

INF2001

Introduction to Software Engineering



Milestone 3:

Design, Test Plan & Prototype

for

<Airline Pilots>

Prepared by

Team's Name: FLYSG AIRLINES

Ivan Ng Say Mun	2003108
Carmen Wong Jiawen	2201680
Kenan Puay-Tee Tan	2100851
Casanas Joseph Christopher	2202186
Cortes	

Lab Group: P9-6

Github handle: <https://github.com/denmire/projects/1/views/1>

Date: 25 November 2023

Table of Contents

1	<i>Introduction.....</i>	2
1.1	References and Acknowledgments.....	3
2	<i>Software Design.....</i>	3
2.1	Architecture Design.....	4
2.2	Detailed Design.....	4
3	<i>Testing.....</i>	4
3.1	Black Box Testing.....	4
3.2	White Box Testing.....	4
4	<i>Project Prototype & Wireframes.....</i>	5
5	<i>Project Management.....</i>	5

1 Introduction

Scheduling is vital for businesses and manual scheduling can be time-consuming and expensive. Airline Pilot aims to optimize and provide a time efficient solution for scheduling a pilots job assignment. Airline Pilot is a work allocation management system that will be used by FLYSG AIRLINE company. The intended purpose is to enhance the work life balance among the pilots, while providing an efficient work allocation process that reduces the workload of FLYSG AIRLINE's management team. In this section, the project scope, related background literature, intended audience and document overview will be discussed.

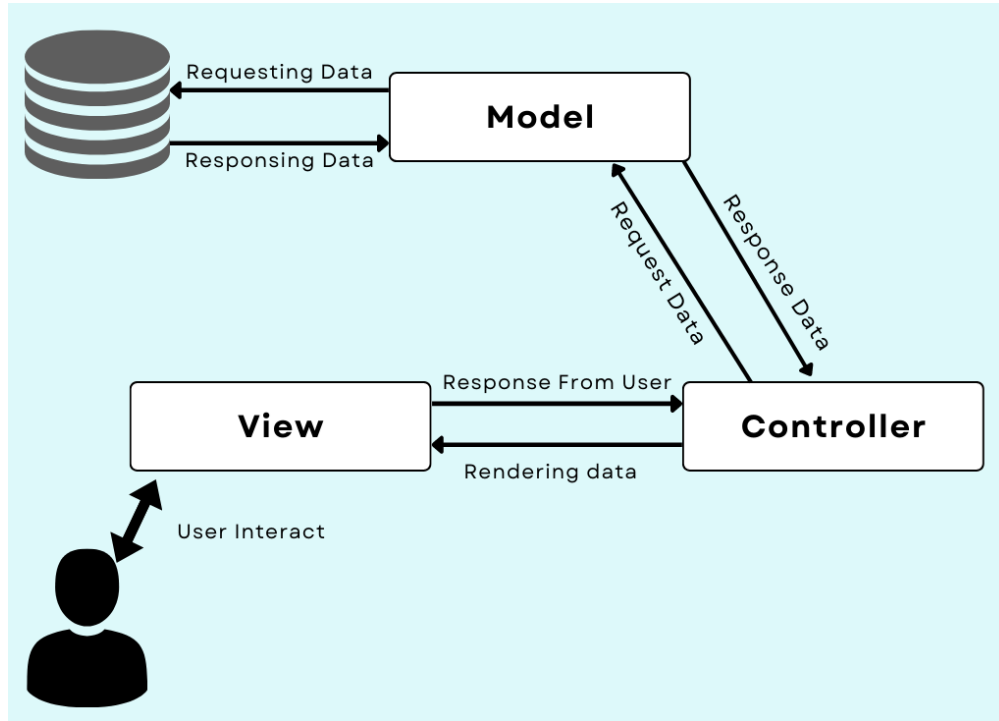
1.1 References and Acknowledgments

[1] A. Kasirzadeh 1, M. Saddoune, and F. Soumis, "Airline Crew Scheduling: Models, algorithms, and data sets," *EURO Journal on Transportation and Logistics*, <https://www.sciencedirect.com/science/article/pii/S2192437620300820#sec2> (accessed Sep. 23, 2023).

[2] L. Software, "Leon for scheduled operators," *Aircraft Management Software - Scheduled Operators*, <https://www.leonsoftware.com/leon/scheduled-operators.html> (accessed Aug. 4, 2023).

2 Software Design

2.1 Architecture Design



Our team has elected to adopt the MVC (Model-View-Controller) architecture for its numerous strategic benefits, which align with our project goals. This architectural pattern significantly enhances parallel development capabilities, allowing different components of the application to be worked on simultaneously by various team members. A developer can focus on refining the data access logic in the Model, while another can concurrently design and implement the user interface within the View, without one activity blocking the other. This not only optimizes development time but also leverages the diverse skill sets within our team, as developers can focus on their areas of expertise or interest.

Furthermore, the MVC framework promotes maintainability through its inherent organizational structure, which clearly separates the business logic from the user interface. This separation of concerns ensures that our codebase is not only well-organized but also more adaptable to change. When business requirements evolve, as they inevitably do, the impact of changes is localized, reducing the risk of unintended side effects. Maintenance becomes a less cumbersome task, and the introduction of new functionalities, or the modification of existing ones, can be executed with surgical precision.

Upgrading the system is also streamlined with MVC architecture. The decoupling of the application's core aspects means that upgrades or changes to one component, such as the View for a UI overhaul, can be carried out independently of the Model or Controller. This not only simplifies

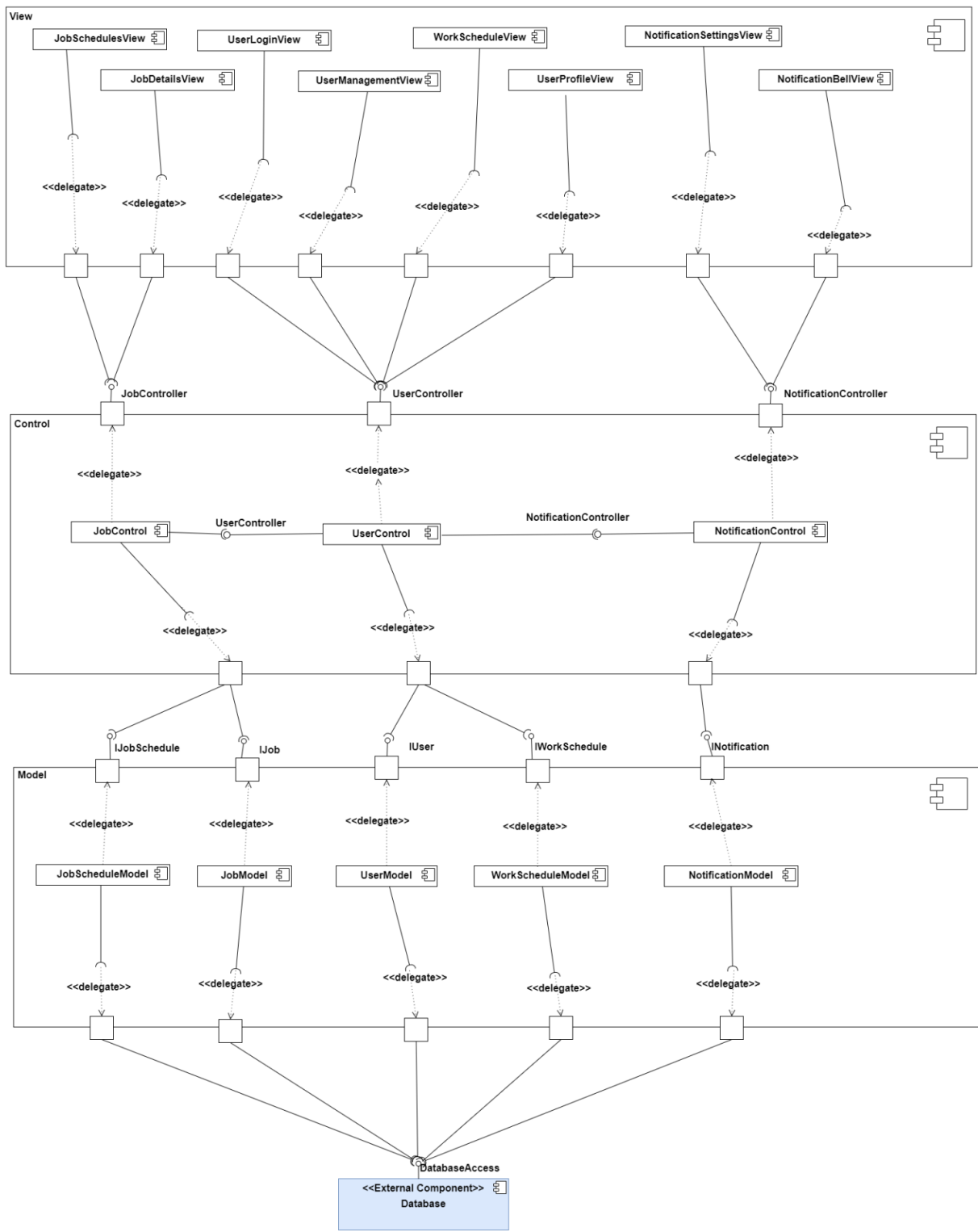
the upgrade process but also allows for continuous deployment of improvements and new features without significant downtime or complete system overhauls.

In the context of testing, MVC provides a robust framework for more efficient and effective quality assurance. By isolating the business logic, data access, and presentation layers, we can implement unit testing and debugging procedures on discrete sections of the application. This modularity in testing leads to quicker issue identification and more reliable validation of each component's performance.

Moreover, the MVC architecture provides us with great flexibility in terms of design and user experience. It allows for multiple Views to be associated with a single Model, facilitating the creation of different user interfaces that can cater to various user devices or preferences without duplicating business logic. This not only enhances the user experience but also future-proofs the application by making it adaptable to new and emerging platforms.

Finally, MVC supports the principles of reusability and interoperability. Components developed within the MVC structure, especially the business logic within the Model, can often be reused across different systems or modules within the same application. This promotes a DRY (Don't Repeat Yourself) development approach, leading to a more efficient development process and a codebase that is easier to extend and maintain.

Through the integration of the MVC architecture, our project is positioned to gain benefits in collaborative development, maintainability, testability, and future scalability. This foundation ensures our application remains robust, flexible, and easy to adapt, keeping pace with the evolving needs of our users and the growth objectives of our organization.



View Layer:

The View layer consists of several user interfaces (UIs) that the end-user will directly interact with. This includes JobSchedulesView for displaying job schedules, UserLoginView for handling user authentication, WorkScheduleView for presenting work schedules, NotificationSettingsView for managing notification preferences, JobDetailView, UserManagementView, UserProfileView, and NotificationDetailView. Each of these Views is designed to display information to the user and capture user inputs.

Control Layer:

The Control layer, or Controller, contains the logic that responds to user interactions from the View. In this layer, we have JobController, UserController, and NotificationController. Each Controller is tasked with handling the input from its respective Views, processing that input, and making calls to the Model layer to retrieve or update data. For instance, JobController would manage the input from JobSchedulesView and JobDetailView, UserController from UserLoginView, UserManagementView, and UserProfileView, and similarly, NotificationController manages the input from NotificationSettingsView and NotificationDetailView.

Model Layer:

The Model layer is responsible for managing the data and the business logic of the application. It contains the JobScheduleModel, JobModel, UserModel, WorkScheduleModel, and NotificationModel. Each Model corresponds to a specific domain within the application and is in charge of handling data operations, such as querying the database, performing business logic operations, and providing data back to the Controller.

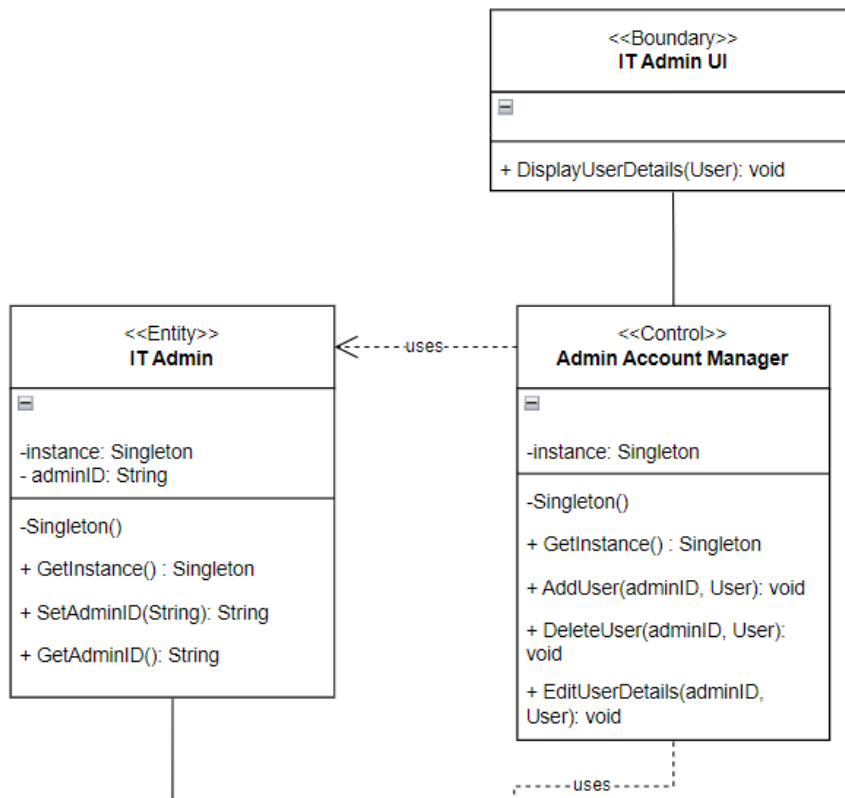
Database Access:

The DatabaseAccess component signifies the system's interaction with the underlying database where it persists data.

Data Flow:

A user interaction begins at the View layer, where the user's input or request is captured. This input is then passed to the corresponding Controller, which handles the request. For example, if a user wants to edit their profile, the UserProfileView passes this request to the UserController. The Controller processes the request and interacts with the Model to retrieve or update data. The UserController, upon receiving the edit request, would instruct the UserModel to update the user information. The Model performs the necessary data manipulation by interacting with the database through DatabaseAccess. Once the Model updates the data, it communicates the changes back to the Controller, which then updates the View. Consequently, the user would see their updated profile on the UserProfileView.

2.2 Detailed Design



Rationale for Implementing Singleton on the Entity (IT Admin):

Centralized Management of Shared Resources:

The IT Admin entity likely represents a shared resource across the application, such as system configurations or administrative privileges. By implementing the Singleton pattern, we ensure that there is a single, consistent point of reference for these configurations, preventing potential conflicts or inconsistencies that might arise from having multiple instances.

Consistency and Integrity:

With a Singleton, any changes made to the settings or data within the IT Admin entity are immediately reflected throughout the application. This is critical for maintaining data integrity and ensuring that all parts of the system act on the most current information.

Controlled Access and Security:

Having a single instance of the IT Admin entity allows for stricter control over access and modifications. It simplifies the enforcement of security measures since there is only one instance to monitor and protect against unauthorized access or changes.

Rationale for Implementing Singleton on the Control (Admin Account Manager):

Centralized Business Logic:

The Admin Account Manager control class is responsible for critical business operations such as user account management. Implementing it as a Singleton ensures that business rules and logic are applied consistently, as all operations are routed through the same instance.

Resource Efficiency:

By using a Singleton, we can optimize resource usage since a single instance of the Admin Account Manager can manage all requests, reducing the overhead associated with creating and disposing of multiple instances.

Synchronized Operations:

In scenarios that require synchronization (like creating or deleting user accounts), a Singleton can be beneficial. It simplifies the synchronization logic because the operations are centralized in a single instance, reducing the complexity of concurrency control.

Reasons for Not Implementing Singleton on the UI (IT Admin UI):**User Experience and Flexibility:**

User interfaces often need to be dynamic and responsive to user actions. Implementing the Singleton pattern can reduce the flexibility needed for a rich user experience, where multiple windows or dialogs might need to be open simultaneously.

State Management Complexity:

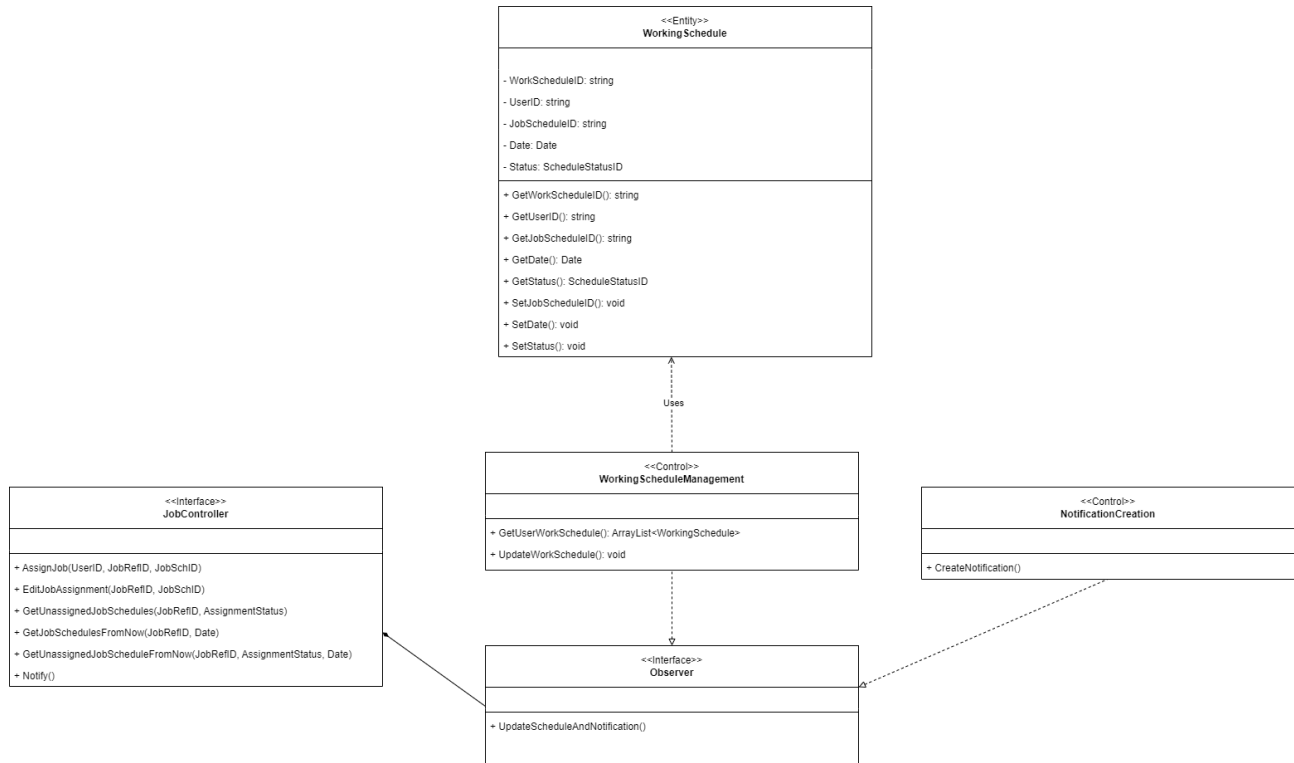
UI components typically maintain state that is specific to user interactions. Singleton UI components can make state management more complex, especially when dealing with asynchronous user actions or multi-threaded environments.

Testing and Maintenance Challenges:

Singletons introduce global state, which can make unit testing and maintenance more challenging. For UI components, this can lead to tests that are dependent on the state from previous tests, thereby making the tests less reliable and the code harder to maintain.

Framework Constraints:

Many UI frameworks are designed with their own lifecycle management for UI components, making it unnecessary or even problematic to enforce a Singleton pattern. These frameworks often have built-in mechanisms to ensure the main application window is unique without the need for a Singleton.



Implementation of the Observer Design Pattern

In pursuit of a more resilient and adaptable system for interactions between closely linked objects, our team has implemented the Observer design pattern in the context of schedule updates. This pattern is a cornerstone in behavioral design techniques, pivotal for enhancing the independence of object communication and facilitating a more modular design structure.

Challenges of Strong Coupling

The initial architecture of our system was constrained by a strong coupling between the JobSchedule and WorkingSchedule entities. This rigid interdependency proved to be a bottleneck, requiring any alteration in JobSchedule to be mirrored by WorkingSchedule, and the inverse was equally true.

The WorkingScheduleManagement class's dual responsibilities for updating schedules and handling notifications exemplified the system's intertwined nature. This setup led to a scenario where changes to one component could have unwarranted repercussions on another, thereby complicating maintenance and updates.

Integration of the Observer Pattern

Our solution to this interdependency was the integration of the Observer pattern. By adapting the JobController class to incorporate an observer interface, we were able to establish a notification mechanism that informs the relevant WorkingScheduleManagement and NotificationCreation control classes about changes indirectly.

Post the integration of the notify() function in the JobController interface, the JobSchedule updates are communicated indirectly, which mitigates the direct impact on WorkingSchedule and Notification classes. This layer of abstraction ensures that updates in JobSchedule can be made independently of WorkingSchedule, significantly enhancing the system's flexibility and ease of maintenance.

3 Testing

3.1 Black Box Testing

Use Case 13 - Allocate Job

<i>Causes</i>		<i>Values</i>	<i>1 - 4</i>	<i>5 - 6</i>	<i>7</i>	<i>8</i>
<i>C1</i>	<i>Pilot Is Available</i>	<i>Y/N</i>	<i>N</i>	<i>Y</i>	<i>Y</i>	<i>Y</i>
<i>C2</i>	<i>Pilot Meets Rank Requirement</i>	<i>Y/N</i>	<i>-</i>	<i>N</i>	<i>Y</i>	<i>Y</i>
<i>C3</i>	<i>Pilot Has Sufficient Training</i>	<i>Y/N</i>	<i>-</i>	<i>-</i>	<i>N</i>	<i>Y</i>
<i>Effects</i>						
<i>E1</i>	<i>Allocatable</i>					<i>x</i>
<i>E2</i>	<i>Not Allocatable</i>		<i>x</i>	<i>x</i>	<i>x</i>	

Test accounts should be prepared with the following settings:

Employee1

Rank: Captain

Completed Training for Boeing 737-8: Yes

Completed Training for Boeing 787-10: Yes

Completed Training for Airbus A350-900: No

Employee2

Rank: Senior First Officer

Completed Training for Boeing 737-8: Yes

Completed Training for Boeing 787-10: Yes

Completed Training for Airbus A350-900: No

Employee3

Rank: Trainee

Completed Training for Boeing 737-8: No

Completed Training for Boeing 787-10: No

Completed Training for Airbus A350-900: No

Test Case	Pre-Condition	Steps	Data	Expected Result	Actual Result	Pass/Fail
Check response of allocating valid main pilot	Manager is logged into the system Job has been created on the schedule Employee1 account indicated availability on selected date	1. Click on Job 2. Click on Allocate Pilot button 3. Choose Pilot 4. Click on Submit button	Aircraft: Boeing 737-8 Main Pilot: Employee1	Success	Success	Pass
Check response of allocating valid assistant pilot	Manager is logged into the system Job has been created on the schedule Employee2 account	1. Click on Job 2. Click on Allocate Pilot button 3. Choose Pilot 4. Click on	Aircraft: Boeing 737-8 Assistant Pilot: Employee2	Success	Success	Pass

	<i>indicated availability on selected date</i>	<i>Submit button</i>				
<i>Check response of allocating invalid main pilot</i>	<i>Manager is logged into the system</i> <i>Job has been created on the schedule</i> <i>Employee2 account indicated availability on selected date</i>	1. Click on Job 2. Click on Allocate Pilot button 3. Choose Pilot 4. Click on Submit button	Aircraft: Boeing 737-8 Main Pilot: Employee2	<i>Fail</i>	<i>Fail</i>	<i>Pass</i>
<i>Check response of allocating invalid assistant pilot</i>	<i>Manager is logged into the system</i> <i>Job has been created on the schedule</i> <i>Employee3 account indicated availability on selected date</i>	1. Click on Job 2. Click on Allocate Pilot button 3. Choose Pilot 4. Click on Submit button	Aircraft: Boeing 737-8 Assistant Pilot: Employee3	<i>Fail</i>	<i>Fail</i>	<i>Pass</i>

3.2 White Box Testing

```
1 public List<Pilot> RecommendPilot(Job){
2     String aircraft = Job.GetAirCraft();
3     List<Pilot> pilots = new List<Pilot>();
4     List<String> ranks = new list<String>();
5     Switch (aircraft){
6         case "boeing 737-8":
7             ranks.add("captain");
8             ranks.add("senior first officer");
9         case "boeing 787-10":
10            ranks.add("captain");
11            ranks.add("senior first officer");
12            ranks.add("first officer");
13        case "boeing A350-900":
14            ranks.add("captain");
15            ranks.add("senior first officer");
16            ranks.add("first officer");
17        default:
18            return pilots;
19    }
20    pilots.AddRange(GetPilots(ranks));
21    foreach(Pilot p in pilots){
22        if(p.CheckTraining(aircraft))
23            continue
24        else
25            pilots.remove(p);
26    }
27    return pilots;
28 }
```

Figure 1: White Box Testing Code

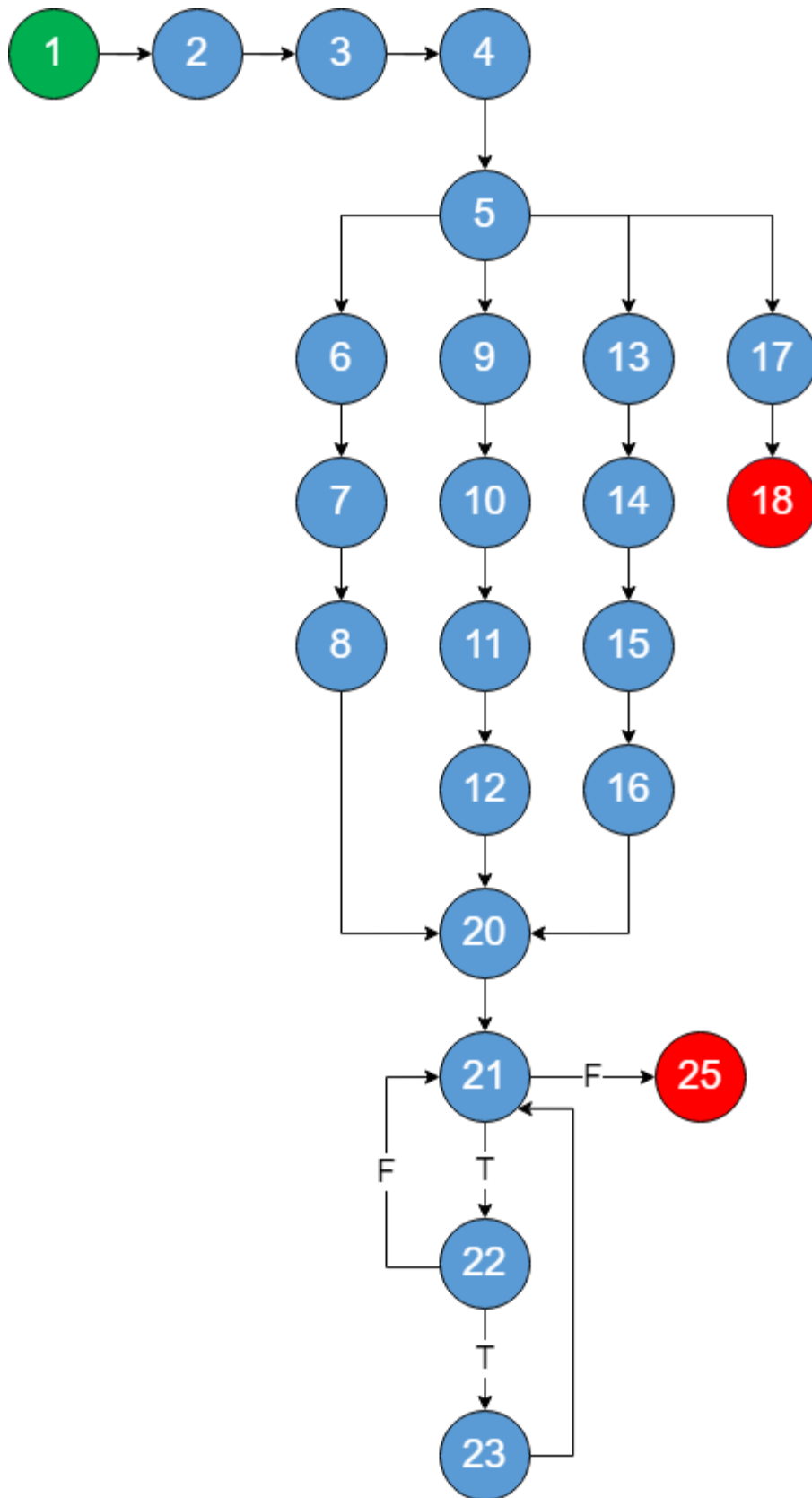


Figure 2: Basis Path Testing

Cyclomatic Complexity

$$M = E - N + 2P$$

where E represents number of edges, N represents number of nodes, P is a constant value of 1

$$M = 26 - 23 + 2(1) = 5$$

Basis Path, $M - 1$ = minimum 4 paths and test cases

Paths

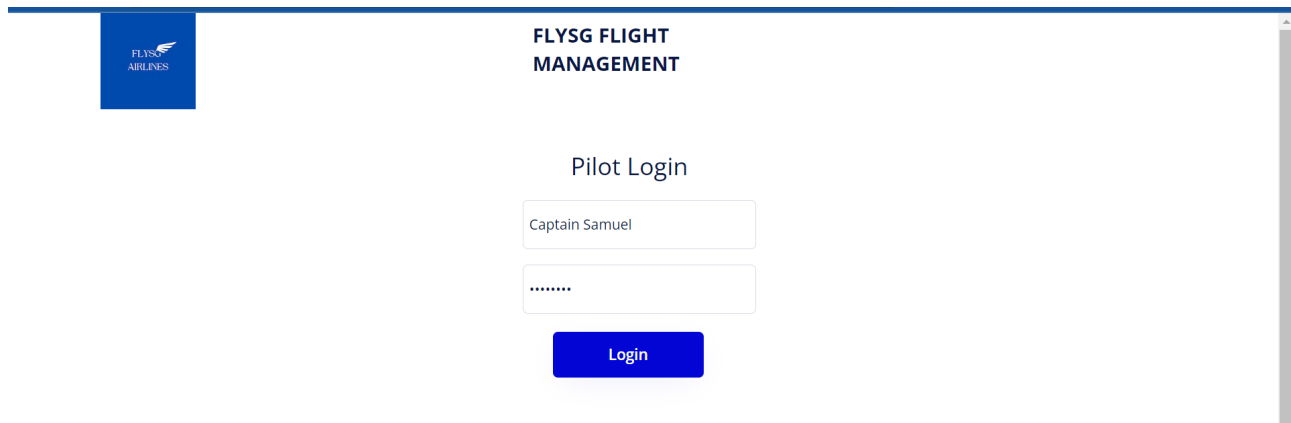
1. 1-2-3-4-5-6-7-8-20-21-22-23-21-25
2. 1-2-3-4-5-9-10-11-12-20-21-22-23-21-25
3. 1-2-3-4-5-13-14-15-16-20-21-22-23-21-25
4. 1-2-3-4-5-17-18

Case Number	Input at entry	Input at 2	Expected Output	Actual Output	Success/Fail
1	Job	aircraft = "boeing 737-8"	{Employee1, Employee2}	{Employee1, Employee2}	Success
2	Job	aircraft = "boeing 787-10"	{Employee1, Employee2}	{Employee1, Employee2}	Success
3	Job	aircraft = "airbus A350-900"	{Employee1, Employee2}	{Employee1, Employee2}	Success
4	Job	aircraft = ""	{}	{}	Success

4 Project Prototype & Wireframes

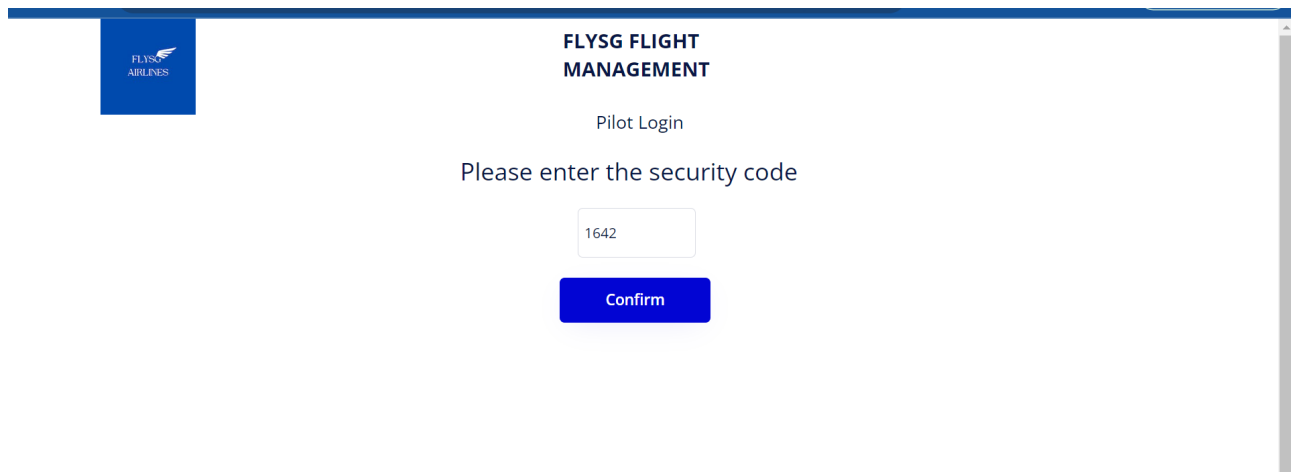
UC-1 : Validate Account

Users will only have access to features after validating their accounts via a logging in and logging out system with two-factor authentication.



The wireframe shows a web page for 'FLYSG FLIGHT MANAGEMENT'. On the left is a blue logo with 'FLYSG AIRLINES' and a wing icon. The page title 'FLYSG FLIGHT MANAGEMENT' is at the top center. Below it is the heading 'Pilot Login'. There are two input fields: the first contains 'Captain Samuel' and the second contains seven dots. A blue 'Login' button is positioned below the password field.

Figure 2: Pilot Login Page



The wireframe shows a web page for 'FLYSG FLIGHT MANAGEMENT'. On the left is a blue logo with 'FLYSG AIRLINES' and a wing icon. The page title 'FLYSG FLIGHT MANAGEMENT' is at the top center. Below it is the heading 'Pilot Login'. The text 'Please enter the security code' is centered. Below it is an input field containing '1642'. A blue 'Confirm' button is positioned below the security code field.

Figure 3: Pilot 2FA page

Pilots will receive their 2FA code upon successful login and be prompt to enter it as shown in figure 3.

FLYSG AIRLINES

FLYSG FLIGHT
MANAGEMENT

Manager Login

Bob

.....

L

Suggest strong password...

Use passwords saved in your Google Account

Figure 4: Manager Login Page

FLYSG AIRLINES

FLYSG FLIGHT
MANAGEMENT

Manager Login

Please enter the security code

2537

Confirm

Figure 5: Manager 2FA Page

Similarly upon successful login, the manager will receive a 2FA code and will be prompt to enter it as shown in figure 5.

UC-2 : View Dashboard

For both pilots and managers, they will have access to their dashboard after logging in using their account.

The Pilot Dashboard interface features a top navigation bar with a notification bell icon and the text "Update Jan Schedule by 1 Dec 2023". The main header area includes the title "Pilot Dashboard", a welcome message "Welcome, Captain Samuel", and two buttons: "Reset Password" and "Logout". The dashboard is divided into three main sections: "Flight Schedule", "Flight Allocation", and "Performance". The "Flight Schedule" section contains a calendar grid for November 2023, with a "Search" button and a "Month" dropdown menu. The "Flight Allocation" section includes a "Select Unavailable Dates" input field with the date "11/19/2023", and two buttons: "Add more dates" and "Submit Dates". The "Performance" section has a "Select Month" input field and a "Search" button.

Figure 6: Pilot Dashboard

The Manager Dashboard interface features a top navigation bar with a "Reset Password" button and a "Logout" button. The main header area includes the title "Manager Dashboard" and a welcome message "Welcome, Bob!". The dashboard is divided into three main sections: "Flight Schedule", "Flight Allocation", and "Performance". The "Flight Schedule" section contains a calendar grid for November 2023, with a "Select Date" input field showing "11/20/2023", a "Search" button, and a "Nov" dropdown menu. The "Flight Allocation" section includes a "Select Date" input field showing "11/20/2023", a "Destination" input field, a "Select Rank" dropdown menu, and a "Search" button. The "Performance" section has a "Select Month" input field, a "Select Rank" dropdown menu, and a "Search" button.

Figure 7: Manager Dashboard

UC-3 Search and filter

This use case includes a search and filtering functionality on the website. The search results will only show pilots since they are the focus of the system. IT Admins will have this functionality so that he will be able to manage accounts.

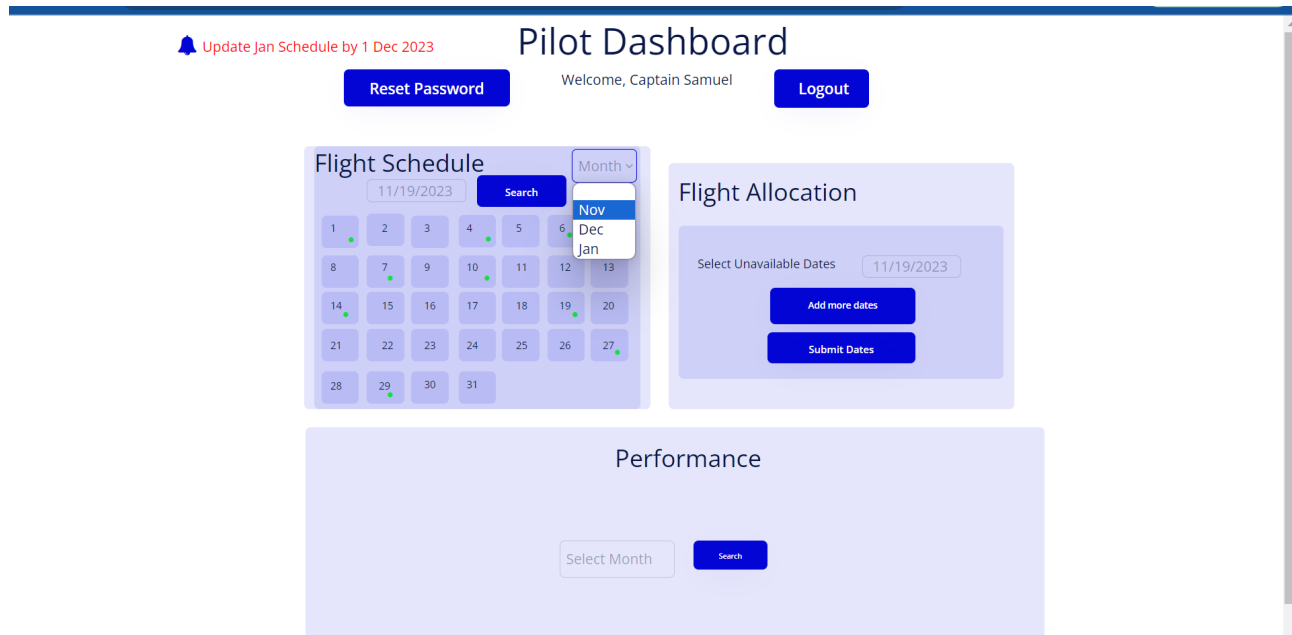


Figure 8.1: Pilot Dashboard

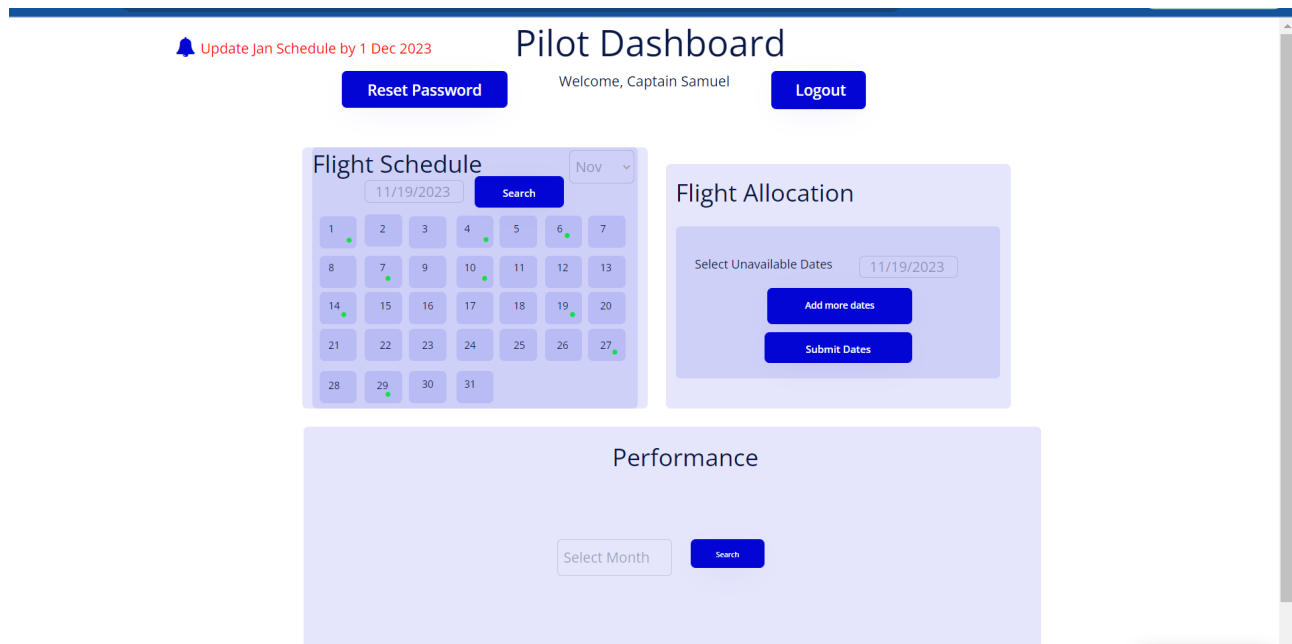


Figure 8.2: Pilot Dashboard

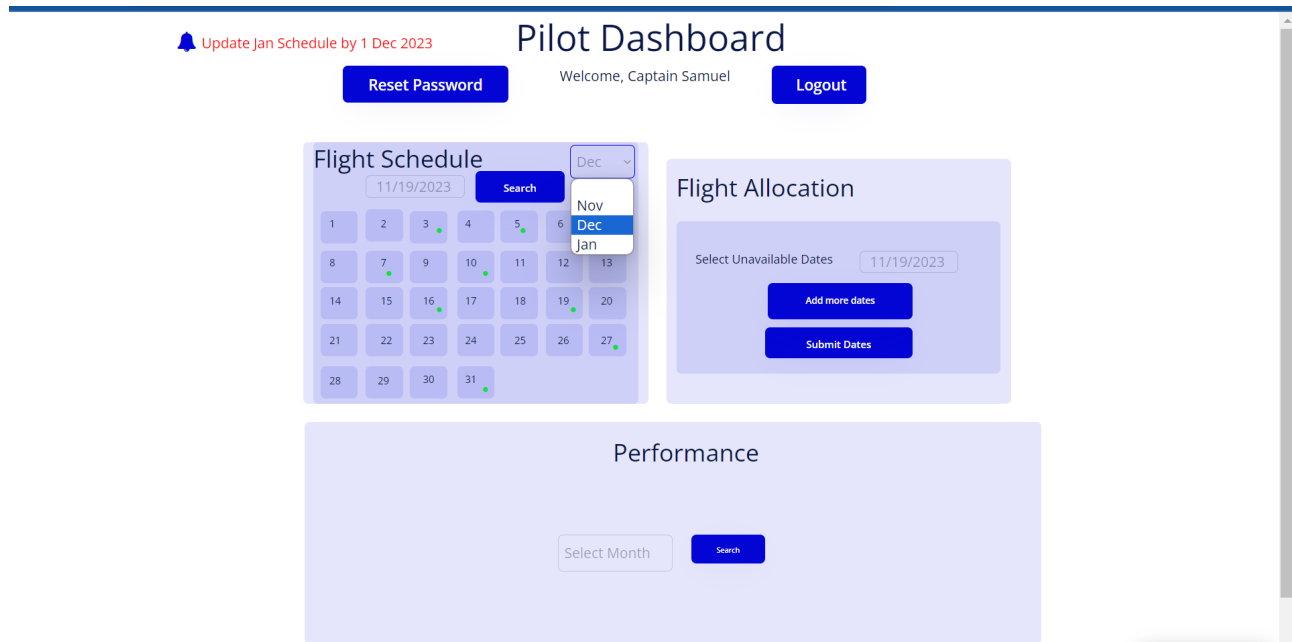


Figure 8.3: Pilot Dashboard

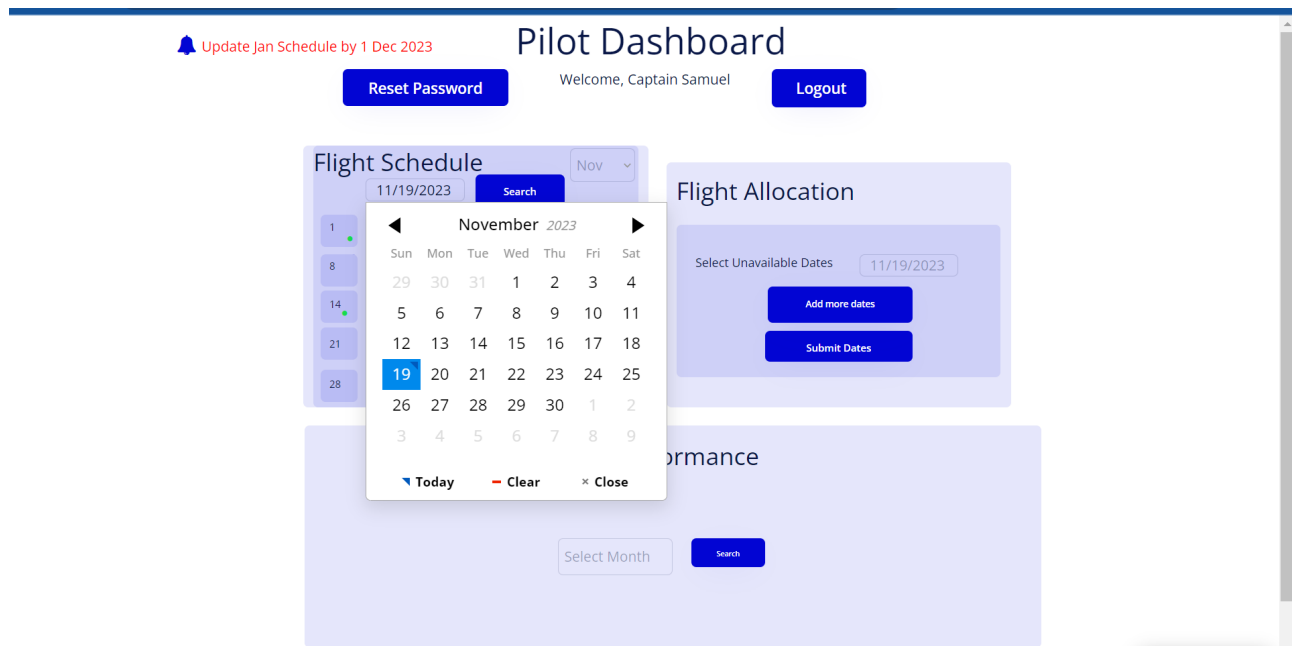


Figure 8.4: Pilot Dashboard

The pilot gets to filter the month of their flight schedule, and/or narrows down to the particular date according to their preference as shown in figure 8.1 to 8.4.

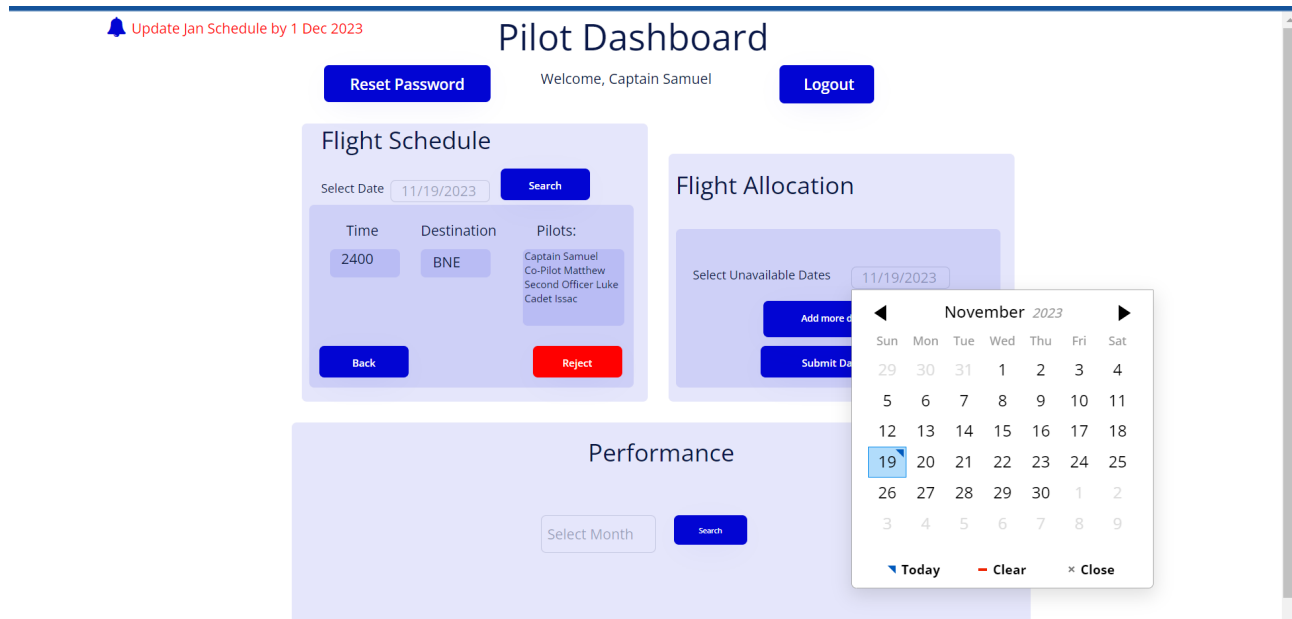


Figure 8.5: Pilot Dashboard

The pilot also gets to choose the date via the calendar drop box when indicating their unavailable date. The pilot gets to add more than 1 date, by clicking on the 'Add more dates' button which will be further discussed in the later use-case.

UC-4 & 5 : View job details and allocation

In use-case 4 and 5, the job assignments and Job details can be viewed on the dashboard respectively.

The screenshot displays the 'Pilot Dashboard' for Captain Samuel. At the top, there is a notification bell icon with the text 'Update Jan Schedule by 1 Dec 2023'. The dashboard includes a 'Reset Password' button, a 'Welcome, Captain Samuel' message, and a 'Logout' button. The main content area is divided into three sections: 'Flight Schedule', 'Flight Allocation', and 'Performance'. The 'Flight Schedule' section features a 'Select Date' dropdown set to '11/19/2023' and a 'Search' button. Below this, there is a table with columns for 'Time', 'Destination', and 'Pilots'. The first row shows '2400' for Time, 'BNE' for Destination, and a list of pilots: 'Captain Samuel', 'Co-Pilot Matthew', 'Second Officer Luke', and 'Cadet Isaac'. There are 'Back' and 'Reject' buttons at the bottom of this section. The 'Flight Allocation' section has a 'Select Unavailable Dates' dropdown set to '11/19/2023' and buttons for 'Add more dates' and 'Submit Dates'. The 'Performance' section includes a 'Select Month' dropdown and a 'Search' button.

Figure 9: Pilot Job Details

As shown in figure 9, the pilot will be able to view the job details upon selecting the month/date which includes the time, destination and the other pilots that are allocated to the flight as well as the 'Reject' button to reject the flight allocation due to any last minute changes that will be further discussed in the following use-case.

The screenshot displays the 'Manager Dashboard' for Bob. At the top, there is a 'Reset Password' button, a 'Welcome, Bob!' message, and a 'Logout' button. The main content area is divided into three sections: 'Flight Schedule', 'Flight Allocation', and 'Performance'. The 'Flight Schedule' section features a 'Select Date' dropdown set to '11/19/2023' and a 'Search' button. Below this, there is a table with columns for 'Time', 'Destination', and 'Pilots'. The first row shows '2400' for Time, 'BNE' for Destination, and a list of pilots: 'Captain Samuel', 'Co-Pilot Matthew', 'Second Officer Luke', and 'Cadet Isaac'. A red text link 'Captain Samuel Rejected Job. Click here to remove' is visible below the first row. The second row shows '0200' for Time, 'HND' for Destination, and an empty 'Pilots' column. The 'Flight Allocation' section has a '11/19/2023' dropdown, a 'Destination' dropdown, and a 'Select Rank' dropdown menu, with a 'Search' button below. The 'Performance' section includes a 'Select Month' dropdown, a 'Search' button, and a 'Select Rank' dropdown menu.

Figure 10: Pilot Job Details

Similarly, the manager shall be able to view the general flight schedule of the selected date as well as shown in figure 10. The details provided are similar to the pilot dashboard with the additional feature to remove the pilot from the schedule when the pilot rejects the job allocation which will be further explained in the next use-case.

UC-6: Reject Job

As mentioned earlier, the pilots are allowed to reject job assignments based on certain regulations such as reaching prior mutual agreement after discussion with manager. However, if they reject a job without first discussing it with his manager, the manager will approach the pilot and decide if they have a valid reason to do so.

The screenshot displays the 'Pilot Dashboard' interface. At the top, a notification bell icon is followed by the text 'Update Jan Schedule by 1 Dec 2023'. The dashboard title 'Pilot Dashboard' is centered, with 'Welcome, Captain Samuel' on the left and a 'Logout' button on the right. Below the title, there are three main sections: 'Flight Schedule', 'Flight Allocation', and 'Performance'. The 'Flight Schedule' section includes a 'Select Date' dropdown set to '11/19/2023' and a 'Search' button. Below this, there are three input fields labeled 'Time', 'Destination', and 'Pilots:'. A red notification 'Schedule Rejected' is visible at the bottom of this section. The 'Flight Allocation' section has a 'Select Unavailable Dates' dropdown set to '11/19/2023', with 'Add more dates' and 'Submit Dates' buttons below it. The 'Performance' section at the bottom has a 'Select Month' dropdown and a 'Search' button.

Figure 11: Pilot Rejected Job

Once the pilot rejects the job allocation, the 'schedule rejected' notification will appear to alert the pilot that their schedule has been rejected as shown in figure 11.

UC-7: Remove Job Assignment

If a job has been rejected, managers can remove the allocation to prevent cluttering the dashboard.

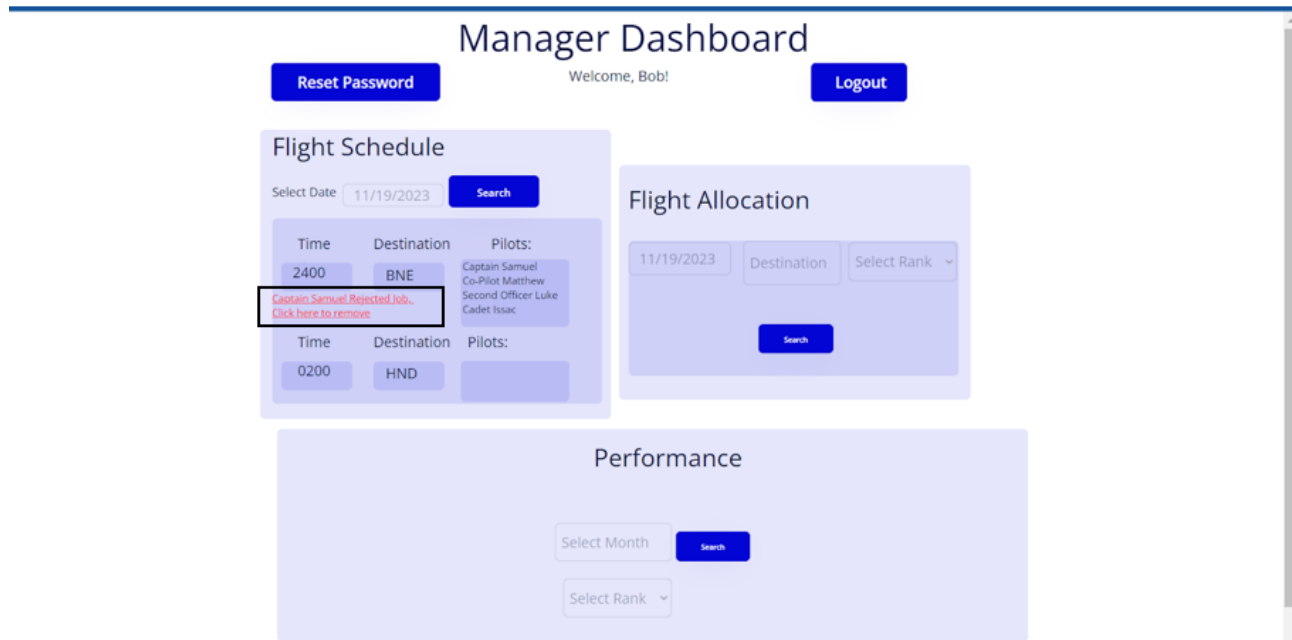


Figure 12: Manager Notified of Rejected Job

When the pilot clicks on the 'Reject' button, the manager will be notified through the red text as shown in figure 12 above.

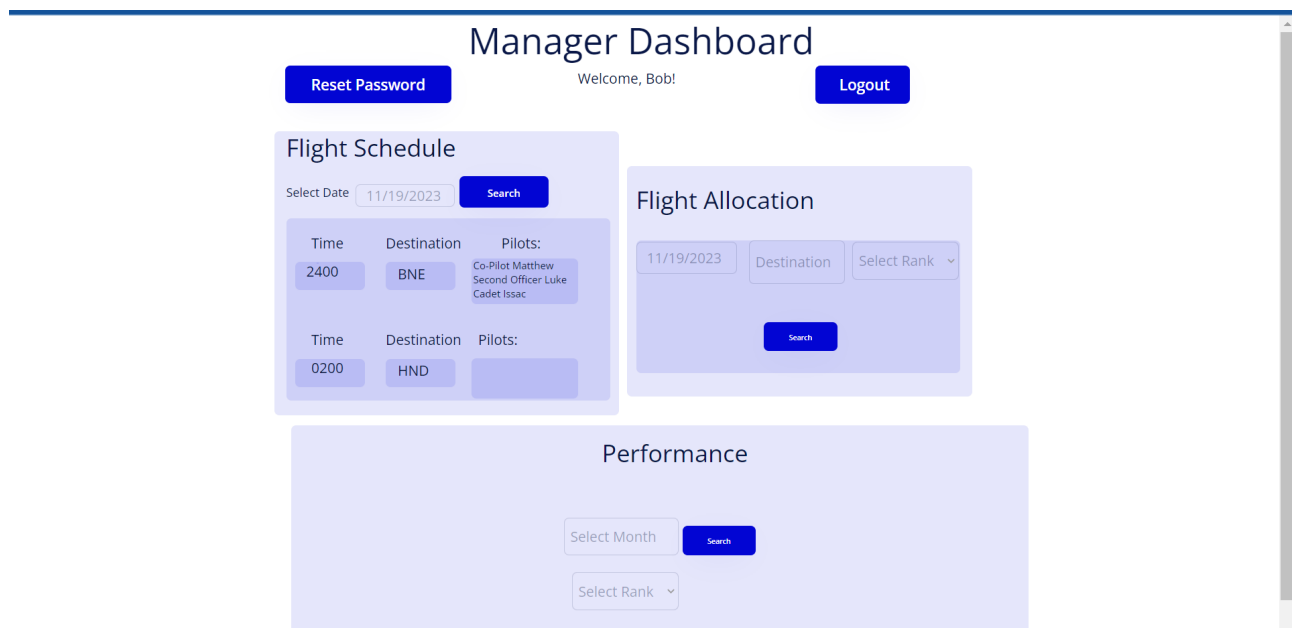


Figure 12.1: Manager Notified of Rejected Job

Once the manager approves of the rejected job, and clicks on red text as shown in figure 12, the pilot name will automatically be removed from the list of pilots, shown in figure 12.1.

UC-8 : Edit Personal Details

In this case, the IT admin will be able to update personal information for pilots if needed. Information like details of his last training, rank and personal information must be updated regularly to the IT admin's end by the pilots. On the pilot and manager end, they will be able to reset their password as shown in figure



Figure 13: Pilot Dashboard Reset Password Button

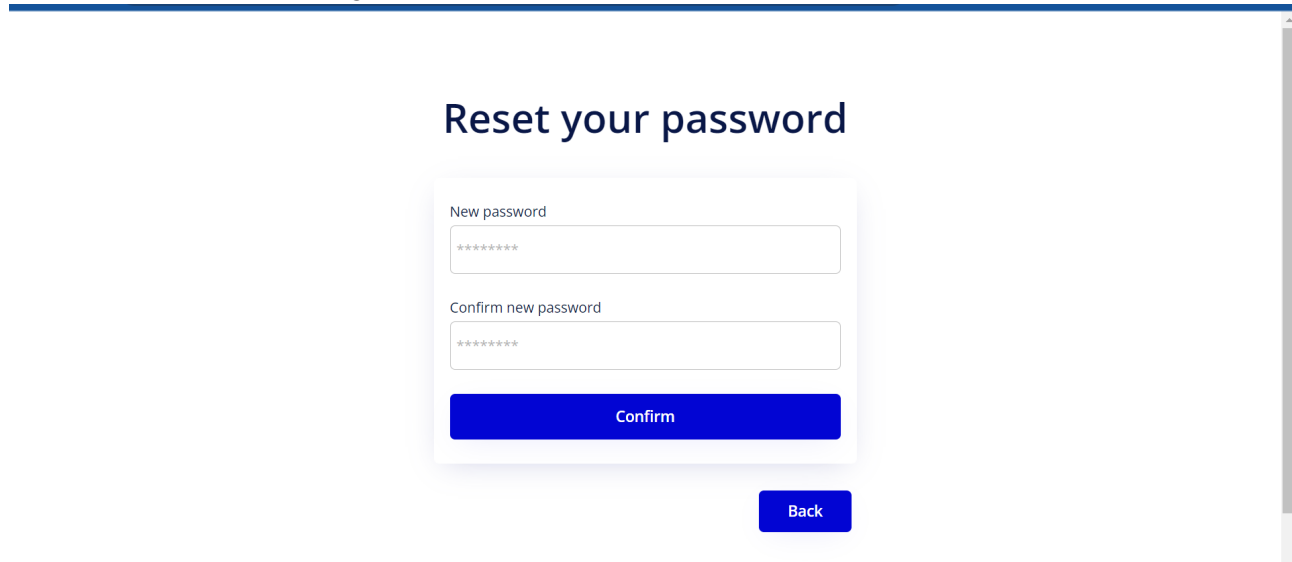


Figure 14: Pilot Dashboard Reset Password Page

When the pilot clicks on the reset password button, he or she will be prompt to update and new password, shown in figure 14.

Reset your password

New password

Confirm new password

Confirm

Password Changed

Back

Figure 15: Pilot Dashboard Password Changed

Once the password has been changed, the 'password changed' alert will appear, as shown in figure 15.

UC-9: View Notification

They should also be able to receive notifications so that they do not miss out on important updates.

Update Jan Schedule by 1 Dec 2023

Pilot Dashboard

Welcome, Captain Samuel

Logout

Reset Password

Figure 16: Pilot Dashboard Notification

Notification

Please submit your available dates by 1 December 2023. If there are any enquires, please contact one of the managers.

Back

Figure 17: Pilot Notification Page

When the pilot clicks on the notification text shown in figure 16, the pilot will be brought to the notification page for more information as shown in figure 17.

UC-10: Indicate Availability/Preference

In this use-case, the pilot will be able to indicate their availability to work.

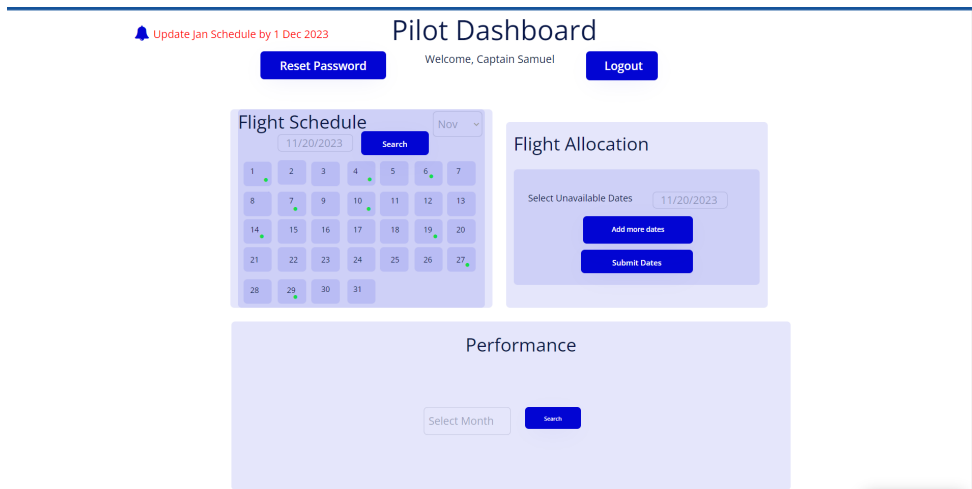


Figure 18: Pilot Dashboard

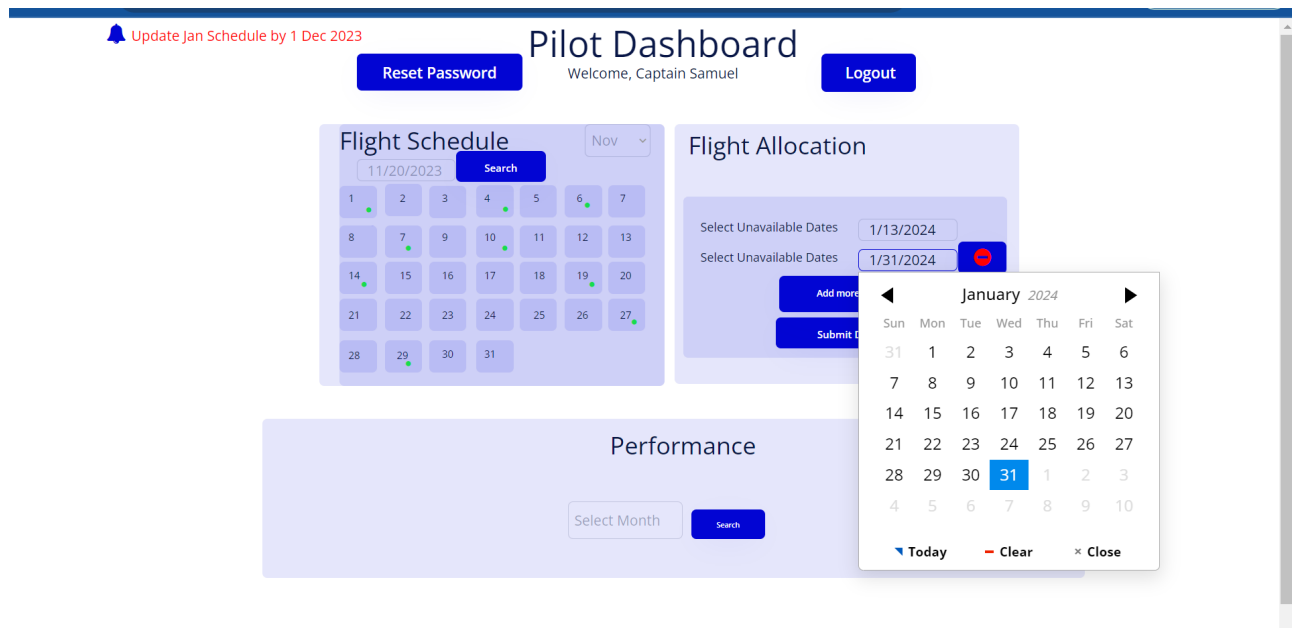


Figure 19: Pilot indicating unavailability

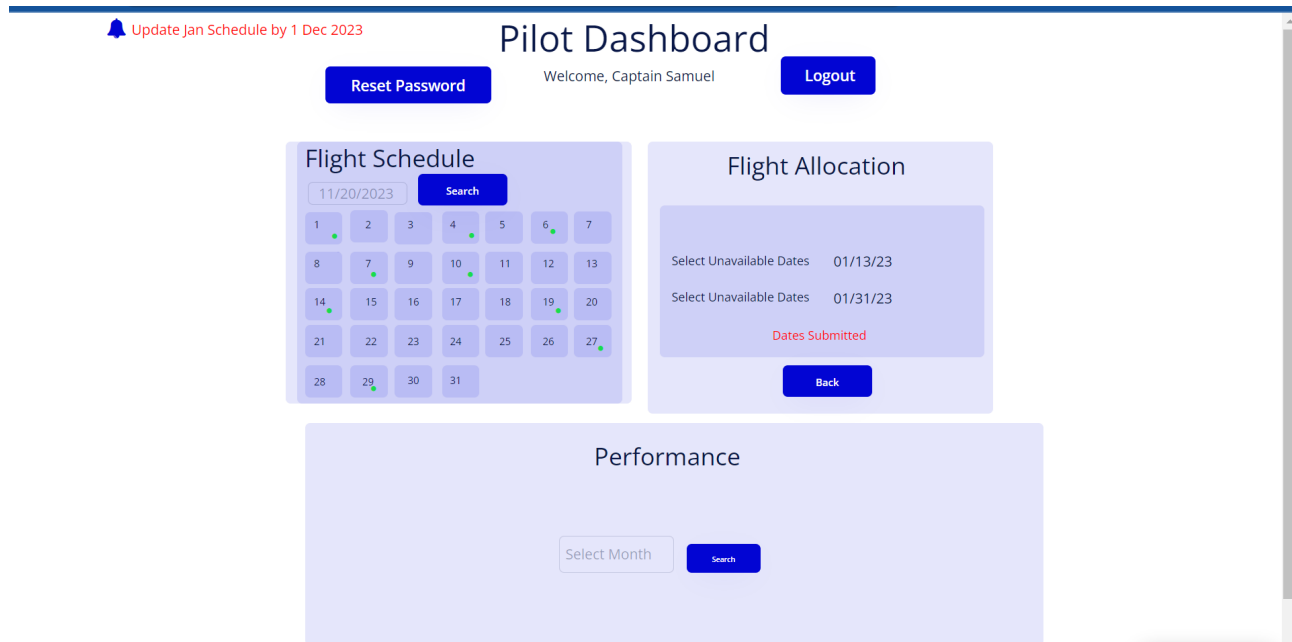


Figure 19.1: Pilot submitted unavailability

Figure 19 and 19.1 shows the pilot indicating the dates that they are not available for job allocation. Upon confirming, the pilot can simply click on the 'Submit Dates' button shown in figure 19 and receive a 'Date Submitted' alert as shown in figure 19.1.

UC-11: Visualise Workload

For pilots, they should be able to visualise their overall workload within the week and month.

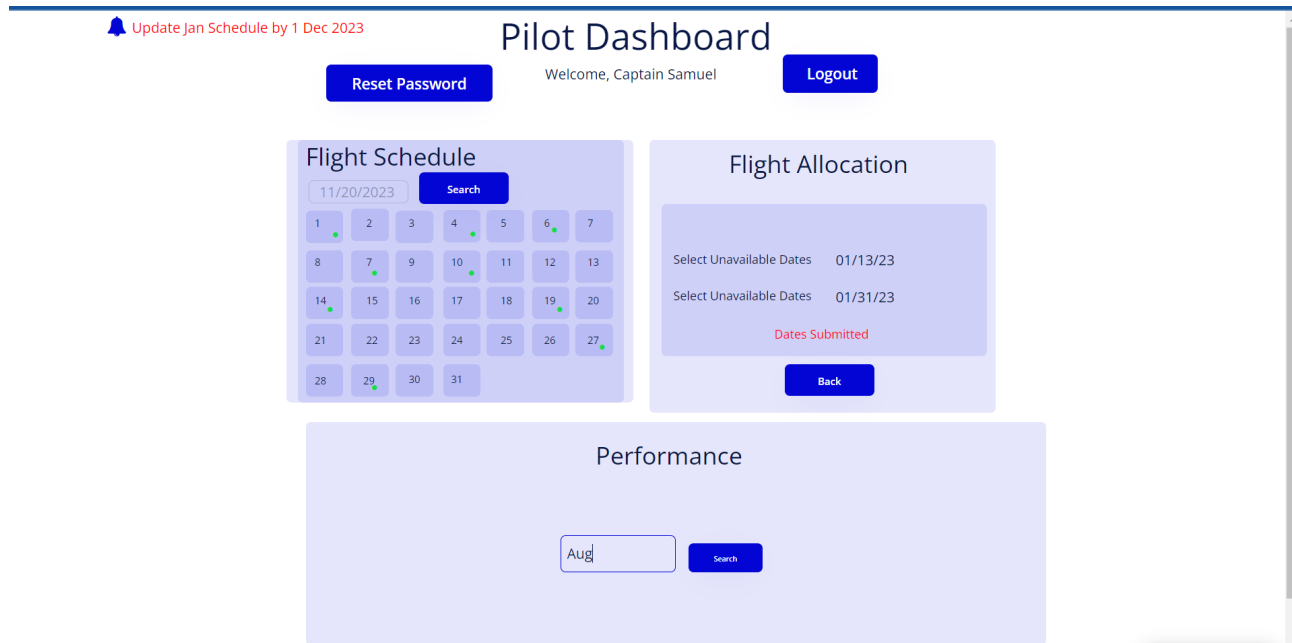


Figure 20: Pilot Indicating the month to see their respective performance

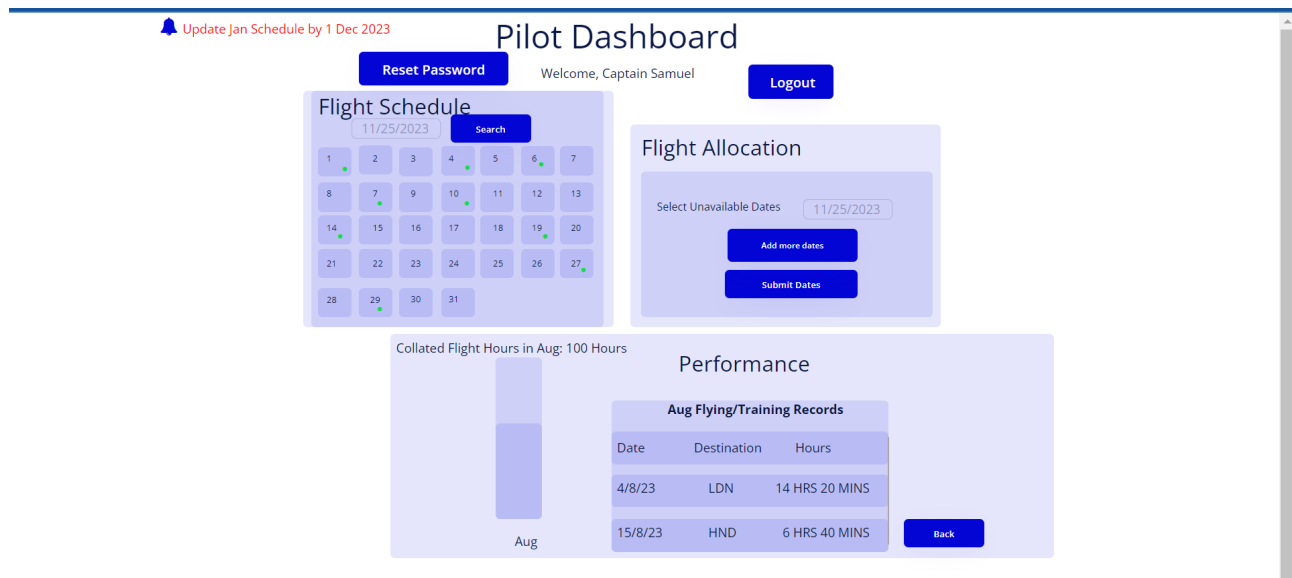


Figure 20.1 Pilot Indicating the month to see their respective performance

The pilot will be able to view the number of flight hours that he or she has accumulated. Besides, they will also be able to view their flying and/or training records in the selected month as shown in figure 20.1.

UC- 12/18 :View Schedule

Th manager will be able to view the schedule to see if there are unassigned jobs.

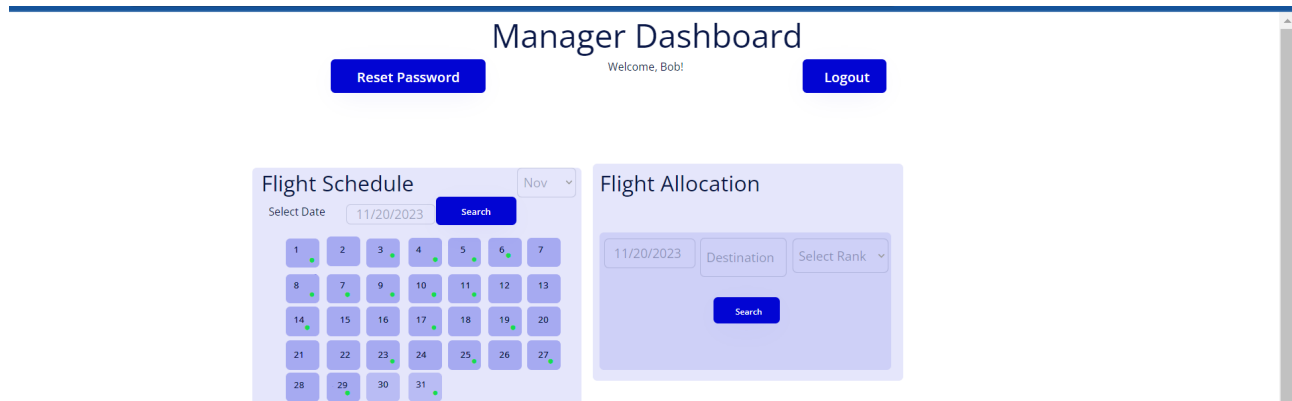


Figure 21: Manager Dashboard with Schedule

The manager will be able to view the schedule via the dashboard as shown in figure 21. The green dots indicate that the particular date has a flight scheduled to it.

UC-13 & 14: View Recommended Pilots and allocate job

In this use-case the manager will be able to allocate jobs to pilots. The system should recommend pilots based on matching requirements to ease the assignment process. It should also include up to 3 pilots should be recommended per job that matches the requirements.

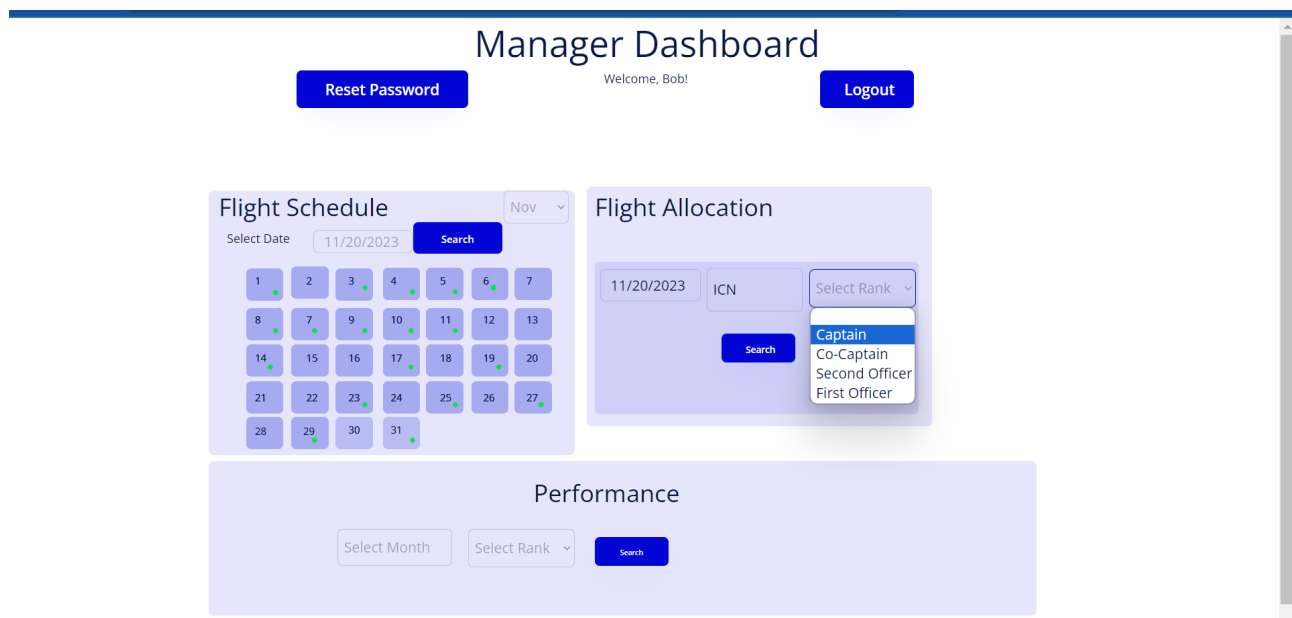


Figure 22: Manager indicating flight details for allocation

Figure 22 shows the manager selecting the rank of the pilot to allocate to the scheduled flight.

The screenshot displays the 'Manager Dashboard' with a header containing 'Reset Password', 'Welcome, Bob!', and 'Logout' buttons. The main content area is divided into three sections: 'Flight Schedule', 'Flight Allocation', and 'Performance'. The 'Flight Schedule' section features a calendar for November 2023 with a date selector set to 11/20/2023 and a 'Search' button. The 'Flight Allocation' section shows the same date and airport (ICN), with a dropdown menu for 'Captain' rank. A list of pilots with checkboxes is shown: Captain Samuel, Captain Tim, Captain Noah, and Captain James. The 'Performance' section at the bottom has fields for 'Select Month', 'Select Rank', and a 'Search' button.

Figure 23: Manager viewing recommended captains

The manager will then be able to see the available pilots on 20 November 2023 as shown in figure 23.

This screenshot shows the 'Manager Dashboard' with the 'Flight Allocation' section updated. The date is still 11/20/2023, but the 'Captain' dropdown is now closed. In the list of pilots, the checkbox next to 'Captain Tim' is now checked, indicating he has been selected for allocation. The other sections of the dashboard remain the same as in the previous figure.

Figure 23: Manager Selects a captain to allocate

Once the manager has decided on the pilot to allocate, he or she will then select on the checkbox located on the left side of the pilot's name, shown in figure 23. The manager can then click on submit once the selection for all the pilots has been completed.

The screenshot displays the Manager Dashboard with the following elements:

- Header:** "Manager Dashboard" title, "Welcome, Bob!" message, and "Reset Password" and "Logout" buttons.
- Flight Schedule:** A calendar for November 2023. The date 11/20/2023 is selected, and a "Search" button is present.
- Flight Allocation:** A section showing the selected date (11/20/2023), location (ICN), and role (Captain). A list of captains is shown with checkboxes: Captain Samuel (unchecked), Captain Tim (checked), Captain Noah (unchecked), and Captain James (unchecked). A "Confirm?" dialog box is open with "Yes" and "No" buttons.
- Performance:** A section with "Select Month" (set to November), "Select Rank" (dropdown), and a "Search" button.

Figure 24: Flight allocation confirmation

When the selection process is completed, the manager will be prompted to confirm on the selection as shown in figure 24.

The screenshot displays the Manager Dashboard after confirmation, with the following elements:

- Header:** "Manager Dashboard" title, "Welcome, Bob!" message, and "Reset Password" and "Logout" buttons.
- Flight Schedule:** A calendar for November 2023. The date 11/20/2023 is selected, and a "Search" button is present.
- Flight Allocation:** A section showing the selected date (11/20/2023), location (ICN), and role (Captain). A red alert message "Flight Allocation Confirmed!" is displayed. A "Back" button is present.
- Performance:** A section with "Select Month" (set to September), "Select Rank" (dropdown), and a "Search" button.

Figure 25: Confirmation alert

Upon confirmation, the alert will appear as shown in figure 25.

UC-15 & 16 :View top 3 pilots with less or more than 40 hours workload

When the manager filters the month and the rank of the pilots on the dashboard, the top 3 pilots with less than 40 working hours in a week should be highlighted.

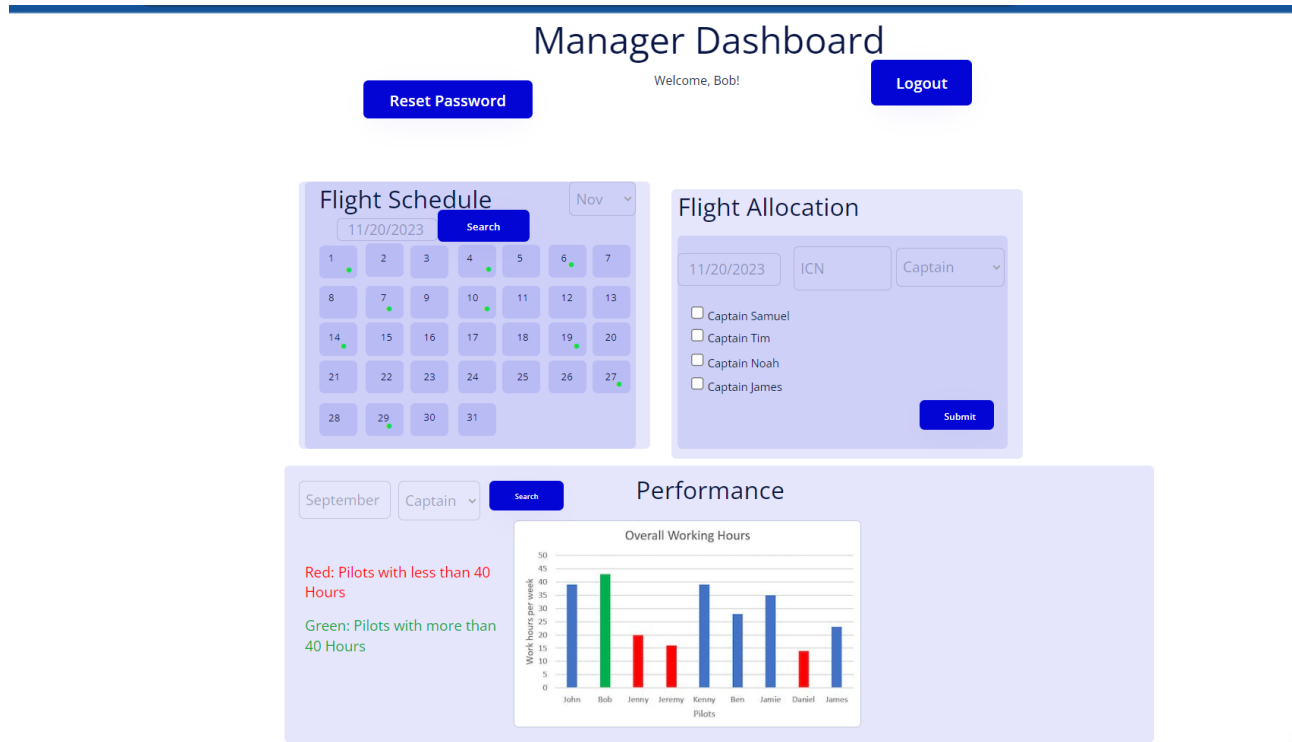


Figure 26: Manager viewing pilot's performance

In the graph shown in figure 26, the number of flight hours of the pilots are sorted according to the various colours. This will allow the manager to easily identify the workload of the pilots.

5 Project Management

<Include the final Gantt chart and highlight any overruns and explain the reasons.>

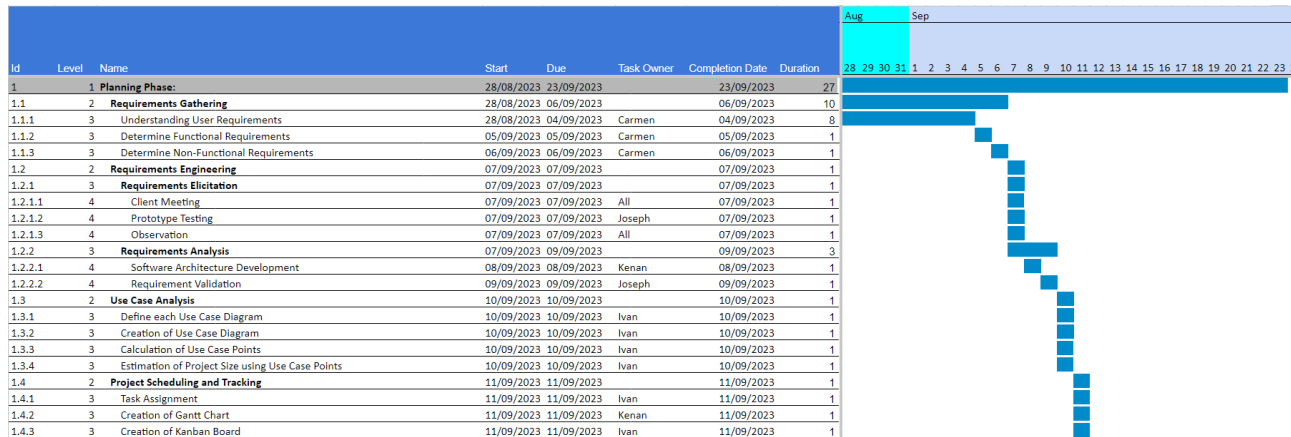


Figure 27: M1-1 Gantt Chart



Figure 28: M1-2 Gantt Chart

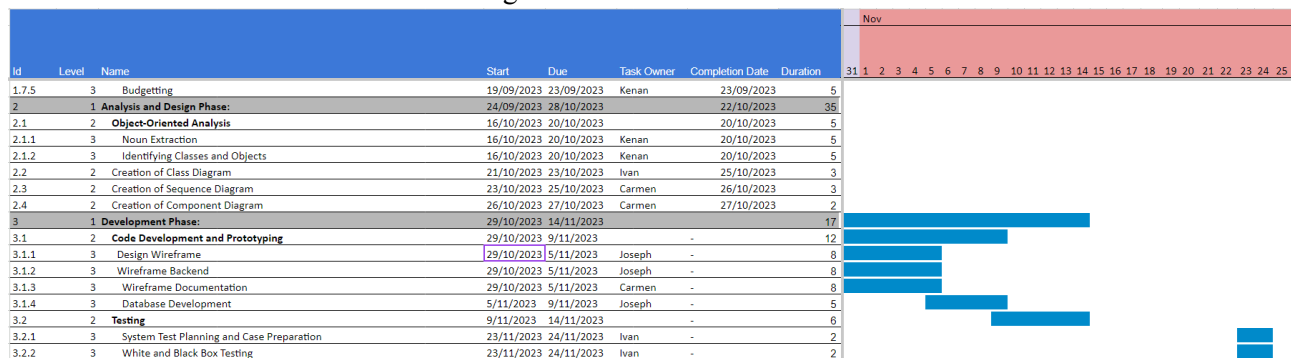


Figure 29: M3 Gantt Chart