INF2001
Introduction to Software Engineering

Milestone 2:

Detailed Design and Test Plan

for

Train Services Workload Management System

Prepared by

Team's Name: P13-1

ONG ZHEN YANG	**2303279**
STEPHANIE LING KHAI-MEI	**2302967**
LIU JIAXIN	**2302951**
MUHAMMAD IRFAN NUR ILHAM BIN HILMY IRWAN	**2302890**
MOHAMMAD HAIKAL BIN MOHAMMAD ZAMRI	**2302938**
SUMAIYA SHAH	**2303102**

Lab Group: P13 - 1

Github handle: https://github.com/GoodbyeKittyy/INF2001-Introduction-to-Software-Engineering-P13-Team-1-

Date: 16 November 2024

Table of Contents

1 Introduction 3
1.1 Final requirements 4
1.2 Final Class Diagram 6
1.3 Component Diagram 8
2 Detailed Design 11
2.1 Architecture Design 11
2.2 Detailed Design 14
3 Testing 16
3.1 Black Box Testing 16
3.2 White Box Testing 20

1 Introduction

1.1 Final requirements

User Interface Requirements:

UIR1: The user interface shall use a color scheme that aligns with the company's branding.
UIR2: The color scheme shall be applied consistently for primary navigation, interactive elements and notifications.
UIR3: The user interface shall use color-coded indicators for statuses of reject requests (e.g., "Pending" in yellow, "Accepted" in green, "Rejected" in red).
UIR4: The user interface shall use red text to highlight the staff's workload status (e.g., 48 hours) for staff who has over 40 total working hours in a month.
UIR5: The Manager's dashboard shall use color-coded indicators for staff availability, with red for "Unavailable" and green for "Available" statuses.
UIR6: In the navigation bar, the current page name shall be displayed in bold.
UIR7: Confirmation messages shall have a green background, while error messages shall have a red background.
UIR8: The company logo shall be centered in the navigation bar across all pages.
UIR9: The user interface shall use a darker color to indicate selected options and a lighter color for unselected options.
UIR10: The staff's job availability and preference information shall be displayed within the specific date box on the calendar.
UIR11: The logout button shall be positioned at the far-right of the navigation bar.
UIR12: Page titles shall be displayed below the navigation bar.

Functional Requirements

Staff Functionalities

FR1: The system shall provide staff with a dashboard displaying their job assignments and workload details (e.g., job role, routes, total working hours of the month, date, and time)
FR2: The system shall provide staff with a calendar that allows staff to interact with.
FR3: The system shall allow the staff to indicate their availability and preference of the week or in subsequent weeks to their manager for job assignments.
FR4: The system shall allow staff to indicate their availabilities and preferences up to one month in advance before the Wednesday deadline.
FR5: The system shall allow staff to reject or accept job allocations.
FR: The system shall allow staff to provide a reason for rejecting the job allocation in a text box.
FR6: The system shall allow staff to reject job allocations by the Wednesday deadline.
FR7: The system shall display a confirmation message to the staff upon submission of indicated availabilities and preferences.
FR8: The system shall display a notification message to notify the staff of the allocated jobs.
FR9: The system shall display a notification message to inform the staff whether their reject request has been approved by the manager.

Manager Functionalities:

FR10: The system shall allow manager to allocate jobs to staff members every week.
FR11: The system shall allow manager to view staff's monthly workload details on a dashboard.
FR12: The system shall allow manager to send notifications to staff.
FR13: The system shall allow manager to reallocate jobs before the Monday deadline.
FR: The system shall allow manager to accept or reject the staff's reject requests.
FR14: The system shall allow the manager to filter the staff workload details based on what the manager wants to see.
FR15: The system shall allow the manager to view the top three staff who has the lowest total working hours of the month.
FR16: The system shall highlight the staff with over 40 total working hours of the month to the manager.

IT Administrator Functionalities:

FR17: The system shall allow the IT administrator to create staff and manager accounts.
FR18: The system shall allow the IT administrator to assign roles to the staff and manager accounts.
FR19: The system shall allow the IT administrator to edit the information of existing staff and manager accounts.

System Functionalities:

FR20: The system shall uniquely identify staff members using a combination of their six-digit staff ID.
FR21: The system shall accept staff availability submissions until every Wednesday for job allocations made every Monday.

Non-functional Requirements

Accessibility Requirements

NFR1: The system shall function on all web browsers to ensure compatibility and usability.
NFR2: The system shall support multiple languages, allowing users to select their preferred language.

Performance Requirements

NFR3: The system shall respond to requests within 3 seconds.
NFR4: The system shall support concurrent users, up to 2500 users.
NFR5: The system shall maintain an uptime of at least 99.99% each month.
NFR6: The system shall save changes within 3 seconds.
NFR7: The saved changes shall be immediately reflected in the database.
NFR8: Adding a new user should take no more than 5 seconds after the submission of the form.
NFR9: The system shall notify managers of any reject requests in real-time, within 5 seconds.
NFR10: The system shall handle an increasing number of staff without any delay.

Safety and Security Requirements

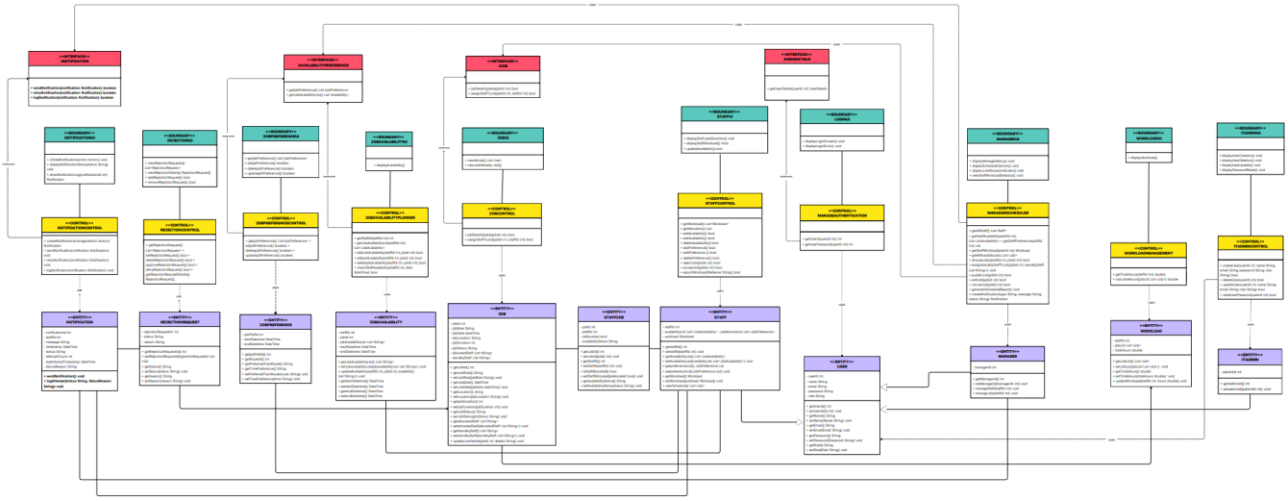
NFR11: The system shall ensure that every single piece of information must be encrypted to prevent unauthorized access.
NFR12: The system shall approve roles using 2-factor authentication, to ensure that approval processes are secure.
NFR13: The system shall have recovery mechanisms to minimize downtime during unexpected events.
NFR14: The system shall ensure that users create strong passwords with at least 12 characters, including numbers and special characters, and use two-factor authentication for password changes.
NFR15: The system shall ensure that all data transmissions are encrypted to avoid man-in-the-middle attacks.
NFR16: The system shall ensure that sensitive information (e.g., user details) are protected from unauthorized access using encryption.
NFR17: The system shall ensure that "Add Users" functionality is restricted to authorized IT administrators only.
NFR18: The system shall ensure that sensitive information such as passwords are stored securely.
NFR19: The system shall log all actions taken by IT administrators, including user creation, to ensure transparency and allow for tracking of administrative actions.

Other Requirements

NFR20: The interface must be intuitive and easy to use, with visual aids (e.g., colour coding or icons) to highlight scheduling conflicts or overworked staff.
NFR21: The system shall highlight the total working hours in red when the total working hours is more than or equal to 40.
NFR22: The user management interface should provide clear feedback on missing or incorrect fields. The system should help the administrator quickly correct any issues, such as a duplicate username.
NFR23: The interface should be user-friendly and intuitive, with visual aids (e.g., colour coding or icons) to highlight scheduling conflicts or overworked staff, allowing managers to assign jobs efficiently.
NFR24: The schedule must be displayed in an easy-to-read format, with clear options to filter by date, route, and time. The interface should be mobile-friendly.
NFR25: The schedule data must be 100% accurate and updated in real-time to reflect any changes.
NFR26: The workload data must be accurate, preventing inconsistencies such as missing or incorrect data.
NFR27: The system must comply with GDPR and PDPA regulations.
NFR28: The system should be subject to an annual security audit by an external contractor.
NFR29: The system must adhere to all company policies and regulatory compliance requirements.
NFR30: The system must be available 24/7 to allow managers to view workload data at any time.
NFR31: The system’s notification functionality must be available 24/7, ensuring that critical job updates or schedule changes can be communicated to staff at any time.
NFR32: If the notification fails due to external system errors (e.g., email or SMS service provider issues), the system should provide the manager with a clear error message and allow for manual resending after the issue is resolved.
NFR33: The system shall prevent duplicate user entries (e.g., identical usernames or emails).

1.2 Final Class Diagram

Please refer to the class diagram attachment for a clearer view.



Our UML class diagram utilizes different entities, interfaces, boundaries, and controls to achieve functionality.

Entities:

- Staff: Represents an individual staff member, containing attributes like staff ID, availability, job preferences, and current workload.
- Job: Represents a job, including job ID, role, location, date, duration, status, allocated staff, and standby staff.
- JobAvailability: Represents the availability of a staff member for a specific job.
- JobPreference: Represents a staff member's preference for a particular job type.
- Manager: Represents a manager responsible for managing staff and jobs.
- User: Represents a system user with attributes like user ID, name, email, password, and role.
- ITAdmin: Represents an IT administrator responsible for managing users and system settings.
- Notification: Represents a notification sent to staff or managers.
- RejectionRequest: Represents a rejection request for a particular job by a staff member.
- Workload: Represents the total workload of a staff member, including job assignments and associated hours.

Interfaces:

- INotification: Defines methods for sending, retrying, and logging notifications.
- IAvailabilityPreference: Defines methods for retrieving job preferences and job availability lists.
- IJOB: Defines methods for updating job details and assigning staff to jobs.
- IUser: Defines methods for retrieving user details.
- IWorkload: Defines methods for calculating workload hours based on job lists.

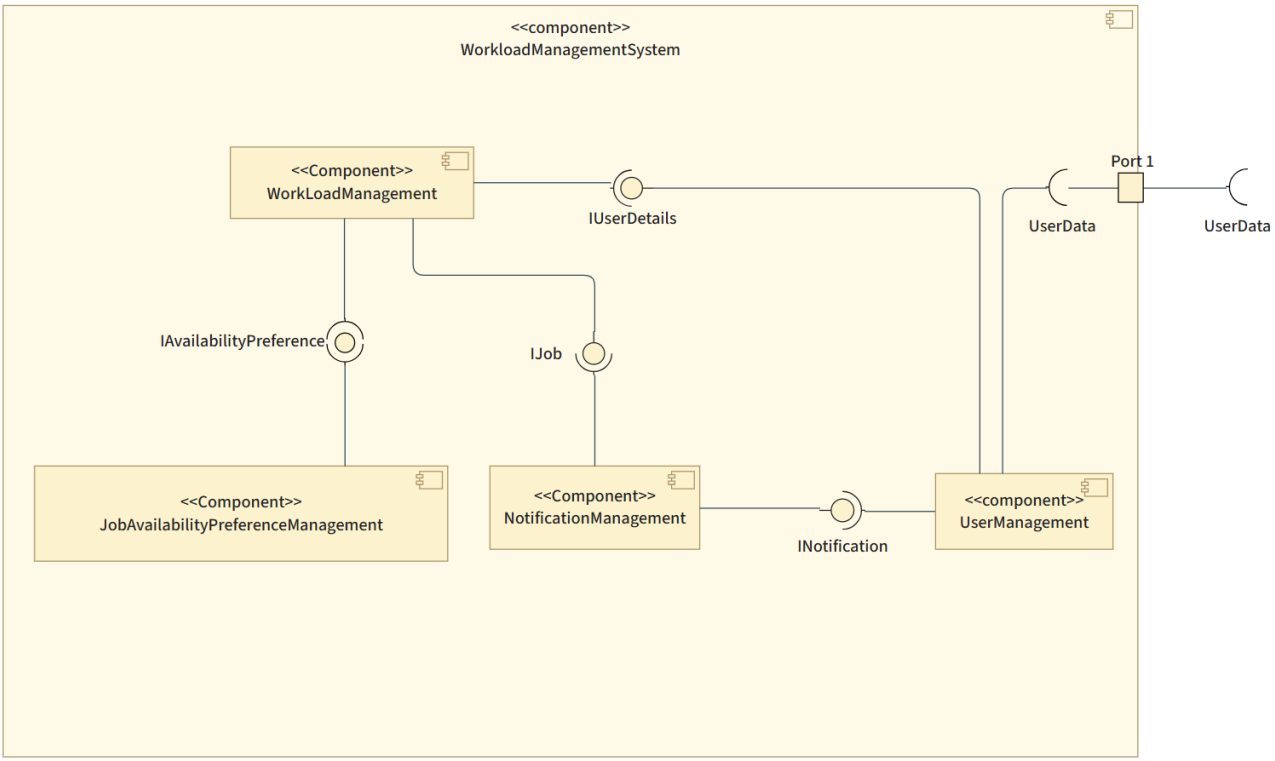
Boundaries:

- NotificationUI: Provides a user interface for managing notifications.
- RejectionUI: Provides a user interface for managing rejection requests.
- JobPreferenceUI: Provides a user interface for managing job preferences.
- JobAvailabilityUI: Provides a user interface for displaying and managing staff availability.
- JobUI: Provides a user interface for viewing and managing job details.
- StaffUI: Provides a user interface for managing staff details.
- ManagerUI: Provides a user interface for managing schedules and allocations.
- LoginUI: Provides a user interface for logging into the system.
- WorkloadUI: Provides a user interface for displaying staff workloads.
- ITAdminUI: Provides a user interface for IT administration tasks.

Controls:

- NotificationControl: Controls the sending, retrying, and logging of notifications.

1.3 Component Diagram

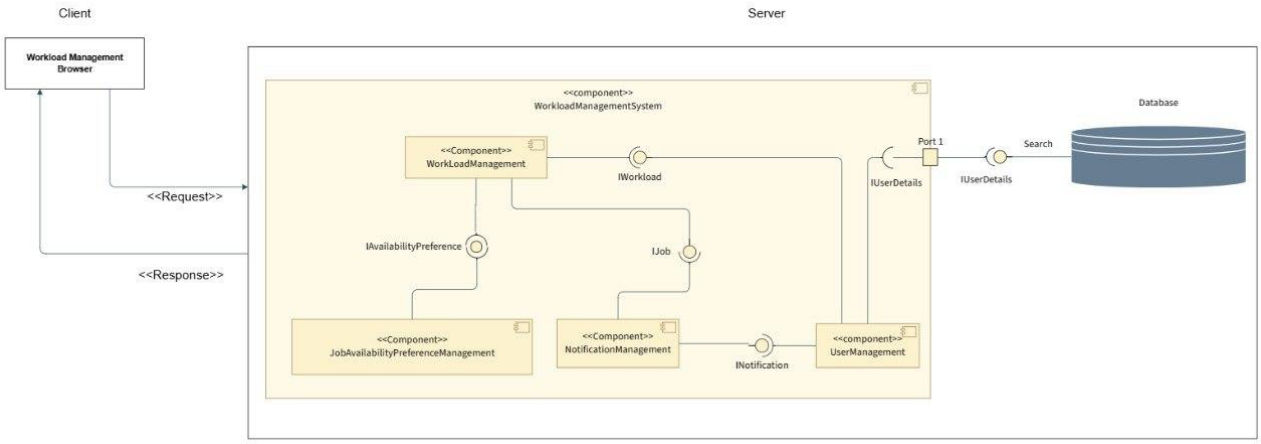


Component	ECB Classes	Description
WorkloadManagement	Entity Classes: - Workload - Job - Manager Control Classes: - WorkloadControl - JobControl - ManagerSchedular Boundary Classes: - WorkloadUI - JobUI - ManagerUI	Handles the calculation and monitoring of staff workloads based on their assigned jobs and total working hours, ensuring balanced job distribution. Provided Interface: IJob WorkloadManagement provides the IJob interface, allowing it to manage and track job allocations related to staff workload. It uses methods like getJobDetails(jobId: int) to retrieve job specifics and assignStaffToJob(jobId: int, staffId: int) to allocate jobs. Required Interface: IUserDetails WorkloadManagement requires the IUserDetails interface from UserManagement to access staff-specific data. It uses methods like getUserDetails(userId: int) to fetch user information that is necessary for workload calculations.
JobAvailabilityPreference Management	Entity Classes: - JobPreference - StaffJob - Staff - JobAvailability Control Classes: - JobAvailabilityPlanner - StaffControl - JobPreferenceControl Boundary Classes: - JobPreferenceUI - JobAvailabilityUI - StaffUI	Manages the availability and job preferences of staff members to be used during job allocations. Provided Interface: IAvailabilityPreference JobAvailabilityPreferenceManagement provides the IAvailabilityPreference interface to access staff availability and preferences. It uses methods like getAvailabilityList() to retrieve a list of date and time of the staff's availability and getJobPreference() to access staff route preferences. Required Interface: IJob JobAvailabilityPreferenceManagement requires the IJob interface to obtain job details, using methods like

		jobDetailUpdate(jobId: int) to update job records based on availability and assignStaffToJob(jobId: int, staffId: int) to assign staff based on preferences and availability.
NotificationManagement	Entity Classes: - RejectionRequest - Notification Control Classes: - RejectionControl - NotificationControl Boundary Classes: - NotificationUI - RejectionUI	Manages sending notifications. Provided Interface: INotification NotificationManagement provides the INotification interface, allowing it to perform notification-related actions. It uses the method sendNotification(notification: Notification) to dispatch notifications to staff. Required Interface: IJob NotificationManagement requires the IJob interface to get the allocated job details by using the method jobDetailUpdate(jobId: int).
UserManagement	Entity Classes: - User - ITAdmin - Control Classes: - ITAdminControl Boundary Classes: - ITAdminUI	Manages user accounts, handling operations like account creation, updating, and deletion, as well as managing user roles and credentials. Provided Interface: IUserDetails UserManagement provides the IUserDetails interface, enabling the WorkloadManagement component to access user-related data. Required Interface: UserData UserManagement required the UserData interface, to retrieve the data of the users from an external component.

2 Detailed Design

2.1 Architecture Design



The Client-Server Architecture was chosen as the project's architectural pattern. The rationale behind this choice is due to the benefits it brings to the Train Services Workload Management System (TSWMS). The Client-Server Software Architecture is made up of 2 main separate components – Client and Server. The Client handles the user interactions and input, while the Server manages the core functionalities and processing of the system with the Database. The following are the reasons behind the selected architecture:

Compared to other architecture patterns, the Client-Server model allows for real-time data sharing and centralised control of data in the TSWMS. Since the system require to display live data and timely updates for accurate job scheduling and workload distribution for all staff, this structure ensures that the information is consistent, up-to-date and displayed real-time throughout all platforms.

The model also enables greater security and role-based access control measures in the server side, managing user authentication and roles centrally and restrictions are applied uniformly to all clients. The TSWMS has different user right permissions for different roles in the organisation (e.g. Only Managers are allowed to decide and allocate jobs to the staff or Only IT Admin are allowed to view account information etc.) The TSWMS handles a large volume of sensitive information retailed to schedules, employee information and workload details and enforcing a robust security is crucial. By centralising on the server, it is easier to enforce access control policies such that only authorised roles have specific functionalities or access certain restricted data.

The Client-Server model brings greater scalability and flexibility to the system. Since the Server is responsible for processing data and managing Client requests, it allows the Server component to be scaled separately from the Client component. Since the TSWMS will cover more users/staff over time due to future staff employment, it is crucial for the system to scale to such increasing data volume and requirements in the future. Same applies for the Client component, the browser application can be increased to access from multiple different device types or locations, without affecting the overall architecture of the system.

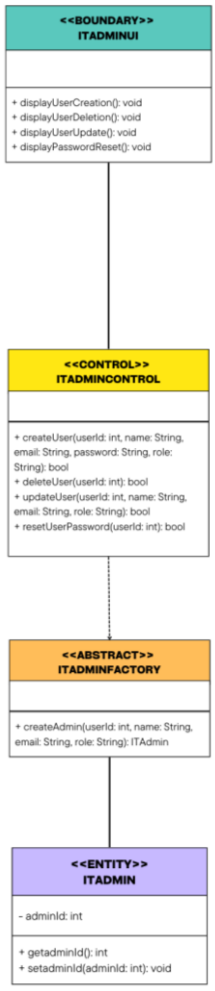
Client	The Client is represented by the Workload Management Browser interface whereby the users (staff/managers) will use to interact with the TSWMS. A user interface for staff to manage their workloads, view job allocations, indicate availability & preferences. Client responsibilities include: • **Sending requests** to the Server to retrieve actions such as view job allocations, submitting staff availability/preference • **Receiving responses** from the Server with the requested actions
Server	The Server is made up of four main components that handle the core business logic of the TSWMS and communicating with the database. Server responsibilities include: 1. **WorkloadManagement –** Manages the workload of staff members. Handling data related to staff schedules, job allocations and jobs 2. **UserManagement –** Manages/stores data about staff such as employee profiles, ID, roles and information required for workload allocation. All the Server components will communicate through the UserManagement component, which is the only component that directly interacts with the Database (Centralised data access) 3. **JobAvailabilityPreferenceManagement –** Manages the job availability and preferences of the staff 4. **NotificationManagement** – Manages the notifications related to the job allocations, alerts or other updates to inform the staff.
Database	The Database stores all the data in the entire system architecture of the TSWMS, ensuring data integrity and real-time availability for the components Database responsibilities include: • **Data storage**: Contains all the system data, including the user information, job data etc. • **Data retrieval**: Based on Server components, perform searches to retrieve necessary data and return the data • **Data integrity**: Provides consistent and up-to-date data for all Server components • **Communication with UserManagement:** Database communicates directly with UserManagement component

Scenario Examples (Client-Server architecture flow):

Staff want to select their preferred train route	1. Workload Management Browser (Client) sends a **request** (request a list of available routes for the staff's job role to choose from) to the Server. 2. The Server receives the request from the Workload Management Browser 3. The Server queries the Database for the available routes that match the job role for that date. The Database performs a search based on the query and retrieves the relevant data (The list of specified train routes) and sends back to the Server. 4. The Server returns the **response** back to the browser 5. The Workload Management Browser interface displays the list for the staff to choose their preferred route. 6. Staff submits the Selected route 7. The Client sends a **request** to the Server to save the selected route for the staff's job schedule 8. The Server receives the submission request from the Client 9. The Server then performs the update operation in the Database, saving the selected route. 10. The Database saves the staff's selected route and sends a confirmation back to the Server. 11. The Server sends a **response** back to the Client. The response confirms that the selected route has been successfully saved. 12. The Browser interface displays a confirmation message to the staff, indicating that their preferred route has been successfully updated.
Staff want to view their dashboard and schedule	1. Workload Management Browser (Client) sends a **request** to the Server to retrieve all the relevant data and information needed to display on the dashboard 2. The Server receives the request from the Client and performs query on the Database to retrieve staff's job allocations and workload details 3. The Database retrieves the requested data and returns it to the Server 4. The Server returns a **response** back to the Client with the dashboard data 5. The Browser interface displays the schedule, showing the staff their current workload schedule

2.2 Detailed Design

2.2.1 Factory Pattern: User



Intent
The purpose of the Factory pattern in this workload management system is to promote code reusability and maintain uniformity in creating instances of User and its subclasses. Factories help ensure that these instances have a consistent structure and behavior, supporting system flexibility to switch between different implementations or add new roles without affecting existing code. Additionally, the Factory pattern facilitates dependency injection, which provides better control over object creation and lifecycle management—useful for operations like user deletion.

Applicability
In this system, User objects can be created with a standardized configuration, decoupling User objects from specific classes and managing their subclass dependencies. Different roles, such as ITAdmin, Staff, and Manager, have unique permissions and actions within the system. A factory allows each role to implement its specific functionality while adhering to the structure of the User class.

Using a factory for creating and managing objects' lifecycles simplifies dependency management and supports unit testing. For example, with this pattern in place, IT Administrators can safely delete a user without disrupting any dependencies.

According to the M2 software requirements specification, the Factory pattern influences the class diagrams related to account components, including ITAdmin, Staff, and Manager. Each of these roles is represented by respective Entity, Control, and Boundary classes.

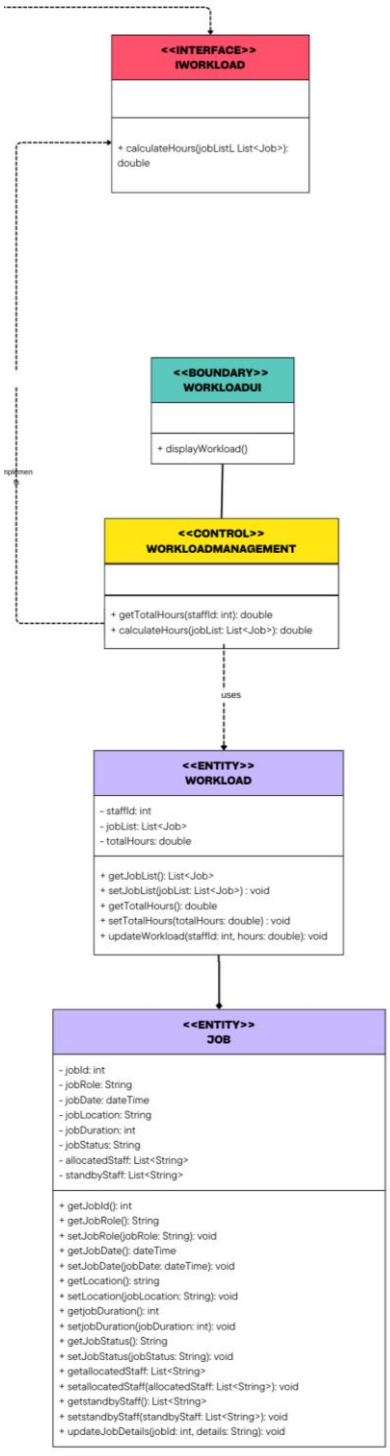
Before implementing the Factory pattern, Staff and ITAdmin classes were tightly coupled, and account creation was managed solely by the ITAdminControl class, making all account-type classes dependent on it, including ITAdmin itself. The introduction of the Factory pattern led to the creation of an Factory class to instantiate each account types (ITAdmin, Staff, and Manager).

The above portion of our class diagram is one of the updates showing how classes implementing the Factory design pattern should be organized.

Justification
The Factory pattern aligns with SOLID principles, specifically the open/close principle and the dependency inversion principle.

The open/close principle is supported by enabling developers to add new object types without modifying the core Factory class or existing code.

3 2.2.2 Composite Pattern



Intent

The Composite pattern is used to enable composition of objects so that clients can treat individual objects and groups of objects uniformly. This pattern is beneficial for managing a one-to-many hierarchy. In this workload management system, it's particularly relevant for managing the Job class within the Workload class, providing a structured design that integrates the interactions between job and workload objects.

Applicability

The Composite pattern is applicable in the Workload and Job classes, which are connected via aggregation. A Workload consists of multiple Jobs, and the Composite pattern allows clients to interact with these classes in a consistent manner. For example, modifying a Job object within a Workload would automatically reflect the update, while deleting a Workload would remove its associated Jobs. This pattern enhances flexibility and consistency in handling these class relationships.

Justification

The updated class diagram section above supports treating composite objects uniformly and adding extensions without altering existing code, thereby satisfying the open/close and Liskov substitution principles. Additionally, this setup establishes a clear hierarchical structure, promoting scalability and uniformity within the class diagram. This structure provides a framework that adheres to the Composite design pattern and serves as a general guide for organizing classes with similar structural needs.

4 Testing

4.1 Black Box Testing

We have selected 'allocateJobs' as the method for black box testing. This method is prominent as it is responsible for the process in which manager assigns jobs to staff with checks on staff's availability, preferences, and job criteria.

A decision table is formed that includes a set of cause based on the checks performed by the method. The values of Yes or No (Y/N) are then used as possible combinations of the causes to conclude their corresponding effects. The number of combination columns are derived from the number of causes where $2^4 = 16$, ensuring all causes and values are considered towards a full decision table. The full decision table is as follows:

FULL DECISION TABLE:

Causes		**Values**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
C1	List of unassigned jobs updated	Y/N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y
C2	Available staff match job criteria	Y/N	N	N	N	N	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y
C3	Staff has no conflicting shifts	Y/N	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y
C4	Staff is not on leave/off	Y/N	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y
Effects																		
E1	Job allocation successful																	X
E2	Job allocation failed		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

From the full decision table, combinations with common effects due to same initial cause are then further reduced and combined into one combination to further streamline and enhance the testing. For example, combinations 1 to 8 have an 'N' for 'C1: List of unassigned jobs updated' where subsequently, regardless of the input of 'C2: Available staff match job criteria', the output will still be 'E2: Job allocation failed'. Similarly, while combinations 9 to 12 have a 'Y' for 'C1: List of unassigned jobs updated', an 'N' for 'C2: Available staff match job criteria' would mean that regardless of the input of 'C3: Staff has no conflicting shifts', the output will still be 'E2: Job allocation failed'. The reduced decision table is as follows:

REDUCED DECISION TABLE:

Causes		**Values**	**1-8**	**9-12**	**13-14**	**15**	**16**
C1	List of unassigned jobs updated	Y/N	N	Y	Y	Y	Y
C2	Available staff match job criteria	Y/N	-	N	Y	Y	Y
C3	Staff has no conflicting shifts	Y/N	-	-	N	Y	Y
C4	Staff is not on leave/off	Y/N	-	-	-	N	Y
Effects							
E1	Job allocation successful						X
E2	Job allocation unsuccessful		X	X	X	X	

With the reduced decision table, we have come up with specific test cases to validate that the method produces the results correctly based on the different combinations. The test cases are as follows:

TEST CASES:

Test Scenario	**Test Case**	**Pre-Conditions**	**Test Data**	**Test Steps**	**Expected Result**	**Actual Result**	**Pass/ Fail**
Job allocation fail	List of unassigned jobs not updated	The manager logged into the system System does not display the updated list of unassigned jobs	Job ID: "Job01" Date: "30-06-2025"	1. Select “Allocate Jobs to Staff” from the main menu. 2. Enter Date 3. Enter Job ID to view list of unassigned jobs	Error will display “List of unassigned jobs not ready. Please try again later”. List of updated unassigned jobs not displayed		

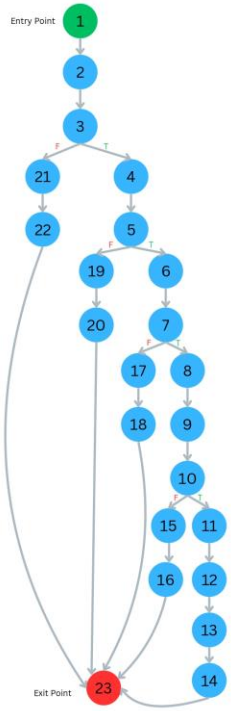
Job allocation fail	Updated list of unassigned jobs but available staff does not match job criteria	The manager logged into the system System displays the updated list of unassigned jobs, but available staff does not match job criteria	Job ID: "Job01" Date: "10-01-2025" Staff: "Ahmad"	1. Select “Allocate Jobs to Staff” from the main menu. 2. Enter Date 3. Enter Job ID to view list of unassigned jobs 4. Enter staff name to allocate job to	Error will display “Staff does not meet the required job criteria. Please select a qualified staff member”, Indicating selected staff is not for the job.		
Job allocation fail	Updated list of unassigned jobs, available staff match job criteria but has conflicting shifts	The manager logged into the system System displays the list of unassigned jobs, available staff match job criteria but has conflicting shifts	Job ID: "Job01" Date: "10-01-2025" Staff: "Zack"	1. Select “Allocate Jobs to Staff” from the main menu. 2. Enter Date 3. Enter Job ID to view list of unassigned jobs 4. Enter staff name to allocate job to	Error will display “Staff is unavailable due to an existing shift. Please select another unassigned job or select an available staff.”, indicating staff with conflicting shifts are no available to job allocation		

Job allocation fail	Updated list of unassigned jobs, available staff match job criteria, has no conflicting shifts but is on leave/off	The manager logged into the system System displays the updated list of unassigned jobs, available staff match job criteria, no conflicting shifts but is on leave/off	Job ID: "Job01" Date: "10-01-2025" Staff: "Ryan"	1. Select “Allocate Jobs to Staff” from the main menu. 2. Enter Date 3. Enter Job ID to view list of unassigned jobs 4. Enter staff name to allocate job to	Error will display “Staff is unavailable due to leave. Please select another unassigned job or select an available staff.”, indicating staff with conflicting shifts are no available to job allocation		
Job allocation success	Updated list of unassigned jobs, available staff match job criteria, has no conflicting shifts and is not on leave/off	The manager logged into the system System displays the updated list of unassigned jobs, available staff match criteria, no conflicting shifts, and is not on leave/off.	Job ID: "Job01" Date: "10-01-2025" Staff: "Michael"	1. Select “Allocate Jobs to Staff” from the main menu. 2. Enter Date 3. Enter Job ID to view list of unassigned jobs 4. Enter staff name to allocate job to 5. Select the ‘confirm’ button.	Job allocated to staff successfully		

4.2 White Box Testing

The team has selected the viewStaffWorkloadStatistics method for white box testing. An efficient workload viewing method holds significant importance for a company, as it provides managers with the necessary insights to make informed decisions for allocating jobs. This is especially crucial since managers need to ensure that staff workloads are balanced, job preferences are considered, and availability is considered. Ensuring the thorough testing of this method contributes to the overall functionality and efficiency of the system

```
1   def viewStaffWorkloadStatistics():
2       displayManagerMenu()
3       if selectViewStaffWorkloadStatistics():
4         jobList = getJobList()
5         if jobList:
6             totalHours = calculateHours(jobList)
7             if totalHours >= 0:
8                 displayWorkloadDashboard(jobList, totalHours)
9                 filterCriteria = getFilterCriteriaFromManager()
10                filteredData = filterWorkloadData(jobList, filterCriteria)
11                if filteredData:
12                    displayFilteredStatistics(filteredData)
13                    detailedStatistics = getDetailedStatistics(filteredData)
14                    displayDetailedStatistics(detailedStatistics)
15                else:
16                    displayErrorMessage("Failed to process filtering request.")
17            else:
18                displayErrorMessage(“"Failed to calculate total hours.")
19        else:
20            displayErrorMessage("No job data available.")
21    else:
22        displayErrorMessage("Failed to select 'View Staff Workload Statistics'.")
23    return
```



Cyclomatic Complexity Calculation	
Number of edges (E)	26
Number of nodes (N)	23
Connected Components (P)	1
M = E – N + 2P	
M = 26 – 23 + 2(1) = 5	

Basis Path for *viewStaffWorkloadStatistics* Method

Basis Path		
Path	Nodes following Path	Test Case Descriptions
Path 1	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 - 10 – 11 – 12 – 13 – 14 - 23	The manager successfully views the staff workload statistics with valid data and filters.
Path 2	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 15 – 16 – 23	The manager attempts to view the staff workload statistics, but the filtering process fails due to missing or corrupt data.
Path 3	1 – 2 – 3 – 4 – 5 – 6 – 7 – 17 – 18 – 23	The manager attempts to view the staff workload statistics, but the total hours calculation fails.
Path 4	1 – 2 – 3 – 4 – 5 – 19 – 20 – 23	The manager attempts to view the staff workload statistics, but no job data is available.
Path 5	1 – 2 – 3 – 21 – 22 – 23	The manager attempts to view the staff workload statistics, but the selection of "View Staff Workload Statistics" fails.

Test Cases for *viewStaffWorkloadStatistics* Method

Test Case	Pre-conditions	Test Steps	Test Data	Expected Result
Test Case 1 (Path 1)	Manager is logged in to the system. Workload data is available.	1. Select the "View Staff Workload Statistics" option. 2. System retrieves the job list and calculate total hours. 3. Display the workload dashboard. 4. Enter filter criteria and apply the filter. 5. View detailed statistics for specific staff.	Filter Criteria: { "staffId": 1, "date": "07-11-2024" }	Success message displays "Workload statistics successfully displayed." Filtered statistics and detailed statistics are displayed.
Test Case 2 (Path 2)	Manager is logged in to the system. Workload data is available.	1. Select the "View Staff Workload Statistics" option. 2. System retrieves the job list and calculate total hours. 3. Display the workload dashboard. 4. Enter filter criteria and apply the filter. 5. Simulate a filtering error due to missing or corrupt data.	Filter Criteria: { "staffId": 0, "date": "07-11-2024" }	Error message displays "Failed to process filtering request." Suggest retrying or contacting support.

Test Case 3 (Path 3)	Manager is logged in to the system. Workload data is available.	1. Select the "View Staff Workload Statistics" option. 2. System retrieves the job list and calculate total hours. 3. Simulate a total hours calculation error.	-	Error message displays "Failed to calculate total hours." Process ends without displaying statistics.
Test Case 4 (Path 4)	Manager is logged in to the system. No workload data is available.	1. Select the "View Staff Workload Statistics" option. 2. System attempt to retrieves the job list.	-	Error message displays "No job data available." Process ends without displaying statistics.
Test Case 5 (Path 5)	Manager is logged in to the system.	1. Select the "View Staff Workload Statistics" option.	-	Error message displays "Failed to select 'View Staff Workload Statistics'." Process ends without displaying statistics.