
Saint Augustine's College, Sydney

**Software Engineering Year 12: Programming
Project**

Project Documentation: Samuel Hildebrandt

FreeSpace Web Application

Table of Contents

Defining and Understanding.....	3
Task Definition.....	3
Legal and Ethical Considerations	3
Functional and Non-Functional Requirements	4
Functional Requirements	4
Non-functional Requirements	5
Implementation Method: Phased	6
Planning and Designing	7
Storyboard	7
Context Diagram	8
Data Flow Diagram	10
Structure Chart / Class Diagram	11
Algorithms.....	12
Flowchart	12
Algorithmic Description	15
GANNT Chart	17
Implementation.....	18
Development log	18
Testing – Test Table	30
Project showcase.....	33
GitHub repository	33
Evaluation	34
Project Reflection	34

Defining and Understanding

Task Definition

Many schools have valuable and specialised machinery in their technology and workshop spaces. However, much of this equipment remains underutilised because teachers face challenges supervising student projects outside regular class time. Without a structured system, providing safe and consistent access to these tools is difficult.

To solve this, I am developing FreeSpace, a web application designed for schools to manage their makerspaces. FreeSpace helps teachers coordinate and control student access to machinery and equipment within supervised learning environments. Inspired by university makerspaces, the application allows students to apply for scheduled use of tools and machines for academic and personal projects.

The web application's key goal is to unlock student creativity by offering structured access to advanced, often underutilised equipment. Through FreeSpace, students can book time to work on major or personal projects beyond regular lessons, promoting deeper hands-on opportunities. The application will support teachers by streamlining supervision through a centralised booking system, making project-based learning easier to integrate into the school timetable.

FreeSpace allows workshop managers to configure machines and workspaces, as individual items or grouped tools, which students and staff can book. While bookings typically require teacher supervision, the application includes a skills/qualifications feature that enables students to earn independent access to machines once they demonstrate safety and competence. The initial setup requires time and a willing teacher or workshop manager to configure the system.

By centralising bookings and supervision, the FreeSpace web application makes managing school equipment safe, efficient and scalable, fostering innovation, responsibility and creativity in school makerspaces.

Legal and Ethical Considerations

When developing FreeSpace, a makerspace management web application for schools, legal and ethical considerations around data privacy and security are essential. The application collects sensitive personal information, including student and teacher names, login details, machine booking logs and licensing progress. Compliance with the Australian Privacy Act 1988 is required. To meet this, FreeSpace will store data securely using strong authentication including hashing passwords to protect against unauthorised access to user credentials. Users and parents will be clearly informed about what data is collected and how it is used. Collecting only the minimum necessary data and obtaining explicit consent aligns with ethical best practice. When installed on a production server https protocol would be used and a more secure database engine such as MySQL to prevent local user access to data without database permissions.

Security measures also restrict access based on user roles; with students, teachers and administrators each having different permissions to book machines or manage data. Teachers can track user bookings to ensure safe use of equipment while the skill/qualification attributes prevent unqualified students from accessing machinery, reducing safety risks.

Accessibility is another critical consideration. FreeSpace must be usable by all students and staff, including those with disabilities. The application promotes equitable access to tools and workspaces regardless of a student's background or available free time, fostering inclusive learning.

Finally, a clear data retention policy will ensure personal information is not stored indefinitely and can be deleted upon request, respecting user privacy and ethical use.

Functional and Non-Functional Requirements

Functional Requirements

Requirement	Description
User Registration and Login	The application should allow teachers and students to securely create accounts and log in.
Booking and Calendar System	The application should allow users to book machines/workspaces for designated time slots and view them in a shared calendar.
Qualifications / Skills System	The application should enable students to earn the right to operate specific machines independently by demonstrating safety compliance and competence.
Machine and Workspace Configuration	Platform managers should be able to create and configure individual machines or grouped workspaces for booking.
User Roles and Permissions	The application should support different access levels for students, teachers and administrators, determining who can book or manage machine access.
Approvals of Bookings	The application should allow administrators or workshop managers to approve or decline student requests for booking slots.

Non-functional Requirements

Requirement	Description
Usability	<p>The application should have an intuitive interface allowing all users to navigate easily without prior training.</p> <p>Responsive web design for use with multiple sized devices.</p>
Performance	<p>Booking requests and data retrieval should process within 3 seconds to ensure a smooth user experience.</p>
Security	<p>User data and bookings must be stored securely with encryption during transmission and storage.</p>
Availability	<p>The application should maintain 99.9% uptime with scheduled maintenance performed outside peak school hours.</p>
Scalability	<p>The application should support growth in user numbers and machine/workspace listings as schools expand.</p>

Implementation Method: Phased

For my project FreeSpace, a school makerspace management web application, the most appropriate implementation method is the Phased approach. This involves gradually rolling out the system in stages, allowing each major feature to be introduced and tested one at a time rather than launching the entire platform at once.

This method suits FreeSpace for several reasons. The application manages safety-sensitive features such as machine bookings and student skills / licensing. A sudden full release could cause serious issues. For example, if students gained access to tools before demonstrating safety competence. Releasing modules like the booking calendar, user access and licensing system in phases allows for safer and more controlled testing of each function.

In addition, FreeSpace is not replacing an existing digital system. Most schools currently rely on manual methods such as spreadsheets or sign-up sheets. Because of this, a parallel rollout is unnecessary. A direct rollout would also be risky, as users would not have had time to become familiar with the system before full implementation.

A pilot approach, where the system is tested with a small group, may seem ideal. However, it is often impractical in a school context. There is rarely a separate group of users available for isolated testing and managing different environments would add unnecessary complexity. A pilot would also be unrealistic in the ten-week scope of this major project.

In contrast, a phased rollout is both achievable and realistic. For example, the system could first be introduced to teachers, allowing them to set up machine spaces and test the calendar. Then the skills system could be enabled for approved students to gain independent access.

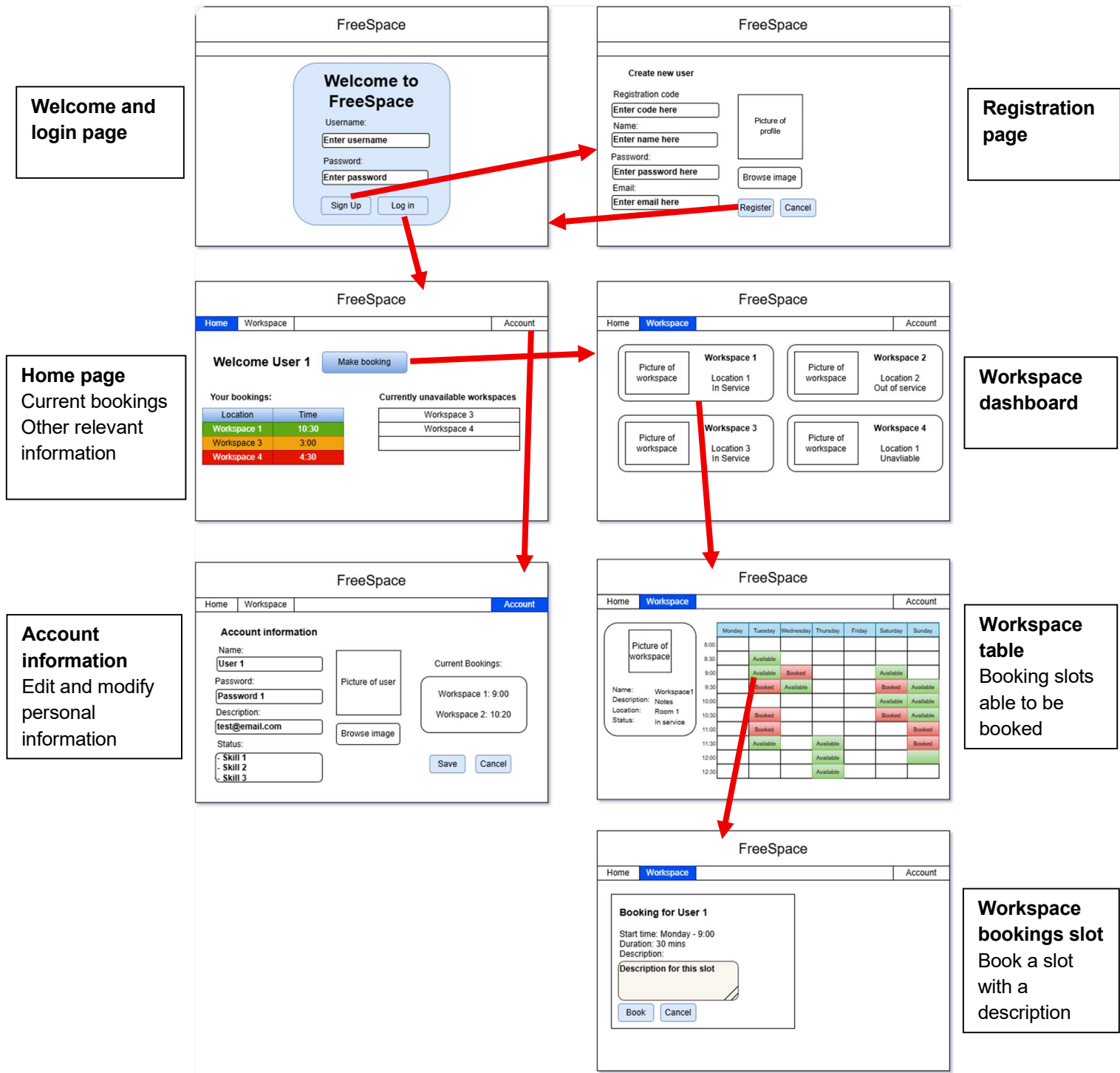
This approach ensures a safe, structured and user-friendly transition that supports the goals of FreeSpace, to empower students, assist teachers and improve access to equipment in a secure and organised way.

Planning and Designing

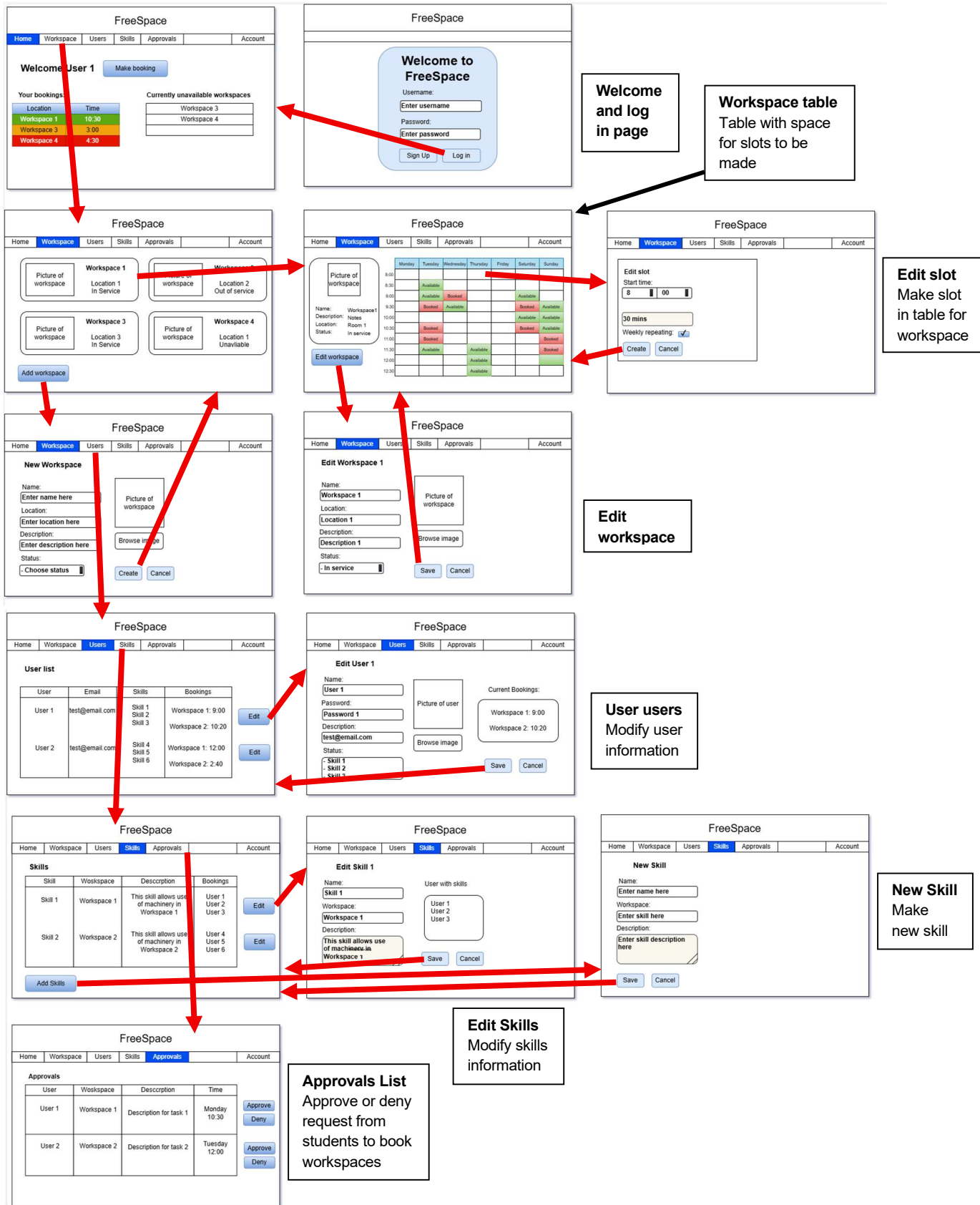
Storyboard

The storyboard for the FreeSpace web application is divided into two parts: the main functionality available to a typical student user and the functionality available to an operations administrator.

User Functionality:

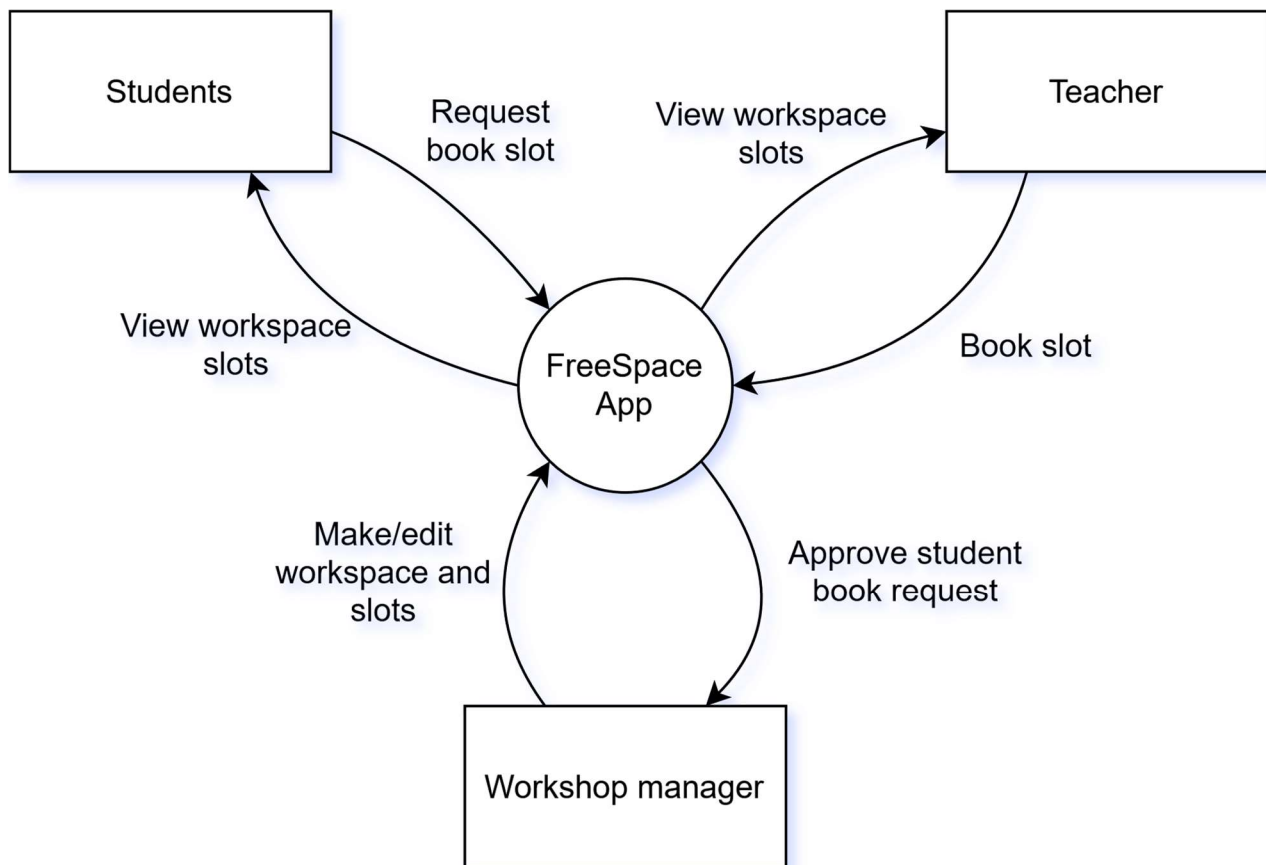


Admin Functionality:



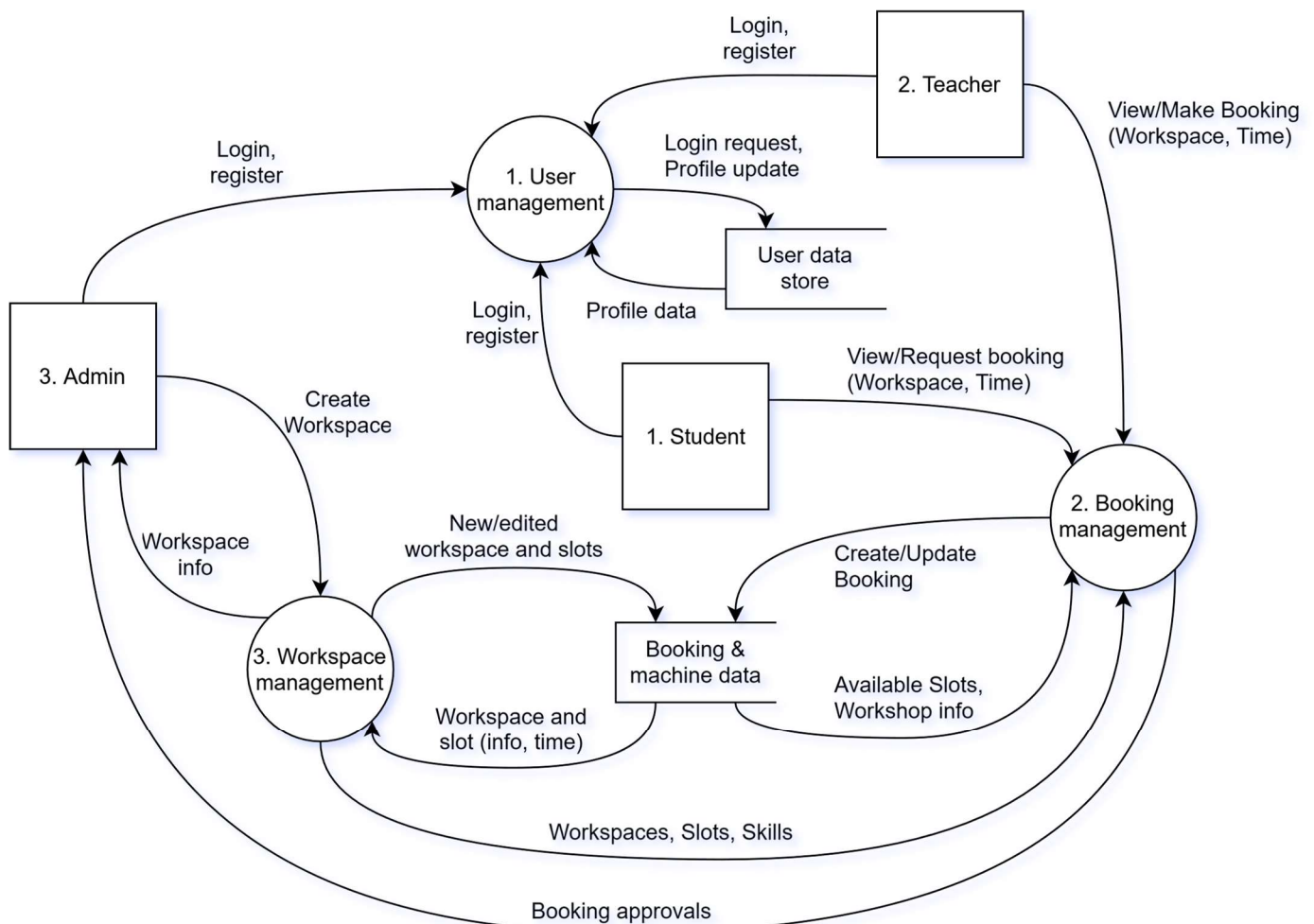
Context Diagram

This context diagram explores the main conjunction between the different types of users being the students, teachers and admin staff and how they interact with the FreeSpace web application.



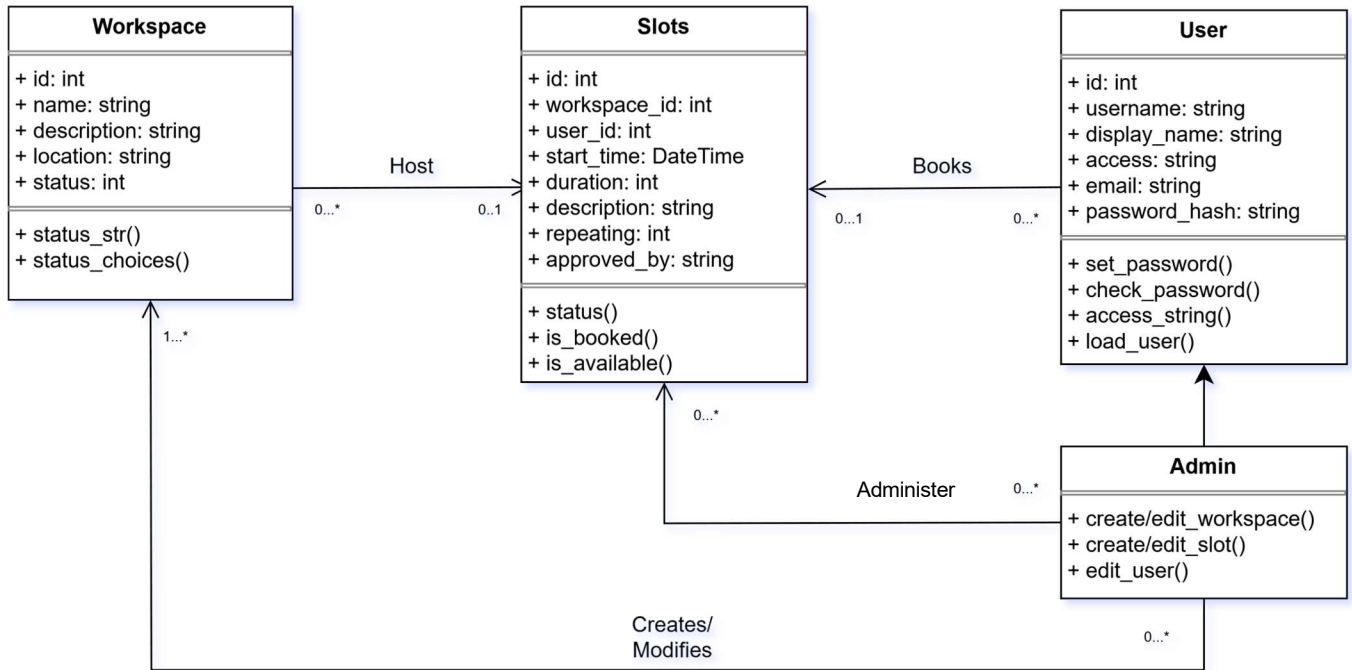
Data Flow Diagram

This data flow diagram shows a detailed breakdown of the system interactions between user entities, main processes and database stores for the FreeSpace web application.



Structure Chart / Class Diagram

This class diagram explores the three main class profiles included in FreeSpace: the Workspace, Slot and User classes. The Admin class is an inherited class of User and follows many of the same specifications as a standard User class. This diagram also illustrates the relational data transferred between the classes.

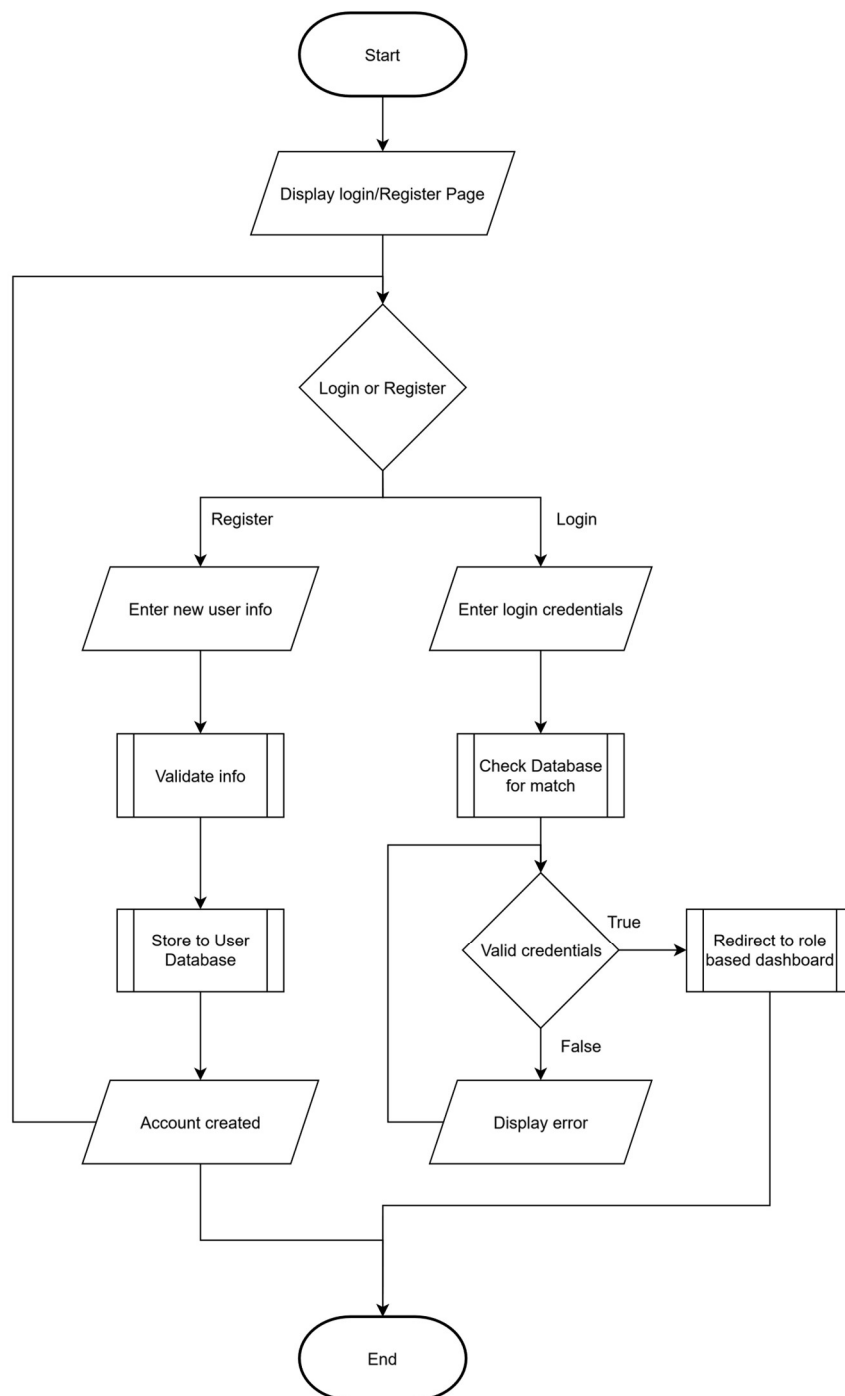


Algorithms

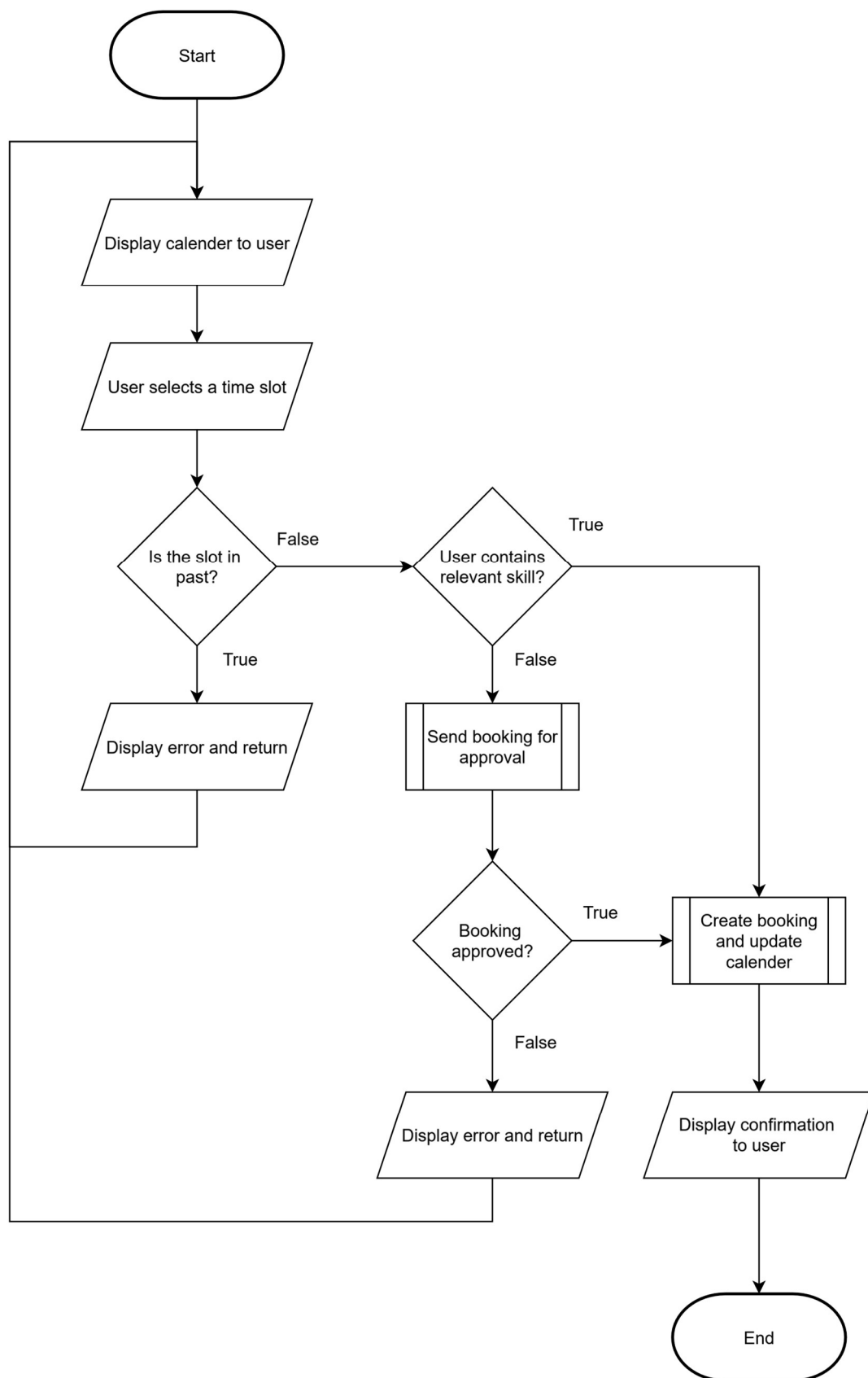
Flowchart

Here are the three main functional processes of the FreeSpace web application: User Management, Booking Management and Workshop Management. Each is represented by its own flowchart diagram, showing all code logic and algorithmic structure.

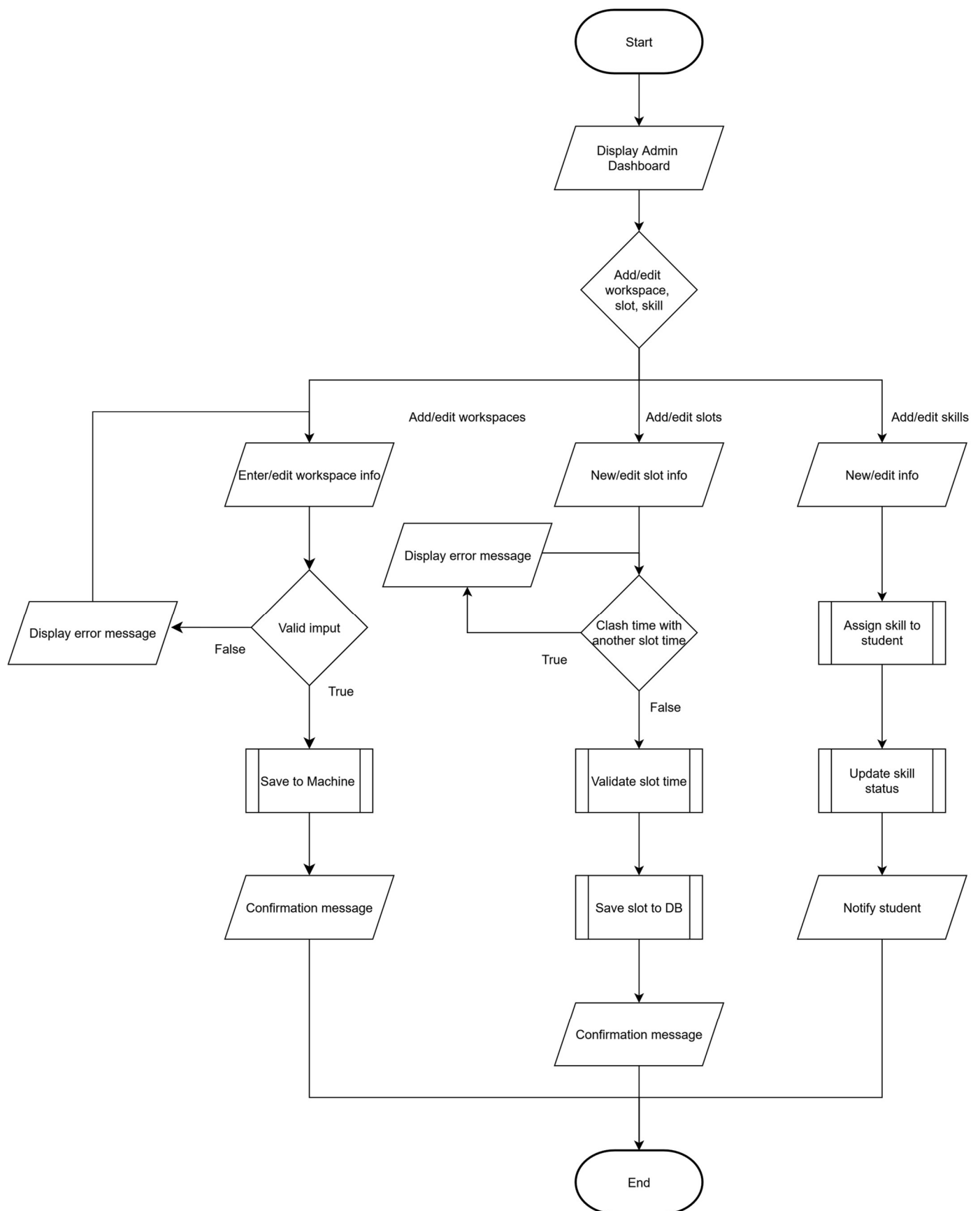
User Management:



Booking Management:



Workshop Management (Admin management)



Algorithmic Description

Below is the breakdown of algorithmic descriptions for the FreeSpace web application, covering the functionality of the three main processes: User Management, Booking Management and Workshop Management.

User Management – (Handle user access function):

1. **Display Login/Register Page:** Show a page prompting the user to either login or register.
2. **User Selection:** Receive input — does the user choose "Login" or "Register"?
3. **If Register:**
 - Prompt the user to enter name, email, password and role (via Registration Code)
 - Validate the input (must not be blank, email format is correct, strong password, valid code).
 - If invalid, display error and return to step 3.
 - Save the user information to the database.
 - Display success message.
4. **If Login:**
 - Prompt the user to enter their email and password.
 - Check credentials against the user database.
 - If match not found, display error and return to step 4.
5. **Determine Role:**
 - Based on stored user role (Student, Teacher or Admin), redirect to the corresponding dashboard.
6. **End Process.**

Booking Management – (Process booking request function):

- **Show Booking Calendar:** Display available machine / workstation time slots.
 - **User Slot Selection:** Receive input — user selects a machine / workstation and time slot.
 - **Validate Time Slot:**
 - Check if the selected time is in the future.
 - If in the past, display error and return to step 1.
 - **Check Qualifications/Skills Required:**
 - If user is qualified for selected machine, proceed to step 7.
 - If not, proceed to step 5.
 - **Approval Required:**
 - Submit booking request to teacher for approval (if booking was for made by a student)
 - Wait for approval decision.
 - If denied, display message and return to calendar.
-

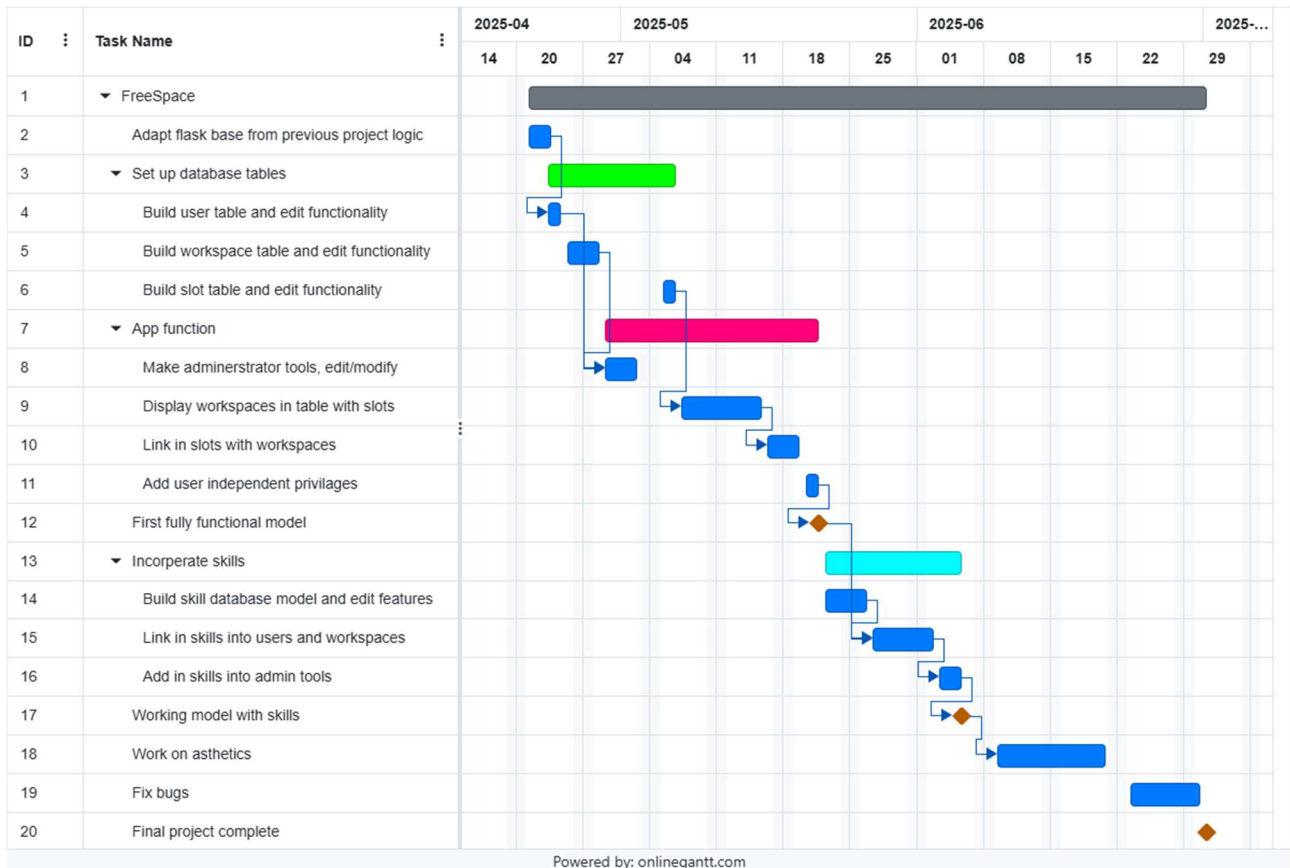
-
- Create Booking:
 - o Save booking to database.
 - o Mark slot as booked.
 - Confirmation:
 - o Display confirmation message to user.
 - End Process.

Workspace Management – (Admin manage workspace function):

1. **Display Admin Dashboard:** Show options to “Add/Edit Workspace,” “Create Booking Slot,” or “Assign License.”
2. **If Add/Edit Machine:**
 - o Prompt for workspace details (name, type, safety notes, required skills, etc.).
 - o Validate all required inputs.
 - o If input is invalid, display error and return to step 2.
 - o Save workspace to database.
 - o Display success message.
3. **If Create Booking Slot:**
 - o Prompt admin to select workspace and define date/time slot.
 - o Validate that time is in the future and does not conflict with existing slots.
 - o If invalid, display error and return to step 3.
 - o Save slot to database.
 - o Display confirmation.
4. **If Assign Qualifications/Skills to User:**
 - o Prompt admin to select a student.
 - o Check that student has met safety/training criteria.
 - o If not eligible, display warning and return to step 4.
 - o Store relevant qualification/skill for user into database.
 - o Notify student of skill assignment.
5. **Return to Admin Menu or End Process.**

GANNT Chart

Here below is the GANNT Chart for the development of the FreeSpace web application, including the full development lifecycle of the application and three major milestones at which development logs were taken and documented as progress to the final product.


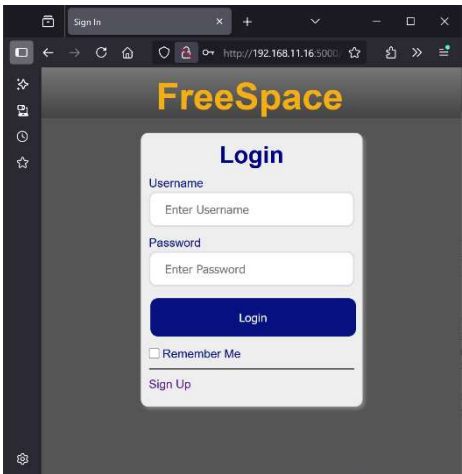
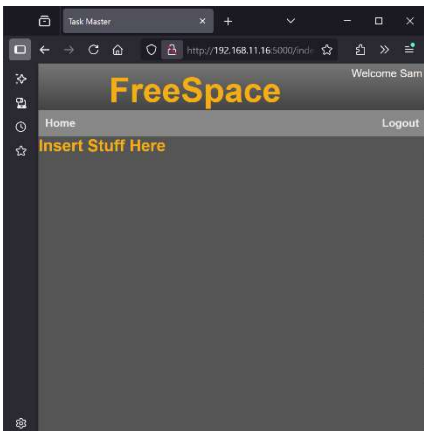


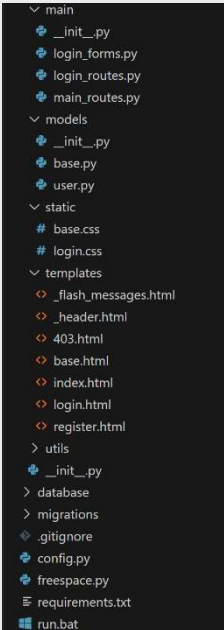
Below is the link to the full GANNT Chart file

https://github.com/SHildebrandt4472/FreeSpace/blob/main/FreeSpace_ganntt_chart.gantt

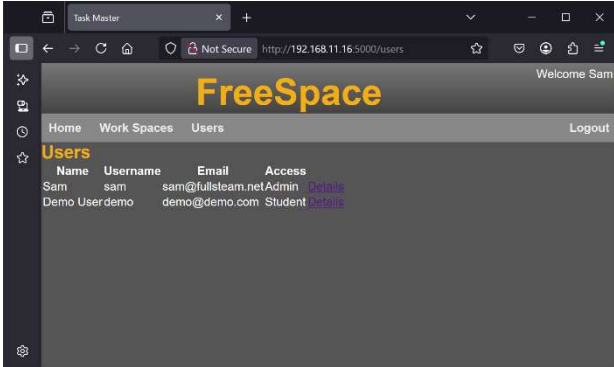
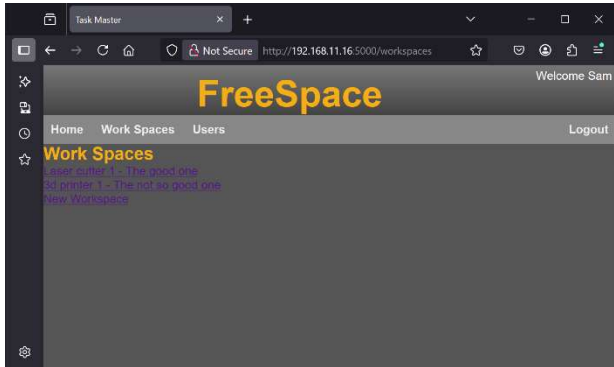
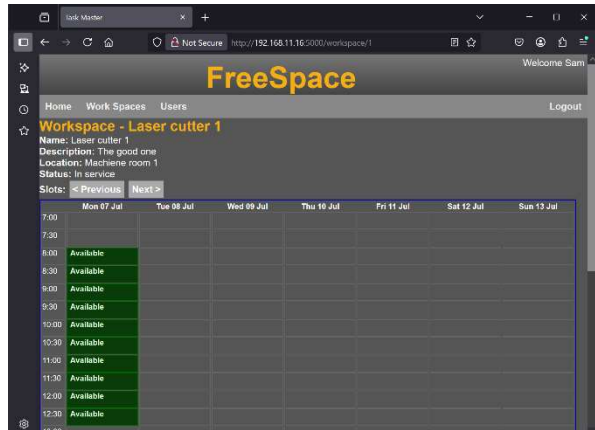
Implementation

Development log

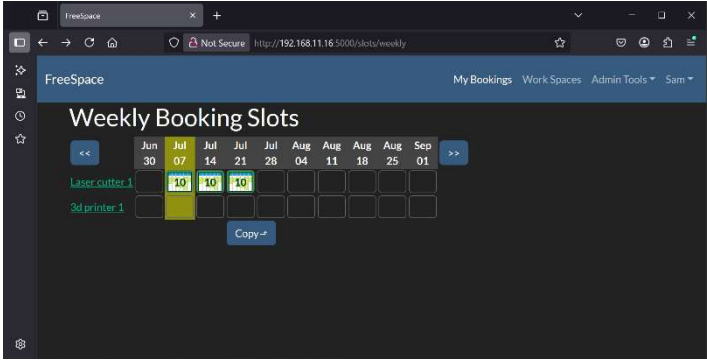
Development Log Entry 1	
Date	21 st of April
Week Number	End of Week 1 into development
Summary of Work Done	<p>This marks the first major milestone in my project where I have created a functioning flask web application. From my previous project, a task Todo web application, I copied over the code to form the backbone of my new project FreeSpace. This task then involved deleting all the Todo application content and transforming the base into a fully operational Flask web application keeping the user database, login and register functionality.</p> <p>This basic app in which the rest of FreeSpace will be built upon is shown below:</p> <ul style="list-style-type: none">- Register and log in users <div></div> <div></div> <ul style="list-style-type: none">- Run a basic web page online <div></div>

	<ul style="list-style-type: none"> - Database routes, forms, models and templates in place <p>These foundational features will then be built upon to include further workspace and slot database tables and numerous other pages to cover all the necessary application requirements to meet all the functionality of the desired application.</p>	
Challenges and Solutions	<p>It was initially difficult to set up and think through an appropriate file structure and code layout, particularly when it came to creating a clear folder breakdown. I ended up defining the following overarching folders:</p> <ul style="list-style-type: none"> - Main (for forms and routes) - Models (for database models) - Static (for CSS pages) - Templates (for html pages) <p>This proved challenging, as organising all the code into separate folders took time and planning. However, establishing a solid baseline file structure will ultimately save time in the future by making it easier to manage and expand the project as more files are added.</p> <p>When cutting out the original content from the previous project, it was sometimes difficult to determine which lines of code were essential for the program's functionality and which were redundant in the context of the new application. For example, many of the route files included imports that served a specific purpose in the original project but may or may not be relevant to FreeSpace. In the end, I chose to comment out many of these lines rather than delete them entirely, as this makes it quick and easy to restore them if the program suddenly stops running or behaves unexpectedly.</p>	 <pre> app > main > main_routes.py > ... 1 from flask import render_template, flash, redirect, url_for, request #,abort, session 2 3 from flask_login import current_user #, login_user, logout_user #, login_required 4 #from urllib.parse import urlparse 5 #from app.models import db, User 6 from app.main import bp 7 #import datetime 8 #from sqlalchemy import text 9 </pre>
Milestones Achieved	<p>This stage marks the first baseline version of my program, which allows a user to register and log in through an authentication interface. The program works without any bugs and runs on the same CSS styles as TaskMaster which will ultimately be improved but for now it allows for a baseline model to be used and tested. From here the functionality of FreeSpace will be implemented using this blueprint code made.</p>	

Development Log Entry 2

Date	22 nd of May
Week Number	Week 5 into development
Summary of Work Done	<p>This is the next major milestone of the first fully functional prototype (with very little attention towards looks/feel). All the database framework and route functionality are operational for the users and workspaces. Functionality has initially been made for administration tools to manage users and workspaces with slots. This involves being able to:</p> <ul style="list-style-type: none">- View, edit and delete users  <ul style="list-style-type: none">- Create, view, edit, delete workspaces  <ul style="list-style-type: none">- Create, view, edit, delete slots for any workspace 

	<p>The workspace booking tables have been created in a basic html table showing all bookings made by the Admin. The individual slots can be booked by any user and is displayed as red and booked when done so.</p> <p>Individual user accounts with different access levels have been created, including the teacher accounts and student accounts.</p>
Challenges and Solutions	<p>Initially designing and building up the several databases and several fields for each of the database tables proved difficult. The main difficulty proved in linking the slots table with all booked slots with appropriate workspace tables. For example, seen below is the final code that joins the slot table with the workspace table. This process took considerable time but ensured a strong underlying database framework for the rest of the program to be built upon. Through careful planning out a numerous testing and debugging of the database tables, all the tables were able to be set up properly in a consistent and efficient way.</p> <pre> 23 created_at = db.Column(db.DateTime, default=db.func.datetime('now')) 24 last_modified = db.Column(db.DateTime, default=db.func.datetime('now'), onupdate=db.func.datetime('now') 25 slots = db.relationship('Slot', backref='workspace', lazy='dynamic') </pre> <p>Also, initially working out a way to integrate the create and edit pages for workspaces and slots by sharing a common function for multiple urls that use the same template for add and edit proved to be difficult at first. Having the pages use the same template was challenging, as it required displaying all the same data but filling in the fields for the edit page and changing the title. Through many trials, a solution was developed where a single HTML template could be used to display the data for both the creation and edit pages and multiple urls could be handles by the same function (with minor differences handled by a simple if block)</p> <pre> 48 @bp.route('/workspace/new', defaults={'id':None}) 49 @bp.route('/workspace/<id>/edit') 50 @login_required 51 def edit_workspace(id): 52 if not current_user.is_admin(): 53 abort(403) 54 55 @bp.route('/workspace/add', methods=['POST'], defaults={'id':None}) 56 @bp.route('/workspace/<id>/update', methods=['POST']) 57 @login_required 58 def update_workspace(id): 59 if not current_user.is_admin(): 60 abort(403) </pre>
Milestones Achieved	<p>This stage marks the first fully functional model, which entails a complete framework of the functional final application. However, this does not yet include administration manager and admin privileges such as approvals or any skill and qualification framework. This revision of the application includes functional features such as...</p> <ul style="list-style-type: none"> - Log in and register - View workspaces - Make bookings on workspaces - View personal bookings <p>Due to this only being a functional model, it still uses the simplistic aesthetic from the initial Flask framework derived from a previous project and minimal data displayed in templates. This is to be addressed in a future revision.</p>

Development Log Entry 3	
Date	10 th of June
Week Number	Week 7-8 into development
Summary of Work Done	<p>This update of the code includes a major upgrade of the user UI through the upgrading of the CSS framework to use Bootstrap. This involved integrating bootstrap classes throughout the HTML code and restructuring all the content in all the pages and upgrading all display functionality to utilise the bootstrap framework. This included:</p> <ul style="list-style-type: none"> - Creating a new title header bar - Applying a new colour scheme, using Bootstrap's primary and secondary colour palette based on the existing colour set  <p>Additionally, to address the lack of administration functionality, a weekly slot calendar was implemented. This allows weekly slots to be automatically copied over to the following week with simplicity and ease, without the need to manually duplicate each slot. This involved updating the weekly repeat field of the slot database, allowing the workshop manager to tick a field when creating the slot to make it repeat weekly or not.</p>
Challenges and Solutions	<p>An aspect that proved to be challenging was designing a user-friendly UI with the new update of the bootstrap CSS upgrade. With the new aesthetic features of Bootstrap automatically making elements look more cleaner and provide quick steps to make the program appear more professional looking, the limitless possibilities of these features were initially challenging to design exactly how each page was ultimately going to look and appear. This required a major rethink of the storyboard and layout and overall look of the ideas for the looks of each page, which then allowed a much faster development of the layout of pages particularly after the initial experimentation of the of the top heading bar. Moving from a CSS FILE based structure (where styles are kept in separate files and shared across HTML elements) to the Bootstrap model where Bootstrap classes are inserted inline in every HTML element was particularly challenging. The HTML code becomes far less readable, and more care needs to be taken to use the same classes across several elements to keep a consistent look. Shown</p>

below is the large number and unfamiliar bootstrap classes that ended up being required to implement part of the header navigation bar.

```

1 <nav class="navbar navbar-expand-lg bg-primary data-bs-theme="dark">
2 <div class="container-fluid">
3 <div class="navbar-brand">FreeSpace</div>
4 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false"
5 <span class="navbar-toggler-icon"></span>
6 </button>
7 <div class="collapse navbar-collapse justify-content-end" id="navbarColor01">
8 <ul class="navbar-nav">
9 <li class="nav-item">
10 <a class="nav-link active" href="{url_for('main.home')}">My bookings
11 <span class="visually-hidden">{current}</span>
12 </a>
13 </li>
14 <li class="nav-item">
15 <a class="nav-link" href="{url_for('main.workspaces')}">Work Spaces</a>
16 </li>
17 </ul>
18 <div class="dropdown">
19 <button class="dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">Admin Tools</button>
20 <ul class="dropdown-menu">
21 <li class="dropdown-item">Approvals</li>
22 <li class="dropdown-item">Booking Slots</li>
23 </ul>
24 </div>
25 <div class="dropdown">
26 <button class="dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">Users</button>
27 <ul class="dropdown-menu">
28 <li class="dropdown-item">Users</li>
29 </ul>
30 </div>
31 </div>
32 </div>
33 </div>
34 </div>
35 </div>
36 </div>
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 </div>
48 </div>
49 </div>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>

```

Another difficulty experienced with the integration of Bootstrap was trying to unpick the collisions of the original CSS competing with the new Bootstrap CSS. In many situations there were elements that didn't appear right and wouldn't change when the CSS was modified for it. For example, below is a section of the original menu bar CSS which initially conflicted with the new Bootstrap menu bar code, in which the old CSS had to be removed. This resulted in a lot of time unpicking a fixing up any underlying CSS that was conflicting with the new Bootstrap code templates which overall allowed for the progress of the UI aesthetics. The final product ended up a combination of CSS files that enhanced the Bootstrap base.

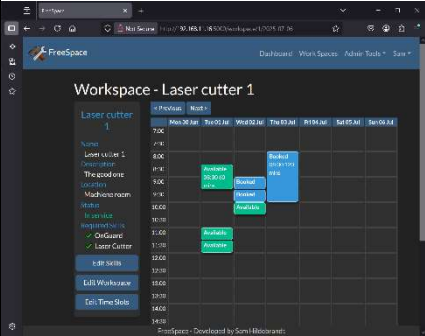
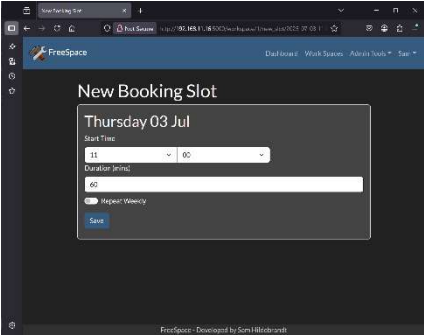
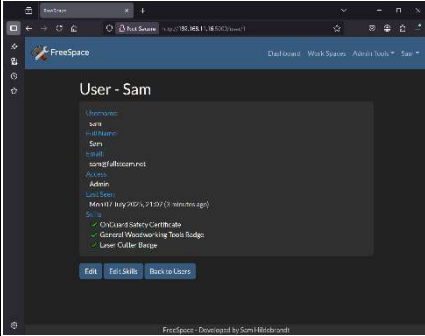
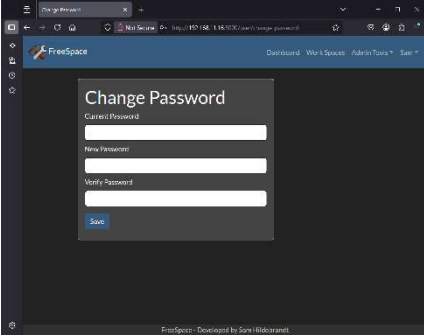
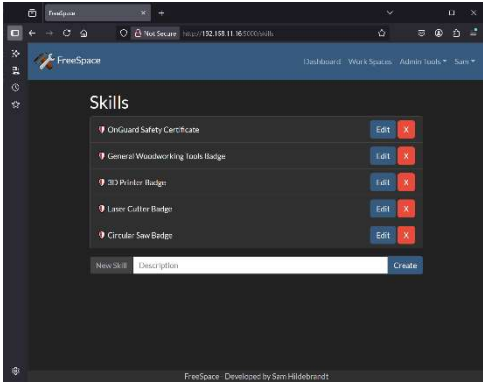
```

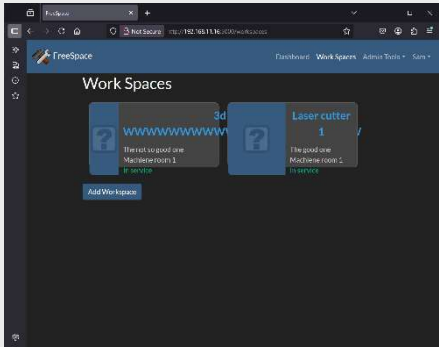
52 #menubar {
53 font-family: Arial, Helvetica, sans-serif;
54 font-weight: bold;
55 color: #eee;
56 background-color: #888;
57 }
58
59 #menubar ul {
60 list-style-type: none;
61 overflow: hidden;
62 margin: 0;
63 padding: 0;
64 }
65
66 #menubar li {
67 display: inline-block;
68 padding: 10px 0px;
69 /*background-color: #037;*/
70 text-align: center;
71 }
72
73 #menubar li.right {
74 float: right;
75 }

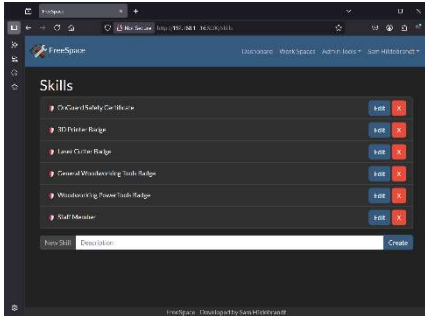
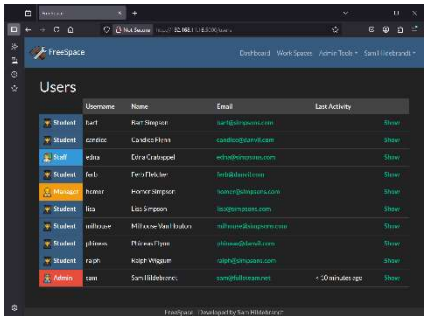
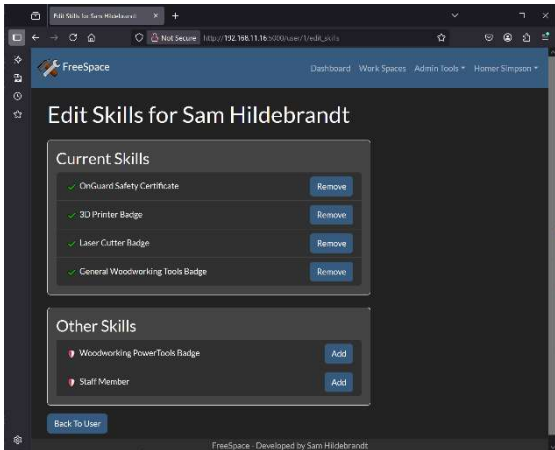
```

One area Bootstrap wasn't able to help was formatting the Slot Table/Booking Calendar. The bookings needed to be able to be placed at exact vertical positions in the columns (corresponding to their times). By looking at the HTML source from the timetable displayed in Seqta I discovered the approach of using Absolute positioning and use specific CSS classes to set the positions for given times. In creating the CSS I also learned about the CSS calc() function to make calculating the positions easy to enter and modify. The below snippet of CSS shows some of these class styles used for the start time and slot duration.

	<pre> 295 .time_19-00 { top: calc(720 * var(--px_per_min)); } 296 .time_19-05 { top: calc(725 * var(--px_per_min)); } 297 .time_19-10 { top: calc(730 * var(--px_per_min)); } 298 .time_19-15 { top: calc(735 * var(--px_per_min)); } 299 .time_19-20 { top: calc(740 * var(--px_per_min)); } 300 .time_19-25 { top: calc(745 * var(--px_per_min)); } 301 .time_19-30 { top: calc(750 * var(--px_per_min)); } 302 .time_19-35 { top: calc(755 * var(--px_per_min)); } 303 .time_19-40 { top: calc(760 * var(--px_per_min)); } 304 .time_19-45 { top: calc(765 * var(--px_per_min)); } 305 .time_19-50 { top: calc(770 * var(--px_per_min)); } 306 .time_19-55 { top: calc(775 * var(--px_per_min)); } 307 308 .dur_10 { height: calc(10 * var(--px_per_min) - 1px); } 309 .dur_15 { height: calc(15 * var(--px_per_min) - 1px); } 310 .dur_20 { height: calc(20 * var(--px_per_min) - 1px); } 311 .dur_25 { height: calc(25 * var(--px_per_min) - 1px); } 312 .dur_30 { height: calc(30 * var(--px_per_min) - 1px); } 313 .dur_35 { height: calc(35 * var(--px_per_min) - 1px); } 314 .dur_40 { height: calc(40 * var(--px_per_min) - 1px); } 315 .dur_45 { height: calc(45 * var(--px_per_min) - 1px); } 316 .dur_50 { height: calc(50 * var(--px_per_min) - 1px); } 317 .dur_55 { height: calc(55 * var(--px_per_min) - 1px); } 318 .dur_60 { height: calc(60 * var(--px_per_min) - 1px); } 319 .dur_65 { height: calc(65 * var(--px_per_min) - 1px); } 320 .dur_70 { height: calc(70 * var(--px_per_min) - 1px); } </pre>	
Milestones Achieved	<p>This stage represents the major UI redesign and upgrade to aesthetics which overall will shape the structure and flow of the program nearer to the final product. Through this milestone in aesthetics, the further integration of bootstrap will enable the further development of the aesthetic and feel of the program. This upgrade has added the features of:</p> <ul style="list-style-type: none"> - More modern and professional looking layout and aesthetics of the application - Better CSS framework making it easier to include more stylistic features that all cohere to the same style - Weekly repeating and weekly copying of slots for future weeks functionality for admin setup <p>From here and through further bootstrap integration, the application user interface needs to be further refined to upgrade the look and feel of the slot tables and workspace displays. Further redesign of the creation and editing pages for both workspaces and slots need to be done.</p>	

Development Log Entry 4	
Date	3 rd July
Week Number	Week 10 of development
Summary of Work Done	<p>Due to unforeseen circumstances, my project due date was extended to 11 weeks which has allowed me to catch up on completing all functionality of the program, including all the administration features such as user management. In this time, the full qualifications/skills functionality has been built into the application. Therefore, the work now completed includes:</p> <ul style="list-style-type: none"> Fully functional admin features including workspace, slot and user management <div>   </div> <ul style="list-style-type: none"> Personal user view and password change features <div>   </div> <ul style="list-style-type: none"> Skill database table implemented <div>  </div> <ul style="list-style-type: none"> Skill creation and management built into workspaces performed by the admin

	<p>This version has also seen a near complete conversion over to the bootstrap CSS styling. Through careful planning and trial of different aesthetic appeals, the application uses a wide range of bootstrap CSS logic and Javascript, especially through a similar base line creating and editing format used for workspace, slot, and user creation and editing. Similarly, a consistent style for approvals and skills has been employed.</p>
Challenges and Solutions	<p>The biggest difficulty was working out the many to many relationships for implementing skills into the application. This challenge involved linking skills as a workspace requirement category whilst also storing a list of the acquired skills on each user in the user table. To solve this issue, separate joining and relationship tables were created, enabling the linking of these database tables and allowing both workspaces and users to each have a list of skills. The following code shows the join table definition and db relationship for linking workspaces and skills as a many to many relationship.</p> <pre> 7 workspace_skill_table = db.Table('workspace_skill', 8 db.Column('workspace_id', db.Integer, db.ForeignKey('work_space.id', ondelete='CASCADE'), primary_key=True), 9 db.Column('skill_id', db.Integer, db.ForeignKey('skill.id', ondelete='CASCADE'), primary_key=True) 10) 37 required_skills = db.relationship('Skill', secondary=workspace_skill_table, backref='workspaces') </pre> <p>A major difficulty as this stage has been dealing with the large number of small bugs found in testing different pages, like breaks in sessions when the user isn't redirected back to the same page, different admin and manager privileges and levels of management, dealing with boundary circumstances in workspace names and descriptions in terms of displaying on the dashboards, and several others. These required a lot of time and debugging to fix. In a lot of cases, testing and debugging involved printing test lines to the command line inside VS code to ensure that functionality was running under different if statements.</p> 
Milestones Achieved	<p>This stage represents a nearly fully functional model except for minor bug and small features to be added. It however requires little tweaks to styling to further enhance and ensure there will be optimal user experience for the users to interact and use the application. Also done was to create a command line function to initialise a full set of demonstration data, making this milestone a fully testable stage. The demo data includes multiple students/teachers/etc, workspaces with bookings/etc.</p> <p>This milestone signifies:</p> <ul style="list-style-type: none"> - A fully operable program to test large scale data in terms of many users, workspaces, slots and skills - Only requires minor bug fixes before completion - Near complete UI upgrade to bootstrap and stylistic arrangement

Development Log Entry 5	
Date	7 th July
Week Number	Week 11 of development
Summary of Work Done	<p>This point in the project has involved completing minor bug fixes and testing over numerous platforms and devices testing browser differences. This stage also required testing completing all core functions, such as making bookings and approving them and analysing how the data is displayed on both student and admin level. This process led to some minor adjustments to user privileges. Managers can now assign skills to students, but they still do not have access to user information management, which remains the responsibility of the admin.</p> <p>Overall, the fine tweaks in program logic includes:</p> <ul style="list-style-type: none"> - New icons and badges for skills to work across multiple platforms - Minor bug fixes in session routing after making booking and editing user skills found when testing <div>   </div> <ul style="list-style-type: none"> - Updated manager privileges to enable them to edit user skills <div>  </div> <p>This stage also involved developing a user guide to help users navigate all the functionality of the app. This includes a guidance around all the different user privileges and user independent functions, allowing the application to be understood from an outside observer's perspective.</p>

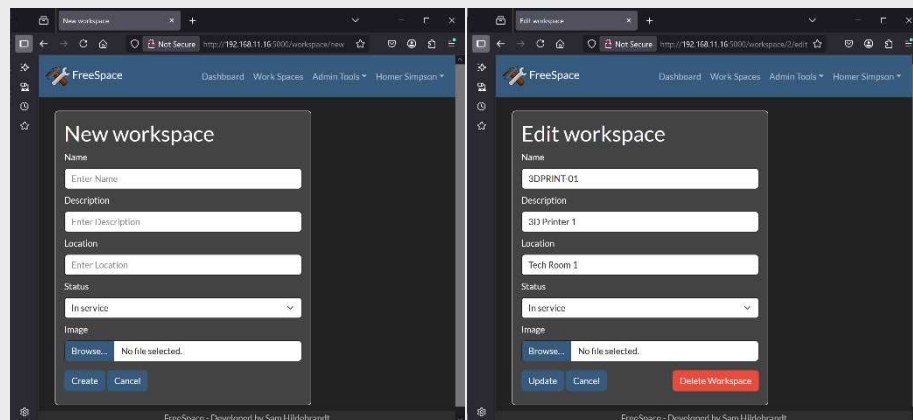
Challenges and Solutions

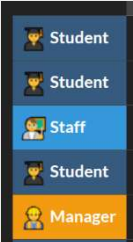
This stage proved challenging especially in testing all links between pages and testing session routing when completing an action and returning to specific stages. Testing all functionality multiple times was made easier with the help of newly created demo test

```
15 users = {
16   'sam': ({'display_name': 'Sam Hildebrandt', 'email': 'sam@fullsteam.net', 'access': 40, 'skills': ['onguard', '3dprint', 'laser', 'woodwork'] }),
17   'homer': ({'display_name': 'Homer Simpson', 'email': 'homer@simpsons.com', 'access': 30, 'skills': ['onguard', '3dprint', 'laser', 'woodwork', 'staff'] }),
18   'edna': ({'display_name': 'Edna Crabbappel', 'email': 'edna@simpsons.com', 'access': 20, 'skills': ['onguard', '3dprint', 'laser', 'woodwork', 'staff'] }),
19   'bart': ({'display_name': 'Bart Simpson', 'email': 'bart@simpsons.com', 'access': 10, 'skills': []}),
20   'lisa': ({'display_name': 'Lisa Simpson', 'email': 'lisa@simpsons.com', 'access': 10, 'skills': ['onguard', 'woodwork', '3dprint', 'laser'] }),
21   'milhouse': ({'display_name': 'Milhouse Van Houten', 'email': 'milhouse@simpsons.com', 'access': 10, 'skills': ['onguard', '3dprint'] }),
22   'ferris': ({'display_name': 'Ferris Fletcher', 'email': 'ferris@danvil.com', 'access': 10, 'skills': ['onguard', '3dprint', 'laser'] }),
23   'phineas': ({'display_name': 'Phineas Flynn', 'email': 'phineas@danvil.com', 'access': 10, 'skills': ['onguard', '3dprint', 'laser'] }),
24   'candice': ({'display_name': 'Candice Flynn', 'email': 'candice@danvil.com', 'access': 10, 'skills': ['onguard', 'woodwork'] }),
25   'ralph': ({'display_name': 'Ralph Wiggum', 'email': 'ralph@simpsons.com', 'access': 10, 'skills': ['onguard', 'woodwork'] }),
26 }
27
28 skills = {
29   'onguard': ({'description': 'Onguard Safety Certificate',
30   '3dprint': ({'description': '3D Printer Badge',
31   'laser': ({'description': 'Laser Cutter Badge',
32   'woodwork': ({'description': 'General Woodworking Tools Badge',
33   'ptools': ({'description': 'Woodworking PowerTools Badge',
34   'staff': ({'description': 'Staff Member',
35
36
37
38 weekday_slots = [
39   {'start': dt.time(7,15), 'duration': 75},
40   {'start': dt.time(12,00), 'duration': 120},
41   {'start': dt.time(15,30), 'duration': 120},
42   {'start': dt.time(17,30), 'duration': 120},
43 ]
44
45 workspaces = {
46   'LASER-01': ({'description': 'Laser Cutter', 'location': 'Machine Room 1', 'skills': ['onguard', 'laser'], 'thumbnail': 'laser.png'},
47   '3DPRINT-01': ({'description': '3D Printer 1', 'location': 'Tech Room 1', 'skills': ['onguard', '3dprint'], 'thumbnail': '3dprinter.png'},
48   '3DPRINT-02': ({'description': '3D Printer 2', 'location': 'Tech Room 1', 'skills': ['onguard', '3dprint'], 'thumbnail': '3dprinter.png'},
49   '3DPRINT-03': ({'description': '3D Printer 3', 'location': 'Tech Room 1', 'skills': ['onguard', 'staff'], 'thumbnail': '3dprinter2.png'},
50   'BENCH-01': ({'description': 'B1 Wood Tech Bench', 'location': 'Timber Room', 'skills': ['onguard', 'woodwork'], 'thumbnail': 'workbench.png'},
51   'BENCH-02': ({'description': 'B2 Wood Tech Bench', 'location': 'Timber Room', 'skills': ['onguard', 'woodwork'], 'thumbnail': 'workbench.png'},
52   'BENCH-03': ({'description': 'B3 Wood Tech Bench', 'location': 'Timber Room', 'skills': ['onguard', 'woodwork'], 'thumbnail': 'workbench.png'},
53   'BENCH-04': ({'description': 'B4 Wood Tech Bench', 'location': 'Timber Room', 'skills': ['onguard', 'woodwork'], 'thumbnail': 'workbench.png'},
54   'BSAW-01': ({'description': 'Band Saw', 'location': 'Machine Room 2', 'skills': ['onguard', 'woodwork', 'ptools'], 'thumbnail': 'bandsaw.png'},
55   'CSAW-01': ({'description': 'Circular Saw', 'location': 'Machine Room 2', 'skills': ['onguard', 'woodwork', 'ptools'], 'thumbnail': 'sawbench.png'},
```

data. To solve this, I carried out planned and structured testing of all pages and functions. This found little bugs in user privileges in some places and incorrect redirects to wrong pages in others. Without clear and careful structured testing, this step would have proved to be harder, but the structured approach allowed me to fix any immediate bugs found on the spot.

Also, this stage also proved challenging in fixing a user experience issue with the Save or Create button used for creating and editing workspaces and bookings. It was discovered through testing that the button originally labelled 'Create Booking' on the make slot page, was being overwritten by the edit pages "Save button". This proved to be confusing for the user experience and hence required a fix to ensure the correct labels appeared in the appropriate places. The change improved user clarity and made the steps in the program's functionality easier to understand.



	<p>In testing across multiple platforms (android/apple/chrome) I discovered the Unicode Characters I had used for small icons (e.g. Ticks/Crosses/user Roles) were quite different in different browsers and not coloured in one browser. In order resolve this I screen captured each character and changed the HTML to display PNGs instead of the Unicode. This guarantees a uniform look across multiple browsers.</p>  <pre> 22 <th scope="row" class="useraccess-{{user.access}}"> 23 24 <td>{{ user.username }}</td> </pre>
Milestones Achieved	<p>This interaction of the application marks a near finished usable version of the FreeSpace application. After thorough testing, it now completes all desired features efficiently and without bugs. Considering the scale of the project, the program has been optimised to deliver the necessary logic within the available time frame, which includes:</p> <ul style="list-style-type: none"> - 4 relational database tables storing users, workspaces, slots and skills - 2 joining tables (for joining Slots to Workspaces and Slots to Users) - User registration, log in and management - User access levels student, teacher, manager, admin all with individual functionality - Ability to make, edit and delete workspaces and individual slots in workspace tables - Functionality to request and book time slots for individual workspaces - Create, edit, delete skills which can be assigned to workspaces and users and is used to control permission for users to book workspaces.

Testing – Test Table

Test ID	Category	Test Case Description	Input to Provide	Expected Output	Actual Output	Pass/Fail	Action Taken
Test 1	Path Coverage	Booking a machine with valid skills	Student books a slot available in the workspace table	Booking is confirmed and added to calendar	Booking successfully confirmed and added to calendar displaying as 'booked'	Pass	N/A
Test 2	Path Coverage	Student successfully makes a booking and returns to the appropriate page (testing session persistence)	Student makes successful booking by clicking the book button ready to be redirected back.	Taken back to the previous page of the workspace slot table	Program displayed TypeError referring to not session appropriate	Fail	Implemented session validation using secure tokens stored in 'localStorage' and added route guards to rehydrate user session state when navigating via history.
Test 3	Boundary Value	Student attempts to book a workspace for a time slot in the past	Select yesterday's date and submit booking request	System displays: "Cannot book time slots in the past"	Booking is accepted and added to calendar, and slot is labelled booked under the student's name	Fail	Added date validation in booking handler to compare selected time with current timestamp. Past dates are now disabled in the booking calendar picker.
Test 4	Boundary Value	Book final available slot on a machine	Last available slot on selected workspace	Booking should be confirmed;	Booking was confirmed under the uses name	Pass	N/A
Test 5	Path Coverage	Teacher books without needing	Teacher books an available slot on a	Booking Approved immediately without approval	Booking was confirmed immediately, and status was	Pass	N/A

		approval	workspace	process check	updated to booked – approved		
Test 6	Faulty Data	Attempt to upload invalid file as slot documentation	Upload .exe file to machine notes	Not visible as a file format to import	Unable to view and select any .exe file importing into the slot documentation	Pass	N/A
Test 7	Path Coverage	Admin creates machine with missing required fields	Machine name field left blank	System prevents creation and prompts for required info	Machine creation blocked with error: “Machine name required”	Pass	N/A
Test 8	Faulty data	Attempt to access different user profile	Manually adjust url to /user/1	Display forbidden error message	Successfully displayed forbidden error	Pass	N/A
Test 9	Faulty data	Change password requiring current password	Type in incorrect current password and set new password	Fail message displayed, “enter correct current password”	Successfully displayed error and didn’t allow user to reset the password	Pass	N/A
Test 10	Abnormal data	Displaying long name or descriptions for workspaces	Edit workspace to have a long 20+ character name and description	Display partial name and description displayed, without upsetting html formatting	No conflict with html formatting and displaying partial string	Pass	N/A
Test 11	Boundary Value	Deleting workspace deletes all its booking slots	Make a booking for a workspace, delete the workspace	Booking should disappear from user dashboard	Booking successfully disappeared and deleted without issues	Pass	N/A
Test 12	Path Coverage	Modifying a rejected student booking	As a student, modify a rejected	Approval status should change to pending and be	Approval successfully changed to pending and	Pass	N/A

		results to approval status to pending	booking	displayed	requires new approval		
Test 13	Path Coverage	Admin may view all bookings from all users in workspace tables	Admin clicks on bookings made by other users to view and edit details	Admin should be allowed to view and access all the bookings displayed	Admin successfully accessed all bookings made by other users	Pass	N/A
Test 14	Boundary Value	Deleting skills removes them from workspaces, users and join tables	Delete a skill from the skill management table that exists for a user and a workspace	Skills deletes without causing disruption to the workspace or user profiles	Successfully deleting from join tables causing no errors	Pass	N/A
Test 15	Abnormal data	Can't update forms after logging out	Open an edit booking page and log out via a different browser tab. Then attempt to update booking	User is redirected to log in page before data is submitted	User successfully redirected to the log in page.	Pass	N/A

Project showcase

GitHub repository

Here is the link the GitHub repository for the FreeSpace web application:



<https://github.com/SHildebrandt4472/FreeSpace?tab=readme-ov-file>

Evaluation

Project Reflection

FreeSpace is a web application designed for schools to manage their makerspaces. FreeSpace helps teachers coordinate and control student access to tech workspaces for personal use. Inspired by university makerspaces, the application allows students to book specific workspaces/equipment to complete curriculum projects or personal projects.

The main objective of this project was to build a professional looking web application that is intuitive to use. A key goal was to learn how to use Bootstrap to enhance visual appearance and user experience.

The most successful aspect of the project was the final user interface. Even though it was simple in design the final product looked professional and is very user friendly. Another major success was the use of SQLAlchemy to model the database tables as python classes instead of having to hand code SQL statements.

One of the main difficulties within the project was learning how to use Bootstrap, as it was more complicated than initially predicted and proved to be a very time-consuming process. Moving from traditional CSS, where styles are shared across HTML elements to inserting Bootstrap classes in line for each HTML element made it difficult to keep a consistent look throughout. The HTML code also becomes far less readable and more difficult to maintain.

Another challenge was handling the navigation between pages, particularly when trying to go back to a previous page after arriving from different pages.

Overall, the sheer amount of code required for a seemingly simple application quickly became overwhelming. Even though a lot of the code is fairly straightforward, FLASK seems to perform a lot of "magic" in the background, and it was often difficult to find exactly which line was causing errors and why.

An area for further development of FreeSpace could be its expansion beyond technical makerspaces to manage other spaces in a school. With the addition of a workspace category field, the application could be expanded to include MusicSpaces (such as practice rooms and recording equipment), ArtSpaces, GymSpaces, etc. Additionally adding a school table and school_id's to the other tables, FreeSpace could be expanded for use by multiple schools from the same application server.

Throughout the project, teacher feedback was especially valuable in the early stages, helping to define realistic goals. Ongoing guidance, particularly with documentation, ensured clear communication of both functional and technical aspects.

Overall, the project provided a valuable learning experience in full-stack development, offering a better understanding of the complexities of front-end design as well as manipulating data in a relational database. It was, however, more complex than originally perceived. The front-end design aspect was very time consuming particularly when it came to handling the responsive design aspects of HTML. Bootstrap helped with this, but the steep learning curve was the most time-consuming aspect of the project. Front-end design, whilst rewarding is very frustrating and is an area that would require more learning to master. This project overall has developed a real appreciation for just how much is required to develop a large-scale commercial application such as Seqta.