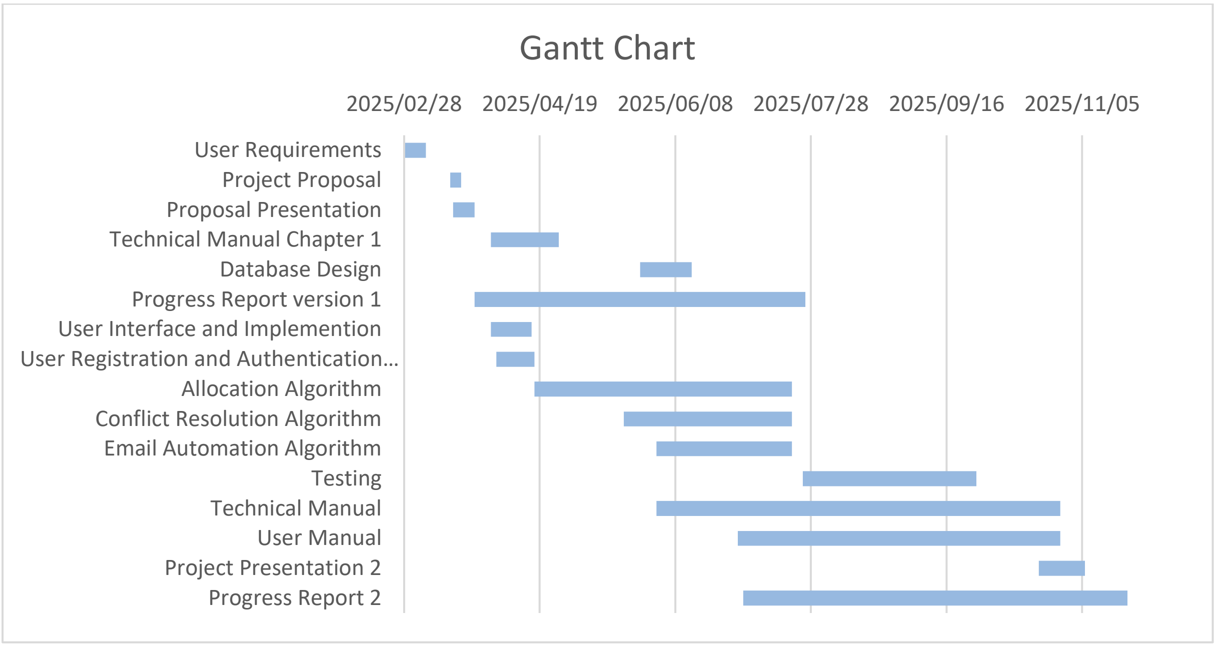


Figure 1: Gantt Chart

1.7 Document Overview

This technical manual follows the following structure:

- Chapter 1 – Introduction
- Chapter 2 – Literature Review
- Chapter 3 – Requirements and Analysis Workflow
- Chapter 4 – Architectural Design
- Chapter 5 – Source Code and Implementation
- Chapter 6 – Evaluation
- References
- Appendices

Chapter 2: Literature Review

2.1 Introduction

With the rising number of students enrolling in higher education globally (Cowling, statista, 2024), the efficient management of academic timetables has become increasingly challenging. Universities often struggle to allocate students to appropriate lecture and practical sessions without causing confusion among students or overwhelming staff through the manual creation of schedules. Traditional systems generally generate static timetables or give students preference in selecting their schedules, which can lead to overcrowded venues, uneven student distribution and scheduling conflicts.

This literature review evaluates current timetable management systems, identifies their limitations and highlights the need for an intelligent, lecturer-driven automated scheduling solution.

2.2 Existing Timetable Management Systems

2.2.1 aSc Timetables

aSc Timetables is a timetable generation software designed to automatically create school schedules. It uses algorithms to handle constraints like teacher and room availability and can quickly regenerate schedules when changes occur.

Although aSc Timetables is perfect for generating schedules and allowing manual adjustments, it does lack features for automatically allocating students to repeated sessions or rule-based allocation like round-robin and venue capacity. This is critical in university settings where modules may have hundreds and hundreds of students.

Moreover, students often receive generalized timetables for the entire class instead of personalised schedules which highlights the very issue at hand where students attend irrelevant sessions they are not required to attend. However, its ability to automatically generate schedules based on constraints such as room and teacher availability is a feature to emulate and scale up so that it is suitable for a university setting.

2.2.2 EduPage

EduPage is a cloud-based school management platform that integrates communication, attendance, grade tracking, homework and timetable sharing. It is much more popular across pre-tertiary environments and works closely with aSc Timetables to allow schedule management functionalities. Students, parents and teachers receive real-time updates via mobile and web apps. It is popular for its user-friendly interface and features like calendar events and class registers.

EduPage is primarily designed for primary and secondary schools. Its features may not be fully adaptable for higher education institutions where module schedules are more complex, students have flexible course combinations and scheduling involves lecture repetitions and dynamic class changes.

Since EduPage works closely with aSc Timetables, it therefore shares much of its shortcomings, notably the lack of automated conflict resolution. While aSc Timetables is highly regarded for its powerful timetable generation capabilities, it still relies heavily on user input and manual adjustments when dealing with complex scenarios such as overlapping sessions, venue constraints or availability of staff. As EduPage is designed to complement aSc, any issues that are unresolved in the core scheduling phase are inevitably carried over into the EduPage environment. A feature that could be improved on, is the conflict resolution mechanism that depends on manual intervention. Instead, the proposed system will automate the process thus improving overall efficiency and productivity.

2.2.3 WebUntis

WebUntis is an online scheduling and class register system. It allows teachers to manage lesson plans, take attendance and book venues. Students can register for courses digitally and all stakeholders can access the current timetable and changes via web or mobile apps.

WebUntis supports online scheduling but it is not optimised to handle highly dynamic or complex university-level timetables with multiple session repetitions, venue constraints or elective groupings. And like the aforementioned systems, WebUntis lacks a feature for automatically assigning students to specific sessions using predefined lecturer rules like round-robin, capacity-based or student-preference based assignment. Although all students can view the schedules, there is no indication of full personalization where students are shown only the sessions they are assigned to rather than all potential options. Since lecturer repetitions are not taken into consideration and they all receive a generic schedule, this is an opportunity to improve on these features.

2.2.4 Vidyalaya School ERP

Vidyalaya is a school ERP solution that includes timetable management, fee tracking, student records, HR and much more. Its timetable functionality supports both manual and automated scheduling across multi-campus institutions. It is known for customization, report generation and workload balancing features. And like Edupage it mainly serves pre-tertiary schools but is also used by some colleges.

The system has functionalities that alert teachers to scheduling conflicts but does not offer dynamic, automatic conflict resolution functionality that reassigns students different sessions or finds alternative slots based on constraints which are features that will be essential to the success of this project.

2.2.5 TimetableMaster

TimetableMaster is a cloud-based scheduling tool powered by AI. It offers automatic conflict detection, drag-and-drop editing and real-time sharing amongst students and teachers. It is designed to reduce admin time and increase scheduling efficiency, with a modern interface and focus on ease of collaboration and accessibility from any device.

TimetableMaster seems to be more admin driven through features like AI-powered scheduling and smart conflict resolution, it in turn overlooks the personalization of student timetables unlike the other systems that allow students some degree of interaction with their schedules, such as session preferences or conflict flagging. This could prove to be problematic for large university modules with repeated sessions.

It would also be highly beneficial for TimetableMaster to incorporate a rules-based notification system integrated with emails. This feature could automatically alert students and lecturers of key updates such as class reassignments and venue changes. AI-powered scheduling and conflict detection from TimetableMaster are both features that need to be explored and emulated in order to enable the system to proactively adapt to scheduling anomalies or institutional adjustments.

2.2.6 Blackboard

Blackboard is a widely-used Learning Management System (LMS) in higher education. It is the current system in use at the University of the Free State. It allows course delivery, student enrolment, assignment submission, grade management, discussion boards and it includes scheduling tools as well.

The current system allows students to select their preferred practical session slots, which can lead to overcrowding in some sessions and underutilization in others as this neglects the even distribution of students across all the venues. Moreover, there is no built-in logic to automatically assign students in a way that balances group sizes, considers venue availability or avoids schedule conflicts. Whilst the current UFS system seems to work, the load-balancing logic to ensure equitable venue usage and avoid overcrowding as an issue that could definitely be worked on.

2.2.7 Gradebook

Another noteworthy platform widely adopted across numerous tertiary institutions including UFS is Gradebook. Gradebook allows lecturers to manage, organise and modify student-related information such as grades and assessment records.

Beyond that, gradebook has a feature that creates personalized schedules for students. This is achieved by consolidating all registered modules and shows theoretical lecture sessions and practical sessions in one single document. Though it does create personalised schedules according to the modules that a student registered, it does not eliminate the session replication predicament, where students may be allocated to multiple instances of the same session. Additionally, it’s rigid structure prevents lecturers from manually moving the students between sessions as they wish. These limitations presents some features to emulate and potentially improve.

2.3 Comparison of Existing Systems

Functionality	**aSc Timetables**	**EduPage**	**WebUntis**	**Vidyalaya**	**TimetableMaster**	**Blackboard**	**Gradebook**
Automatic timetable generation	Yes	Yes	Yes	Yes	Yes	Basic	Yes
Personalized student timetables	No	No	No	No	No	Partial	Yes
Automated student allocation	No	No	No	No	No	No	No
Conflict detection	Partial	No	No	Partial	Yes	No	No
Automatic conflict resolution	No	No	No	No	Partial	No	No
Lecturer-defined allocation rules	No	No	No	No	No	No	No

Table 1: Existing Systems

2.4 Chapter Overview

An analysis of the aforementioned systems reveals several common limitations. None of the systems support allocation strategies such as round-robin, venue-priority or academic grouping. Some of the systems struggle to handle overlapping schedules, student reallocation requests or late enrolments efficiently. When conflicts arise, most systems require manual intervention rather than automatically resolving and reallocating the affected students.

Although current systems such as EduPage, Blackboard and TimetableMaster offer powerful features for timetable generation, they are limited in their ability to dynamically allocate students to sessions using customizable lecturer-defined criteria.

The inefficiencies found in the existing systems highlight a growing need for the platform I am developing. The proposed system should allow lecturers to define custom rules for students and demmies to allocate them to various sessions through round-robin, academic performance, etc. The system should generate student-specific timetables with only relevant sessions displayed to the student, reducing confusion and improving clarity.

Dynamic conflict detection and reallocation should also be taken into consideration, ensuring timetables adapt in real-time to overlapping sessions, changes to the schedules or any issue encountered. Providing a balanced session distribution to prevent overcrowding and underuse of educational resources whilst ensuring the system facilitates lecturer overrides and manual adjustments is also a critical component of the proposed system.

Future developments should focus on integrating Artificial Intelligence and Machine Learning to include analytics dashboards to help lecturers monitor allocations, attendance trends and resource utilisation.

The proposed solution should bridge this gap by implementing intelligent allocation strategies, personalized student timetables and robust conflict resolution mechanisms. This advancement would significantly enhance scheduling efficiency, reduce administrative burden and improve the overall student experience in higher education institutions.

3. Chapter 3: Requirements and Analysis Workflow

3.1 Introduction

Following the literature review presented in Chapter 2, which examined existing timetable management systems and their limitations, this chapter focuses on defining the requirements and analytical framework that guided the design of the Personal Scheduling Assistant for Classes and Practicals. The insights gained from the previous chapter highlighted the need for intelligent allocation, personalized scheduling and automated conflict resolution, features that directly inform the system’s core requirements.

The following chapter will present a detailed analysis of the functional system requirements that inform the design and development of the proposed Personal Scheduling Assistant for Classes and Practicals. It outlines the core capabilities essential to the successful execution of this project. A comprehensive use case diagram accompanied by individual use case descriptions and scenarios for each use case will also be included to demonstrate practical, real-world interactions between users and the system.

3.2 Functional System Requirements

3.2.1 User Registration

This process allows Lecturers and Demmies to register in the system before they can access any role-specific functionalities. This is a one-time setup that stores the essential user details and defines their role within the system as a Lecturer or a Demmie.

3.2.2 Authentication

This functionality is designed to ensure that only authorised lecturers can access the system. The system will verify the credentials entered by the user against the established database of authorised users to ensure the secure handling of sensitive student information.

3.2.3 Personalised Timetable Generation

Each student will be presented with a timetable tailored to include only the specific sessions to which they are assigned. This functionality directly addresses the inefficiencies associated with generalized timetables and eliminates student confusion regarding their attendance obligations.

3.2.4 Automated Session Allocation

Students will be automatically allocated to specific sessions specified by the lecturer. This feature significantly reduces the manual workload for academic staff and improves overall productivity and efficiency.

3.2.5 Dynamic Conflict Detection and Resolution

The system will monitor all the students' schedules and it should be able to not only identify conflicts such as overlapping sessions but also be capable of resolving the conflict automatically without admin intervention.

3.2.6 Lecture-Based Configuration

A dedicated configuration interface will enable lecturers to define and manage their own custom allocation rules, thus providing lecturers with full control over session management.

3.2.7 Manual Override Capabilities

To accommodate exceptions such as late enrolments or academic exemptions, the system will support manual override functionalities. The lecturers or admins will have special rights that allow them to override the automated allocations when necessary.

3.2.8 Demmie Distribution

The system will include a functionality that allows lecturers to allocate demmies across multiple venues based on student group sizes or venue capacities. This ensures optimal support coverage and effective session management.

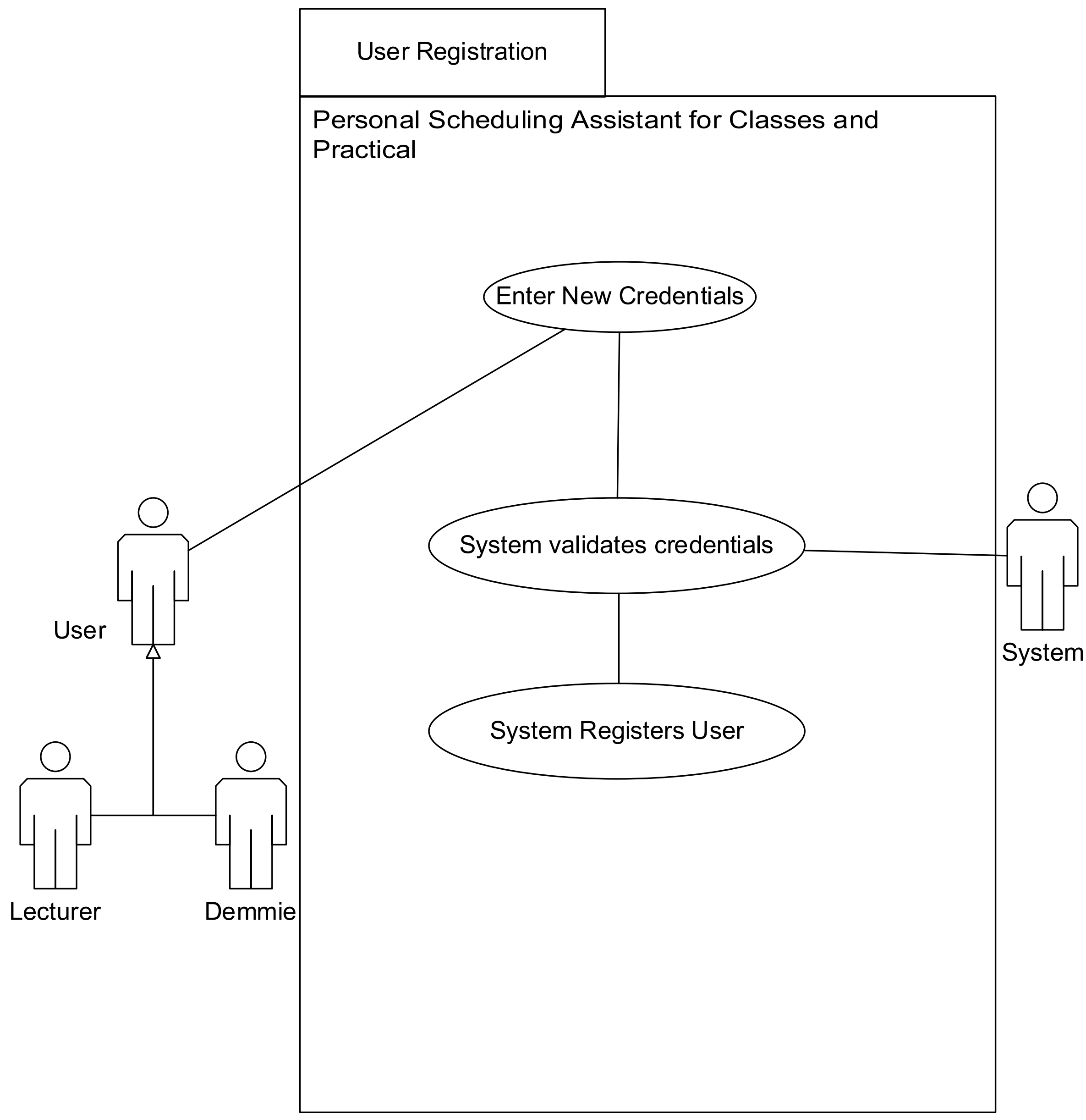
3.2.9 User Notifications

An integrated notification system will automatically inform students and lecturers of key scheduling events like reassignments, session changes and conflict resolutions. These notifications will be delivered via email to ensure timely communication.

3.3 Use Cases

3.3.1 User Registration

Use Case Diagram


User Registration
Personal Scheduling Assistant for Classes and Practical
Enter New Credentials
System validates credentials
System Registers User
User
System
Lecturer
Demmie

Description

All first-time users of the system will be required to register a profile with a username and a password before gaining access to the system.

Step by step

1. The user accesses the system registration page.

2. They fill in the required details which are name, staff number, email address, department, password and role (Lecturer or Demmie).

3. The system validates the email, staff number and password for validity and uniqueness.

4.1. If the provided staff number, email address, password **do not already exist** in the current database of users, the system saves the information and confirms successful registration.

4.2. If the provided credentials staff number, email address, password **do exist** in the current database of users, the system pops up an error, 'This user already exists,' and prompts the user to try again.

5. The user can then use their newly registered email as a username and use the newly created password to login into the system in future.

Best Case Scenario

1. Prof. Michaels visits the system's registration page.

2. She fills in her details as required including names, staff number, department, role and her password.

3. She proceeds to click "Register".

4. The system validates her credentials with the following steps:

4.1. It confirms the email format is valid and not already registered.

4.2. Checks if the staff number is valid and not duplicated.

4.3. Confirms the password is strong and both fields match

5.1. If all the information is valid, the system saves her profile successfully and shows a message, "Registration successful."

6. Prof. Michaels is then redirected to the login page.

Worst Case Scenario

1. Thabo goes to the registration page.

2. He fills out the form and provides all the required fields (First and last names, staff number, department, role and password).

3. He clicks “Register”.

4. The system displays 2 errors: “Staff number already in use” and “Password must be at least 8 characters and include a symbol”.

5. Thabo tries again and updates his staff number to ‘7891011’ and uses ‘ThaboPass’ as his new password.

6. Clicks Register again.

7. The staff number turns out to be correct but now the system displays another error: “Password must include a number.”

8. Thabo grows frustrated but eventually enters a valid password.

9. The system successfully saves his credentials and he is redirected to the login page.

Alternative Scenario

1. Thabo returns as a Demmie for the second semester.

2. He opens the system’s registration page to register for the new semester.

3. He fills in her names, staff number, department, role and password in the available fields.

4. He clicks “Register”.

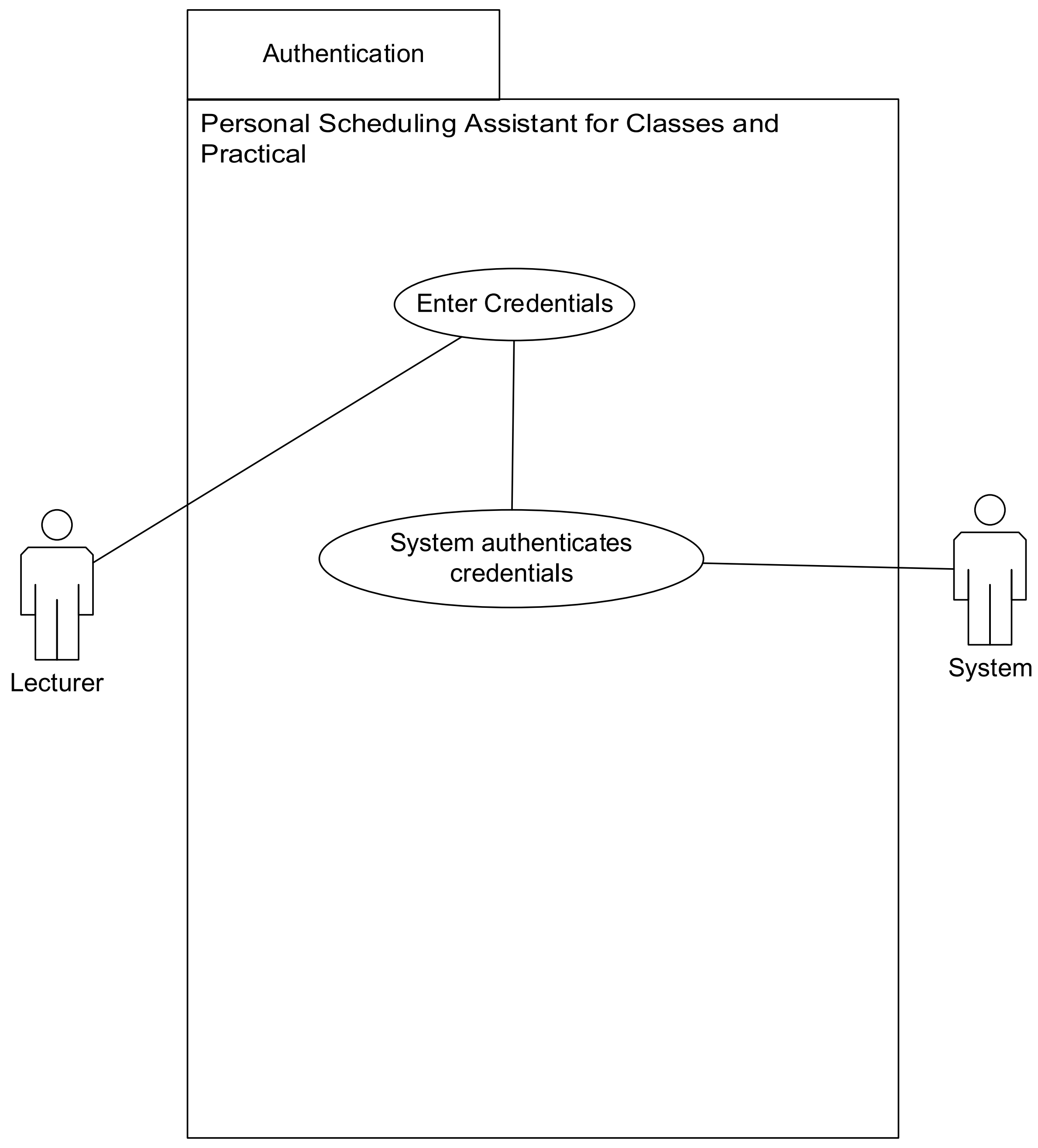
5. The system attempts to validate her credentials.

6. This time the system returns an, “Email or staff number already exists” error.

7. Upon realising the system had already captured her details from the first semester, she proceeds to the login page and logs on successfully.

3.3.2 Authentication

Use Case Diagram


Authentication
Personal Scheduling Assistant for Classes and Practical
Enter Credentials
System authenticates credentials
Lecturer
System

Description

This functionality controls access to the system by authenticating all the users that will be using the system.

Step-by-step

1. Lecturer navigates to the application login page.

2. The user then proceeds to enter their login credentials which are the username and their password.

3. Upon clicking the login button, the system then validates the credentials of the user by cross-checking them against the established database of users.

4.1. If the user's credentials do not match the credentials any of the registered users, the user is prompted to re-enter both the username and password.

4.2. If the user's credentials do match the credentials of one of the registered users, the user is granted access to the system.

5. The user is then directed to the home page.

Best Case Scenario

1. Dr. Molefe navigates to the login page of the application.

2. He enters her correct username and password in the provided text fields.

3. The system verifies his credentials by cross-checking them with the records in the user database.

4. The system confirms his role as a lecturer.

5. He is successfully authenticated and redirected to the home page dashboard.

Worst Case Scenario

1. Dr. Molefe attempts to log in but uses an incorrect password.

2. The system notifies him that his credentials are incorrect and he has 2 attempts remaining.

3. He tries twice more, still using the wrong password.

4. The system automatically locks her account after 3 failed attempts to prevent unauthorized access.

5. Dr. Molefe contacts Mr. Sibanda, the system administrator via email.

6. Mr. Sibanda unlocks Dr. Molefe’s account and sends him a secure password reset link.

7. Dr. Molefe resets his password and logs in successfully.

Alternative Scenario

1. Dr. Molefe forgets his password and clicks the “Forgot Password” link.

2. The system prompts him to enter his official lecturer email.

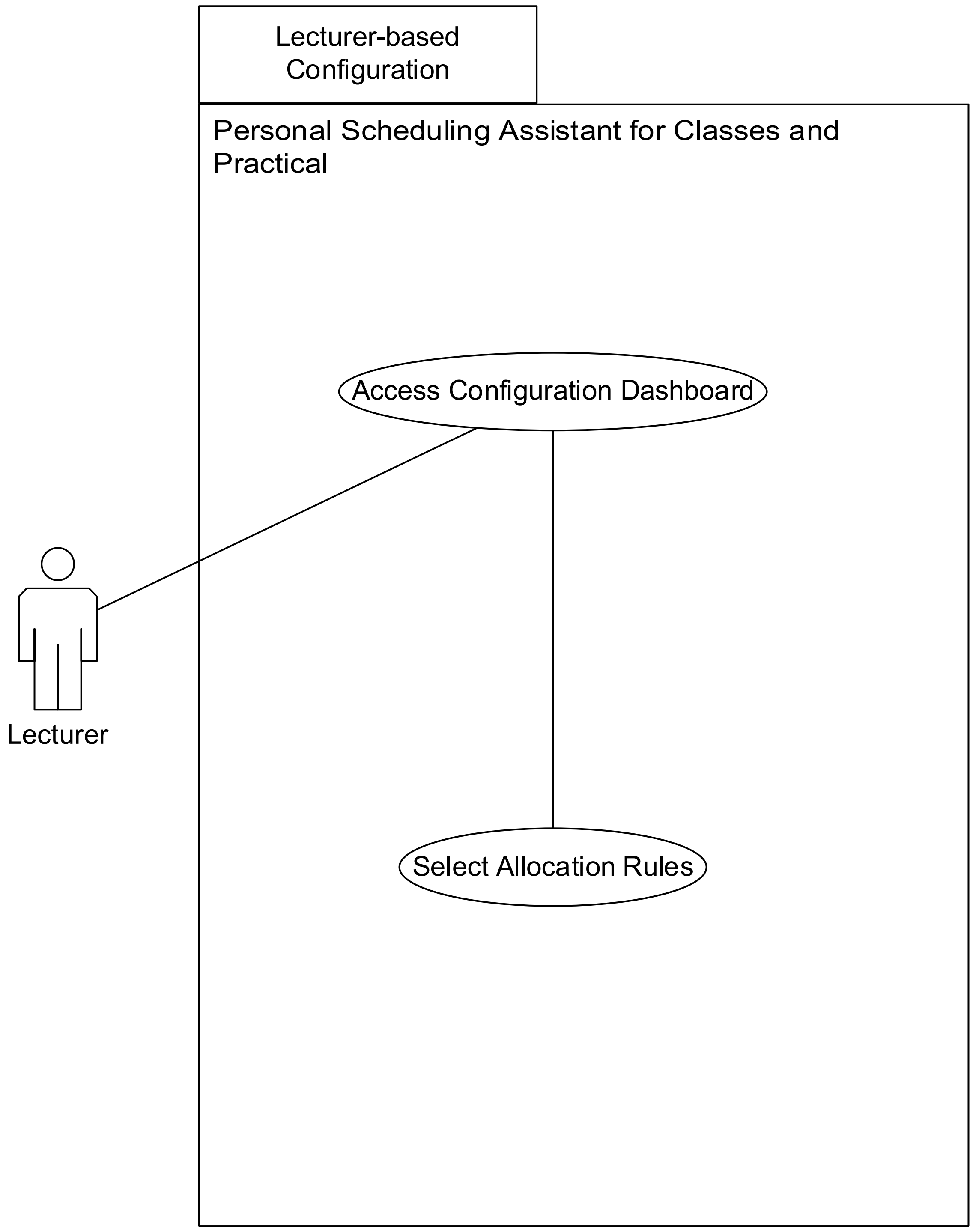
3. A password reset link is emailed to her staff account.

4. She follows the link, sets a new secure password and confirms it.

5. Dr. Molefe logs in successfully and resumes her duties.

3.3.3 Lecture-based Configuration

Use Case Diagram


Lecturer-based Configuration
Personal Scheduling Assistant for Classes and Practical
Access Configuration Dashboard
Select Allocation Rules
Lecturer

Description

With this use case, lecturers will be able to select session allocation rules using a dedicated configuration interface.

Step-by-step

1. User accesses the configuration dashboard.

2. User selects the relevant course to which they want to apply the changes.

3. User inputs or adjusts the allocation rules (e.g. venue preference, round robin).

4. The rules are saved and applied to the sessions.

5. The system validates and displays the rule preview.

6. The user confirms and saves the configuration.

7. Rules are stored and applied during session allocation.

Best Case Scenario

1. Dr. Naidoo logs into the configuration dashboard.

2. She selects her course BCIS2323.

3. She inputs her preferred allocation rules:

- Venue preference: Lab A and Lab B.
- Allocation type: Round robin.
- Group size limit: 20 students per session.

4. She clicks “Apply Rules”.

5. The system validates the input and displays a preview of how students will be distributed.

6. Dr. Naidoo confirms and saves the configuration.

7. The rules are successfully stored and reflected in the system.

8. During automated session allocation, the system uses the saved rules to distribute students accordingly.

Worst Case Scenario

1. Dr. Naidoo navigates to the configuration dashboard.

2. She selects to configure CSIS1535.

3. She mistakenly sets the venue preference to Lab C, not realising that it was already booked manually and is therefore unavailable.

4. She clicks “Apply Rules”.

5. The system flags the double-booking error and Dr. Naidoo is forced to change the lab to a different venue.

6. She confirms and saves the configuration.

7. The rules are successfully stored and reflected in the system.

Alternative Scenario

1. Dr. Naidoo navigate to the configuration dashboard.

2. She selects her course BCIS2323.

3. She selects ‘default’ as her allocation rules:

4. The system uses the ‘first come, first served’ rule as the default.

5. She clicks “Apply Rules”.

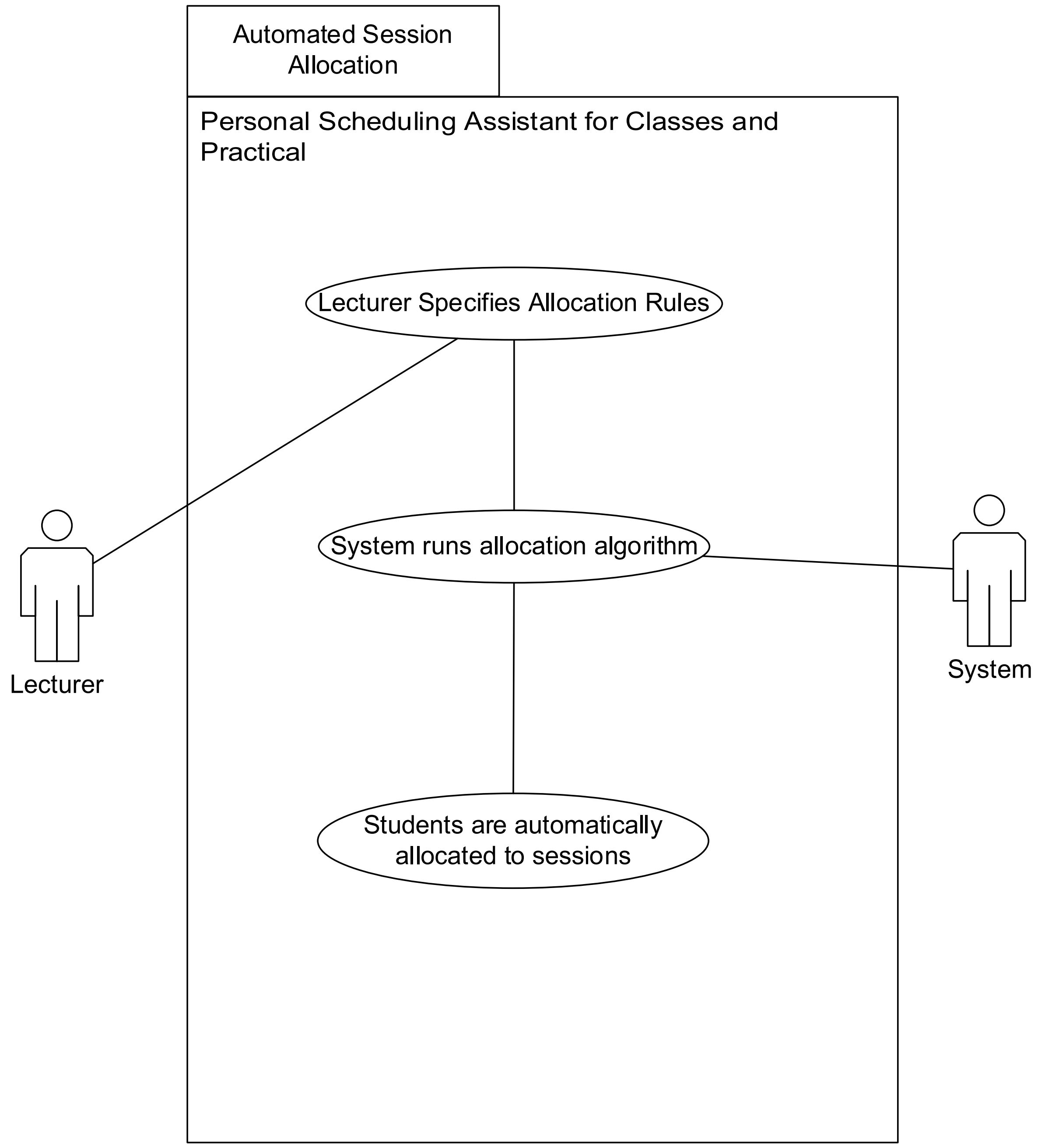
6. And the system validates the input and displays a preview of how students will be distributed.

7. She confirms and saves the configuration.

8. The rules are successfully stored and reflected in the system

3.3.4 Automated Session Allocation

Use Case Diagram



Automated Session Allocation
Personal Scheduling Assistant for Classes and Practical
Lecturer Specifies Allocation Rules
System runs allocation algorithm
Students are automatically allocated to sessions
Lecturer
System

Description

This use case demonstrates how the system assigns students to various sessions automatically based on rules set by lecturers.

Step-by-step

1. User navigates to the session configuration page.

2. User inputs or updates the allocation rules (round-robin; first-come, first-served, etc).

3. The system validates the entered rules.

4. The system runs the allocation algorithm.

5. Students are automatically allocated to sessions by the system.

6. The new allocations are stored in their own databases grouped by venue.

7. The allocations are then reflected in student profiles.

Best Case Scenario

1. Dr. Parker accesses the Session Configuration page for CSIS2343 practicals.

2. He specifies the distribution rule as round-robin and sets a session limit of 25 students per venue.

3. The system validates the rule and checks room availability.

4. The system then distributes a total of 73 students across 3 different venues,

5.1. If the lecturer specifies to prioritise venue capacity, the students are then distributed across all three venues with a maximum of 25 students in each venue (Lab A: 25, Lab B: 25, Lab C: 23).

5.2. If the lecturer specifies to distribute the students equally, the students are then distributed across all three venues equally as far as possible (Lab A: 23, Lab B: 24, Lab C: 24).

6. Dr. Parker then reviews and approves the distribution

Worst Case Scenario

1. Dr. Parker accesses the Session Configuration page for CSIS2343 practicals.

2. He specifies the distribution rule as ‘first come, first served’ and neglects to input the venue capacity limit.

3. The system validates the rule and checks room availability.

4. The system allocates 40 students to Lab A, which can only accommodate 30.

5. This leads to students being stranded for majority of the practical till an alternative venue is found.

Alternative Scenario

1. Dr. Parker accesses the Session Configuration page for CSIS2343 practicals.

2. He specifies the distribution rule as ‘random’ across three different venues.

3. The system validates the rule and checks room availability.

4. The system automatically assigns students accordingly.

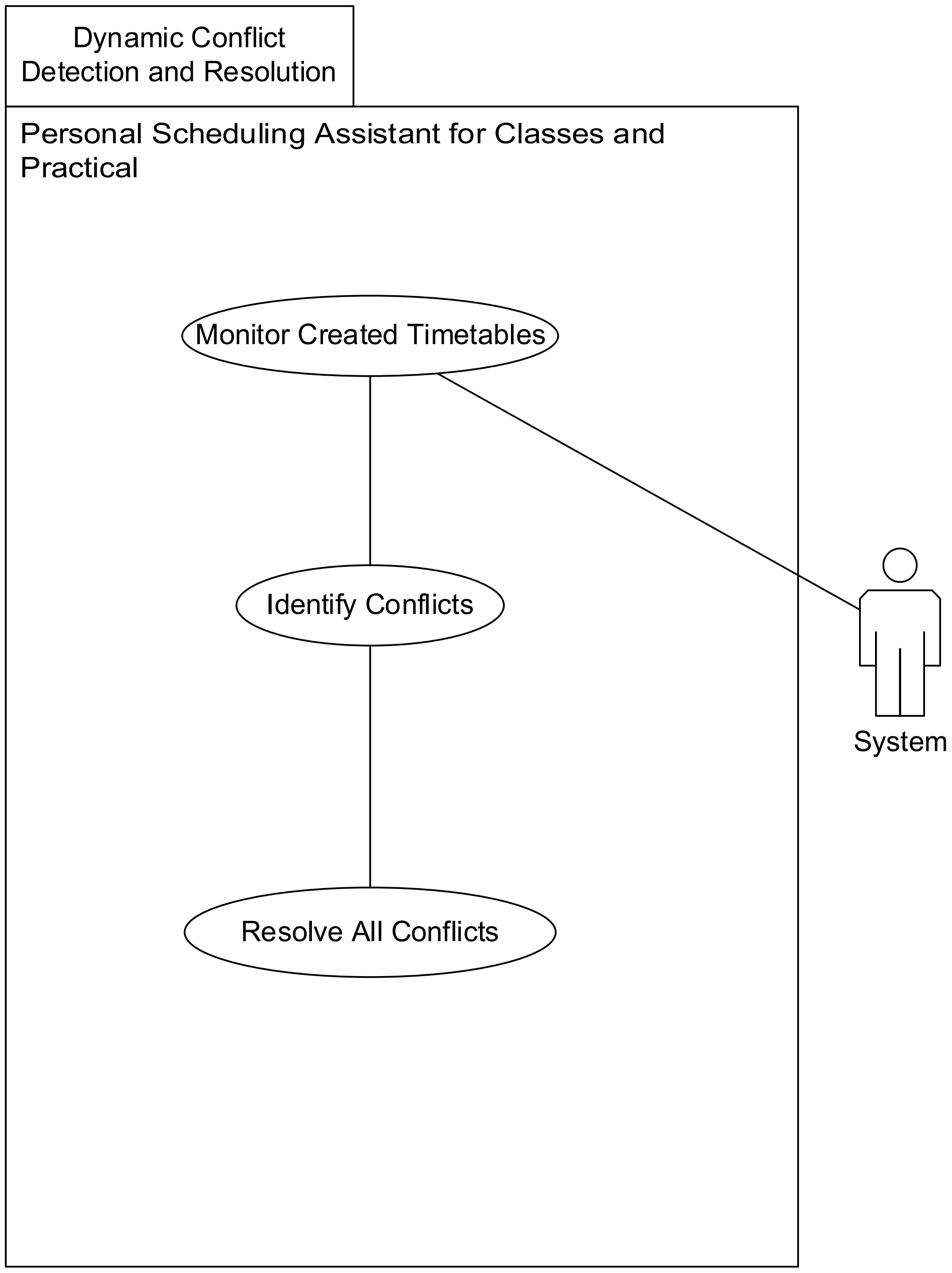
5. A student, Lunga, who lives off-campus and struggles to make it to campus early sends the lecturer an email requesting to be placed in the evening session.

6. Dr. Parker acknowledges Lunga’s issue and uses the manual override feature to replace Lunga in a different session.

7. The system updates the database and Lunga’s profile is corrected within minutes.

3.3.5 Dynamic Conflict Detection and Resolution

Use Case Diagram


Dynamic Conflict Detection and Resolution
Personal Scheduling Assistant for Classes and Practical
Monitor Created Timetables
Identify Conflicts
Resolve All Conflicts
System

Description

This feature allows the system to monitor all the conflicts created along with the schedules and dynamically resolves them with no human intervention.

Step-by-step

1. After the system creates all the students' timetables.

2. The system runs the conflict resolution algorithm.

3. It identifies overlapping sessions for each student or if there are any duplicate sessions for each student.

4. The system then analyses the available alternative sessions.

5. It applies the resolution logic.

6.1. If a suitable alternative session exists, students are then reassigned to the respective alternative sessions and conflicts are resolved automatically.

6.2. If an alternative does not exist, the changes are not saved.

7. The affected timetables are then updated with the new changes.

8. Users are notified of changes.

Best Case Scenario

1. The system finalizes Naledi's timetable after session allocations.

2. It then runs the conflict detection algorithm.

3. It detects that Naledi has two practicals scheduled on Wednesday between 14:00-16:00, one for BCIS2432 and another for EBUS1452.

4. The conflict detection algorithm identifies this as an overlapping conflict.

5. The system scans for available alternatives.

6. It finds a Friday slot between 10:00-12:00 for EBUS1452 with available space.

7. The system reassigns Naledi to the Friday EBUS1452 practical.

8. Naledi's updated timetable is regenerated and saved.

Worst Case Scenario

1. The system finalizes Naledi’s timetable and initiates conflict detection.

2. It identifies an overlap between CSIS2442 and STSA2432, both scheduled for Monday 17:00–19:00.

3. The system scans for alternatives, but all the other CSIS2442 sessions are at capacity and STSA2432 only runs once.

4. The system identifies that no viable alternative sessions exist.

5. The system flags the conflict but cannot resolve it automatically.

6. The admin is then notified of the conflict.

7. The system saves the conflict status but does not make changes.

Alternative Scenario

1. The system finalizes Naledi’s timetable and the conflict detection is executed.

2. It identifies an **hour** overlap between CSIS2442 and EBUS2424, one scheduled for Thursday 15:00-17:00 and the other 16:00-18:00, respectively.

3. The system finds no alternative sessions after scanning through the entire database.

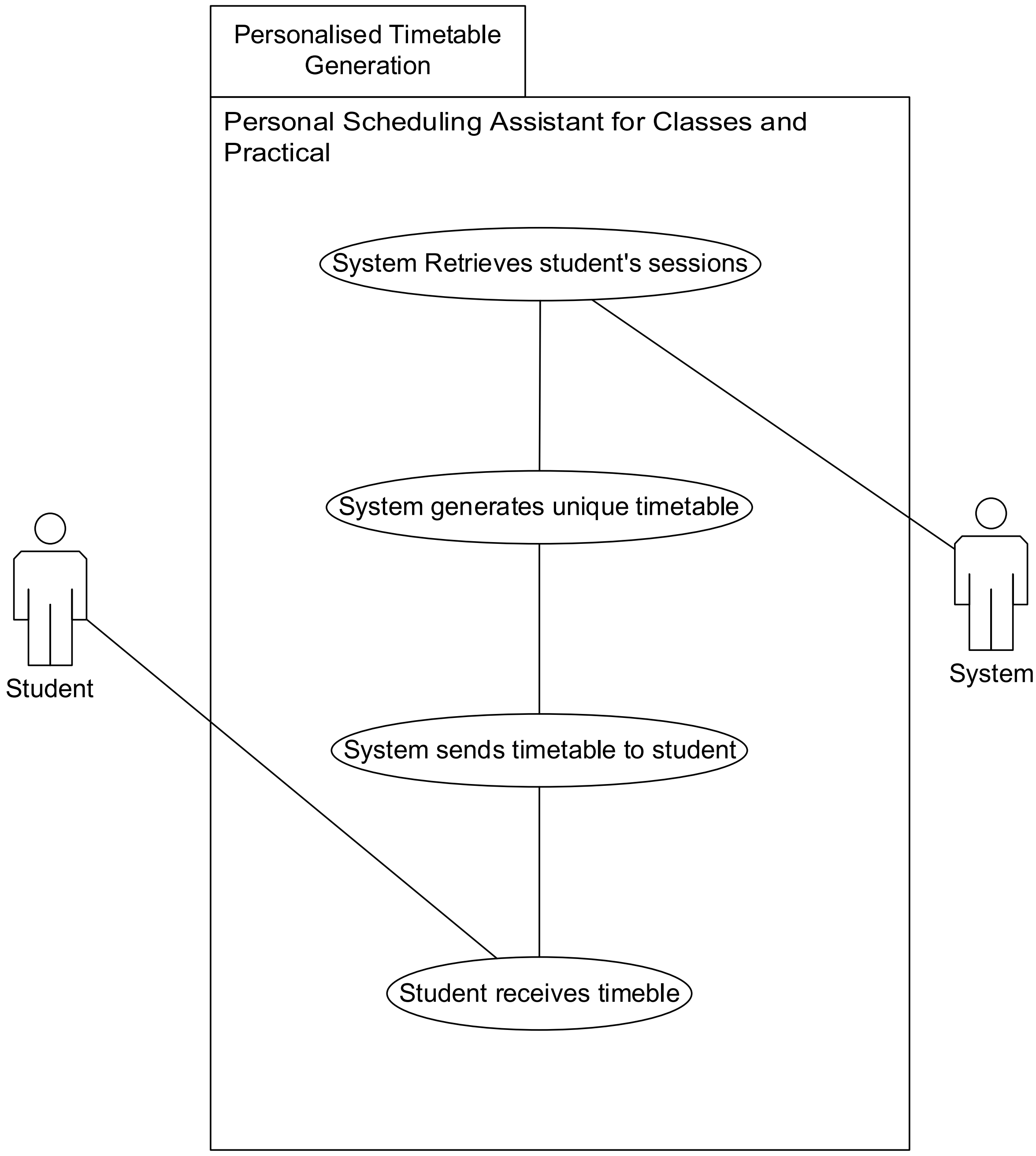
4. The system flags the conflict and notifies the admin of the conflict.

5. The admin grants Naledi the permission to arrive an hour later for the EBUS2424 session (17:00) in order to accommodate the session before.

6. The admin then clears the conflict manually.

3.3.6 Personalised Timetable Generation

Use Case Diagram


Personalised Timetable Generation
Personal Scheduling Assistant for Classes and Practical
System Retrieves student's sessions
System generates unique timetable
System sends timetable to student
Student receives timeble
Student
System

Description

This functionality will generate a unique timetable for each student that includes only the specific sessions to which the students are assigned.

Step-by-step

1. Upon completion of the student allocations by the system, the system retrieves session allocations from the created Allocations database.

2. The system uses the data from the student’s allocations and registered modules to compile a personalised timetable.

3. The generated timetables are sent to the students.

4. Students are able to view and optionally download the timetable.

Best Case Scenario

1. The system fetches Thandi’s modules from the database.

2. Thandi is enrolled in three modules which are CSIS1664, CSIS1212 and CSIS3142.

3. The system compiles a personalised timetable using the retrieved modules.

4. The timetable is emailed to Thandi and appears on her student portal.

5. She is able to view the schedule and attend the correct sessions without confusion.

Worst Case Scenario

1. The system fetches Thandi’s modules from the database.

2. Thandi is enrolled in three modules which are CSIS1664, CSIS1212 and CSIS3142 in the CSI department.

3. The system compiles a general timetable with all the modules taken by all the students in the CSI department.

4. The timetable is then emailed to Thandi as it is.

5. This causes confusion for Thandi as she believes she has to attend all the modules presented in her timetable.

Alternative Scenario

1. Thandi registers a new module (STSA1234) one week after the semester starts.

2. The system had already completed initial timetable generation for all students. Upon detecting the late enrolment, the system flags her record.

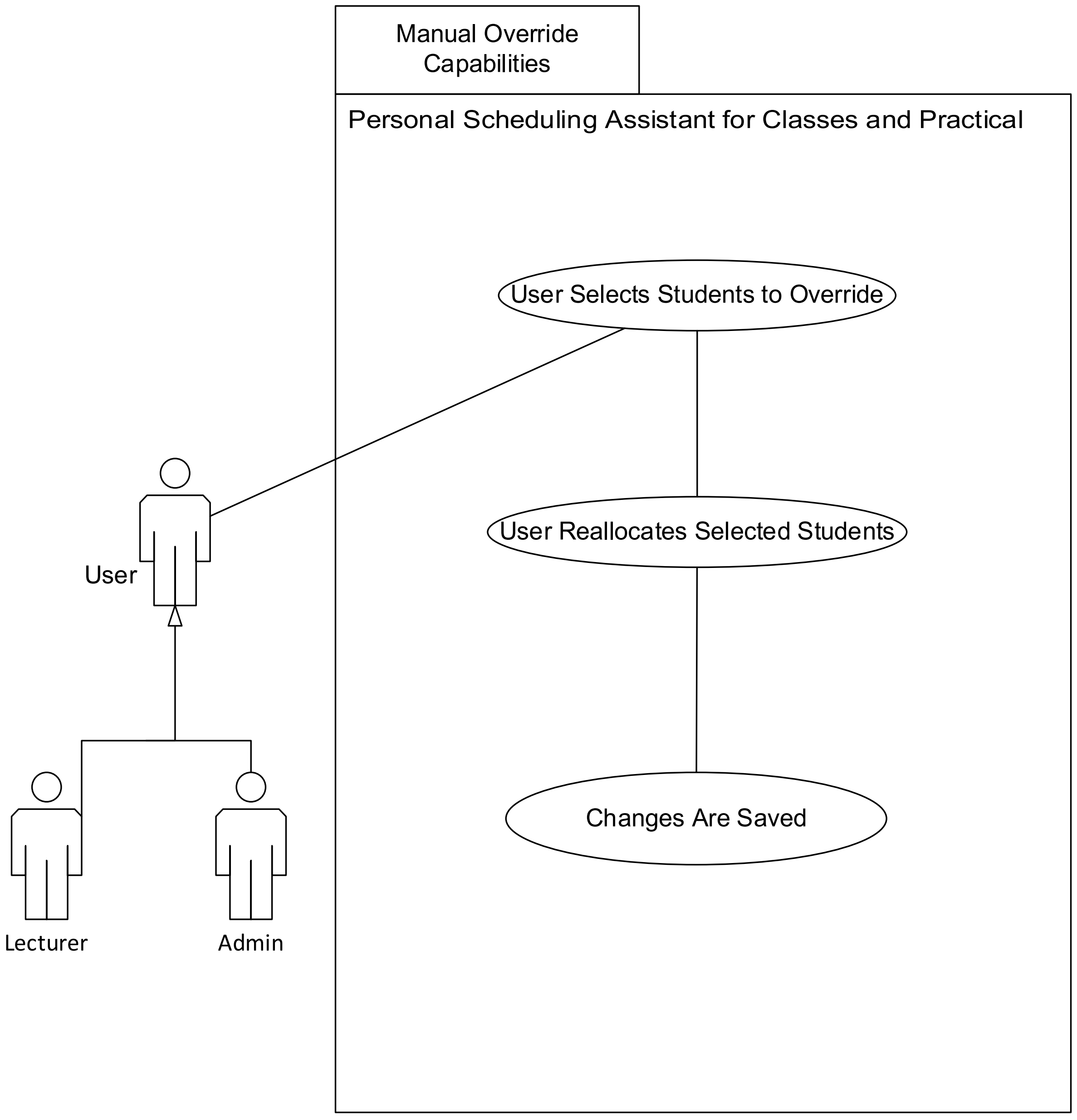
3. The allocation engine checks available STSA1234 practical sessions and assigns her to the least-full slot.

4. A new timetable is generated for her, now including STSA1234 on Tuesday at 13:00.

5. Thandi receives an updated version via email and her student portal.

3.3.7 Manual Override Capabilities

Use Case Diagram


Manual Override Capabilities
Personal Scheduling Assistant for Classes and Practical
User Selects Students to Override
User Reallocates Selected Students
Changes Are Saved
User
Lecturer
Admin

Description

Under special circumstances or upon request, lecturers and system admins will have the capability of overriding the schedules through this functionality.

Step-by-step

1. User accesses the overview of the allocations.

2. User searches for the student they intend to reallocate.

3. The user can then select and remove the student manually from their current session.

4. The student is then manually added to the newly selected session by the user.

5. The system first checks for potential conflicts and violations, which would be overlapping sessions and clashes.

6.1. If there are no conflicts, the system updates the student’s timetable with the newly scheduled classes and notifies the affected student.

6.2. If conflicts are found, the admin is notified and the changes are not saved.

Best Case Scenario

1. Zanele emails Prof. Mthembu requesting to change her BCIS lab session due to a clash with her part-time job.

2. Prof. Mthembu logs into the system and accesses the student allocation overview.

3. He searches for Zanele under the BCSI module.

4. He removes her from Lab Session A (Wednesdays 14:00–16:00) to Lab Session B (Fridays 08:00–10:00) as the alternative.

5. The system checks for timetable conflicts and room capacity.

6.1 If no conflicts are detected. The system updates Zanele’s timetable and sends her an email notification confirming the change.

6.2. If a conflict is detected, the system does not save the changes.

7. A confirmation message is displayed on-screen that the override was successful.

Worst Case Scenario

1. Lebo requests a session change due to overlapping sports practice.

2. Prof. Mthembu logs into the system and finds Lebo’s current allocation.

3. She moves Lebo to another group’s session on Thursday, 15:00–17:00.

4. Prof. Mthembu fails to remove Lebo from the first session, which leads to Lebo being enrolled in two sessions.

5. The system runs a conflict check and detects that Lebo is enrolled in two sessions and resolves the problem automatically.

6. The change is made and saved.

Alternative Scenario

1. The system has already finalized timetable allocations for the semester.

2. Tebogo approaches Prof. Mthembu asking for a session change due to personal obligations.

3. Prof. Mthembu accesses the override dashboard and locates Tebogo’s current allocation.

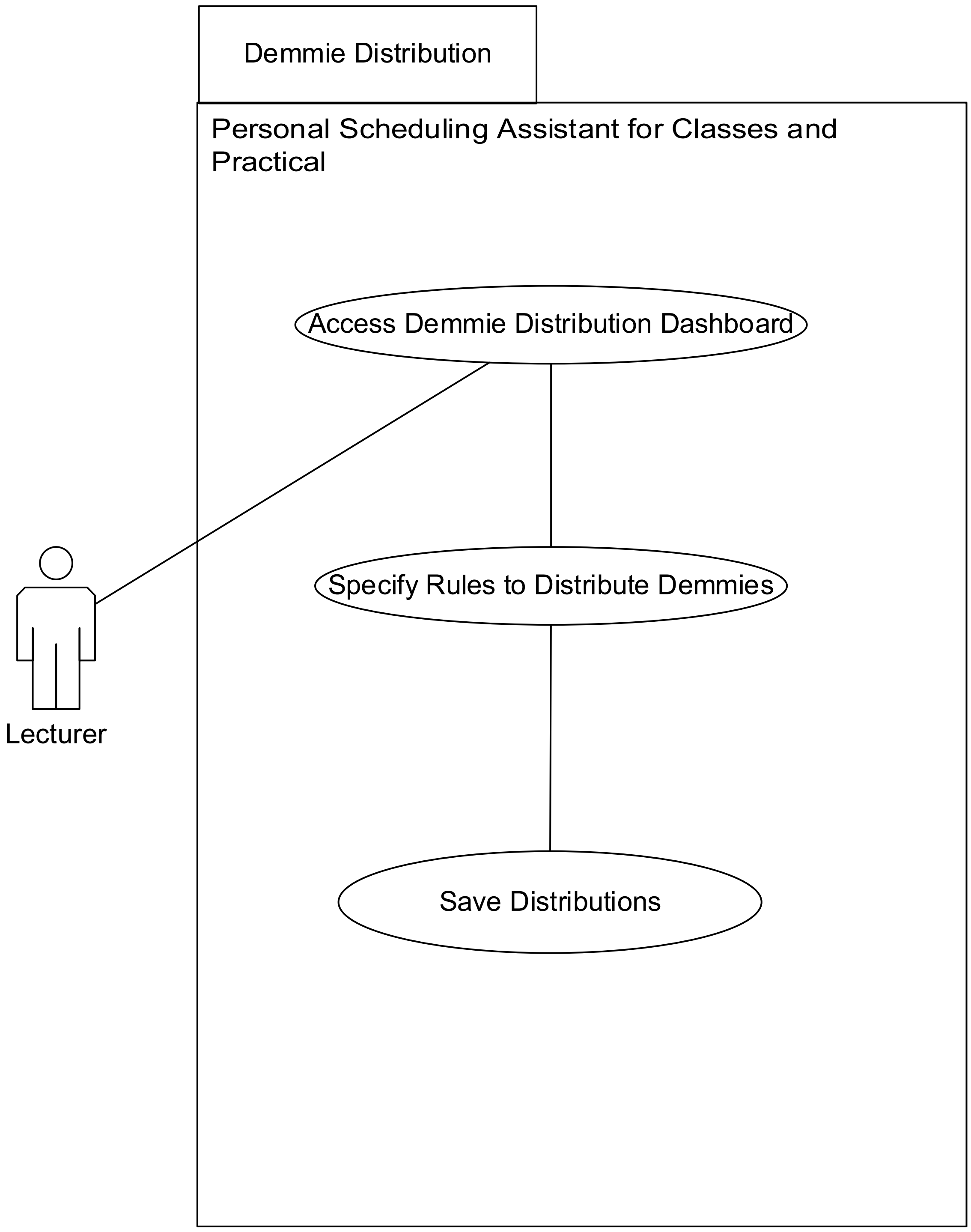
4. Prof. Mthembu realises that the alternative session Tebogo would prefer is already filled to capacity.

5. After learning this, Tebogo approaches one of his classmates from the alternative session in order to do a direct swap.

6. After both classmates come to the conclusion, they inform Prof Mthembu and he then makes a direct swap manually.

3.3.8 Demmie Distribution

Use Case Diagram


Demmie Distribution
Personal Scheduling Assistant for Classes and Practical
Access Demmie Distribution Dashboard
Specify Rules to Distribute Demmies
Save Distributions
Lecturer

Description

Lectures can choose how they want to distribute demmies across sessions with this feature.

Step-by-step

1. Lecturer accesses the demmie distribution interface.

2. The lecturer selects all the venues that will be in use.

3. The lecturer views how many students will be in each venue.

4. Lecturer assigns demmies to venues based on ratios or preferences.

5. Allocations are confirmed and saved.

6. Demmies are notified.

Best Case Scenario

1. Ms. Mathew accesses the demmie distribution interface for her CSIS1614 sessions.

2. She selected three venues: Lab A, Lab B and Lab C from the lecture configuration dashboard.

3. The system shows group sizes of 30 students, 25 students and 25 students across all labs, respectively.

4. She assigns Sipho and Amanda to Lab A, Thandi and Mike to Lab B and finally she assigns Mpho and Bandile to Lab C.

5. She confirms the distribution.

6. The system validates the input and saves the allocations.

7. Sipho and Thandi all the booked demmies receive emails with venue assignments and session times.

Worst Case Scenario

1. Mr. Mathew logs into the demmie distribution interface for his STSA1253 tutorial groups.

2. He selects Lecture Rooms 2 and 3 as venues.

3. He views the group sizes of 15 students and 17 students across the 2 venues respectively.

4. He assigns Thabo to Lecture Room 2 and Tshepang to Lecture Room 3.

5. However, Tshepang failed to declare that he has a class at the same time as the tutorial which leads to an unforeseen clash.

6. Upon reception of the email with venue assignments and session times, Tshepang realises that he cannot make it to the tutorial and he informs Mr. Mathew.

7. Mr Mathew is forced to find an emergency replacement for the tutorial.

Alternative Scenario

1. Ms. Daniels assigns Lerato to a Tuesday session at 13:00.

2. The allocation is confirmed and saved.

3. A day later, Lerato submits a schedule conflict to Ms. Mathew on time due to a doctor’s appointment.

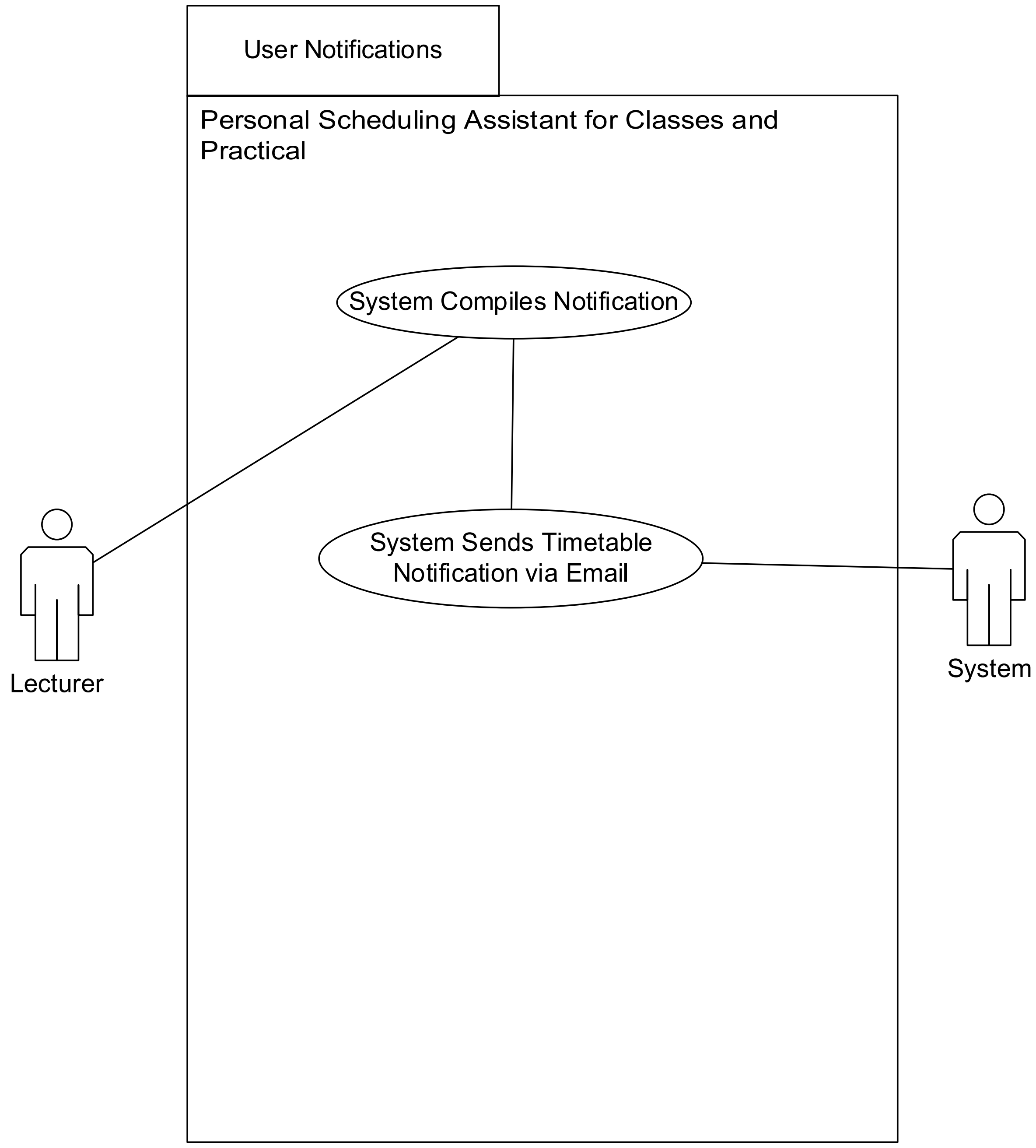
4. Ms. Daniels reopens the interface, removes Lerato from the Thursday slot and assigns another available demmie.

5. The system revalidates and updates notifications.

6. The new demmies are notified of the revised allocation.

3.3.9 User Notifications

Use Case Diagram


User Notifications
Personal Scheduling Assistant for Classes and Practical
System Compiles Notification
System Sends Timetable Notification via Email
Lecturer
System

Description

After creating all the student schedules, the system will send the timetables to the students via email.

Step-by-step

1. After making all the necessary changes in the previous steps, the lecturer confirms all the changes made to the schedule.

2. They are then prompted to confirm whether they want to send out all the emails with the new schedules.

3. The system generates the notification content in-app with the updated schedules.

4. The notification is then sent via email.

Best Case Scenario

1. Mrs. Kooper finishes session adjustments and clicks "Send Notifications."

2. The system validates the changes and begins generating emails.

3. Within minutes, all the students receive emails with accurate, neatly formatted timetables.

4. They then have access to download the PDF schedule with no issues and plan their weeks accordingly.

Worst Case Scenario

1. Mrs. Kooper makes the final changes to the sessions and clicks "Send Notifications."

2. The system validates the changes and begins generating emails.

3. Due to a configuration error, the system attaches incorrect timetables that are not personalised to the emails.

4. This leads to a lot of confusion for the students.

5. After informing Mrs Kooper, she rectifies the error and resends the emails.

6. Students are later notified of the corrected schedule.

Alternative Scenario

1. Mrs Kooper sends out the notification emails.

2. The system processes most emails successfully but a few are not sent due to outdated or invalid email addresses.

3. Josh, whose email had a typo in the system, doesn’t receive his timetable and he informs Mrs Kooper.

4. Mrs. Kooper makes an announcement for the rest of the affected students to notify her.

5. After receiving the list, she checks the system and updates their contact info and manually re-sends their timetables.

3.4 Chapter Overview

This chapter presented the comprehensive requirements and analysis workflow for the Personal Scheduling Assistant for Classes and Practicals. Each functionality, from user registration and authentication to conflict detection, personalized timetable generation and demmie distribution was designed to address real-world academic scheduling challenges.

The use case descriptions provided practical demonstrations of how end-users will interact with the system, ensuring that all key processes are user-centred and goal-oriented. These models collectively serve as the blueprint for the system’s development, providing a logical foundation for both the design and implementation phases presented in the following chapters.

4. Chapter 4: Architectural Design

4.1 Introduction

Building upon the requirements analysis outlined in Chapter 3, this chapter presents the architectural design of the Personal Scheduling Assistant for Classes and Practicals system. The architectural design serves as the blueprint for implementation, defining how various system components interact and how data flows between them.

This chapter presents a detailed architectural design of the system, providing a comprehensive foundation for its implementation. It will outline the structural blueprint through a site map to illustrate the navigational flow and system modules. A detailed Entity-Relationship (ER) diagram that emphasizes referential integrity and the relationships between core entities will follow to illustrate the database design.

Additionally, the chapter offers a justification for the selected technologies, tools and frameworks used in building the system. It will reference the project objectives, user needs and design constraints. Together, these elements ensure that the system is scalable, coherent and aligns with the functional requirements defined in the previous chapter.

4.2 Sitemap

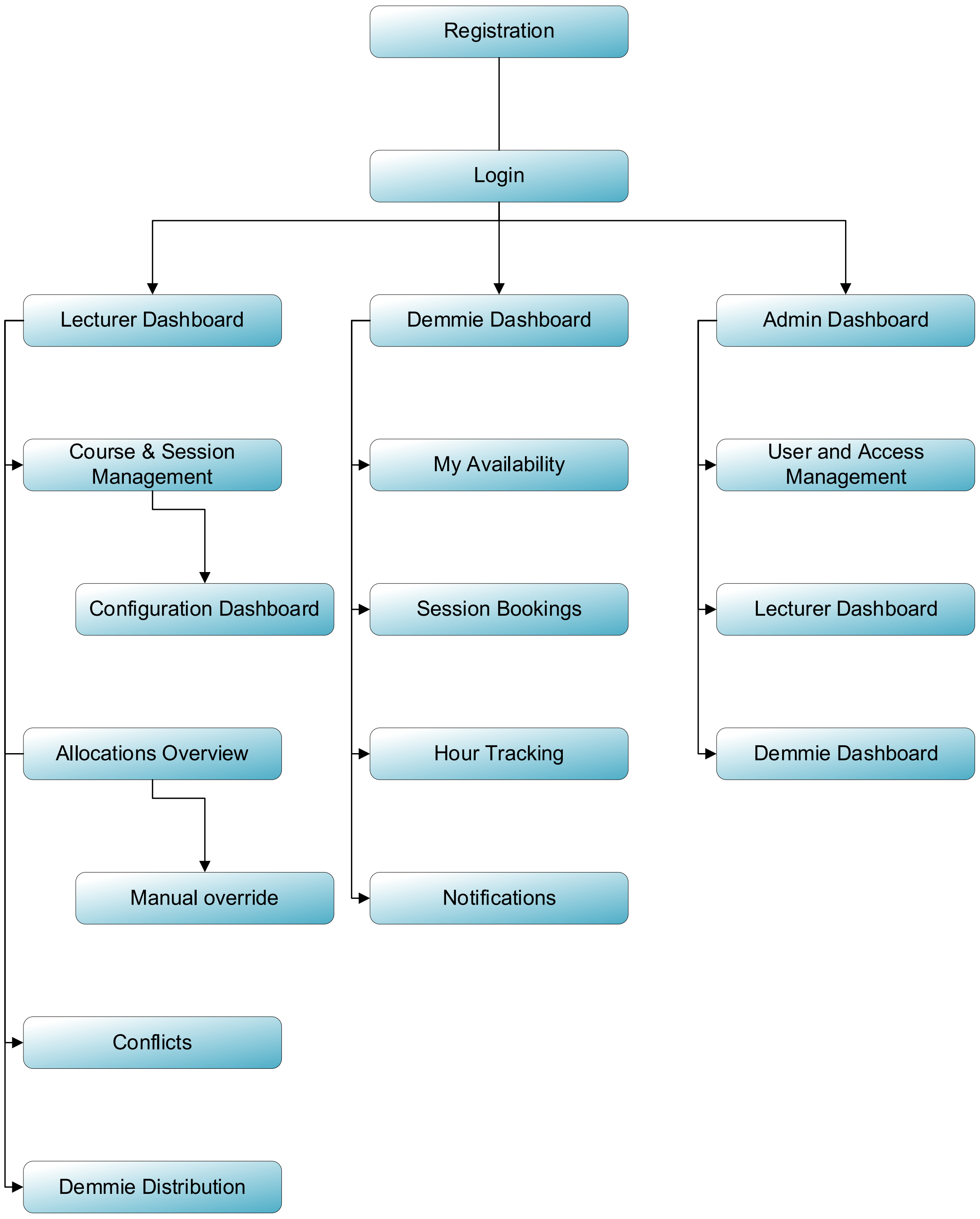


Figure 2: Sitemap

4.3 Database Design

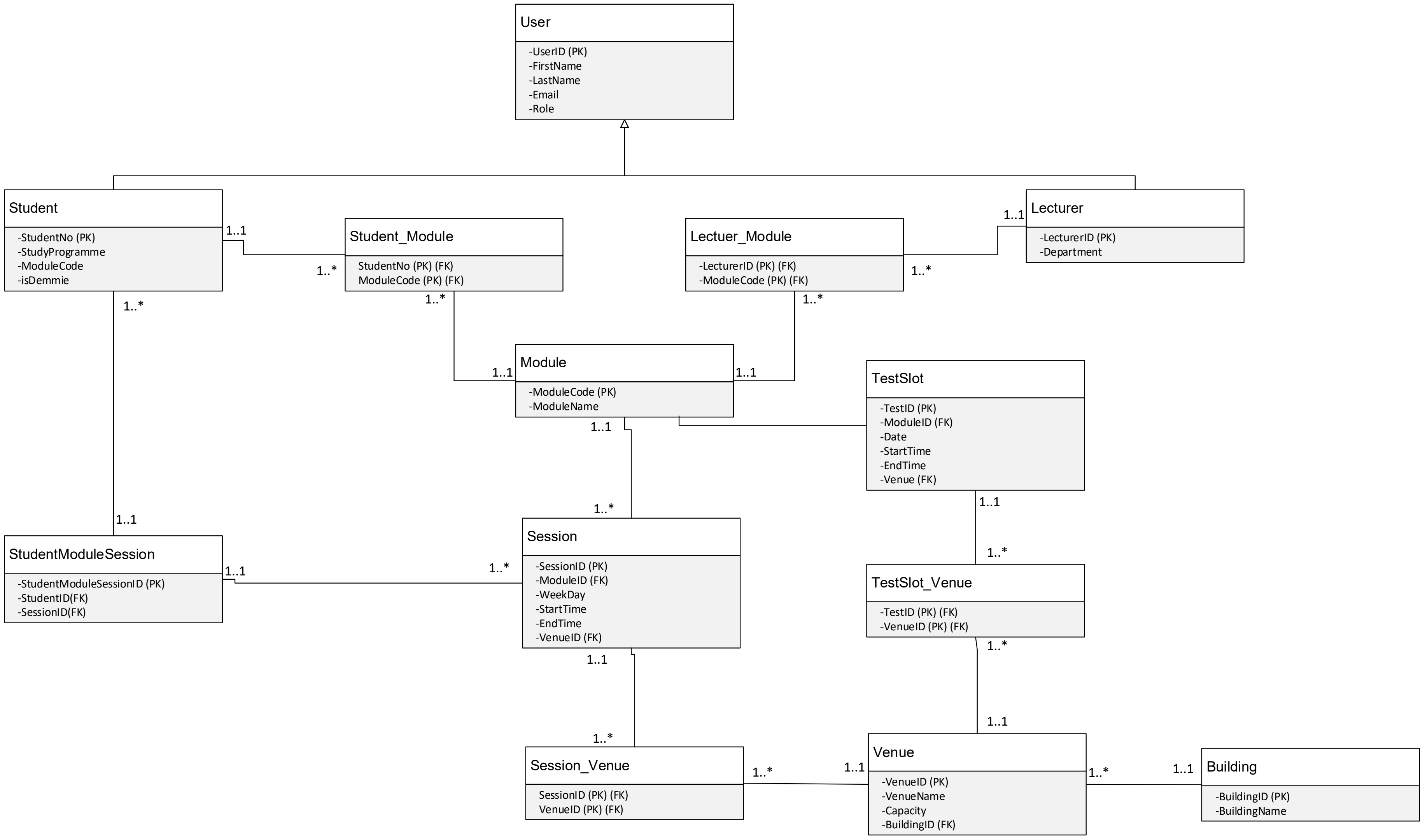


Figure 3: ERD

4.4 Justification For Technology

A variety of technologies will be used to develop the system. The choice of technologies for this system was guided by the need for a secure, scalable and maintainable solution that can efficiently support large volumes of data and concurrent users. Below is a discussion on what technologies were used together with the reasons they were selected.

Integrated Development Environment

Visual Studio 2022

Microsoft Visual Studio is an IDE (integrated development environment) created by Microsoft and used for different types of software development. It is one of the widely used IDEs and has proven to be very reliable. With various features that increase one's productivity like IntelliSense and advanced debugging utilities, it is an ideal choice for building this project (Wickramasinghe, 2024).

Programming Languages

C#

C# was selected as the primary programming language for this particular project due to its strong support for modular and object-oriented development, which significantly enhances code maintainability and scalability (Dang, 2024) . As a cross-platform, general-purpose language, C# enables developers to be highly productive while writing highly performant code. It provides developers with a rich set of libraries and a mature ecosystem that is well-suited for high-grade applications.

One of the key advantages of using C# lies in its seamless integration with the .NET Framework, which is a free, cross-platform, open source development environment. With millions of developers, C# is the most popular .NET language.

Another major advantage of using C# is its integration with the ASP.NET Core and Entity Framework Core frameworks. ASP.NET simplifies the creation of dynamic and interactive web applications. It provides powerful tools and libraries for building web pages and real-time applications. With its robust features like model-view-controller (MVC) architecture and built-in authentication, ASP.NET was the best choice for this project.

Entity Framework Core on the other hand simplifies database interactions in web applications. It will allow me to work with databases by using object-oriented principles, eliminating the need for manual SQL queries.

Together, these frameworks offer powerful tools for building secure, database-driven web applications and include built-in support for user authentication and authorization which are critical features for ensuring role-based access control and data confidentiality in the academic domain.

HTML, CSS, and JavaScript

For the frontend development of the project, the project will leverage standard web technologies such as HTML, CSS and JavaScript, ensuring responsive design and a seamless user experience across different screen sizes and input devices. The system is designed to be platform-independent, accessible from any modern device with internet connectivity, this includes desktops, laptops and tablets running various operating systems.

Database Management System

Microsoft SQL Server

To support the system's data-intensive operations, Microsoft SQL Server was chosen as the database management system. SQL Server is optimised for handling large-scale data operations and provides high reliability, robust transaction support and tools for ensuring referential integrity. Its tight integration with C# enables the building of robust and scalable applications that are efficient with querying and data manipulation (Godel, 2025).

4.5 Chapter Overview

This chapter outlined the architectural design of the Personal Scheduling Assistant for Classes and Practicals system. It began with a sitemap illustrating the navigational flow across different user roles, followed by a database design that ensures efficient and secure data handling through well-defined relationships and integrity constraints.

The chapter also justified the selection of technologies and frameworks that form the backbone of the system's implementation. The chosen stack comprising C#, ASP.NET Core MVC, Entity Framework Core and Microsoft SQL Server provides the scalability, modularity and security necessary for a university-level scheduling application.

Together, these architectural and technological decisions establish a solid foundation for the system implementation, which will be discussed in the following chapter.

5. Chapter 5: Source Code and Implementation

5.1 Introduction

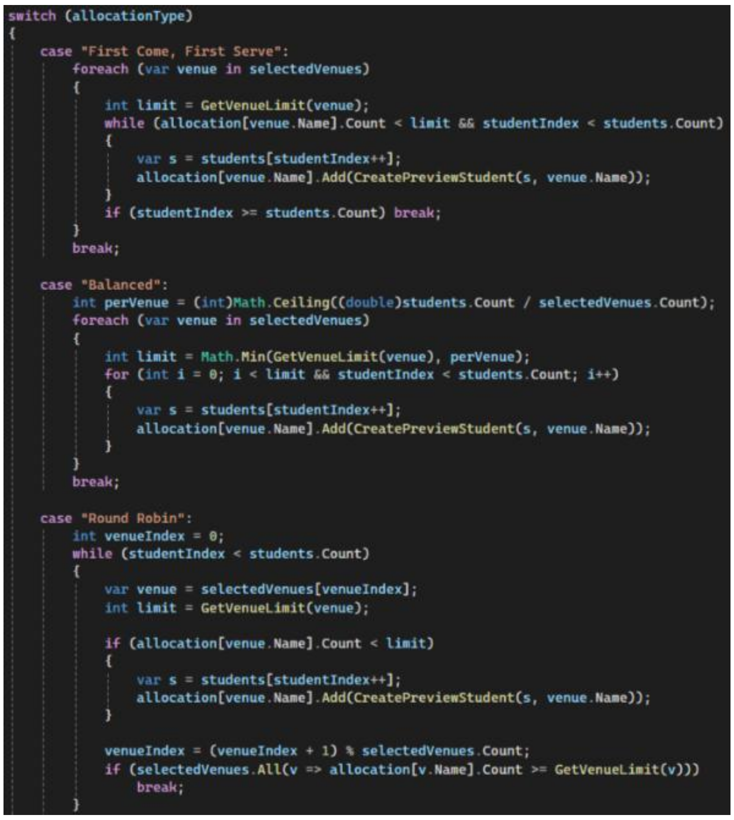
Building on the architectural foundation discussed in Chapter 4, this chapter will present the source code implementation of the ‘Personal Scheduling Assistant for Classes and Practicals’ system. The purpose of this chapter is to demonstrate the technical realization of the system’s core functionalities, focusing on the design patterns, algorithms and coding decisions that drive its operation.

This chapter provides excerpts of key coding sections that best illustrate the system’s logic and the complexity of the implementation process. These include the automatic allocation logic, conflict management, schedule generation, notification management and demmie distribution. Each excerpt is accompanied by an explanation that clarifies its purpose, logicyue and the rationale behind its implementation choices.

As mentioned in the previous chapter (Chapter 4), the system was developed using C#’s ASP.NET Core MVC framework with Entity Framework Core for data access and management. The Model-View-Controller (MVC) architecture was chosen to maintain a clear separation of concerns, ensuring that the system remains scalable, modular and easy to maintain.

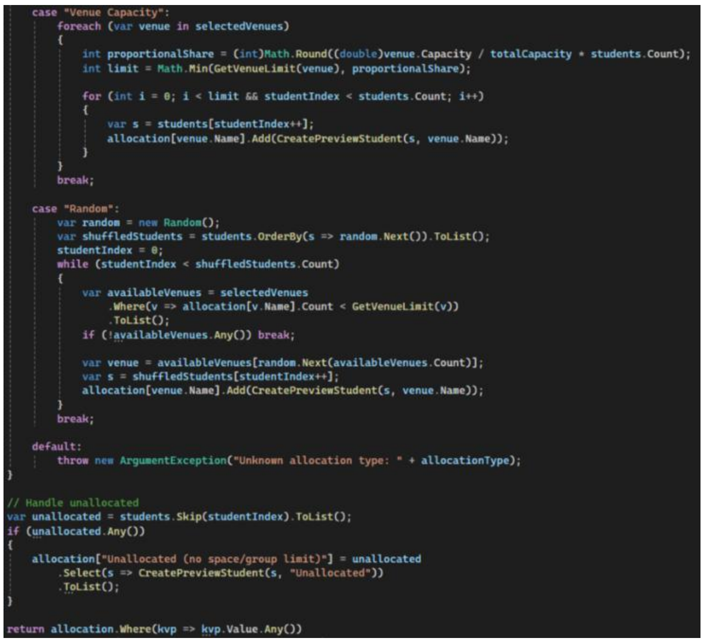
5.2 Methods and Materials for evaluation

5.2.1 Allocation Logic



```
switch (allocationType)
{
    case "First Come, First Serve":
        foreach (var venue in selectedVenues)
        {
            int limit = GetVenueLimit(venue);
            while (allocation[venue.Name].Count < limit && studentIndex < students.Count)
            {
                var s = students[studentIndex++];
                allocation[venue.Name].Add(CreatePreviewStudent(s, venue.Name));
            }
            if (studentIndex >= students.Count) break;
        }
        break;

    case "Balanced":
        int perVenue = (int)Math.Ceiling((double)students.Count / selectedVenues.Count);
        foreach (var venue in selectedVenues)
        {
            int limit = Math.Min(GetVenueLimit(venue), perVenue);
            for (int i = 0; i < limit && studentIndex < students.Count; i++)
            {
                var s = students[studentIndex++];
                allocation[venue.Name].Add(CreatePreviewStudent(s, venue.Name));
            }
        }
        break;

    case "Round Robin":
        int venueIndex = 0;
        while (studentIndex < students.Count)
        {
            var venue = selectedVenues[venueIndex];
            int limit = GetVenueLimit(venue);

            if (allocation[venue.Name].Count < limit)
            {
                var s = students[studentIndex++];
                allocation[venue.Name].Add(CreatePreviewStudent(s, venue.Name));
            }

            venueIndex = (venueIndex + 1) % selectedVenues.Count;
            if (selectedVenues.All(v => allocation[v.Name].Count >= GetVenueLimit(v)))
                break;
        }
```

Figure 4: Allocation Algorithm



```
    case "Venue Capacity":
        foreach (var venue in selectedVenues)
        {
            int proportionalShare = (int)Math.Round((double)venue.Capacity / totalCapacity * students.Count);
            int limit = Math.Min(GetVenueLimit(venue), proportionalShare);

            for (int i = 0; i < limit && studentIndex < students.Count; i++)
            {
                var s = students[studentIndex++];
                allocation[venue.Name].Add(CreatePreviewStudent(s, venue.Name));
            }
        }
        break;

    case "Random":
        var random = new Random();
        var shuffledStudents = students.OrderBy(s => random.Next()).ToList();
        studentIndex = 0;
        while (studentIndex < shuffledStudents.Count)
        {
            var availableVenues = selectedVenues
                .Where(v => allocation[v.Name].Count < GetVenueLimit(v))
                .ToList();
            if (!availableVenues.Any()) break;

            var venue = availableVenues[random.Next(availableVenues.Count)];
            var s = shuffledStudents[studentIndex++];
            allocation[venue.Name].Add(CreatePreviewStudent(s, venue.Name));
        }
        break;

    default:
        throw new ArgumentException("Unknown allocation type: " + allocationType);
}

// Handle unallocated
var unallocated = students.Skip(studentIndex).ToList();
if (unallocated.Any())
{
    allocation["Unallocated (no space/group limit)"] = unallocated
        .Select(s => CreatePreviewStudent(s, "Unallocated"))
        .ToList();
}

return allocation.Where(kvp => kvp.Value.Any())
```

Figure 5: Allocation Algorithm

This section of the implementation defines how students are automatically distributed across venues based on the allocation strategy selected by the lecturer. Each case implements a distinct allocation algorithm to ensure fair or efficient use of space, depending on instructional needs.

1. First Come, First Serve

Students are assigned sequentially to available venues until each venue reaches its capacity. This mimics a natural queueing system where early students get priority for limited spaces.

2. Balanced Allocation

Students are divided evenly across all available venues to ensure consistent class sizes. The algorithm calculates a per-venue quota based on total students and venues, ensuring a fair spread. For example, 3 different venues will have the same number of students regardless of their venue capacity, provided the student number does not exceed the venue capacity.

3. Round Robin

Students are distributed cyclically between venues which means the one student is allocated to a venue with each cycle. This ensures rotational fairness and is effective when all venues are similar in capacity and facilities.

4. Venue Capacity-Based Allocation

Students are distributed proportionally to the capacity of each venue. For example, a 100-seat lecture hall will receive twice as many students as a 50-seat lab, ensuring optimal utilization of available space.

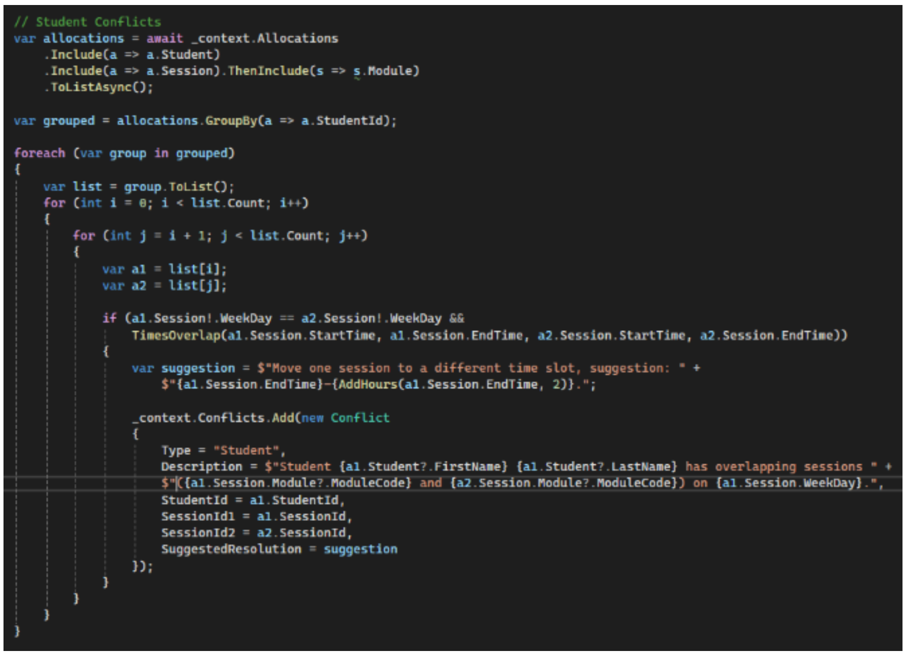
5. Random Allocation

A random distribution is used when an unbiased or unpredictable grouping is desired. The student list is shuffled and each student is assigned randomly to a venue with available space, preventing clustering or bias.

After the main allocation loop, any remaining unallocated students are captured under "Unallocated (no space/group limit)", ensuring that no student record is lost in the process.

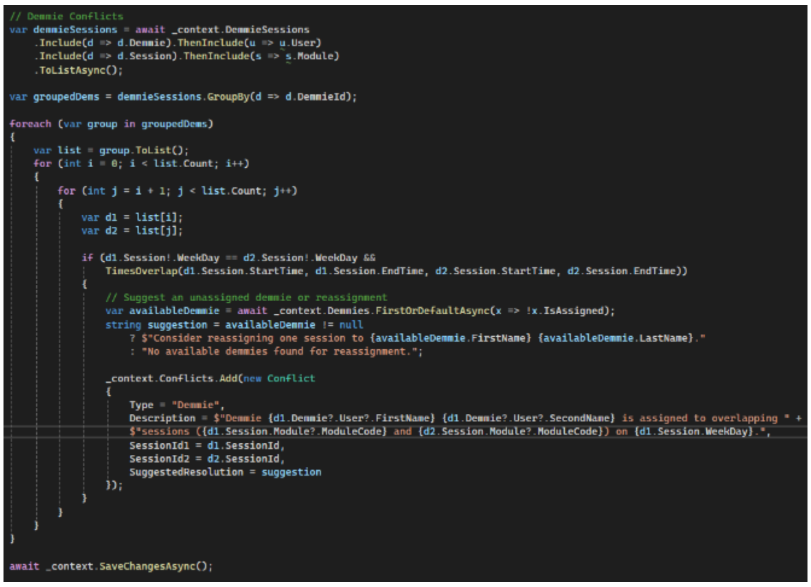
This switch-based approach centralizes multiple allocation algorithms within a single, extensible structure. Each algorithm adheres to the same input and output pattern (Dictionaries), operates with clear separation of logic and enables easy addition of new allocation strategies.

5.2.2 Conflict Management



```csharp
// Student Conflicts
var allocations = await _context.Allocations
    .Include(a => a.Student)
    .Include(a => a.Session).ThenInclude(s => s.Module)
    .ToListAsync();

var grouped = allocations.GroupBy(a => a.StudentId);

foreach (var group in grouped)
{
    var list = group.ToList();
    for (int i = 0; i < list.Count; i++)
    {
        for (int j = i + 1; j < list.Count; j++)
        {
            var a1 = list[i];
            var a2 = list[j];

            if (a1.Session!.WeekDay == a2.Session!.WeekDay &&
                TimesOverlap(a1.Session.StartTime, a1.Session.EndTime, a2.Session.StartTime, a2.Session.EndTime))
            {
                var suggestion = $"Move one session to a different time slot, suggestion: " +
                    $"{a1.Session.EndTime}-{AddHours(a1.Session.EndTime, 2)}.";

                _context.Conflicts.Add(new Conflict
                {
                    Type = "Student",
                    Description = $"Student {a1.Student?.FirstName} {a1.Student?.LastName} has overlapping sessions " +
                    $"({a1.Session.Module?.ModuleCode} and {a2.Session.Module?.ModuleCode}) on {a1.Session.WeekDay}.",
                    StudentId = a1.StudentId,
                    SessionId1 = a1.SessionId,
                    SessionId2 = a2.SessionId,
                    SuggestedResolution = suggestion
                });
            }
        }
    }
}
```

Figure 6: Conflict Management Algorithm



```csharp
// Demmie Conflicts
var demmieSessions = await _context.DemmieSessions
    .Include(d => d.Demmie).ThenInclude(u => u.User)
    .Include(d => d.Session).ThenInclude(s => s.Module)
    .ToListAsync();

var groupedDems = demmieSessions.GroupBy(d => d.DemmieId);

foreach (var group in groupedDems)
{
    var list = group.ToList();
    for (int i = 0; i < list.Count; i++)
    {
        for (int j = i + 1; j < list.Count; j++)
        {
            var d1 = list[i];
            var d2 = list[j];

            if (d1.Session!.WeekDay == d2.Session!.WeekDay &&
                TimesOverlap(d1.Session.StartTime, d1.Session.EndTime, d2.Session.StartTime, d2.Session.EndTime))
            {
                // Suggest an unassigned demmie or reassignment
                var availableDemmie = await _context.Demmies.FirstOrDefaultAsync(x => !x.IsAssigned);
                string suggestion = availableDemmie != null
                    ? $"Consider reassigning one session to {availableDemmie.FirstName} {availableDemmie.LastName}."
                    : "No available demmies found for reassignment.";

                _context.Conflicts.Add(new Conflict
                {
                    Type = "Demmie",
                    Description = $"Demmie {d1.Demmie?.User?.FirstName} {d1.Demmie?.User?.SecondName} is assigned to overlapping " +
                    $"sessions ({d1.Session.Module?.ModuleCode} and {d2.Session.Module?.ModuleCode}) on {d1.Session.WeekDay}.",
                    SessionId1 = d1.SessionId,
                    SessionId2 = d2.SessionId,
                    SuggestedResolution = suggestion
                });
            }
        }
    }
}

await _context.SaveChangesAsync();
```

Figure 7: Conflict Management Algorithm

This method is responsible for identifying and logging scheduling conflicts across sessions, students and demmies. This functionality ensures timetable integrity by automatically flagging overlapping sessions or double allocations. The system detects three main conflict types, venue, student and demmie conflicts, each handled by a distinct logic block.

The method first clears unresolved conflicts from the database to avoid duplication. It then analyzes current session and allocation data to detect potential overlaps. Once a conflict is identified, a new Conflict record is created in the database, complete with a descriptive message and a suggested resolution. This automated detection mechanism supports lecturers and administrators by preventing timetable clashes.

1. Venue Conflicts

The venue conflict section compares every pair of sessions to check if they share the same venue, the same day and if they have overlapping time slots. If all three conditions are met, it means two sessions are double-booked in the same room.

The system then searches for alternative venues with similar or greater capacity and suggests up to three replacement options using a descriptive message (e.g., “Consider moving one session to: Lab 3, Lab 4, Lab 5”). It then proceeds to log the conflict in the Conflicts table with type "Venue". This ensures that double-bookings are immediately identifiable and correctable.

2. Student Conflicts

Next, the method examines student allocations. Students are grouped by StudentId, and each student’s assigned sessions are checked for same-day time overlaps. If two of a student’s sessions conflict, a new conflict is created of type "Student", describing the overlap (e.g., “Student Mamello Kgabi has overlapping sessions CSIS1614 and BCIS2614 on Monday”).

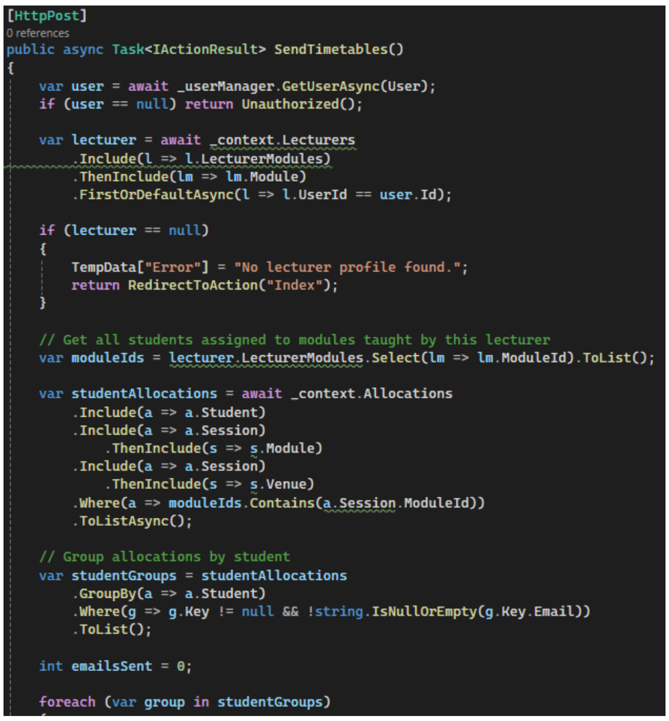
A time adjustment suggestion is generated dynamically, proposing a new non-overlapping timeslot (e.g., “Move one session to a different time slot, suggestion: 10:00–12:00.”). This proactive detection helps lecturers identify timetable clashes affecting student attendance.

3. Demmie Conflicts

Finally, the system checks demmie assignments, ensuring no Demie is assigned to two overlapping sessions. Similar to student detection, Sessions assigned to each demmie are grouped and compared by day and time. When an overlap is found, the system searches for an available demmie who can be reassigned. If one is available, the system suggests the replacement, otherwise it advises manual resolution. Each case is recorded in the database as a “Demmie” conflict with a recommendation (e.g., “Consider reassigning one session to Karabo Monaisa.”).

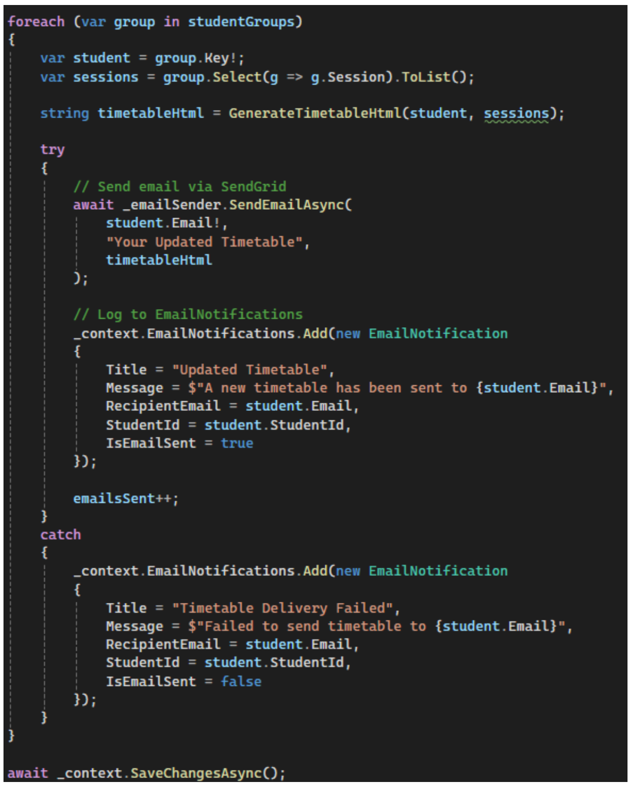
The DetectConflictsAsync() method automates a critical quality assurance step in the scheduling workflow. By programmatically identifying and logging overlapping sessions, student timetable clashes and demmie assignment conflicts, the system minimizes human error and enhances scheduling reliability. This approach demonstrates a practical use of algorithmic validation, database automation and context-aware recommendations within a real-world academic scheduling system.

5.2.3 Schedule Generation



```
[HttpPost]
0 references
public async Task<IActionResult> SendTimetables()
{
    var user = await _userManager.GetUserAsync(User);
    if (user == null) return Unauthorized();

    var lecturer = await _context.Lecturers
        .Include(l => l.LecturerModules)
        .ThenInclude(lm => lm.Module)
        .FirstOrDefaultAsync(l => l.UserId == user.Id);

    if (lecturer == null)
    {
        TempData["Error"] = "No lecturer profile found.";
        return RedirectToAction("Index");
    }

    // Get all students assigned to modules taught by this lecturer
    var moduleIds = lecturer.LecturerModules.Select(lm => lm.ModuleId).ToList();

    var studentAllocations = await _context.Allocations
        .Include(a => a.Student)
        .Include(a => a.Session)
            .ThenInclude(s => s.Module)
        .Include(a => a.Session)
            .ThenInclude(s => s.Venue)
        .Where(a => moduleIds.Contains(a.Session.ModuleId))
        .ToListAsync();

    // Group allocations by student
    var studentGroups = studentAllocations
        .GroupBy(a => a.Student)
        .Where(g => g.Key != null && !string.IsNullOrEmpty(g.Key.Email))
        .ToList();

    int emailsSent = 0;

    foreach (var group in studentGroups)
```

Figure 8: Schedule Generation



```
foreach (var group in studentGroups)
{
    var student = group.Key!;
    var sessions = group.Select(g => g.Session).ToList();

    string timetableHtml = GenerateTimetableHtml(student, sessions);

    try
    {
        // Send email via SendGrid
        await _emailSender.SendEmailAsync(
            student.Email!,
            "Your Updated Timetable",
            timetableHtml
        );

        // Log to EmailNotifications
        _context.EmailNotifications.Add(new EmailNotification
        {
            Title = "Updated Timetable",
            Message = $"A new timetable has been sent to {student.Email}",
            RecipientEmail = student.Email,
            StudentId = student.StudentId,
            IsEmailSent = true
        });

        emailsSent++;
    }
    catch
    {
        _context.EmailNotifications.Add(new EmailNotification
        {
            Title = "Timetable Delivery Failed",
            Message = $"Failed to send timetable to {student.Email}",
            RecipientEmail = student.Email,
            StudentId = student.StudentId,
            IsEmailSent = false
        });
    }
}

await _context.SaveChangesAsync();
```

Figure 9: Schedule Generation Algorithm

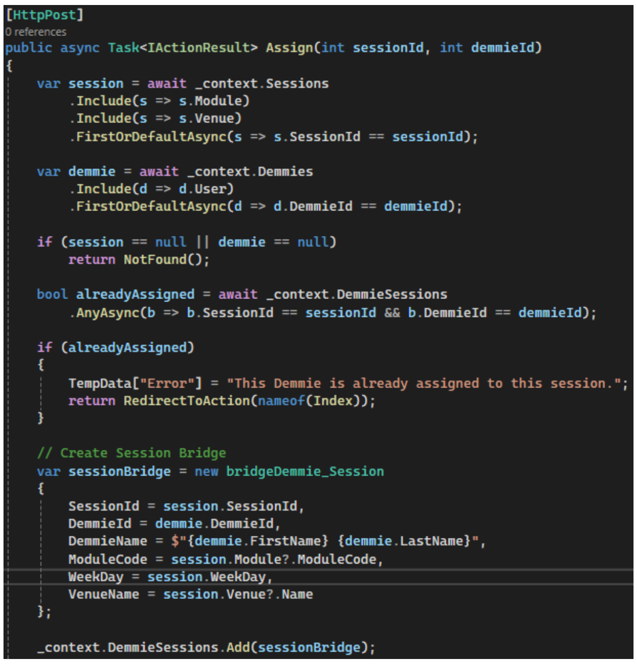
This method automates the distribution of personalized timetables to students via email. This functionality is a key feature of the Personal Scheduling Assistant, ensuring that each student receives only their allocated sessions in an organized, professional format. The implementation combines role-based access, data aggregation and HTML email generation to streamline the communication of finalised timetables. The purpose of this feature is to eliminate manual communication, reduce administrative effort and ensure students receive accurate, up-to-date schedules automatically.

When a lecturer initiates the send operation, the Post method first retrieves the currently authenticated user, it then matches this user to a corresponding lecturer record in the database. This step ensures that only users with the Lecturer role can send timetable emails and that the operation is scoped to the modules they are responsible for. Once the lecturer is identified, the method retrieves all student allocations associated with that lecturer's modules.

Students are grouped by their StudentId to collate all their allocated sessions into one timetable. Each group is passed to the helper method GenerateTimetableHtml(), which dynamically builds an HTML table representing the student's timetable. The HTML format ensures the email is both visually clear and accessible across devices. The system then iterates through all grouped students, sending each one an individual email through the configured SendGrid email service. Each attempt is logged in the EmailNotifications table to maintain traceability.

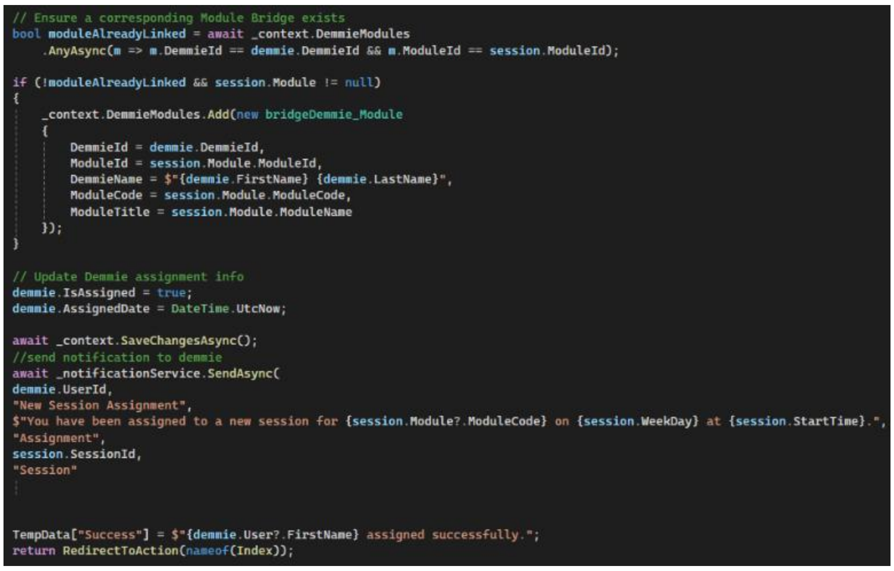
The timetable email dispatch feature automates one of the most time-consuming administrative tasks in academic scheduling. It personalizes communication between lecturers and students, ensures accuracy by sourcing directly from allocation data and logs all email activity for accountability. This implementation demonstrates the integration of data-driven logic together with automated communication, collectively enhancing usability, efficiency, and transparency within the scheduling system.

5.2.4 Demmie Distribution



```
[HttpPost]
0 references
public async Task<IActionResult> Assign(int sessionId, int demmieId)
{
    var session = await _context.Sessions
        .Include(s => s.Module)
        .Include(s => s.Venue)
        .FirstOrDefaultAsync(s => s.SessionId == sessionId);

    var demmie = await _context.Demmies
        .Include(d => d.User)
        .FirstOrDefaultAsync(d => d.DemmieId == demmieId);

    if (session == null || demmie == null)
        return NotFound();

    bool alreadyAssigned = await _context.DemmieSessions
        .AnyAsync(b => b.SessionId == sessionId && b.DemmieId == demmieId);

    if (alreadyAssigned)
    {
        TempData["Error"] = "This Demmie is already assigned to this session.";
        return RedirectToAction(nameof(Index));
    }

    // Create Session Bridge
    var sessionBridge = new bridgeDemmie_Session
    {
        SessionId = session.SessionId,
        DemmieId = demmie.DemmieId,
        DemmieName = $"{demmie.FirstName} {demmie.LastName}",
        ModuleCode = session.Module?.ModuleCode,
        WeekDay = session.WeekDay,
        VenueName = session.Venue?.Name
    };

    _context.DemmieSessions.Add(sessionBridge);
```

Figure 10: Demmie Distribution Algorithm

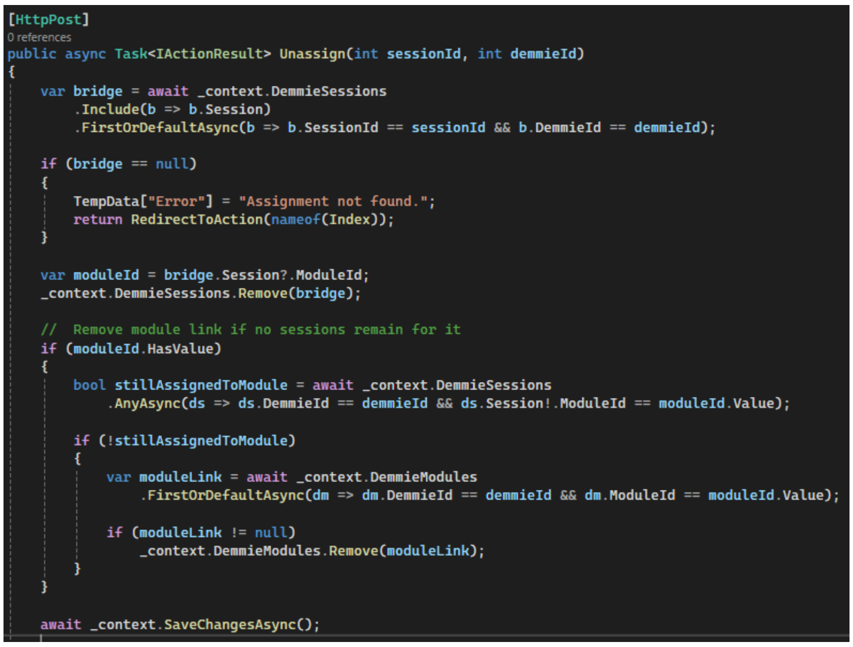


```
// Ensure a corresponding Module Bridge exists
bool moduleAlreadyLinked = await _context.DemmieModules
    .AnyAsync(m => m.DemmieId == demmie.DemmieId && m.ModuleId == session.ModuleId);

if (!moduleAlreadyLinked && session.Module != null)
{
    _context.DemmieModules.Add(new bridgeDemmie_Module
    {
        DemmieId = demmie.DemmieId,
        ModuleId = session.Module.ModuleId,
        DemmieName = $"{demmie.FirstName} {demmie.LastName}",
        ModuleCode = session.Module.ModuleCode,
        ModuleTitle = session.Module.ModuleName
    });
}

// Update Demmie assignment info
demmie.IsAssigned = true;
demmie.AssignedDate = DateTime.UtcNow;

await _context.SaveChangesAsync();
//send notification to demmie
await _notificationService.SendAsync(
demmie.UserId,
"New Session Assignment",
$"You have been assigned to a new session for {session.Module?.ModuleCode} on {session.WeekDay} at {session.StartTime}.",
"Assignment",
session.SessionId,
"Session"



TempData["Success"] = $"{demmie.User?.FirstName} assigned successfully.";
return RedirectToAction(nameof(Index));
```

Figure 11: Demmie Distribution Algorithm

The Assign and Unassign methods manage the allocation of Demmies to class sessions. This functionality automates what would traditionally be a manual administrative process, ensuring that demmies are linked to sessions and modules accurately, duplicate assignments are prevented, notification alerts are automatically sent upon assignment and data consistency is maintained across bridging tables that represent many-to-many relationships.

When a lecturer selects a Demmie and assigns them to a session, the system executes the Post Assign() action. The method begins by fetching the relevant Session and Demmie objects from the database, including their related entities such as Module, Venue, and associated User details. Before proceeding with the assignment, the system validates whether the selected Demmie has already been assigned to the same session to ensure that duplicate records are not created in the bridge table. If the assignment is valid, a bridge record is created between the Demmie and the Session using the bridgeDemmie_Session entity. After the bridge records are created, the Demmie's profile is updated to reflect their active assignment and the system then sends an automated notification using the centralized NotificationService.



```
[HttpPost]
0 references
public async Task<IActionResult> Unassign(int sessionId, int demmieId)
{
    var bridge = await _context.DemmieSessions
        .Include(b => b.Session)
        .FirstOrDefaultAsync(b => b.SessionId == sessionId && b.DemmieId == demmieId);

    if (bridge == null)
    {
        TempData["Error"] = "Assignment not found.";
        return RedirectToAction(nameof(Index));
    }

    var moduleId = bridge.Session?.ModuleId;
    _context.DemmieSessions.Remove(bridge);

    //  Remove module link if no sessions remain for it
    if (moduleId.HasValue)
    {
        bool stillAssignedToModule = await _context.DemmieSessions
            .AnyAsync(ds => ds.DemmieId == demmieId && ds.Session!.ModuleId == moduleId.Value);

        if (!stillAssignedToModule)
        {
            var moduleLink = await _context.DemmieModules
                .FirstOrDefaultAsync(dm => dm.DemmieId == demmieId && dm.ModuleId == moduleId.Value);

            if (moduleLink != null)
                _context.DemmieModules.Remove(moduleLink);
        }
    }

    await _context.SaveChangesAsync();
```

Figure 12: Unassign Demmie Algorithm

The Post Unassign method performs the reverse operation, removing the link between a Demmie and a session when a lecturer decides to reallocate resources.

The method retrieves the corresponding record from bridgeDemmie_Session and removes it. After removal, the system checks whether the Demmie is still linked to any other sessions under the same module. If not, the corresponding module bridge (bridgeDemmie_Module) is also removed to keep relationships consistent:

The overall purpose of this implementation is to maintain synchronization between session allocations, module participation and Demmie workload tracking while minimizing manual data entry errors. By automating these tasks, not only does the system minimize administrative effort, it also maintains data accuracy, transparency and user engagement.

5.3 Future Development

Although the Personal Scheduling Assistant for Classes and Practicals prototype successfully meets its current objectives, there are several opportunities for further enhancement to increase scalability, usability and institutional value. Future development could focus on the following areas:

1. Strict Hour Monitoring for Demmies

Although demmie hour tracking is currently implemented, future development should include automated hour validation and enforcement mechanisms. The system could integrate real-time tracking through session attendance logs. Lecturers would then have the authority to verify and approve the demmies working hours. Then automatic alerts would notify demmies and lecturers when hour thresholds are nearing or exceeded, ensuring fair workload distribution and compliance with institutional policies.

2. Integration with Institutional Systems

Currently, the system operates as a standalone scheduling assistant. Future versions could connect with university management systems (e.g., PeopleSoft, Oracle Campus Solutions, or Moodle API) to synchronize student enrollments, venue data, and timetable updates automatically, reducing manual data entry and improving accuracy

3. Advanced Demmie Management

Future development could introduce availability-aware scheduling, automatically matching demmies to sessions based on declared time slots, workload limits, and historical performance metrics. Additionally, integration with calendar APIs could synchronize demmie sessions with their personal schedules.

4. AI-Assisted Conflict Resolution

The current conflict-detection logic identifies and suggests resolutions for clashes using a fixed template, however future work could introduce AI-driven decision support. Using natural-language explanations and predictive analytics, the system could recommend optimal reallocation strategies for students, demmies and even venues in real time.

5. Cloud Deployment and Multi-Campus Scaling

Migrating the system to a cloud-based infrastructure like Microsoft Azure or AWS would allow for seamless scalability, data redundancy and centralized access across multiple campuses. This would position the system for university-wide or even inter-institutional deployment

5.4 Chapter Overview

This chapter discussed the source code and implementation aspects of the Personal Scheduling Assistant for Classes and Practicals system, demonstrating how theoretical design was translated into functional software components.

Key modules such as automatic allocation, conflict detection, personalized timetable generation and demmie distribution were explained in depth, illustrating the system's logical workflow and modular structure. The integration of automated email dispatch and data validation mechanisms further enhances the platform's operational efficiency and reliability.

Finally, the chapter proposed several future development paths to improve scalability, intelligence and integration with institutional infrastructures. These enhancements would not only expand the system’s capabilities but also future-proof it for widespread deployment across higher education institutions.

6. Chapter 6: Evaluation

6.1 Introduction

This chapter presents the evaluation of the Personal Scheduling Assistant for Classes and Practicals prototype. The purpose of this evaluation is to determine the system's effectiveness, usability and reliability in meeting its intended goals of improving scheduling efficiency, reducing manual allocation effort and enhancing communication between lecturers, students and demmies. The assessment focuses on both functional and user experience aspects, with feedback obtained from prospective users like lecturers and demmies.

This chapter outlines the evaluation process, describes how the system was tested by end-users, summarizes the results and discusses the improvements made following the evaluation. The goal was to ensure that the final system is practical, user-friendly and robust enough for real-world academic deployment.

6.2 Evaluation Approach

This section describes the evaluation population, sampling strategy, sample size, and data collection methods that guided the process.

6.2.1 Population

The target population for the evaluation consisted of University of the Free State academic staff consisting of lecturers and demmies. These users represent the system's intended end-users and are best positioned to provide meaningful insights into its practicality and effectiveness within the academic scheduling context.

6.2.2 Sampling Strategy

A non-probability, convenience sampling strategy was adopted for participant selection. Participants were chosen based on their accessibility, availability and willingness to engage in the study.

This approach was particularly appropriate due to the prototype nature of the system, the limited timeframe of the research and the need to gather rapid, cost-effective feedback from knowledgeable users.

6.2.3 Sample Size

The sample consisted of five (6) participants, representing both lecturer and demmie user roles and an additional admin role:

- Three (3) participants assumed the Lecturer role, testing features such as registration, student allocation and conflict management.
- One (2) participant assumed the Demmie role, testing availability management, session assignment and workload tracking.
- One (1) additional participant contributed feedback as an administrative user focusing on usability, user management and data accuracy.

While small, this sample size was sufficient for identifying key usability patterns and system-level improvements, as recommended by Nielsen's heuristic that five users can uncover approximately 80% of usability issues in a prototype (Nielsen, 1994).

6.2.4 Data Collection Methods

The evaluation combined quantitative and qualitative data collection methods to obtain a comprehensive understanding of the system's effectiveness.

(a) **Task Completion Rate**

Participants were instructed to complete a series of predefined tasks that reflected real academic scheduling operations such as registering, creating allocations and running conflict detection.

The task completion rate was measured as the percentage of successfully completed tasks without external assistance. This metric provided a direct indicator of the system’s usability and intuitiveness.

(b) **Questionnaires and Feedback Reports**

Users will be asked to fill a pre-test questionnaire and after completing the task, fill a post-test questionnaire. They will also have to complete a NPS questionnaire.

Overall system satisfaction and Net Promoter Score (NPS)

In addition, participants provided open-ended feedback regarding frustrations, improvement suggestions, and preferred enhancements. This data provided qualitative insights into user experience and emotional engagement with the interface.

6.2.5 Procedure Protocol

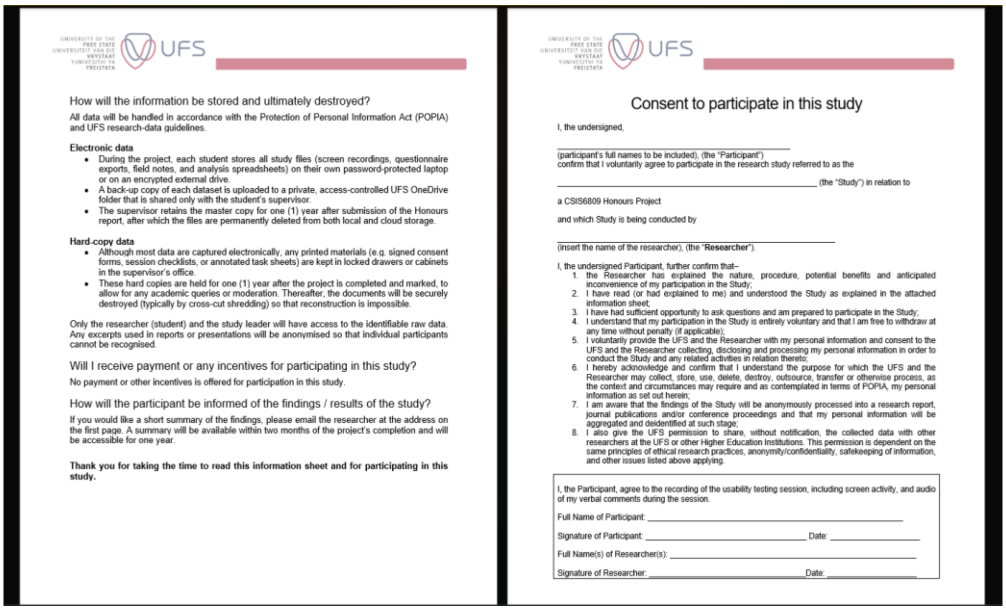
The evaluation process followed a structured sequence to ensure consistency and reliability across all participants. Each test session was conducted individually, lasting approximately 15–25 minutes per participant. The process was as follows:

1. Pre-Test Briefing

Participants were introduced to the objectives of the evaluation, purpose of the system, and confidentiality considerations. They were informed that their feedback would be used solely for research and system improvement purposes.

2. Consent Form

The participants were then handed a consent form to indicate their willingness to participate in the evaluation.



How will the information be stored and ultimately destroyed?

All data will be handled in accordance with the Protection of Personal Information Act (POPIA) and UFS research-data guidelines.

Electronic data

- During the project, each student stores all study files (screen recordings, questionnaire exports, field notes, and analysis spreadsheets) on their own password-protected laptop or on an encrypted external drive.
- A back-up copy of each dataset is uploaded to a private, access-controlled UFS OneDrive folder that is shared only with the student's supervisor.
- The supervisor retains the master copy for one (1) year after submission of the Honours report, after which the files are permanently deleted from both local and cloud storage.

Hard-copy data

- Although most data are captured electronically, any printed materials (e.g. signed consent forms, session checklists, or annotated task sheets) are kept in locked drawers or cabinets in the supervisor's office.
- These hard copies are held for one (1) year after the project is completed and marked, to allow for any academic queries or moderation. Thereafter, the documents will be securely destroyed (typically by cross-cut shredding) so that reconstruction is impossible.

Only the researcher (student) and the study leader will have access to the identifiable raw data. Any excerpts used in reports or presentations will be anonymised so that individual participants cannot be recognised.

Will I receive payment or any incentives for participating in this study?

No payment or other incentives is offered for participation in this study.

How will the participant be informed of the findings / results of the study?

If you would like a short summary of the findings, please email the researcher at the address on the first page. A summary will be available within two months of the project's completion and will be accessible for one year.

Thank you for taking the time to read this information sheet and for participating in this study.

Consent to participate in this study

I, the undersigned,

____

(participant's full names to be included), (the "Participant")
confirm that I voluntarily agree to participate in the research study referred to as the

______________________________ (the "Study") in relation to

a CSIS6809 Honours Project

and which Study is being conducted by

____

(insert the name of the researcher), (the "**Researcher**").

I, the undersigned Participant, further confirm that–

1. the Researcher has explained the nature, procedure, potential benefits and anticipated inconvenience of my participation in the Study;
2. I have read (or had explained to me) and understood the Study as explained in the attached information sheet;
3. I have had sufficient opportunity to ask questions and am prepared to participate in the Study;
4. I understand that my participation in the Study is entirely voluntary and that I am free to withdraw at any time without penalty (if applicable);
5. I voluntarily provide the UFS and the Researcher with my personal information and consent to the UFS and the Researcher collecting, disclosing and processing my personal information in order to conduct the Study and any related activities in relation thereto;
6. I hereby acknowledge and confirm that I understand the purpose for which the UFS and the Researcher may collect, store, use, delete, destroy, outsource, transfer or otherwise process, as the context and circumstances may require and as contemplated in terms of POPIA, my personal information as set out herein;
7. I am aware that the findings of the Study will be anonymously processed into a research report, journal publications and/or conference proceedings and that my personal information will be aggregated and deidentified at such stage;
8. I also give the UFS permission to share, without notification, the collected data with other researchers at the UFS or other Higher Education Institutions. This permission is dependent on the same principles of ethical research practices, anonymity/confidentiality, safekeeping of information, and other issues listed above applying.

I, the Participant, agree to the recording of the usability testing session, including screen activity, and audio of my verbal comments during the session.

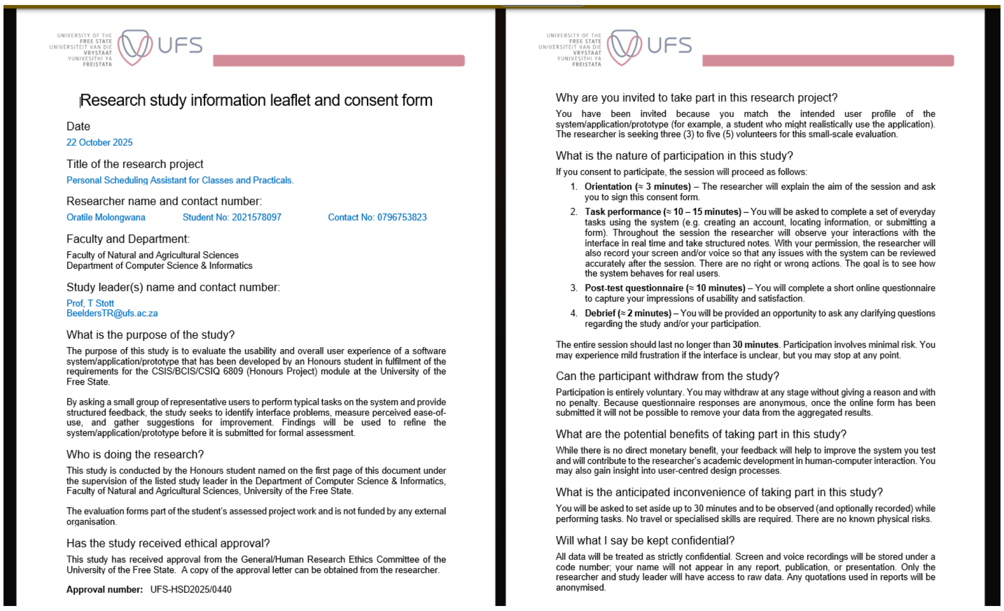
Full Name of Participant: ______________________________

Signature of Participant: ____________________ Date: __________

Full Name(s) of Researcher(s): ______________________________

Signature of Researcher: ____________________ Date: __________

Figure 13: Consent Form 1



Research study information leaflet and consent form

Date

22 October 2025

Title of the research project

Personal Scheduling Assistant for Classes and Practicals.

Researcher name and contact number:

Oratile Molongwana Student No: 2021578097 Contact No: 0796753823

Faculty and Department:

Faculty of Natural and Agricultural Sciences
Department of Computer Science & Informatics

Study leader(s) name and contact number:

Prof, T Stott
BeeldersTR@ufs.ac.za

What is the purpose of the study?

The purpose of this study is to evaluate the usability and overall user experience of a software system/application/prototype that has been developed by an Honours student in fulfilment of the requirements for the CSIS/BCIS/CSIQ 6809 (Honours Project) module at the University of the Free State.

By asking a small group of representative users to perform typical tasks on the system and provide structured feedback, the study seeks to identify interface problems, measure perceived ease-of-use, and gather suggestions for improvement. Findings will be used to refine the system/application/prototype before it is submitted for formal assessment.

Who is doing the research?

This study is conducted by the Honours student named on the first page of this document under the supervision of the listed study leader in the Department of Computer Science & Informatics, Faculty of Natural and Agricultural Sciences, University of the Free State.

The evaluation forms part of the student's assessed project work and is not funded by any external organisation.

Has the study received ethical approval?

This study has received approval from the General/Human Research Ethics Committee of the University of the Free State. A copy of the approval letter can be obtained from the researcher.

Approval number: UFS-HSD2025/0440

Why are you invited to take part in this research project?

You have been invited because you match the intended user profile of the system/application/prototype (for example, a student who might realistically use the application). The researcher is seeking three (3) to five (5) volunteers for this small-scale evaluation.

What is the nature of participation in this study?

If you consent to participate, the session will proceed as follows:

1. **Orientation (≈ 3 minutes)** – The researcher will explain the aim of the session and ask you to sign this consent form.
2. **Task performance (≈ 10 – 15 minutes)** – You will be asked to complete a set of everyday tasks using the system (e.g. creating an account, locating information, or submitting a form). Throughout the session the researcher will observe your interactions with the interface in real time and take structured notes. With your permission, the researcher will also record your screen and/or voice so that any issues with the system can be reviewed accurately after the session. There are no right or wrong actions. The goal is to see how the system behaves for real users.
3. **Post-test questionnaire (≈ 10 minutes)** – You will complete a short online questionnaire to capture your impressions of usability and satisfaction.
4. **Debrief (≈ 2 minutes)** – You will be provided an opportunity to ask any clarifying questions regarding the study and/or your participation.

The entire session should last no longer than **30 minutes**. Participation involves minimal risk. You may experience mild frustration if the interface is unclear, but you may stop at any point.

Can the participant withdraw from the study?

Participation is entirely voluntary. You may withdraw at any stage without giving a reason and with no penalty. Because questionnaire responses are anonymous, once the online form has been submitted it will not be possible to remove your data from the aggregated results.

What are the potential benefits of taking part in this study?

While there is no direct monetary benefit, your feedback will help to improve the system you test and will contribute to the researcher's academic development in human-computer interaction. You may also gain insight into user-centred design processes.

What is the anticipated inconvenience of taking part in this study?

You will be asked to set aside up to 30 minutes and to be observed (and optionally recorded) while performing tasks. No travel or specialised skills are required. There are no known physical risks.

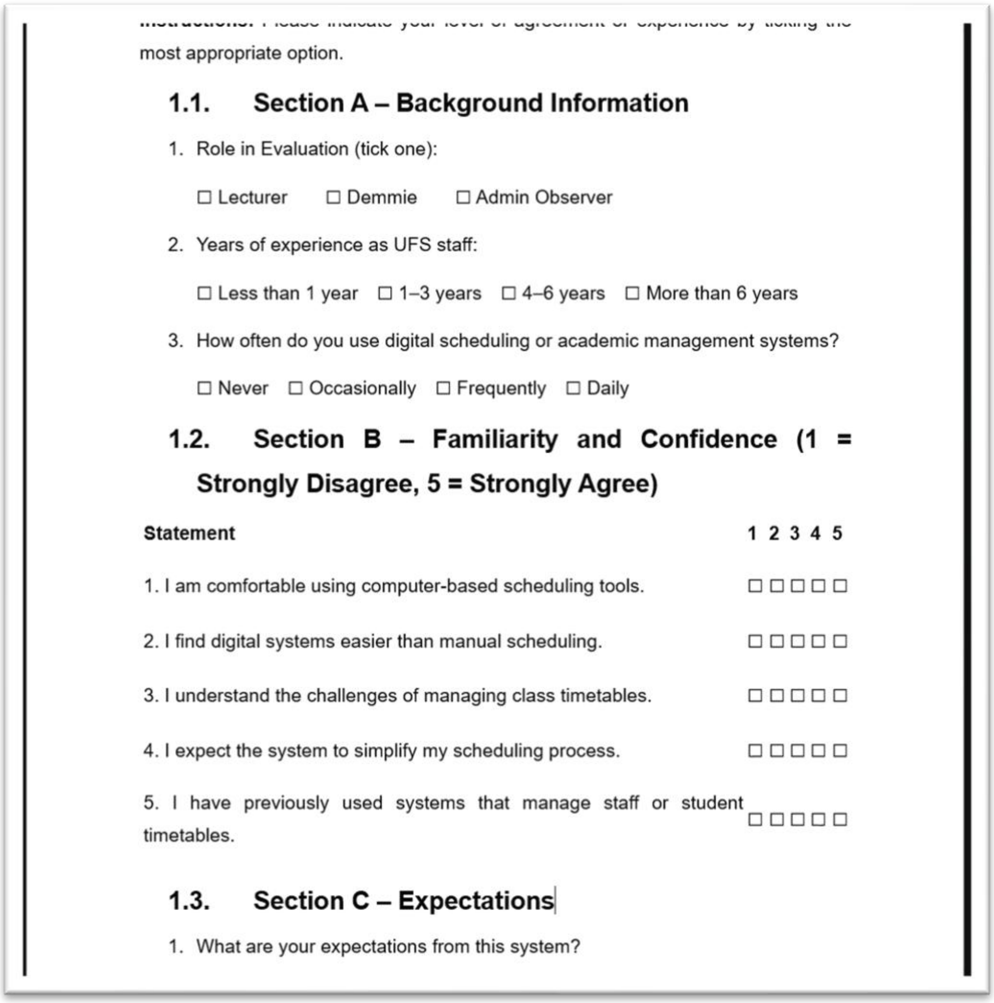
Will what I say be kept confidential?

All data will be treated as strictly confidential. Screen and voice recordings will be stored under a code number; your name will not appear in any report, publication, or presentation. Only the researcher and study leader will have access to raw data. Any quotations used in reports will be anonymised.

Figure 14: Consent Form 2

3. Pre-Test Questionnaire

Before interacting with the system, participants completed a short pre-test questionnaire to capture their prior experience with scheduling systems, digital tools and general comfort with technology.



Instructions: Please indicate your level of agreement or experience by ticking the most appropriate option.

1.1. Section A – Background Information

1. Role in Evaluation (tick one):

   ☐ Lecturer ☐ Demmie ☐ Admin Observer

2. Years of experience as UFS staff:

   ☐ Less than 1 year ☐ 1–3 years ☐ 4–6 years ☐ More than 6 years

3. How often do you use digital scheduling or academic management systems?

   ☐ Never ☐ Occasionally ☐ Frequently ☐ Daily

1.2. Section B – Familiarity and Confidence (1 = Strongly Disagree, 5 = Strongly Agree)

Statement	1 2 3 4 5
1. I am comfortable using computer-based scheduling tools.	☐ ☐ ☐ ☐ ☐
2. I find digital systems easier than manual scheduling.	☐ ☐ ☐ ☐ ☐
3. I understand the challenges of managing class timetables.	☐ ☐ ☐ ☐ ☐
4. I expect the system to simplify my scheduling process.	☐ ☐ ☐ ☐ ☐
5. I have previously used systems that manage staff or student timetables.	☐ ☐ ☐ ☐ ☐

1.3. Section C – Expectations

1. What are your expectations from this system?

4. Task Execution Phase

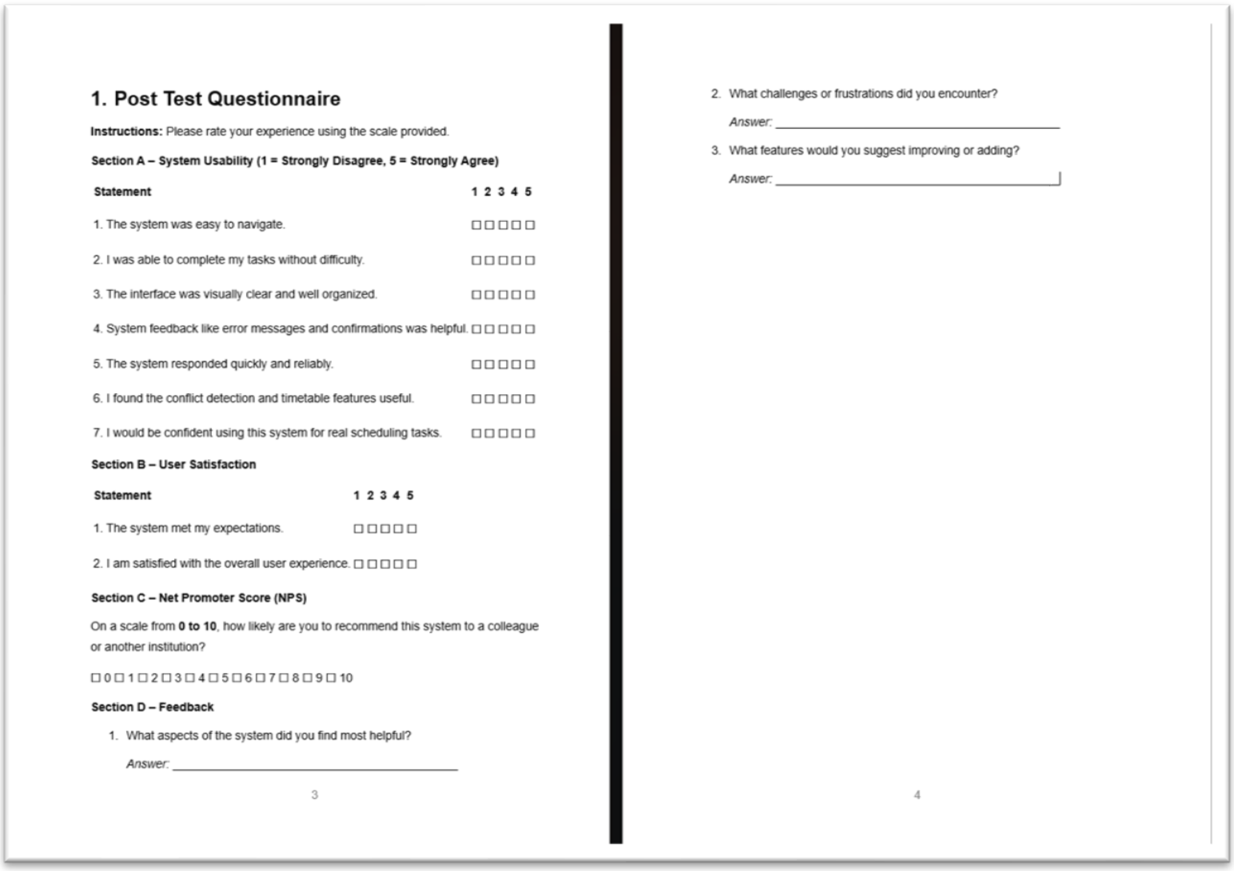
Participants were assigned a set of core tasks depending on their designated role (Lecturer, Demmie or Admin).

- Lecturers were asked to register, allocate students to venues, detect scheduling conflicts and distribute available demmies.
- Demmies tested availability management, session assignment and workload tracking.
- The Admin evaluated usability, user management and data accuracy.

During this phase, I recorded task completion times, errors encountered and assistance required.

5. Post-Test Questionnaire

After completing the tasks, participants filled out a post-test questionnaire assessing system usability, task difficulty, navigation clarity and satisfaction. This included Likert-scale ratings and the Net Promoter Score questionnaire.



1. Post Test Questionnaire

Instructions: Please rate your experience using the scale provided.

Section A – System Usability (1 = Strongly Disagree, 5 = Strongly Agree)

Statement	**1 2 3 4 5**
1. The system was easy to navigate.	☐☐☐☐☐
2. I was able to complete my tasks without difficulty.	☐☐☐☐☐
3. The interface was visually clear and well organized.	☐☐☐☐☐
4. System feedback like error messages and confirmations was helpful.	☐☐☐☐☐
5. The system responded quickly and reliably.	☐☐☐☐☐
6. I found the conflict detection and timetable features useful.	☐☐☐☐☐
7. I would be confident using this system for real scheduling tasks.	☐☐☐☐☐

Section B – User Satisfaction

Statement	**1 2 3 4 5**
1. The system met my expectations.	☐☐☐☐☐
2. I am satisfied with the overall user experience.	☐☐☐☐☐

Section C – Net Promoter Score (NPS)

On a scale from **0 to 10**, how likely are you to recommend this system to a colleague or another institution?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10

Section D – Feedback

1. What aspects of the system did you find most helpful?

   Answer: ____________________

3

2. What challenges or frustrations did you encounter?

   Answer: ____________________

3. What features would you suggest improving or adding?

   Answer: ____________________

4

6. Feedback and Debriefing

Participants provided open-ended feedback on their overall impressions, identifying strengths, weaknesses and potential improvements. Suggestions were discussed collectively in a short debriefing session.

This evaluation protocol ensured that quantitative performance data (e.g., task completion rates) and qualitative feedback (e.g., user satisfaction) were both captured systematically.

6.3 Evaluation Results

6.3.1 Lecture

User Role	Number of Tasks	Tasks Completed Successfully	Completion Rate	Average Time per Task
Lecturer 1	4	4	100%	15 minutes 24 seconds
Lecturer 2	4	3	75%	16 minutes 54 seconds
Lecturer 3	4	4	100%	16 minutes 36 seconds

Table 2: Lecturer Evaluation

Average Completion Rate: 91.7%

Average Time per Task: 16 minutes 18 seconds

6.3.2 Demmie

User Role	Number of Tasks	Tasks Completed Successfully	Completion Rate	Average Time per Task
Demmie 1	3	3	100%	9 minutes 6 seconds
Demmie 2	3	2	67%	10 minutes 46 seconds.

Table 3: Demmie Evaluation

Average Completion Rate: 83.5%

Average Time per Task: 9 minutes 56 seconds

6.3.3 Admin

User Role	Number of Tasks	Tasks Completed Successfully	Completion Rate	Average Time per Task
Admin	3	3	100%	12 minutes 53 seconds

Table 4: Admin Evaluation

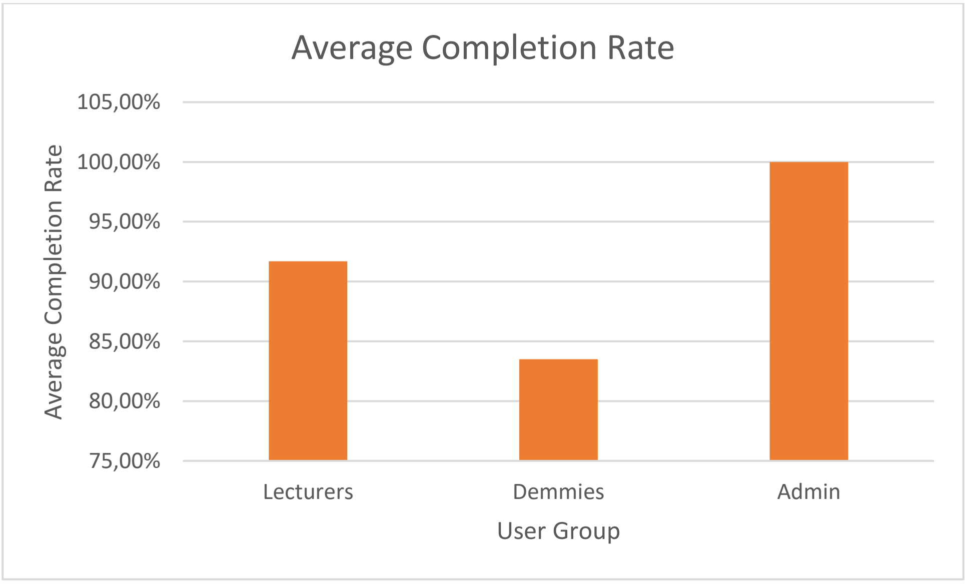
Average Completion Rate: 100%

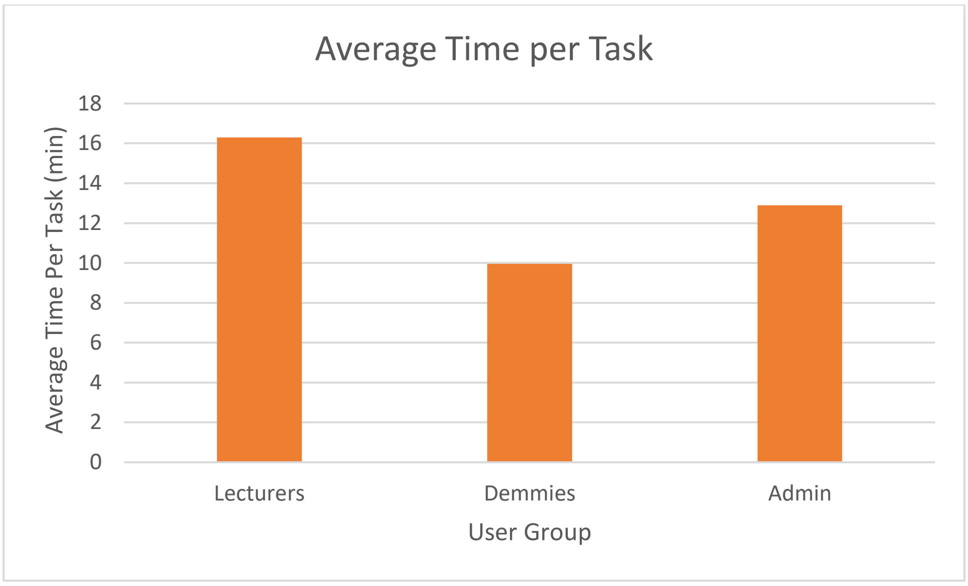
Average Time per Task: 12 minutes 53 seconds

6.3.4 Summary

User Group	Average Completion Rate	Average Time per Task
Lecturers	91.7%	16 minutes 18 seconds
Demmies	83.5%	9 minutes 56 seconds
Admin	100%	12 minutes 53 seconds
Overall Average	**91.7%**	**13 minutes 42 seconds**

Table 5: Evaluation Summary





6.3.5 Net Promoter Score

Category	Count	Percentage
Promoters (9–10)	3	50%
Passives (7–8)	2	33%
Detractors (0–6)	1	17%

Table 6: Net Promoter Score

NPS = Promoters% - Detractors%

NPS = 50% - 17%

NPS = 33%

6.4 Interpretation

The overall task completion rate of 91.7% indicates that the system is both functional and user-friendly across all roles.

Lecturers achieved near-perfect completion rates suggesting that the registration, allocation, conflict detection and demmie distribution functionalities are intuitive and stable. The slightly longer task duration of approximately 16 minutes reflects the greater complexity of lecturer tasks involving multi-step operations such as configuring sessions and managing allocations.

Demmies demonstrated strong performance, completing an average of 83.5% of their tasks successfully. The main challenges observed involved navigating to the availability management interface and identifying assigned sessions, indicating areas where UI guidance and labelling could be improved.

The Admin role achieved a perfect success rate with moderate task times, showing that administrative features such as user monitoring and report generation are streamlined and reliable.

Overall, these results reflect high usability and functional accuracy, with most users completing their tasks successfully without assistance. Minor issues related to navigation and icon clarity were identified and later corrected by improving button labels, adding tooltips and refining interface organization.

6.5 Improvements Implemented

Based on the feedback and task observations, the following adjustments were made:

1. **Enhanced Button Labels and Tooltips** – Icons were supplemented with text descriptions to improve discoverability and usability.
2. **Notification Filtering Feature** – Added filters to distinguish between “Read” and “Unread” notifications after users noted clutter. A button to indicate whether you want to mark a notification as “Read” was added.
3. **Send Timetables Confirmation** – Implemented a preview confirmation modal before sending timetables, ensuring lecturers do not mistakenly send mass emails to students.

4. **Refined Error Messages** – Improved exception handling messages for better clarity.

7. References

Alouffi, B., & Hasnain, M. (2021). IEEE Access. *A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies*.

Blackboard. (2025). Retrieved April 26, 2025, from Blackboard: https://help.blackboard.com/Blackboard_App/Feature_Guide

Broke, J. (1996). *A quick and dirty usabilty scale.* Usability Evaluation in Industry.

Brown, B., & Sugarman, J. (2020, February). HPTN Ethics Guidance for Research. Retrieved from https://www.hptn.org/sites/default/files/inline-files/HPTNEthicsGuidanceDocument_2.26.20.pdf

Cowling, N. (2024, Nov 4). *statista*. Retrieved from statista: https://www.statista.com/statistics/1261626/south-africa-gross-tertiary-school-enrollment-ratio/#:~:text=The%20gross%20tertiary%20enrollment%20ratio,Index%20(GPI)%20in%20youth%20literacy

Cowling, N. (2024, Nov 4). *statista*. Retrieved March 3, 2025, from statista: https://www.statista.com/statistics/1261626/south-africa-gross-tertiary-school-enrollment-ratio/#:~:text=The%20gross%20tertiary%20enrollment%20ratio,Index%20(GPI)%20in%20youth%20literacy

Dang, T. (2024, 01 20). *Why C# for Web Development is A Great Choice in The Modern Era*. Retrieved 08 21, 2025, from orientsoftware: https://www.orientsoftware.com/blog/csharp-for-web-development/

EduPage. (2025). Retrieved April 26, 2025, from EduPage: https://www.edupage.org/

Godel, J. (2025, 04 24). *Advanced database programming with c-sharp 14 and microsoft sql server*. Retrieved 08 21, 2025, from c-sharpcorner: https://www.c-sharpcorner.com/article/advanced-database-programming-with-c-sharp-14-and-microsoft-sql-server/

Nielsen, J. (1994). *Usability Engineering.* Morgan Kaufmann Publishers.

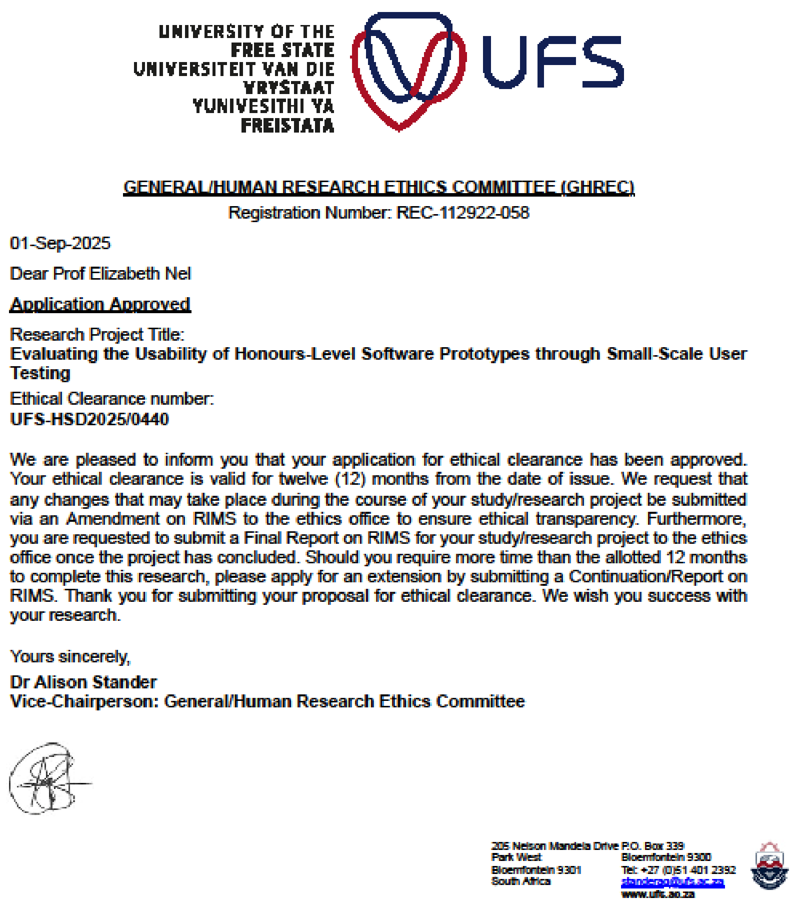
Timetable Master. (2025). Retrieved April 26, 2025, from Timetable Master: https://www.timetablemaster.com/

Vidyalaya. (2023). Retrieved April 26, 2025, from Vidyalaya: https://vidyalayaold.sapphiresolutions.net/features/school-timetable-management-system

WebUntis. (2025). Retrieved April 26, 2026, from WebUntis: https://www.untis.at/en/products/webuntis

Wickramasinghe, S. (2024, 08 22). *What Makes Visual Studio Code So Popular?* Retrieved 08 21, 2025, from medium: https://medium.com/adl-blog/what-makes-visual-studio-code-so-popular-54206c386503

Appendix A – Ethical Clearance



UNIVERSITY OF THE FREE STATE
UNIVERSITEIT VAN DIE VRYSTAAT
YUNIVESITHI YA FREISTATA
UFS

GENERAL/HUMAN RESEARCH ETHICS COMMITTEE (GHREC)

Registration Number: REC-112922-058

01-Sep-2025

Dear Prof Elizabeth Nel

Application Approved

Research Project Title:
Evaluating the Usability of Honours-Level Software Prototypes through Small-Scale User Testing

Ethical Clearance number:
UFS-HSD2025/0440

We are pleased to inform you that your application for ethical clearance has been approved. Your ethical clearance is valid for twelve (12) months from the date of issue. We request that any changes that may take place during the course of your study/research project be submitted via an Amendment on RIMS to the ethics office to ensure ethical transparency. Furthermore, you are requested to submit a Final Report on RIMS for your study/research project to the ethics office once the project has concluded. Should you require more time than the allotted 12 months to complete this research, please apply for an extension by submitting a Continuation/Report on RIMS. Thank you for submitting your proposal for ethical clearance. We wish you success with your research.

Yours sincerely,

Dr Alison Stander
Vice-Chairperson: General/Human Research Ethics Committee

[illegible]

205 Nelson Mandela Drive
Park West
Bloemfontein 9301
South Africa

P.O. Box 339
Bloemfontein 9300
Tel: +27 (0)51 401 2392
standerao@ufs.ac.za
www.ufs.ac.za