COMP2043.GRP Interim Group Report

[P2024-18] Dynamic Restaurant Staff Scheduling Tool

Team Name/Identifier: Team2024.09

Group Members

Name	**School of CS Username**
Ziyu JIA	scyzj7 (20513021)
Minghe XU	saymx1 (20513973)
Aijia YU	shyay2 (20516770)
Peifeng LIU	scypl5 (20516924)
Jey Vi TAN	hcyjt5 (20614137)
Lik Wei CHAN	scylc5 (20512215)

Supervisor: Ning Xue

December 8, 2024

Contents

Chapter 1

Introduction

1.1 Context

Schedule management is essential to the running of a restaurant, and yet, it's hard for most companies to come up with a simple schedule. This complexity comes from changes in customer flow, shift demand and employee lateness. Although good algorithms can make schedules more reliable, they are hard to adopt. A lot of restaurant chains are utilizing conventional practices like alternate work and rest times which can not deal with dynamic demands and do not take into account individual employee preferences. And that can mean a decrease in employee happiness and performance.

1.1.1 Gap Analysis

Software such as ScheduleFlex or Deputy are some of the systems that try to solve some of these staffing issues, but they are not flexible and do not offer real-time solutions. These softwares are usually very corporate which allows you to overstaff, understaff and your employees are usually more unsatisfactory. In addition to that, they do not cut operational expenses like overtime or forming efficient workflows.

1.1.2 Objective

A scheduling tool should be flexible, designed concerning data, and friendly to the employees, as much as possible, which is exactly what our team's tool does. Key features include:

- **Real-time Scheduling:** Able to solve unforeseen situations like employee tardiness or whenever an employee suddenly falls sick.
- **Automated Conflict Prevention:** This continuously validates the schedules making sure it is in line with labour laws, and employee satisfaction as well as preventing understaffing or overstaffing.
- **Employee Satisfaction:** Employees can have working preferences like what time they prefer their shifts to be in, etc. These preferences will be taken into account which will be reflected in the schedule creation as soft constraints.

- **Check-in Feature:** Employees can check in. When an employee does not check in after 15 minutes, the system sends an automated message to enquire about the tardiness. If that employee isn't able to come to work, a replacement will occur.
- **Data-driven Insights:** By using trends and data, the system will be able to make predictions and make changes to the algorithm accordingly.

1.2 Background Information & Literature Review

An adequate design of a restaurant scheduler should include three elements: fair task assignments to employees, minimization of weekly costs from employee pay, and maximum efficiency. Restaurants in particular have unpredictable customer flows, unforeseen situations and workers having varying priorities, thus a dynamic scheduling system is a must. Although current scheduling systems make scheduling easy, most of them forget about external factors such as employees' preferences or even the customers' needs for fast service.

1.2.1 Key Studies and Approaches

Partitioning Tasks for Simplified Scheduling

Breaking down tasks for different roles in a company is a way to simplify the scheduling problem. For example, separating tasks for cooks, servers, cleaners, etc (Blöchliger 2004). This can eliminate the interdependencies as well as put focus on each department rather than solving the problem as a whole. .

Mathematical Programming for Optimal Solutions

Mathematical programming is one of the best ways to achieve a highly effective schedule as it produces low-cost solutions and functions best in a regulated environment (Ernst et al. 2004). However, it is not the best when taking into account real-world nuances. Things like employee preferences and changes in client demand can easily reduce the reliability of the schedule.

Heuristic Algorithms for Practical Solutions

Restaurants require quick decisions to the regular unforeseen scenarios they get. Thus, an option that has been used by many is using heuristic algorithms. Caprara et al. (2003) state that the heuristic model focuses on several objectives to ensure fast response and decent efficiency. These objectives include balancing workloads, avoiding staff fatigue, adapting to everyday shift patterns, etc. However, compared to a straightforward mathematical algorithm, the solution given by the heuristic algorithm is not guaranteed to be the best solution.

Metaheuristic Algorithms for Enhanced Performance

Metaheuristic is mostly used for more complex problems where the search space of solutions is large and using traditional mathematical models is too computationally intensive (Konjaang & Xu 2021). The solutions metaheuristic models provide are more "good enough" and the solutions are provided in a reasonable time.

Advanced Scheduling Techniques

There is much research that extends heuristics and metaheuristics models and brings a whole new idea. Thesen (1978) explores heuristics models usage in resource-limited environments and certain tasks have dependencies. It focuses on delivering schedules as quickly as possible prioritizing tasks based on urgency and availability. Thesen (1978) also introduced a tool (WASP program) to iteratively refine schedules to ensure a solution closer to the optimal solution.

Industry Applications and Relevance

All the studies above from heuristic algorithms to metaheuristic algorithms are highly relevant to the restaurant industry where efficient workforce management is crucial. These algorithms can help optimize staff schedules which in turn increase efficiency by balancing workloads, task urgency and employee preferences. Although both heuristic and metaheuristic algorithms do not guarantee optimal results, they do provide a cost-effective solution to companies as well as "good enough" schedules.

Using these studies as references, our team aims to produce a dynamic scheduling web application tailored to restaurants with features including, real-time schedule adjustments, automated conflict detection and taking into account the preferences of employees.

Chapter 2

Requirement Specifications

2.1 Functional Requirements

Shift Data Import and Export

The system must support the import and export of shift data via Excel. This functionality simplifies the entry of existing employee information and allows managers to process and analyze data efficiently. Employees should be able to view their schedules directly through the system. Verification involves testing the system's ability to accurately import and export Excel files without data loss or errors. The tool should handle large datasets efficiently, ensuring smooth operation even with extensive employee records.

Automatic Shift Assignment

The tool should automatically assign shifts, adhering to employee preferences such as weekly work hour limits and preferred shift timings. It must ensure that the demand for each shift is met, balancing staffing levels to avoid over-staffing or under-staffing. Verification includes creating various scheduling scenarios and checking that the system produces schedules meeting employee preferences and shift demands. The system should also consider legal requirements, such as mandatory breaks and maximum working hours, to ensure compliance with labour laws.

Real-Time Scheduling Adjustments

The system must be able to respond to last-minute changes like employee call-offs or unexpected increases in customers. The tool should be able to increase the number of employees for that time period and provide notifications to managers and employees about the schedule changes so that everyone is informed promptly.

Shift Swap and Replacement Feature

The tool should include a feature allowing managers to swap shifts of employees easily with the system validating the shift swaps requested to be made. Additionally, the system should find replacements for shifts that employees cannot cover, ensuring that all

shifts aren't under or over-staffed.

2.2 Non-Functional Requirements

User-Friendliness

The software should have a user-friendly interface such that employees and managers are able to navigate through the page without much trouble and do not need a tutorial to find what they need.

Accessibility

The software should be able to be used on different devices without affecting the usage and functionality between devices. To achieve this, the interface should be responsive such that the interface adjusts according to the device's screen size.

Performance and Scalability

The tool should be able to handle large datasets like employee data and schedule data. Not only that, the tool should also be able to have many concurrent users without having much downside effect towards the performance.

Maintainability

The software should be designed with long-term maintainability in mind where modular design principles and comprehensive documents are used so that future maintenance is much easier.

Security

The system must ensure the security of restricted data. It should implement secure access controls so that unauthorized access and data breaches will not occur.

Chapter 3

Assumptions & Key Functionalities

3.1 Assumptions

Our team has assumed that when the software is put into use, the restaurants already had data of the current staff shifts, staff profiles, etc.

3.2 Key Functionalities

Our team has designed two different sets of user interfaces for different user groups: employees and managers, while each set has different function sections.

3.2.1 Employee

First, view schedule. This enables employees to view their own timetables.

Second, leave request submission. This enables employees to apply absent in their own shift, due to sickness, annual holidays, emergencies, etc.

Third, export data. This enables employees to download data from the website to their own devices, which includes their own schedules, working data, etc. Our team also support employees to choose their preferred file format during exporting data, such as PDF, CSV and XLS.

Fourth, check in. This enables employees to check in before their shift, hence the manager could check if any employee is late. If an employee still has not checked in after 15 minutes of his/her shift, our software will send him/her an alert to ask the reason for the tardiness and whether he/she needs any help. And if this employee is unable to come for his/her current shift, the software will notify the nearest available employee to take over this shift.

Lastly, view and edit profile. This enables employees to view their profile, which contains their employee IDs, phone numbers, e-mail addresses, age, name, work preferences, weekly overviews, etc. Among them, employees will be able to edit their phone numbers,

e-mail addresses, name, age and part of their working preferences, such as their preferred shifts.

3.2.2 Manager

First, view schedule. This enables manager to view the whole schedule with all employees schedules.

Second, view calendar. This enables manager to have the big picture of the attendance and absence records straightforwardly.

Next, approve and deny leave request. When employees apply for leave, the manager will receive their applications. Therefore, this function enables the manager to review the applications and give responses, and these responses will be further sent to corresponding employees.

Fourth, swap shifts. This function enables the manager to swap or change employees' shifts. The notifications will be sent to the related employees as soon as the manager used this function.

Then, import and export data. This enables managers to entry the existing employee information into the system which would facilitate processing and analyzing data. The manager is also able to download the the data from the software, which includes schedule data, calendar data, costs, etc. Our team also support employees to choose their preferred file format during exporting data, such as PDF, CSV and XLS.

Lastly, view profile. This enables managers to view their profile, which contains their own manager IDs, positions, phone numbers, e-mail addresses, employees' overview which involves employees' names, IDs and TDDs, and weekly overview. Managers are also allowed to edit their own information , such as phone number and e-mail address.

Chapter 4

System Design

4.1 System Architecture

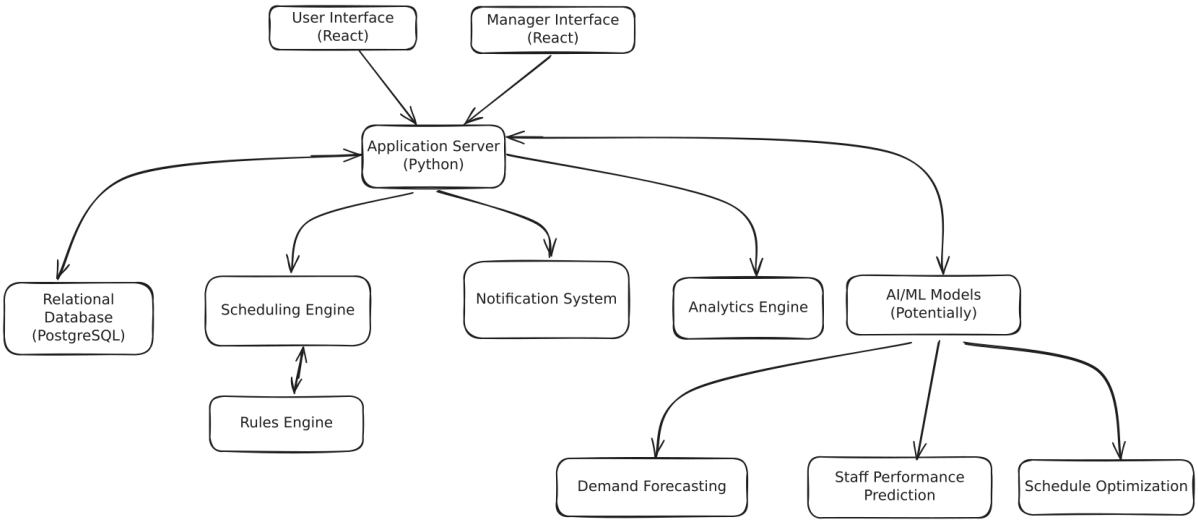


Figure 4.1: System Architecture

The system architecture is made up of many different components and systems surrounding the main application server. The components includes:

- **User Interface:** The User Interface for employees developed with React Js.
- **Manager Interface:** The User Interface for managers developed with React Js.
- **Relational Database:** The main database developed with PostgreSQL or SQLite (Not much differences between either).
- **Scheduling Engine:** The system for schedule creation with main logic developed with JavaScript and Python (Reasoning will be explained in Section 6.2).
- **Rules Engine:** The system that works closely with the scheduling engine to implement all the constraints and rules such as max working hours (hard constraints) and preferred shift hours (soft constraints). This will also be developed with JavaScript and Python.

- **Notification System:** The system that sends notifications to managers for conflict management and leave requests received. For employees, it will send the results of leave requests as well as when there are changes to their shifts.
- **Analytics Engine:** The system that calculates all the KPI and analyses data of employees to improve the algorithms. This system is a lower priority task as it is not one of the main functionalities.
- **AI/ML Models:** This is the AI/ML models that can help with predicting employee tardiness patterns, demand prediction and other data that can help with improving the algorithm for better accuracy. This system is not a functionality, it is an extra component that will be made if there is extra time after the high priority tasks.

All those components will connect to the main application server which serves as an API between every component and be the foundation of our whole system.

4.2 UML Diagrams

4.2.1 Use Case Diagram

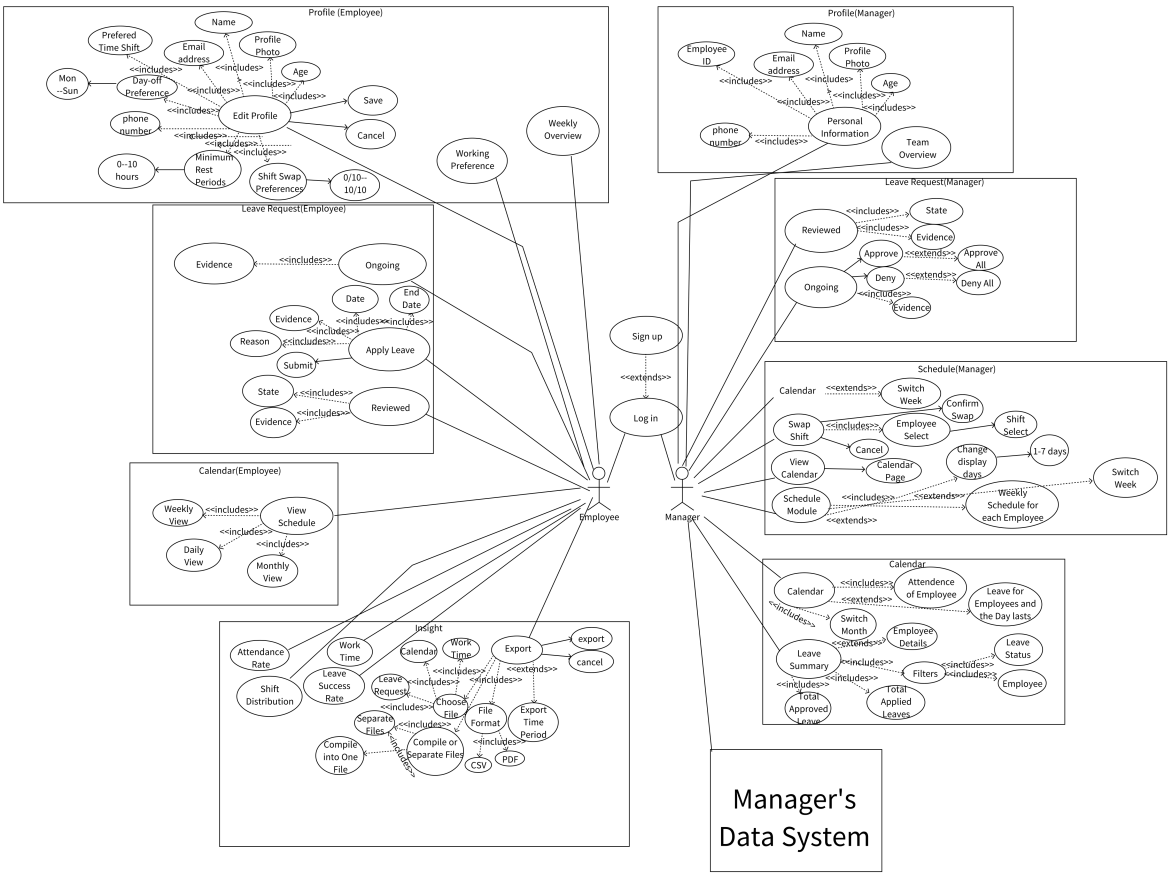


Figure 4.2: Use Case Diagram

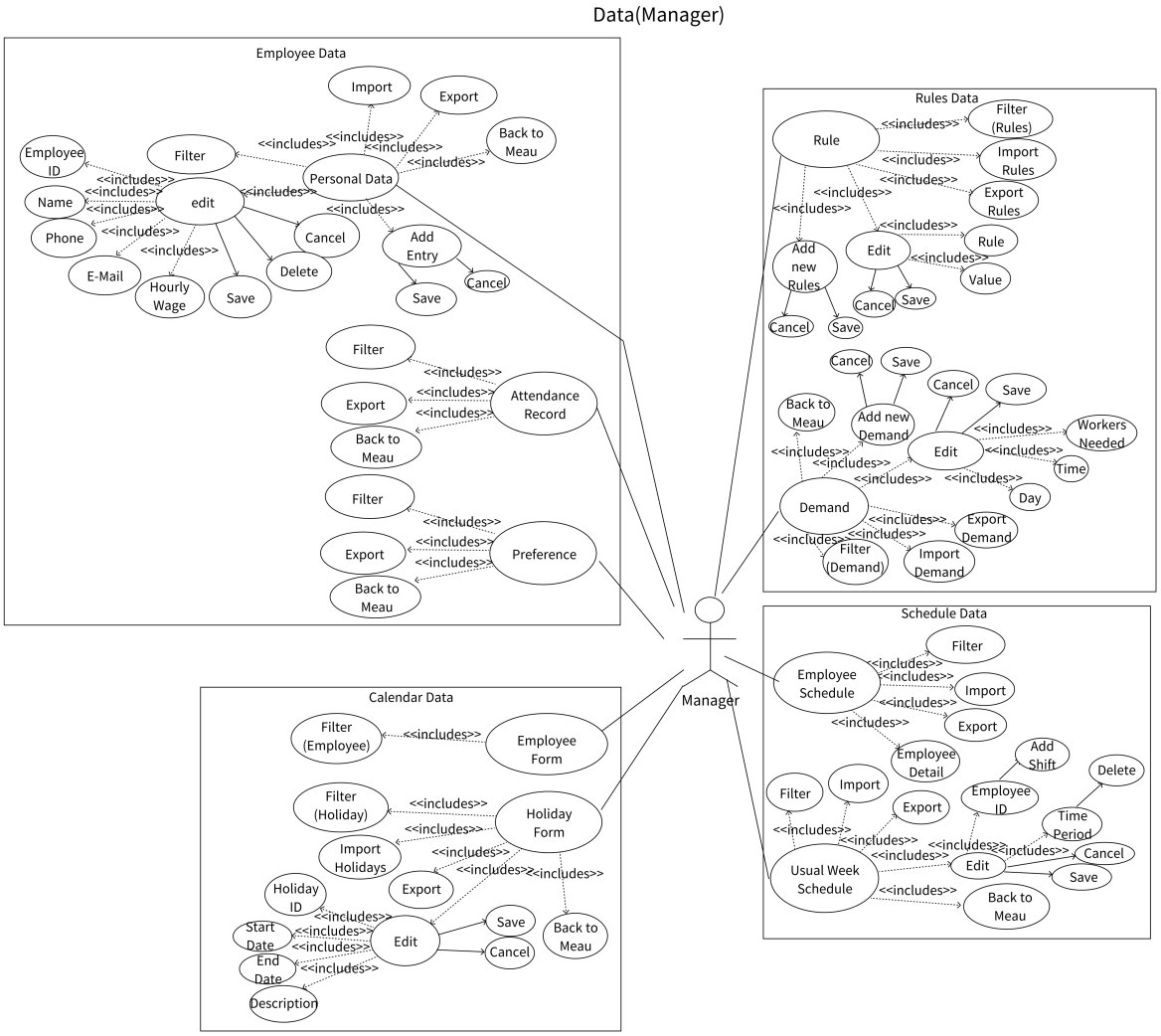


Figure 4.3: Use Case Diagram - Manager Data

The Use Case Diagram in Figure 4.2 shows the different functions and permissions of the dynamic scheduling system in the employee and manager roles. Employees' functions include calendar viewing, check-in, submitting leave requests, and exporting insights/calendar. The manager's functions include viewing and editing employee schedules, viewing calendar detailing employee leaves and holidays, dealing with leave requests, making shift adjustments, setting up rules for employees to go to work (e.g. maximum working hours), and importing and exporting other data. Both employees and managers first need to sign up for an account and then log in to the system.

Both Employees and Managers have a profile system, leave request system, calendar system and insights (for employee)/data (for manager) system. Manager has the schedule system additionally.

In the profile system, both the employee and manager have the same personal information screen, which includes information such as name, phone number, etc. The employee's profile system also includes a Working Preference and Weekly Overview section, where the user can view their shift and fill in the working preference to make it easier

for the system and managers to arrange more reasonable time slots for their shifts (Figure 6.8). In contrast, the manager's profile system has a team overview, which makes it easy for each manager to view information about their employees and their work (Figure 6.36).

In the leave request system, the manager can view the leave requests in the reviewed or ongoing states and can approve or reject leave requests in the ongoing state with evidence. To make it easier for the manager to process leave requests efficiently, we have also added a batch-processing feature (Figure 6.12, Figure 6.13). Employee's leave request system consists of a section where employees can view past or current leave requests and a section where employees can fill out a leave request that includes the time, reason, and evidence for their leave (Figure 6.4, Figure 6.5).

In the calendar system, the employee keeps track of their scheduling through a calendar that adjusts the number of days displayed (day, week, month) (Figure 6.1, Figure 6.2, Figure 6.3). The manager has a different calendar system compared to the employee, they have a large calendar that allows them to see the attendance of the employee and also the holiday remaining for the employee on leave. In addition, the manager also has a leave summary that includes filters, which not only filter what is shown on the calendar for the manager but also contain employee details (Figure 6.10).

In the insight system of employees, the employee can export information for a selected period, including hours worked, attendance, days off and shift details. After a specific period has been selected, the user selects the output file format, file content and single or multiple file output through the export function (Figure 6.6, Figure 6.7).

The manager's data system, as depicted in Figure 4.3, comprises four components: Employee Data, Rule Data, Calendar Data and Schedule Data. Employee data encompasses personal information, attendance records and preferences that can be edited or added by the manager. It also allows for importing or exporting through Excel and other file formats (Figure 6.21, Figure 6.24, Figure 6.25). In calendar data, the manager can view employees' leave records and export files via Excel etc., while also being able to import and edit holiday information to remind employees of vacation dates (Figure 6.26). Rules data includes regulations that all employees must adhere to (e.g., maximum working hours), which can be edited, added or imported/exported via Excel. Additionally, it contains the number of employees required for each shift on every workday - this is editable/added/importable/exportable by the manager as well (Figure 6.28). Finally, schedule data is divided into employee schedules and usual week schedules; managers may import/export each employee's schedule within their respective category (Figure 6.31, Figure 6.34).

In the manager schedule system, the manager can view a general overview of the scheduling of all employees for a given week through the calendar and can switch time slots. In addition, the manager can use the swap shift function, using this function they must select the employee who needs to be swapped and the shift, then fill in the employee to swap. Afterwards, the system will change shifts based on the employee's scheduling and make recommendations (if the shift swap is unreasonable). To make it easier for managers to get more detailed scheduling information, a function that switches to the calendar page is set in the system (Figure 6.14).

4.2.2 Sequence Diagrams

Schedule Creation

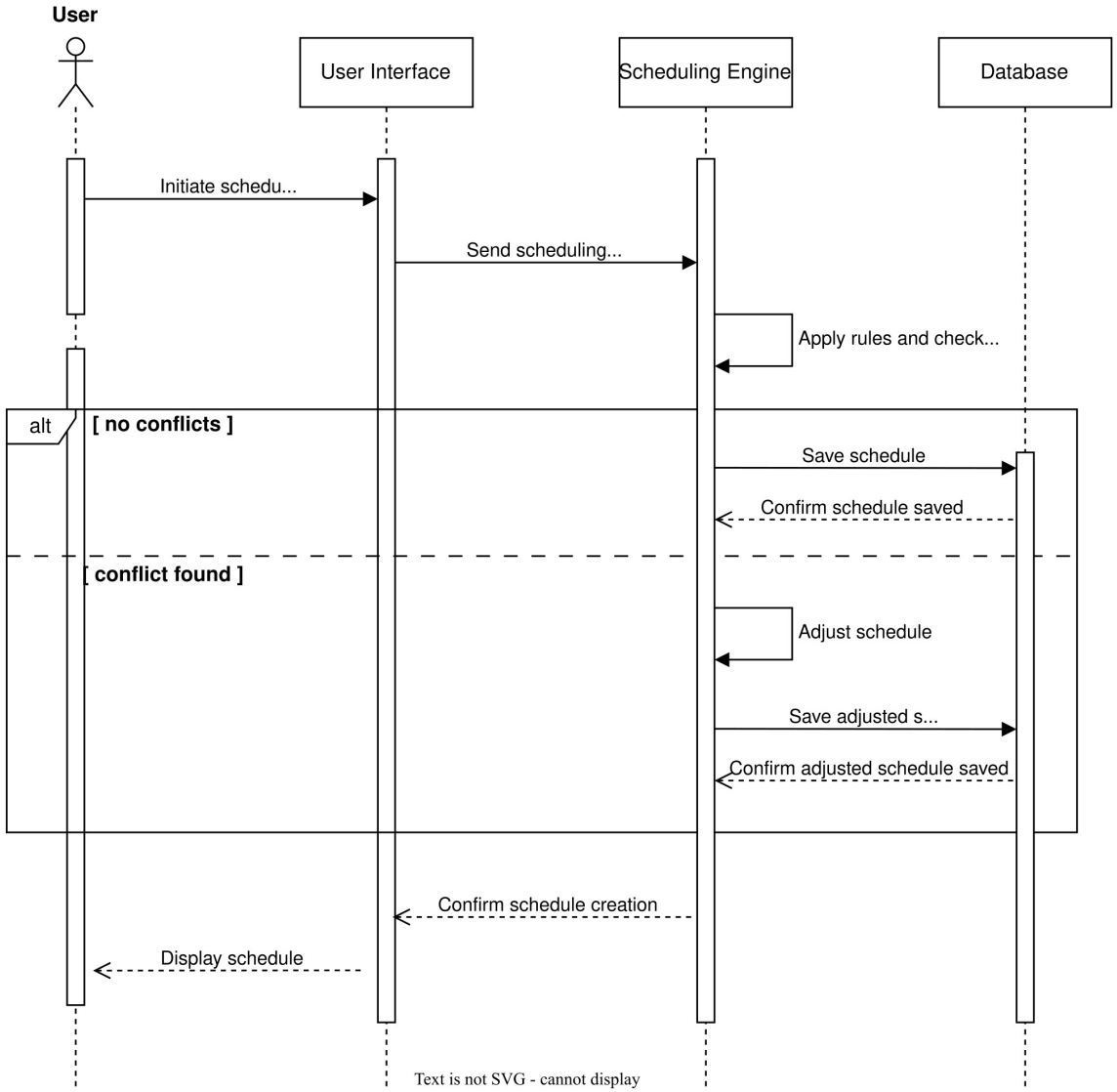


Figure 4.4: Schedule Creation

For Schedule Creation, firstly the user would initiate a schedule request. This could be either importing new data or just adding new real time data that may change the schedule. The user interface then sends a request to the scheduling engine which works with the rules engine to make a new schedule. Once everything is validated with no conflicts, it saves the schedule into the database which in return replies with confirmation. However, if there is conflict, it adjusts the schedule until there is no conflict left. Finally we send the schedule back to the user interface which displays the new schedule to the user.

Staff Leave Request

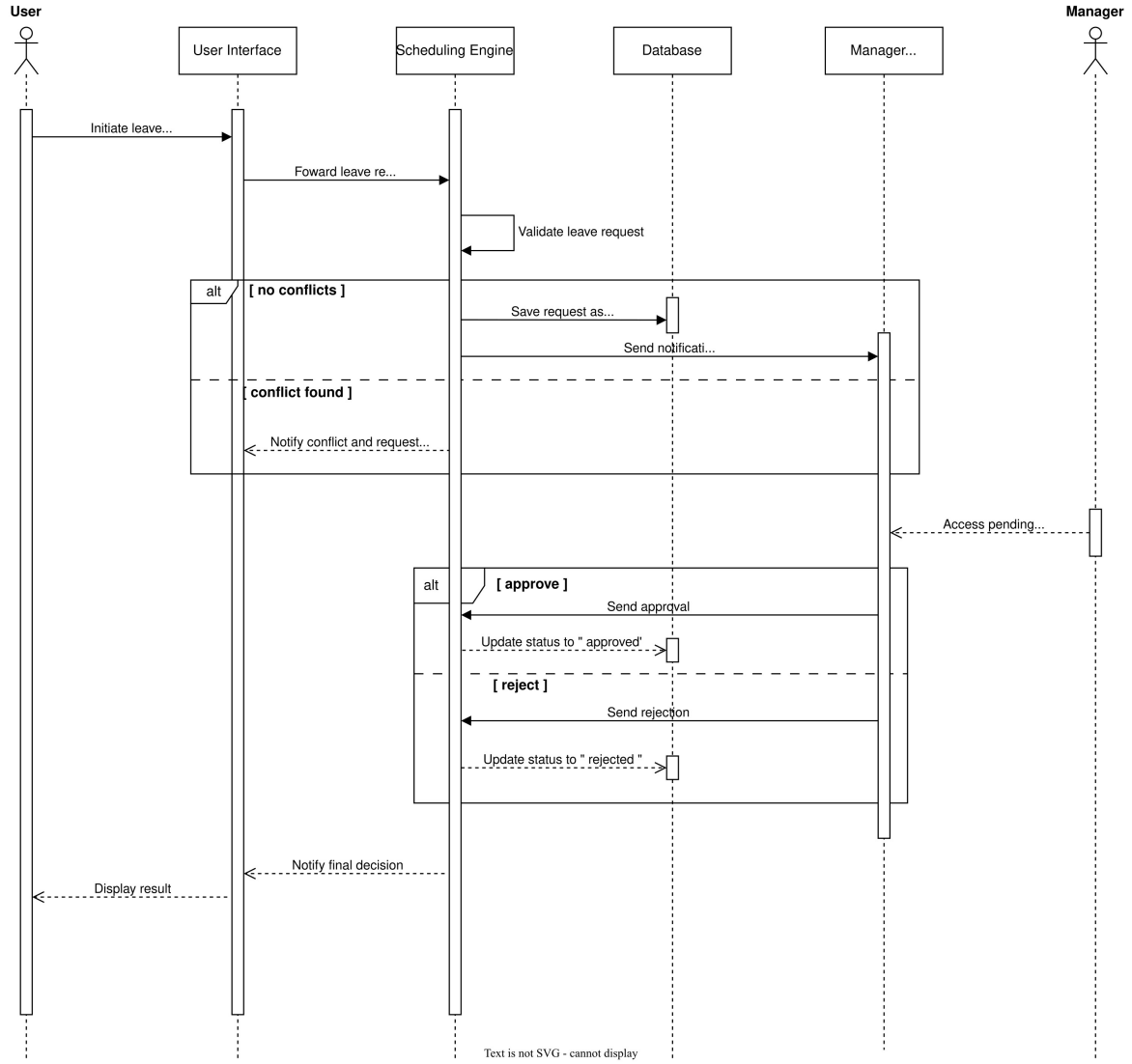


Figure 4.5: Employee Apply Leave

For Employee Leave Requests, the employee firstly makes a request to the user interface using the apply leave form. The user interface sends the input to the scheduling engine which automatically validates the request. If the apply leave requested time has conflict, it will send a notification back to the user. If the validation is passed, the request is sent to the database which then sends to the manager. The manager will then either approve or reject the request. The decision will then be sent back to the user.

User Login

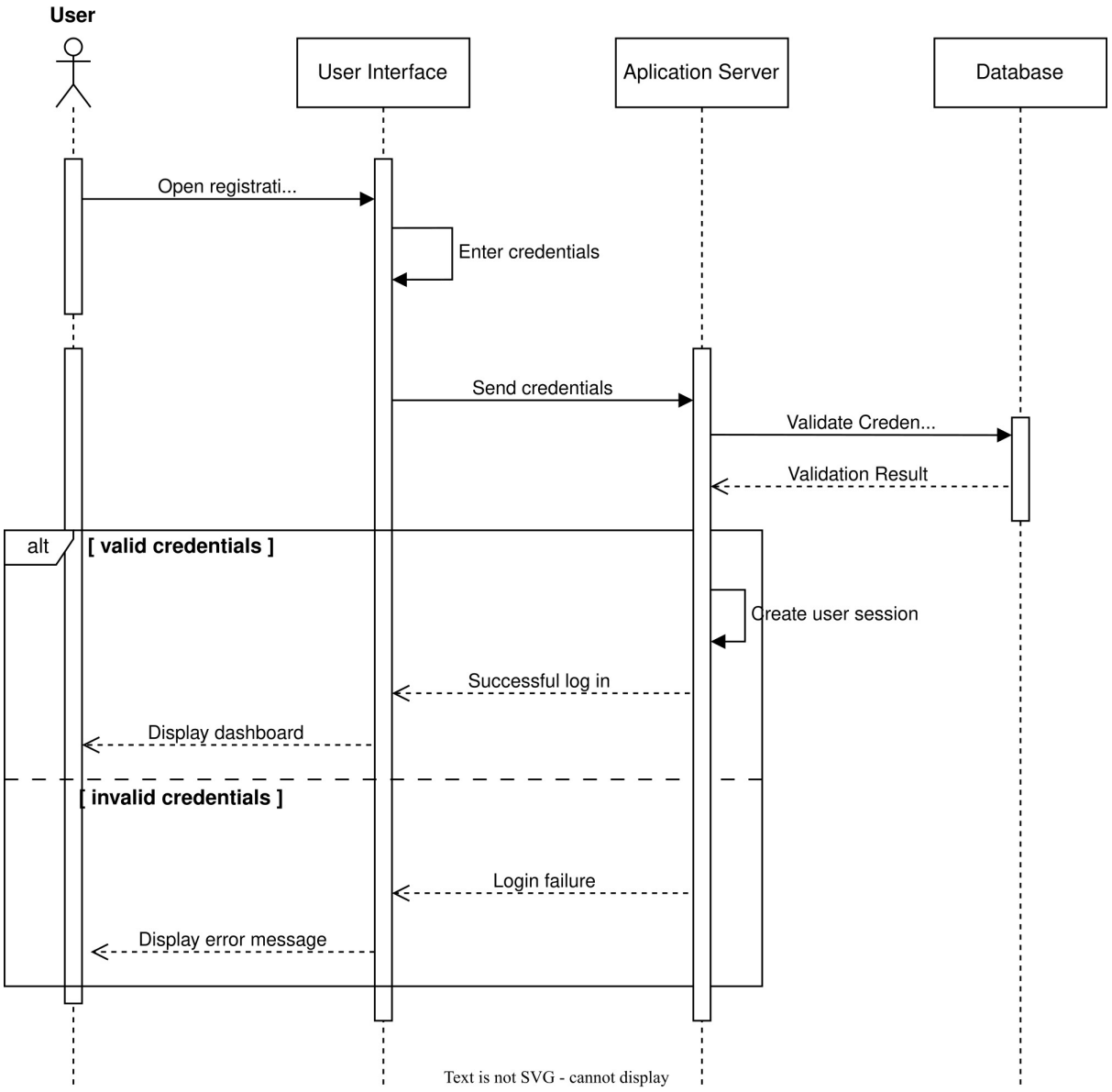


Figure 4.6: User Login

For User Login, the user first enter their credentials and data like username and password. Once they hit submit, the data is sent to the application server which validates the credentials with the database. If the credentials are valid, then the user will be redirected to the main dashboard. If the credentials are invalid, then the user will be denied entry and a error message will be shown to the user.

4.2.3 Class Diagram

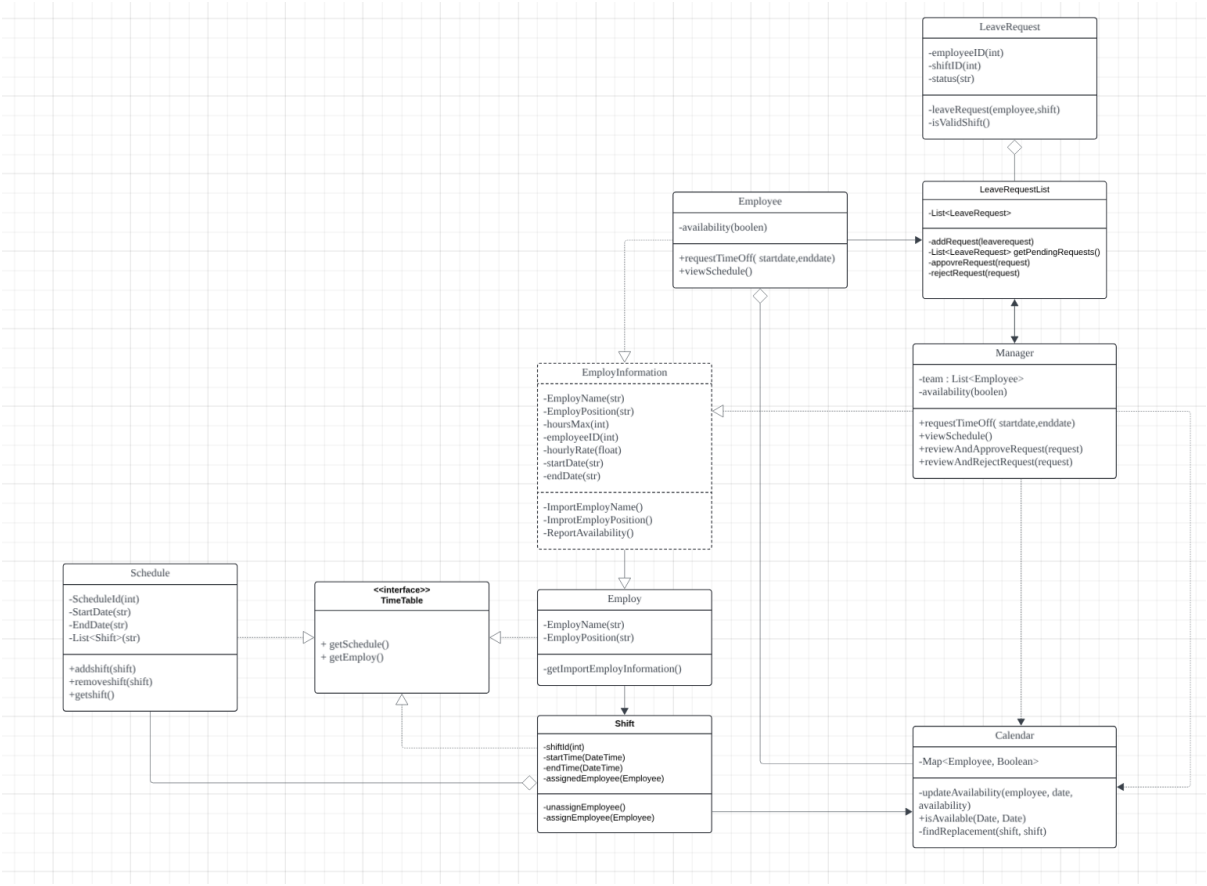


Figure 4.7: Class Diagram

The class diagram shown at Figure 4.7 is the basic class diagram for the main application server. It does not include the scheduling system nor the rules system. In the class diagram, the key functionalities are shown for the two users. For employees, they are able to request time off as well as view their own schedule. Not only that, the employees are also able to indicate their availability. For managers they are able to review leave requests sent by the employees and able to view schedules as well. For now this class diagram achieve the bare minimum thus not including import/export data or being able to check in. This class diagram focuses on providing a dynamic scheduling system without additional features.

Chapter 5

User Interface Design

5.1 User Interface (UI)

The UI design of this Dynamic Restaurant Staff Scheduling Tool aims to provide a clear and convenient experience for users when browsing and interacting with schedules. The target audience includes the restaurant manager, whose primary need is to be able to manage employees' schedules, and restaurant employees, whose primary need is to view their own schedules and apply for leave.

5.2 UI Designs

This section outlines the evolution of UI design and the enhancements made to improve user experience.

5.2.1 Initial Design

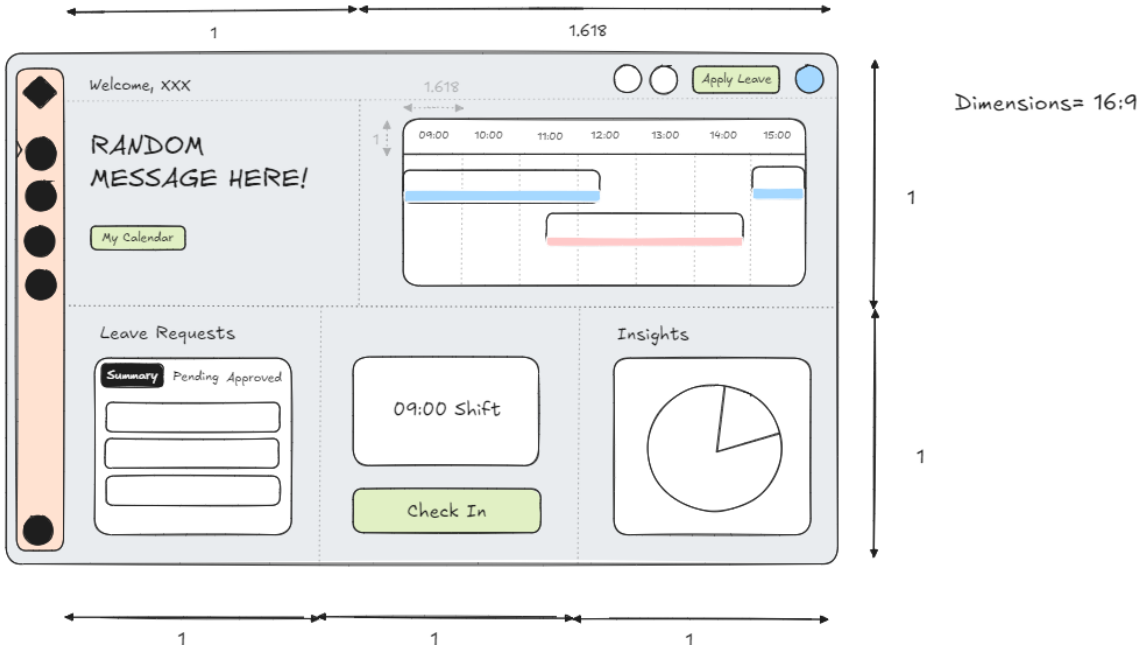


Figure 5.1: Initial Design

The theme and general look of our application was built on the initial design shown in Figure 5.1. This design was made using the website excalidraw, an online diagramming and brainstorming tool (Excalidraw n.d.). In terms of color matching and accessibility considerations, our team chose a minimalist color scheme, using low-saturation colors to highlight the text on the web page. The web pages are mainly composed of white, gray, red and green, which avoids color clutter without being too monotonous, and can effectively distinguish different functional sections.

A study done by Nikolic et al. (2011), shown that younger people tend to prefer simpler shapes like triangles with the golden ratio and older consumers prefer more complicated shapes like hexagonal with the golden ratio which might be explained by the Gestalt psychology. Most restaurants hires workers around the age of 18 to 24 (National Restaurant Association 2024). This meant that they are not more likely to prefer a complicated shape rather than a simple shape. Thus, a rectangle with the golden ratio, which was used extensively in our design, can be said to be neither too simple nor too complex.

Initial Design Board

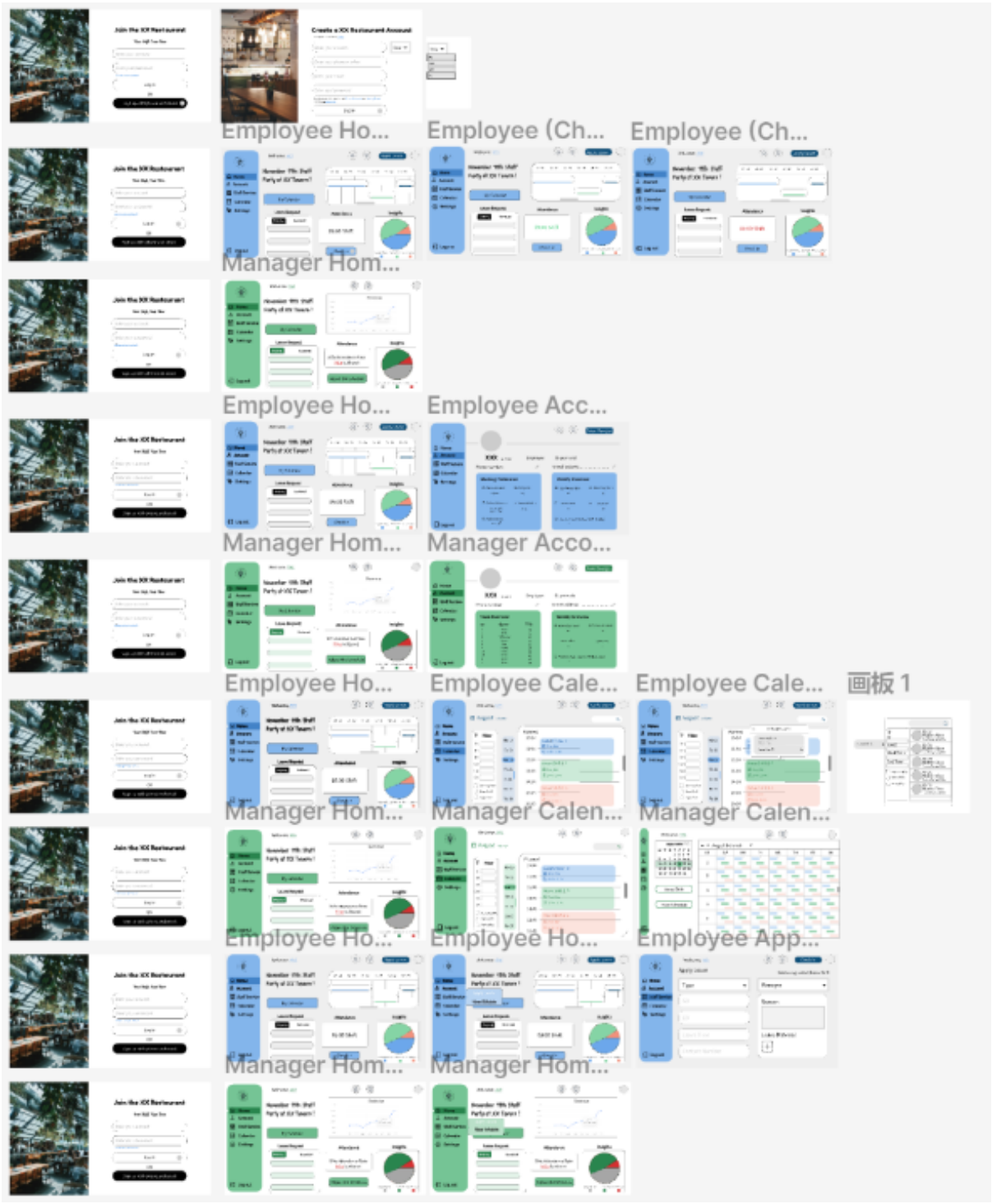


Figure 5.2: Design Board

Figure 5.2 shows the design board made by our design team. By combining the color and dimensions of the initial design and the designs shown in the design board, allows us to achieve an appealing web application design shown in Sub-Section 6.4.1.

5.2.2 UI Design Principles

Our team's design goal is usability, which ensures a clean, intuitive interface with straightforward navigation, and responsiveness. The styling was integrated with the components by the module CSS, creating flexible layout components, and dynamically resizing charts and tables, which enabled the interface to adapt to a variety of screen sizes, ensuring a consistent experience across desktop and mobile devices (Figure 5.3, Figure 5.4). In other words, a responsive layout enables the system to be more inclusive and adaptable, which enhances user experience and engagement.

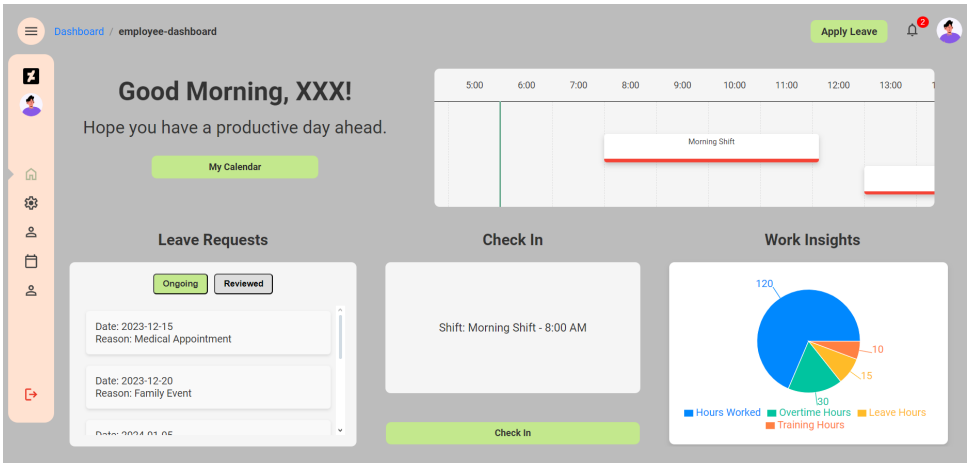


Figure 5.3: Dashboard Full Width

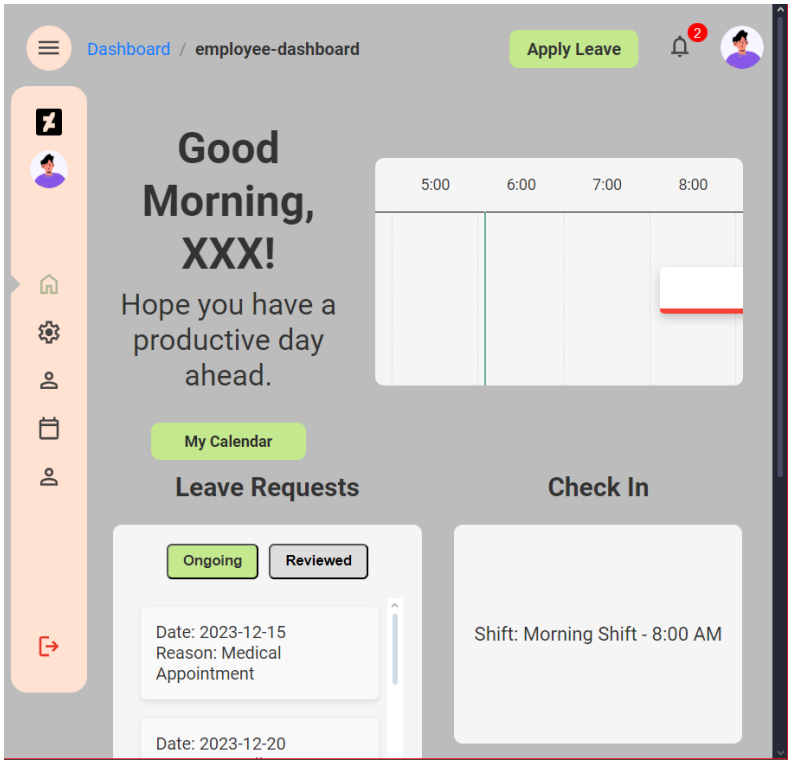


Figure 5.4: Dashboard Half Width

For typography, our team chose the "Roboto" font (Arial and Helvetica as backup when Roboto doesn't load) because it's easy to read on both desktop and mobile devices and has appropriate bold and size variations to highlight key points. In the web page layout, our team used a card-style layout to distinguish different functions, allowing users to find the functions they need efficiently. At the same time, the appropriate spacing between images and text enhances readability.

Chapter 6

Implementation Methods

6.1 Software Development Method

Our team used Windows as the development environment due to its high level of adaptability, and Visual Studio Code as the development tool to deploy the React environment. During the development process, both React and Django/JavaScript runtime environments can be built smoothly on Windows to ensure development efficiency. The deployment process is designed to ensure cross-platform consistency. To enable the software to be used by a wider group of users, our team has also ensured support for Mac. The final Dynamic Restaurant Staff Scheduling Tool will be able to run on Windows 10, Windows 11 and Mac OS as long as they are running on a suitable web browser.

6.2 Programming Languages

During implementation, our team used a variety of programming languages and technical approaches depending on the different aspects of the software.

In the front end, our team used ReactJs with Vite, whose componentized design facilitates high reusability and dynamic interface interactions. Furthermore, the high-performance development environment provided by Vite enables the rapid development of responsive pages.

In the back end, our team chose JavaScript as the main language, with its compatibility with our ReactJs frontend. We understadn the struggles JavaScript might have when it comes to mathematical logic. This meant that python will also be used, with its rich libraries and frameworks (e.g. Django) that can efficiently support complex business logic processing, especially for real-time dynamic scheduling updates.

Finally, in the aspect of databases, we will use SQLite as the main relational database for storing scheduling information and user data according to the project requirements.

6.3 Software and Tools

In addition to Jira for project management (Atlassian 2024), our team also used other software, websites and tools in project development. GitLab is used as the version control and collaboration platform for code management. Visual Paradigm, Draw.io and Lucidcharts are used as UML diagram creation tools (Ltd. 2024) (Inc. 2024). js.design (即时设计) and excalidraw are used to make prototypes and UI designs (js.design 2024) (Excalidraw n.d.).

6.4 Results of Implementation

This section shows the progress made in initial implementation of the project. It outlines the features that have been implemented, feedback received from stakeholders, challenges encountered, and the outcomes achieved so far.

6.4.1 Implemented Features

Employee Features

- **Schedule Visualization:** Employees are able to view their daily, weekly, and monthly schedules.

Figure 6.1: Daily Schedule

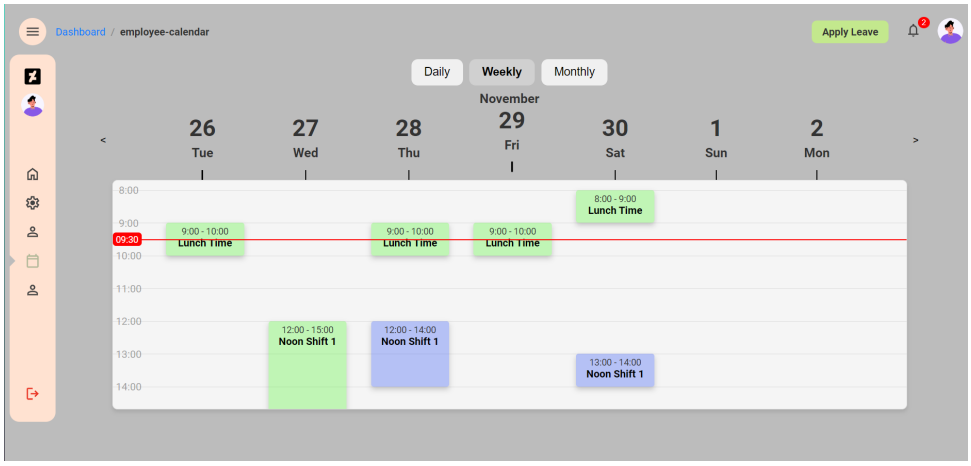


Figure 6.2: Weekly Schedule

Figure 6.3: Monthly Schedule

- **Leave Request Submission:** Employees can submit leave requests by specifying the dates and reason in the leave request form.

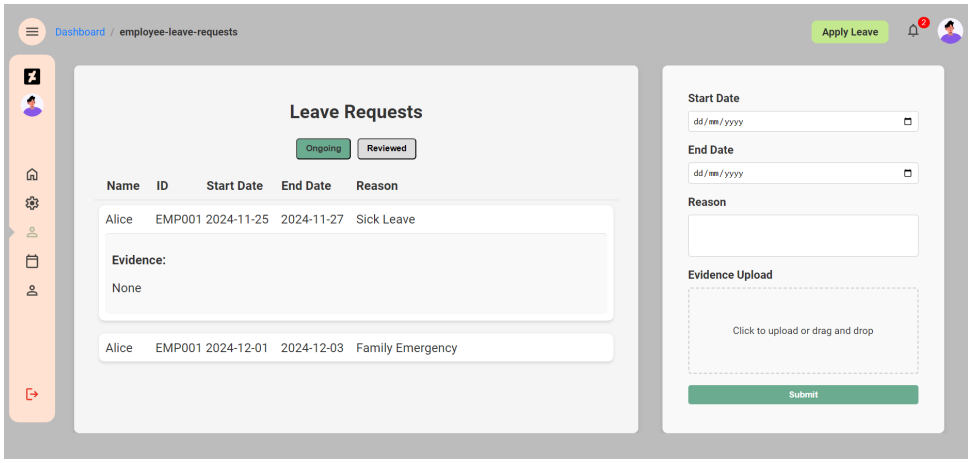


Figure 6.4: Ongoing Leave Requests

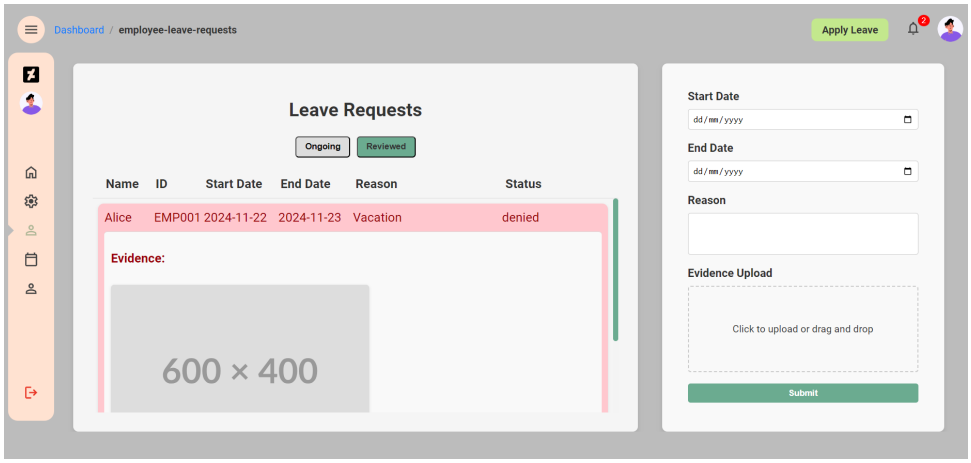


Figure 6.5: Reviewed Leave Requests

- **Export Data:** Employees can export their working data in the format they preferred. They are able to choose to separate files or compile into one file.

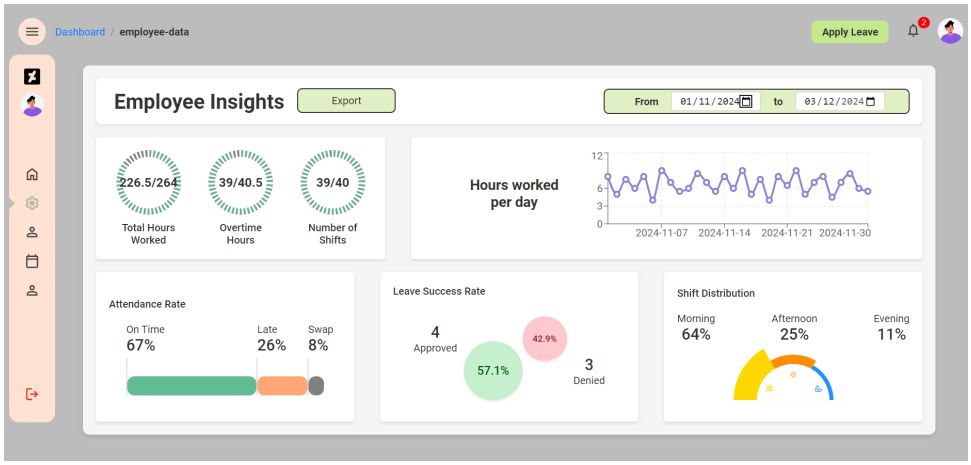


Figure 6.6: Employee Insights Page

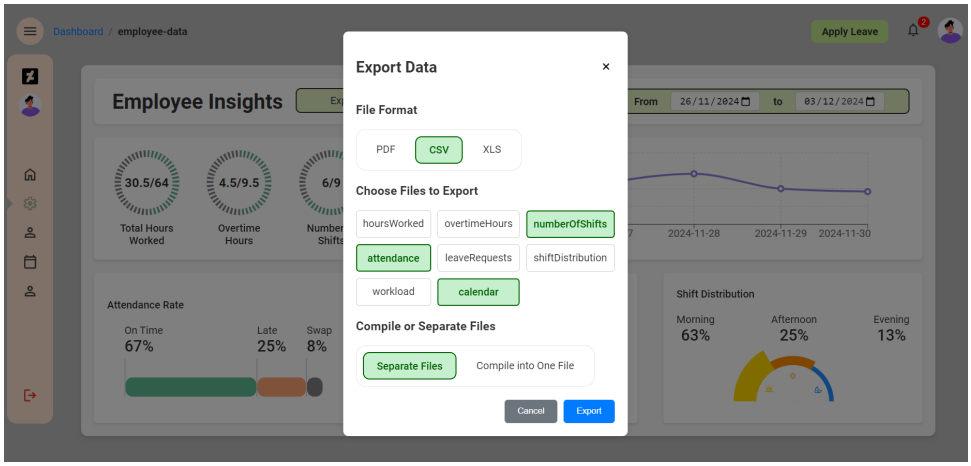


Figure 6.7: Export Data Form

- **Edit Profile:** Employee can view their personal details, working preference and weekly overview. Additionally,they edit their personal information, preferred shift time, minimum rest periods, day off and shift swap preferences.

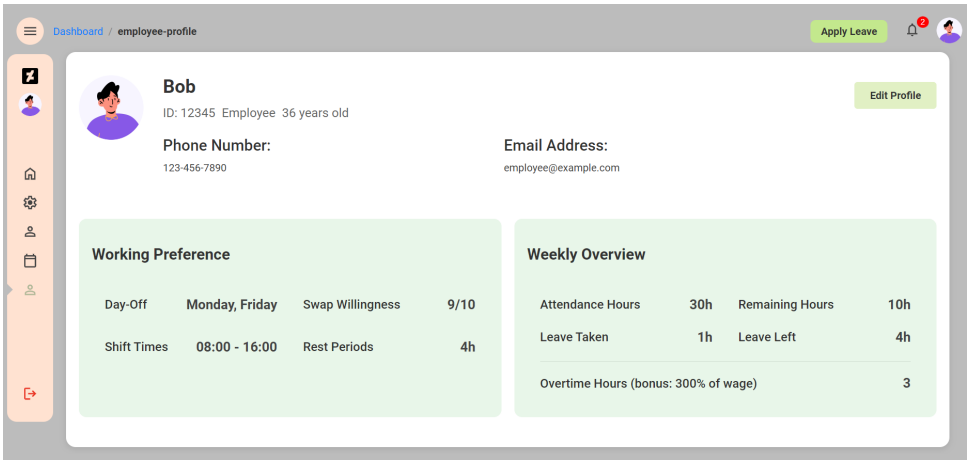


Figure 6.8: Edit Profile Page

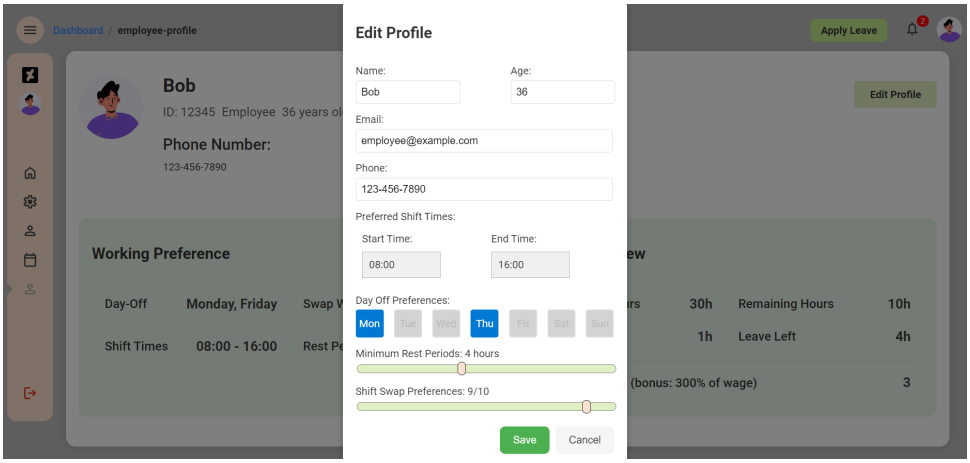


Figure 6.9: Edit Profile Form

Manager Features

- **View Calendar:** Manager are able to view a calendar detailed with employee leaves and holidays.

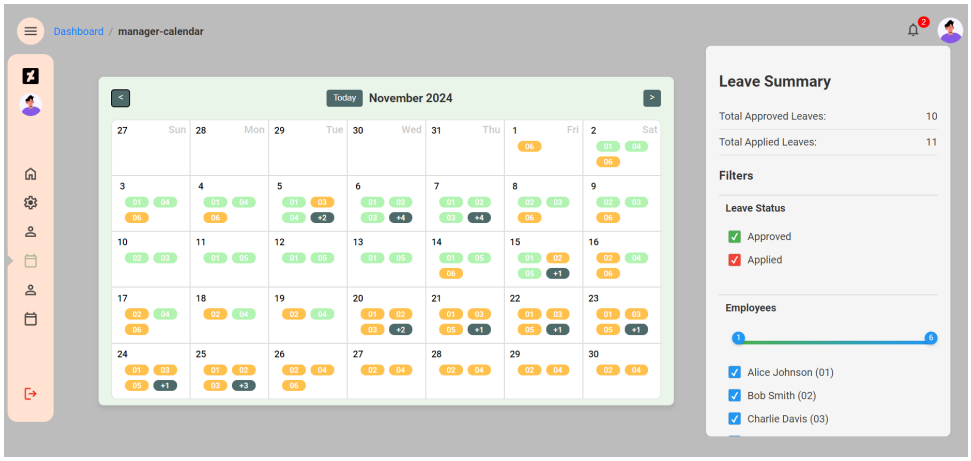


Figure 6.10: Manager Calendar

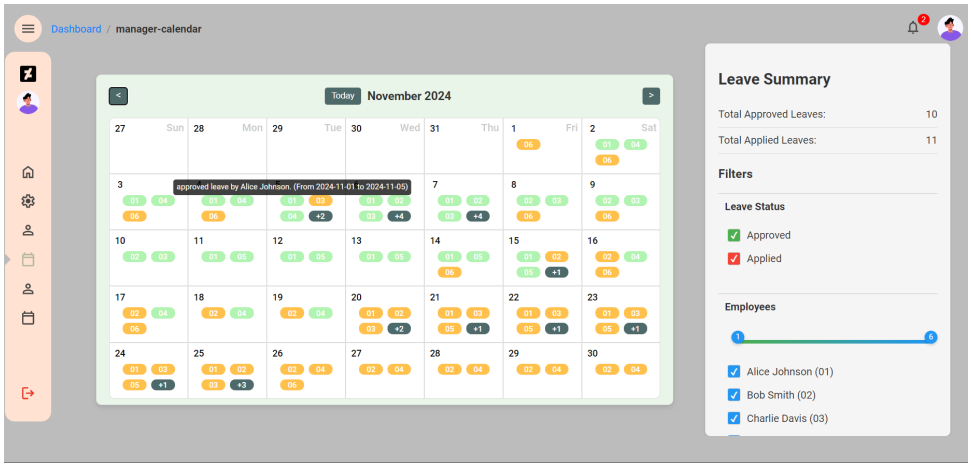


Figure 6.11: Manager Calendar Hover

- **Review Leave Request:** Manager are able to approve or deny employee's leave requests.

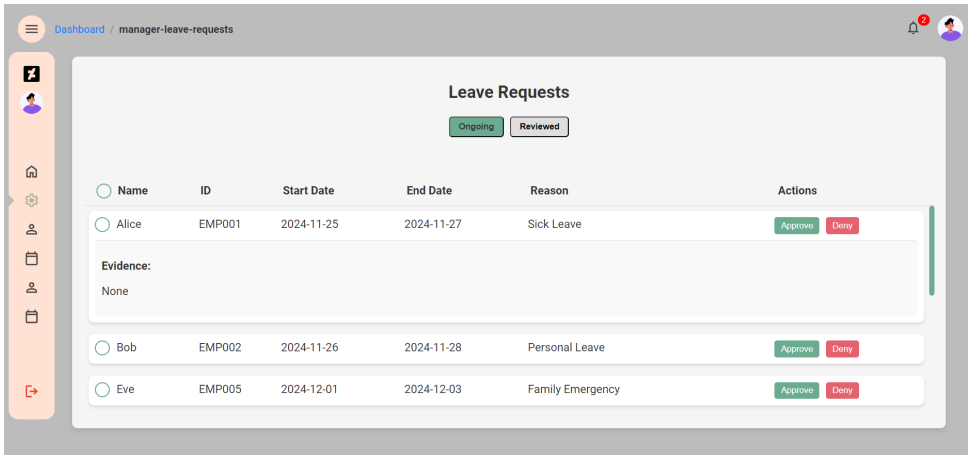


Figure 6.12: Manager Ongoing Leave Requests

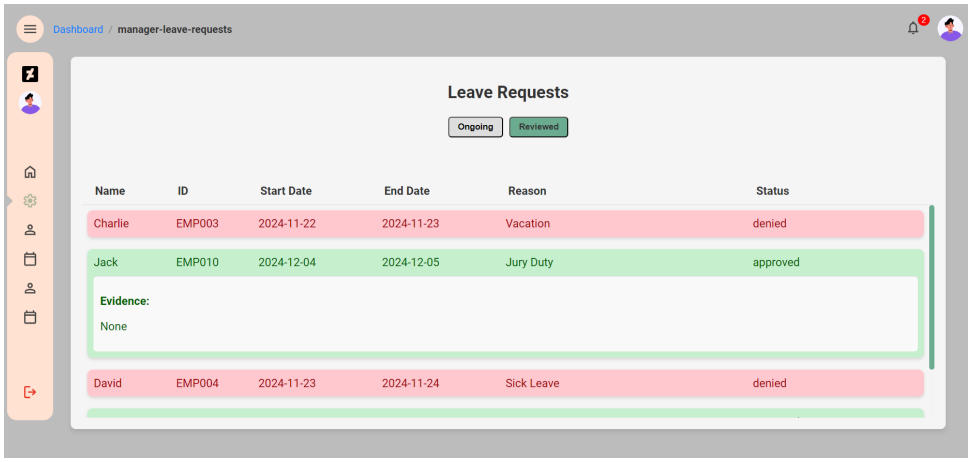


Figure 6.13: Manager Reviewed Leave Requests

- **Schedule Management:** Managers can view employee's schedules from a day to seven days. Manager can swap employee's shifts.

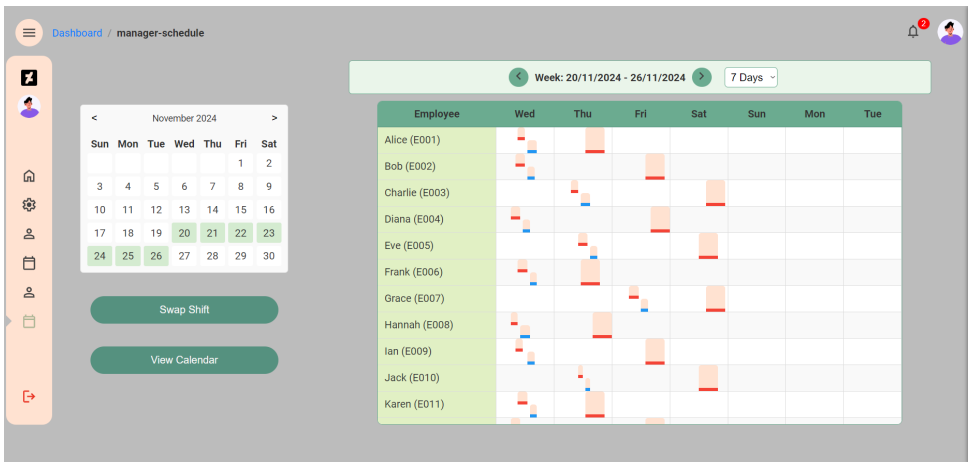


Figure 6.14: Manager Schedule Page

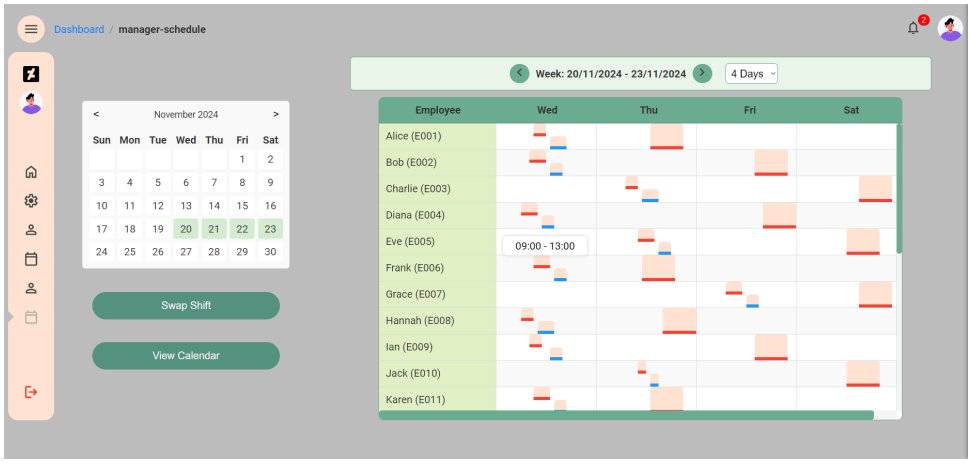


Figure 6.15: Manager Schedule Page 4 Days View

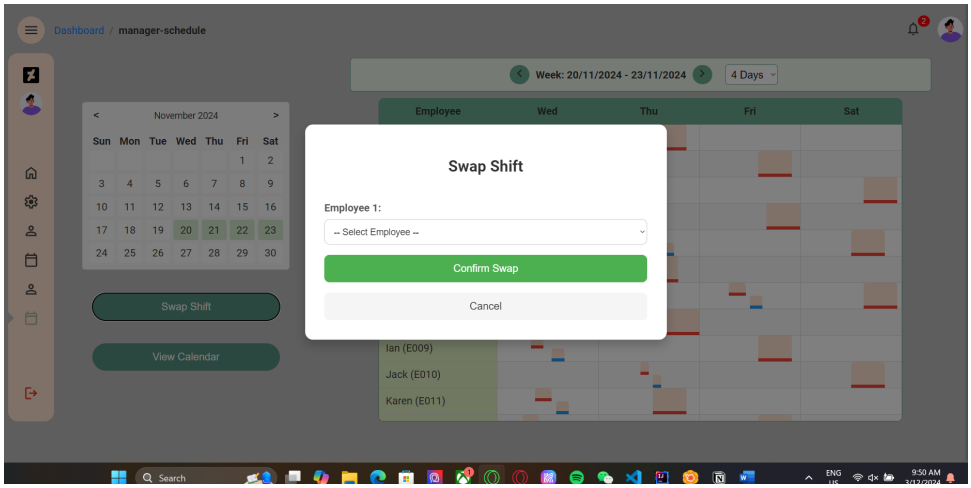


Figure 6.16: Manager Shift Swap Form Collapsed

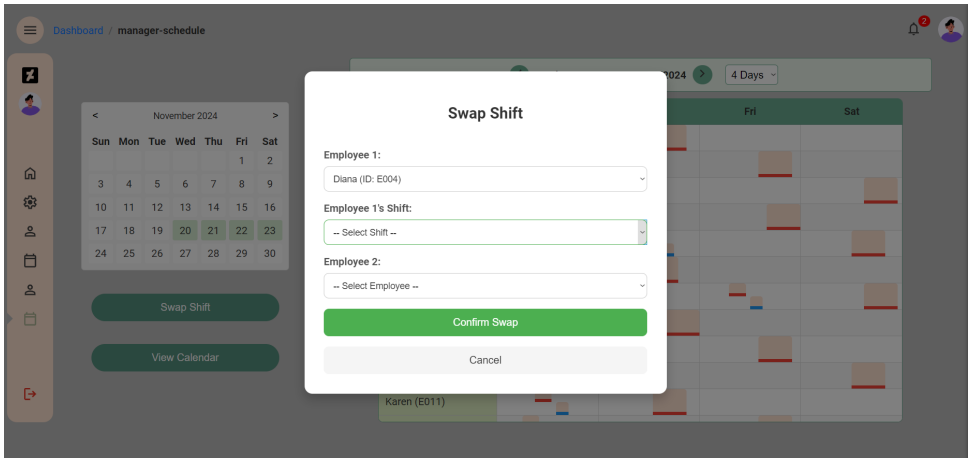


Figure 6.17: Manager Shift Swap Form Expanded

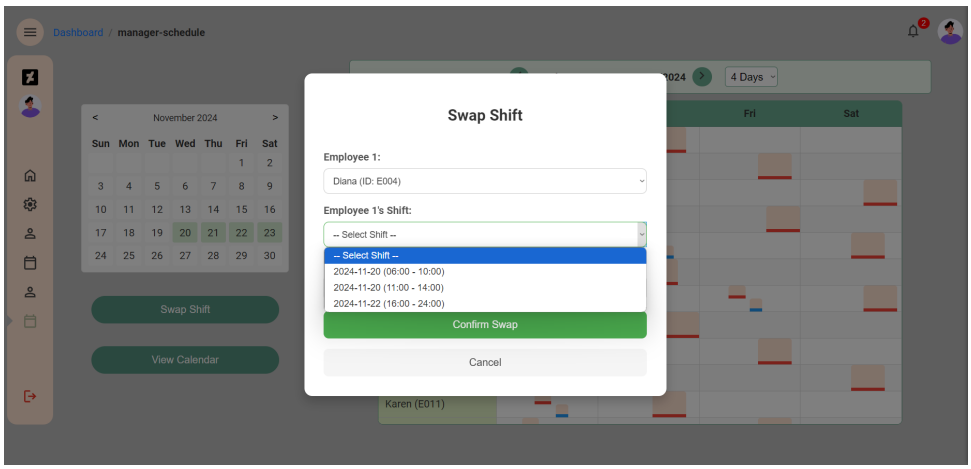


Figure 6.18: Manager Shift Swap Form Expanded 2

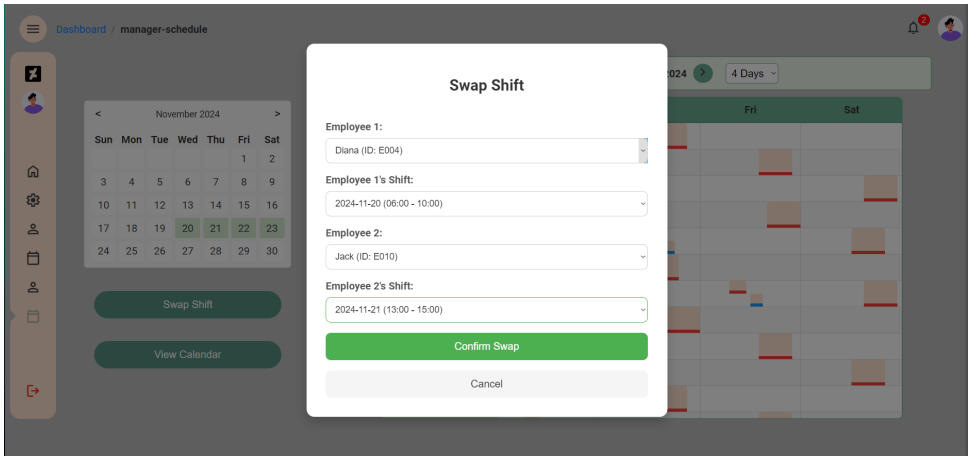


Figure 6.19: Manager Shift Swap Form Expanded 3

- **Import/Export Data:** Manager can choose to import or export employee data, calendar data, rules data and schedule data. For employee data as shown in Figure 6.21, FIgure 6.24, Figure 6.25 we can export employee personal data, attendance records and also their shifts preferences.

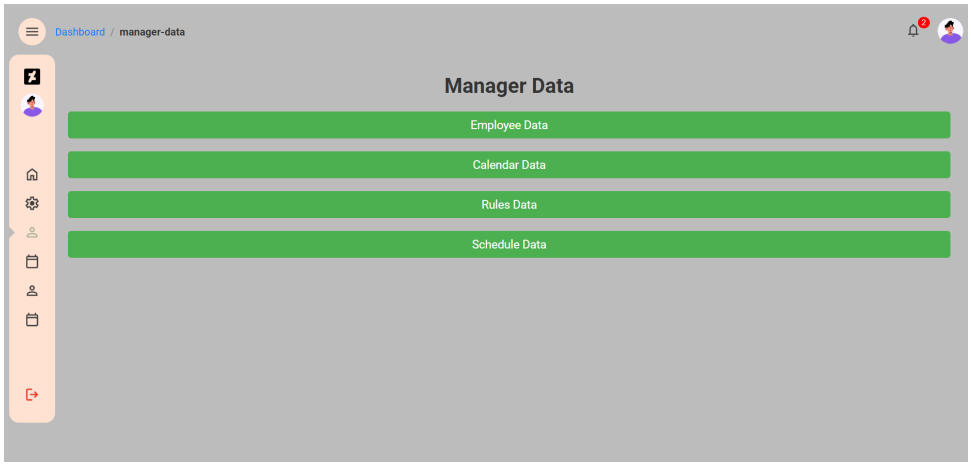


Figure 6.20: Manager Data Page

For employee data, as illustrated in Figure 6.21, managers can export personal data, attendance records, and shift preferences.

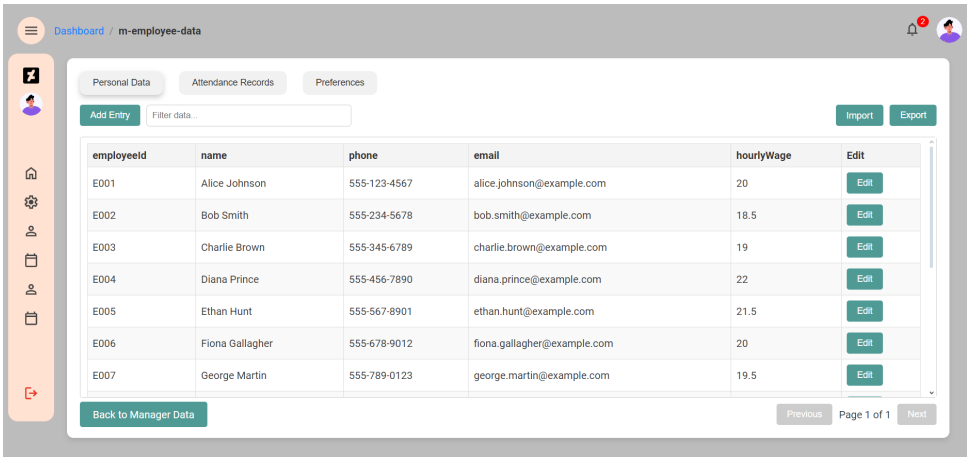


Figure 6.21: Manager Data on Employee Personal Data

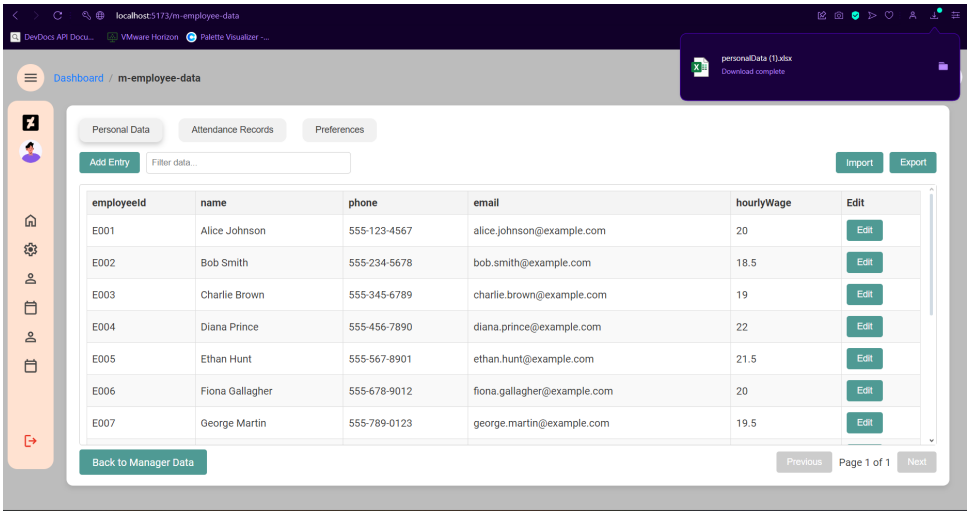


Figure 6.22: Manager Export Employee Personal Data

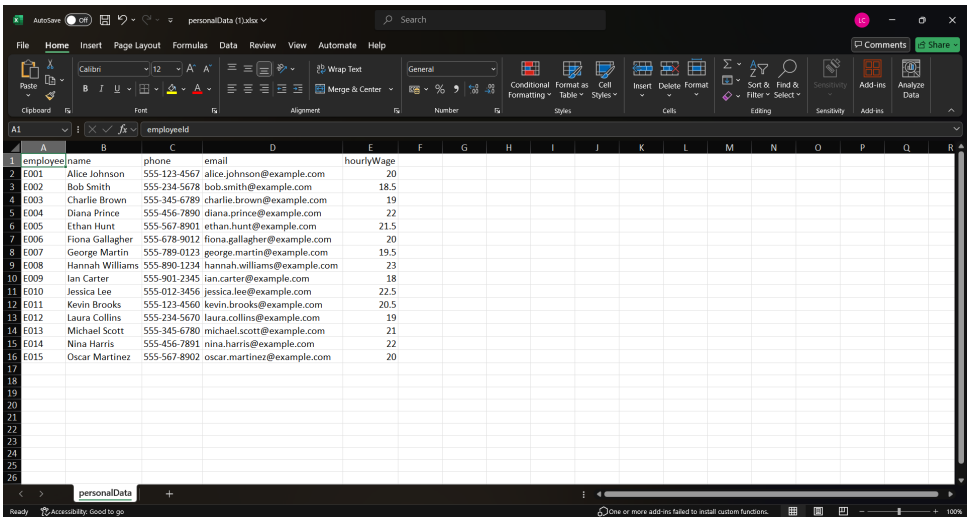


Figure 6.23: Employee Personal Data Excel

For the Attendance Records and Preferences, they follow the same procedure to export their data.

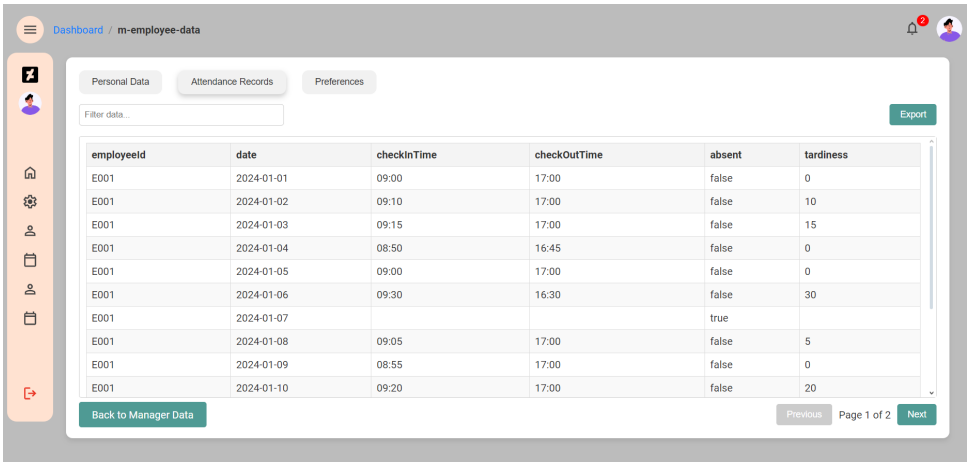


Figure 6.24: Manager Data on Employee Attendance Records

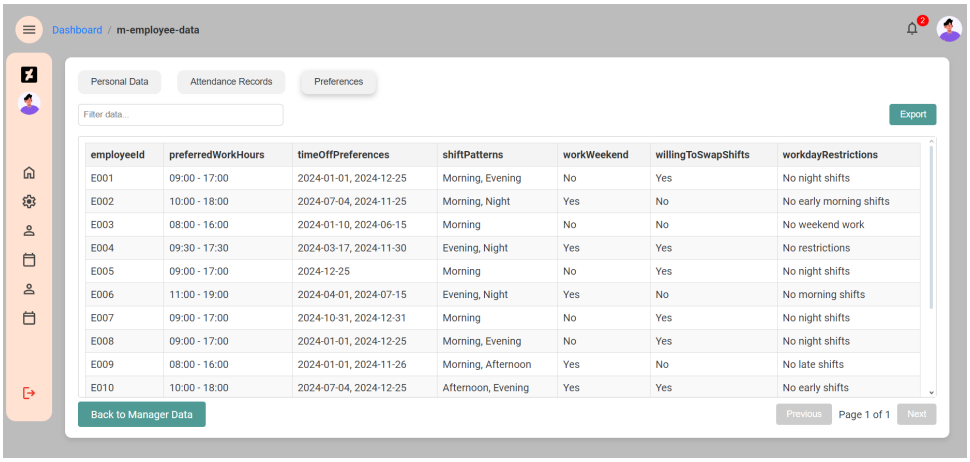


Figure 6.25: Manager Data on Employee Preferences

For calendar data in figure 6.27, manager are able to edit holiday details.

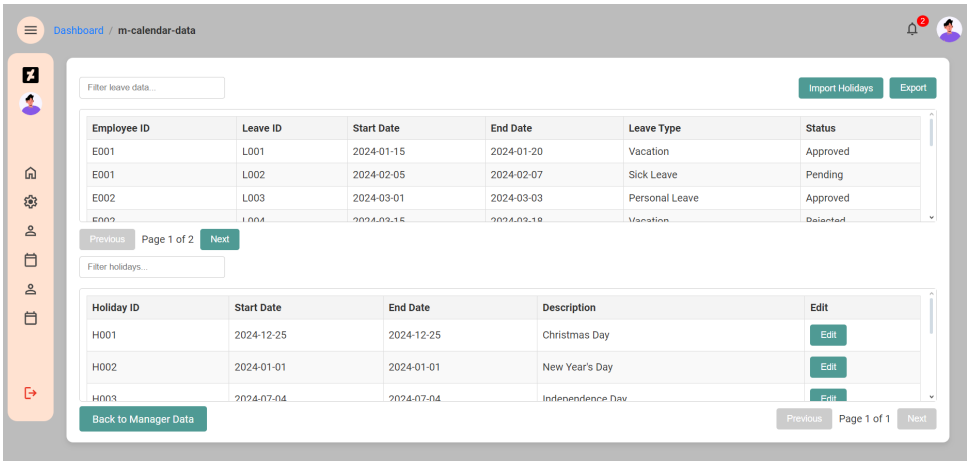


Figure 6.26: Manager Data on Calendar Data

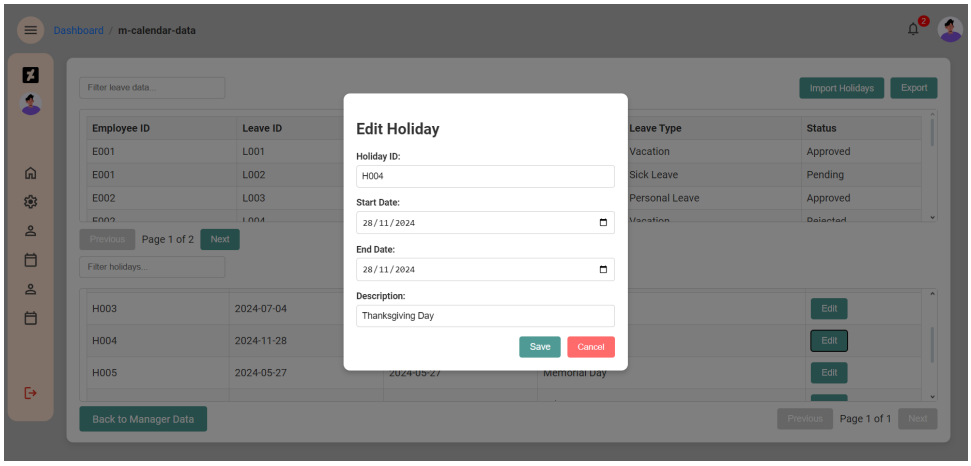


Figure 6.27: Manager Edit Holiday Data Form

As illustrated in Figure 6.28 , manager are able to edit and add both rules and demands.Additionally, managers can use filter to efficiently locate specific rules and demands.

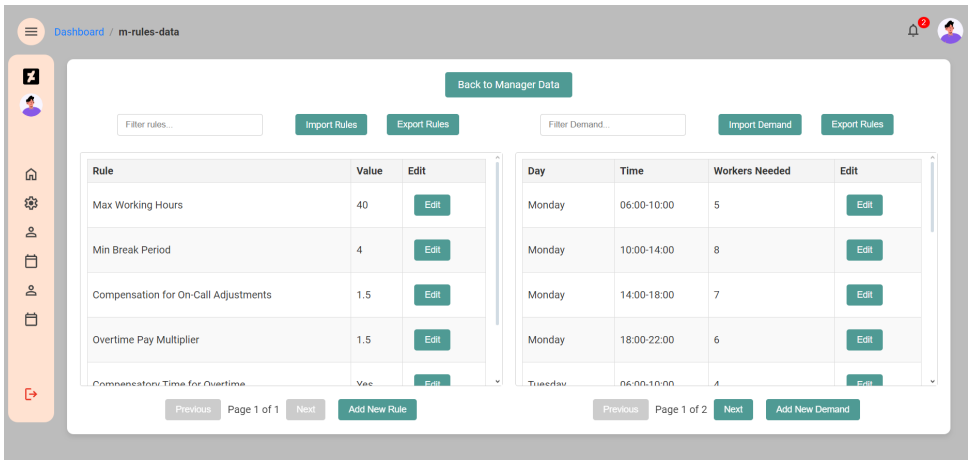


Figure 6.28: Manager Data on Rules

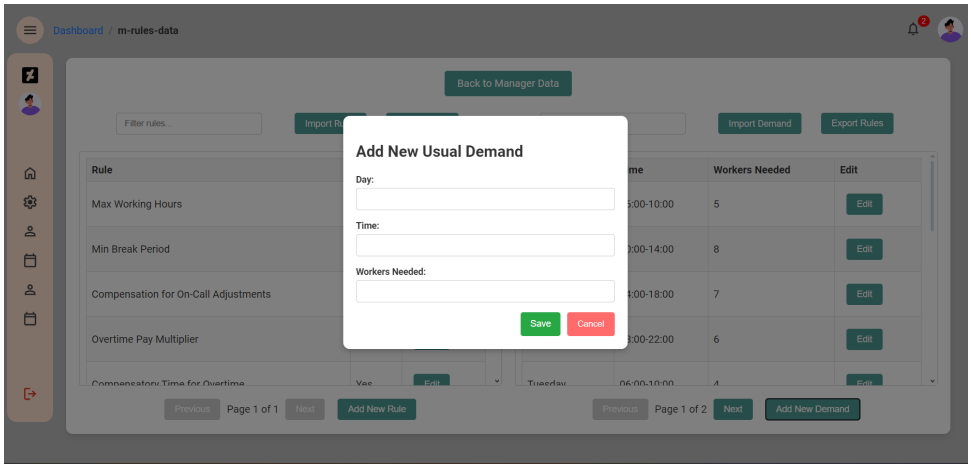


Figure 6.29: Manager Add New Demand

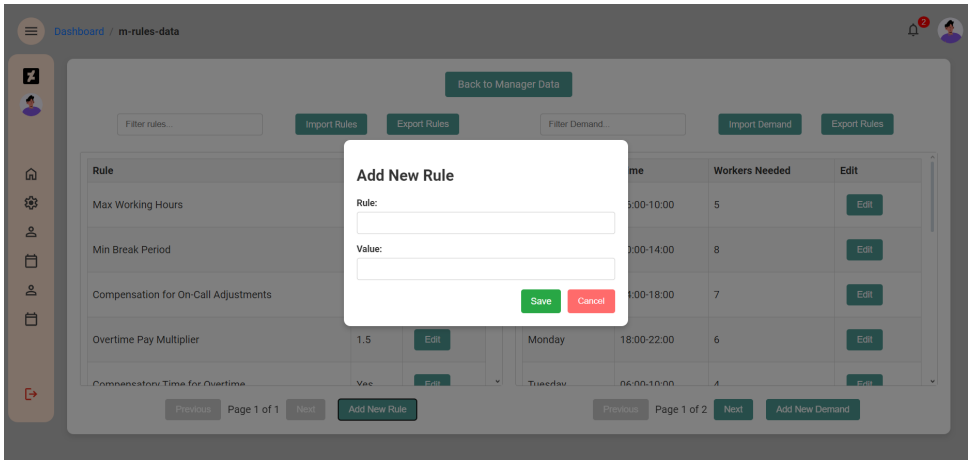


Figure 6.30: Manager Add New Rule

As shown in Figure 6.31, managers can export or import employee schedules as well as the regular weekly schedule, shown in figure 6.34.

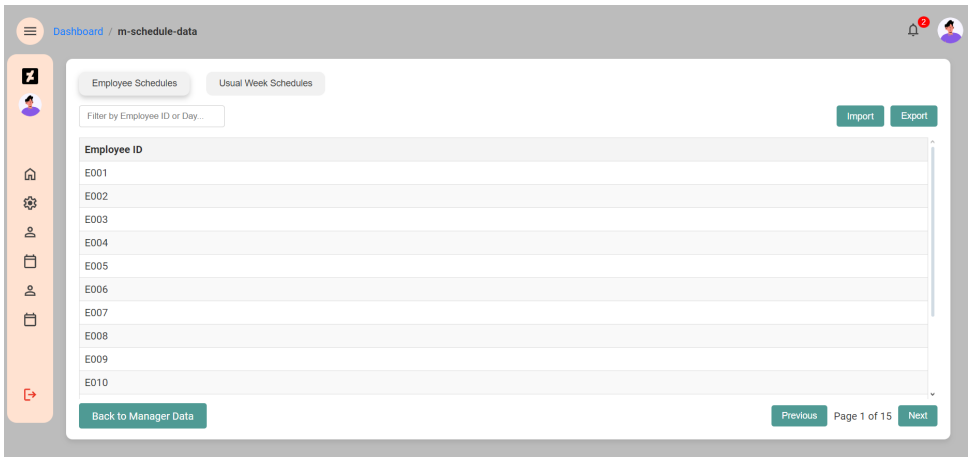


Figure 6.31: Manager Data on Employee Schedules Collapsed

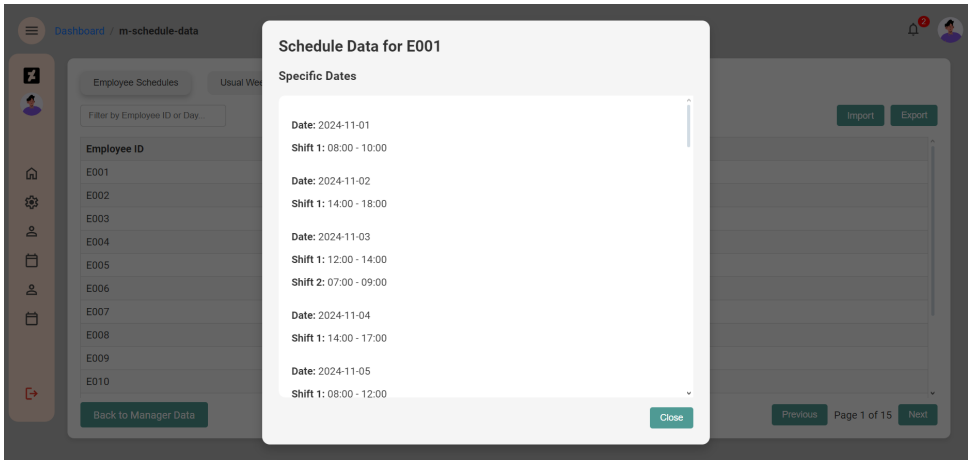


Figure 6.32: Manager Data on Employee Schedules Expanded

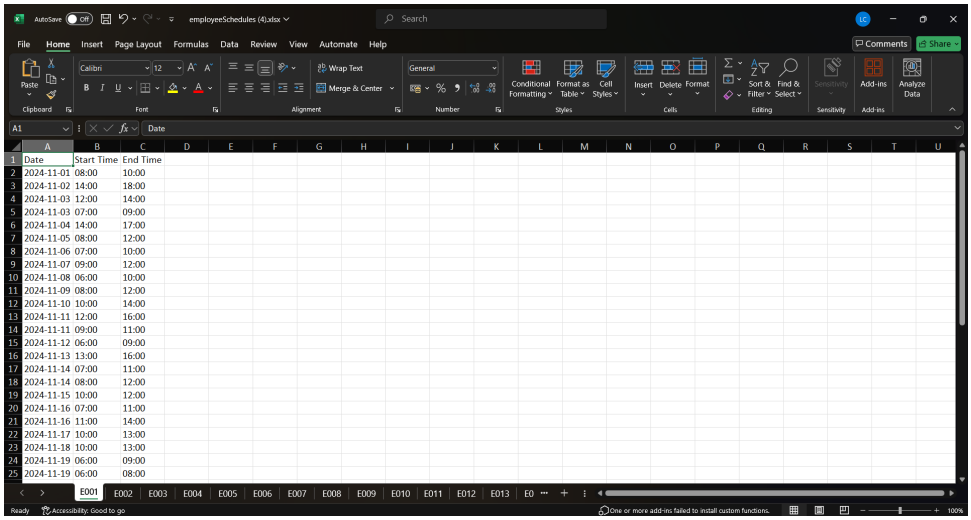


Figure 6.33: Employee Schedules Exported Excel

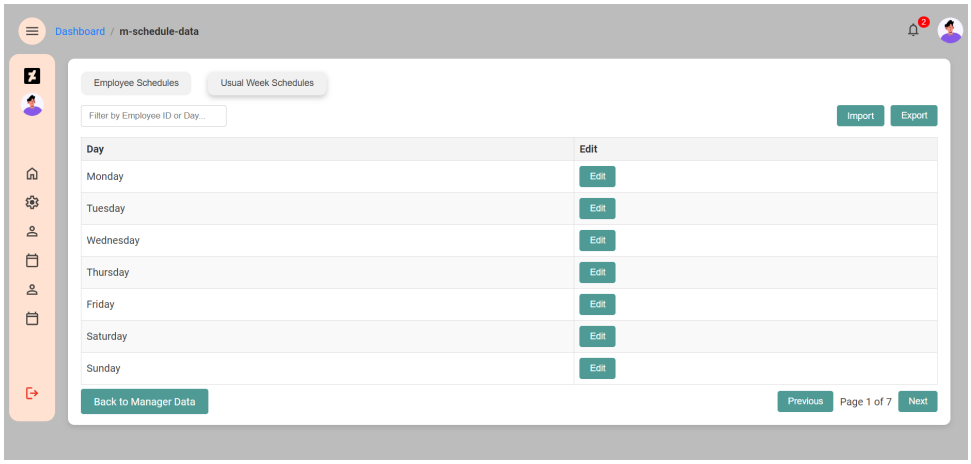


Figure 6.34: Manager Data on Usual Schedules

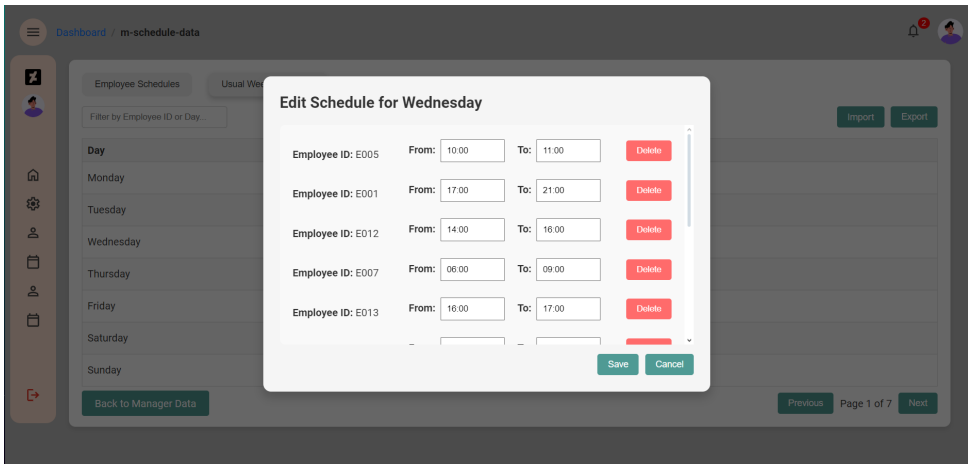


Figure 6.35: Manager Edit Usual Schedule

- **Profile Viewing:** Enables managers to view personal details, team overview,

weekly overview and edit profile.

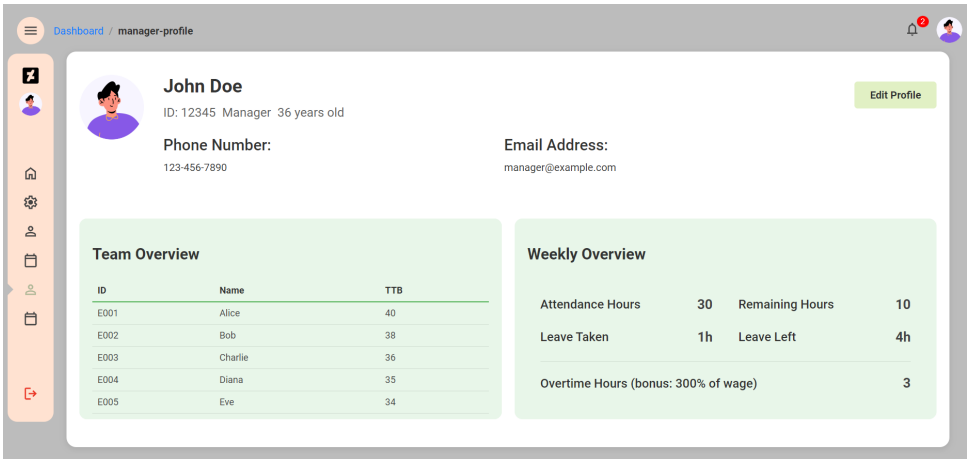


Figure 6.36: Manager Profile Page

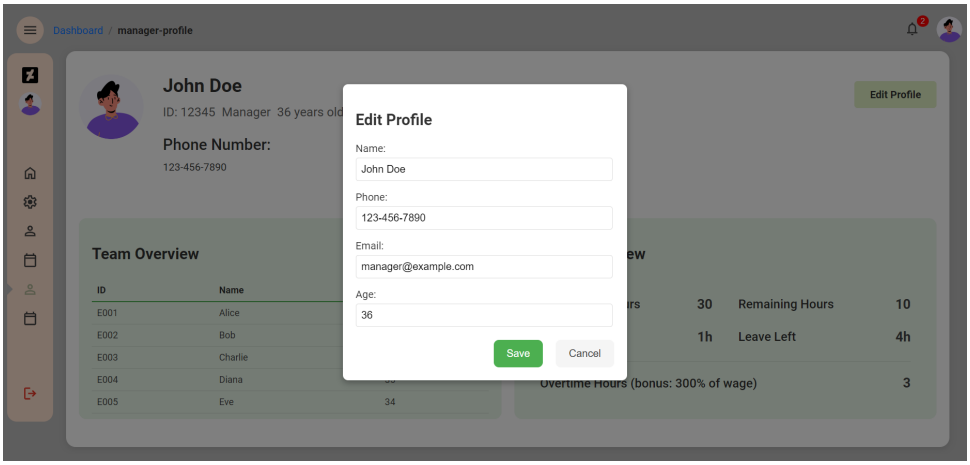


Figure 6.37: Manager Edit Profile

6.4.2 Stakeholder Feedback

- Positive feedback on the ability for employees to choose between daily, weekly, or monthly views of their schedules.
- Requests for notification systems when manager approve or deny leave requests, assign new shifts or approve shift swap.
- Appreciate for the feature that allows data export and import

6.4.3 Challenges and Fixes

- **User Interface Issues:** Initial interfaces were cluttered, making navigation difficult.
  - *Fix:* Redesigned interface with usability testing to improve categorization and filtering.

- **Data Integration:** Formatting inconsistencies during schedule imports caused errors.
  - *Fix:* Implemented validation to ensure uniform formatting and notify users of errors.

6.4.4 Outcome

- Core functionalities for employees and managers have been completed and tested.
- Stakeholder feedback has shaped priorities for the next phase

Chapter 7

Challenges & Problems

One challenge that our team encountered was that we weren't familiar with ReactJs, the framework we are using for frontend development. To prevent inefficiencies in future development, we deliberately had lesser workload in the first week of the first sprint to allow the development team members to get used to the programming language.

Another potential risk that we realized is the risk of in-fighting where a member has a trait or common action that might trigger another member. A study by Hajcak et al. (2005) shown that the ERN (brain response to losses) is larger when the loss is unexpected. This means that if we know what another member might do that trigger us, we can know about it beforehand to reduce the ERN, therefore not having as much emotional response to the issue. Before the sprint began, the team had a meeting where each member shared a behaviour that they think is probably a "red flag" to others.

One mistake that we realized most teams might make is the tendency to delay the documentation and interim report all the way till the end. We believe this will lead to rushed work which causes a decrease in quality of the report. To avoid this issue, we split our team into two sub-groups, development team (4 people) and design team (2 people). The design team will then focus on the drawn designs as well as documentation and interim report. This then allows the development of the web application to be in concurrent with the development of the interim report.

However, another issue arose when we split the teams. The workload for the design team started to pile up which caused ineffective work. To solve this, we stop the drawn designs of the pages such that the design team will be able to focus on the report and documentation side. And since the development team has already made several pages at that point, they were fine without the designs as the theme and designs are pretty similar.

Chapter 8

Time Plan

This section aims to outline the completed tasks, ongoing tasks as well as upcoming milestones. This section also addresses any deviations from our original plan.

8.1 Original Plan

The original project plan followed the Waterfall software development methodology, which emphasizes sequential development Royce (1970). Table 8.1, shows the original timeline divided into ten phases, each with the duration of the phase as well as what should be done in those phases.

Phase	Duration	Key Activities
Planning & Analysis	Oct 13 - Nov 9, 2024	Scope definition, requirement gathering
Design	Nov 10 - Nov 30, 2024	System architecture, UI design, technology selection
Initial Implementation	Dec 1 - Dec 21, 2024	Core functionality development
Continued Implementation	Dec 22, 2024 - Jan 11, 2025	Feature development, code reviews
Testing	Jan 12 - Feb 1, 2025	Comprehensive testing phase
Refinement	Feb 2 - Feb 22, 2025	Issue resolution, performance optimization
Final Implementation	Feb 23 - Mar 15, 2025	Final changes, deployment preparation
Deployment Preparation	Mar 16 - Mar 29, 2025	Final testing, environment setup
Deployment & Submission	Mar 30 - Apr 12, 2025	Software deployment, final reports
Project Showcase	Apr 13 - Apr 19, 2025	Open Day, presentations

Table 8.1: Original Time Plan

While the waterfall methodology was chosen at first because of its simplicity and structured approach to project development, the Waterfall methodology makes it challenging to adapt to changes, which causes delays when addressing issues in later phases Royce (1970).

8.2 General Workflow

After researching different methodologies, the methodology of Scrumban was decided to be used. Scrumban combines Scrum and Kanban such that it allows the team to adapt to changing requirements and focusing on developing high priority tasks Alqudah & Razali (2018).

Figure 8.1 below, shows a general workflow time plan that was made after the methodology Scrumban decision.

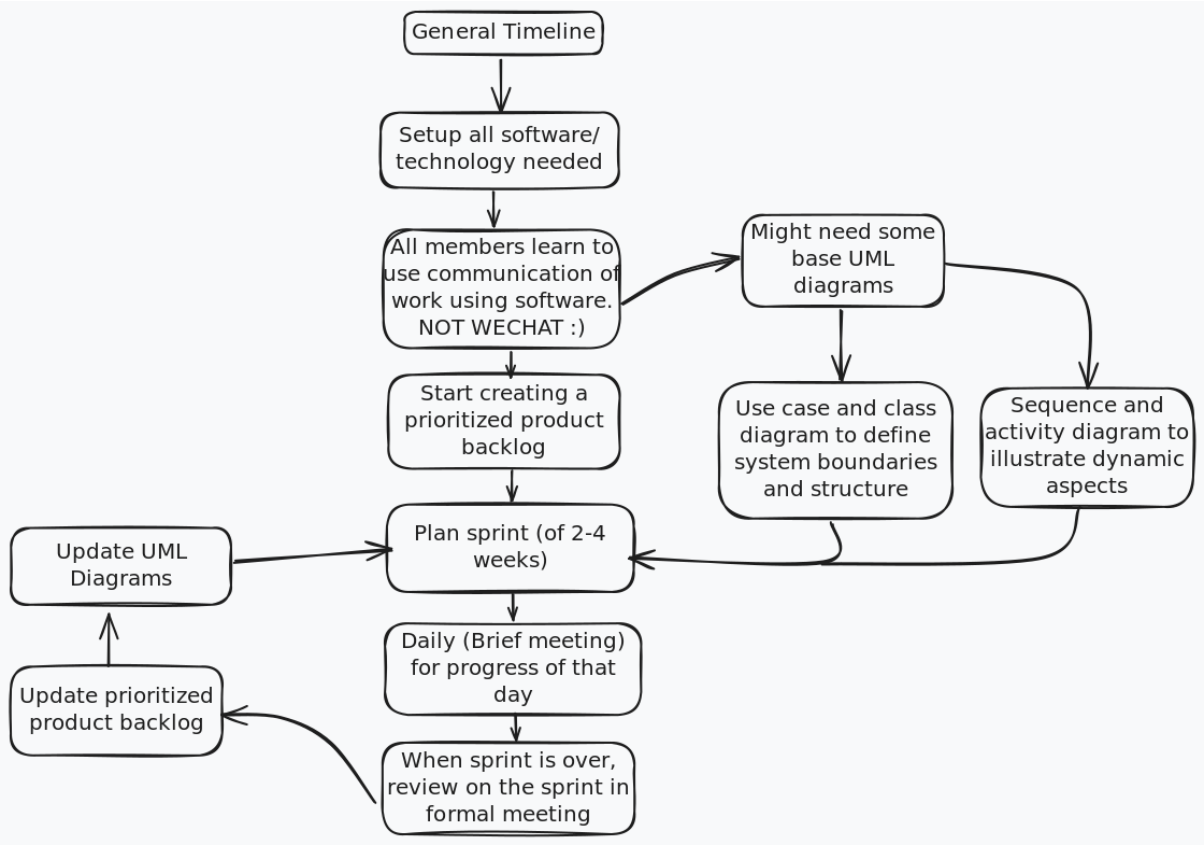


Figure 8.1: General Workflow

The main workflow includes planning sprints, having daily meetings (stand ups), sprint reviews, and continuously updating product backlog and UML diagrams.

8.3 Updated Time Plan

Figure 8.2, shows a provisional Gantt Chart with all the details collapsed that will be updated as needed during the project. For the expanded Gantt chart, see Appendix A.

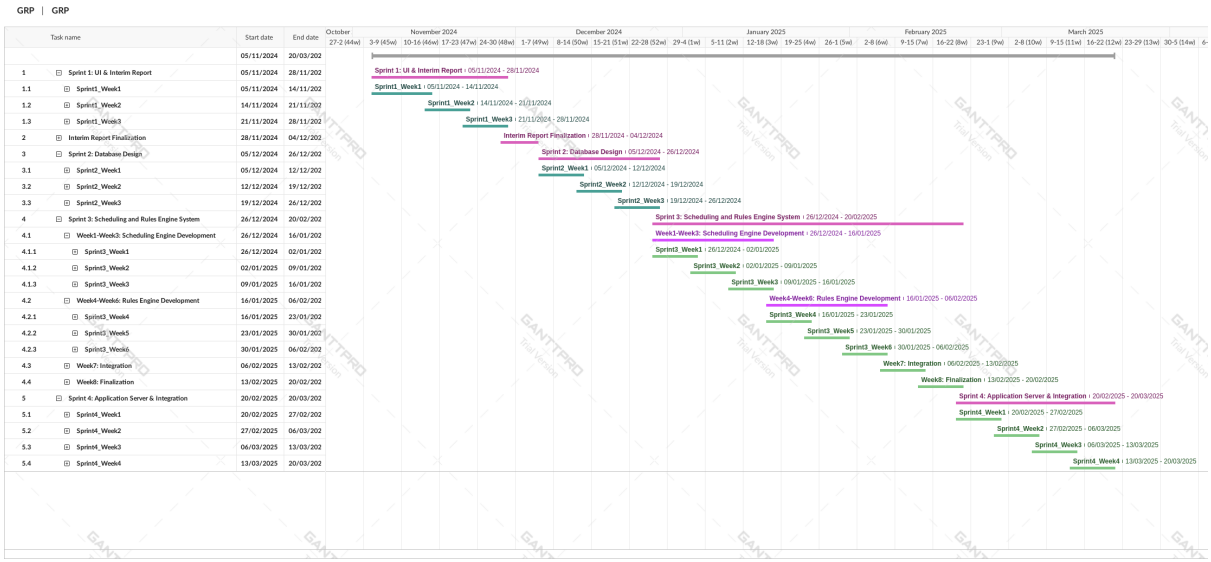


Figure 8.2: Collapsed Gantt Chart

8.3.1 Current Progress

The first sprint from 5th November 2024 to 28th November 2024 ran successfully without any time extension needed. The first sprint was planned to complete the UI of the Web Application.

8.3.2 Upcoming Phases

The second sprint from 5th December 2024 to 26th December 2024 will focus on database design which includes setting up the database as well as designing the database schema. The third sprint from 26th December 2024 to 20th February 2025 will focus on developing the scheduling and rules system. The two systems are being developed simultaneously because of their close interdependence.

The final sprint from 20th February 2025 to 20th March 2025 will focus on building the main application server as well as its integration with the other systems like database, scheduling system, rules system, and the user interfaces. The final weeks of each sprint will be dedicated to integration testing and bug fixing, whilst TDD will be done where unit tests are taken place throughout the development process.

Bibliography

Alqudah, M. & Razali, R. (2018), 'An empirical study of scrumban formation based on the selection of scrum and kanban practices', *Int. J. Adv. Sci. Eng. Inf. Technol* **8**(6), 2315–2322.

Atlassian (2024), 'Jira for project management'.
URL: *https://www.atlassian.com/software/jira*

Blöchliger, I. (2004), 'Modeling staff scheduling problems. a tutorial', *European Journal of Operational Research* **158**(3), 533–542.

Caprara, A., Monaci, M. & Toth, P. (2003), 'Models and algorithms for a staff scheduling problem', *Mathematical programming* **98**, 445–476.

Ernst, A. T., Jiang, H., Krishnamoorthy, M. & Sier, D. (2004), 'Staff scheduling and rostering: A review of applications, methods and models', *European journal of operational research* **153**(1), 3–27.

Excalidraw (n.d.), 'Excalidraw'.
URL: *https://excalidraw.com/*

Hajcak, G., Holroyd, C. B., Moser, J. S. & Simons, R. F. (2005), 'Brain potentials associated with expected and unexpected good and bad outcomes', *Psychophysiology* **42**(2), 161–170.

Inc., L. S. (2024), 'Lucidchart diagramming tool'.
URL: *https://www.lucidchart.com*

js.design (2024), 'js.design (jiShi sheJi)'.
URL: *https://js.design*

Konjaang, J. K. & Xu, L. (2021), 'Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review', *Journal of network and systems management* **29**(2).

Ltd., J. (2024), 'draw.io diagramming and flowcharting tool'.
URL: *https://app.diagrams.net*

National Restaurant Association (2024), 'Restaurant employee demographics data brief – april 2024'.
URL: *https://restaurant.org/getmedia/6f8b55ed-5b3f-40f5-ad04-709ff7ff9f0f/nra-data-brief-restaurant-employee-demographics.pdf*

Nikolic, S. T., Cosic, I., Pecujlija, M. & Miletic, A. (2011), 'The effect of the'golden ratio'on consumer behaviour', *African Journal of Business Management* **5**(20), 8347.

Royce, W. W. (1970), Managing the development of large software systems, *in* 'IEEE WESCON'.

Thesen, A. (1978), 'Heuristic project scheduling', *Project Management Quarterly* **9**(1), 23–29.

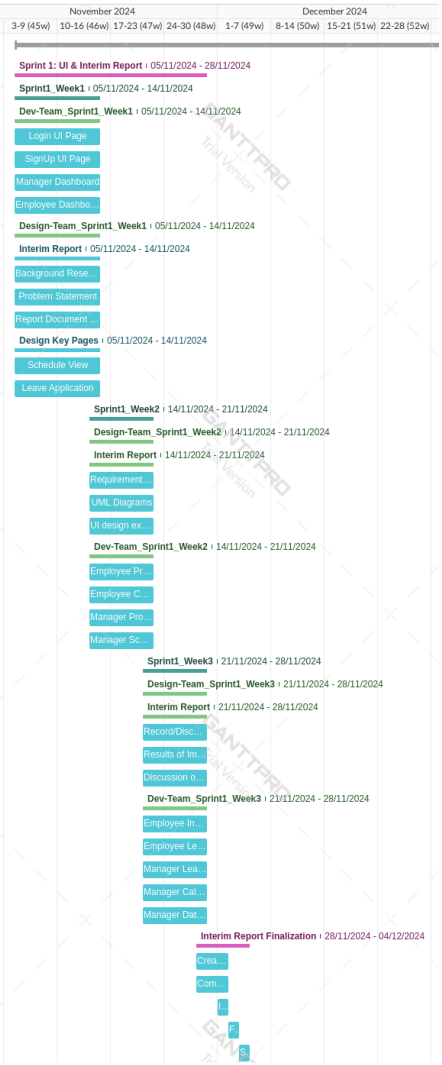
Appendices

Appendix A

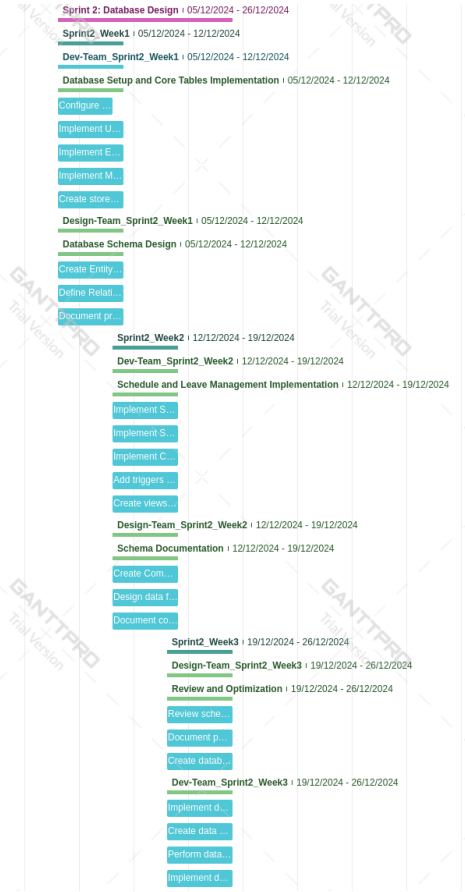
Expanded Gantt Chart

GRP | GRP

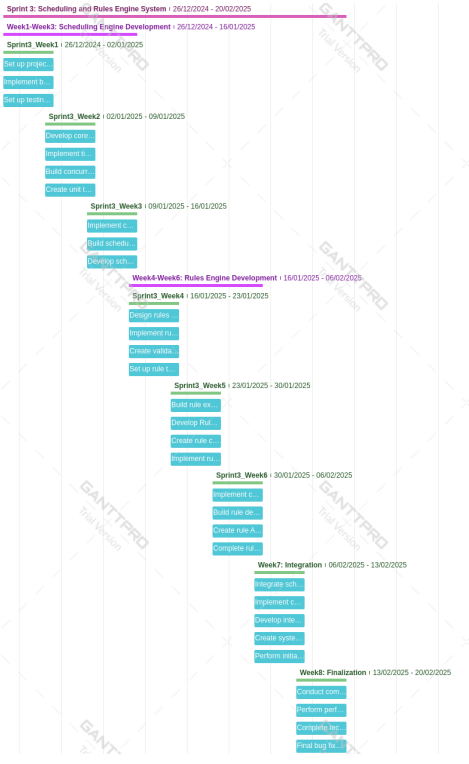
	Task name	Start date	End date
		05/11/2024	**20/03/2025**
1	**Sprint 1: UI & Interim Report**	**05/11/2024**	**28/11/2024**
1.1	**Sprint1_Week1**	**05/11/2024**	**14/11/2024**
1.1.1	**Dev-Team_Sprint1_Week1**	**05/11/2024**	**14/11/2024**
1.1.1.1	Login UI Page	05/11/2024	14/11/2024
1.1.1.2	SignUp UI Page	05/11/2024	14/11/2024
1.1.1.3	Manager Dashboard	05/11/2024	14/11/2024
1.1.1.4	Employee Dashboard	05/11/2024	14/11/2024
1.1.2	**Design-Team_Sprint1_Week1**	**05/11/2024**	**14/11/2024**
1.1.2.1	**Interim Report**	**05/11/2024**	**14/11/2024**
1.1.2.1.1	Background Research (Literature Review)	05/11/2024	14/11/2024
1.1.2.1.2	Problem Statement	05/11/2024	14/11/2024
1.1.2.1.3	Report Document Setup	05/11/2024	14/11/2024
1.1.2.2	**Design Key Pages**	**05/11/2024**	**14/11/2024**
1.1.2.2.1	Schedule View	05/11/2024	14/11/2024
1.1.2.2.2	Leave Application	05/11/2024	14/11/2024
1.2	**Sprint1_Week2**	**14/11/2024**	**21/11/2024**
1.2.1	**Design-Team_Sprint1_Week2**	**14/11/2024**	**21/11/2024**
1.2.1.1	**Interim Report**	**14/11/2024**	**21/11/2024**
1.2.1.1.1	Requirements Specification	14/11/2024	21/11/2024
1.2.1.1.2	UML Diagrams	14/11/2024	21/11/2024
1.2.1.1.3	UI design explanation and justification	14/11/2024	21/11/2024
1.2.2	**Dev-Team_Sprint1_Week2**	**14/11/2024**	**21/11/2024**
1.2.2.1	Employee Profile	14/11/2024	21/11/2024
1.2.2.2	Employee Calendar Page	14/11/2024	21/11/2024
1.2.2.3	Manager Profile Page	14/11/2024	21/11/2024
1.2.2.4	Manager Schedule Page	14/11/2024	21/11/2024
1.3	**Sprint1_Week3**	**21/11/2024**	**28/11/2024**
1.3.1	**Design-Team_Sprint1_Week3**	**21/11/2024**	**28/11/2024**
1.3.1.1	**Interim Report**	**21/11/2024**	**28/11/2024**
1.3.1.1.1	Record/Discussion of Key Implementati...	21/11/2024	28/11/2024
1.3.1.1.2	Results of Implementation steps/prototy...	21/11/2024	28/11/2024
1.3.1.1.3	Discussion of problems encountered	21/11/2024	28/11/2024
1.3.2	**Dev-Team_Sprint1_Week3**	**21/11/2024**	**28/11/2024**
1.3.2.1	Employee Insights Page	21/11/2024	28/11/2024
1.3.2.2	Employee Leave Requests Page	21/11/2024	28/11/2024
1.3.2.3	Manager Leave Requests Page	21/11/2024	28/11/2024
1.3.2.4	Manager Calendar Page	21/11/2024	28/11/2024
1.3.2.5	Manager Data Page	21/11/2024	28/11/2024
2	**Interim Report Finalization**	**28/11/2024**	**04/12/2024**
2.1	Create and improve Time Plan	28/11/2024	02/12/2024
2.2	Complete all parts	28/11/2024	02/12/2024
2.3	Integrate everything together	02/12/2024	02/12/2024
2.4	Finalize	03/12/2024	03/12/2024
2.5	Submission	04/12/2024	04/12/2024



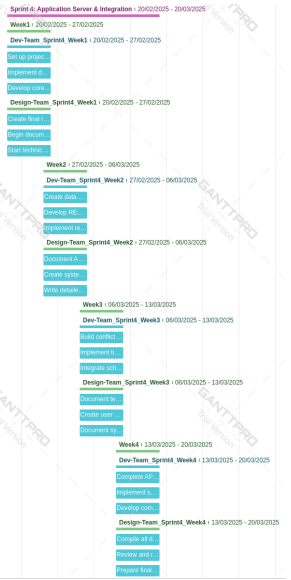
3	**Sprint 2: Database Design**	**05/12/2024**	**26/12/2024**
3.1	**Sprint2_Week1**	**05/12/2024**	**12/12/2024**
3.1.1	**Dev-Team_Sprint2_Week1**	**05/12/2024**	**12/12/2024**
3.1.1.1	**Database Setup and Core Tables Implement...**	**05/12/2024**	**12/12/2024**
3.1.1.1.1	Configure database environment	05/12/2024	12/12/2024
3.1.1.1.2	Implement User Table with fields	05/12/2024	12/12/2024
3.1.1.1.3	Implement Employee Table with fields	05/12/2024	12/12/2024
3.1.1.1.4	Implement Manager Table with fields	05/12/2024	12/12/2024
3.1.1.1.5	Create stored procedures for common e...	05/12/2024	12/12/2024
3.1.2	**Design-Team_Sprint2_Week1**	**05/12/2024**	**12/12/2024**
3.1.2.1	**Database Schema Design**	**05/12/2024**	**12/12/2024**
3.1.2.1.1	Create Entity-Relationship Diagram	05/12/2024	12/12/2024
3.1.2.1.2	Define Relationships between tables	05/12/2024	12/12/2024
3.1.2.1.3	Document primary and foreign key struc...	05/12/2024	12/12/2024
3.2	**Sprint2_Week2**	**12/12/2024**	**19/12/2024**
3.2.1	**Dev-Team_Sprint2_Week2**	**12/12/2024**	**19/12/2024**
3.2.1.1	**Schedule and Leave Management Impleme...**	**12/12/2024**	**19/12/2024**
3.2.1.1.1	Implement Shift Table	12/12/2024	19/12/2024
3.2.1.1.2	Implement Schedule Table	12/12/2024	19/12/2024
3.2.1.1.3	Implement Calendar Table	12/12/2024	19/12/2024
3.2.1.1.4	Add triggers for automatic status update	12/12/2024	19/12/2024
3.2.1.1.5	Create views for easy shift management	12/12/2024	19/12/2024
3.2.2	**Design-Team_Sprint2_Week2**	**12/12/2024**	**19/12/2024**
3.2.2.1	**Schema Documentation**	**12/12/2024**	**19/12/2024**
3.2.2.1.1	Create Comprehensive database schema...	12/12/2024	19/12/2024
3.2.2.1.2	Design data flow diagrams	12/12/2024	19/12/2024
3.2.2.1.3	Document constraints and validation rul...	12/12/2024	19/12/2024
3.3	**Sprint2_Week3**	**19/12/2024**	**26/12/2024**
3.3.1	**Design-Team_Sprint2_Week3**	**19/12/2024**	**26/12/2024**
3.3.1.1	**Review and Optimization**	**19/12/2024**	**26/12/2024**
3.3.1.1.1	Review schema design for optimization	19/12/2024	26/12/2024
3.3.1.1.2	Document performance considerations	19/12/2024	26/12/2024
3.3.1.1.3	Create database maintenance guidelines	19/12/2024	26/12/2024
3.3.2	**Dev-Team_Sprint2_Week3**	**19/12/2024**	**26/12/2024**
3.3.2.1	Implement database security measures and ...	19/12/2024	26/12/2024
3.3.2.2	Create data migration scripts	19/12/2024	26/12/2024
3.3.2.3	Perform database performance testing	19/12/2024	26/12/2024
3.3.2.4	Implement data backup and recovery proce...	19/12/2024	26/12/2024



4	**Sprint 3: Scheduling and Rules Engine System**	**26/12/2024**	**20/02/2025**
4.1	**Week1-Week3: Scheduling Engine Development**	**26/12/2024**	**16/01/2025**
4.1.1	**Sprint3_Week1**	**26/12/2024**	**02/01/2025**
4.1.1.1	Set up project structure and dependencies	26/12/2024	02/01/2025
4.1.1.2	Implement basic time slot management	26/12/2024	02/01/2025
4.1.1.3	Set up testing framework	26/12/2024	02/01/2025
4.1.2	**Sprint3_Week2**	**02/01/2025**	**09/01/2025**
4.1.2.1	Develop core scheduling algorithm	02/01/2025	09/01/2025
4.1.2.2	Implement time zone handling	02/01/2025	09/01/2025
4.1.2.3	Build concurrent operation handling	02/01/2025	09/01/2025
4.1.2.4	Create unit tests for core component	02/01/2025	09/01/2025
4.1.3	**Sprint3_Week3**	**09/01/2025**	**16/01/2025**
4.1.3.1	Implement conflict detection system	09/01/2025	16/01/2025
4.1.3.2	Build scheduling optimization logic	09/01/2025	16/01/2025
4.1.3.3	Develop scheduling API endpoints	09/01/2025	16/01/2025
4.2	**Week4-Week6: Rules Engine Development**	**16/01/2025**	**06/02/2025**
4.2.1	**Sprint3_Week4**	**16/01/2025**	**23/01/2025**
4.2.1.1	Design rules engine architecture	16/01/2025	23/01/2025
4.2.1.2	Implement rule definition system	16/01/2025	23/01/2025
4.2.1.3	Create validation mechanisms	16/01/2025	23/01/2025
4.2.1.4	Set up rule testing framework	16/01/2025	23/01/2025
4.2.2	**Sprint3_Week5**	**23/01/2025**	**30/01/2025**
4.2.2.1	Build rule execution pipeline	23/01/2025	30/01/2025
4.2.2.2	Develop Rule priority system	23/01/2025	30/01/2025
4.2.2.3	Create rule configuration interface	23/01/2025	30/01/2025
4.2.2.4	Implement rule evaluation engine	23/01/2025	30/01/2025
4.2.3	**Sprint3_Week6**	**30/01/2025**	**06/02/2025**
4.2.3.1	Implement complex rule relationships	30/01/2025	06/02/2025
4.2.3.2	Build rule dependency management	30/01/2025	06/02/2025
4.2.3.3	Create rule API endpoints	30/01/2025	06/02/2025
4.2.3.4	Complete rules engine testing suite	30/01/2025	06/02/2025
4.3	**Week7: Integration**	**06/02/2025**	**13/02/2025**
4.3.1	Integrate scheduling and rules engines	06/02/2025	13/02/2025
4.3.2	Implement combined API endpoints	06/02/2025	13/02/2025
4.3.3	Develop integration test suite	06/02/2025	13/02/2025
4.3.4	Create system monitoring tools	06/02/2025	13/02/2025
4.3.5	Perform initial performance testing	06/02/2025	13/02/2025
4.4	**Week8: Finalization**	**13/02/2025**	**20/02/2025**
4.4.1	Conduct comprehensive system testing	13/02/2025	20/02/2025
4.4.2	Perform performance optimization	13/02/2025	20/02/2025
4.4.3	Complete technical documentation	13/02/2025	20/02/2025
4.4.4	Final bug fixes and system hardening	13/02/2025	20/02/2025



5	Sprint 4: Application Server & Integration	**20/02/2025**	**20/03/2025**
5.1	**Week1**	**20/02/2025**	**27/02/2025**
5.1.1	**Dev-Team_Sprint4_Week1**	**20/02/2025**	**27/02/2025**
5.1.1.1	Set up project infrastructure and server env...	20/02/2025	27/02/2025
5.1.1.2	Implement database connections and sche...	20/02/2025	27/02/2025
5.1.1.3	Develop core user management system	20/02/2025	27/02/2025
5.1.2	**Design-Team_Sprint4_Week1**	**20/02/2025**	**27/02/2025**
5.1.2.1	Create final report template and outline	20/02/2025	27/02/2025
5.1.2.2	Begin documenting system architecture	20/02/2025	27/02/2025
5.1.2.3	Start technical documentation of impleme...	20/02/2025	27/02/2025
5.2	**Week2**	**27/02/2025**	**06/03/2025**
5.2.1	**Dev-Team_Sprint4_Week2**	**27/02/2025**	**06/03/2025**
5.2.1.1	Create data access layer for all entities	27/02/2025	06/03/2025
5.2.1.2	Develop RESTful API endpoints for all inter...	27/02/2025	06/03/2025
5.2.1.3	Implement request/response handling	27/02/2025	06/03/2025
5.2.2	**Design-Team_Sprint4_Week2**	**27/02/2025**	**06/03/2025**
5.2.2.1	Document API specifications and integratio...	27/02/2025	06/03/2025
5.2.2.2	Create system flow diagrams	27/02/2025	06/03/2025
5.2.2.3	Write detailed component descriptions	27/02/2025	06/03/2025
5.3	**Week3**	**06/03/2025**	**13/03/2025**
5.3.1	**Dev-Team_Sprint4_Week3**	**06/03/2025**	**13/03/2025**
5.3.1.1	Build conflict detection system	06/03/2025	13/03/2025
5.3.1.2	Implement time slot management	06/03/2025	13/03/2025
5.3.1.3	Integrate scheduling engine with applicatio...	06/03/2025	13/03/2025
5.3.2	**Design-Team_Sprint4_Week3**	**06/03/2025**	**13/03/2025**
5.3.2.1	Document test cases and results	06/03/2025	13/03/2025
5.3.2.2	Create user manuals	06/03/2025	13/03/2025
5.3.2.3	Document system performance metrics	06/03/2025	13/03/2025
5.4	**Week4**	**13/03/2025**	**20/03/2025**
5.4.1	**Dev-Team_Sprint4_Week4**	**13/03/2025**	**20/03/2025**
5.4.1.1	Complete API documentation	13/03/2025	20/03/2025
5.4.1.2	Implement system optimizations	13/03/2025	20/03/2025
5.4.1.3	Develop comprehensive test suite	13/03/2025	20/03/2025
5.4.2	**Design-Team_Sprint4_Week4**	**13/03/2025**	**20/03/2025**
5.4.2.1	Compile all documentation sections	13/03/2025	20/03/2025
5.4.2.2	Review and refine technical content	13/03/2025	20/03/2025
5.4.2.3	Prepare final report submission	13/03/2025	20/03/2025



Appendix B

Meeting Minutes

B.1 Meeting Minutes 1

MEETING MINUTES

Date	17th October 2024	**Time**	4:00 PM
Secretary	Ziyu Jia	**Chairperson**	Lik Wei CHAN
Attendees	Ziyu Jia, Lik Wei Chan, Aijia Yu, Peifeng Liu, Jey Vi Tan, Minghe Xu		
Absentees (Reason)	NONE		
Location	PMB449		

AGENDA DETAILS

Agenda	Discussion
Project Overview	- Main focus is on the dynamic part of the scheduling system. - Dynamic in terms of schedule changing when employees have sudden problems that cause them to not be able to go to work. - Should allow planning horizon from 1 week to 1 month but focus on 1 week for now. - Shift durations should be from 2 to 12 hours. - The core features of the software should be, data import and export, KPI dashboard, real-time schedule update, simulation of unexpected events and UI for restaurant managers.
Scheduling Process	Three main stages of the scheduling process were outlined:

	Stage 1: Demand Modeling Discussion focused on determining worker requirements across different time slots within the planning horizon. **Stage 2: Shift Designing** Creation of shift sets and calculation of required workers to meet modeled demand was discussed. **Stage 3: Shift Assignment** Methods for allocating workers to individual shifts while satisfying demand and constraints were explored. A sample solution and demand data were shown to us by the advisor. He advised us to focus on how and what data we need from the manager to be able to make these schedules. This includes information like max working hours, workers needed for each day, employee information, etc.
Project Timeline	The provisional timeline was established: • Planning & Analysis: Oct 13 - Nov 9, 2024 • Design: Nov 10 - Nov 30, 2024 • Initial Implementation: Dec 1 - Dec 21, 2024 • Continued Implementation: Dec 22, 2024 - Jan 11, 2025 • Testing: Jan 12 - Feb 1, 2025 • Refinement: Feb 2 - Feb 22, 2025 • Final Implementation: Feb 23 - Mar 15, 2025 • Deployment Preparation: Mar 16 - Mar 29, 2025 • Deployment & Submission: Mar 30 - Apr 12, 2025 • Project Showcase: Apr 13 - Apr 19, 2025

	This is subjected to adjustments based on which methodology we want to follow.
Key Deliverables	The following deliverables and deadlines were established: • Equipment requests (from Oct 16, 2024) • Completed Ethics forms (Oct 21, 2024) • Group project site (Oct 31, 2024) • Interim reports (Dec 5, 2024, TBC) • Team final reports and software (Apr 2, 2025, TBC) • Software Demonstration recording (Apr 9, 2025, TBC) • Team Presentation recording (Apr 9, 2025, TBC) • Team promotional digital artifact (Apr 9, 2025, TBC) • Open Day participation (Apr 16, 2025, TBC) • Individual final reports (Apr 22, 2025)

ACTION POINTS

Person in charge	Finish time	Requirement
ALL	19th October 2024	Complete the Jung Typology Personality test and use the graph of team role shown in lecture as reference, choose a team role that aligns with you. Note: Assignment of roles will be in our next informal meeting, 21st October. The guide given from lecture will be used as

		reference for assigning roles. These roles aren't finalized.
Minghe XU & Aijia YU	21st October 2024	Research on the methodology of Kanban and their suitability for our project.
Jey Vi TAN & Lik Wei CHAN	21st October 2024	Research on the methodology of Scrum and their suitability for our project.
Ziyu JIA & Peifeng LIU	21st October 2024	Research on the methodology of Waterfall and their suitability for our project. Note: For all the research done, during our informal meeting on 21st October we will have a group discussion. We will then make a collective decision on the methodology to adopt.

Next Meeting

Date	24th October 2024	**Time**	4:00 PM
Secretary	Minghe XU	**Chairperson**	Lik Wei CHAN
Location	PMB449		

B.2 Meeting Minutes 2

MEETING MINUTES

Date	23rd October 2024	**Time**	20:00 – 20:40
Secretary	Aijia Yu	**Chairperson**	Minghe Xu
Attendees	Ziyu Jia, Lik Wei Chan, Aijia Yu, Peifeng Liu, Jey Vi Tan, Minghe Xu		
Absent person (Reason)	N/A		
Location	Online (Teams)		

AGENDA DETAILS

Agenda	Discussion
Review this week's informal meeting	• Initial role of members • Software development method: Scrum + Kanban
Review the software development method and timeline	Discussed how we are going to develop software by using Scrum and Kanban.
Review tech stack and tools	Discussed the potential software and tools we are going to use.
Team conflicts	We shared potential sources of conflict during the meeting, such as things we do that might annoy other members and things others do that might annoy us.

Next tasks	1. Team members must get familiar with software and tools that we are going to use during the project, such as GitLab, python, etc. (Due 29th October 2024) 2. Design system architecture. (Due 27th October 2024) 3. Design UML diagram. (Start on 28th October 2024)
Next informal meeting	28th October 2024

ACTION POINTS

Person in charge	Finish time	Requirement
Lik Wei Chan	20:30	Sharing team conflicts.
Minghe Xu, Lik Wei Chan	20:40	1. All team members must get familiar with software and tools that we are going to use during the project ASAP. (Due 29th October 2024) 2. Designing system architecture. (Due 27th October 2024) 3. Start to design UML diagram after next informal meeting. (Start on 28th October 2024)

Next Meeting

Date	30th October 2024	**Time**	16:00 – 17:00
Secretary	Peifeng Liu	**Chairperson**	Lik Wei Chan

Location	PMB449

B.3 Meeting Minutes 3

MEETING MINUTES

Date	31st October 2024	**Time**	4:00 PM
Secretary	Peifeng LIU	**Chairperson**	Lik Wei CHAN
Attendees	Ziyu JIA, Lik Wei CHAN, Aijia YU, Peifeng LIU, Jey Vi TAN, Minghe XU		
Absentees (Reason)	NONE		
Location	PMB449		

AGENDA DETAILS

Agenda	Discussion
Confirmed Software	- Jira - GitLab - X2go (Website Update)
System Architecture	Our architecture comprises the following key components: • User Interface • Manager Interface • Application Server • Database • Scheduling Engine

	• Rules Engine • Notification System • Analytics Engine **AI/ML Integration (Potentially)** • Demand Forecasting • Staff Performance Prediction • Schedule Optimization
Functional Requirements	User Management: • User registration and authentication • Role-based access control • User profile management Schedule Creation: • Automated schedule generation • Manual schedule adjustments • Conflict detection and resolution Staff Management: • Staff information management • Leave and time-off requests • Shift swapping and trading Reporting and Analytics: • Generate utilization reports • KPI dashboard Notifications:

	• Automated notifications • Multi-channel support Data Import/Export: • Automated data collection • Automated data export
Implementation Phases	We decided to separate our work into several implementation phases **Phase 1: Core Functionality** • User management and authentication • Basic schedule creation and management • Staff information management **Phase 2: Advanced Features** • Automated scheduling algorithm • Conflict resolution • Reporting and analytics • Import/Export capabilities **Phase 3: AI/ML Integration** • AI-powered demand forecasting • ML-optimized schedule generation • Predictive analytics for staff performance
Implementation Plan	**Team Structure** Our team is divided into two subgroups:

	• Designing Subgroup: Aijia & Minghe • Development Subgroup: LikWei, Peifeng, JeyVi & Ziyu **Development Approach** TDD (Test-Driven Development) should be used.
Project Website	The website is live. To make changes: • Log into x2go with cslinux account • Edit files in the public_html folder • Update corresponding files in GitLab

ACTION POINTS

Person in charge	Finish time	Requirement
Aijia & Ziyu	2nd November 2024	Create Use Case diagram. Visualize main functionalities and actors
Peifeng & Jey Vi	2nd November 2024	Create Squence diagram. Show interactions between key components
Lik Wei & Minghe	2nd November 2024	Create Class diagram. Define core entities and their relationships.
ALL	5th November 2024	Create a comprehensive product backlog starting Saturday. Include tasks and

		features. Open for team modifications.
ALL	TBC	Initiate first sprint planning (2-4 weeks) after product backlog completion. Seek advisor's input in the next formal meeting.

Next Meeting

Date	7th November 2024	**Time**	4:00 PM
Secretary	Jey Vi TAN	**Chairperson**	Lik Wei CHAN
Location	PMB449		

B.4 Meeting Minutes 4

MEETING MINUTES

Date	7th November 2024	**Time**	4:00 PM
Secretary	Jey Vi TAN	**Chairperson**	Lik Wei CHAN
Attendees	Ziyu JIA, Lik Wei CHAN, Aijia YU, Peifeng LIU, Jey Vi TAN, Minghe XU		
Absentees (Reason)	NONE		
Location	PMB449		

AGENDA DETAILS

Agenda	Discussion
Class Diagram	Advisor said looks good and no major error can be seen.
Use Case Diagram	Discussion of potentially using another actor “Chief Executive” to manage the managers who each manages several employees.
Sequence Diagram	Some adjustments are needed. There should be a communication between scheduling engine and the rules engine. Should also expand more on the sequence diagrams, currently too simplistic.
Product Backlog	Priority high:

	- User interfaces (Employee Interface, Manager Interface and Chief Executive Interface, make the UI responsive) - Application Server (User Management System, Schedule Management System, Leave Request System) - Database - Scheduling Engine
Sprint 1 Plan	Sprint Time: 5th November 2024 – 28th November 2024 Development Team: First week: - Set up the project environment, create reusable UI components, and develop key Employee pages. Second week: - Finish remaining Employee-specific pages, expand reusable components, begin building Manager pages, and add the Import/Export Excel page. Third week: - Implement Chief Executive pages, finalize any remaining components, and conduct comprehensive testing and quality assurance. Design Team: First week: - Begin with setting up the report structure, defining the problem, and conducting background research. Start initial wireframes for critical pages. Second week: - Document system requirements and create initial high-fidelity prototypes for critical pages. Present these prototypes to collect feedback.

	- Refine and expand prototypes based on feedback, document the initial system design, and provide detailed explanations and justifications for UI choices. Third Week: - Complete remaining report sections, finalize prototypes, and compile the report with all required content.
Initial UI design	The overall theme of our application should be minimalistic with light colors.

ACTION POINTS

Person in charge	Finish time	Requirement
Design Team	13th November 2024	Start working on the interim report. This week should start with the problem statement and do literature review (Can use this as guide: https://www.scribbr.com/methodology/literature-review/)
Development Team	13th November 2024	Start making several reusable components as well as completion of several pages including: Sign up page, Log in page, Employee home page, Manager home page and Chief Executive home page.

Next Meeting

Date	14th November 2024	**Time**	4:00 PM
Secretary	Ziyu JIA	**Chairperson**	Lik Wei CHAN
Location	PMB449		

B.5 Meeting Minutes 5

MEETING MINUTES

Date	14th November 2024	**Time**	16:00 – 17:00
Secretary	Aijia Yu	**Chairperson**	Lik Wei Chan
Attendees	Ziyu Jia, Lik Wei Chan, Aijia Yu, Peifeng Liu, Jey Vi Tan, Minghe Xu		
Absent person (Reason)			
Location	PMB449		

AGENDA DETAILS

Agenda	Discussion
Reviewed the progress of the project.	Shared the work we've done during the last week, including prototypes and UI implementation.
Discussed areas of current progress that need improvement.	1. The Manager's calendar could be changed to weekly viewed. 2. The calendar should be designed in a more user-friendly way. 3. An employee should not be able to view others' shifts from their own calendar. 4. When an employee wants to sway his/her shift, he/she should only submit a request. 5. The content of 'Problem Statement' does not meet the requirement.

Assigned work for the next week.	1. **Development Team:** Create Employee Calendar Page, Employee Profile Page, Manager Profile Page, Manager Calendar Page. Continue working on other pages if finished early. 2. **Design Team:** write description of the problem, background information & research (literature review) and requirements specification into formal format. Look through UML Designs and see if any changes should be made and shown and written out in the report.

ACTION POINTS

Person in charge	Finish time	Requirement
All	By next formal meeting	Everyone should finish the work assigned to themselves.

Next Meeting

Date	21st November 2024	**Time**	16:00 – 17:00
Secretary	[NAME]	**Chairperson**	Lik Wei Chan
Location	PMB449		

B.6 Meeting Minutes 6

MEETING MINUTES

Date	21st November 2024	**Time**	16:00 – 17:00
Secretary	Peifeng LIU	**Chairperson**	Lik Wei CHAN
Attendees	Ziyu JIA, Lik Wei CHAN, Aijia YU, Peifeng LIU, Minghe XU		
Absentees (Reason)	Jey Vi TAN (Feeling unwell, thus unable to come)		
Location	PMB449		

AGENDA DETAILS

Agenda	Discussion
Development Team	Employee Profile Page: - For now, since there are not much info can be put at the weekly overview part, just leave it for now. But when there is extra information, we can change the data. - If possible change it such that the shift times are shift times of several different days as preferred shift times can differ every single day. Employee Calendar: - All good, no comment. Maybe for improvement, onclick for the calendar shifts, show shifts info on side. Manager Profile:

	- For weekly overview, the data can include data of weekly schedule cost. This is the cost of each employee hourly wage multiply by their shift times added all together. Thus for the employee profile, it can include their salary or hourly wage Manager Schedule: - All good, no comment. For improvement, allow drag and drop for swap shift (additional non prioritized task).
Design Team	- Employee shouldn't be allowed to apply shift swap as this may cause a lot of trouble.

ACTION POINTS

Person in charge	Finish time	Requirement
Design Team	27th November 2024	Write up on UML Designs, Presentation and justification of UI design writeup, Record/Discussion of Key Implementation Decisions (OS, Programming Language, Hardware, Software)
Development Team	27th November 2024	Create Manager Calendar Page, Employee Leave Requests Page, Employee Data Page, Manager Leave Requests Page, Manager Data Page

Next Meeting

Date	28th November 2024	**Time**	4:00 PM
Secretary	Jey Vi TAN	**Chairperson**	Lik Wei CHAN
Location	PMB449		

B.7 Meeting Minutes 7

MEETING MINUTES

Date	28th November 2024	**Time**	16:00 – 17:00
Secretary	Jey Vi TAN	**Chairperson**	Lik Wei CHAN
Attendees	Ziyu JIA, Lik Wei CHAN, Jey Vi TAN, Aijia YU, Peifeng LIU, Minghe XU		
Absentees (Reason)	NONE		
Location	PMB449		

AGENDA DETAILS

Agenda	Discussion
Development Team	- Change data type of employee preference for shift time to a list of times such that there are times for each day of the week.
Design Team	- For Manager, they should be able to view their calendar as well. - Try to explain the details between calendar and schedule earlier in the report to not confuse the readers. - It's fine to use several different programming languages for backend. For example since we are using React JS for front end, it is better to use Javascript for backend as well as Python to do logic calculations as it is easier that way. For databases, any language can be used as they are pretty much the

	same. Thus using SQLite might be better since we've learnt it before.

ACTION POINTS

Person in charge	Finish time	Requirement
ALL	3rd December 2024	Discussion of problems encountered (technical, personal, management, ...), Time Plan for the project, look for missing parts and try to perfect the report

Next Meeting

Date	5th December 2024	**Time**	4:00 PM
Secretary	Ziyu JIA	**Chairperson**	Lik Wei CHAN
Location	PMB449		