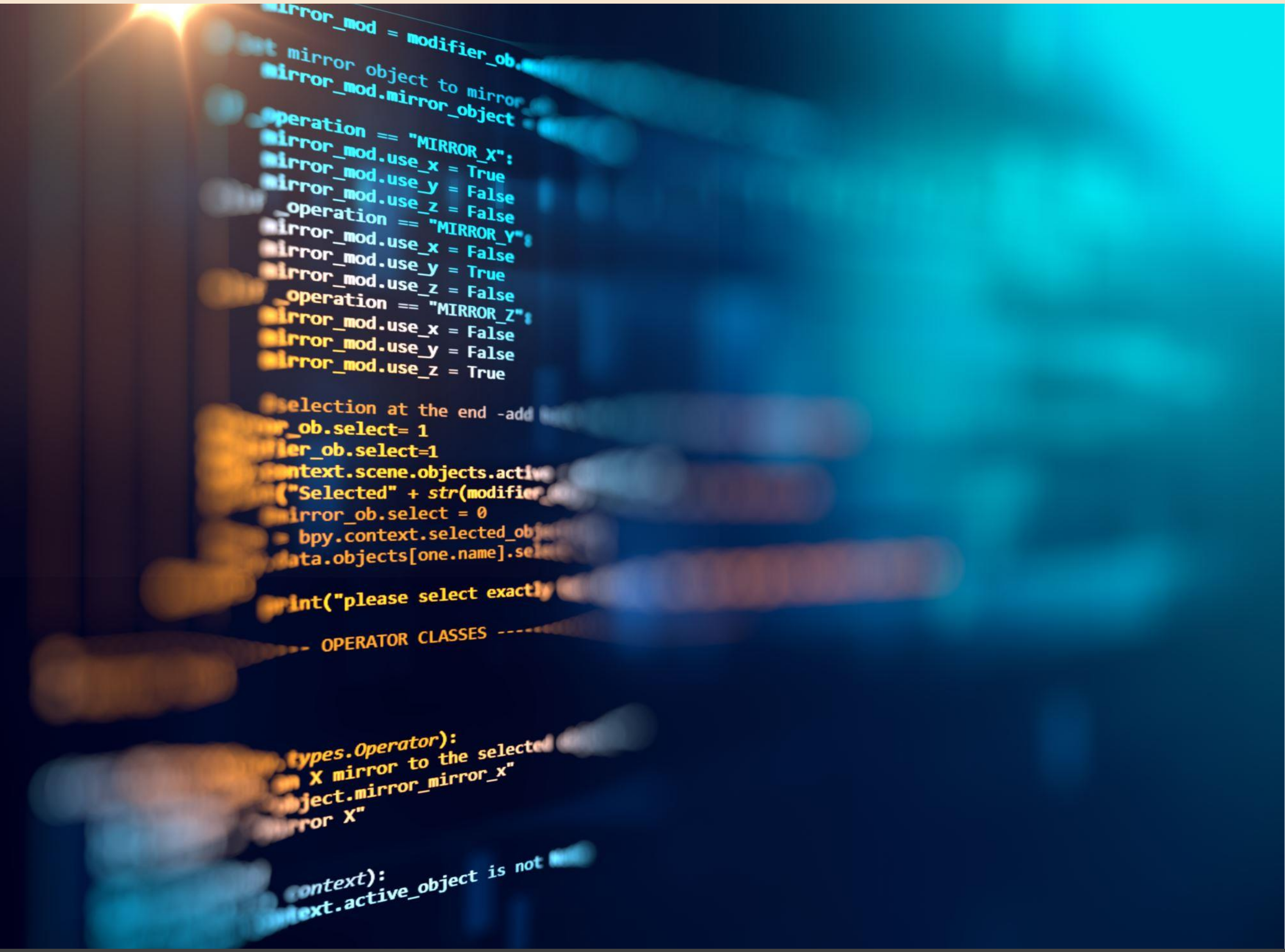


# System Maintenance Document: Latrobe Scheduling System

CSE3PAX – Assessment 4

2023| Prepared by: Alexx, Chareen, Laura & Michael

## Copyright Information

## Warranties & Contractual Obligations

**Software Disclaimer**

**IMPORTANT: READ CAREFULLY**

The Group 1 Students (**"**the **Developers**"), having developed the Latrobe Scheduling System Software ("the **Software**") under Latrobe University ("the **University**") Bachelor of Applied Information Technology program warrant the following:

**EDUCATION PURPOSE DISCLAIMER:**

1. The parties acknowledge the primary purpose of the Software development project is educational. The project is conducted as part of the Latrobe's Bachelor of Applied Information Technology program to enhance the learning experience of the students. Any rights, licenses, or obligations established under this agreement are in line with the educational context and goals of the program.
2. The Software' warranty is limited to its use as a tool to assess the Developer's learning. The Developer's do not warrant its use for any other purpose (express or implied), including, but not limited to, merchantability, fitness for a particular purpose, non-infringement, and negligence.

**USE AT YOUR OWN RISK:**

3. The use of this Software is at your own risk. The Developers make no guarantees or representations regarding the accuracy, reliability, or effectiveness of the Software. You are solely responsible for any consequences, or effectiveness resulting from the use of this Software.

**NO GUARANTEE OF RESULTS:**

4. The Software may not achieve results of any purpose other than that warranted, namely, to assess the developer's learning. The Developers do not guarantee the suitability of the Software for any other purpose whatsoever.

**NO LIABILITY:**

5. In no event that the Developers are adjudged liable to you or any third party for any Software omission or error (including those caused by negligence) any damages including but not limited to, direct, indirect, special, incidental, or consequential damages, arising out of the use or inability to use the Software are limited to the cost of resupplying the software.

**NO SUPPORT:**

6. The Developers do not provide technical support, maintenance, or updates for this Software. You are responsible for any modifications, enhancements or troubleshooting required.

**DATA PRIVACY AND SUPPORT:**

7. The Software may collect and process data in accordance with its intended functionality. It is your responsibility to ensure the privacy and security of any data you input or handle using the Software.

**COMPLIANCE WITH LAWS:**

8. You agree to use the Software in compliance with all applicable laws, regulations, and policies, including but not limited to data protection and intellectual property laws.

**MODIFICATIONS AND ALTERATIONS:**

9. You are not authorised to modify, reverse engineer, decompile, disassemble, or create derivative works based on the Software without the express written consent of the Developers, and may be held liable for breach of copyright and any damages, including direct, indirect, special, incidental and / or consequential damages should you do so.

**ENTIRE AGREEMENT:**

10. This Software Disclaimer constitutes the entire agreement between you and the Developers regarding the use of the Software and supersedes all prior and contemporaneous understanding, agreements, representations, and warranties.

By using this Software, you acknowledge that you have read and understood this disclaimer and agree to be bound by its terms and conditions.

## Table of changes

| Published Date | Status | Document Version | Page(s) Affected | Description of Revision | Author(s) |
|---|---|---|---|---|---|
| 21/01/2024 | Published | 1.0 | All Pages | Initial Release | Alexx Verdichizzi / Chareen Elton<br>Laura Searle / Michael Little |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

## Table of Contents

# Table of Figures

# 1.0 Introduction

## 1.1 System Overview

The Latrobe Scheduling System is designed to address the client-defined constraints and improve the operational efficiency of managing employee information, subject instance scheduling, and overall scalability. It includes various operational components to achieve this:

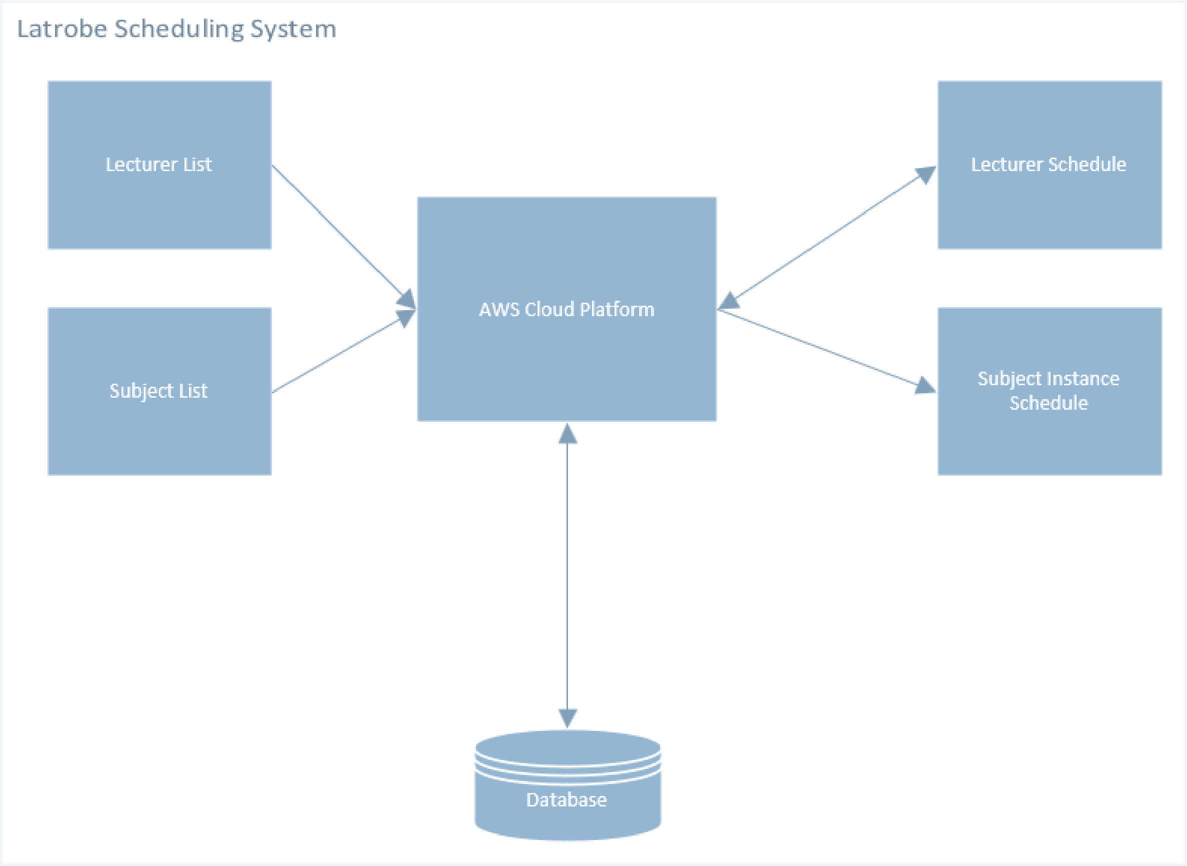


*Figure 1: Block Diagram.*

*Employee Information Management:*

The system solves the problems associated with manual data entry by automating the recording of employee information. It includes fields to record expertise, availability, workload balance, employment type, course development, and additional support.

*Subject Instance Scheduling:*

The system introduces validation and verification processes to address the issues with constantly changing student enrolment. It includes real-time updates for current instance details such as start and end dates, assigned lecturers and support personnel, course duration (set at 12 weeks), enrolment details, and required resources.

*Scalability:*

The system overcomes the limitations of the current structure by providing a dynamic and scalable solution. It can accommodate increasing demand and promptly process enrolments.

*Cloud-based Environment:*

The system replaces the current standalone system with a cloud-based environment. This addresses the issues of accessibility, reliability, and security. The system ensures convenient accessibility for all users while controlling access and actions based on user roles and permissions.

*User-friendly Interface:*

The system includes a user-friendly interface that improves user experience and reduces errors. It caters to different user roles, allowing different users to interact with the system based on their perspective roles.

*Centralised Database:*

The system utilises a centralised database for easy updating and accessing of staff details. This centralisation improves resource utilization by enabling efficient allocation and improved workload distribution.

*Data Security and Integrity:*

The system provides robust data security measures to protect sensitive information. It includes encryption for data protection, and access control through user permissions and authentication.

*Complex Schedule Support:*

The system supports complex schedules, enabling efficient management of subject instances with varying durations, lecturer availability, and other scheduling constraints.

## 1.2 Audience Description

The intended audience for the La Trobe Schedule System consists of three user types: managers who can create and manage a schedule for a lecturer, lecturers who can access the system and view their assigned schedule, and administrators who administer the system.

Administrator users will require level 2 or 3 experience in System Administration, SQL database management, and front-end and back-end development, including ASP.NET. Administrators will receive training on the new system to understand how to administer it effectively.

Managers are required to have previous experience in managing staff and have been trained on La Trobe's scheduling requirements. Managers will receive training on the new system to understand how to create and manage lecturer schedules.

Lecturers do not require previous experience, and training will be provided on how to access the system and view their assigned schedules.

## 1.3 Applicability Statement

The infrastructure for our system will be established on Amazon Web Services (AWS). This includes the hosting of both the web system and the associated data. By leveraging the capabilities of AWS, we aim to ensure a reliable and scalable environment for our web-based application, covering both its functionality and the storage and retrieval of data. This choice of hosting on AWS is intended to provide a robust foundation for the performance and accessibility of our system.

| Development | Technology | Version |
|---|---|---|
| Front-End | HTML | HTML5 |
| | CSS | CSS3 |
| | JavaScript | ECMAScript 2021 |
| | jQuery | 3.6.0 |
| | Bootstrap | 5 |
| Back-End | C# | 9.0 |
| | ASP.NET | Core 6 |
| Data Source | Microsoft SQL Database | SQL Server 2019 |
| Version Control | GitHub | 3.11.1 |

## 1.4 Purpose Statement

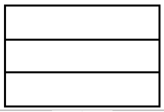
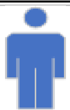
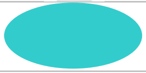
The software is designed to address the challenges associated with manual data entry and unreliable employee information management. By automating processes, the software ensures accurate recording of relevant staff details, including availability, workload balance, employment type, and course development involvement. It enables effective staff utilisation by assigning subject instances based on specific criteria such as technical skills, concurrent availability, and workload limits. The systems scalability allows for accommodating increasing demand and prompt processing of enrolments. It enhances accessibility, reliability, and security, promoting user convenience and control access based on user roles and permissions.

## 1.5 Document Usage Description

| Relationship | Document Section | Usage |
|---|---|---|
| Planning and Overview | Table of Contents | Sections of the document are listed to provide an overview of the document contents. |
| | Introduction | Provides an overview of the documents purpose and relations to the project. |
| | System Overview | Highlights the scheduling system, features, constraints, and benefits. |
| | Audience Description | Identifies the intended audience of the document. |
| | Applicability Statement | Identifies scenarios in which the system can be used and highlights its importance to the organisation. |
| | Purpose Statement | Clear statement of the document's main objective. |
| | Conventions | Specifies guidelines and protocols. |
| | Change Log | Records modifications made to the document chronologically. |
| Development Process | Software Design Scope | Defines the extent of the software design process. |
| | Major Software Functions | Outlines major functional requirements of the scheduling system. |
| | Major Design Constraints and other Requirements | Identifies restrictions or prerequisites that should be considered. |
| | Reference Documents | Lists additional relevant reference material. |
| | Existing Software Documentation | Provides overview of relevant existing software documentation. |
| | System Documentation | Outlines documentation necessary for the scheduling system. |
| | Vendor Documentation | Outlines vendor-specific documentation requirements. |
| User Story Development | User Stories | Provides an overview of the user stories for the scheduling system and how they relate to the design process. |
| | User Story Dictionary | All-inclusive glossary containing user story terminology and definitions. |
| | Iterative User Story Documents | Documents the sequential process of developing user stories. |
| | User Story Definition | Outlines each user story and its relevant requirements. |
| Design Development | Flow of Interaction Diagram | Displays a visual representation of the order in which system components interact with each other. |
| | Object-Oriented Design | Provides an overview of the system's design using an object-oriented approach. |
| | High-Level System Architecture | Provides a high-level overview of the scheduling systems architecture. |
| | High Level Package Diagram and Components | Outlines the systems high-level component diagram and overall structure. |
| | Use Case Analysis | Describes the use cases of the scheduling system in detail. |
| | Domain Model and Class Diagram | Presents a diagram and model depicting the systems domain and class. |

| | | |
|---|---|---|
| | Establishment of the Database Objects and Data Access Strategy | Defines the plan for creating database objects and managing data access. |
| | Sequence Diagram | Identifies sequence diagram requirements. |
| | Object Dictionary | Provides a detailed glossary of terms and definitions related to objects. |
| Testing and Reporting | User Story Testing | Explains how the testing process for user stories correlates with the overall system development process. |
| | Integration Test | Identifies integration testing components. |
| | Wireframes | Offers an initial concept for the interface design. |
| | Software Release Report | Details the process of releasing the software for the scheduling system. |
| | Usability Test Report | Explains the process and outcomes of usability testing. |
| | System (User Story) Test Report | Provides a report on the results of the system test based on user story requirements. |
| Additional Design Requirements | Additional User Interface Design | Outlines the process for integrating additional features into the user interface. |
| | Additional Input and Output Screens | Outlines optional additional input and output screens and their integration. |
| | Additional Data Display Screens | Outlines incorporation of extra screens for displaying data. |

## 1.6 Conventions

| # | Conventions | Description |
|---|---|---|
| 1. | **Bold Text** | Highlights important information within the sub-headings |
| 2. | *[Placeholder Text]* | Indicates the information is not yet available |
| 3. | Heading 1 | Represents the description heading or title of a section |
| 4. | Heading 2 | Represents the subheading or subsection within the main section |
| 5. | Heading 3 | Represents a further sub-level topic of a subheading or subsection |
| *6.* | *Heading 4* | Represents a further sub-level topic of a Heading 3 |
| 7. | Figure Reference | References the figures provided throughout the document |
| 8. | Hyperlink Text | Represents a link to an external document or website |
| 9. | ***Bold green text*** | Represents the user story requirements |
| 10. |  | Class – defines properties and behaviours of that class. |
| 11. |  | Actor – an entity that interacts with the system to perform a task. |
| 12. |  | Use case attribute – information associate with that use case. |

## 1.7 Change Log

*[To be provided upon first release of system maintenance documentation with relevance to the document modification in the table of changes]*

| Change Number | Date of Change | Author | Description of Change | Status | Affected Components | Comments/ Notes |
|---|---|---|---|---|---|---|
| CN1.0 | | | | | | |
| CN2.0 | | | | | | |
| CN3.0 | | | | | | |
| CN4.0 | | | | | | |
| CN5.0 | | | | | | |
| CN6.0 | | | | | | |
| CN7.0 | | | | | | |

# 2.0 Software Design Scope

The software design scope encompasses the management of staff information, workload distribution, resource allocation, and schedule optimization. This includes two sections: Major Software Functions (which refers to the primary tasks and capabilities that the system will perform), and Major Design Constraints (limitations and requirements involved in system development).

## 2.1 Major Software Functions

The major software functions have been broken down into separate sections based on their requirements and specific functionalities. This breakdown allows for a clear understanding of the different roles and tasks within the software, ensuring efficient management of staff information, workload distribution, resource allocation, and optimisation of schedules.

### *User Roles:*

- Administration: Entering and editing of data.
- Management: Creating and overriding staff schedules.
- Lecturers: read-only access.

### *Concurrent Instances:*

- Enforce maximum concurrent instances assigned to a lecturer.

### *Staff Database:*

- Store all staff member details including expertise, subject development, availability, and workload.

### *Subject Instance:*

- Extract subject information including start date, duration, and student count.

### *Lecturer Association:*

- Associate lecturers with specific instances based on expertise, availability, and workload.

### *Schedule:*

- Dynamically create a schedule considering staff availability.

### *Benefits to Users:*

- Access control and permissions ensure appropriate access to sensitive data, maintaining integrity and security.
- Ensures fair distribution of workloads preventing overload.
- Centralised storage for easy access and management of information.
- Provides necessary information about subjects to assist with planning and scheduling.
- Considers various factors, improving efficiency and minimising conflicts.

### *Benefits to Latrobe:*

- Ensures data confidentiality and reduces risks associated with unauthorised access and breaches.
- Assists with workload management.
- Facilitates decision-making processes and resource allocation.
- Provides data necessary for planning.
- Maximises utilisation of lecturers’ expertise improving teaching quality.
- Assists in planning and managing content development.

## 2.2 Major Design Constraints and other Requirements

Designing and implementing Latrobe's staff scheduling system requires careful planning and consideration of the following major constraints to effectively manage resources, optimise lecturer assignments and ensure seamless integration and operation of Latrobe's unique scheduling framework.

*Performance & Scalability* – the system must process and maintain substantial amounts of data including employee details, subject instances, and scheduling information without significant disruptions or performance and reliability issues. The system's modular and extensible architecture accommodates increasing workload and data throughput.

*User friendly interface* – multiple user-friendly interfaces (including admin, manager, and standard user) must be developed and implemented to facilitate different access levels and user functionality. The intuitive and responsive interfaces must support a logical design for easy navigation and include a modern visually appealing aesthetic.

A tailored dashboard (for each user type) supports easy navigation, providing an overview of relevant scheduling information including filtering, and search capabilities, simplifying the scheduling process.

*Cloud Environments* – the software must run on cloud infrastructure to leverage scalability and availability benefits cloud platforms provide. The system must be compatible with selected programming languages, frameworks, and databases.

A responsive web-based infrastructure enables high availability, and accessibility for various devices, and users providing positive user experiences.

*Security and Access Control* – to protect sensitive and confidential employee and scheduling information, security measures including encryption techniques will be implemented.

Access control mechanisms guarantee only authorised personnel can create, view, and modify scheduling data. Access control processes include user authentication, role-based access control (RBAC) and data encryption techniques.

*Integration capabilities* – the system must be compatible and integrate seamlessly with Latrobe's existing IT infrastructure including the learning and student management system, and financial systems.

*Complex scheduling logic* – the system must handle complex scheduling scenarios, including different term start times, varying frequencies of subject instances, and overlapping of instances for lecturers.

Algorithms must balance lecturer workloads, while considering their current commitments to teaching and content development.

*Workload* – the system must differentiate between part-time and full-time lecturers and dynamically adjust workloads based on their status.

Teaching and content development participation is considered to maintain maximum workload thresholds, and automatically trigger warnings or flags when approaching workload capacity.

*Assigning Subject Instances* – lecturer and specific subject instances must be appropriately assigned and based on expertise, availability, and development workload. A robust database containing lecturer skills and current commitments is required.

*Budget* – the system must consider budget constraints or limitations including cloud-based solutions with associated ongoing operational costs.

*Compliance* – when collecting, using, disclosing, and securing personal information, the system must adhere to data protection laws outlined in the Privacy Act 1988 (Cth) and the Australian Privacy Principles (APPs).

## 3.0 Reference Documents

This section provides an overview of the existing software documentation for Latrobe's online scheduling system, focusing on the unique challenges in delivering online courses. It outlines the considerations and criteria for associating lecturers with their specific subject instances, emphasizing the need for a staff scheduling system that can efficiently manage various factors.

### 3.1 Existing Software Documentation

*Project Description:*

**Staff Scheduling Overview**

We have an issue with our staff scheduling. Our organisation develops and delivers online Bachelor level degree courses on behalf of La Trobe University. We have an ever-growing number of subjects that we need to develop courseware material for and deliver against.

The delivery of subjects at our organisation is a little unusual when compared with that of other University courses. Most university courses run subjects over 12 weeks during two standard semesters. Semester One being in the first half of the year starting in early March, and Semester Two being in the second half of the year starting in early August after a mid-year gap. Many courses now also give the option of taking subjects during a summer semester.

In contrast at our organisation, we have terms that start every month i.e. there are 12 terms every year. Each term is 12 weeks or approximately three months long. Students complete terms back-to-back. In this manner students' study in four different terms every year.

The subjects that are running in each term are referred to as a Subject instance. For example, the subject Information Technology Fundamentals runs every month. This subject will therefore have a January instance, a February instance, a March instance etc all the way through to December every year.

Other subjects don't run every month. The second year subject Application Development in the Cloud for example only has an April and a November instance every year at present.

The lecturers that work within our organisation have a broad role. They are not only responsible for presenting lectures for content that is within their area of expertise, but also playing a significant part in developing that content in the first place. They need to be skilled in working in front of a camera in an online environment, and in dealing with the challenges of supporting students who are undertaking their studies remotely.

Lecturers with this level of expertise are prized, and we are lucky to have the group of academic staff that we have. It is however something of a challenge to manage their time effectively. It is important that no one lecturer becomes overburdened with too much development work while they are delivering a lot of subject instances. Likewise, no individual lecturer should have their talents underutilised through their being associated with too few subject instances at any given time.

How we schedule specific instances is something that we are currently able to manage ourselves. What we need some assistance with is how to associate individual lecturers with those specific instances.

Factors that we need to take into consideration when associating a lecturer with an instance are:

- Does the lecturer have the required level of expertise to deliver that subject instance?
- How many subject instances is the lecturer already associated with at that time?
- Is the lecturer currently working on developing one or more new subjects i.e. do they have scope to incorporate an extra instance?
- If they are currently developing a new subject, how hard it is to develop?
- How many students are enrolled in the subject instance?
- Is the lecturer running their current subject by themselves or with support?
- Is the lecturer full time or part time?

So, the question is what are we looking for from a staff scheduling system?

We want something that will:

- Store all staff members and their details relevant to the scheduling system.
- Be able to incorporate instance scheduling information extracted from the subject schedule.
- Use a combination of the above information to create a complete staff schedule.
- Be developed in a cloud-based environment.
- Has a user-friendly interface.
- Has an admin, manager, and standard user interface.
    - Admin interface used for entering base data and editing base data.
    - Manager interface used to manage development of the staff schedule.
    - Standard user interface for staff (read only).

Below is some extra relevant information:

- **Subject Lecturer Concurrent Instances**
    - Due to how subject instances are scheduled, it is common for an individual lecturer to be associated with instances that start one month apart. i.e. a lecturer delivering Information Technology Fundamentals could be associated with instances starting in January, February, and March. As each instance runs for approximately three months, the lecturer would be associated with the January and February instances during February, and all three instances during March.
- **Maximum Number of Concurrent Instances**
    - The maximum number of concurrent instances that a full-time lecturer can be assigned at any one time is 6. Depending on their load, part-time lecturers will have a maximum number of concurrent instances less than 6.

## 3.2 System Documentation

System authentication will be integrated into La Trobe University's existing Microsoft Azure Active Directory service. Additionally, the AWS SQL database will be incorporated into La Trobe University's established Microsoft Azure cloud infrastructure.

*System Integration:*

1. **System Authentication**

- **Overview:** The authentication mechanisms of the La Trobe scheduling system will be integrated into La Trobe University's existing Microsoft Azure Active Directory service, ensuring secure and consistent authentication across the organisation.
- **Service:** Microsoft Entra ID.

2. **System Data**

- **Overview:** The SQL database used for the scheduling system will be migrated and incorporated into La Trobe University's existing Microsoft Azure cloud infrastructure environment.
- **Service:** Microsoft Azure SQL Server Cloud.

*Vendor Documentation Links:*

- Microsoft Entra ID
- Microsoft Azure SQL Server Cloud

## 3.3 Vendor Documentation

The scheduling system is designed to integrate with Microsoft SQL Server for database management, AWS for cloud-based services, and ASP.NET Web Application for front-end user interactions.

*Third-Party Products:*

1. **Microsoft SQL Server**
    - **Version:** SQL Server 2019
    - **Purpose:** Database management system for storing and managing application data.
2. **Amazon Web Services (AWS)**
    - **Services:** AWS S3 for scalable object storage infrastructure, AWS Lambda for serverless functions, and AWS RDS for database support.
    - **Purpose:** Cloud-based services.
3. **ASP.NET Web Application**
    - **Framework:** ASP.NET Core
    - **Purpose:** Front-end user interface for interaction with the application and database connectivity.

*Vendor Documentation Links:*

- Microsoft SQL Server Documentation
- AWS Documentation
- ASP.NET Core Documentation

*Component Integration:*

1. **Microsoft SQL Server Integration:**
    - **Data Storage:** Storing user and application data in a SQL database.
    - **Data Retrieval:** Retrieving relevant information from a SQL database.
    - **Data Management:** Executing SQL queries and transactions for data manipulation.
2. **Amazon Web Services (AWS) Integration:**
    - **Object Storage:** Storing and retrieving files using AWS S3.
    - **Serverless Functions:** Utilising AWS Lambda for functions and tasks.
    - **Database Support:** Utilising AWS RDS for database hosting.
3. **ASP.NET Web Application Integration:**
    - **User Interface:** Presenting data from the back end to users.
    - **User Interaction:** Capture user inputs and interactions.
    - **Database connectivity:** Connect to SQL database.

*Requirements & Compatibility:*

1. **Microsoft SQL Server:**
    - Microsoft Server 2019 or later
2. **Amazon Web Services (AWS):**
    - AWS Account
    - Amazon S3
    - AWS RDS for SQL Server
    - AWS Lambda
    - AWS Identity and Access Management (IAM)
3. **ASP.NET Web Application:**
    - Ensure compatibility with ASP.NET Core Framework.

*Testing & Validation*

Comprehensive testing will be conducted to ensure integration with Microsoft SQL Server, AWS services, and ASP.NET web application.

Testing components include:

- Unit testing.
- Integration testing.
- Component testing.
- Security testing.
- Documentation review and validation.

*Documentation Review & Updates:*

Regularly reviewing updates and changes in Microsoft SQL Server, AWS, and ASP.NET Core documentation for any new features, best practices, or security considerations. By staying informed on changes to these technologies, our team aims to incorporate enhancements and proactively adapt our system and enhance the overall effectiveness and performance of the system.

## 3.4 Other Documentation

The inclusion of the following technologies is essential to the development process:

Front-end technologies, such as HTML5, CSS3, and JavaScript, provide a modern and responsive design. Frameworks like jQuery and Bootstrap will further enhance the front-end development process. On the back end, the system will be developed with C# and Razor Pages to deliver efficient server-side architecture. To maintain version control and collaboration throughout the development lifecycle, the project will be hosted on GitHub.

*System Development Technologies:*

1. **HTML**
   - **Version:** HTML5
   - **Purpose:** Structure the web system and its content.
2. **CSS**
   - **Version:** CSS3
   - **Purpose:** Styling the web system.
3. **JavaScript**
   - **Version:** ECMAScript 2021
   - **Purpose:** Implement web page interactivity.
4. **jQuery**
   - **Version:** 3.6.0
   - **Purpose:** Add additional features and functionality to JavaScript.
5. **Bootstrap**
   - **Version:** 5
   - **Purpose:** Incorporate additional design features into the system.
6. **C#**
   - **Version:** 9.0
   - **Purpose:** Enable server-side processing.
7. **GitHub**
   - **Version:** 3.11.1
   - **Purpose:** Version control and project collaboration.
8. **Razor Pages**
   - **Version:** ASP.NET Core 6
   - **Purpose:** Embed server-side code.

*Documentation Links:*

- HTML
- CSS
- JavaScript
- jQuery
- Bootstrap
- C#
- GitHub
- Razor Pages

*Requirements & Compatibility:*

1. **HTML**
    - Supported by all modern web browsers.
    - DOCTYPE Declaration.
    - W3C Markup Validation Service can be used to check HTML documents for compliance with W3C standards.
2. **CSS**
    - Supported by all modern web browsers.
    - Some features may require vendor prefixes for compatibility.
    - W3C CSS Validation Service can be used to check CSS files for compliance with W3C standards.
3. **JavaScript**
    - Supported by all modern web browsers.
    - Different browsers may have varied levels of support for specific features.
    - DOCTYPE Declaration.
    - ECMAScript models included in ES6 help structure and organise code.
4. **jQuery**
    - Cross-browser compatible, supported by all modern web browsers.
    - DOCTYPE Declaration.
5. **Bootstrap**
    - Cross-browser compatible, supported by all modern web browsers.
    - HTML5 and CSS3.
    - jQuery dependency.
    - Some components may require JavaScript.
    - DOCTYPE Declaration.
6. **C#**
    - .NET Framework or .NET Core.
    - ASP.NET.
7. **GitHub**
    - GitHub account.
    - GitHub application.
8. **Razor Pages**
    - ASP.NET Core.

*Documentation Review & Updates:*

Each technology mentioned above will be reviewed for changes and updates to reflect a proactive approach within the development process. The aim is to stay informed about updated features, security updates, and requirements and compatibility considerations. Reviewing each technology ensures that our team can strategically implement updates, maintain version control, implement the latest features, address security vulnerabilities, and build a robust, up-to-date system.

# 4.0 User Stories

The following user stories are short descriptions of specific features and / or functionality of the Latrobe scheduling system.

## 4.1 User Story Dictionary

| | USER STORY |
|---|---|
| 1. | As an administrator, I want to ***create new users***, so I can allow users access to the system. |
| 2. | As an administrator, I want to ***edit existing user details***, so I can maintain accurate user information in the database. |
| 3. | As an administrator, I want to ***delete existing user accounts***, so I can prevent unauthorised access & maintain accurate information in the database. |
| 4. | As an administrator, I want to ***create new subjects***, so I can effectively manage resources. |
| 5. | As an administrator, I want to ***edit existing subjects,*** so I have accurate information stored in the database. |
| 6. | As an administrator, I want to ***delete existing subjects***, so I only have current information stored in the database. |
| 7. | As an administrator, I want to ***create new subject instances***, so I can effectively manage subject instances and accommodate changing requirements. |
| 8. | As an administrator, I want to ***edit existing instances***, so I can make necessary adjustments as required. |
| 9. | As an administrator, I want to ***delete existing instances***, so I can accommodate changing demands. |
| 10 | As a user, I want to ***login using my credentials***, so I can access the dedicated user interface. |
| 11 | As an administrator, I want to be able ***to assign roles and permissions*** to control access, activities, and user functionality |
| 12 | As a user, I want to ***securely logout of the system***, so I can secure my account and ensure privacy. |
| 13 | As a manager, I want the ***system to automatically assign suitable staff members*** to subject instances based on expertise, workload & availability so I can optimise staff allocations and effectively manage resources. |
| 14 | As a manager, I want to ***receive notifications when there is a conflict***, so I can make informed decisions. |
| 15 | As a manager, I want to ***override the system***, so I can adjust and balance workloads. |
| 16 | As an administrator, I want the ***system to be cloud-based***, so I can allow high availability and easy user accessibility. |
| 17 | As a user, I want ***a read-only interface***, to view my assigned subject instances and schedule, so I can plan accordingly. |
| 18 | As an administrator, I want ***a user-friendly interface*** for entering & editing base data, so I can effectively manage essential information used by the system. |
| 19. | As a manager, I want to ***access a user-friendly manager interface*** so I can manage staff schedules, development and make necessary adjustments. |
| 20 | As a manager, I want the ***system to generate reports***, so I can make informed scheduling decisions. |
| 21 | As a lecturer, I want to be able ***to enter my availability***, so it can be considered when assigning subject instances and creating the schedule. |
| 22 | As a manager, I want to ***view lecturer availability, expertise and workload***, so I can make informed decisions when allocating subject instances. |

## 4.2 Iterative User Story Documents

This iterative section provides a structured roadmap for the development and enhancement of the scheduling system. It provides a clear framework for identifying and outlining user stories, acceptance criteria and expected outcomes, thereby enabling effective guidance and refinement throughout the development process.

### 4.2.1 User Story Definition

Below, you will find detailed descriptions for each user story outlined in section 4.1.

1. Create New Users:
    - User Story: As an administrator, I want to create new users, so I can allow users access to the system.
    - Description: The system should include a user creation interface, as well as the capability for administrators to perform bulk user creation. This functionality should allow administrators to input user details and any additional required information. The system will conduct validation checks to ensure data accuracy and create a new user account with the appropriate access permissions.
2. Edit User Details:
    - User Story: As an administrator, I want to edit existing user details, so I can maintain accurate user information in the database.
    - Description: The system should provide a user management interface allowing administrators to search for and edit the details of existing users. This includes the ability to modify personal details, employment status, and any other relevant user information. Changes made should be validated before updating the database to ensure data accuracy.
3. Delete User Accounts:
    - User Story: As an administrator, I want to delete existing user accounts, so I can prevent unauthorised access & maintain accurate information in the database.
    - Description: The system should feature a user management interface with the capability to delete user accounts. Upon deletion, the system must handle permissions appropriately to maintain data integrity and prevent unauthorised access.
4. Create New Subjects:
    - User Story: As an administrator, I want to create new subjects, so I can effectively manage resources.
    - Description: The system should have an intuitive interface for administrators to input details for a new subject, including the subject name, subject difficulty/year, and any other relevant information. Upon submission, the system should validate the data and create a new subject in the database for efficient resource management.
5. Edit Existing Subjects:
    - User Story: As an administrator, I want to edit existing subjects, so I have accurate information stored in the database.
    - Description: The system should provide an interface for administrators to search for and edit details of existing subjects. This includes the ability to modify subject names, difficulty/year, and other relevant information. Changes should be validated before updating the database to ensure data accuracy.

6. Delete Existing Subjects:
    - User Story: As an administrator, I want to delete existing subjects, so I only have current information stored in the database.
    - Description: The system should have an interface for administrators to delete subjects. Upon deletion, the system should handle any dependencies or relationships including subject instances, to ensure data integrity and maintain an up-to-date database.
7. Create New Subject Instances:
    - User Story: As an administrator, I want to create new subject instances, so I can effectively manage subject instances and accommodate changing requirements.
    - Description: The system should provide an interface for administrators to input details for a new subject instance, such as the instance name, associated subject, start and end dates, content development requirements, and any other relevant information. Upon submission, the system should validate the data and create a new subject instance in the database, facilitating effective management.
8. Edit existing Instances:
    - User Story: As an administrator, I want to edit existing instances, so I can make necessary adjustments as required.
    - Description: The system should offer an interface for administrators to search for and edit details of existing subject instances. This includes the ability to modify instance names, associated subjects, start and end dates, content development requirements, and other relevant information. Changes should be validated before updating the database to maintain data accuracy.
9. Delete Existing Instances:
    - User Story: As an administrator, I want to delete existing instances, so I can accommodate changing demands.
    - Description: The system should provide an interface for administrators to delete subject instances. Upon deletion, the system should handle any dependencies or relationships to ensure data integrity and accommodate changing demands.
10. User Login:
    - User Story: As a user, I want to login using my credentials, so I can access the dedicated user interface.
    - Description: The system should offer a secure login interface for users to enter their credentials. Upon successful authentication, users should gain access to their respective user interface, ensuring a seamless and secure login experience.
11. Assign Roles and Permissions:
    - User Story: As an administrator, I want to be able to assign roles and permissions to control access, activities, and user functionality.
    - Description: The system should provide an interface for administrators to assign specific roles and permissions to users. This includes controlling access to features and functionalities based on the assigned role, ensuring proper access control.
12. Secure Logout:
    - User Story: As a user, I want to securely logout of the system, so I can secure my account and ensure privacy.
    - Description: The system should offer a secure logout feature that terminates the user session, ensuring the account is protected, and user privacy is maintained.

13. Automatic Staff Assignment:
    - User Story: As a manager, I want the system to automatically assign suitable staff members to subject instances based on expertise, workload & availability so I can optimise staff allocations and effectively manage resources.
    - Description: The system should have an automated process that considers staff expertise, workload, and availability to intelligently assign appropriate staff members to subject instances. This feature optimizes resource allocation for effective management.
14. Conflict Notifications:
    - User Story: As a manager, I want to receive notifications when there is a conflict, so I can make informed decisions.
    - Description: The system should be designed to detect conflicts in staff assignments or scheduling and notify managers promptly. Notifications should include details of the conflict to facilitate informed decision-making.
15. Override Workloads:
    - User Story: As a manager, I want to override the system, so I can adjust and balance workloads.
    - Description: The system should allow managers to manually override conflicting scheduling assignments, enabling adjustments to staff workloads as needed. This feature provides flexibility in managing resources.
16. Cloud-based System:
    - User Story: As an administrator, I want the system to be cloud-based, so I can allow high availability and easy user accessibility.
    - Description: The system should be hosted on a cloud infrastructure, providing high availability, scalability, and easy accessibility for users. This ensures efficient system performance and accessibility from different locations.
17. Read-Only Interface for Users:
    - User Story: As a user, I want a read-only interface, to view my assigned subject instances and schedule, so I can plan accordingly.
    - Description: The system should offer users a read-only interface, allowing them to view details of their assigned subject instances and schedules without the ability to make changes. This feature supports effective planning.
18. User-Friendly Interface for Base Data:
    - User Story: As an administrator, I want a user-friendly interface for entering & editing base data, so I can effectively manage essential information used by the system.
    - Description: The system should provide administrators with a user-friendly interface for inputting and editing base data. This includes data validation and error handling to ensure accuracy, facilitating efficient management of crucial information.
19. Manager Interface:
    - User Story: As a manager, I want to access a user-friendly manager interface so I can manage staff schedules, development and make necessary adjustments.
    - Description: The system should offer a user-friendly manager interface that allows managers to efficiently manage staff schedules, development, and make necessary adjustments. This interface should be intuitive and accessible for effective managerial tasks.

20. Generate Reports:

- User Story: As a manager, I want the system to generate reports, so I can make informed scheduling decisions.
- Description: The system should have the capability to generate reports that provide insights into staff assignments, resource utilisation, and other relevant data. These reports assist managers in making informed decisions when scheduling resources.

21. Lecturer Availability Entry:

- User Story: As a lecturer, I want to be able to enter my availability, so it can be considered when assigning subject instances and creating the schedule.
- Description: The system should provide a feature for lecturers to input their availability. This information should be considered during the automated assignment of subject instances and the creation of schedules, ensuring efficient use of lecturer availability.

22. View Lecturer Information:

- User Story: As a manager, I want to view lecturer availability, expertise and workload, so I can make informed decisions when allocating subject instances.
- Description: The system should offer a manager interface that allows managers to view detailed information about lecturers, including availability, expertise, and workload. This information supports managers in making informed decisions when assigning subject instances and managing resources effectively.

### 4.2.2 Flow of Interaction Diagram

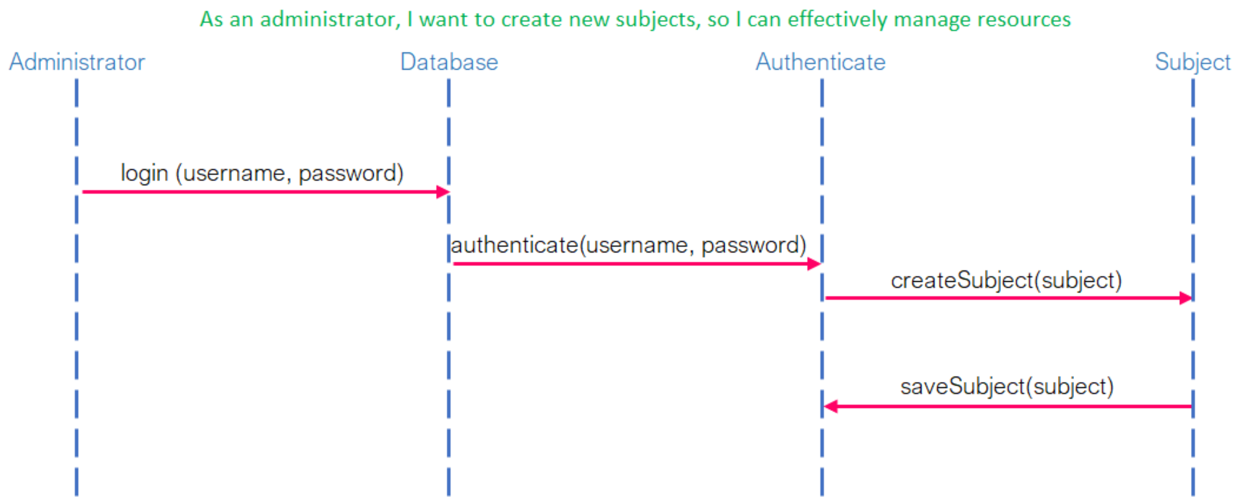

As an administrator, I want to create new subjects, so I can effectively manage resources
Administrator
Database
Authenticate
Subject
login (username, password)
authenticate(username, password)
createSubject(subject)
saveSubject(subject)

*Figure 2: Flow of interaction diagram - create subjects.*

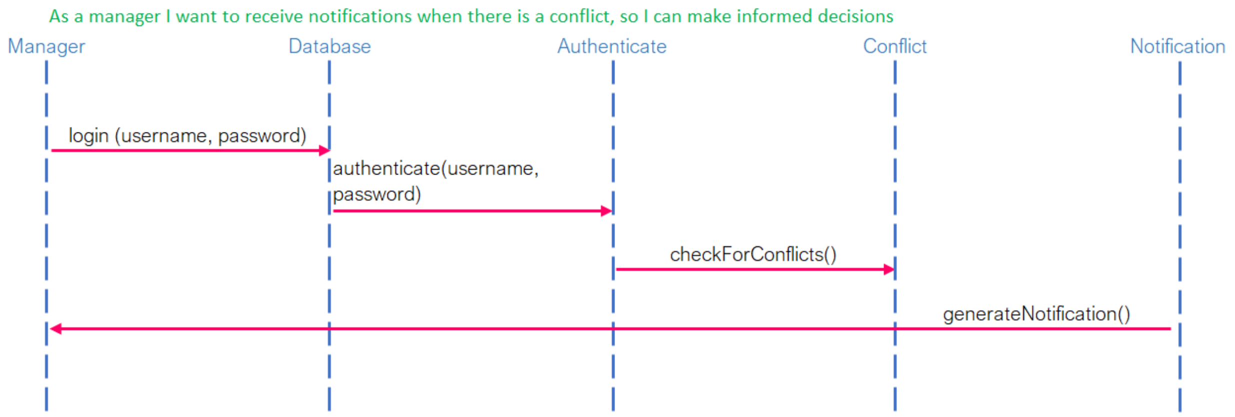

As a manager I want to receive notifications when there is a conflict, so I can make informed decisions
Manager
Database
Authenticate
Conflict
Notification
login (username, password)
authenticate(username, password)
checkForConflicts()
generateNotification()

*Figure 3: Interaction diagram – generate notifications.*

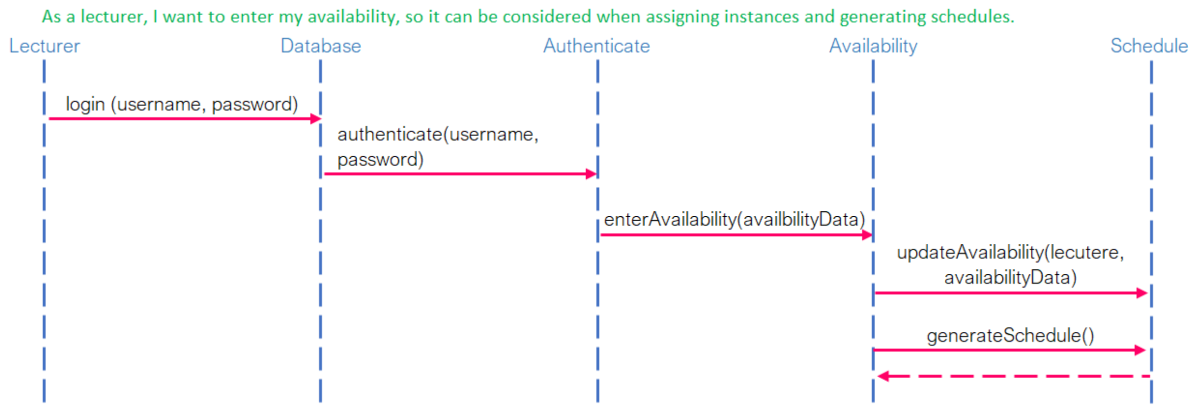

As a lecturer, I want to enter my availability, so it can be considered when assigning instances and generating schedules.
Lecturer
Database
Authenticate
Availability
Schedule
login (username, password)
authenticate(username, password)
enterAvailability(availbilityData)
updateAvailability(lecutere, availabilityData)
generateSchedule()

*Figure 4: Interaction diagram - enter availability.*

### 4.2.3 User Story Testing

| Test ID | User Story | Test Objective | Acceptance Criteria | Expected Output | Status (Pass/Fail) |
|---|---|---|---|---|---|
| LSS001 | As an administrator, I want a user-friendly interface for entering & editing base data, so I can effectively manage essential information used by the system | To verify the usability and intuitiveness of the admin interface | Administrators access the dedicated administration interface. Easily navigate the system including locating required functionalities and perform tasks without confusion | The user interface should have clear navigation, logical organization of features, and an overall intuitive design | *[To be provided upon testing completion ]* |
| LSS002 | As an administrator, I want the system to be cloud-based so I can allow high availability and easy user accessibility | To ensure the system is cloud-based, providing necessary features for availability and accessibility | The system should be accessible from any location with an internet connection, ensuring high availability and convenient usage | The system should provide smooth access to all authorised administrators regardless of their location | *[To be provided upon testing completion ]* |
| LSS003 | As an administrator, I want to be able to assign roles and permissions to control access, activities, and user functionality | To validate the role-based access control and permissions are functioning as intended | The administrator should be able to assign different roles (such as administrator, manager, and lecturer) with appropriate permissions to control access and functionality | The system should restrict access and functionalities based on assigned roles and permissions | *[To be provided upon testing completion ]* |
| LSS004 | As a manager, I want to access a manager interface so I can manage staff schedules, development and make necessary adjustments | To verify the usability and intuitiveness of the manager interface | Authenticated managers access the dedicated manager interface. Locate required functionalities, and easily navigate the system | The manager interface should have clear navigation, logical organisation of features, and an overall intuitive design | *[To be provided upon testing completion ]* |
| LSS005 | As a manager, I want to view lecturer availability, expertise, and workload, so I can make informed decisions when allocating subject instances | To verify the system accurately displays lecturer information including availability, expertise and workload | The manager dashboard provides a list of lecturers with their current workload. A manager can click on a lecturer name to view detailed information including available days and timeslots | A list stores accurate lecturer information. When a name is clicked, further details are provided | *[To be provided upon testing completion ]* |
| LSS006 | As a manager, I want to receive notifications for conflicts to make informed decisions | To ensure the manager receives notifications for any scheduling conflicts | The system should notify the manager when there are conflicts such as overlapping schedules or lecturers exceeding workload limits | The manager should be alerted of any conflicts, allowing them to make informed decisions and resolve the issues | *[To be provided upon testing completion ]* |

| | | | | | |
|---|---|---|---|---|---|
| LSS007 | As a manager, I want to override the system so I can adjust and balance workloads | To validate the managers ability to manually adjust and balance lecturers' workloads | The manager should be able to override the systems automatic workload distribution and make manual adjustments to balance workloads | The system should reflect the managers adjustments, ensuring balanced workloads among lecturers | *[To be provided upon testing completion ]* |
| LSS008 | As a manager, I want the system to generate reports, so I can make informed scheduling decisions | To verify the system can generate relevant reports for managers | The system should generate reports containing schedule information, lecturer workloads, availability, and any other necessary details for informed decision-making | The generated reports should provide accurate and comprehensive information for scheduling decisions | *[To be provided upon testing completion ]* |
| LSS009 | As a lecturer, I want a read-only interface to view my assigned subject instances and schedule so I can plan accordingly | To validate the lecturer's ability to access and view their schedule easily | The lecturer should be able to access their schedule effortlessly, seeing detailed information about their classes, subjects, and timings | The schedule is displayed accurately, allowing the lecturer to manage their professional and personal life effectively | *[To be provided upon testing completion ]* |
| LSS010 | As a lecturer, I want to be able to enter my availability, so it can be considered when assigning subject instances and generating the schedule | To verify lecturers can enter, review, and edit valid availability | A lecturer can enter, review, and edit availability.<br>Availability can be entered by selecting preferred days of the week and timeslots.<br>A data range sets short and long-term availability.<br>The system validates availability, ensuring no conflicts. Changes made are reflected in the system. | Lecturers can enter, edit and review, availability.<br>A prompt appears confirming successful availability has been captured by the system | *[To be provided upon testing completion ]* |
| LSS011 | As an administrator, I want to create new users, so I can allow users access to the system | To verify, administrators can successfully create new users, and data is accurately captured by the system | Admin can enter required user details including name, address, contact details, expertise, and employment status. Form validation ensures data integrity. New user details are accurately reflected in the system | The "add user" form displays correctly.<br>Administrators can input and save user details.<br>Validated data is accurately and reflected in the database | *[To be provided upon testing completion ]* |
| LSS012 | As an administrator, I want to edit existing user details, so I can maintain accurate user information in the database | To verify, admin can successfully edit user details, and changes are accurately reflected in the system | Admin can search, select and edit existing user account. Changes are reflected in the database.<br>Users receive notifications regarding updates | Admin can search, edit and save existing user details.<br>Updated information is reflected in the database.<br>Notifications confirm user updates | *[To be provided upon testing completion ]* |

| | | | | | |
|---|---|---|---|---|---|
| LSS013 | As an administrator, I want to delete existing user accounts so I can prevent unauthorised access & maintain accurate information in the database | To verify admin can delete existing user accounts, and to ensure changes are reflected in the system | Admin can search and select existing user accounts. A confirmation request verifies deletion, preventing accidental removal. Selected users are then removed from the system and users can no longer access the system | Selected existing user accounts are removed from the system.<br>Users can no longer access the system | *[To be provided upon testing completion ]* |
| LSS014 | As an administrator, I want to create new subjects so I can effectively manage resources | To verify admin can create new subjects, and to ensure details are captured by the system | Admin can enter subject details including name, code, description, duration, difficulty and required support.<br>Subject details are captured by the system | Admin can create a new subject.<br>Subject details are reflected in the system. A prompt confirms successful creation of the new subject | *[To be provided upon testing completion ]* |
| LSS015 | As an administrator, I want to edit existing subjects, so accurate information is stored in the database | To verify, admin can successfully edit subject details, and ensure updates are captured by the system | Admin can search, select, edit and save modified details.<br>Changes are reflected in the database | Admin can search and edit subject details. Changes are reflected in the system. A prompt confirms successful update | *[To be provided upon testing completion ]* |
| LSS016 | As an administrator, I want to delete existing subjects that are superseded so current information is stored in the database | To verify admin can delete subjects, and ensure changes are accurately captured by the system | Admin can search, select and delete existing subjects. A confirmation request verifies deletion, preventing accidental removal. Subjects are permanently deleted from the system | Subjects are removed from the system.<br>A confirmation prompt verifies deletion.<br>Audit logs timestamp deleted subject | *[To be provided upon testing completion ]* |
| LSS017 | As an administrator, I want to create new subject instances, so I can effectively manage subject instances and accommodate changing requirements. | To verify admin can create a new subject instance and ensure details are captured by the system. | Admin can enter subject instance details including start date, end date, assigned lecturer and enrolment count.<br>Details are recorded in the database. | A new subject instance is created.<br>Details are reflected in the system.<br>A prompt appears confirming successful creation. | *[To be provided upon testing completion ]* |
| LSS018 | As an administrator, I want to edit existing instances so I can make necessary adjustments as required. | To verify, admin can edit details, and ensure details are captured by the system | Admin can search, edit and save existing instances details.<br>Changes are accurately reflected in the database. | Updated details are captured in the database.<br>A prompt confirming successful update of instance details | *[To be provided upon testing completion ]* |

| | | | | | |
|---|---|---|---|---|---|
| LSS019 | As an administrator, I want to delete existing instances so I can accommodate changing demands. | To verify admin can delete existing subject instances and ensure updates are accurately captured by the system. | Admin can search and select subject instances to delete.<br>A confirmation request verifies deletion, preventing accidental removal.<br>Selected instances are then permanently deleted from the system | Deletion of existing subject instance.<br>A confirmation prompt is displayed prior to deletion, preventing accidental removal.<br>A prompt confirms successful deletion. | *[To be provided upon testing completion ]* |
| LSS020 | As a user, I want to login using my credentials so I can access the dedicated user interface | To verify users can login using valid credentials and can access dedicated user interfaces | Users can access login landing page.<br>Users must provide valid credentials.<br>The system validates credentials against stored email and password.<br>Successful login redirects them to the users dedicated interface.<br>Failed attempts, displays appropriate error message | Successful login redirects users to their dedicated interface.<br><br>Failed attempts are redirected back to landing page | *[To be provided upon testing completion ]* |
| LSS021 | As a user, I want to securely logout of the system, so I can secure my account and ensure privacy | To verify users can securely logout of the system | The logout button is visible.<br>When clicked, a prompt confirms users' intention to logout.<br>Upon confirmation, active sessions are terminated.<br>Users are redirected to landing page. | Users are successfully logged out.<br>A prompt request to confirm logout.<br>Active sessions are terminated.<br>Users are redirected to landing page. | *[To be provided upon testing completion ]* |
| LSS022 | As a manager, I want the system to automatically assign suitable staff members to subject instances based on expertise, workload & availability so I can optimise staff allocations and effectively manage resources | To verify the system can automatically generate schedules based on expertise, availability, and workload | The system should display a list of suggested lecturers for each subject instance.<br>The system should consider lecturer expertise, current workload and availability when generating schedules | A generated schedule has considered expertise, current workload, and availability | *[To be provided upon testing completion ]* |

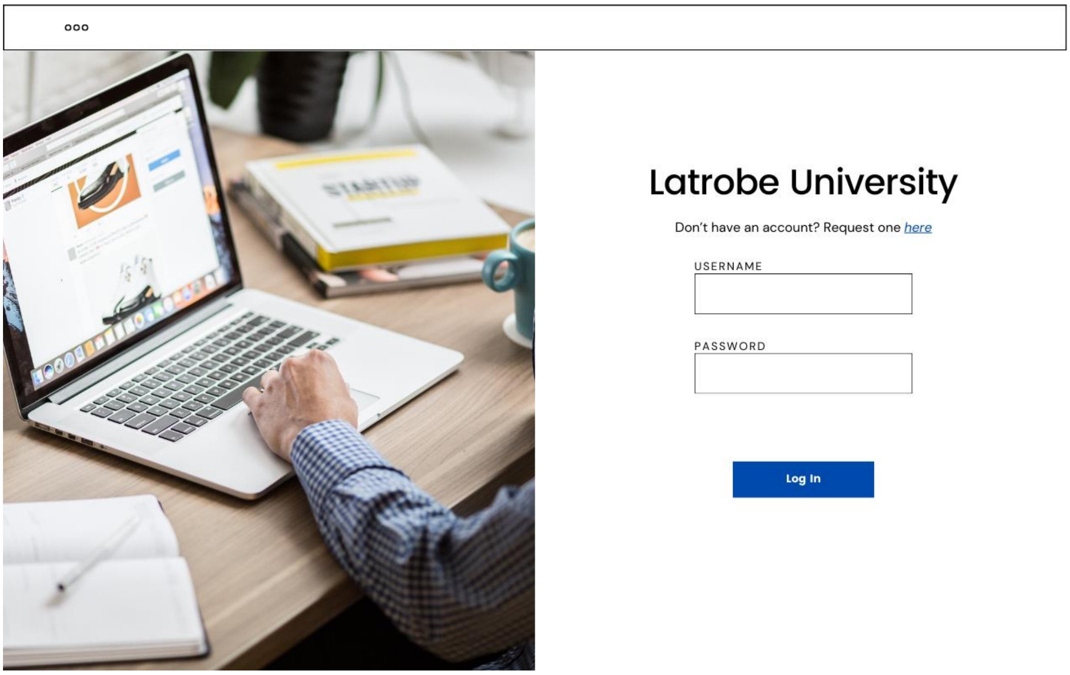
### 4.2.4 Integration Test (optional)

*[To be provided upon completion of system in order to test individual components]*

### 4.2.5 Wireframes

The following wireframes showcase the visual layout and structure of the software user interfaces for Latrobe's online scheduling system. These wireframes include key screens such as the login screen, administration screen, and management screen. These wireframes provide a simplified representation of the UI elements and their arrangements, providing visual references for understanding the design. (Mock-ups created using Canva)

*Login Page*

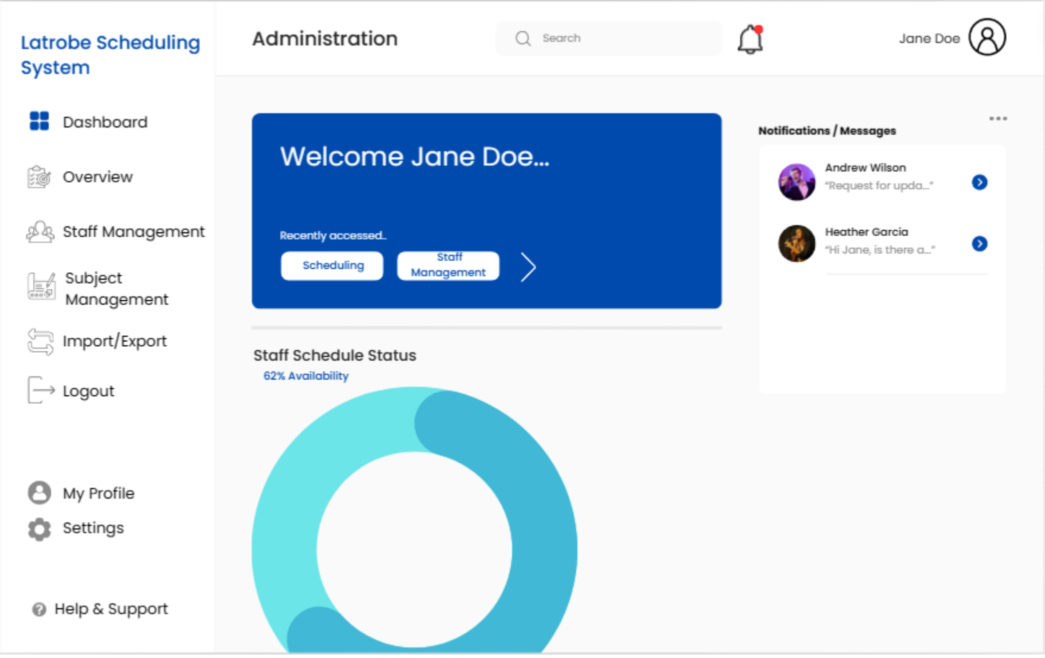


Latrobe University

Don't have an account? Request one here

*Figure 5: Login page.*

*Administration Page*



Latrobe Scheduling System

Administration

Welcome Jane Doe...

Notifications / Messages

Andrew Wilson "Request for upda..."

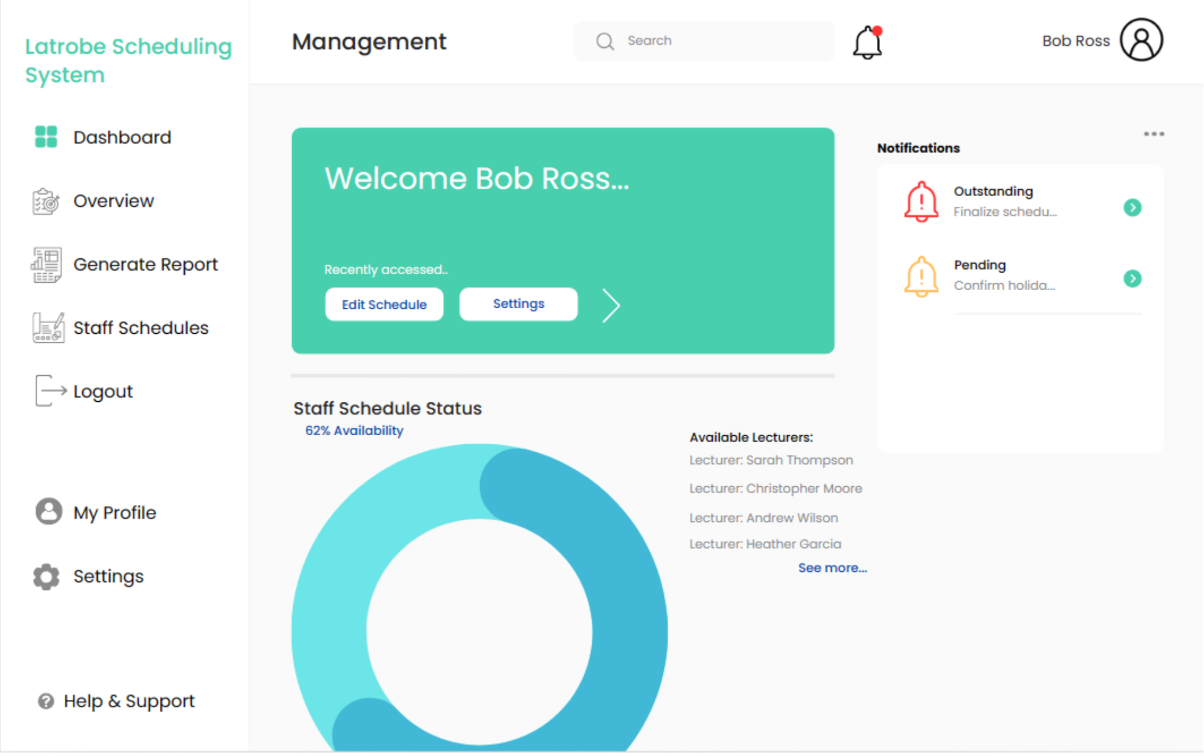
Heather Garcia "Hi Jane, is there a..."

Staff Schedule Status

62% Availability

*Figure 6: Administration page.*

## Management Page



Latrobe Scheduling System

Welcome Bob Ross...

Recently accessed..

Notifications

Outstanding
Finalize schedu...

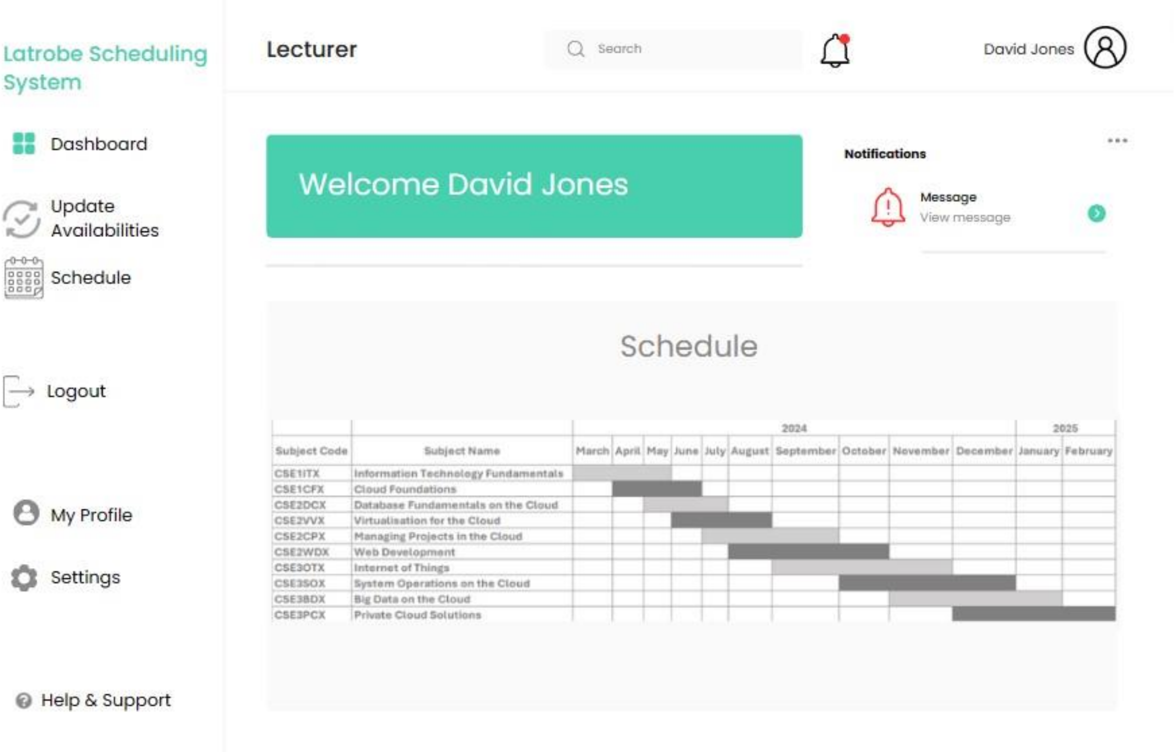
Pending
Confirm holida...

Staff Schedule Status
62% Availability

Available Lecturers:
Lecturer: Sarah Thompson
Lecturer: Christopher Moore
Lecturer: Andrew Wilson
Lecturer: Heather Garcia
See more...

*Figure 7: Manager page.*

## Lecturer Page



Latrobe Scheduling System

Welcome David Jones

Notifications

Message
View message

Schedule

| | | 2024 | | | | | | | | | | 2025 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subject Code | Subject Name | March | April | May | June | July | August | September | October | November | December | January | February |
| CSE1ITX | Information Technology Fundamentals | ░ | ░ | ░ | | | | | | | | | |
| CSE1CFX | Cloud Foundations | | ▓ | ▓ | ▓ | | | | | | | | |
| CSE2DCX | Database Fundamentals on the Cloud | | | ░ | ░ | ░ | | | | | | | |
| CSE2VVX | Virtualisation for the Cloud | | | | ▓ | ▓ | ▓ | | | | | | |
| CSE2CPX | Managing Projects in the Cloud | | | | | ░ | ░ | ░ | | | | | |
| CSE2WDX | Web Development | | | | | | ▓ | ▓ | ▓ | | | | |
| CSE3OTX | Internet of Things | | | | | | | ░ | ░ | ░ | | | |
| CSE3SOX | System Operations on the Cloud | | | | | | | | ▓ | ▓ | ▓ | | |
| CSE3BDX | Big Data on the Cloud | | | | | | | | | ░ | ░ | ░ | |
| CSE3PCX | Private Cloud Solutions | | | | | | | | | | ▓ | ▓ | ▓ |

*Figure 8: Lecturer page.*

# 5.0 Object-Oriented Design

The following object-oriented design element of this documentation provides fundamental aspects of the Latrobe Scheduling System's architecture. It involves a structural representation of entities, interactions, and integrations, as well as the relationships between them. This section also showcases the systems package diagram, use case analysis, and class diagram.

## 5.1 High Level System Architecture

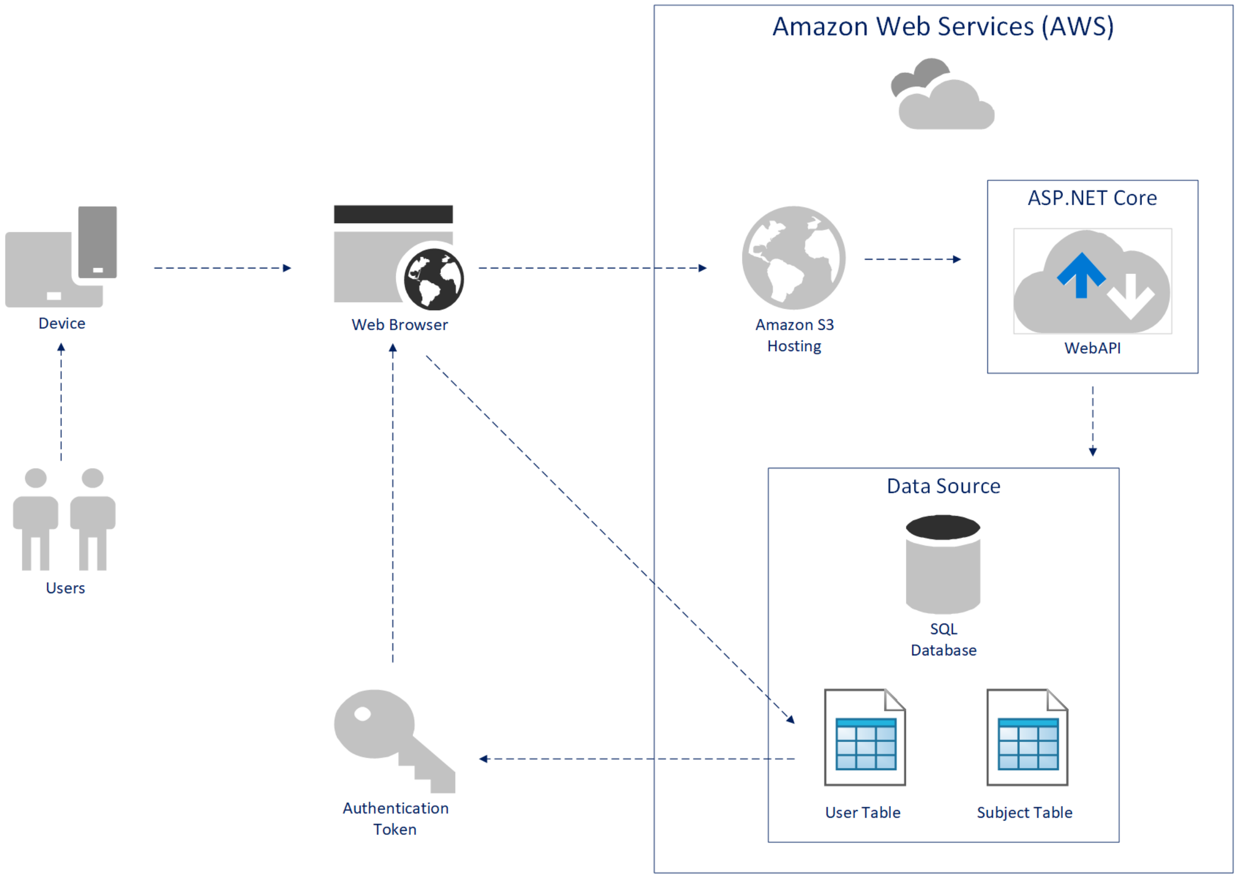


*Figure 9: High level system architecture.*

The La Trobe Scheduling system will be hosted within Amazon Web Services (AWS), specifically using the S3 hosting service. This enables users to access the application from any device with an internet connection. A relational SQL database will be hosted within AWS to manage system authentication, subjects, subject instances, and lecturer scheduling information. The ASP.NET Core framework will be used for the development of the system, and it will be integrated into AWS, utilising AWS services for hosting and infrastructure requirements. The data source of the application will be connected and hosted with AWS to facilitate data flow between the application and the hosted database.

### 5.1.2 High Level Software Architecture

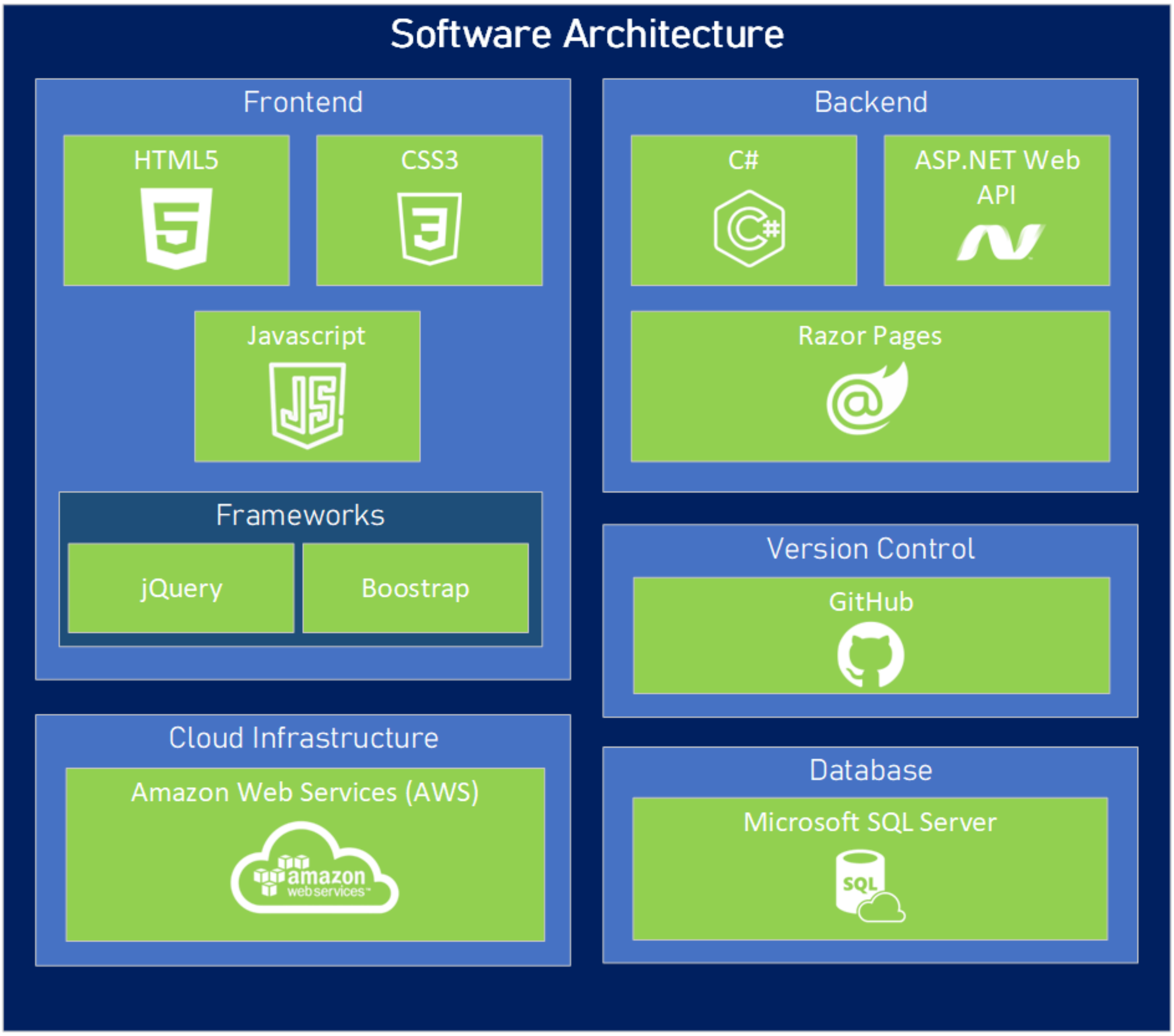


*Figure 10: High level Software architecture.*

The software architecture design includes various technologies for the development of the web system. The technology stack comprises HTML, CSS, and JavaScript for the frontend development of the system, along with the utilisation of the jQuery and Bootstrap frameworks to assist with development. ASP.NET, C#, and Razor Pages will be used for the backend development and server-side technologies' connectivity to the database, which will be an SQL database hosted on AWS. AWS will be used for hosting the web application, including the database. GitHub will be utilized for version control and project collaboration.

## 5.2 High Level Package Diagram and Components

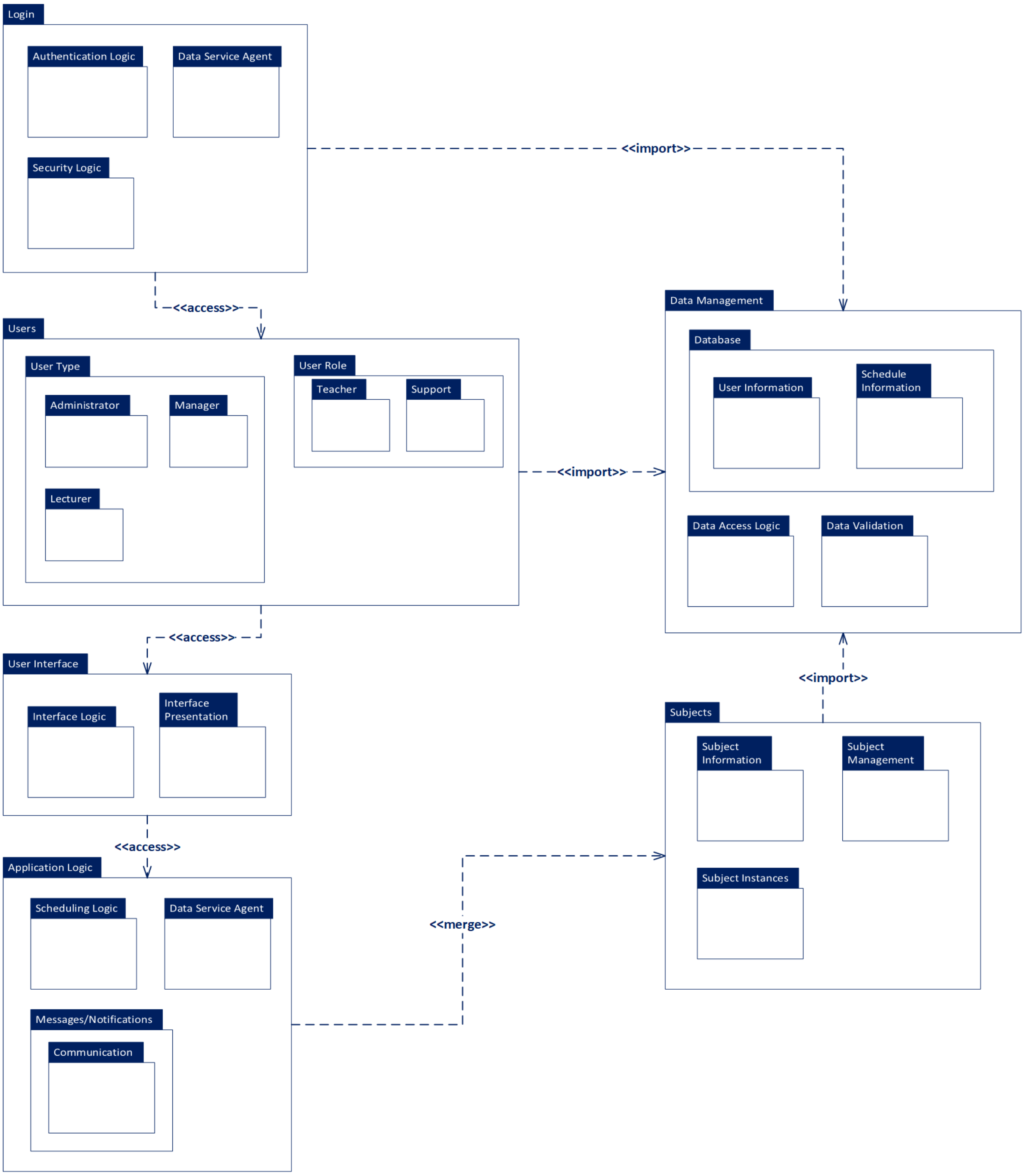


*Figure 11: High level package diagram and components.*

The figure above displays the packages and components used within the web system. Each major component of the system design has been added to provide a high-level view of what makes up the components and how they interact with each other.

## 5.3 Use Case Analysis

| **Name:** | **Schedule Data** |
|---|---|
| **Goal:** | To detail the process involved in efficiently maintaining lecturer work schedules, ensuring accurate and up-to-date scheduling information. |
| **Pre-condition:** | A lecturer schedule exists within the scheduling system. |
| **Post-condition:** | Scheduling data was entered and saved into the system. |
| **Constraints/Issues/Risks:** | • Incomplete schedule data.<br>• User input errors when entering scheduling data.<br>• Inaccurate lecturer preferences.<br>• Inaccurate process for updating scheduling information when changes occur.<br>• System outage.<br>• Poor communication.<br>• Version control if multiple managers attempt to create and/or modify a schedule simultaneously.<br>• Limited accessibility.<br>• Security vulnerabilities.<br>• Integration issues. |
| **Trigger Event(s):** | A schedule is created and/or modified through the system, and the schedule data is recorded. |
| **Primary Actor:** | • Managers.<br>• Lecturers. |
| **Secondary Actor:** | Administrators. |

| **Name:** | **Lecturer Management** |
|---|---|
| **Goal:** | To detail the process of maintaining and managing lecturer preferences. |
| **Pre-condition:** | A lecturer account has been successfully created and entered into the system. |
| **Post-condition:** | Lecturer preference information is accurately entered and recorded into the system. |
| **Constraints/Issues/Risks:** | • System outage.<br>• Unintuitive user interface.<br>• User error when entering information.<br>• Unauthorised access to lecturer information.<br>• Incomplete work schedule preferences.<br>• Data validation errors.<br>• Intended or unintended changes in work schedule preferences.<br>• Insufficient user training.<br>• Limited system scalability.<br>• Poor communication. |
| **Trigger Event(s):** | A lecturer account has been successfully created, and work schedule preferences have been recorded in the system. |
| **Primary Actor:** | • Managers.<br>• Lecturers. |
| **Secondary Actor:** | Administrators. |

| | |
|---|---|
| **Name:** | **Subject Management** |
| **Goal:** | To detail the process of maintaining and managing subject instances. |
| **Pre-condition:** | A subject instance has been successfully created and entered into the system. |
| **Post-condition:** | Subject instance information is accurately entered and recorded into the system. |
| **Constraints/Issues/Risks:** | • System outage.<br>• Unintuitive user interface.<br>• Unauthorised access to subject instance information.<br>• Limited system scalability.<br>• User error when entering subject instance information.<br>• Inaccurate subject instance requirements.<br>• Lecturer feedback.<br>• System limitations.<br>• Unauthorised access to subject instance data. |
| **Trigger Event(s):** | A subject instance has been successfully created and recorded in the system. |
| **Primary Actor:** | Administrators. |
| **Secondary Actor:** | Managers. |

| | |
|---|---|
| **Name:** | **Data Visualisation** |
| **Goal:** | To describe how a lecturer’s work schedule can be displayed visually. |
| **Pre-condition:** | A work schedule has been created and assigned to a lecturer. |
| **Post-condition:** | A lecturer's work schedule can be viewed in the system. |
| **Constraints/Issues/Risks:** | • System outage.<br>• Data validation.<br>• Data collection inconsistencies.<br>• User error when entering scheduling information.<br>• Unauthorised access to scheduling data.<br>• Unintuitive user interface.<br>• Lecturer work preference changes. |
| **Trigger Event(s):** | A lecturer wants to view their work schedule. |
| **Primary Actor:** | • Lecturers.<br>• Managers. |
| **Secondary Actor:** | Administrators. |

*Use Case Diagram*

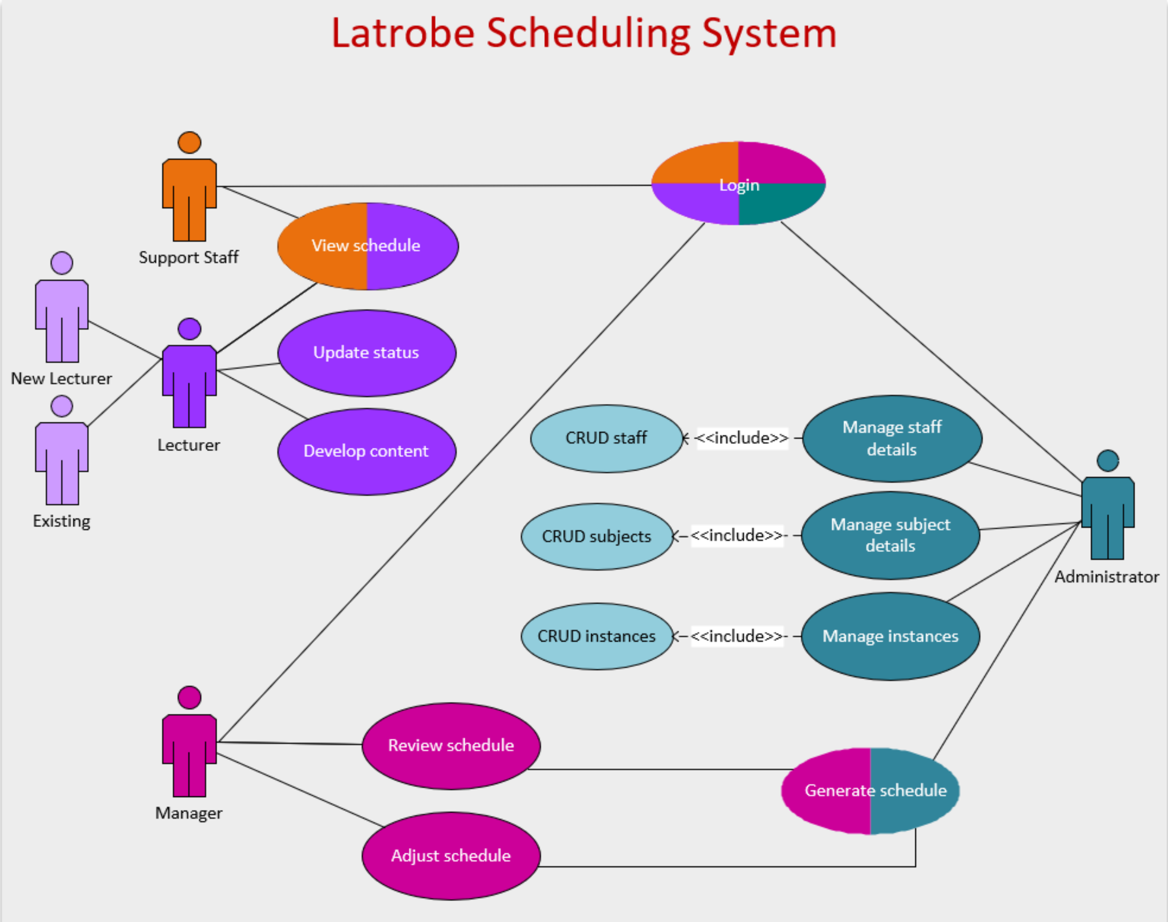


*Figure 12: Use case diagram.*

## 5.4 Domain Model and Class Diagram

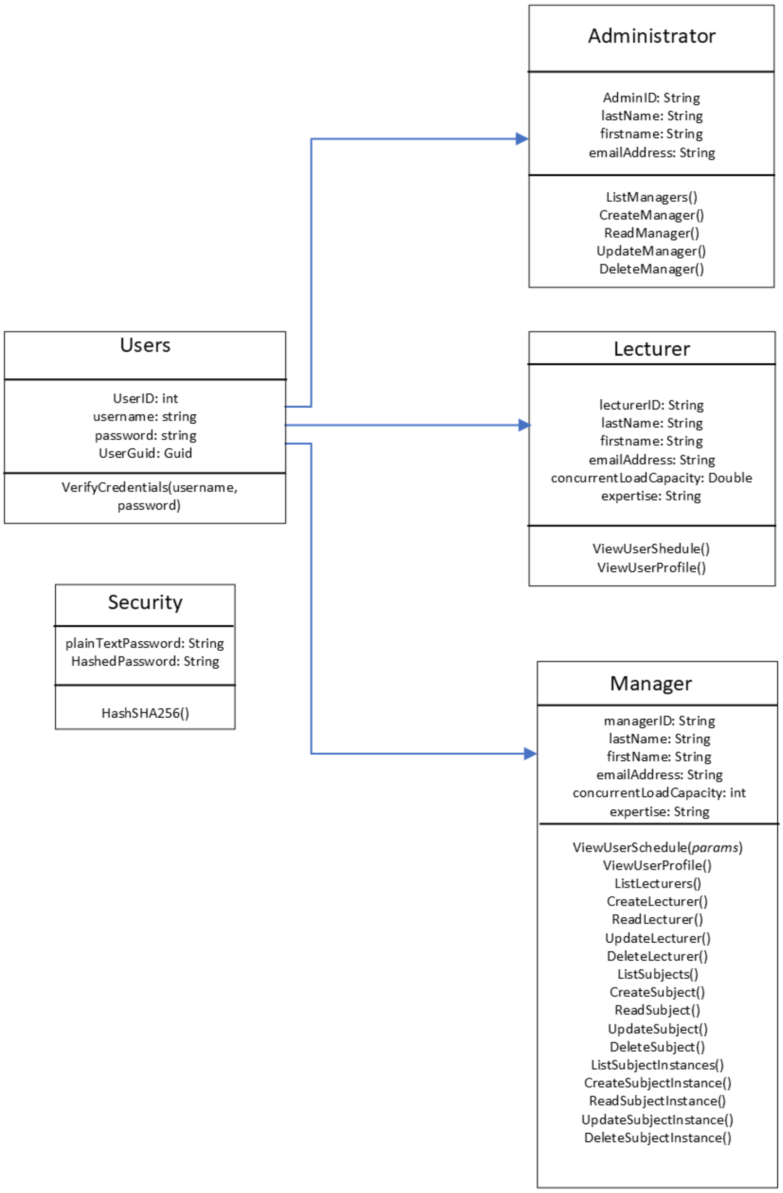


*Figure 13: Class diagram.*

## 5.5 Establishment of the Database Objects and Data Access Strategy

The database connection string will be stored in the appsettings.json file, enabling easy modification without the need to recompile the application.

The connection string format will follow the standard SQL server convention including parameters for server address, database name, user ID and password.

Multiple Classes prefixed with Data Access will be implemented in C# to handle database operations. These classes will use instances of the SQL Connection class to establish connections with the database.

SQL commands will be executed using the SqlCommand object from the System.Data.SqlClient module.

### *Entity Relationship Diagram*

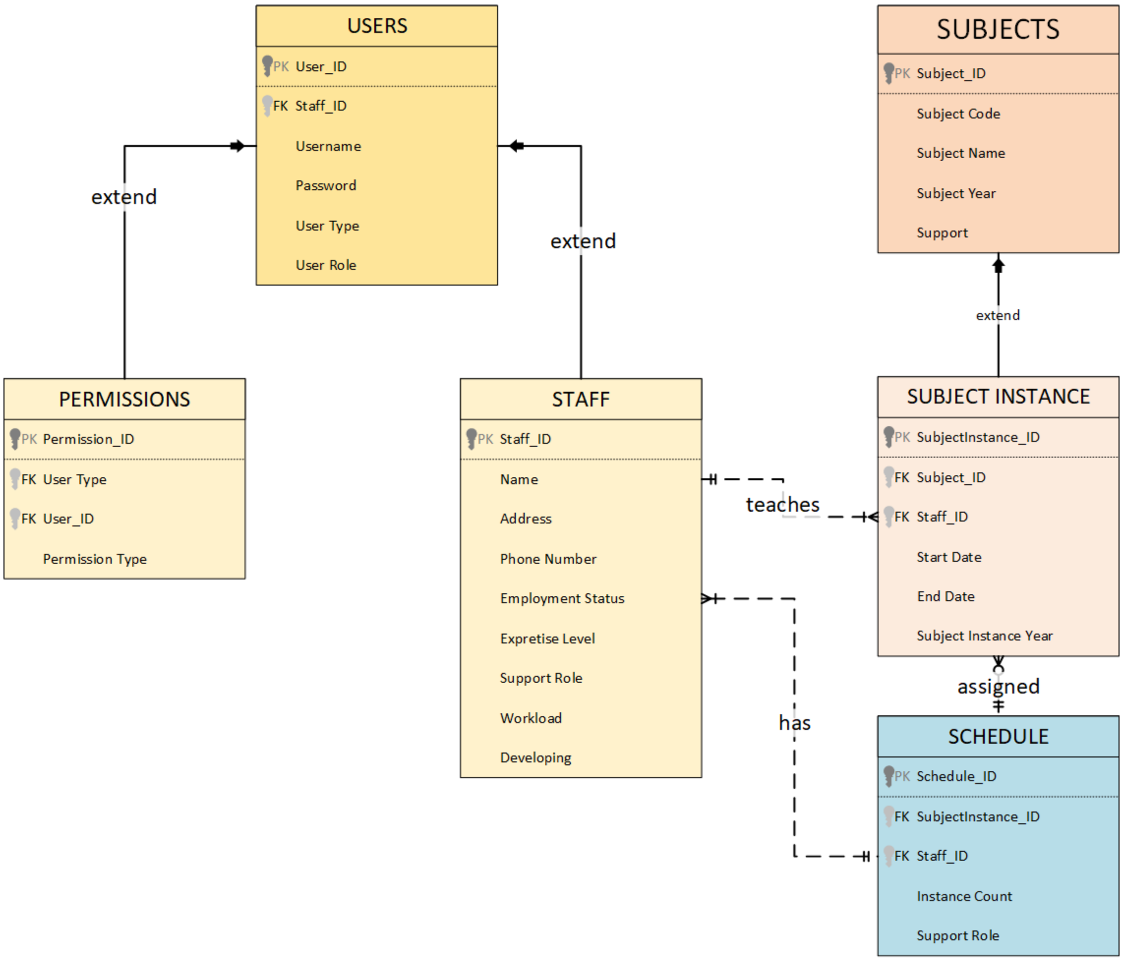


*Figure 14: Entity relationship diagram.*

## 5.6 Sequence Diagram (optional)

TO DO

## 5.7 Object Dictionary

**Users.**

Stores information about users.

| | | |
|---|---|---|
| UserID | INT, Primary Key , Identity | A unique Identifier for each user |
| Email | NVARCHAR(100) | Email Address of the user |
| Password | NVARCHAR(255) | SHA-256 Hashed Password |
| UserGuid | Unique Identifier | Password Salt |
| FirstName | NVARCHAR(50) | First name of the user |
| LastName | NVARCHAR(50) | Last name is the user |
| IsAdmin | Bit, Default 0 | Flag indicating if the user is an Administrator |
| IsManager | Bit, Default 0 | Flag indicating if the user is a manager |
| IsLecturer | Bit, Default 0 | Flag indicating if the user is a Lecturer |
| IsPasswordResetRequired | Bit, Default 0 | Flag indicating if the user must reset their password on their next log in |
| CreatedOn | DateTime. Default GETDATE() | Date and time that the user was created on. |

**Password Resets.**

Manages Password reset Requests.

| | | |
|---|---|---|
| Email | NVARCHAR(150) Primary Key | Email address associated with the password reset request. |
| Token | NVARCHAR(150) | Token generated for the password reset process. |
| CreatedOn | DateTime. Default GETDATE() | The date and time the reset request was created. |

**Lecturers.**

Containers lecturer specific information about users when required.

| | | |
|---|---|---|
| LecturersID | INT Identity | Unique identifier for lecturers. |
| UserID | INT | Foreign Key References UserID from Users |
| Expertise01 | NVARCHAR(255) | Stores lecturer assigned expertise. |
| Expertise02 | NVARCHAR(255) | Stores lecturer assigned expertise. |
| Expertise03 | NVARCHAR(255) | Stores lecturer assigned expertise. |
| Expertise04 | NVARCHAR(255) | Stores lecturer assigned expertise. |
| Expertise05 | NVARCHAR(255) | Stores lecturer assigned expertise. |
| Expertise06 | NVARCHAR(255) | Stores lecturer assigned expertise. |
| ConncurrentLoadCapacity | DECIMAL | Represents the lecturer's capacity to handle concurrent subject instances. |

**Subjects.**

Details about subjects offered.

| | | |
|---|---|---|
| SubjectID | INT, Identity | Unique identifier for subject. |
| SubjectCode | NVARCHAR(255) | Code to reference subject. |
| SubjectName | NVARCHAR(255) | Name of subject |
| SubjectClassification | NVARCHAR(255) | Category of subject. |
| YearLevel | INT | Academic year level of students required for this subject |
| DevelopmentDifficulty | NVARCHAR(255) | Indicates the difficulty in developing the subject’s content. |

**Subject Instances.**

Instances of subjects being offered.

| | | |
|---|---|---|
| SubjectInstanceID | INT, Identity | Unique identifier for each subject instance |
| SubjectID | INT, Foreign Key | References SubjectID from the subject’s table. |
| SubjectInstanceName | NVARCHAR(255) | Unique name of for the specific instance of the subject |
| SubjectInstanceCode | NVARCHAR(255) | Short code for specific instance of the subject. |
| LecturerID | INT | Identifies the lecturer responsible for the subject instance |
| StartDate | DATETIME | Start date for subject instance |
| EndDate | DATETIME | End date for subject Instance. |

# 6.0 Software Release Report

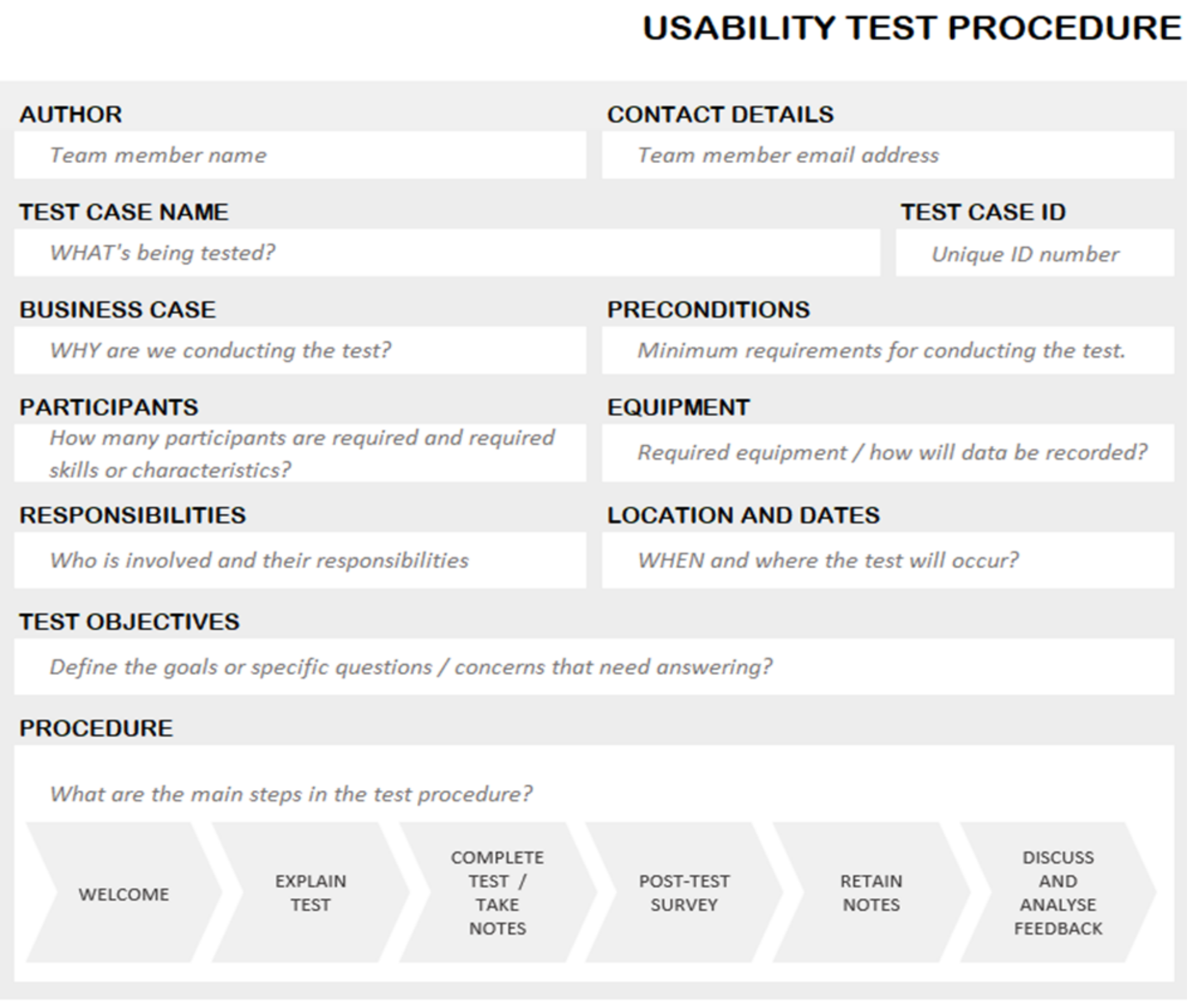
Usability testing is an iterative and systematic approach used to evaluate product usability and overall user experience by observing users interacting with software to:

- Gather insights,
- Identify usability issues and defaults,
- Understand user behaviour, and
- Determine areas for improvement.

## 6.1 Usability Test Report

Usability testing validates and verifies product functionality and examines user behaviour and their interaction with the system.

The following procedure will be moderated using an explorative approach (including open-ended questions) and used to provide consistency, direction, and a systematic approach when developing and conducting usability testing.



**USABILITY TEST PROCEDURE**

**AUTHOR**
*Team member name*

**CONTACT DETAILS**
*Team member email address*

**TEST CASE NAME**
*WHAT's being tested?*

**TEST CASE ID**
*Unique ID number*

**BUSINESS CASE**
*WHY are we conducting the test?*

**PRECONDITIONS**
*Minimum requirements for conducting the test.*

**PARTICIPANTS**
*How many participants are required and required skills or characteristics?*

**EQUIPMENT**
*Required equipment / how will data be recorded?*

**RESPONSIBILITIES**
*Who is involved and their responsibilities*

**LOCATION AND DATES**
*WHEN and where the test will occur?*

**TEST OBJECTIVES**
*Define the goals or specific questions / concerns that need answering?*

**PROCEDURE**
*What are the main steps in the test procedure?*

WELCOME
EXPLAIN TEST
COMPLETE TEST / TAKE NOTES
POST-TEST SURVEY
RETAIN NOTES
DISCUSS AND ANALYSE FEEDBACK

*Figure 15:Usability test procedure.*

*PROCEDURE NOTES*

### 1. WELCOME

Welcome the participants and conduct pre-test interviews to understand their level of experience / address any concerns or questions. NDAs are not applicable for this scenario as it is for educational and training purposes.

### 2. EXPLAIN TEST

Provide an introduction, outlining the purpose and test procedure and instruct the participants to ***think aloud*** (and be open and honest) as they perform the assigned tasks.

### 3. COMPLETE TEST

Conduct the test. The moderator must observe participants, noting all system interactions, comments and difficulties encountered by the participants. All notes must be recorded in the "Usability Observation Checklist." Please see below.

### 4. POST – TEST SURVEY

Complete the post-test survey, allowing participants to share overall impressions, thoughts and record any additional feedback in the comments section of the "Usability Observation Checklist."

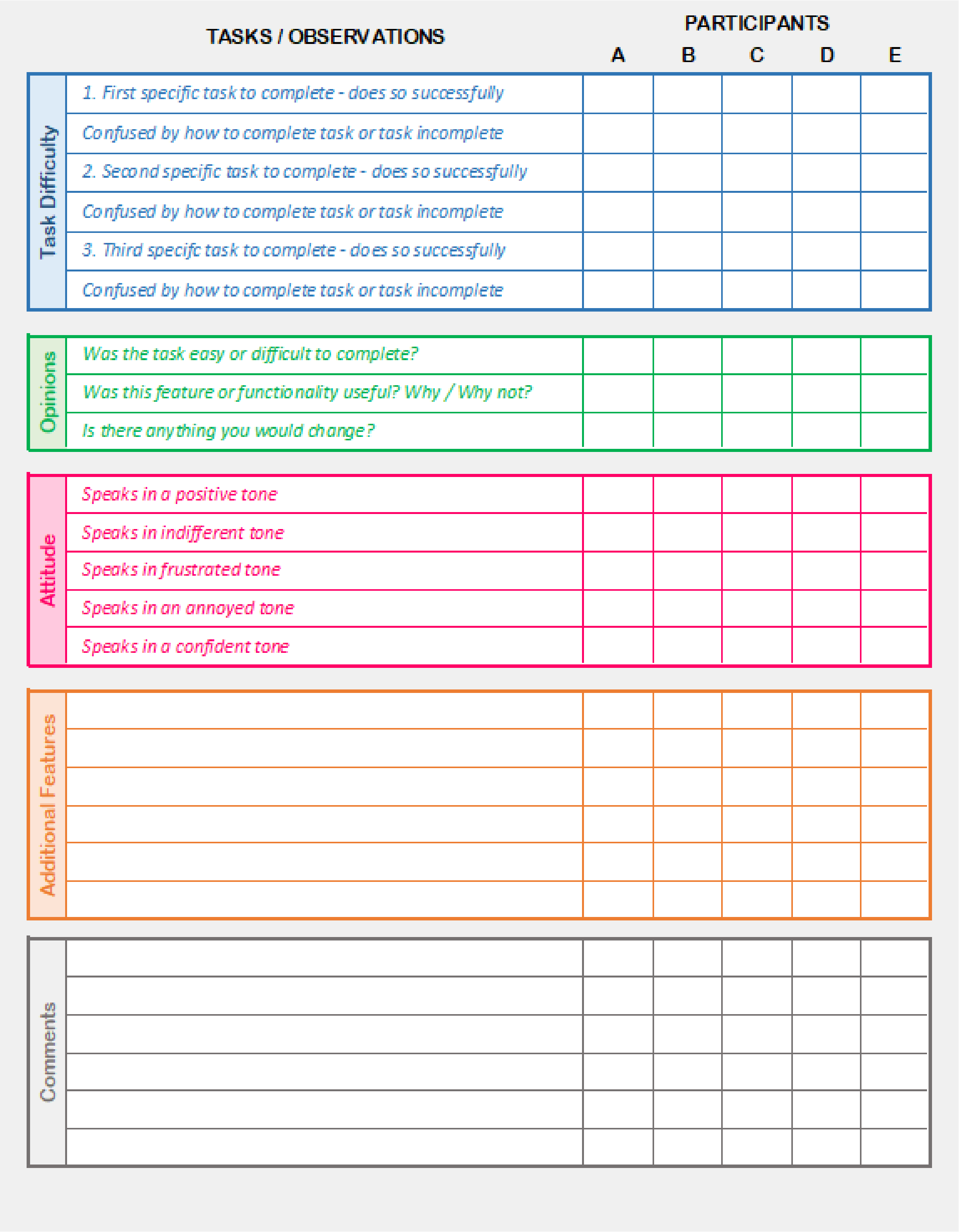
### 5. COLLECT DATA

Collect both qualitative (including observations and participant feedback) and quantitative including successful task completion and time taken.

### 6. DISCUSS AND ANALYSE FEEDBACK

The team will review and analyse collected data to identify patterns or common usability issues that emerged during the testing phase. Once these issues are identified we will categories and priorities issues based on severity and impact to user experience.

## Usability Task Observation Checklist



| | TASKS / OBSERVATIONS | PARTICIPANTS | | | | |
|---|---|---|---|---|---|---|
| | | A | B | C | D | E |
| Task Difficulty | *1. First specific task to complete - does so successfully* | | | | | |
| | *Confused by how to complete task or task incomplete* | | | | | |
| | *2. Second specific task to complete - does so successfully* | | | | | |
| | *Confused by how to complete task or task incomplete* | | | | | |
| | *3. Third specifc task to complete - does so successfully* | | | | | |
| | *Confused by how to complete task or task incomplete* | | | | | |
| Opinions | *Was the task easy or difficult to complete?* | | | | | |
| | *Was this feature or functionality useful? Why / Why not?* | | | | | |
| | *Is there anything you would change?* | | | | | |
| Attitude | *Speaks in a positive tone* | | | | | |
| | *Speaks in indifferent tone* | | | | | |
| | *Speaks in frustrated tone* | | | | | |
| | *Speaks in an annoyed tone* | | | | | |
| | *Speaks in a confident tone* | | | | | |
| Additional Features | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| Comments | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

*Figure 16: Usability test observation checklist.*

*USABILITY OBSERVATION CHECKLIST NOTES*

Enter “1” in the appropriate box when the statement is true. Leave the box empty when the statement is false.

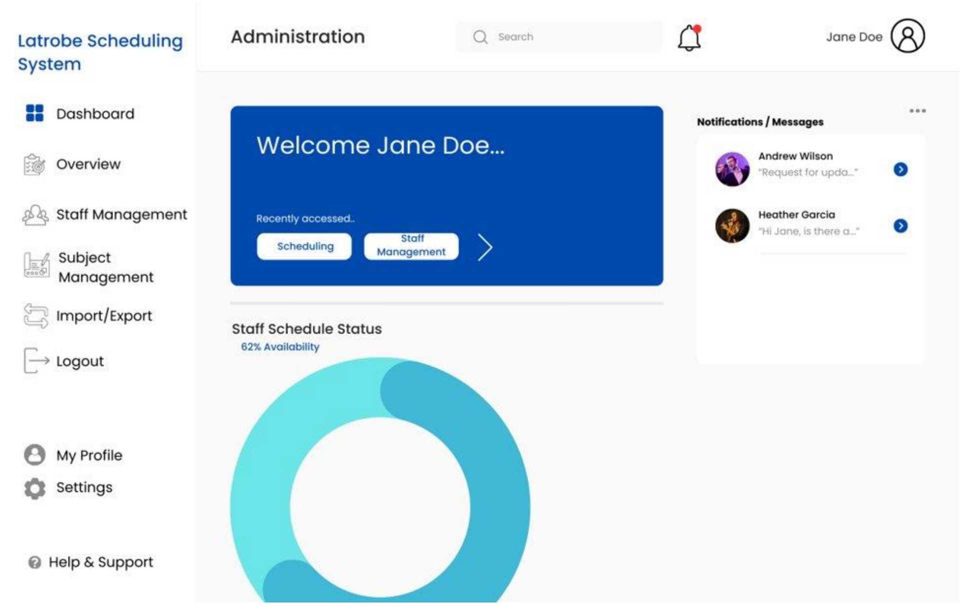
*[To be completed]* - usability testing will be conducted at a later date – including modifications to be made.

## 6.2 System (User Story) Test Report

*[To be completed]* – system testing will be conducted at a later date.

# 7.0 Additional User Interface Design

*User Interfaces*

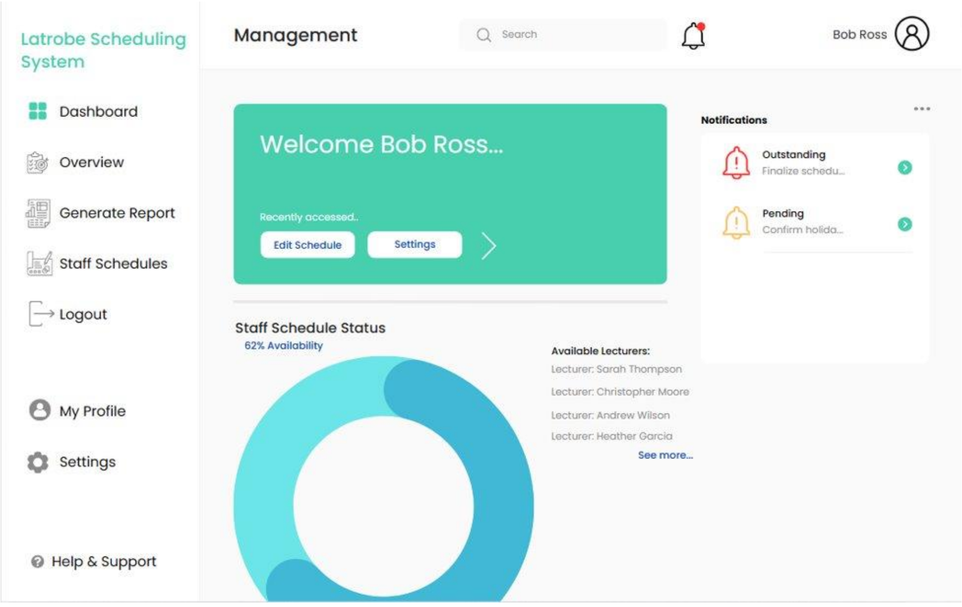


Welcome Jane Doe...

Andrew Wilson "Request for upda..."

Heather Garcia "Hi Jane, is there a..."

*Figure 17: Administration interface.*



Welcome Bob Ross...

Outstanding Finalize schedu...

Pending Confirm holida...

Available Lecturers:

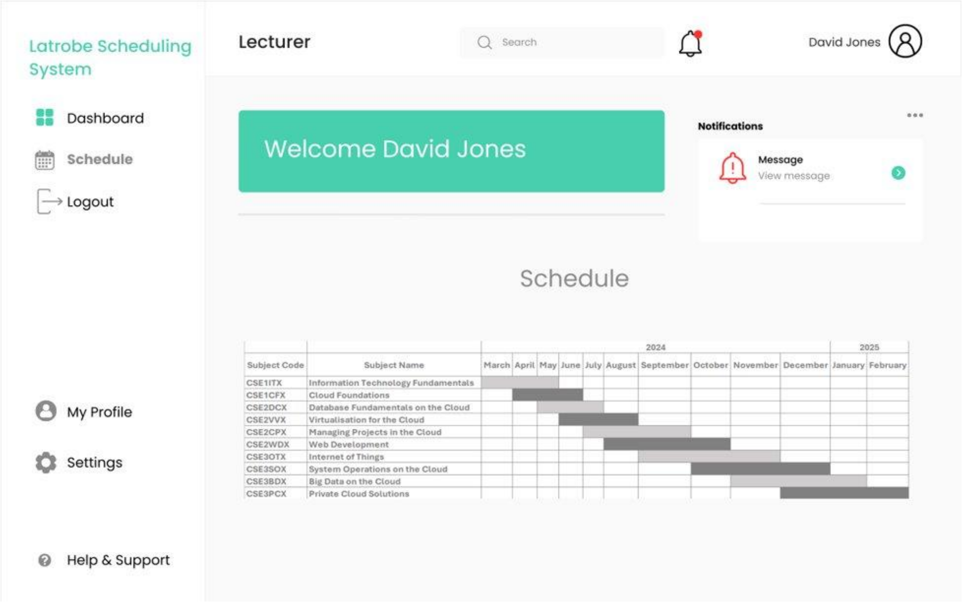
Lecturer: Sarah Thompson

Lecturer: Christopher Moore

Lecturer: Andrew Wilson

Lecturer: Heather Garcia

*Figure 18: Manager interface.*



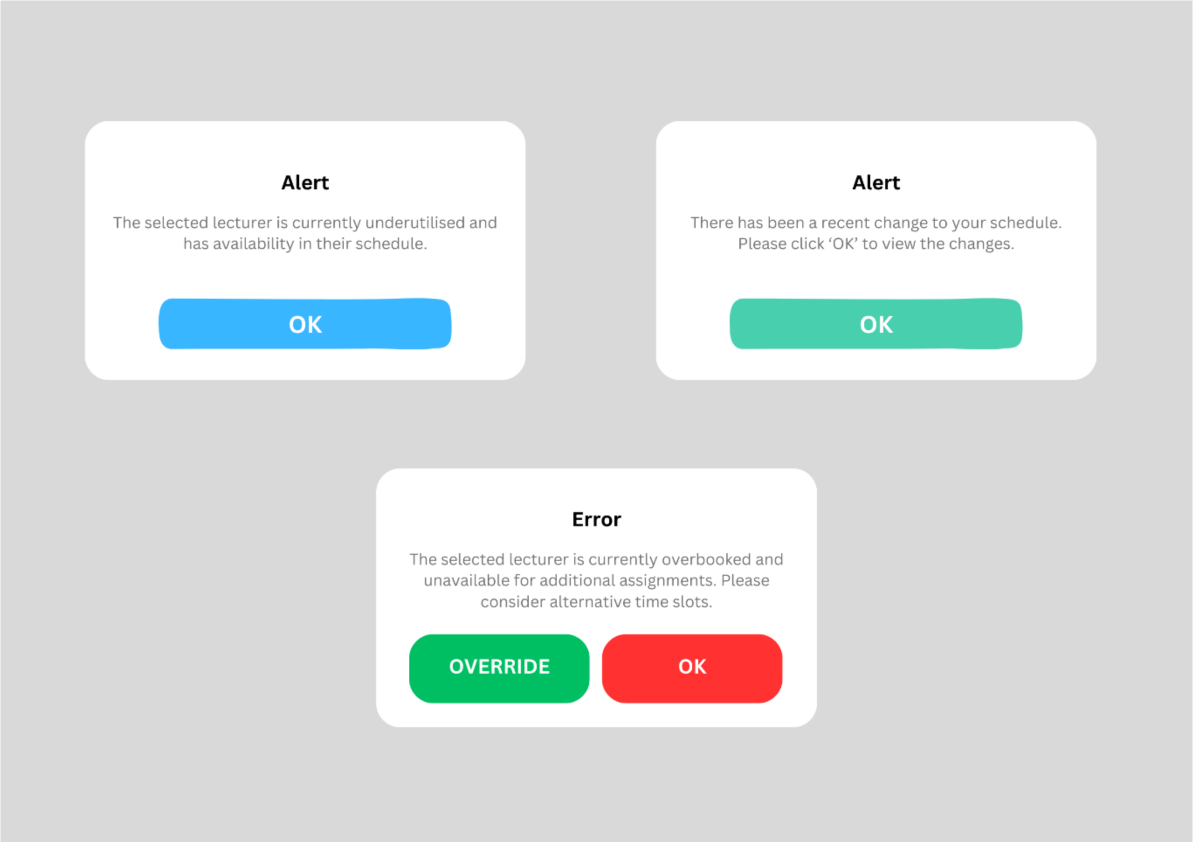
Welcome David Jones

Schedule

| | | 2024 | | | | | | | | | | 2025 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subject Code | Subject Name | March | April | May | June | July | August | September | October | November | December | January | February |
| CSE1ITX | Information Technology Fundamentals | ▨ | ▨ | ▨ | | | | | | | | | |
| CSE1CFX | Cloud Foundations | | ■ | ■ | ■ | | | | | | | | |
| CSE2DCX | Database Fundamentals on the Cloud | | | ▨ | ▨ | ▨ | | | | | | | |
| CSE2VVX | Virtualisation for the Cloud | | | | ■ | ■ | ■ | | | | | | |
| CSE2CPX | Managing Projects in the Cloud | | | | | ▨ | ▨ | ▨ | | | | | |
| CSE2WDX | Web Development | | | | | | ■ | ■ | ■ | | | | |
| CSE3OTX | Internet of Things | | | | | | | ▨ | ▨ | ▨ | | | |
| CSE3SOX | System Operations on the Cloud | | | | | | | | ■ | ■ | ■ | | |
| CSE3BDX | Big Data on the Cloud | | | | | | | | | ▨ | ▨ | ▨ | |
| CSE3PCX | Private Cloud Solutions | | | | | | | | | | ■ | ■ | ■ |

*Figure 19: Lecturer interface.*

## *Alerts & Error Messages*



**Alert**

The selected lecturer is currently underutilised and has availability in their schedule.

**Alert**

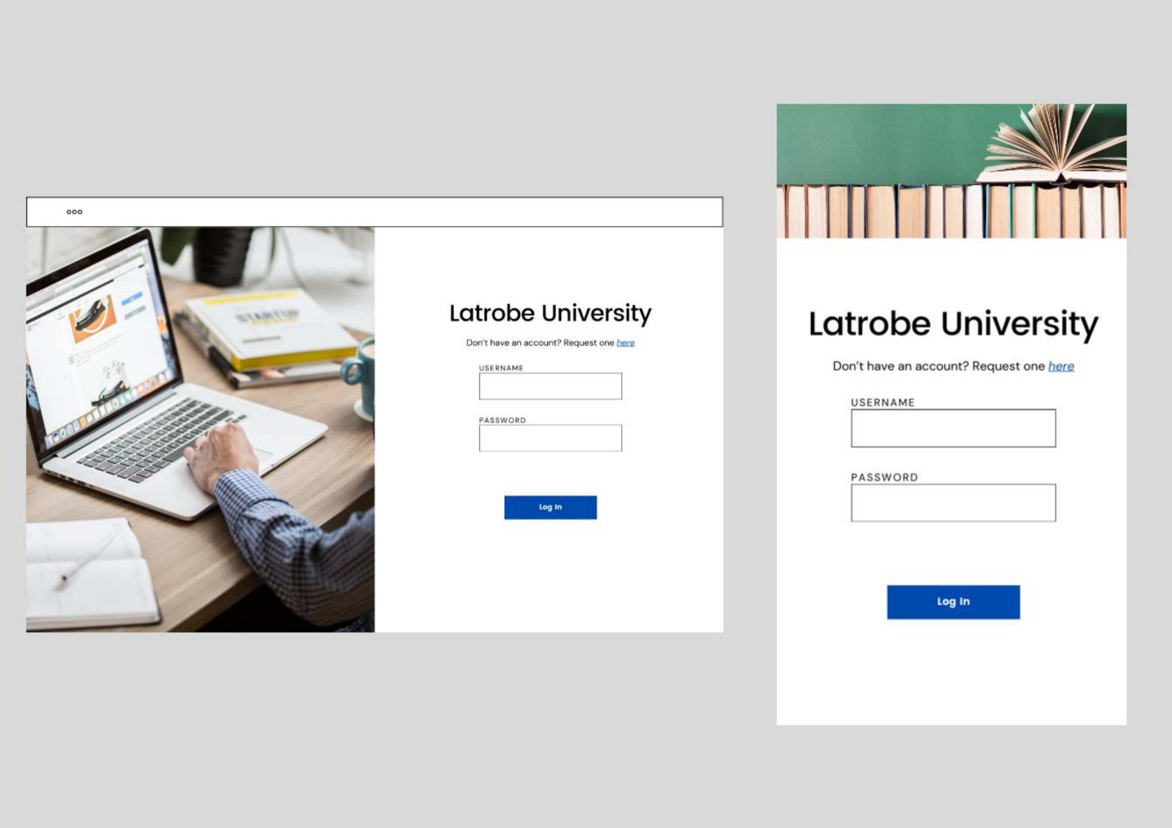
There has been a recent change to your schedule. Please click 'OK' to view the changes.

**Error**

The selected lecturer is currently overbooked and unavailable for additional assignments. Please consider alternative time slots.

*Figure 20: Notification and error messages.*

## Desktop & Mobile Login Interface



Latrobe University

Don't have an account? Request one here

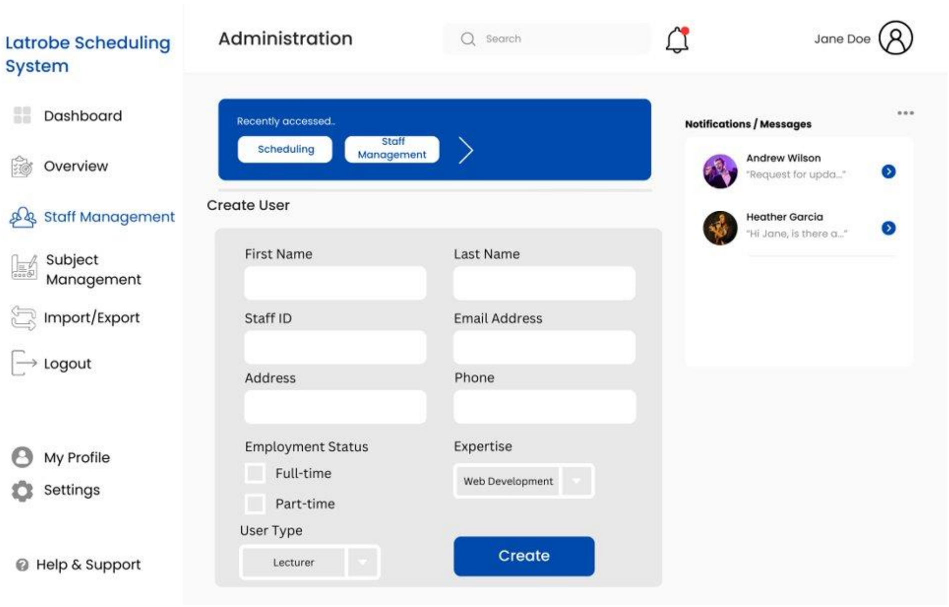
Latrobe University

Don't have an account? Request one here

*Figure 21: Login page.*

## 7.1 Additional Input and Output Screens

### *User Management*



Notifications / Messages

Andrew Wilson "Request for upda..."

Heather Garcia "Hi Jane, is there a..."

Create User

First Name

Last Name

Staff ID

Email Address

Address

Phone

Employment Status

□ Full-time

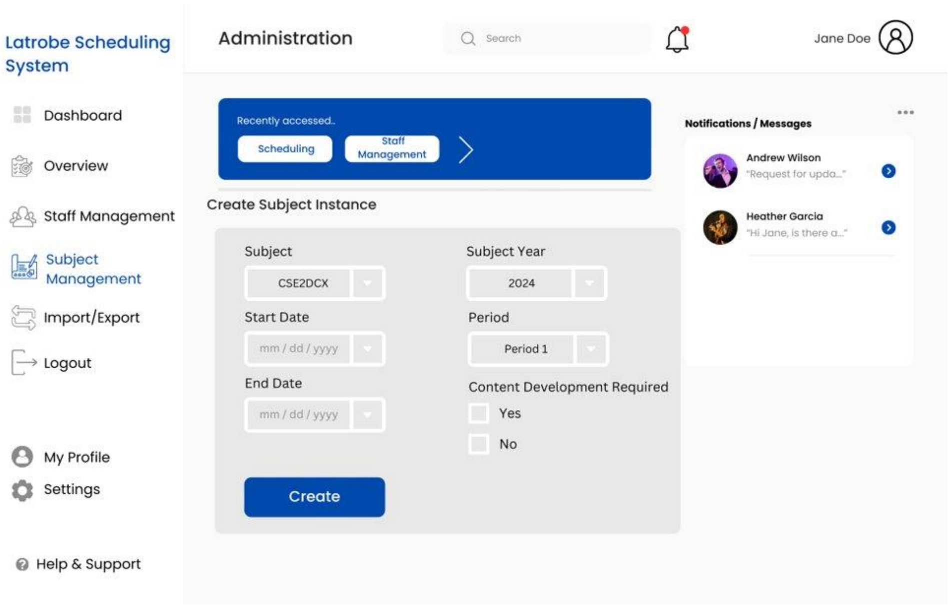
□ Part-time

Expertise: Web Development

User Type: Lecturer

*Figure 22: Admin - create user.*

### *Subject Management*



Notifications / Messages

Andrew Wilson "Request for upda..."

Heather Garcia "Hi Jane, is there a..."

Create Subject Instance

Subject: CSE2DCX

Subject Year: 2024

Start Date: mm / dd / yyyy

Period: Period 1

End Date: mm / dd / yyyy

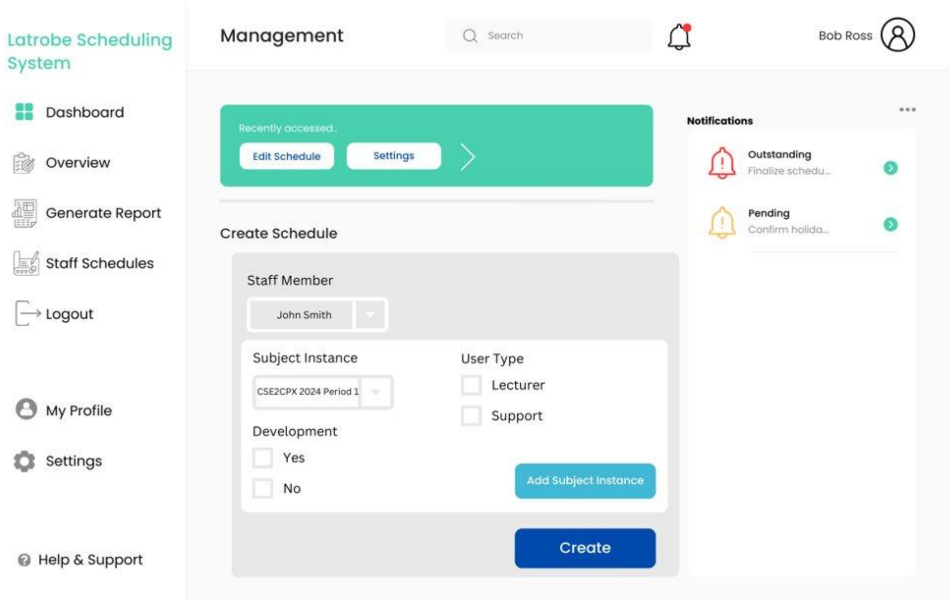
Content Development Required

□ Yes

□ No

*Figure 23: Admin - create subject.*

## *Schedule Management*



Create Schedule

Staff Member

John Smith

Subject Instance

CSE2CPX 2024 Period 1

User Type

□ Lecturer

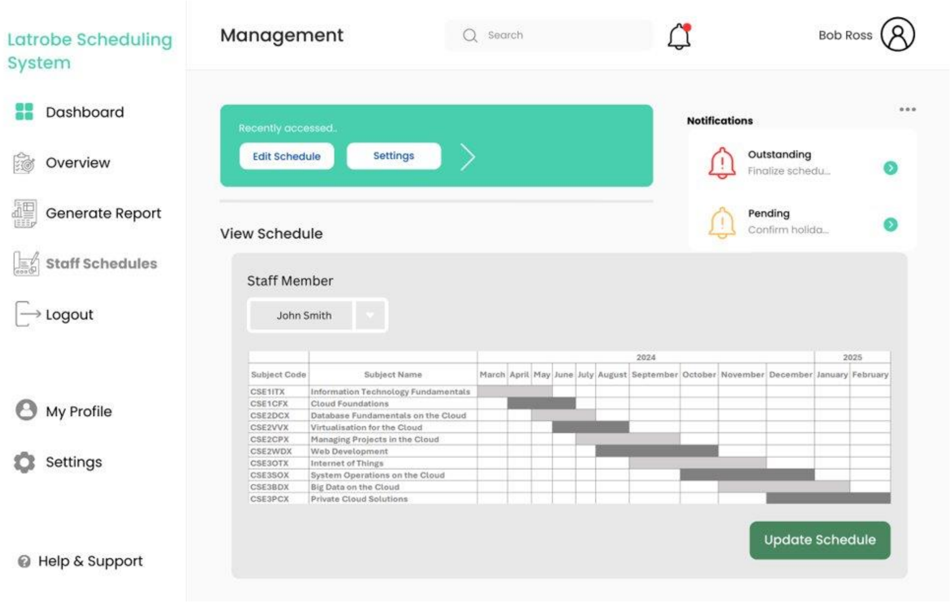
□ Support

Development

□ Yes

□ No

*Figure 24: Manager - create schedule.*



View Schedule

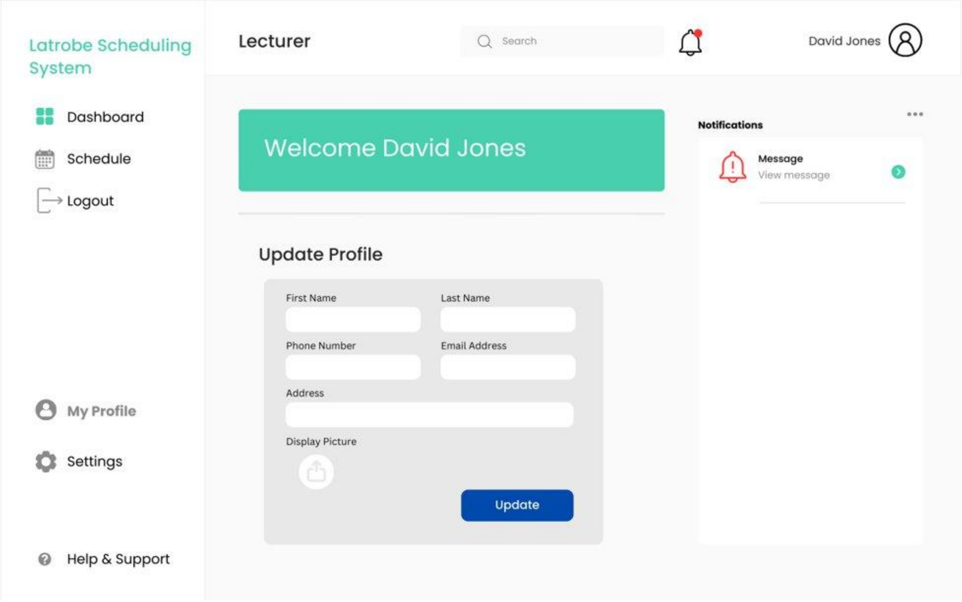
Staff Member

John Smith

| | | 2024 | | | | | | | | | | 2025 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subject Code | Subject Name | March | April | May | June | July | August | September | October | November | December | January | February |
| CSE1ITX | Information Technology Fundamentals | ░ | ░ | ░ | | | | | | | | | |
| CSE1CFX | Cloud Foundations | | ▓ | ▓ | ▓ | | | | | | | | |
| CSE2DCX | Database Fundamentals on the Cloud | | | ░ | ░ | ░ | | | | | | | |
| CSE2VVX | Virtualisation for the Cloud | | | | ▓ | ▓ | ▓ | | | | | | |
| CSE2CPX | Managing Projects in the Cloud | | | | | ░ | ░ | ░ | | | | | |
| CSE2WDX | Web Development | | | | | | ▓ | ▓ | ▓ | | | | |
| CSE3OTX | Internet of Things | | | | | | | ░ | ░ | ░ | | | |
| CSE3SOX | System Operations on the Cloud | | | | | | | | ▓ | ▓ | ▓ | | |
| CSE3BDX | Big Data on the Cloud | | | | | | | | | ░ | ░ | ░ | |
| CSE3PCX | Private Cloud Solutions | | | | | | | | | | ▓ | ▓ | ▓ |

*Figure 25: Manager - view schedule.*

*User Profile*



Welcome David Jones

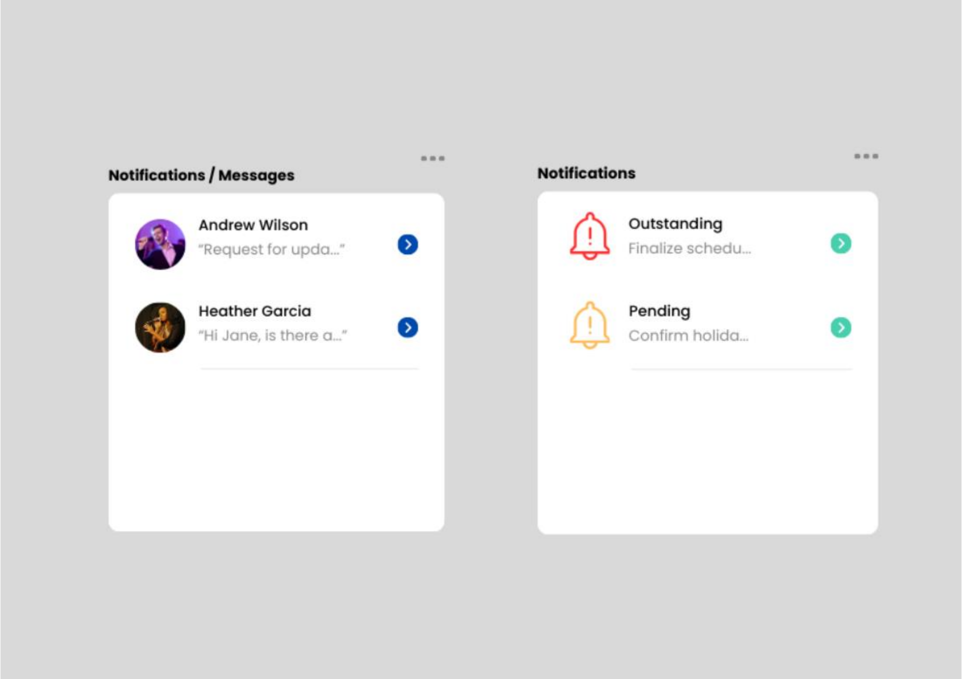
Update Profile

First Name
Last Name
Phone Number
Email Address
Address

*Figure 26: Lecturer - update profile.*

## 7.2 Additional Data Display Screens

### *Notifications & Messages*



Andrew Wilson
"Request for upda..."

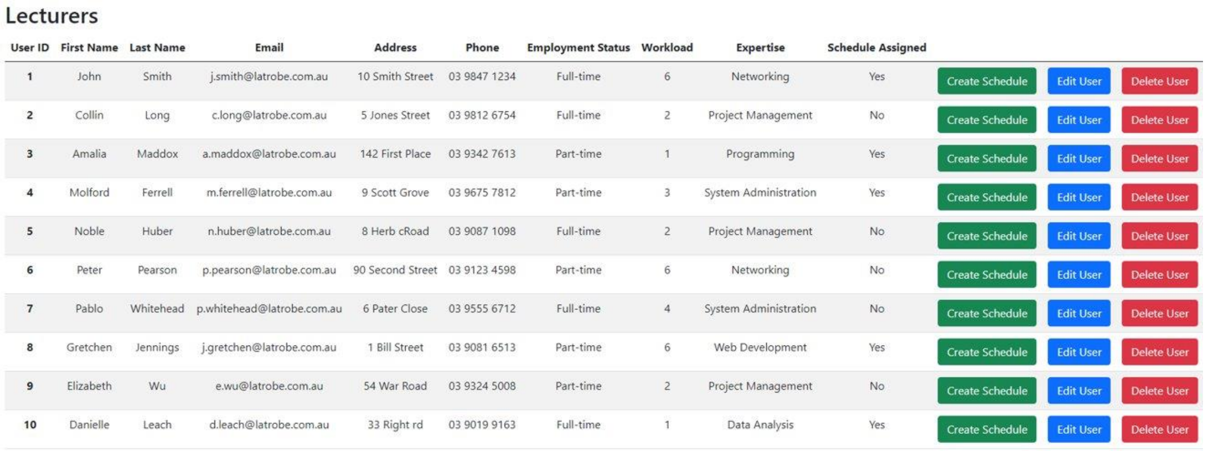
Heather Garcia
"Hi Jane, is there a..."

Outstanding
Finalize schedu...

Pending
Confirm holida...

*Figure 27: Notifications & messages.*

## User Management



### Lecturers

| User ID | First Name | Last Name | Email | Address | Phone | Employment Status | Workload | Expertise | Schedule Assigned | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | John | Smith | j.smith@latrobe.com.au | 10 Smith Street | 03 9847 1234 | Full-time | 6 | Networking | Yes | Create Schedule | Edit User | Delete User |
| 2 | Collin | Long | c.long@latrobe.com.au | 5 Jones Street | 03 9812 6754 | Full-time | 2 | Project Management | No | Create Schedule | Edit User | Delete User |
| 3 | Amalia | Maddox | a.maddox@latrobe.com.au | 142 First Place | 03 9342 7613 | Part-time | 1 | Programming | Yes | Create Schedule | Edit User | Delete User |
| 4 | Molford | Ferrell | m.ferrell@latrobe.com.au | 9 Scott Grove | 03 9675 7812 | Part-time | 3 | System Administration | Yes | Create Schedule | Edit User | Delete User |
| 5 | Noble | Huber | n.huber@latrobe.com.au | 8 Herb cRoad | 03 9087 1098 | Full-time | 2 | Project Management | No | Create Schedule | Edit User | Delete User |
| 6 | Peter | Pearson | p.pearson@latrobe.com.au | 90 Second Street | 03 9123 4598 | Part-time | 6 | Networking | No | Create Schedule | Edit User | Delete User |
| 7 | Pablo | Whitehead | p.whitehead@latrobe.com.au | 6 Pater Close | 03 9555 6712 | Full-time | 4 | System Administration | No | Create Schedule | Edit User | Delete User |
| 8 | Gretchen | Jennings | j.gretchen@latrobe.com.au | 1 Bill Street | 03 9081 6513 | Part-time | 6 | Web Development | Yes | Create Schedule | Edit User | Delete User |
| 9 | Elizabeth | Wu | e.wu@latrobe.com.au | 54 War Road | 03 9324 5008 | Part-time | 2 | Project Management | No | Create Schedule | Edit User | Delete User |
| 10 | Danielle | Leach | d.leach@latrobe.com.au | 33 Right rd | 03 9019 9163 | Full-time | 1 | Data Analysis | Yes | Create Schedule | Edit User | Delete User |

*Figure 28: Lecturer - view schedule.*

# 8.0 Special Notes

## 8.1 Expected System Behaviour

**Temporary Downtime**: The Latrobe Scheduling System may experience temporary unavailability during maintenance activities and downtime.

**New Features**: The system may introduce at times new features and functionalities to enhance the Latrobe system capabilities.

**Security Updates**: Security or system updates may be implemented to address vulnerabilities towards the systems security.

**Data Archiving**: Archiving of outdated information may occur for optimisation performance.

**Compliance Adjustments**: System modifications may occur due to compliance with relevant regulations and standards.

## 8.2 Warnings & Risks

| **Risk** | **Precaution** |
|---|---|
| Data Loss | Backup of data prior to initiation of procedures |
| Downtime | Communicate maintenance schedule in advance |
| Security | Follow best practices and conduct security assessments |
| User Access Issues | Document user roles and permissions and ensure correct configurations during maintenance |
| Overloading of Resources | Monitor resources during system maintenance, and conduct maintenance during periods of high and low activity. |
| Inadequate Documentation | Maintain up-to-date documents on all procedures |
| Insufficient Testing | Create complete testing scenarios to identify all potential issues within the system and environment. |
| Change of Management | Ensure plans are complete for management processes and all team members are aware of any changes. |
| Lack of Logging | Ensure monitoring and logging mechanisms and monitor all system behaviour for comparison during planned maintenance periods. |

## 8.3 Contact Information

For any technical issues, questions, or support regarding the Latrobe Scheduling System and the system maintenance documentation, please contact our Technical Support Team or System Administrators.

### *Technical Support Team*

- **Email**: technicalsupport@latrobe.com.au
- **Phone**: 03 1234 1234

### *System Administrators*

- **Email**: systemadmin@latrobe.com.au
- **Phone**: 03 5678 5678

## Appendices

*[Not applicable at this time]*

## Bibliography

*[Not applicable at this time – All material used throughout this document have been mentioned in references]*

## Glossary of Terms

Listed alphabetically, the glossary features abbreviations and relevant terms for this project.

| Term | Abbreviation | Definition |
|---|---|---|
| Active Server Pages | ASP | Scripting technology for creating web pages and applications. |
| Amazon Web Servers | AWS | Cloud computing platform. |
| Application Programming Interface | API | Set of rules and protocols. |
| C Sharp Programming Language | C# | A style sheet language for creating applications and web services. |
| Cascading Style Sheets | CSS | A language used to describe visual representations. |
| Create, Read, Update and Delete | CRUD | Operations used for database representation. |
| European Computer Manufacturers Association | ECMAScript | A standard scripting language specification. |
| Entity Relationship Diagram | ERD | Visual representation of logical structures and relationships. |
| Hypertext Markup Language | HTML | A language used for creating web pages and documents. |
| Identity and Access Management | IAM | Framework of AWS used for managing resources. |
| Microsoft Structured Query Language Server | MSSQL | Relational database management system. |
| Non-Disclosure Agreement | NDA | Legally binding contract outlining confidential information. |
| Role-Based Access Control | RBAC | Principles of managing access to resources. |
| Structured Query Language | SQL | A programming language useful in database management. |
| Unified Modelling Language | UML | A modelling language used for visually representing software systems. |
| User Interface | UI | An application or system interface with graphical elements for users. |
| User Story Dictionary | USD | A document used for defining requirements, features, and functionalities from a user's view. |
| World Wide Web Consortium | W3C | Standards and guidelines developed by communities. |

## Reference Materials

- *CANVA: Visual Suite for everyone* (no date). Available at: https://www.canva.com/ (Accessed: 24 December 2023)
- Legal Vision: *Software and Website Development Agreement*. Available at: https://legalvision.com.au/ (Accessed: 20 December 2023)
- Usability testing: *Usability Testing 101.* Available at: https://www.nngroup.com/ (Accessed: 3 January 2024)
- Rolyon (no date) *Microsoft Entra ID Documentation – Microsoft Entra ID | Microsoft Learn*. Available at: https://learn.microsoft.com/en-us/entra/identity/ (Accessed December 2023)
- *Azure SQL documentation – Azure SQL* (no date) *Azure SQL | Microsoft Learn.* Available at: https://learn.microsoft.com/en-us/azure-sql/?view=azuresql (Accessed: December 2023)
- Shirgolbird (2024). *Microsoft SQL Documentation – SQL Server | Microsoft Learn*. Available at: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16 (Accessed December 2023)
- *Welcome to AWS Documentation* (no date). *AWS Documentation.* Available at: https://docs.aws.amazon.com/ (Accessed: December 2023)
- *ASP.NET Documentation* (no date). *Microsoft Learn.* Available at: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0 (Accessed: December 2023)
- Hickson, I. (2011). *A vocabulary and associated APIs for HTML and XHTML, HTML5.* Available at: https://www.w3.org/TR/2011/WD-html5-20110405/ (Accessed: December 2023)
- Bos, B. (2024) *Cascading Style Sheets, W3C.* Available at: https://www.w3.org/Style/CSS/Overview.en.html (Accessed: January 2024)
- *MozDevNet* (2023) *JavaScript: MDN, MDN Web Docs.* Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript (Accessed: January 2024)
- *JQuery API* (no date) *jQuery API Documentation.* Available at: https://api.jquery.com/ (Accessed: January 2024)
- *Bootstrap* (no date) *Introduction – Bootstrap.* Available at: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (Accessed: January 2024)
- *Microsoft Learn* (no date) *C# docs – getting started, tutorials, reference | Microsoft Learn.* Available at: https://learn.microsoft.com/en-us/dotnet/csharp/ (Accessed: January 2024)
- *GitHub.com help documentation* (no date) *GitHub Docs.* Available at: https://docs.github.com/en (Accessed: January 2024)
- Anderson, R. Brock, D. and Larkin, K. (2023) *Introduction to Razor Pages in ASP.NET Core, Microsoft Learn.* Available at: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-8.0&tabs=visual-studio (Accessed: January 2024)

# Index

## Statement of Effort

Within this statement of effort, each section of the document is defined, accompanied by a list of team members accountable for its production.

| Document Section | Personnel Responsible |
|---|---|
| Copyright, warranties & contractual obligations | Chareen Elton |
| 1.0 Introduction | Laura Searle |
| 1.1 System Overview | Laura Searle |
| 1.2 Audience Description | Alexx Verdichizzi |
| 1.3 Applicability Statement | Alexx Verdichizzi |
| 1.4 Purpose of Statement | Laura Searle |
| 1.5 Document Usage Description | Laura Searle |
| 1.6 Conventions | Laura Searle / Chareen Elton |
| 1.7 Change Logs | *(not yet required)* |
| 2.0 Software Design Scope | Laura Searle |
| 2.1 Major Software Functions | Laura Searle |
| 2.2 Major Design Constraints & Other Requirements | Chareen Elton |
| 3.0 Reference Documents | Laura Searle |
| 3.1 Existing Software Documentation | Alexx Verdichizzi |
| 3.2 System Documentation | Alexx Verdichizzi / Michael Little |
| 3.3 Vendor Documentation | Alexx Verdichizzi |
| 3.4 Other Documentation | Alexx Verdichizzi |
| 4.0 User Stories | Chareen Elton |
| 4.1 User Story Dictionary | Alexx Verdichizzi / Chareen Elton / Laura Searle / Michael Little |
| 4.2 Iterative User Story Documents | Chareen Elton / Alexx Verdichizzi |
| 4.2.1 User Story Definition | Alexx Verdichizzi |
| 4.2.2 Flow of Interaction Diagram | Chareen Elton |
| 4.2.3 User Story Testing | Laura Searle / Chareen Elton |
| 4.2.4 Integration Testing | *(not yet required)* |
| 4.2.5 Wireframes | Laura Searle / Alexx Verdichizzi |
| 5.0 Object Oriented Design | Laura Searle |
| 5.1 High Level System Architecture | Alexx Verdichizzi / Michael Little |
| 5.1.2 High Level Software Architecture | Alexx Verdichizzi / Michael Little |
| 5.2 High Level Package Diagram and Components | Alexx Verdichizzi / Michael Little |
| 5.3 Use Case Analysis | Alexx Verdichizzi / Chareen Elton |
| 5.4 Domain Model and Class Diagram | Michael Little |
| 5.5 Establishment of the Database Objects and Data Access Strategy | Michael Little / Chareen Elton / Alexx Verdichizzi |
| 5.6 Sequence Diagram - optional | TO DO – complete if there is time |
| 5.7 Object Dictionary | Michael Little |
| 6.0 Software Release Report | Chareen Elton |
| 6.1 Usability Test Report | Chareen Elton |
| 6.2 System (User Story) Test Reports | *(not yet required)* |
| 7.0 Additional User Interface Design | Alexx Verdichizzi |
| 7.1 Additional Input and Output Screens | Alexx Verdichizzi |

| | |
|---|---|
| 7.2 Additional Data Display Screens | Alexx Verdichizzi |
| Special Notes | Laura Searle |
| Appendices | *(not yet required)* |
| Bibliography | *(not yet required)* |
| Glossary of Terms | All team members |
| Reference Materials | All team members |
| Index | Chareen Elton |