

Software Engineering Group Project

COMP2002 / G52GRP: Group Report

Project information:

Project title: Support Engineers Rostering

Project sponsor: Capital One

Academic supervisor: Dario Landa Silva

Industry Supervisor(s): Madhu Prashanth

Team information:

Team number: 35

Kejia Wu, 20030616, scykw1

Nicole Millinship, 14329892, psynm6

Gurjyot Kaur, psygk2

Liam Orrill, psylo, 14324120

Tajin Tasnuva, psytt1, 14331739

Xuanhao Li, scyxl3

Documentation (links):

Code repository: https://projects.cs.nott.ac.uk/COMP2002/2019-2020/team35_project/tree/master/Prototype

Document repository: https://projects.cs.nott.ac.uk/COMP2002/2019-2020/team35_project/tree/master/Meeting_Minutes

Further Online Documentation (if applicable):
https://teams.microsoft.com/_#/school/files/General?threadId=19%3Aa829ae09041e4cbd8d28936fe8ed3d8d%40thread.skype&ctx=channel&context=General&rootfolder=%252Fsites%252FTeam35%252FShared%2520Documents%252FGeneral

Table of Contents

Part 1: Main Report	2
Changes Section.....	错误!未定义书签。
Introduction.....	2
The Brief	2
The Result	2
Requirements Gathering	2
Initial Requirements	2
Second Requirements.....	3
Third Requirements	4
User Stories.....	5
Use-Case Diagram.....	6
Sequence Diagram.....	6
Project Management.....	7
Methodology	7
Documentation.....	8
Project Management Tools	8
Team Skills	9
Project Plan.....	9
Technical Approach	10
Design	10
Development Tools.....	13
Database Management	13
Data management and manipulation.....	14
Testing	14
Reflection.....	15
Project Management Challenges.....	15
Technical Challenges.....	15
Contingency Measures	16
Future Plans.....	16
Conclusion	16
Part 2: Software Manual.....	18
Part 3: User Manual.....	34

Part 1: Main Report

INTRODUCTION

The Brief

In this project, our team is developing an online web application for Capital One. Our sponsor, Madhu Prashanth, is the administrator of a small team, and she wants to save herself and other administrators from the redundant work of manually checking a team member's criteria to check they can fit them into different positions in a timetable. This timetable tracks on which weeks employees have been allocated a support role for the software they have developed. The system should announce sudden changes regarding an employee's availability to the administrator, and it should also be able to generate a report tracking the frequency that an engineer is on support.

Our team initially considered developing an auto-rostering function, which could automatically select a qualified employee and put them into the timetable. We also considered creating an auto-email system, which emails the administrator the timetable for the next week.

When we first pitched for this project, we analysed currently available products like [Deputy](#) and [findmyshift](#). We noticed that most current commercial systems use a drag-and-drop approach to make things as easy to use as possible. However, they do not meet the specific requirements of this project; for example, the sponsor has three different job roles that need an employee to be assigned to them each week, but most other products just use one. Furthermore, we later discovered that the sponsor wants the system to operate locally on Capital One's intranet, and wants the system to display the timetable without having to log in, which current software wouldn't allow.

The Result

Our team has created a web application which employees at Capital One can use to view the timetable of who is on-support each week. To edit the timetable, an administrator should log in using a username and password stored in our MySQL database, and they can then choose an employee to add to the timetable, as long as the employee is active and is not on holiday or already on the timetable. Administrator could also add the employee multiple times by selecting the repeat button, which will add then every 2,3 or more weeks.

Details about employees are also stored in the database, and a user can view and edit their profile, such as changing their profile picture. The administrators can view all the employees and edit other employee's profiles or create new ones. They can also click a button to generate an employee report that details what days the employee was on-support for a given month. When an employee tries to take a leave from a scheduled on-duty week, this web application should automatically send an email to notify the administrators.

REQUIREMENTS GATHERING

Initial Requirements

When our team viewed the "Call for Expressions of Interest" for this project, we proposed these requirements:

Functional

1. (Basic) An employee or administrator will log in to view the timetable
2. (Basic) An administrator can view previous assignments of any employee
3. (Basic) An administrator can assign or modify any employee's future working schedule
4. (Important) An administrator can generate a report detailing how many times an employee has been on support, so they can ensure that the number of times employees are on duty are not unbalanced
5. (Important) The administrator can attempt to assign an employee to a job for a given week. The system will automatically check whether one employee is available to be on support for that week according to several attributes:
 - a. The employee must not be in their probation period (3 months after joining)
 - b. The employee's performance rating must be at least 'strong', These ratings could only be modified by the administrator each quarter
 - c. The employee must not be on holiday or have just been on holiday for that week
 - d. The employee cannot be doing deployments within that week
 - e. Given a team size of N employees, the same employee cannot be assigned again for $n-2$ weeks since their last assignment
6. (Extensive) When an employee requests a sudden leave, the administrator will automatically receive an email sent from the system, which prompts them to approve or reject the leave. If the leave is approved the administrator should be able to update the schedule stored in this system
7. (Extensive) An auto-rostering system should be able to automatically generate a timetable for the administrator without violating any of the attributes of requirement 5

Non-Functional

8. (Basic) Automated emails should be sent to the relevant administrators less than a minute after an employee uses the system to request leave
9. (Basic) The system should have a clear GUI that is easy to use and looks professional.
10. (Basic) The webpages should function on most desktop browsers – Google Chrome, Mozilla Fire.
11. (Extensive) The webpage can be viewed and navigated on mobile browsers without difficulty

According to our team's discussion, from the initial requirements provided by the "Call for Expression of Interest", we thought our sponsor requires an efficient rostering system with clear GUI and some particular evaluation criteria towards the scheduling. The most important functionalities should be the evaluation of inserted working schedules and user account management. Therefore, we decide to generate an auto-rostering system as an extensive function to save more workload for the administrator. Furthermore, because the first offline meeting is planned to be held on the 7th of November, we decided to do some research in some current commercial rostering systems first. This could significantly benefit our first offline meeting with our sponsor. And part of the group started developing some components which could be used in future development.

Second Requirements

After we met with Madhu Prashanth on 7th November 2019, we discovered some more requirements for this project:

Functional

1. (Basic) Criteria 1 of the previous requirements has been modified:
 - a. Anyone can view the timetable without having to log in to the system. To make changes to the timetable or to view personal employee profile, the clients must log in. This is because

the system will be used on a local network so there's not a privacy concern with displaying the timetable without needing any log in credentials

2. (Basic) Each week, the timetable begins on a Sunday and ends on a Saturday. New assignments to the rota occur on Tuesdays
3. (Basic) The three roles an employee can be allocated – Primary Engineer, Secondary Engineer and Escalation Manager, are colour coded red, blue and yellow respectively on the timetable
4. (Basic) To manually add an employee to the timetable, administrator should click their name then select the date that they should be placed on the timetable
5. (Basic) Employees are members of either 'Customer Nottingham', 'Customer London' or 'Grow'. Each group has its own timetable
6. (Basic) Sudden leave requested by the employee does not need to be shown on the timetable
7. (Extensive) A Search input will allow users to search for specific employees
8. (Important) Criteria 5a of the previous requirements has been modified:
 - a. An employee has a status attribute that can either be active or inactive. The employee is inactive when they are on a probation period, but their status could also be changed for other reasons. We found that there were other reasons that an employee shouldn't be put on the timetable for a while, not just because they're on their probation period, so it was more useful to have an attribute that covered all of these reasons in one.

The following initial requirements were removed:

1. (Extensive) An auto-rostering system should be able to automatically generate a timetable for the administrator without violating any of the attributes of requirement 5. The sponsor didn't see this as a key part of the project, and she preferred that we created a feasible system with a few important features than one that tried to do a lot of things poorly

These changes make the implementation of the system more explicit and more feasible, and our team has changed our design and project plan simultaneously. According to this offline meeting, we have a more specific overview of our sponsor's requirements. Moreover, we found that the most critical functionalities should be the evaluation of inserted working schedules and user account management. According to our discussion, we decide to focus on realizing these most important functionalities first, to produce a rough framework of this system. After that, we could discuss detailed implementation with our sponsors in future meetings. For the mail system and report generator, because they are both extensive functionalities, we decided to develop them in the spring semester.

Third Requirements

In the second semester, the requirements were again refined further as we began to discuss some of the additional features of the system in more detail with the sponsor, and the priority of some requirements has been rearranged:

Functional

1. Criteria 4 of the first requirements has been modified:
 - a. (Important) The report details what days an employee is on the rota for a specific month. It contains the employee's name and ID, and their payroll instructions
2. Criteria 6 of the first requirements, the automated emails, has been reprioritized to 'Important'. We asked the sponsor which of the extensive requirements was more beneficial to her, and she said that she values the automated email system the most because the emails would inform her of changes that may have to be made to the timetable and would improve organisation.
3. Criteria 4 of the second requirements has been modified:

- a. (Basic) An administrator can drag and drop an employee onto the timetable in order to manually assign them to the timetable. This means that you can either click on an employee and then choose a date or drag an employee to a week to add them to the timetable.

The following second requirements were removed:

4. (Extensive) A Search input will allow users to search for specific employees

Through our meetings with our primary stakeholder Madhu, we inferred that the most crucial aspect of the project for our sponsor is to save time Madhu and her other team members waste using the current system. Moreover, since the main part of the system is already functional, she suggested we start developing the extensive functionalities. We have reprioritized the requirements in order to focus on the features that will help Madhu and her team save time the most, for example, the report generator. And part of our group started investigating the mail system.

Achieved Requirements

Given the requirements from our sponsor throughout the creation of our project and the system currently, we can highlight what we have currently achieved and what is yet to be added or those retracted from the sponsor.

Throughout this project, we have achieved a wide range of the requirements that were requested from the sponsor, such as adding a timetable set up in an easily identifiable fashion and gives key details that are needed for the employees. This shows a monthly calendar, which is allocated to each day and can be traversed to see any months rostered schedule that all fall under the requirements set from the sponsor. Another requirement that closely links to the timetable would be adding a new entry to it for particular employees. When doing this, we split it up into three sections for the three roles that the team has and checked that set criteria were met before assigning them to the timetable.

Alongside this, a feature that became of utmost importance to the sponsor was the ability for employees to request leave if needed, which would trigger an email to be sent, notifying the admins of this change so they can sort it out. We were able to implement a working version of this to the application and works as expected.

We were also given the task of allowing admins to generate reports for different months for a specific employee giving information on their working hours for the month and how much they should be paid for hours worked. This functionality is also achieved in the spring semester.

User Stories

We created some user stories from the requirements in order to understand the system from the user's perspectives. We created a story for a regular employee, and an administrator:

Initial Requirements:

As an administrator, I want to be able to assign employees to the timetable every week to the three different roles. I want to be able to log into the system and edit employee's details, such as their performance rating. I also want the system to generate a report detailing how often users are rostered. I want to automatically receive emails when an employee requests leave, so that I can accept/reject their request. I also would like the system to automatically create the timetable for me if I press a button.

As an employee, I want to be able to update my personal details once I've logged into the system. After I've logged in, I should be able to view the timetable and request time off from the administrators, which would mean I wouldn't be assigned to the timetable.

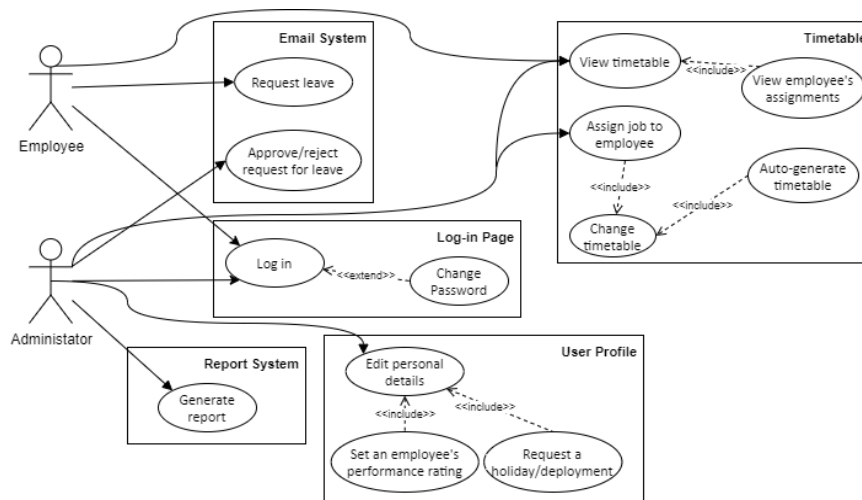
Third Requirements:

As an administrator, when viewing a month of the timetable I want to generate a report for that month that details which employees were on the rota each day. I want to be able to add an employee to the timetable repeatedly, for instance every 5 weeks, without having to add each assignment individually.

As an employee, I want to be able to view the timetable without having to log in. Once I've logged in, I should be able to change my profile picture and contact details.

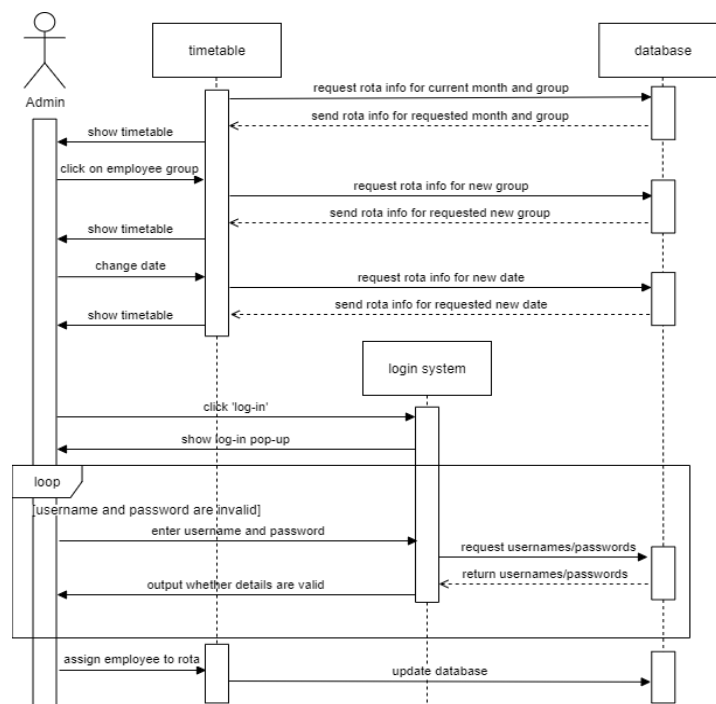
Use-Case Diagram

From the first user stories we made a use-case diagram to get a high-level overview of the system and its requirements.



Sequence Diagram

We also made a sequence diagram from the second requirements to show how the different components work together in order to help us when we began combining components into one system.



Critical Analysis of Tools

The main structure of the web application was constructed using HTML, CSS, JavaScript, PHP, and SQL as all our group members are well versed in these languages as we have completed previous projects using these tools. We considered using frameworks like VUE and Angular, but the time saved using frameworks was miniscule against the time it would take to learn these frameworks in the first place. We asked Madhu her thoughts on what framework to use, and she felt that we should use whatever suits the team the best. Therefore, there are no urgent needs in applying any framework in this project. The school server allows us to test our code quickly, and we all have accounts on PHPMyAdmin to be able to create databases.

Two libraries were utilized in the creation of the mail system, PHPMailer and SheetJS. PHPMailer provides a better alternative to the mail () function as the mail () function is not flexible enough. Moreover, PHPMailer is object oriented as opposed to the mail () function. \$headers strings need to be created while sending emails using the mail() function however this can be tedious as they require a lot of escaping, and also the writing of messy code for escaping characters, formatting and encoding to send attachments and HTML based emails, however, PHPMailer makes this a much simpler process. PHPMailer can also use a non-local mail server (SMTP) with authentication. Further advantages to using this tool for our project include integrated SMTP protocol support and authentication over SSL and TLS and that it has a very active developer community making it secure and up to date.

We had to use the community version for SheetJS, which meant that we are unable to recreate the fundamental requirements of our client. Upon further consultation, our client directly changed their template and told us to focus on the main structure of our system and not bother about the aesthetics.

PROJECT MANAGEMENT

Methodology

Throughout the project, we have followed the Agile development practice of using Sprints. Every sprint cycle is typically one weeklong. These short cycles ensure that work is produced quickly and that everyone is continuously working on the project, rather than people being given a month and waiting until the last minute to start working. Every day the team leader checks in on the other team member's progress in case there are any challenges. Our academic supervisor could check the meeting minutes folder on Gitlab to see what work a team member has been assigned.

We originally intended to meet with our project sponsor at the beginning of every month, where we present our work so far. By meeting frequently, we ensure that we are on the right track and are meeting the requirements of our sponsor. However, for various reasons, we have not met with Madhu every month, but we have held a face-to-face meeting with her on 7th November 2019 and 13th February 2020. After that, we also held an online conference with her on 18th March 2020.

In the second semester, our team realized that some tasks could take less than one week's time. We decided to meet more frequently so that team members could move on to new tasks quickly and efficiently. From March, because of the campus' closure, we had to switch to online meetings, which made it harder to interact with each other and work together. We tried to address this problem by increasing the frequency of meetings.

The challenges of this methodology are making sure that the work allocated is reasonable, given the amount of time. This methodology can be especially challenging as different team members work at different speeds or have different commitments during the sprint that means they cannot be expected to produce a high quantity of work. As such, there must be some flexibility regarding the work assigned.

Documentation

The minutes of our formal meetings are stored on our GitLab repository in the 'Meeting_Minutes' folder. This folder is divided into months so that our academic supervisor can find the relevant minutes for each month. Other relevant files are also included in the folder that demonstrates the work we did that week that was not code related. For example, in the folder [25 Oct 2019](#), the meeting minutes and a copy of an email sent by Geert from that week are included.

The minutes were written by Kejia and Nicole and follow a consistent format to make them easy to understand. They begin with what the aim of that conference was, who attended and what work was allocated.

The inline documentation mainly focuses on explaining how we implemented functionalities and clarifying the purpose of different components. This documentation is especially important for back-end components because during the first semester other team members did not have access to the database that other team members were using and could not see the component working, so we relied on comments and live demos during our meetings to understand their part of the system. All comments are formatted in the same way, being placed above the line of code the comment is explaining. For example, here is a snippet from the file [build.js](#):

```
/**
 * Display all the employees from a particular group
 * @param {string} group - The group to display
 */
function buildGroupEmployees(group) {
    var status;
    var html = "";

    for (var index = 0; index < employeeInfo.length; index++) {
        // if employee isn't in this group, move to the next employee
        if (employeeInfo[index]["group_id"] != group.toString()) {
            console.log(group.toString() + " " + employeeInfo[index]["group_id"]);
            if (group.toString() != "Other" || employeeInfo[index]["group_id"] != "") {
                continue;
            }
        }
    }
    // ...
}
```

The code files themselves are also organized in a way that are easy to navigate and follow the code conventions of encapsulation. When a team member is working on a new feature on their own, they use the 'Components' folder and create a new folder for each new feature they work on. The entire system is stored in the 'Prototype' folder, and it is broken up into the different components, e.g. the timetable, the mail system, the employee interface... Each component folder contains subfolders for the PHP and JavaScript files. A JavaScript file has been set up, which builds the entire database, to help clients deploy the entire system automatically.

Project Management Tools

Our team has tried a few different project management tools, but currently, we use Microsoft Teams to store files that we can all edit together. Initially, we used OneDrive to store our files, but Microsoft Teams enabled all of our team members to edit a file together, which made it easier to work together. Moreover, Microsoft Teams offers more features as group chat members could send messages to each other, schedule meetings on the calendar, and add channels to the group.

Our team initially used a WhatsApp group chat for discussion or emailing for more formal communication. We switched to using Microsoft Teams chat as we believed the team would be able to respond quicker using it. We used Zoom to host video conferences with our sponsor Madhu. In the last few weeks, because the university has moved to online learning using Teams, we also held our online meetings on there to make things simpler.

Team Skills

We have all studied web development last year, so we all had some experience with HTML, PHP, CSS, and JavaScript to create websites. It therefore made sense to use them, rather than learning a new programming language like C++.

We tried to assign work based on the skills and experience of our team members, and each of our team members has enhanced or developed different technical skill based on this teamwork experience:

- Because Kejia has created a Hotel Management system before, he has experience making a calendar that displays each month. It therefore made sense to put him in charge of the timetabling view, which shows what employees are assigned to roles each month. He then integrated the web pages together to create a complete prototype with Liam during the winter break.
- Nicole was chosen to keep the meeting minutes and oversee the report writing because of her writing skills and background studying English Literature. She also contributed to the codebase by creating the front end of the employee interface and administrator interface simultaneously (from Sep 2019 to Dec 2019).
- Gurjyot has the most experience in art and design, so she produced the frontend login system, where she focused on the GUI. After each team member's components were combined into a complete prototype, she began creating the stylesheets to ensure that the pages all follow a consistent and clean design.
- Liam is good at managing and maintaining a database and is a reliable coder, so he developed the database to store rostering information of employees. He also took on the task of designing the automated email system.
- Tasnuva has experience working in Student Services, which meant she knew the importance of making a system accessible for users. Therefore, she was focusing on the development of the employee interface and making the system easy to use.
- Since Xuanhao had not used Git before this project, the other team members, including the Git Master Gurjyot, assisted him so that he could push to the project. He can now use Git successfully and made 13 commits to the master branch in the first semester.

Project Plan

Following our EOI, we initially divided the project into three stages:

1. First stage: Framework development
November 2019 to January 2020
2. Second stage: Mail system development
February 2020 to March 2020
3. Third stage: Code refactoring and testing
March 2020 to April 2020

When we began the first stage on the 8th of November, our team decided to divide the entire project into six components to enable each of the group members to focus on one thing. Within this period, merge conflicts rarely happened as everyone had separate files. Everyone's work within each week is easily accessible, as everyone has their own 'components' folder on Gitlab. Our supervisor could also use the 'history' button on a team member's folder to view their commits.

On the 22nd November, in order to work on the project itself and the interim report in parallel, we divided our group into two teams. The three members who were working on the backend components opened a 'dev' branch to merge their backend system with another team member's frontend system, and the other three began writing sections of the interim report. This means that once the report was finished, a full system

prototype had been made. After testing each component of the system, the 'dev' branch on Gitlab was merged with the 'master' branch and deleted manually.

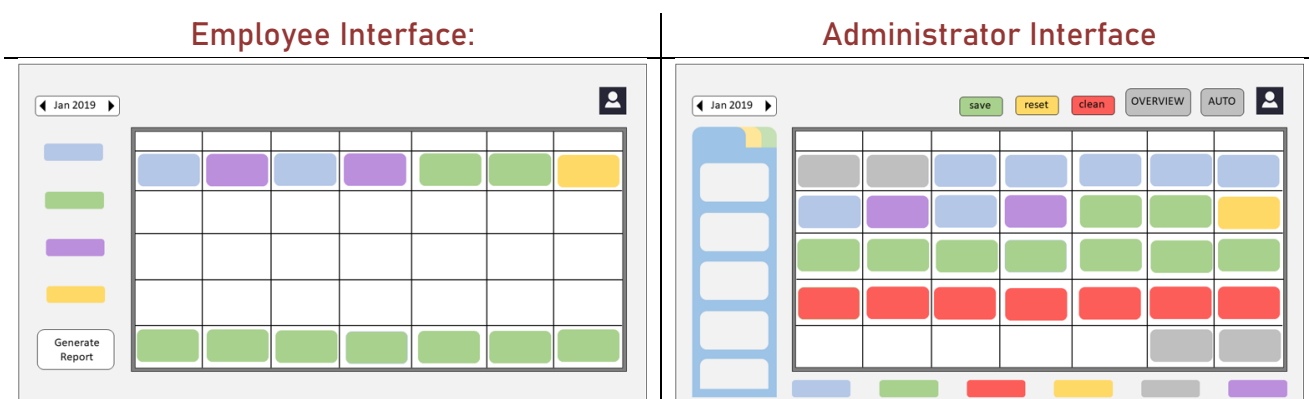
In the second semester, the entire team began to develop more features to the system, and a new dev branch was started. Gurjyot began improving the CSS, Kejia began testing and bug-fixing, Liam worked on the automated email system, Tasnuva investigated adding drag-and-drop functionality, and Nicole began improving the employee profile page. This differs from our original plan of the second stage of the project because we felt it unnecessary to make every team member work on the mail system when there were other features to consider. By the end of February, bug-fixing was finished, so Kejia began working on the report generation.

On the 12th March, we again split the group, so that half (Kejia, Liam and Gurjyot) continued improving the system, and the others started on the final report (Nicole and Tasnuva). In April, Liam continued working on the stylesheet for the system while everyone joined the report writing. This was also changed from the original plan because auto-rostering was no longer a requirement of the project.

TECHNICAL APPROACH

Design

After researching current commercial rostering systems we found online, we generated a rough idea of the system. Below is our team's preliminary design of the two main user interfaces, which we used in our pitch:



The different coloured rectangles represent different events on the timetable, such as an employee rota assignment, absences and holidays, etc. The employee interface allows an employee to view their job role's timetable, and they can drag different events onto the timetable to update it. The administrator interface shows a view of the timetable for a certain role, with the three boxes on the left representing the different types of roles - primary engineer, secondary engineer, or escalation manager. These boxes will be filled with all the employees assigned to that role.

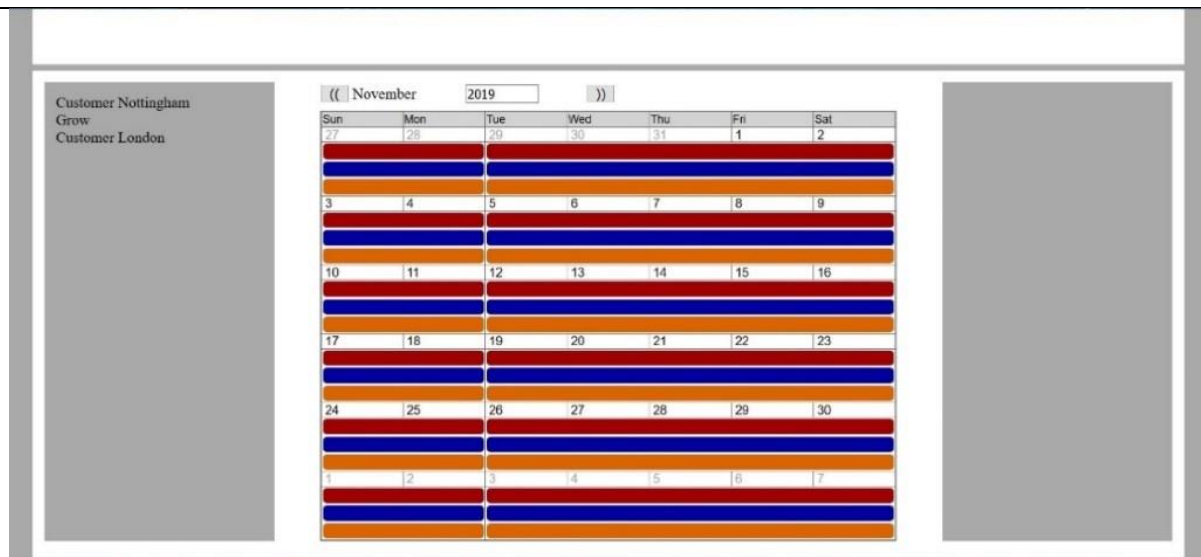
After the first discussion with our sponsor on the 7th of November 2019, our team received information about the current system used by our sponsor and more specific information about what the sponsor wants. We rearranged the project and decided to develop three main systems:

1. Timetable Interface – a calendar that shows a monthly view of the timetable and what employees are assigned to what role. There are multiple different timetables for the different teams at Capital One to use that can be switched between.
2. Log in – before making changes to the timetable, a login prompt will ask for a username and password. Usernames and passwords are stored in a database, as well as whether the user is an administrator or not

- Employee Profile – the name, email, telephone number, etc. of an employee. Employee details are also stored in the database. Employees can change some of their personal details, while administrators can view all employees and change any of their details. They can also create or delete employees.

Our framework prototypes of these three systems in the first semester are seen below:

Timetable Interface:



Log-in:

Admin System

Login

- User name:
- Password:
-

Register

center(Can modify informations)

Username(must):

First Name:

Last Name:

Email:

Age:

Employee Profile:



Employee Name: Betty James

Work ID: 4

Email Address: BettyJames@gmail.com

Telephone Number: 07583520000

Current Job Role: None

Slack ID: BJHSFF

Status: Inactive

Future Holidays:

Start Date	End Date
2020-01-22	2020-01-25



Change Profile Picture here

First Name:

Surname:

Email Address:

Telephone Number:

Add a new holiday:

Start date:

End date:

In the second semester, all components were integrated together, and the interfaces were improved. Below are demonstrations of the different pages in our completed system:

Timetable Interface:

Customer Nottingham
Grow
Customer London


December
2019

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1 Kobe	2 Gurjyot	3 Laim	4 Nicole	5 Xuanhao	6 Tajin	7 Laim
8 Kobe	9 Nicole	10 Gurjyot	11 Xuanhao	12 Tajin	13 Laim	14 Kobe
15 Kobe	16 Nicole	17 Gurjyot	18 Xuanhao	19 Tajin	20 Laim	21 Kobe
22 Kobe	23 Nicole	24 Gurjyot	25 Xuanhao	26 Tajin	27 Laim	28 Kobe
29 Kobe	30 Nicole	31 Gurjyot	1 Xuanhao	2 Tajin	3 Laim	4 Kobe
5 Kobe	6 Nicole	7 Gurjyot	8 Xuanhao	9 Tajin	10 Laim	11 Kobe

Employee Interface:

Time table

Admin interface



Upload new image

Name

Kobe

Status

✓ Active

Working Id

scykw1

Account type

admin

Job role

Primary Engineer

Slack Id

Norris

Group

Customer Nottingham

Email

scykw1@nottingham.ac.uk

Telephone

18030700990

Edit

Holiday

Start date

mm/dd/yyyy

End date

mm/dd/yyyy

submit

Deployment

Start date

mm/dd/yyyy

End date

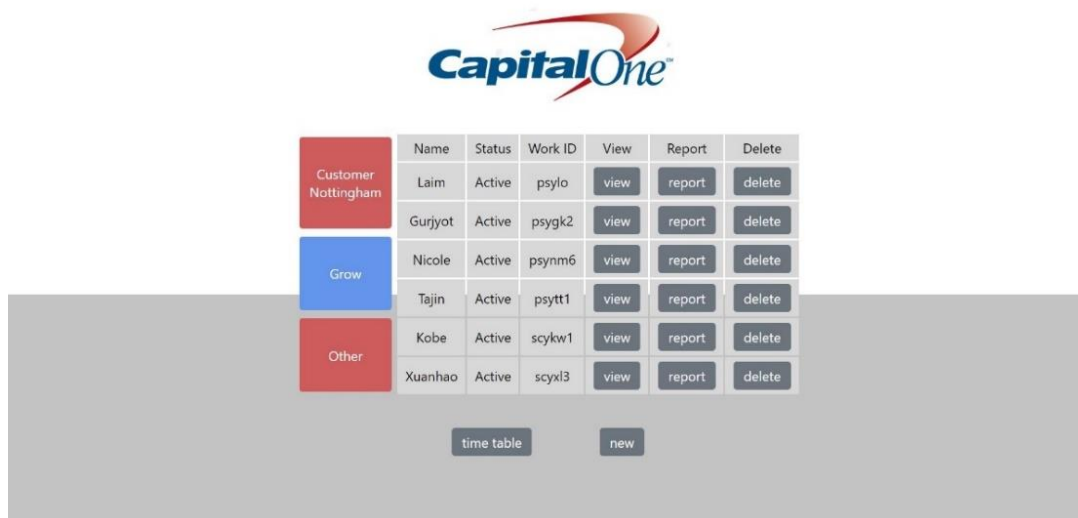
mm/dd/yyyy

submit

Change password

Log out

Administrator Interface:

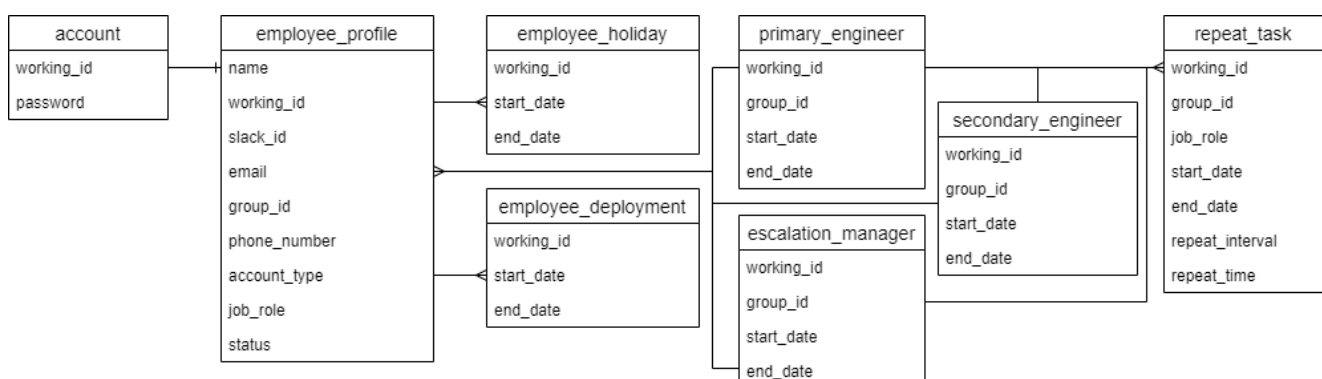


Development Tools

Our team has been using:

1. HTML, CSS, and JavaScript for web design as our team have experience using these languages from designing websites with them last semester
2. XAMPP, a free application for connecting to a local MySQL server where we can create the database for our web application. It also can be used to view our webpages
3. PHP and SQL for database creation and connection between the backend and front end as our team has experience using PHP and SQL and has easy access to PhpMyAdmin using XAMPP
4. GitLab for version control, and for creating branches for different features team members may be working on until we can merge our work
5. Visual Studio Code as our chosen IDE as it is installed across all lab machines as well as in our virtual environment. It includes a GUI for version control, and it has extensions for our chosen languages to help with debugging and code formatting
6. Microsoft Teams for documentation and project coordination because we can work concurrently, with multiple people editing the same documents at once
7. Agile development for continuous integration of any changed requirements and because it supports flexibility rather than just sticking to one plan
8. Incremental development to produce small amounts of high-quality code in a short amount of time. It is useful for ensuring that all of the requirements are met

Database Management



Following the principles of normalization, we divided the entire database into multiple sections.

The login system:

The account table contains the user's login details. The `working_id` attribute functions as the username and is the primary key.

Employee profile:

`Employee_profile` contains employees' personal information, including their Work IDs, Slack IDs, status, etc. The second database consists of three tables in total. The `working_id` is the primary key. The other two tables, `employee_holiday` and `employee_deployment`, store the start and end date of employees' holidays and deployment periods. They use the employee's work id as the foreign key to link employees to holidays and deployments.

Timetable Assignments:

The three tables, `primary_engineer`, `secondary_engineer` and `escalation_manager`, have the same attributes but store the timetable assignments for their respective role. The tables include the employee's Work ID, their Group ID, and the assignment's start and end date. The `repeat_task` table is used when an employee is assigned at repeated intervals onto the timetable, and so must also specify their job role, how long the repeated assignment should last, and the length of the interval between assignments.

Data management and manipulation

Each system we designed in this project is managing and manipulating data in different ways:

Timetable Interface

Whenever a new assignment to the timetable is created in the front end, a JSON package will be generated and delivered to the back end. The package contains the employee's work ID and the start and end date of the assignment. By combining data from the employee profile table and the timetable assignment tables, the system will determine whether the new assignment is valid. If it is valid, then the front end will update the timetable. Otherwise, the user sees an error message.

Another situation is when the client switches the timetable to a different month. The frontend system will send the month selected to the back end. The back end will respond with a JSON package containing that month's schedule information, i.e., what day of the week each day of that month falls on.

Log-in

When the user logs in, their inputted username and password are passed into the back end to check whether they match a record stored in the 'account' table.

Employee Profile

When a user tries to edit their profile, any changes are validated at the front end, e.g., the length of their entered telephone number is checked, the end date of any new holidays must be after the start date, etc. The back-end employee profile timetables can then be updated straight away, knowing that data integrity has been ensured already.

Testing

Because this project is a web application, the system integrity can be quickly evaluated by running the entire project or part of it on a local virtual webserver software called XAMPP.

To evaluate the programs' functionality, the testing processes between the front end and backend is slightly different:

- For the front end, dummy data has been created in a JavaScript file to simulate the process of receiving data from the backends. A PHP file acts as a fake database that tries to listen to the front end. And the software maintainers could manually test the functionalities by conducting the test document provided in the "Overview of testing" folder.

- For the backend, another PHP file connects to a local MySQL database created by XAMPP. The database contains test data we made up. The main testing is carried out by manipulating the database and checking our code responds appropriately. An integrated testing program is generated to execute all tests by clicking on the “Run Test” button.

The entire test strategy is conducted in a document stored in git lab. Later software maintainer could generate an integrated test process according to this document. More detailed information about the practical implementation could be found in the software manual.

REFLECTION

Project Management Challenges

One issue has been that people have not always uploaded their work onto Microsoft teams or GitLab, so everyone else cannot keep track of their progress. This brought tension and confusion onto the rest of the group as there's confusion about whether a task has been done, and therefore make them more inclined to do the task themselves just in case, which leads to multiple people unknowingly doing the same work. To solve this, we have been encouraging people to upload their work frequently, to ensure that everyone understands where each other are at. In the second semester, we tried to set a final due date on each task. By applying this brand-new strategy, our team's efficiency has been significantly improved. However, because every team members' personal advantages and capacities are different, it is still hard to set an accurate and suitable overdue date.

Another difficulty our team is currently confronting when building this system is individuals not being present for our meetings when we discuss our progress and potential new tasks that can be accomplished. When meetings have been planned for instances, there have been occasions where people have not responded, given days in advance. At the beginning of the second semester, we lost all connections with Xuanhao that we cannot get any response from him, and he is always absent from the meeting. To address this problem, we have reported this situation to our academic supervisor, and our supervisor has sent lots of emails to Xuanhao. However, we still get no response from Xuanhao. We are still trying to address this by keeping a record of attendance and making everyone aware of the importance of attending meetings.

Technical Challenges

One of the difficulties our team has faced when carrying out tasks within the project is getting to grips with using JSON files as a way of passing information from pages to pages, and manipulating the information into a format ready to be used on both the frontend and backend. Because the timetable interface needs to update the rostering data frequently, we decide to use AJAX for data transmission between frontend and backend and the backend needs to encrypt the data from the database into JSON and pass it to frontend. By solving this problem, Kejia and Liam are more familiar with AJAX protocol and gained lots of experience in connecting frontends with backends.

Another difficulty appeared when we are developing the mail system. To force our web application to send mails to a target mail address needs to apply the simple mail transfer protocol. By applying this SMTP, our web application should pin the SMTP server of the mail address and log into the server. According to Kejia's trials, it seems that the SMTP is blocking the access from our server, which is a virtual one set up on a personal computer, to the target STMP server. Although Kejia has spent lots of time in trying different approaches, he can hardly get any progress. Liam successfully addressed this problem by avoiding sending emails from a STMP server to another one since the access between them is blocked. He set up a mail system that sends mail to itself. By doing so, though the problem reminds unsolved, this mail system could still provide the same service to our customers.

The final difficulty is generated from the creation of the file generator. The sponsor specified that the report should be an excel file with specific formatting. However, it is hard to freely edit the cell colours, font size, etc. with JavaScript or PHP. Kejia researched online and found the only possible approach is to apply a library's professional version. After having a discussion with our sponsor, we reached an agreement that we are only allowed to use the library's community version to generate an excel file with simple formatting. The sponsor requested formatting and our actual formatting generated by this system can be found on Gitlab. By addressing this, Kejia has gained more experience in web development and has learned to negotiate effectively with the sponsor.

Contingency Measures

One contingency measure we made was for emergency considerations one or more of our team members may have that prevent them from working. If a team member has an emergency, we give each team member two or three extra weeks to complete their work before the task they were assigned is assigned to somebody else. In our original plan, we aimed to finish the system by the beginning of April because that would give time to complete any extra tasks like report writing and would allow team members to meet their deadlines in other courses. This meant that when the campus was shut down, the system was in a finished state, so the impact of not being able to meet in person was alleviated.

Another contingency was that we failed to meet all the requirements of the system before the deadline. Our measure for this was to rank the importance of our requirements as basic, important or extension. Basic requirements are essential functionalities of this web application. Basic requirements are ones that should be simple to implement and do not require much time. Important requirements are crucial features of the system that we have identified from the sponsor. Extensions are requirements that should only be considered when all the other requirements are met. This meant that we could guarantee that even if we did not meet all the requirements, we did meet the important ones.

Our third contingency measure was a backup plan in case the mail system could not be created. If we could not apply the SMTP protocol and address the port block problems, we could create an internal mail system that did not actually send emails, but just allowed administrators to view all leave requests from employees. This backup approach is easy to implement and provided similar functionalities to the original requirement.

Future Plans

We hope that the system we developed will be used by Madhu and her team, and it is possible that they will find more features to add to the system.

One possible extension could be to allow a month's roster to be generated automatically. It could look at all the employees in a group and assign them a role for a week as long as they are available. This could then be developed further by not assigning an employee too often, in order to make the timetable fairer.

Furthermore, an AI learning system could be created to learn from administrators' workload allocation strategies to provide better services.

Another extension could be developing an internal communication channel within the system to announce the administrator that an employee requests leave from the workload schedule. Moreover, this system should be able to directly lead the administrator to the target schedule, which could save a considerable amount of time for them to approve the leaving request. Furthermore, this system could also remind the administrators and employees of the progress of validating their leaving requests.

Conclusion

Overall, we have met most of the requirements of this project. During our final online meeting with our sponsor, Madhu said she was satisfied with the system, and she said it was ready for the presentation on May

14th. As the requirements and priorities of the project changed, we have altered our system to accommodate them. Although in the first semester, there were some issues with the functionality of our team, we have alleviated them in the second semester. In the second semester, we were more realistic with what progress could be made, and we communicated more with each other, which unified our team.

Part 2: Software Manual

PROJECT INTRODUCTION

This project's goal is to develop a rostering system to support Capital One's administrators to schedule their teams of employees onto a timetable.

This system will be applied in Capital One's local area network (**LAN**), and an **Apache** server is provided by the project sponsor for this web application's deployment. The project sponsor also desired a specific **UI** design for this system, and an example has been provided.



Figure. Target UI design

According to the sponsor, there is no budget on dependencies, libraries, or any external code bases applied in this project. This web application should be able to operate smoothly on **Chrome 77** and **Firefox 75.0** and above.

TERMINOLOGY

AJAX: Asynchronous JavaScript & XML. This is used to hold the communication between JavaScript and PHP files.

SMTP: Simple Mail Transfer Protocol. This is the internet protocol of sending mails by PHP programs.

PDO: PHP Data Objects. This is used to hold the communication between PHP files and MQL databases.

Apache HTTP Server: A free and open-source cross-platform web server software. It is applied to store web application programs.

SQL Database: Structured Query Language database. This is the database our team decided to use to store all data, including accounts, profiles, and rostering information.

XAMPP: A free and open-source cross-platform web server solution stack package developed by Apache Friends.

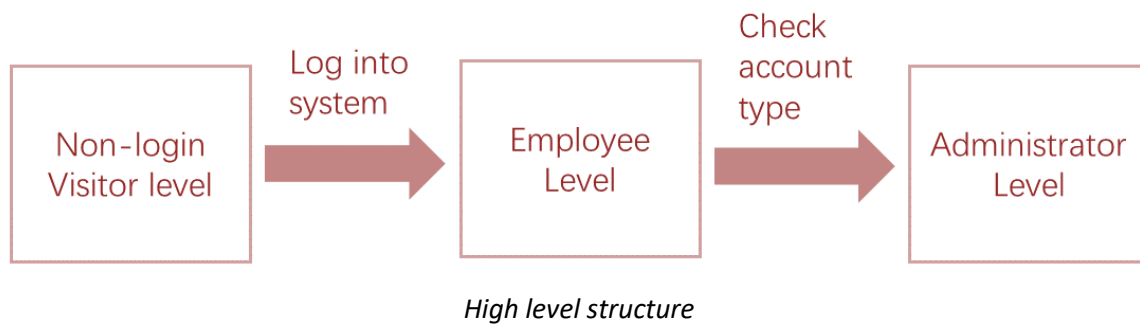
Timetable Interface: The home page of this web application that anyone can view. It contains the schedule timetable.

Employee Interface: This page is showing an employee's profile, including their holidays and deployment information. It can be edited by the employees themselves and administrators.

Administrator Interface: This page allows administrators to manage all employee accounts. It is also where reports for employees are generated.

Protected Data: This refers to the critical data that only administrators have the permission to edit, including employee status, working id, account type, and job role.

HIGH LEVEL STRUCTURE



Above is the high-level structure of the system. The latter levels give users more permissions, and visitors can jump to a higher level according to different actions.

Here is the list of permissions of 3 different levels' visitors, the latter level's visitor preserves all permissions from former levels:

Non-login Guest level

- Access rostering information of all workgroups
- Access which employees are in which group

Employee Level

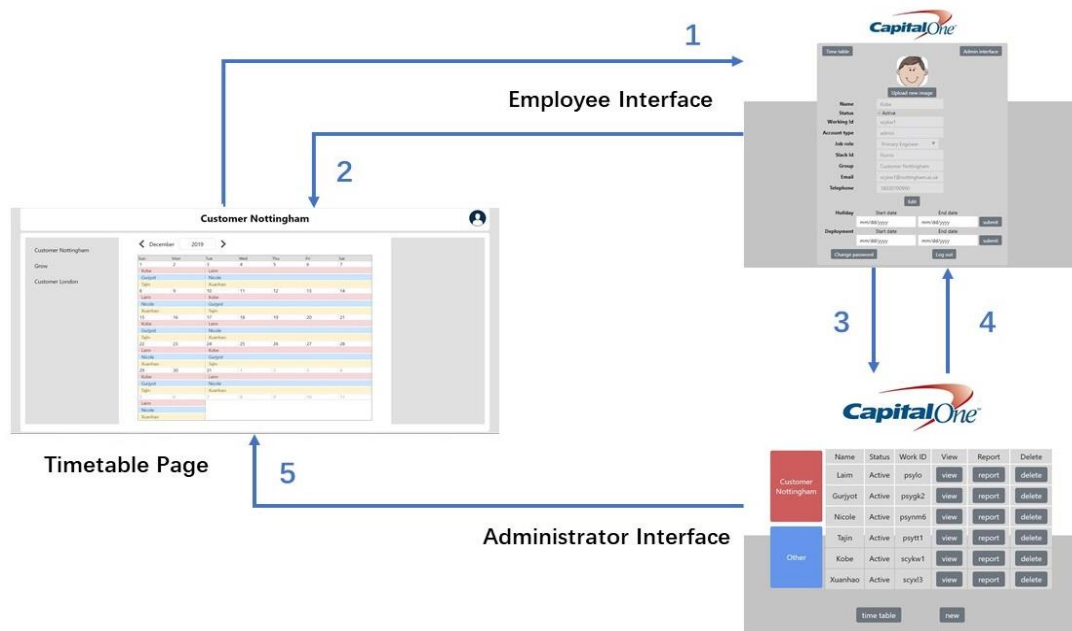
- Access and edit own profile (excluding protected data), holidays and deployments
- Request leave from own working schedule
- Change own password

Administrator Level

- Edit the rostering information of any workgroup
- Access and edit all employees' profiles
- Create and delete employees or change employee to the administrator
- Generate reports according to rostering information

ARCHITECTURAL DESIGN

This web application consists of three pages: the timetable interface, employee Interface, and administrator interface.



Five paths between pages are possible:

- 1) A visitor clicks on the profile picture in the top right corner to access his/her employee profile. If the user has not logged into the system yet, a login window will pop out to ask the user to login.
- 2) From the employee interface, the user clicks a button to return to the timetable, and the session will preserve the user's working id and password, so they do not have to log in again.
- 3) If the user's account type is "admin", then from their profile, they click a button to access the administrator interface.
- 4) An administrator can click on an employee to view their profile.
- 5) Administrators can return to the timetable by clicking a button.

DATA STORAGE

A typical SQL database is applied in this system for data storage, which is divided into eight tables in total. Here are the descriptions of the structure and function of each table:

Account

- This table is created to store employees' working id and password. The working id is set to be the primary key of this table because each employee has a unique working id.

Employee_deployment & employee_holiday

- This table is generated to store employees' holiday and deployment schedules. Working_id is used as the foreign key to connect records to specific employees.

Employee_profile

- The employee profile table is created to keep the employees' personal information. Working_id is set to the primary key. For new employees, only the working id and status will be initialized, the rest of the attributes are left blank.

Primary_engineer & secondary_engineer & escalation_manager

- This table has a similar structure to employee deployment and holiday. It has an extra column to store the group id of the schedule to distinguish employees belonging to different working groups.

Repeat_task

- This table is created to keep track of employees that have been repeatedly assigned to the timetable.

EXTERNAL DEPENDENCIES

The external dependencies and libraries of this system are listed below. Links towards more resources for these libraries are provided in the last section.

JQuery

JQuery is a small JavaScript library. It is applied in timetable interface files, employee interface, and test framework. Both part of the code base, JQuery is applied to select global elements. By applying JQuery, "document.getElementById("ID")" could be substituted by "\$("ID)". In this system, JQuery is imported as a min JavaScript file named "jquery-3.5.0.min.js".

SheetJS

SheetJS is an Excel JavaScript library that enables users to create, edit, and export excel files by using JavaScript code. This library is applied in "FileSaver.js" in administrator interface to generate a report in excel files according to rostering information. In this system, SheetJS is imported as a min JavaScript file named "xlsx.full.min.js".

File Saver

File Saver is a JavaScript codebase developed by Eli Grey and shared on Git hub. This codebase is applied in administrator interface to apply "xlsx.full.min.js" generating report in excel files according to rostering information. In this system, File Saver is imported as a JavaScript file named "FileSaver.js".

PHPMailer

PHPMailer is a code library to send (transport) emails safely and easily via PHP code from a web server. This library is applied in timetable interface files to send mails to an embedded mailbox to inform the administrator that an employee requested to leave from his/her duty. This library is installed by the composer and applied by “sendMail.js”. In this system, PHPMailer is imported as a file named “phpmailer” by the composer.

Composer

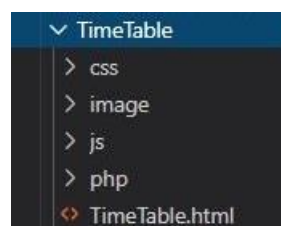
Composer is an application-level package manager for the PHP programming language that provides a standard format for managing dependencies and libraries. It is applied to manage the PHPMailer library and possible future maintenance.

DESIGN PATTERNS AND CODING CONVENTIONS

An MVC design pattern has been applied to each page. The Model is realized in PHP files, and the View is implemented by HTML and CSS files. JavaScript acts as the Controller by connecting the View and Model.

The entire system is broken up into different folders for different components, e.g. the timetable, the mail system, the employee interface, etc. Each component folder contains subfolders for the PHP and JavaScript files. A JavaScript file has been set up, which builds the entire database, to help clients deploy the entire system automatically. All comments are formatted in the same way, being placed above the line of code the comment is explaining.

Within each web page, timetable interface, employee interface, and administrator interface, all code files are categorized into “CSS”, “JS” and “PHP”. HTML files are placed in the outer folder to be applied as a framework of the web page. For example, this is the folder hierarchy of the timetable interface.



Within this page, “TimeTable.html” performs as the view component in an MVC system, which could be directly manipulated by JavaScript files according to AJAX protocol. JavaScript files perform as the controller components, which connect the view component with the model components to make sure that all changes conducted by the model components will be updated on the view components as soon as possible. The PHP files perform as the model components which update their data from the database and pass it to the controller components for further processes. Consequently, “TimeTable.html” is separated from the backend database and only acknowledges the existence of JavaScript files. Furthermore, the entire code structure satisfied the software engineering’s principles of high cohesion and low coupling, which could significantly benefit future software maintenance and testing process.

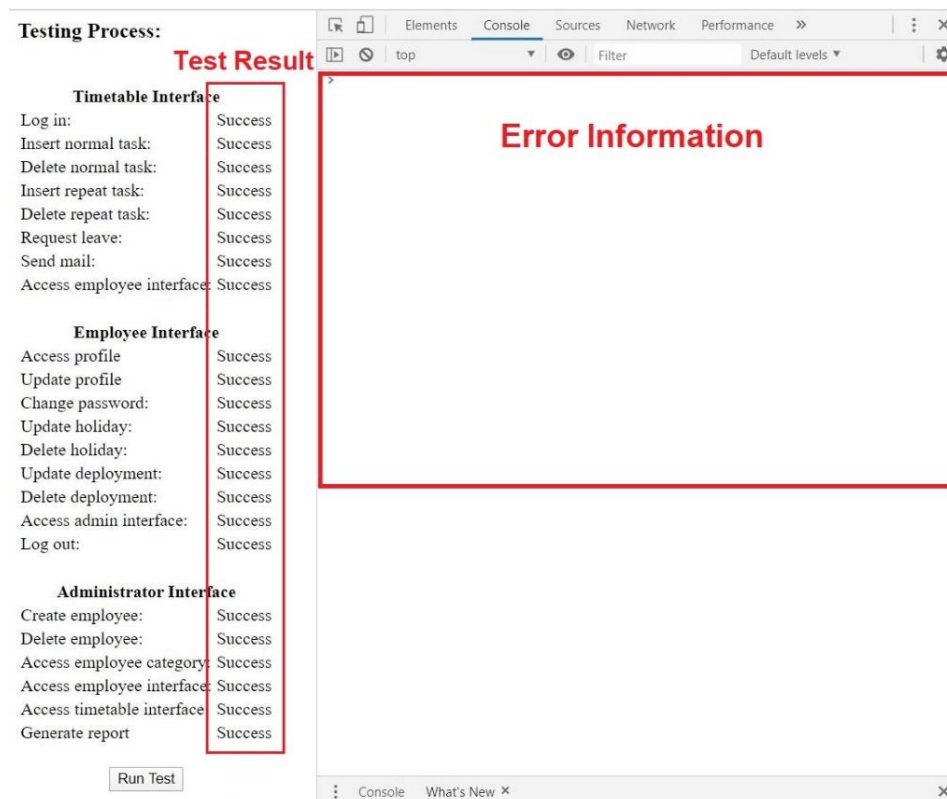
TESTING STRATEGY

Within this project, a folder named “Brief overview of testing” has been created to contain all test documents and testing codebase. The testing strategy of this system is divided into two parts, frontend testing and backend testing.

To test the frontend of this system, software maintainers must deploy the entire system on their server. It is hard to generate a testing program towards the frontend because the interactions between HTML files and JavaScript files are impossible to simulate without completely build up the web page. Therefore, the best approach to test the frontend is to go through all the vulnerable boundaries within this system manually. We have created a test plan that helps test each part of the frontend individually. This means testing the

individual components to ensure they are showing the correct behaviour intended. This test plan is constructed according to tips from software maintenance. Moreover, all components' features and possible test steps are listed in a word document named "Testing for the current system from employee and admin point of view". By following this document, the client or software maintainer could quickly test this system manually. This document also demonstrates the correct system response after specific actions. Consequently, future code maintainers could quickly debug this web application.

A testing interface, which is called "Testing", is developed to help code maintainers and clients run an integrated test process towards the backend. The picture below is a screenshot of this integrated test program.



According to this picture, the entire system backend is divided into three different web pages. And all of the functionalities will be tested after maintainer click the "Run Test" button. After that, all test results will be demonstrated on the right-hand side of the corresponding function names. If part of the testing is failed, the detailed error information will be shown in the browser's console, which could be observed by using the developer tools (Tab F12 key on the keyboard).

This test program enabled the backend PHP files to operate separately by embedding data into the codebase. Consequently, instead of extracting data from the HTML files, JavaScript file could directly get the embedded data value, and the testing process becomes controllable. Future maintainers could edit the testing data themselves to generate a new testing program for their version.

```

Prototype > Brief overview of testing of current system (excluding mailing system) > js > JS TestAdminInterface.js > finishAI
1  var employeeReport;
2  var employeeProfile;
3  var targetWorking_id = "scykw1";
4  var targetJobRole = "PrimaryEngineer";
5  var payment = "35";
6  var targetYear = "2019";
7  var targetMonth = "12";
8

```


This integrated test program is mainly constructed by rewriting the original JavaScript files to manipulate the backend database by applying the original PHP files. The screen shot below compares the disparity between the original JavaScript file and the rewritten one in this testing program. The red block selected out two different approach of obtaining target value. The picture on the right-hand side obtained these data directly from the HTML file using “document.getElementById()” function, while within the testing, these data is embedded in the integrated program in left-hand side program.

```
function uploadHoliday() {
    newHolidayStart_date = document.getElementById("newHolidayStart_date").value;
    newHolidayEnd_date = document.getElementById("newHolidayEnd_date").value;

    if (checkLoadDeployment()) {
        var url = "../php/timeManagement.php";
        var data = "target=holiday&action=upload&start_date=" + newHolidayStart_date
            + "&end_date=" + newHolidayEnd_date + "&working_id=" + targetWorking_id

        AJAX.post(url, data,
            function (responseText) {
                if (responseText == "Success") {
                    getEmployeeHoliday();
                    document.getElementById("newHolidayStart_date").value = "";
                    document.getElementById("newHolidayEnd_date").value = "";
                }
            }
        );
    }
}
```

```
function updateHoliday() {
    newHolidayStart_date = "2020-09-08";
    newHolidayEnd_date = "2020-09-13";

    if (true) {
        var url = "../EmployeeInterface/php/timeManagement.php";
        var data = "target=holiday&action=upload&start_date=" + newHolidayStart_date
            + "&end_date=" + newHolidayEnd_date + "&working_id=" + "scykwl";

        AJAX.post(url, data,
            function (responseText) {
                if (responseText == "Success") {
                    document.getElementById("UH").innerHTML = "Success";
                    deleteHoliday();
                } else {
                    document.getElementById("UH").innerHTML = "Failed";
                    console.log(responseText);
                }
            }
        );
    }
}
```

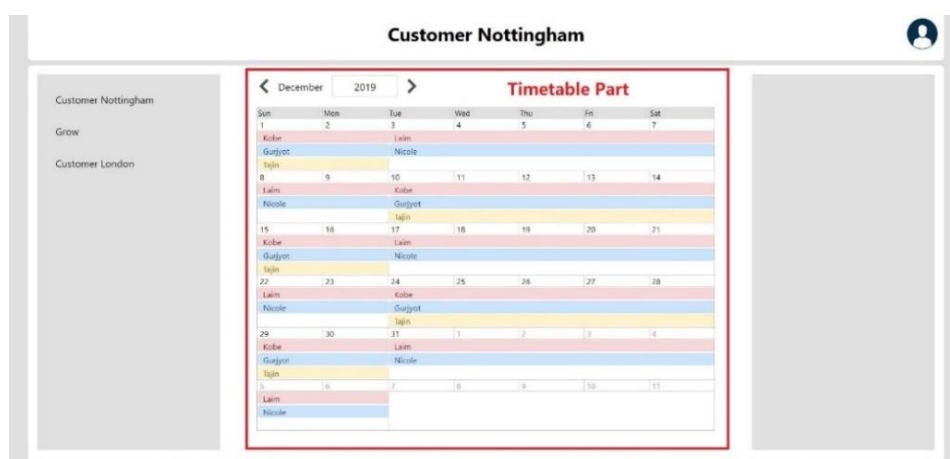
However, there are two limitations of this program: First, maintainers must execute this program right after database initialization to ensure the correctness of embedded data. Second, the maintainer must wait for this testing program terminates automatically. Otherwise, it could end up modifying the database permanently, and the maintainer must reinitialize the database again to pass this integrated test program.

FUNCTIONALITY IMPLEMENTATION

All implementations of specific functionalities are demonstrated below, which is divided into three different interfaces for detailed explanations.

Timetable Interface

1) Timetable Part



The timetable part of timetable interface is constructed by JavaScript in real time because of its high frequency of update process. Each time after the time selector has been reset, new schedule has been inserted or old schedule has been deleted, a function named “buildTimeTable()” will be called to generate the HTML code of the entire timetable part and insert it into the timetable interface. An array consists of twenty-one objects is passed from the database, which contains all data of the selected

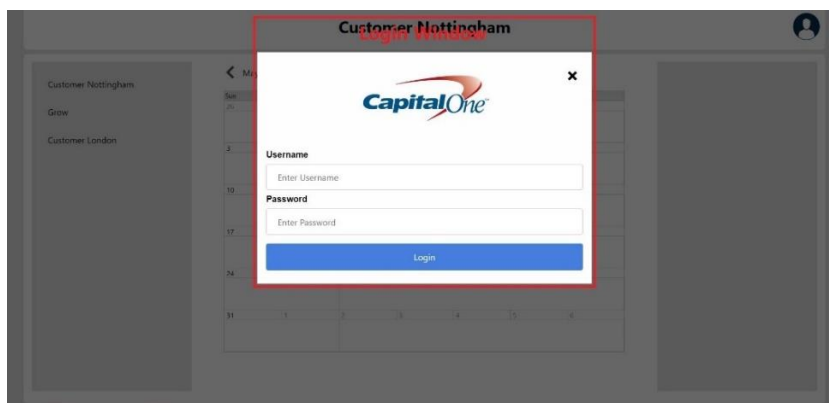

```

{Grow: Array(1), Customer Nottingham: Array(6)}
  ▼ Customer Nottingham: Array(6)
    ▶ 0: {name: "Tajin", working_id: "psytt1", group_id: "Customer Nottingham", job_role: "EscalationManager"}
    ▶ 1: {name: "Nicole", working_id: "psynm6", group_id: "Customer Nottingham", job_role: "SecondaryEngineer"}
    ▶ 2: {name: "Laim", working_id: "psylo", group_id: "Customer Nottingham", job_role: "PrimaryEngineer"}
    ▶ 3: {name: "Kobe", working_id: "scykwl", group_id: "Customer Nottingham", job_role: "PrimaryEngineer"}
    ▶ 4: {name: "Gunjyot", working_id: "psyk2", group_id: "Customer Nottingham", job_role: "SecondaryEngineer"}
    ▶ 5: {name: "Dario", working_id: "scyk33", group_id: "Customer Nottingham", job_role: "PrimaryEngineer"}
    length: 6
    __proto__: Array(0)
  ▼ Grow: Array(1)
    ▶ 0: {name: "Nike", working_id: "scyk3", group_id: "Grow", job_role: "SecondaryEngineer"}
    length: 1
    __proto__: Array(0)
    proto: Object

```

Each entry of this array contains four attributes, including: name, working id, group id and job role. All of these four attributes are embedded into the div block within this employee category as well. By doing so, the listener components could directly apply these attributes as parameters according to the “event” object. However, the drawback of this method is it slightly jeopardizes the efficiency of building these blocks. But considering the high power of present browser, this drawback could be ignored.

3) Log in window



After guests have clicked on the timetable or employee category, an account checking program will be triggered. This program will check the existence of the session which stores the account information. If the session does not exist, the login window will automatically pop out to ask the guest to log into this system. This program is also able to check the type of the account and match two identical accounts. To achieve this functionality, this login program will extract employee account information from the database to obtain the account type.

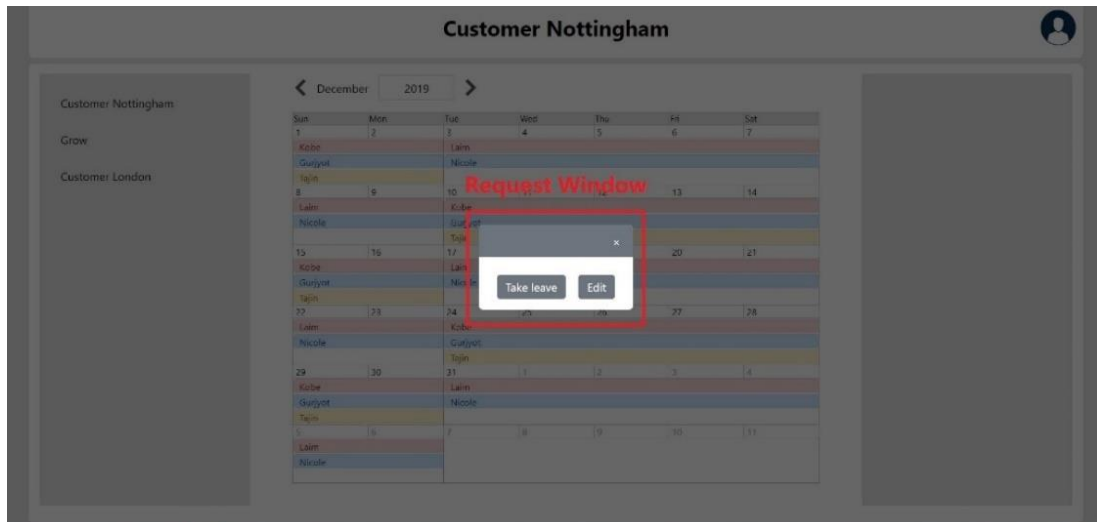
```

Prototype > TimeTable > php > login.php
1  <?php
2  include_once("../db_connection.php");
3
4  isset($_SESSION) OR session_start();
5
6  $action = $_POST['action'];
7
8  switch ($_POST['action']) {
9      case 'login':
10         login();
11         break;
12
13     case 'check':
14         checkLogin();
15         break;
16
17     case 'match':
18         checkMatch();
19         break;
20
21     default:
22         echo "Wrong instruction ".$_POST['action'];
23         break;
24 }
25

```

This php file has been applied throughout these three pages, which has a high reusability. And by storing user's login information into a session instead of cookie, the system becomes more secure because session stores in the backend (constructed by PHP) and could hardly be modified in the frontend.

4) Request window



After the user have logged into this system, he could request a leave from existing schedule. If the working id of that schedule is identical with the user's working id, an email will be generated by the "sendLeaveRequest()" function then be automatically sent to a pre-set mail box. Within this process, those attributes embedded into the div block will be used

```
//SMTP Settings
$mail->isSMTP();
$mail->Host = "smtp.gmail.com";
$mail->SMTPAuth = true;
$mail->Username = "psylo1998@gmail.com";
$mail->Password = 'Team35#Capital';
$mail->Port = 465; //587
$mail->SMTPSecure = "ssl"; //tls

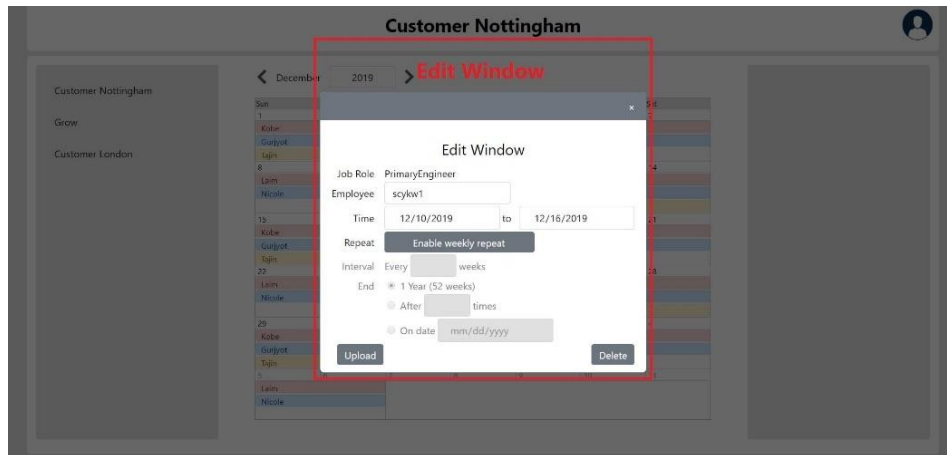
//Email Settings
$mail->isHTML(true);
$mail->setFrom($email, $name);
$mail->addAddress("psylo1998@gmail.com", "Administrator");
$mail->Subject = $subject;
$mail->Body = $body;
```

Here is the code of SMTP settings of this system. By applying PHPMailer, a ping request will be sent from local computer through port 465, which is a particular mail service port, to the Gmail SMTP server. And after received a correct response from the server, local computer will request to log into the server by entering above username and password. Once the local computer has successfully logged into the server, a message encoded in ssl method will be sent to the server, and the server will send this message directly to address stored in the email settings part with the target mail subject and mail body.

The mail subject and body are two local variables which store a block of text. These two variables contain target schedule's information, such as start date, end date, job role and working id. According to the information stores within, administrators are able to know which schedule has been requested for a leave. However, a latency period could be observed before the system response to the user. This is cause

by the process of connecting the target SMTP server. An asynchronous execution method should be considered in future project which might improve the user experience.

5) Edit window



From previous step, if the user request to edit the target job block and clicked on the “Edit” button, the login program will automatically check the account type. If user’s account type is “admin”, this user will be permitted to edit this job block and an edit window will pop out.

Here exist two approaches to edit the rostering information:

First, edit one task only within a single edition process. By applying this approach, a data block will be generated to store the schedule information. Here is an example block.

```
{group_id: "Customer Nottingham", job_role: "PrimaryEngineer", working_id: "scykw1", start_date: "2020-04-28", end_date: "2020-05-04"}
end_date: "2020-05-04"
group_id: "Customer Nottingham"
job_role: "PrimaryEngineer"
start_date: "2020-04-28"
working_id: "scykw1"
proto : Object
```

Five attributes have been stored within this object, including group id, job role, working id, start date and end date. After that, a function named “normalUpload()” will be called to upload the data block into the data according the job role attribute. Vice versa, if the user request to delete this schedule, the same data block will be created and sent to the database by another function named “normalDelete()”. However, this time, the database will delete the identical data block from the table.

The second approach is to repeatedly assign tasks into the database. A similar data block with some extra attributes will be generated by applying this approach.

```
{group_id: "Customer Nottingham", job_role: "PrimaryEngineer", working_id: "scykw1", start_date: "2020-04-28", end_date: "2020-05-04", ...}
end: "Year"
end_date: "2020-05-04"
group_id: "Customer Nottingham"
interval: "3"
job_role: "PrimaryEngineer"
start_date: "2020-04-28"
working_id: "scykw1"
proto : Object
```

Besides all attributes mentioned in above data block, an end type and interval attribute have been added into this object simultaneously. The process of repeatedly assign tasks to simply repeatedly call the previous “normalUpload()” function with a number of times calculated from the end type and

repeat interval. In order to enable administrators to manipulate repeated tasks, all repeated tasks' information is stored in a repeat task table. Consequently, all of those repeatedly assigned tasks could be quickly traced by this system according to simple calculation.

Within the process of uploading, the schedule information will be evaluated by a PHP program automatically, which decides whether this schedule violates some criteria. The picture below demonstrates the evaluation PHP program.

```
//Check empty
foreach($block as $key => $value){
    if($value == ""){
        echo $key." is emptied";
        return false;
    }
}
```

First, this program will check whether target period is empty for work allocation.

```
//Status
$profile_sql = "SELECT * FROM employee_profile WHERE working_id=\"\".$block->working_id.\"\"";
try{
    $dbh=PDOProvider();
    $stmt=$dbh->prepare($profile_sql);
    $stmt->execute();

    $row=$stmt->fetch(PDO::FETCH_ASSOC);

    if(!$row['status']){
        echo "The employee is inactive";
        return false;
    }else if($row['job_role'] != $block->job_role){
        echo "The employee's correct job role is ".$row['job_role'];
        return false;
    }
}catch(PDOException $error){
    echo "SQL Query: ".$profile_sql."<br>";
    echo "Connection failed: ".$error->getMessage();
}
```

Second, this program will request the employee profile information from the database to check whether the employee's status is active. If not, this request will be rejected.

```
//Holiday
$prevDay = date("Y-m-d",strtotime("-1 day",strtotime($block->start_date)));
$holiday_sql = "SELECT * FROM employee_holiday WHERE working_id=\"\".$block->working_id.\"\" " .
    "AND (( start_date>=\"\".$prevDay.\"\" AND start_date<=\"\".$block->end_date.\"\" ) " .
    "OR ( end_date>=\"\".$prevDay.\"\" AND end_date<=\"\".$block->end_date.\"\"));";
try{
    $dbh=PDOProvider();
    $stmt=$dbh->prepare($holiday_sql);
    $stmt->execute();

    $row=$stmt->fetch(PDO::FETCH_ASSOC);

    if($row != null){
        echo "The employee's holiday from ".$row['start_date']." to ".$row['end_date'];
        return false;
    }
}catch(PDOException $error){
    echo "SQL Query: ".$holiday_sql."<br>";
    echo "Connection failed: ".$error->getMessage();
}
```

```
//Deployment
$deployment_sql = "SELECT * FROM employee_holiday WHERE working_id=\"\".$block->working_id.\"\" " .
    "AND (( start_date>=\"\".$block->start_date.\"\" AND start_date<=\"\".$block->end_date.\"\" ) " .
    "OR ( end_date>=\"\".$block->start_date.\"\" AND end_date<=\"\".$block->end_date.\"\"));";
try{
    $dbh=PDOProvider();
    $stmt=$dbh->prepare($deployment_sql);
    $stmt->execute();

    $row=$stmt->fetch(PDO::FETCH_ASSOC);

    if($row != null){
        echo "The employee's holiday from ".$row['start_date']." to ".$row['end_date'];
        return false;
    }
}catch(PDOException $error){
    echo "SQL Query: ".$deployment_sql."<br>";
    echo "Connection failed: ".$error->getMessage();
}
```

After that, this system will also check the schedule to make sure it will not be interrupted by any holidays or deployments.

```
//TimeTable
$employeeNum = getNumOfEmployeeWithSameJobInOneGroup($block->group_id, $block->job_role);
if(date("Y-m-d",strtotime("+6 day",strtotime($block->start_date))) != $block->end_date){
    echo "Wrong time interval ".$block->start_date." " . $block->end_date;
    return false;
}else if(date("w", strtotime($block->start_date)) != 2){
    echo "Wrong weekday";
    return false;
}
if($employeeNum >= 2){
    $employeeNum = $employeeNum - 2;
}else{
    $employeeNum = 0;
}
```

Finally, the last criteria will be evaluated, that the associate must appear no more than once every 'n - 2' weeks, given there exists 'n' associates on the rota.

Employee Interface

1) Employee Profile Part

The screenshot shows the 'Employee Profile Part' of the Capital One interface. It features a central form with a red border. The form contains a profile picture placeholder with an 'Upload new image' button. Below this, there are several fields for employee information: Name (Kobe), Status (Active), Working Id (scykw1), Account type (admin), Job role (Primary Engineer), Slack Id (Norris), Group (Customer Nottingham), Email (scykw1@nottingham.ac.uk), and Telephone (18030700990). There are also buttons for 'Time table', 'Admin interface', and 'Edit'.

The employee profile part of employee interface is constructed by JavaScript in real time, which is similar to timetable part. A function named "refreshEmployeeProfile()" is called to request the employee profile data block from the database and insert corresponding data into the correct place demonstrated in above picture. Here is an example data block.

```
▼ [{...}] ⓘ
▼ 0:
  account_type: "admin"
  email: "scykw1@nottingham.ac.uk"
  group_id: "Customer Nottingham"
  job_role: "PrimaryEngineer"
  name: "Kobe"
  phone_number: "18030700990"
  slack_id: "Norris"
  status: "1"
  working_id: "scykw1"
  ▶ __proto__: Object
length: 1
▶ __proto__: Array(0)
```

By observing above picture, all attributes are stored in this data block, such as the account type, group id and job role. If the user requests to update his employee profile, this system will check the account type first. If user's account type is "admin", he will be enabled to manipulate all data within. Otherwise, account critical attributes, for example, account type, job role, group id, working id and status could not be manipulated by the user, and the profile will only be partially updated.

By applying this implementation, the code reusability has been significantly improved. And the structure of "employeeInterface.html" becomes more readable.

2) Holiday & Deployment

A PHP program named “refresh()” is automatically executed before each time, to clean out all expired holidays and deployment to maintain the database space. In the frontend, two functions named “buildEmployeeHoliday()” and “buildEmployeeDeployment()” have been called after user has modified his holiday or deployment schedule.

This implementation addressed the expired data problem efficiently, and the design of the “Input” type components make sure that users could not enter an already expired period, or an invalid period, for example, end date is earlier than start date. This design makes the validity checking process more efficient and simple, which saves a great amount of calculation power of the browser.

3) Log out

By clicking on the “Log out” button, the session storing user’s working id and password will be destroyed and he will be automatically sent back to the timetable interface.

Administrator Interface

1) Employee Information Part

	Name	Status	Work ID	View	Report	Delete
Customer Nottingham	Laim	Active	psylo	view	report	delete
	Gurjyot	Active	psyk2	view	report	delete
Grow	Nicole	Active	psynm6	view	report	delete
	Tajin	Active	psytt1	view	report	delete
Other	Kobe	Active	scykw1	view	report	delete
	Dario	Active	scykw33	view	report	delete

The employee information part of administrator interface is also constructed by JavaScript. A function named “buildEmployeeTable()” is called to request the employee information data block from the database and insert corresponding data into the correct place demonstrated in above picture. Here is an example data block.

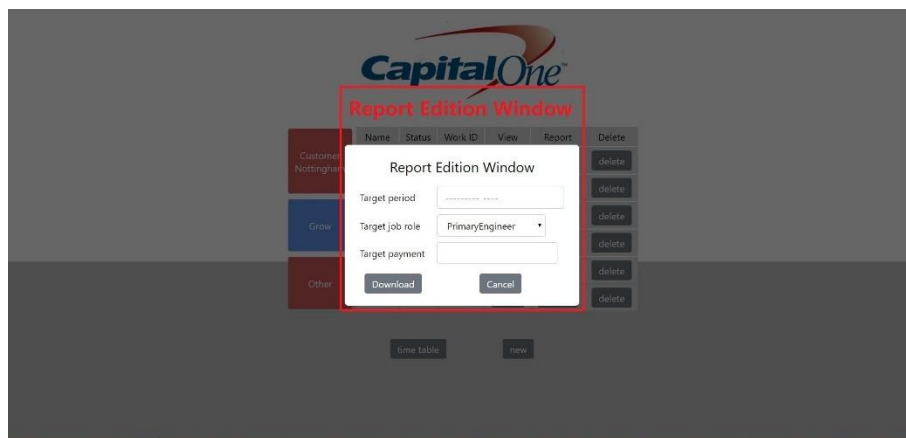
```

▼ (7) [{...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ
  ▶ 0: {name: "Laim", group_id: "Customer Nottingham", status: "1", working_id: "psylo"}
  ▶ 1: {name: "Gurjyot", group_id: "Customer Nottingham", status: "1", working_id: "psygk2"}
  ▶ 2: {name: "Nicole", group_id: "Customer Nottingham", status: "1", working_id: "psynm6"}
  ▶ 3: {name: "Tajin", group_id: "Customer Nottingham", status: "1", working_id: "psytt1"}
  ▶ 4: {name: "Kobe", group_id: "Customer Nottingham", status: "1", working_id: "scykw1"}
  ▶ 5: {name: "Dario", group_id: "Customer Nottingham", status: "1", working_id: "scykw33"}
  ▶ 6: {name: "Nike", group_id: "Grow", status: "1", working_id: "scykw3"}
    length: 7
    ▶ proto : Array(0)

```

Above data block contains all employees' information within this system, including their name, group id, status and working id. They are separately demonstrated in different groups. These attributes has been embedded into the employee's entry too.

2) Document Generator



Once the administrator clicked on the report button, a report edition window will pop out to collect the target period, job role and payment of the employee. After that, all data are passed to a function named "reportDownloadRequest()" which applies the "documentGenerator.js" file to create a excel file with the assistance of SheetJS.

```

var report = [
  [],
  ["Associate Name", employeeProfile['group_id']],
  ["Department Name", "Software Engineering"],
  ["Cost Centre", "50587"],
  ["Employee ID", targetWorking_id],
  ["Employee Number", employeeProfile['phone_number']],
  ["Employee Job Role", targetJobRole],
  ["Claim Month/Year", targetMonth + "/" + targetYear],
  []
]

employeeReport.forEach(element => {
  report.push([element["date"], element["onDuty"]]);
  if (element["onDuty"] == "Y")
    onduyCounter++;
});

report.push([]);
report.push(["Payroll Instruction: ", "Please pay " + onduyCounter + " days ( £" + payment + "/day )"]);
report.push(["Total: ", "£" + onduyCounter * payment]);

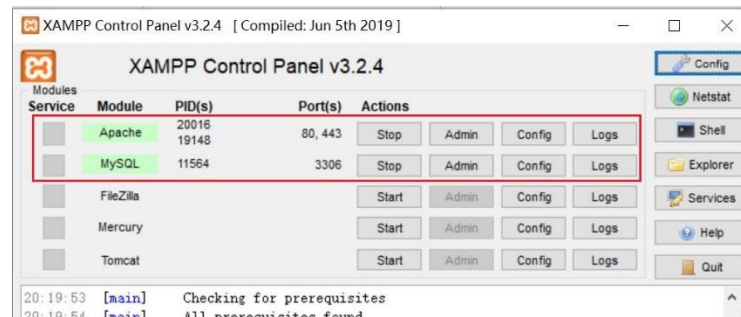
```

All information will be inserted in a variable named report following a particular formatting. By passing the variable to a min JavaScript file called "xlsx.full.min.js", an excel file will be generated. Then, "FileSaver.js" could be used to covert the excel file into a binary big object and automatically download it

into the user's personal computer. Within the process of development, we obtained a huge amount of experience in conducting online document. And we believe we could do better in a future project.

DEPLOYMENT

This system has been developed to be deployed on an Apache server and requires a SQL database. A client could install the SQL database on a company's server, or he could install an integration platform such as XAMPP, which integrates an Apache server and SQL database inside.



After the installation of the Apache server and SQL database, the client can directly clone the codebase from the GitLab repository. The user will need to edit the "db_connection.php" file to enable the system access to the SQL database on your server.

```
1 <?php
2 function PDOProvider(){
3     $dsn = 'mysql:host=localhost;dbname=rosteringsystem';
4     $user = 'team35';
5     $password = 'team35';
```

A PHP program is developed to automatically setup and initialize the SQL database. This file will output a response message to inform the client whether the setup and initialization succeeded. The client can edit the contents of "account.json", "timetable.json" and "employeeProfile.json" to modify the data inserted into the system during the process of initialization.

MAINTENANCE

Some strategies developed for the maintenance of this system are listed below:

Duplicate data

For this system, duplicated data could exist in "employee deployment", "employee holiday", "primary engineer", "secondary engineer" and "project escalator" tables, which do not have a primary key. A PHP program is applied to check duplication before each insertion operation.

```
74 function checkDuplicate($working_id, $employeePassword){
75
76     $sql = "SELECT COUNT(*) FROM employee_profile WHERE working_id='".$working_id."'";
77
78     try {
79         $dbh=PDOProvider();
80         $stmt=$dbh->prepare($sql);
81         $stmt->execute();
82
83         $row=$stmt->fetch(PDO::FETCH_ASSOC);
84
85         if(implode($row) == "0"){
86             uploadEmployee($working_id, $employeePassword);
87         }else{
88             echo "duplicate working_id";
89         }
90     } catch (PDOException $error) {
91         echo 'SQL Query:'. $sql. '<br>';
92         echo 'Connection failed:'. $error->getMessage();
93     }
94 }
```

The above code will check whether the inputted working id is already in the table. And the result will be transmitted back to the front end for further process.

Furthermore, duplicate data will be deleted by the search mechanism of the SQL database. Because SQL operation involves condition search, which will list out all entries that matches the target condition.

Expired data

Expired data could exist in “employee deployment”, “employee holiday” tables. As time passes by, past holidays and deployments are no longer useful and should be eliminated to create more storage space. A PHP program is executed each time one visitor accesses his/her employee interface.

```

3
4  $date = $_POST['presentDate'];
5
6  $sql = "DELETE from employee_holiday WHERE end_date < \"\".$date.\"\"";
7  $sql = $sql."DELETE from employee_deployment WHERE end_date < \"\".$date.\"\"";
8

```

EVOLUTION

Here are some user-friendly functionalities which could be developed by later maintainers in the future:

Leave request approval

Although a functioned mail system has already been developed, it is still a fairly slow process for administrators because when they receive a leave request, they must remove the employee from the schedule manually. An inner communication system could be developed where users click a link on the email and go straight to the leave date.

Search engine

If the number of employees grows significantly, it could be much more difficult for an administrator to find and select one specific employee. As a result, a search engine could be developed to enable administrators to search employees by their working id or name.

AI rostering system

An AI learning system could be developed to assign employees to the timetable automatically. This system should be able to analyse how employees have previously been rostered, including things like if an employee tends to go on deployment at the same time, the AI should not assign them at that time. Administrators could save much time and obtain a more optimal schedule that should not have to be tinkered with and changed.

LINKS TO EXTRA RESOURCES

- JQuery: <https://jquery.com/>
- SheetJS: <https://sheetjs.com/>
- File Saver: <https://github.com/eligrey/FileSaver.js>
- PHPMailer: <https://github.com/PHPMailer/PHPMailer>
- Composer: <https://getcomposer.org/>
- Test Strategy: [https://projects.cs.nott.ac.uk/COMP2002/2019-2020/team35_project/tree/master/Prototype/Brief%20overview%20of%20testing%20of%20current%20system%20\(excluding%20mailing%20system\)](https://projects.cs.nott.ac.uk/COMP2002/2019-2020/team35_project/tree/master/Prototype/Brief%20overview%20of%20testing%20of%20current%20system%20(excluding%20mailing%20system))

Part 3: User Manual

INTRODUCTION

This web application is designed for Capital One's administrators and engineers to provide them a better online workload schedule environment to improve their efficiency. It covers almost all requirements from the project sponsor, and it is also simple for our clients to deploy and maintain.

In this user manual, we will teach you how to install this web application on your server and how you could apply it in an industrial environment. Several use case examples will be provided to simulate real world scenarios, and the requirements towards hardware and software are demonstrated as well.

INSTALLATION AND CONFIGURATION MANUAL

This system can be installed on an Apache server using these steps:

- 1) A virtual Apache server should be installed on the user's personal device. Here we suggest our clients use XAMPP, which is what we have used throughout the project.



- 2) Download the codebase from GitLab and store it into the server. You could either clone the entire repository, or just download the 'Prototype' folder, which contains the complete system.
- 3) Edit the "db_connection.php" file to enable the system to access your SQL database. Using XAMPP, you should only need to change the user and password variables.

```

1  <?php
2  function PDOProvider(){
3      $dsn = 'mysql:host=localhost;dbname=rosteringsystem';
4      $user = 'team35';
5      $password = 'team35';
6  }

```

- 4) Run the "setUpDatabase.php" file to create the entire database and all the tables. If the database has been set up successfully, then a success message will be displayed. Otherwise, an error message will be outputted.
- 5) Run the "initDatabase.php" file to initialize the entire database with dummy data. Edit this file to change what data is stored in the database. A success or error message will be displayed.

HARDWARE AND SOFTWARE REQUIREMENTS

This web application could be installed on any Apache server with a SQL database. If the client wants to install it on his/her personal computer, a virtual Apache server environment and a SQL database should be set up to simulate the operating environment of this web application. This could be easily achieved by applying the XAMPP platform, which integrates the Apache server and SQL database together and works smoothly on Mac and Windows devices.

Any computer can access the webpage using Chrome 77 and Firefox 75.0 and above. The computer does not need to have an internet connection because all required libraries are inserted in the project, and all data transmission is handled on the server side.

How To USE

Below are how to use the different features of the system.

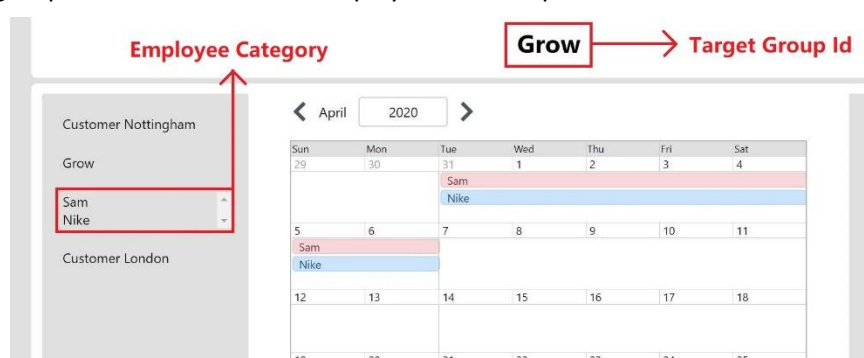
Search for schedule information within a specific period:

To search for schedule information, users do not have to log into the system. The system will automatically demonstrate the schedule information for the current month. If the user wants to view the rostering information in other months or other years, he/she could input the target year into the year selection bar and click on the left or right arrows to jump to a specific month.



Switch and view other groups' schedule information:

To view other groups' rostering information, users must switch the target group of their timetable. Click on the desired group name on the left-hand side of the home page. A list of all the employees in that group will pop out, and the group will be switched and displayed at the top.



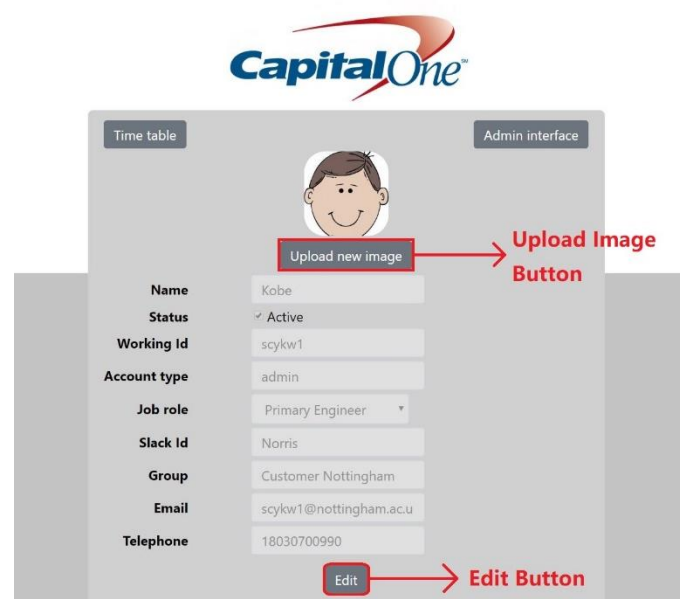
Access employee interface

After logging into the system, an employee could access his/her own profile by clicking the user logo on the top right corner.



Edit Profile

By clicking the edit button, the user can edit their personal profile, including their name, slack ID, group ID, email address, and telephone number. By clicking the upload new image button below the avatar, the user can change their profile picture.



Edit holiday and deployment schedule

To upload a personal holiday or deployment schedule, employees could enter the start date and end date of the target period, and the system will automatically check the correctness of the input period to make sure it is valid.



Change user password

Employees could change their own password by clicking the change password button located at the bottom.

Telephone 18030700990

Edit

Holiday Start date mm/dd/yyyy End date mm/dd/yyyy submit

Deployment Start date mm/dd/yyyy End date mm/dd/yyyy submit

Change password Log out

Change Password Button

Log out of the system

Employees could log out of the system by clicking the log out button, and the session which was used to keep users' working id and password will be destroyed immediately to protect the user's personal data.

Telephone 18030700990

Edit

Holiday Start date mm/dd/yyyy End date mm/dd/yyyy submit

Deployment Start date mm/dd/yyyy End date mm/dd/yyyy submit

Change password Log out

Log Out Button

Request leaves from the schedule

An employee could request a leave from his/her own work schedule by right clicking on the target schedule then choose the "Take Leave" button after the request window is popped out.

Customer Nottingham

Customer Nottingham

Grow

Customer London

December 2019

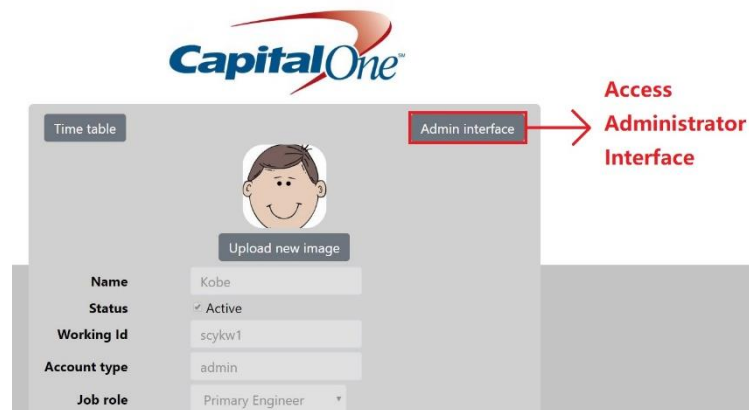
Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
Kobe		Laim				
Gurjyot		Nicole				
Tajin		Xuanhao				
8	9	10	11	12	13	14
Laim		Kobe				
Nicole		Gurjyot				
Xuanhao		Ta				
15	16	17			20	21
Kobe		L				
Gurjyot		N				
Tajin		X				
22	23	24			27	28
Laim		Kobe				
Nicole		Gurjyot				
Xuanhao		Tajin				
29	30	31	1	2	3	4
Kobe		Laim				
Gurjyot		Nicole				
Tajin		Xuanhao				

Take Leave Button

Take leave Edit

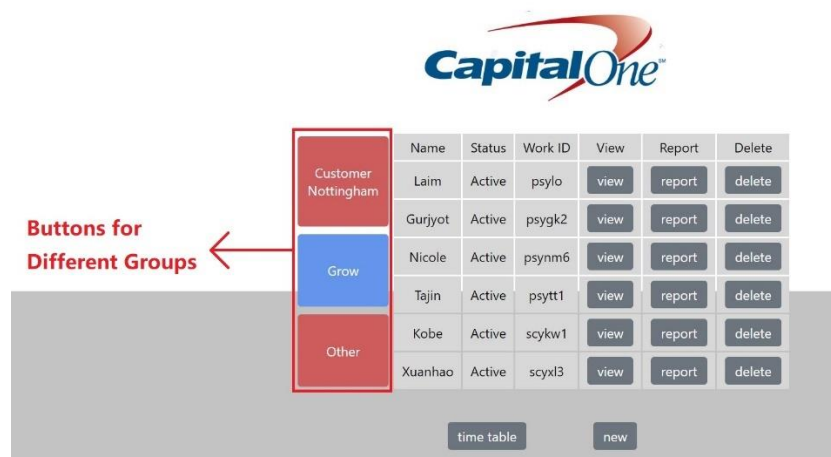
Access administrator interface

If you log in as an administrator, you can access the administrator interface by clicking on the "Admin Interface" button on the top right of your profile.



Switch target group of employee list

The administrator interface will allow you to see all the employees in the first group. To view the employees from different groups, click on the desired group using the buttons on the left side.



Create and delete employees

Administrators can create new employees by clicking the “new” button at the bottom. After clicking the button, a window will pop out to ask the administrator to enter the working id and password of the new employee. The system will automatically check that the working id is not already in use. Administrators can delete an employee by clicking the “delete” button next to the employee to remove.



Jump to another employee's profile

Administrators can access any employee's profile by clicking the “view” button next to the desired employee.

	Name	Status	Work ID	View	Report	Delete
Customer Nottingham	Laim	Active	psylo	view	report	delete
	Gurjot	Active	psyk2	view	report	delete
Grow	Nicole	Active	psym6	view	report	delete
	Tajin	Active	psyt1	view	report	delete
Other	Kobe	Active	scykw1	view	report	delete
	Xuanhao	Active	scyxl3	view	report	delete

time table new

Access All Employees' Profile

Edit the timetable

Administrators can edit the rostering information in the home page by right clicking the schedule blocks and choose “Edit”. An edit window will be displayed. Within this timetable, you could change the block or delete it entirely.

Customer Nottingham

Customer Nottingham
Grow
Customer London

< December 2019 >

Edit Window

Job Role: SecondaryEngineer
Employee: psyk2
Time: 12/10/2019 to 12/16/2019
Repeat: ☒ Enable weekly repeat
Interval: Every 1 weeks
End: ☒ 1 Year (52 weeks)
☐ After 1 times
☐ On date mm/dd/yyyy
Upload Delete

In this window, administrators can allocate an employee to a role by specifying the employee’s working id, how long they will be allocated, and their job role. You can also repeatedly allocate tasks by clicking the “Enable weekly repeat” button. With this functionality, you can assign a repeat tasked using an interval and end date. After clicking the “load” button, this system will automatically check that the employee is available during that interval. And if the target schedule violets some criteria listed in the requirement, a warning message with pop out to interrupt the edition process and inform the user which criteria has been violeted.

TROUBLESHOOTING GUIDE

Error message when I run the “setUpDatabase.php” or “initDatabase.php”

This could be because you have not connected the system to your database. Check that the connectionTest.php files have the correct login information. Once the database has been created once and you run “setUpDatabase” again, an error message will occur as the database already exists. Users should check their database if they still have an error message.

Warning message “Blank space detected” pop-up

This is always caused by inputting an empty block when employees are trying to upload their holiday or deployment schedule. To solve this problem, users simply have to fill the blank space and try to upload the schedule again.

Warning message “No permission” pop-up

This is probably generated from a request towards an account relative action which the user does not have permission to do. For example, if the user is an employee, he/she could not access the administrator interface. Administrators can promote existing employees to administrators if needs be.

Warning message “Failed to send the mail” pop-up

This is probably caused by the firewall of the WLAN you connected to. You could change the firewall setting of the WLAN. Otherwise, you must switch to another WLAN connection. A mobile devices’ hotspots always work.

All the employees have accidentally been deleted or are missing

This could be easily solved by running the “initDatabase.php” file again, which will add them back.

DATA GATHERING AND DATA PROTECTION POLICIES

All data stored in this system belongs to the sponsor of this web application. To protect the data, the system will only give administrators permission to edit all data, and other account types’ user’s permission are limited within a safe range, which only affect his/her personal data.

This web application is using **session** to store visitor’s working id, password and target working id, which points to the target employee. These sessions will be destroyed after visitors log out of this system from the employee interface.

Rostering information is read only for all employees and non-login visitors to prevent the data from being mistakenly edited. Employees are enabled to edit their own profiles and edit holiday schedules and deployment. Administrators are the most powerful and are allowed to manipulate all data within this system.

FREQUENTLY ASKED QUESTIONS**1) Why does the appearance of the web pages change according to the client’s window size?**

This is caused by the CSS code, which takes the percentage instead of pixel to render web pages. If the window’s ratio between width and height is formal, which is 16:9, the web page should be the same. However, different computers with different browsers could generate windows with different sizes. Still, the appearance of this web application should be at least acceptable.

2) Why does the system only recognize “admin” in terms of account type?

Because it is hard and complex to create objects of new classes in JavaScript, this system uses strings to represent employees’ account types. For account type, only “admin” could be recognized as the administrator, and all other account types are recognized as normal employees.

3) How to activate newly created employees’ accounts?

To activate newly created employees’ accounts, administrators only have to access the employees’ interfaces and click on the activate button in their profiles.

4) How could I find my password back if I forget it?

To find back a specific employee’s password could be achieved by directly accessing the account table from the database.