CM2020 Agile Software Projects
Final Report

Title: Cloud HRMS Web application

Team 75

Ng Yong Xin Jesse
(UOL ID: 220253424)

Joel Ang Kai Yu
(UOL ID: 220253413)

Pang Hwee Min, June
(UOL ID: 220253561)

Jacky Wijaya
(UOL ID: 210485763)

Table of Contents

Background

I. Project Introduction

‘The Smurfy Gang’ is a leading SME board game cafe in Singapore. The organisation has approached our team (Team 75) to implement a web application for a transformation to their current Human Resource (HR) operations.

In the beginning of the discussion with the stakeholders, our team has suggested Cloud HRMS web application because cloud is more efficient, reduces costs, is more scalable, more secure and only requires internet connection to view files or documents. It is agreed by the CEO that if the project is successful, he will approach us again for an upgrade or maintenance of the website in the future.

Team 75 consists of four members. Since our application consists of quite a number of web pages, each member is responsible for several web pages, including the back-end functionality.

This report will cover in detail the process of how team 75 has worked together to achieve the final product.

II. Literature and sources of information

In our mid-term report under the “Literature Research and Review” section, we looked at the duties of a HR Department, what is cloud computing and HRMS. Adding on to that, it has been found that as the number of staff increases, the HR-to-employee ratio decreases (Workforce Analytics: A Critical Evaluation, 2015). In layman terms, it means that as more staff join the company, the number of HR staff stays relatively the same, and each HR staff has to take on more responsibilities to manage the growing number of staff.

As mentioned in our mid-term report, in ‘The Smurfy Gang’, the manager is actually the one taking on HR responsibilities since the organisation is relatively small. The CEO of ‘The Smurfy Gang’ has once mentioned that other than the concern over the manager’s turnover rate, one of the reasons he wants to make a transformation to the current HR operations is due to the fact that the business is constantly growing everyday, which means that in time to come, more staff are needed to deal with the workload. The CEO is concerned that with more staff joining the company, they have the choice to either hire more managers, or to improve their HR processes such that each manager has a lighter workload.

Therefore, our team suggested Cloud HRMS web application which provides several benefits such as improving service and access to data, Streamline HR processes and reducing administrative burdens, and many more which we have covered in our mid-term report.

With such benefits, 'The Smurfy Gang' will no doubt be able to thrive even with a lower HR-to-employee ratio as each manager is not overworked due to most of the HR processes being automated. This is the more cost effective way in the long run, rather than hiring more managers to cope with the increasing workload.

In our mid-term report, we also introduced two other existing HRMS applications available in the market; Info-Tech and justlogin. Our final product will incorporate the functions that both of these HRMS applications have. Mainly the ability to manage payroll, leave and claim.

III. Project Aims

Our main aim is to build our application according to what we had laid out in our midterm report, which is to provide a better platform for employees of 'The Smurfy Gang' to clock in and out of their working hours (time-tracking purpose), view monthly payslip and submit leave applications or requests without the hassle of approaching managers.

It will be designed according to the organisation's terms and policies, and contract. The purpose is to get rid of the CEO's concern over the managers turnover rate since this HRMS web application will assist the company such that the managers need not consolidate punch cards and manually calculate payroll for each staff monthly anymore.

We will keep the stakeholders in the loop as we move along to build the web application for them, and to test the functions to see if it works exactly how our team expects it to.

IV. Project Scope

These are the functions that our HRMS web application will have:

Login
The login page is the landing web page of our HRMS application. A valid username and password is required in order to access all the other web pages of the application. This is to ensure user authentication.

Homepage
The homepage will be displayed after a successful login. This page will display the logged in user's name, username, and the time of login.

Register (only admin access)
The purpose of the register page is for the Admin to create new user accounts. This page is only accessible to the "Admin" role to prevent non-staff from creating an account.

Time Tracking
This page allows users to clock in when starting their work shift, and clock out after ending their work shift.

Schedule
This page allows users to schedule their work days by selecting the dates of their preference from the calendar displayed.

Leave Application (View, Add, Edit, Delete)
The leaves page displays all the leave applications submitted by the logged-in user, displayed in table form, along with the status of each leave application.

Users are only able to edit and delete leaves that are of "Pending" or "Saved as Draft" status.

Users are also able to create a new leave application by clicking on the "Add Leave" button. This will direct them to the create leave application page to create a new leave application.

Claims (View, Add, Edit, Delete)
The claims page displays all Claims Application submitted by the logged-in user, displayed in table form, along with the status of each claim.

A claim application can be edited or deleted by selecting a checkbox located on each row of the table, and clicking on the “Edit Claim” or “Delete Claim” button depending on which action the user wants to take.

Users are able to create a new claim application by clicking on the “New claim” button, which will direct them to the “New Claim” page to create a new claim.

Payslip
This page displays the payslip of the user according to the month selected.

Feedback
The feedback page is for users to submit any feed that they have regarding the application.

All Feedback (only admin access)
The all feedback page displays all feedback that has been submitted by users. The user’s name is displayed along with the feedback.

V. Deviations from proposal

In our mid-term proposal, we mentioned that the sign up page which was originally for new staff to create a new account has been changed to be only made accessible to users of “Admin” role to prevent unauthorised users from creating an account. However, we did not take into account that the navigation bar would have to be different for users of “Staff” and “Admin” roles following the change. This is further discussed later in this report.

In the mid-term proposal, we also missed out the “All Feedback” page which allows the Admin to view all the feedback that has been submitted by users.

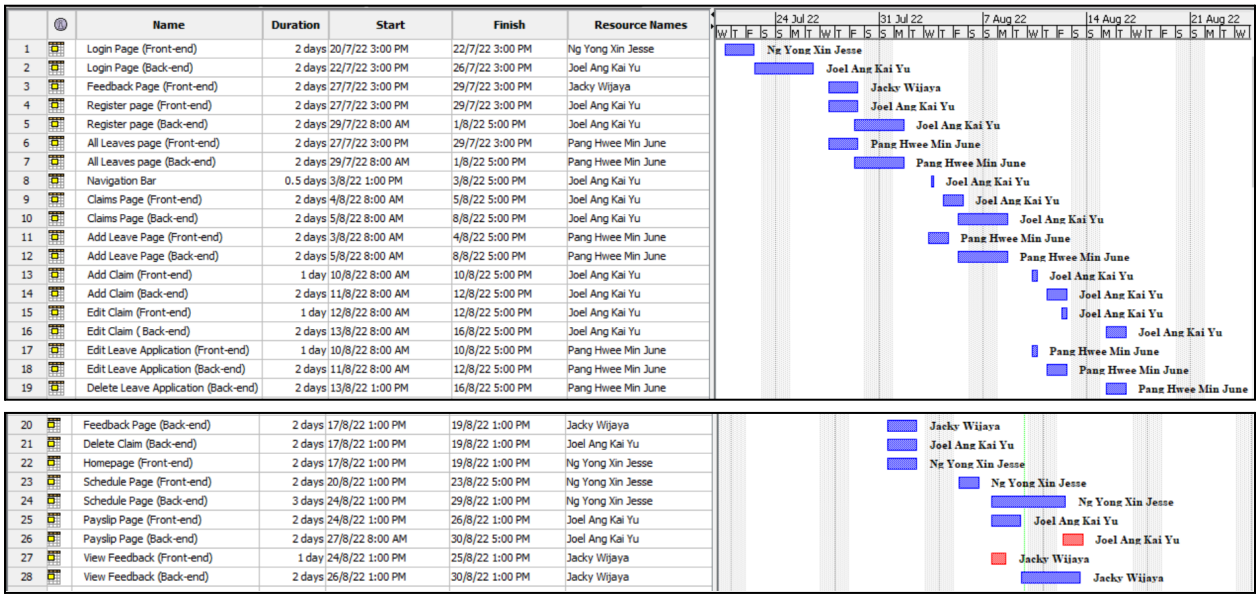
The two deviations above have been implemented in our application.

Planning and Research

The workload has been split such that those who have designed the prototype will work on that function since we know best how it should look and be. The ones who are better at writing the report, will be tasked to do so.

Gantt Chart

The gantt chart (created using ProjectLibre) shows the estimated duration required for each task that is to be completed for this project. The rough timeline allows us to gauge whether we are on schedule.



	Name	Duration	Start	Finish	Resource Names
1	Login Page (Front-end)	2 days	20/7/22 3:00 PM	22/7/22 3:00 PM	Ng Yong Xin Jesse
2	Login Page (Back-end)	2 days	22/7/22 3:00 PM	26/7/22 3:00 PM	Joel Ang Kai Yu
3	Feedback Page (Front-end)	2 days	27/7/22 3:00 PM	29/7/22 3:00 PM	Jacky Wijaya
4	Register page (Front-end)	2 days	27/7/22 3:00 PM	29/7/22 3:00 PM	Joel Ang Kai Yu
5	Register page (Back-end)	2 days	29/7/22 8:00 AM	1/8/22 5:00 PM	Joel Ang Kai Yu
6	All Leaves page (Front-end)	2 days	27/7/22 3:00 PM	29/7/22 3:00 PM	Pang Hwee Min June
7	All Leaves page (Back-end)	2 days	29/7/22 8:00 AM	1/8/22 5:00 PM	Pang Hwee Min June
8	Navigation Bar	0.5 days	3/8/22 1:00 PM	3/8/22 5:00 PM	Joel Ang Kai Yu
9	Claims Page (Front-end)	2 days	4/8/22 8:00 AM	5/8/22 5:00 PM	Joel Ang Kai Yu
10	Claims Page (Back-end)	2 days	5/8/22 8:00 AM	8/8/22 5:00 PM	Joel Ang Kai Yu
11	Add Leave Page (Front-end)	2 days	3/8/22 8:00 AM	4/8/22 5:00 PM	Pang Hwee Min June
12	Add Leave Page (Back-end)	2 days	5/8/22 8:00 AM	8/8/22 5:00 PM	Pang Hwee Min June
13	Add Claim (Front-end)	1 day	10/8/22 8:00 AM	10/8/22 5:00 PM	Joel Ang Kai Yu
14	Add Claim (Back-end)	2 days	11/8/22 8:00 AM	12/8/22 5:00 PM	Joel Ang Kai Yu
15	Edit Claim (Front-end)	1 day	12/8/22 8:00 AM	12/8/22 5:00 PM	Joel Ang Kai Yu
16	Edit Claim (Back-end)	2 days	13/8/22 8:00 AM	16/8/22 5:00 PM	Joel Ang Kai Yu
17	Edit Leave Application (Front-end)	1 day	10/8/22 8:00 AM	10/8/22 5:00 PM	Pang Hwee Min June
18	Edit Leave Application (Back-end)	2 days	11/8/22 8:00 AM	12/8/22 5:00 PM	Pang Hwee Min June
19	Delete Leave Application (Back-end)	2 days	13/8/22 1:00 PM	16/8/22 5:00 PM	Pang Hwee Min June
20	Feedback Page (Back-end)	2 days	17/8/22 1:00 PM	19/8/22 1:00 PM	Jacky Wijaya
21	Delete Claim (Back-end)	2 days	17/8/22 1:00 PM	19/8/22 1:00 PM	Joel Ang Kai Yu
22	Homepage (Front-end)	2 days	17/8/22 1:00 PM	19/8/22 1:00 PM	Ng Yong Xin Jesse
23	Schedule Page (Front-end)	2 days	20/8/22 1:00 PM	23/8/22 5:00 PM	Ng Yong Xin Jesse
24	Schedule Page (Back-end)	3 days	24/8/22 1:00 PM	29/8/22 1:00 PM	Ng Yong Xin Jesse
25	Payslip Page (Front-end)	2 days	24/8/22 1:00 PM	26/8/22 1:00 PM	Joel Ang Kai Yu
26	Payslip Page (Back-end)	2 days	27/8/22 8:00 AM	30/8/22 5:00 PM	Joel Ang Kai Yu
27	View Feedback (Front-end)	1 day	24/8/22 1:00 PM	25/8/22 1:00 PM	Jacky Wijaya
28	View Feedback (Back-end)	2 days	26/8/22 1:00 PM	30/8/22 1:00 PM	Jacky Wijaya

Milestones

Date	Task	Done by
13 - 19 July 2022	Create GitHub Repository	Joel Ang Kai Yu
	Base code added into GitHub repository	
	Write SQL statements to create database tables	Pang Hwee Min June
20 - 26 July 2022	Login Page (Front-end)	Ng Yong Xin Jesse
	Login Authentication (Back-end)	Joel Ang Kai Yu
27 July - 2 Aug 2022	Feedback Page (Front-end)	Jacky Wijaya
	Register page (Front-end & Back-end)	Joel Ang Kai Yu
	All Leaves page (Front-end & Back-end)	Pang Hwee Min June
3 Aug - 9 Aug 2022	Navigation Bar	Joel Ang Kai Yu
	Claims page (Front-end & Back-end)	
	Add Leave Page (Front-end & Back-end)	Pang Hwee Min June
10 Aug - 16 Aug 2022	Add Claim (Front-end & Back-end)	Joel Ang Kai Yu
	Edit Claim (Front-end & Back-end)	
	Edit Leave Application (Front-end & Back-end)	Pang Hwee Min June
	Delete Leave Application (Front-end & Back-end)	
17 Aug - 23 Aug 2022	Feedback Page (Back-end)	Jacky Wijaya
	Delete Claim (Front-end & Back-end)	Joel Ang Kai Yu
	Homepage (Front-end)	Ng Yong Xin Jesse
	Schedule Page (Front-end)	
24 Aug - 30 Aug 2022	Payslip Page (Front-end & Back-end)	Joel Ang Kai Yu
	View All Feedback (Front-end & Back-end)	Jacky Wijaya
	Carry out unit testing	All
31 Aug - 4 Sept 2022	Carry out system testing	All

Product Backlog (Kanban Board, JIRA Software)

Our team utilised the Kanban Board (in JIRA) as our product backlog throughout the whole process of building the HRMS web application.

A Kanban board is useful for keeping track of what tasks need to be done, what has been done and what should be marked the highest to lowest priority. Tasks can also be allocated to team members.

In our Kanban board, tasks are listed such that it is in line with the prototype that we had designed previously in our mid-term proposal.

The “To Do” section of the Kanban board consists of all the tasks that are to be done. During each scrum meeting, the Kanban board is updated according to what the team has agreed on. If all members agree that a task can be marked as completed, the task is moved to the “Done” column of the Kanban board. Afterwards, the team will discuss which task to work on next, and that task is moved to the “In Progress” section of the Kanban board. This is done continuously until there are no more tasks left in the “To Do” section.

One mistake we made initially was to not split each function into smaller tasks. This makes it difficult and not as beneficial to keep track of the team member’s progress. Splitting it into subtasks makes everything clearer, and takes us more in depth on what needs to be done, instead of a broad term of each function. With each user story becoming smaller, we can then separate them into different sprints and tasks more accurately.

So instead of having “Leave application” as one of the user stories, we separated it into “view leave”, “add leave”, “edit leave” and “delete leave”.

“View leave” and “add leave” can be done in one sprint, while “edit leave” and “delete leave” can be done in another sprint.

Scrum meeting summary

A daily scrum requires a short meeting of 15 minutes or less on every work day. However, since we are not a large scale project requiring daily updates, we decided to hold a scrum meeting every Wednesday for at least an hour instead.

Every Wednesday, our team will meet in school to update each other on the things we have completed since the last scrum meeting, share difficulties faced, as well as decide on the deliverables for the next scrum meeting. In the weeks where the team was not able to meet face-to-face, we hosted the scrum review meeting over discord.

The following pages show summaries of scrum meetings we have had, along with screenshots of the Kankan board showing the tasks that are “In Progress” and “Done” after each meeting. We have also highlighted the issues and difficulties faced during the process since not every journey is a smooth sailing one, together with a team reflection after each meeting.

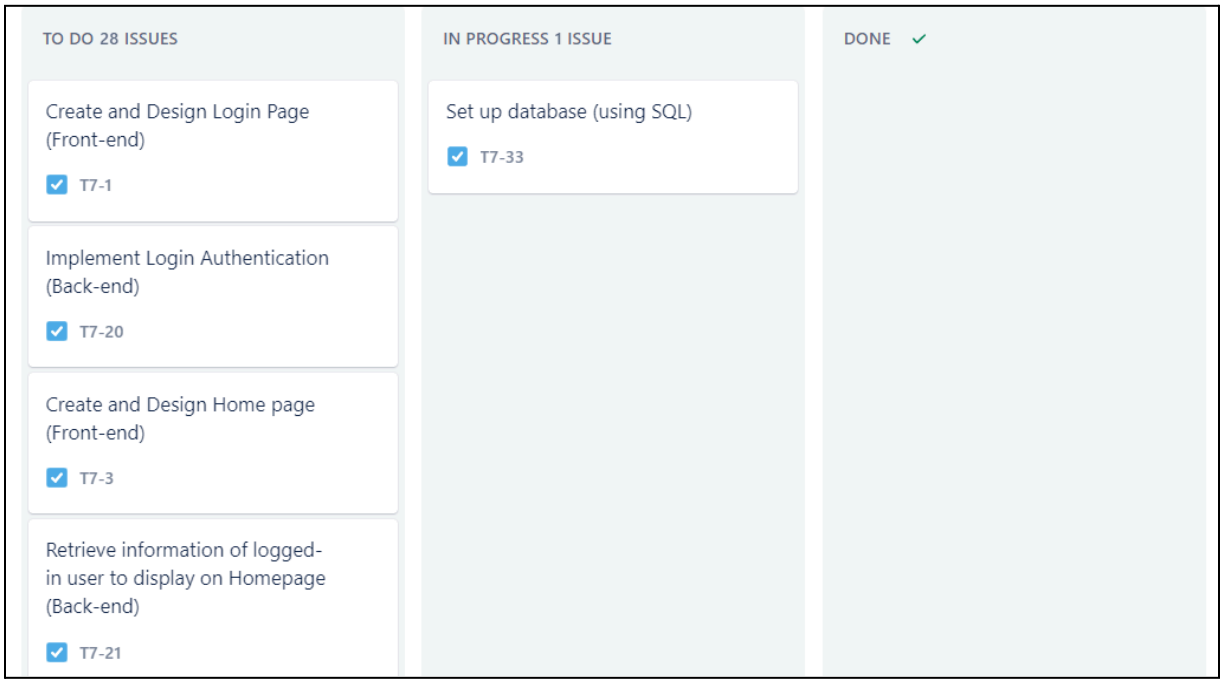
The following scrum meetings are only summaries of each meeting. Fully detailed scrum meeting logs can be found under **Appendix A**.

Scrum meeting 1 (13 July 2022)

In this meeting, the team agreed on the programming languages and frameworks we will use to create the application.

Deliverables

- Create GitHub repository
- Write SQL statements to create database tables



(Snippet of Kanban board as of 13 July 2022)

Issues faced:

There was a minor conflict as some members wanted to use Visual Dot Net to build the web application while some wanted to use JavaScript. The team leader stepped in and suggested using Javascript, Node.js and Express.js since all of us have taken a module that teaches those languages and framework, and the team agreed.

Reflection of team:

The team leader should be strict about conveying our ideas across in a conductive feedback manner to prevent conflicts that might hinder our progress.

Scrum meeting 2 (20 July 2022)

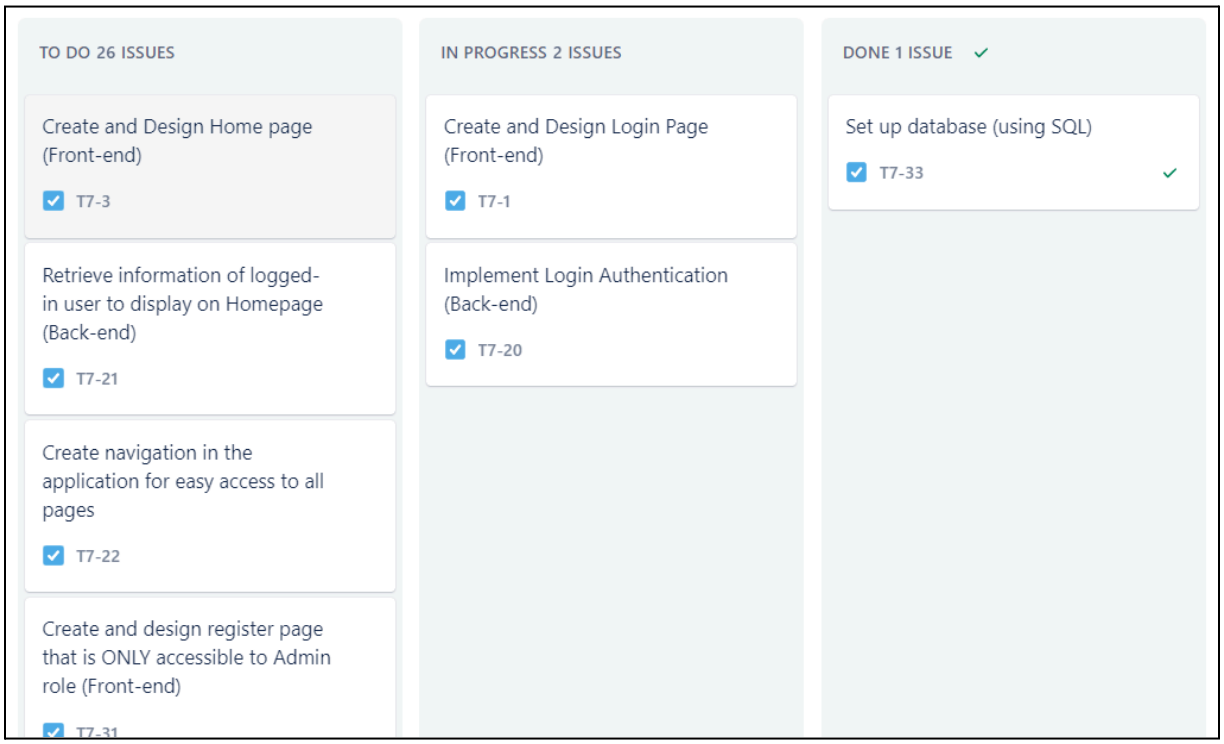
Tasks completed:

1. GitHub Repository created with base code added.
2. SQL statements to create the database tables written.

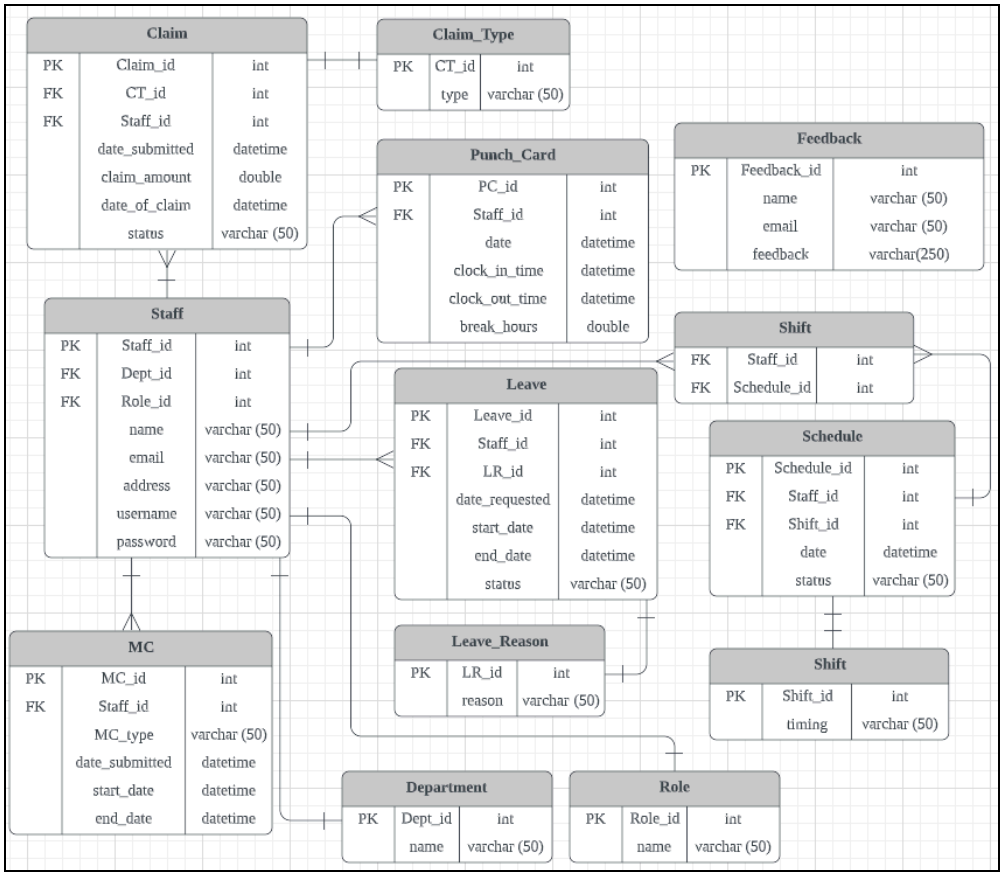
The team agreed to work on the login functionality before diving into other functionalities.

Deliverables:

- Create and Design Login Page
- Implement Login Authentication (server-side functionality)
- Start working on Final Report



(Snippet of Kanban Board as of 20 July 2022)



(ERD Diagram)

Issues faced:

With 13 tables in our ERD diagram, we faced difficulties writing SQL statements that would correctly reflect the relationship between each table using Foreign Keys, especially when there are more than 1 foreign key in a single table (E.g. Schedule table). This was resolved by using the SQL FOREIGN KEY Constraint to specify the relationship between each table.

We also had to ensure that the tables were created in the correct order to prevent getting the error MySQL Error Code: 1824. Failed to open the referenced table. For example, “Dept_id” is a foreign key in the “Staff table”. Therefore, the “Department” table should be created before the “Staff” table can be created.

Reflection of team:

We learnt that we should never rush with the creation of the database tables and that the order of the creation is critical if there is a foregin key involved. This serves as a great reminder to look closely and carefully at our database table for the relationship between tables before creating the tables.

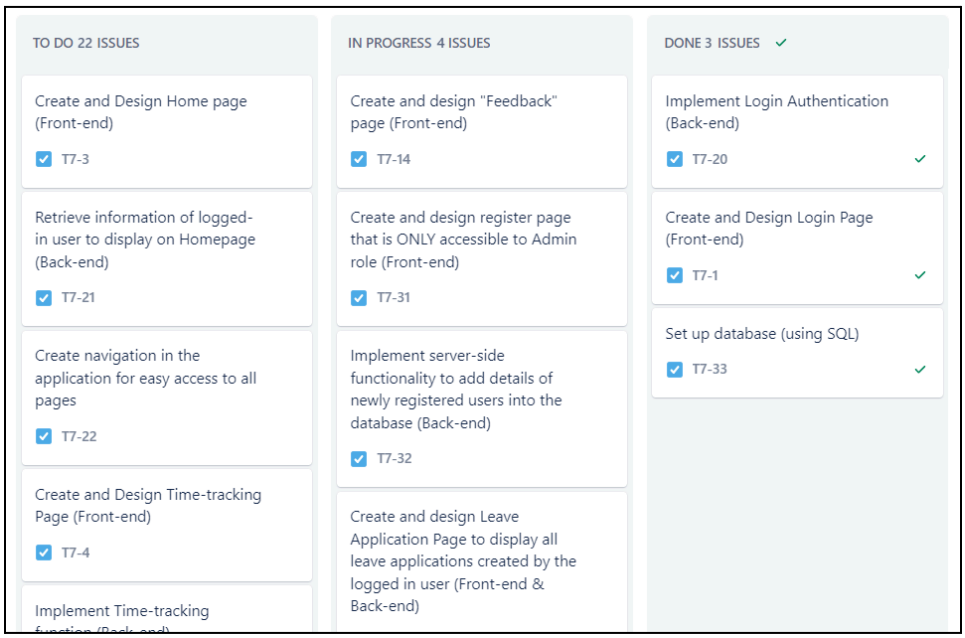
Scrum meeting 3 (27 July 2022)

Tasks Completed:

1. Login page created with HTML (front-end)
2. Login authentication implemented (back-end)

Deliverables:

- Register Page (Admin Only)
- Leaves (front & back-end)
- Feedback (front-end)



(Snippet of Kanban Board as of 27 July 2022)

Issues faced:

Faced issues with creating sessions on the server side whenever a login request is made, as well as working with cookies used for storing each session's unique id. These difficulties were due to the team not having experience working with sessions and cookies in Node.js. We overcame it by doing readings online and watching tutorials on how everything should come together, and we managed to implement it successfully.

Reflection of team:

It is okay to not have experience, but the team must have the will to learn how to overcome obstacles together in the application building process. If we did not make the effort to look for our best friend, Google, for further assistance, these tasks will be marked under 'In progress' of our Kanban board until we resolve it, and tasks will just continue to pile up.

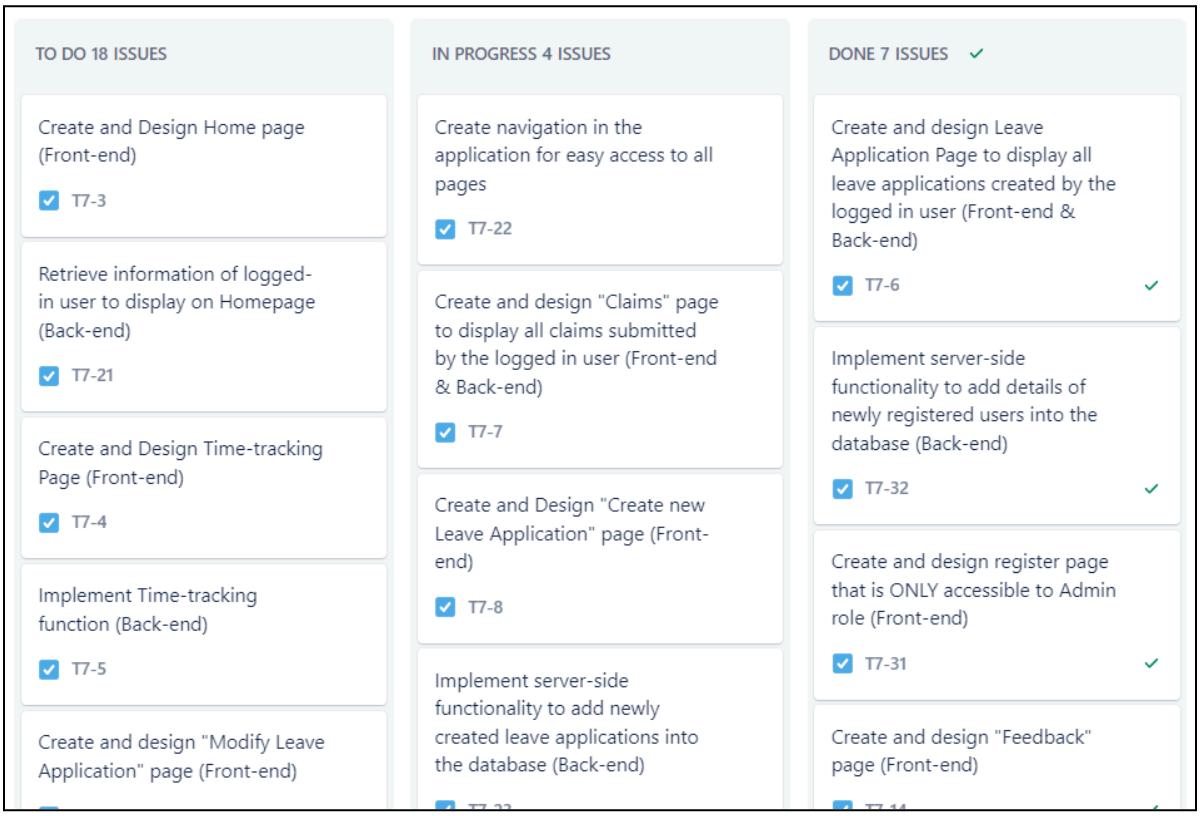
Scrum meeting 4 (3 August 2022)

Tasks Completed:

1. Register web page (Client-Server side)
2. 'All Leave' web page (Client-Server side)
3. Feedback Form (Client-side)

Deliverables:

- Navigation Bar (client-server side)
- Claims (client-server side)
- Add Leave (client-server side)



(Snippet of Kanban Board as of 3 August 2022)

Issues faced:
We faced an issue with restricting access to only users with the “Admin” role. This means that all users were able to access the “register’ page. This was resolved by doing a check on the server side. Such that if the role of the logged in user is “Admin”, then the “register” web page would be rendered. Otherwise, the user will be redirected back to the homepage.

Reflection of team:
With many cyber criminals still around, we have to look out for the security measures we could introduce and implement to prevent the company from being the next victim while using our web application. This is to ensure our web application is a safe platform for our users.

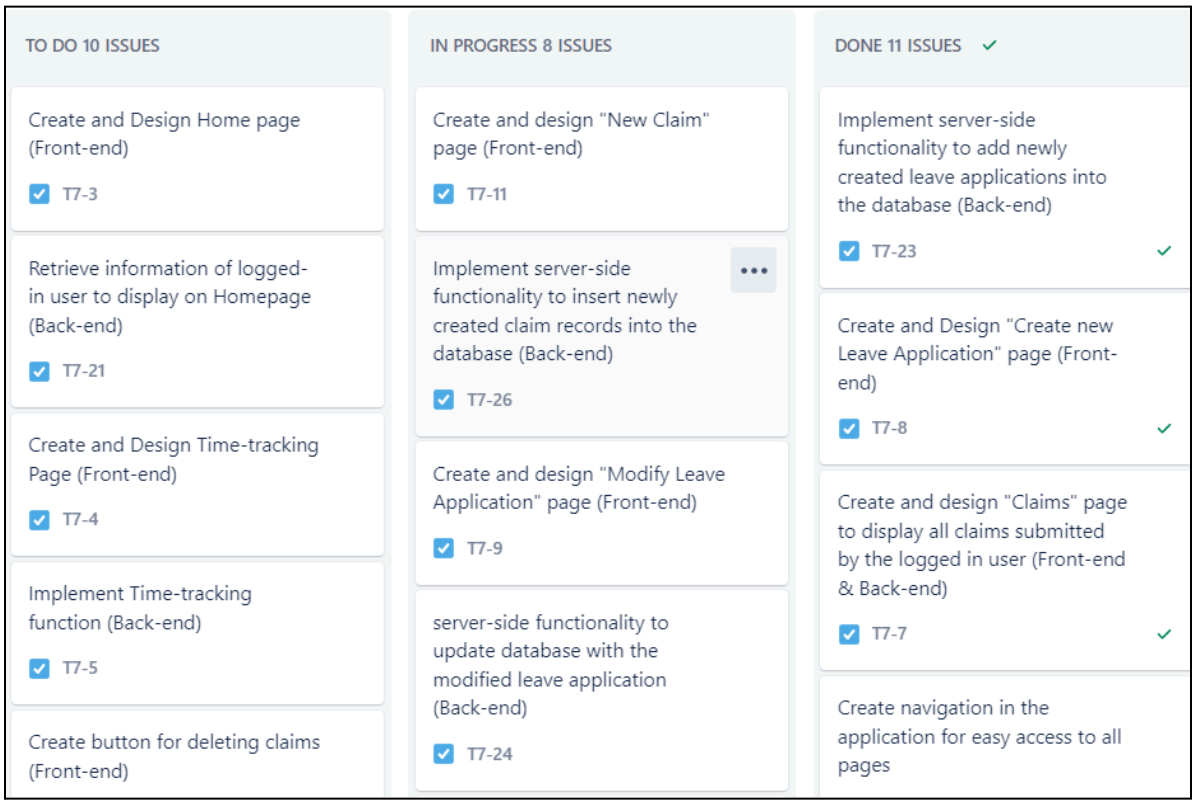
Scrum meeting 5 (10 August 2022)

Tasks Completed:

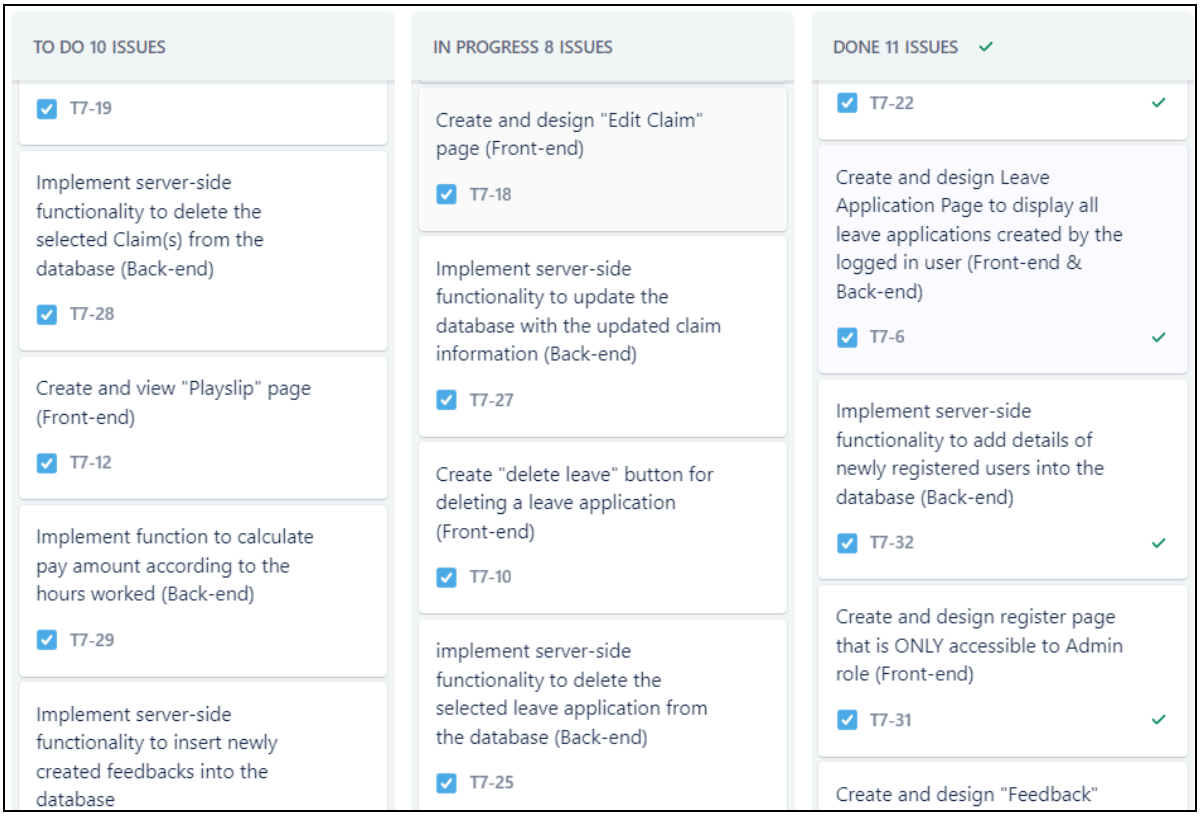
1. Navigation Bar
2. “Claims” Page (client-server side)
3. “Add Leave” Page (client-server side)

Deliverables:

- Add Claim (client-server side)
- Edit Claim (client-server side) *Only if time permits*
- Edit Leave (client-server side)
- Delete Leave (client-server side) *Only if time permits*



(Snippet of Kanban Board as of 10 August 2022) (1)



(Snippet of Kanban Board as of 10 August 2022) (2)

Issues faced:
Faced an issue with auto-filling the “name of applicant” field with the name of the current logged-in user in the “Add Leave” page. This was resolved by retrieving the name of the user on the server side by running a SELECT query, and passing it into the html form on the “Add Leave” page upon rendering the page.

Reflection of team:
We should go beyond what we can if we have the time. After all, the faster we finish the current sprint, the more sprint we can do and better satisfy our stakeholders with a higher value product of what they want.

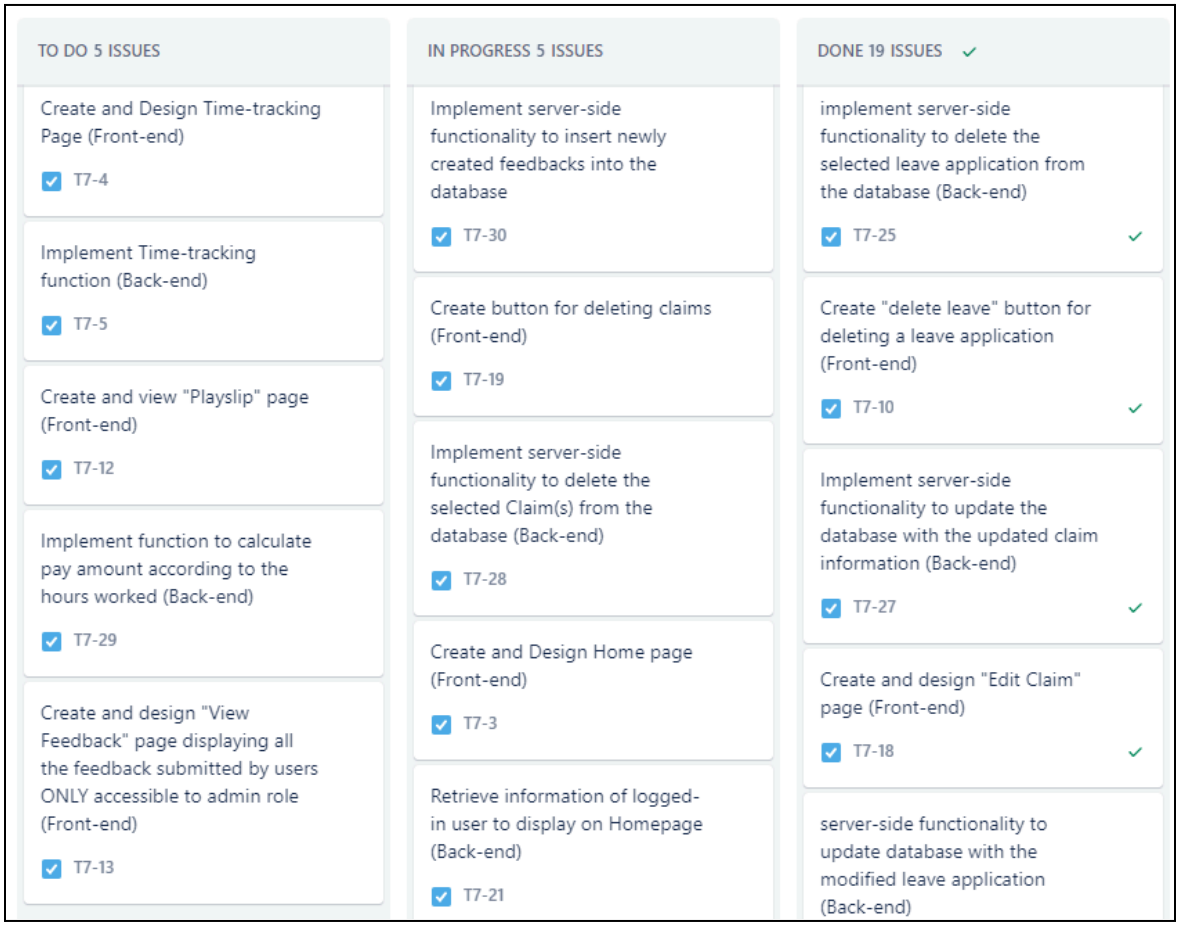
Scrum meeting 6 (17 August 2022)

Tasks Completed:

1. ‘Add Claim’ web page (client-server side)
2. ‘Edit Claim’ web page (client-server side)
3. ‘Edit Leave’ web page (client-server side)
4. ‘Delete Leave’ web page (client-server side)

Deliverables for the next meeting:

1. Delete Claim (client-server side)
2. Homepage (client-server side)
3. Feedback form (server-side)
4. Schedule Page (client-side)



(Snippet of Kanban Board as of 17 August 2022)

Issues faced:
One member faced an issue with displaying images in the homepage. As the images are saved in the computer locally, she had trouble using the “img src<>” tag in the html file for the CSS styling. This was resolved by creating a public folder to store the image, and specifying that directory path where the image was stored in, in the main.js file. Using this workaround, we were able to display the image using the “img src<>” tag.

Reflection of team:
If any of us face an issue with our codes, we are not afraid to speak up and seek help from one another. This way, we learnt as a team as we collaborated closely. With our combined efforts, it allows us to each bring our skills, talents and experiences towards the same scrum goal quicker.

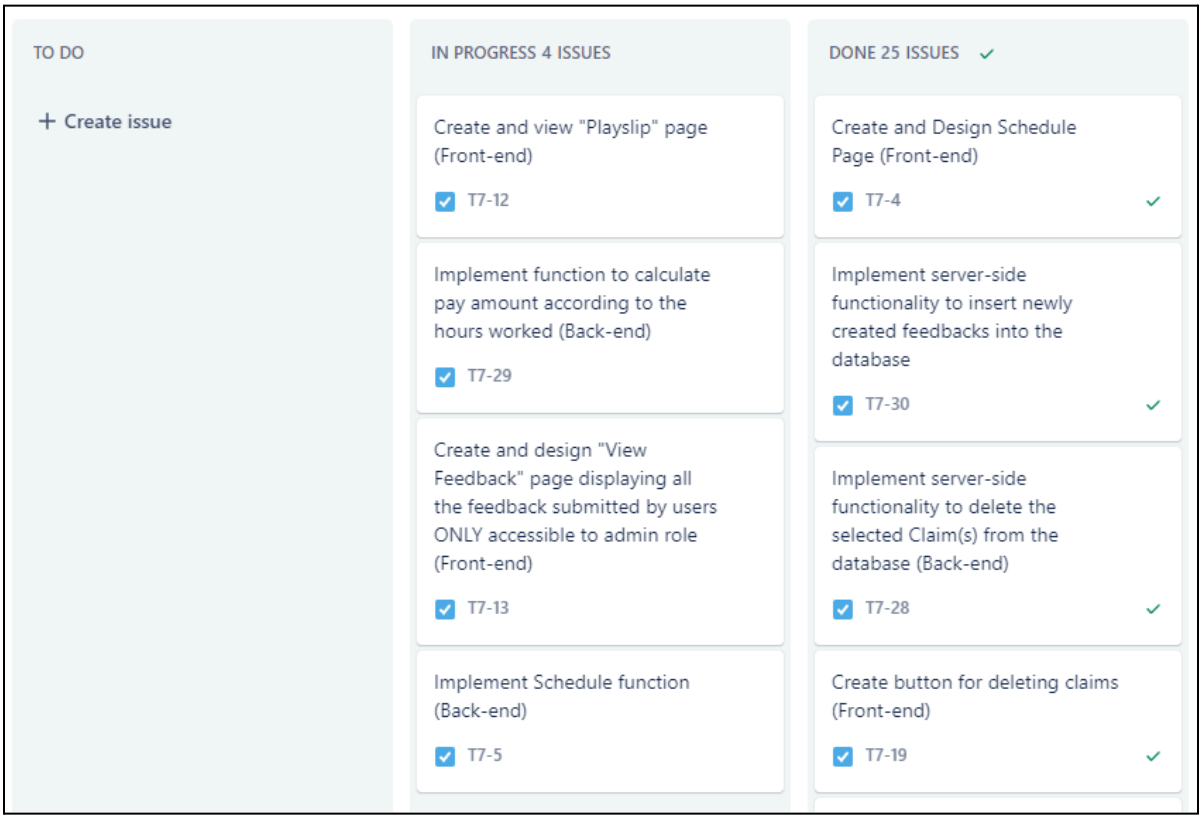
Scrum meeting 7 (24 August 2022)

Tasks Completed:

1. Feedback (Server-side)
2. ‘Delete Claim’ page (Client-server side)
3. ‘Homepage’’ page (Client-server side)
4. ‘Schedule’’ page (Client-server side)

Deliverables for the next meeting:

1. Payslip Page (Client-server side)
2. View Feedback for ADMIN (Client-server side)
3. Unit testing



(Snippet of Kanban Board as of 24 August 2022)

Issues faced:
While trying to display the user's time to login on the homepage, we realised that our database did not have a field storing the user's time of login. After this issue was brought up by the member assigned to this task, another member suggested that we store the login time when creating a session upon a successful login. That way, we can display it in the front-end. The member assigned to the task took up the suggestion and was able to implement it successfully.

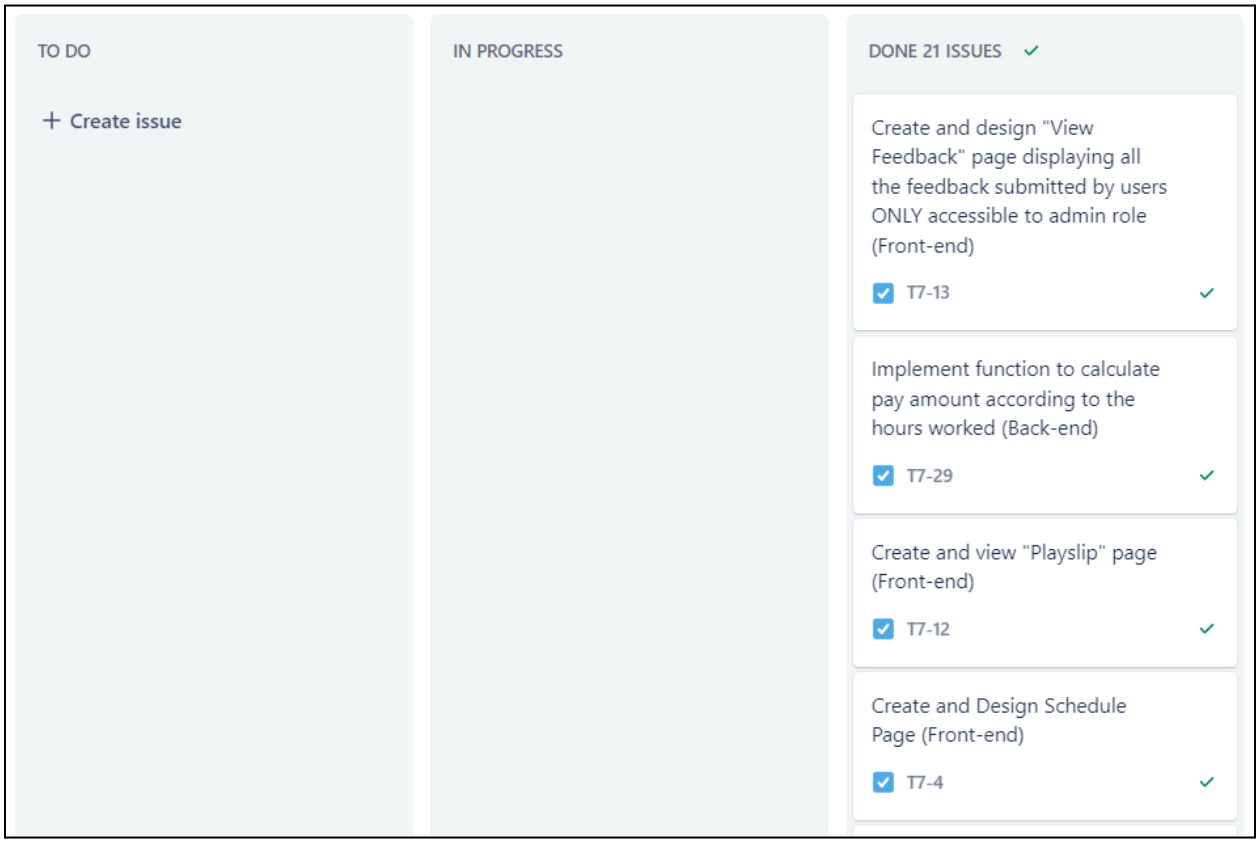
Reflection of team:
It is important to value the feedback from one another as it will improve our workflow and allow us to solve challenges as a team.

Scrum meeting 8 (31 August 2022)

Tasks Completed:

- Payslip Page (Client-server side)
- View Feedback for ADMIN (Client-server side)
- Unit testing.

In this meeting, the team has confirmed that all of the tasks that are needed to be worked on have all been completed. With that, each member is tasked to do unit testing on the components that we have implemented.



(Snippet of Kanban Board as of 31 August 2022)

Issues faced:
Faces issues in calculating the net pay in the “Payslip” page. This is because there were several factors which contribute to the net pay. For example, the overtime pay and CPF contribution amount. This was resolved by creating several JavaScript functions to manage those calculations.

Reflection of team:
This step (unit testing) is very important before the integration stage because all our codes represent a function in the application. Combining everything together without checking individually might cause a bunch of errors that we need to debug, which is time-consuming and inefficient, and our team would like to avoid that.

Scrum meeting 9 (3 September 2022) - Final meeting

In this meeting, the team updated each other on the results of the unit testing. After which, we integrated our codes together to see if all functionality and components worked as intended. Only then, our team can carry out system testing.

As we want the best for ‘The Smurfy Gang’, our team wants to touch up on our codes with proper code formatting before finalising the final report with the remaining time we have left.

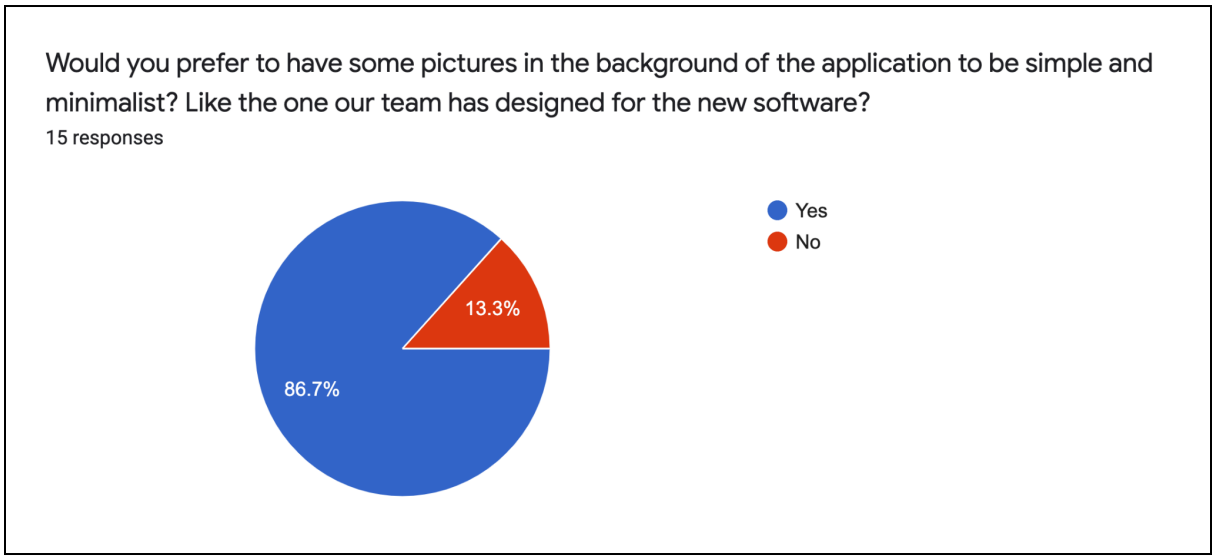
Reflection of team:
It is important to do a final test after the integration stage to ensure everything is working smoothly before the sprint review. Otherwise, we will be wasting the stakeholders’ precious time when we discover issues with our codes, and cannot proceed to the deployment stage.

We also have to carry out the system testing as a team. Otherwise, we will be approaching one another to ask what is the expected result because we do not understand what was written in the test case. Only the person who designed the prototype and built the function knows it best. Therefore, for a more accurate result, everyone should come together and agree if it works according to how we want it to be.

Prototyping and Iteration

Before the team started on the coding aspect of this HRMS application, we created prototypes of 3 different fidelity levels. Low-fidelity, medium-fidelity, and high-fidelity, which we have shared in our mid-term report.

Our team had also created a google form to gather feedback on our high-fidelity prototype, in which we also shared in our mid-term report. Based on the survey responses, the majority of respondents prefer a simple and minimalist theme so that it will be user friendly and easy to use even for users who are not tech-savvy.



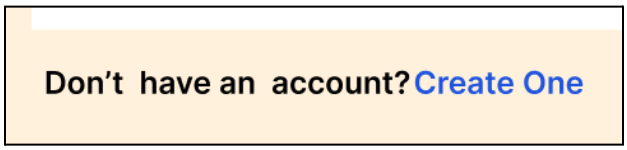
(Google form response)

Therefore, one of our main focus in terms of the client-side aspect is to make the user interface simple and easy to navigate. For instance, our login page has a simple bisque coloured background with 2 input boxes with the text holders “Username” and “Password” to tell users where they should key in each input. There are no additional components other than a submit button and a ‘remember me’ checkbox which are all clearly labelled and will not confuse any users.

Since our team initially designed the prototype with a “minimalist” theme in mind, we agreed to not deviate too much from our high-fidelity prototype in our actual application.

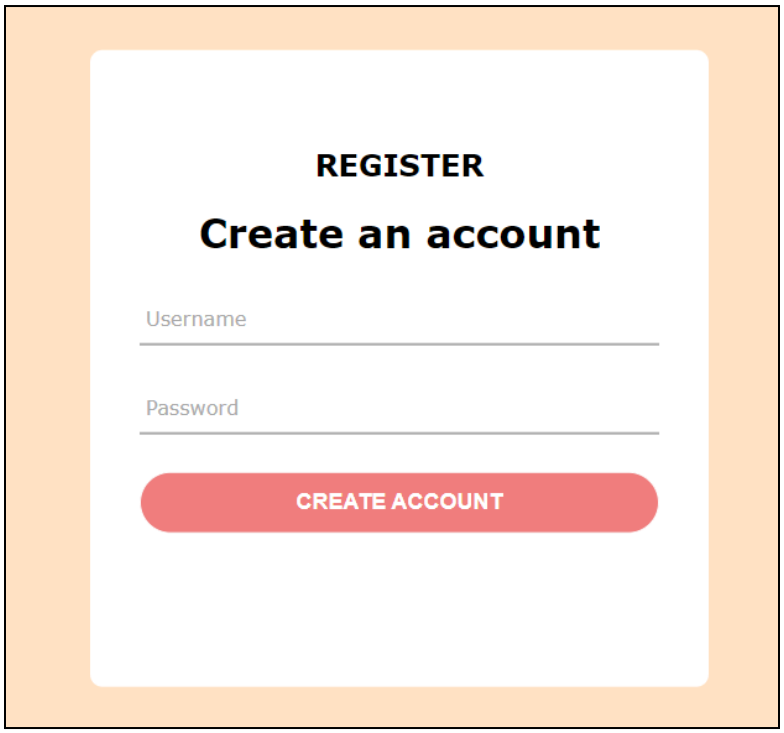
Deviation in prototype from proposal

In the prototype shown in our mid-term report, a link (see Figure A below) was added to the high-fidelity prototype of the login page which directs users to the signup page to create their own account. This posed a risk in having non-staff creating an account to access the application. After a discussion with the stakeholders, we decided to forego it and instead add a “register” page which will only be accessible to users with the “Admin” role.

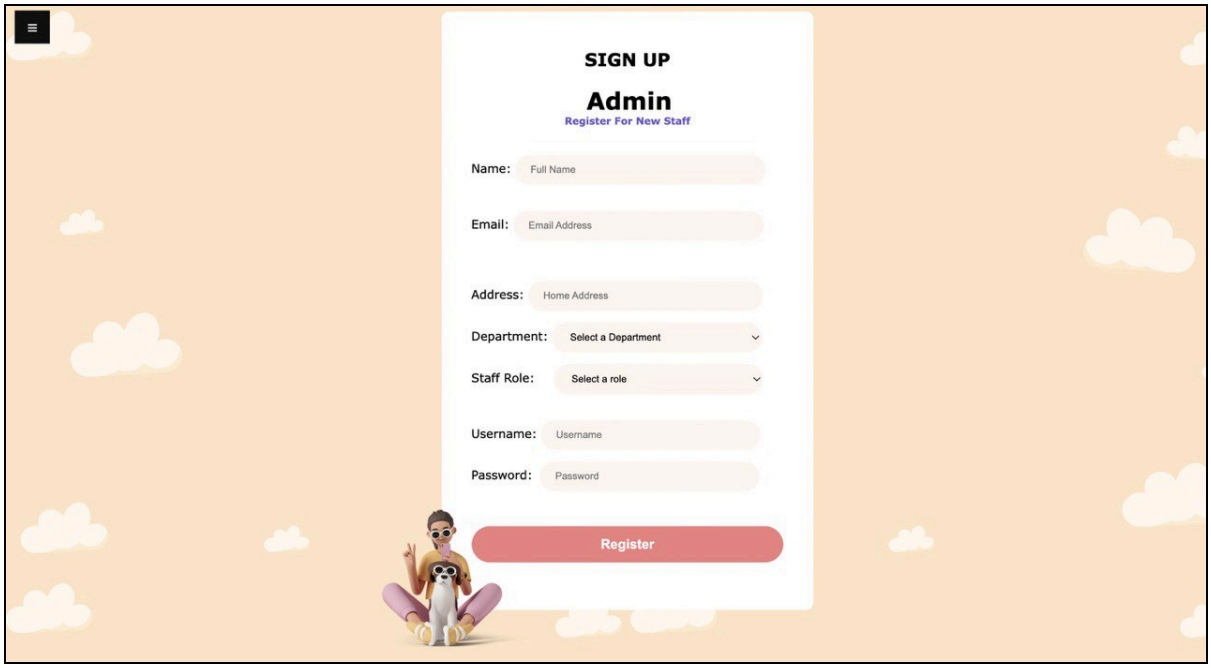


(Figure A: Taken from high-fidelity prototype of login page)

Figure B below shows the high-fidelity prototype of the register page which we have just designed. The low fidelity and medium fidelity prototype of the register page can be found under Appendix B.



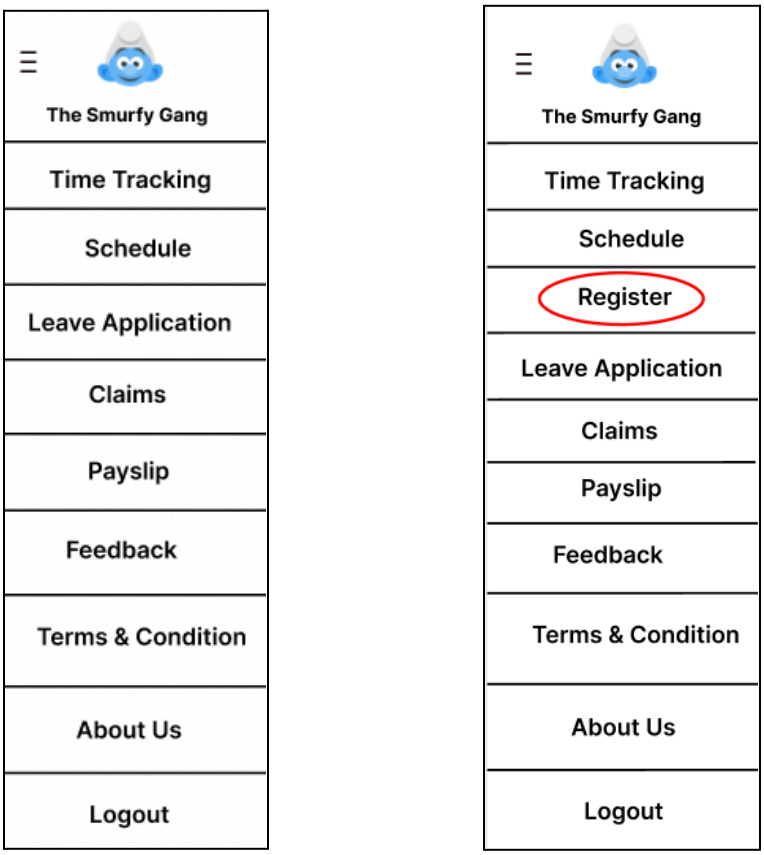
(Figure B: High-fidelity of Register Page (using Figma))



(Figure C: Actual product based on prototype)

With the addition of the register page that is only accessible to users of the “Admin” role, the side-navigation bar for users of the “Admin” role has to be different from the one of a regular staff. Therefore, we created another prototype for the navigation bar of the “Admin” role which reflects the “register” tab.

The screen capture on the left below shows the **high-fidelity prototype of the side navigation bar for users of “Staff” role**, while the one on the right shows the **high-fidelity prototype of the side navigation bar for users of “Admin” role** with a “register” tab added.



(High-fidelity prototype of side navigation bar (using Figma))

Unit Testing

Unit testing is an important process when developing an application. In unit testing, the different units of code are being tested against a set of tests to ensure that the code written is stable and works as we expect it to.

Below shows the unit tests for each component of our HRMS application. The “description” column shows a brief description of each test case, the “expected result” column shows what is the expected output. If the actual output matches the expected output, the status is a “pass”, and “fail” otherwise.

Unit Test for Login

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
1.1	Only registered users can login to the web application.	1. Key in a valid username 2. Key in a valid password 3. Click on the “Login” button	The user is directed to the homepage upon successful login.	Pass	-
1.2	Entering an invalid username and/or password should redisplay the login page	1. Key in an invalid username and/or password 2. Click on the “Login” button	The login page is redisplayed.	Pass	-
1.3	Clicking on “Login” without inputting username and/or password should display a prompt for the user to provide an input.	1. Click on login button	The error prompt is displayed with the text “Please fill out this field”.	Pass	-

Unit Test for Logout

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
2.1	Logged-in users can log out of the web application.	1. Open up the side menu by clicking on the menu icon. 2. Click on the 'log out' tab.	The user is directed to the sign in page.	Pass	-

Unit Test for Register (Admin)

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
3.1	Users who logged in with an "Admin" role account are able to view the "Register" tab in the side menu and the "Register" page.	1. Login with an "Admin" role account 2. Click on the menu icon on the top left to bring out the side menu 3. Click on the "Register" tab	The register page should be displayed when the user clicks on the "Register" tab on the side menu.	Pass	-
3.2	Users who logged in with a non-admin role should not see the "Register" tab in the side menu	1. Login with a "Staff" role account 2. Click on the menu icon on the top left to bring out the side menu.	The user should not be able to see the "Register" tab in the side menu.	Pass	-

3.3	Admin is able to register a new user after filling up the required fields.	1. Fill in the “Name”, “Email”, “Address”, “Department”, “Role”. “Username” and “Password” fields. 2. Click on “register”	The success page will be displayed.	Pass	-
3.4	Admin is only able to register users after filling up all the required fields.	1. Click on submit without filling up any of the fields.	An error box is displayed with the text “Please fill out this field”.	Pass	-

Unit Test for Homepage

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
4.1	Users can view the home page.	1. Login with a valid username and password. OR 1. Click on the "Home" tab on the side navigation bar.	The homepage is displayed with the user's full name, username and the user's time of login.	Pass	-
4.2	Users can navigate to the company's website.	1. Click on 'Company website' on the homepage.	Users are directed to 'https://www.sim.edu.sg/'.	Pass	-
4.3	Users can navigate to other web pages by clicking on the side-navigation bar.	1. Click on the menu icon on the top left on any page. 2. Click on any of the tabs.	Users are directed to the web page of whichever tab they have pressed.	Pass	-

Unit Test for Time Tracking Page

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
5.1	Users can clock in to start their shift when they arrive at work.	1.Select staff name from the dropdown list and click on 'clock in'	The user's clock in time will be recorded in the database. There will be a pop up message that informs the user he has successfully clocked in for his shift.	Pass	-
5.2	Users can clock in to start their shift when they arrive at work.	1.Select staff name from the dropdown list and click on 'clock out'	The user's clock out time will be recorded in the database. There will be a pop up message that informs the user he has successfully clocked in for his shift.	Pass	-

Unit Test for Schedule

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
6.1	Users can select their availability date by navigating to the desired month. (This is a draft requested by the stakeholders as claims, register of new staff and leave application are their main priority. This full functionality has been put on hold for future development).	1.Click on the year or month.	Users can navigate between the months or year. They can only change the web page to light or dark mode depending on their preferences.	Pass	-

Unit Test for Leave

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
7.1	Users can view the leave page that displays all the leave applications they have submitted.	1. After logging in, click on the menu icon to bring up the side navigation bar, and click on the “Leave” tab.	The “All Leaves” page is displayed. The table in the All Leaves page shows all leave applications submitted	Pass	The table should be empty if the user has not submitted any leave applications .

Unit Test for Add Leave

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
8.1	Users can access the "Create Leave" page.	1. On the "All Leaves" page, click on the "Add Leave" button.	The "Create Leave Application" page is displayed.	Pass	-
8.2	"Name of Applicant" and "Request Date" fields should be automatically populated.	1. On the "All Leaves" page, click on the "Add Leave" button.	The "Name of Applicant" field is populated with the user's name, while the "Request Date" field should be populated with the current date.	Pass	

8.3	The “Expected no. of days” field should be calculated automatically.	1. On the “Create Leave” page, fill up the Start date and End date field.	The “Expected no. of days” automatically shows a count based on the number of days between the start and end date selected.	Pass	Users will not be able to choose a date earlier than the start date in the end date field.
8.4	Users are able to create a new leave application.	1. Fill in the required fields in the “Create Leave application” page. 2. Click on the “Submit” button.	Users are directed back to the “All Leaves” page, with the newly created leave application displayed in the table.	Pass	Newly created leave applications are set to “Pending” status.
8.5	Clicking on “Submit” while leaving any of the fields blank should show a prompt for the user to fill up the field.	1. On the “Create Leave Application” page, leave any of the fields empty. 2. Click on the submit button	An error box with the text “Please fill out this field” is displayed.	Pass	-
8.6	Users are able to save their leave application as a draft.	1. Click on the “Save as Draft” button.	Users are directed back to the “All Leaves” page with the leave application added into the table with the status of “Saved as Draft”	Pass	-

8.7	Users are able to cancel their leave application halfway through creation	1. Click on the "Cancel" button.	Users are redirected back to the "All Leaves" page.	Pass	-

Unit Test for Edit Leave

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
9.1	The "edit" button should be visible for leave applications that are of "Pending" and "Saved as Draft" status.	1. Navigate to the "All Leaves" Page.	Beside the "status" column, the "edit" button should be visible if the status is "Pending" or "Saved as Draft".	Pass	Status that is not "Pending" or "Saved as Draft" will show a dash instead of the edit button.
9.2	Users are able to view the "Edit Leave Application" page.	1. On the "All Leaves" page, click on the "edit" button.	The "Edit Leave Application" page is displayed with form data pre-populated with data saved in the database.	Pass	-
9.3	Clicking on "Submit" while leaving any of the fields blank should show a prompt for the user to fill up the field.	1. On the "Edit Leave Application" page, remove input from any of the fields. 2. Click on the submit button	An error box with the text "Please fill out this field" is displayed.	Pass	-

9.4	The "Expected no. of days" field should be calculated automatically.	1. On the "Edit Leave Application" page, choose a Start date and End date.	The "Expected no. of days" automatically shows a count based on the number of days between the start and end date selected.	Pass	Users will not be able to choose a date earlier than the start date in the end date field.
9.5	Users are able to modify the leave application.	1. On the "Edit Leave Application" page, modify any of the fields. 2. Click on the "Update" button.	Users are redirected back to the "All Leaves" page with the table reflecting the new changes made.	Pass	The "Request Date" field is non-modifiable.

Unit Test for Delete Leave

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
10.1	The "delete" button should be visible for leave applications that are of "Pending" and "Saved as Draft" status.	1. Navigate to the "All Leaves" Page.	Beside the "status" column, the "delete" button should be visible if the status is "Pending" or "Saved as Draft".	Pass	Status that is not "Pending" or "Saved as Draft" will show a dash instead of the delete button.

10.2	Confirmation message should show when the delete button is clicked.	1. Click on the “delete” button.	A pop-up message with the text “Are you sure you want to delete?” is shown, with the buttons “Ok” and “Cancel”.	Pass	-
10.3	Users are able to delete a leave application.	1. On pop-up message, click on “Ok”	The row where the delete button is clicked is deleted from the database and no longer shown in the table.	Pass	-
10.4	Users are able to cancel the delete action when the confirmation message is shown.	1. On pop-up message, click on “Cancel”	The message box will be closed, and no changes will be made to the table in “All Leaves” page.	Pass	-

Unit Test for Claim

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
11.1	Users can view the Claim page that displays all the claims they have submitted.	1. After logging in, click on the menu icon to bring up the side navigation bar, and click on the "Claim" tab.	The "Claims" page is displayed. The table in the Claims page shows all claims submitted	Pass	The table should be empty if the user has not submitted any claims.

Unit Test for Add Claim

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
12.1	Users can access the "New Claim" page.	1. On the "Claims" page, click on the "New Claim" button.	The "New Claim" page is displayed.	Pass	-
12.2	The "Staff ID", "Name", and "Department" fields should be automatically populated.	1. On the "Claims" page, click on the "New Claim" button.	The "Staff ID" field is populated with the staff's ID, "Name" with the user's name, and "Department" with the user's department.	Pass	-

12.3	Users are able to create a new claim.	1. Fill in the required fields in the "New Claim" page. 2. Click on the "Submit" button.	Users are directed back to the "Claims" page, with the newly created Claim displayed in the table.	Pass	Newly created Claims are set to "Pending" status.
12.4	Clicking on "Submit" while leaving any of the fields blank should show a prompt for the user to fill up the field.	1. On the "New Claim" page, leave any of the fields empty. 2. Click on the submit button	An error box with the text "Please fill out this field" is displayed.	Pass	-
12.5	Users are able to save their claim as a draft.	1. Click on the "Save as Draft" button.	Users are directed back to the "Claims" page with the claim added into the table with the status of "Saved as Draft"	Pass	-
12.6	Users are able to cancel their claim halfway through creation	1. Click on the "Cancel" button.	Users are redirected back to the "Claims" page.	Pass	-

Unit Test for Edit Claim

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
13.1	After clicking on any checkbox, clicking the "Edit Claim" button will show the "Edit Claim" page.	1. On the "Claims" page, check the checkbox of any row. 2. Click on the "Edit Claim" button.	The "Edit Claim" page is displayed with the fields pre-populated with values retrieved from the database.	Pass	-
13.2	Users are able to modify details of a claim.	1. On the "Edit Claim" page, modify any of the fields. 2. Click on the "Submit Claim" button.	Users are redirected to the "Claims" page, with the edited information reflected in the table.	Pass	"Staff ID", "Name" and "Department" cannot be modified.
13.3	Users can only select one claim to modify	1. Click on more than one checkbox on any row. 2. Click on the "Edit Claim" button.	An error message saying "Please select only one claim to edit" will show up.	Pass	
13.4	Users must select one claim to edit.	1. Click on the "Edit Claim" button.	An error message saying "Please select a claim to edit" will show up.	Pass	

Unit Test for Delete Claim

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
14.1	When deleting claim(s), a confirmation pop-up should appear.	1. On the "Claims" page, check the checkbox of any row. 2. Click on the "Delete Claim" button.	The page would prompt a confirmation pop-up asking "Are you sure you want to delete".	Pass	-
14.2	Users can cancel deletion of claim(s) using the confirmation pop-up	1. On the "Claims" page, check the checkbox of any row. 2. Click on the "Delete Claim" button. 3. On the pop-up click the "Cancel" button.	The selected claim(s) will not be deleted.	Pass	-

14.3	Users can delete the claim(s) after confirming on the confirmation pop-up.	1. On the "Claims" page, check the checkbox of any row. 2. Click on the "Delete Claim" button. 3. On the pop-up click the "Ok" button.	The selected claim(s) will be deleted from the list.	Pass	-
14.4	The checkbox for Approved claims will not appear.	1. Using the side menu click on "Claims".	The Approved claims checkbox will not appear in that row.	Pass	

Unit Test for Feedback

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
15.1	Uses are able to view the feedback page	1. Click on the "feedback" tab in the side menu	The feedback form should be displayed with the "Name" and "Email" fields auto-filled.	Pass	"Name" and "Email" fields are auto-filled with the logged in user's name and email respectively .

15.2	Users should be able to submit feedback.	1. In the “feedback” page, fill up the “feedback” field. 2. Click on ‘submit’.	The feedback page redisplays.	Pass	-
15.3	Users are not able to submit an empty feedback.	1. In the “feedback” page, click on submit without inputting anything.	An error prompt displays with the text “Please fill out this field”.	Pass	-
15.4	Users are able to double-check the feedback.	1. Click on the submit button after filling up the feedback form. 2. A prompt will pop up and let the user double check their feedback.	A prompt with the message “Are you sure you want to submit the feedback?” will pop up	Pass	-

Unit Test for All Feedback (Admin)

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
16.1	Users with the "Admin" role will be able view all submitted staff feedback.	1. Click on the "View Feedbacks" tab from the side menu	A table is displayed, showing all the submitted staff feedback.	Pass	-
16.2	Users with the "Staff" role are not able to see the "View Feedbacks" tab on their side menu.	1. Open the side menu.	The side menu should not have the "View Feedbacks" tab.	Pass	-
16.3	The admins will be able to delete unnecessary feedback.	1. Click on the delete button beside the feedback the admin desires to delete.	The feedback that wants to be deleted will be removed from the table of feedbacks	Pass	-

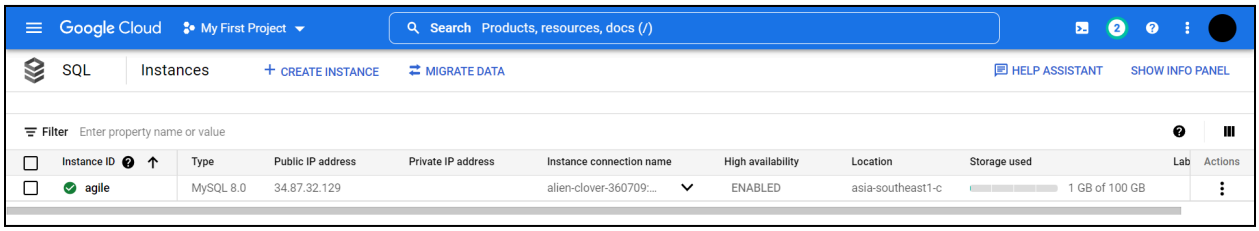
Unit Test for All Payslips

Test No.	Test Description	Test Steps	Expected Result	Status (Pass/Fail)	Remarks
17.1	Users able to select and view the selected month's payslip	1. Using the side menu click on "Payslip". 2. Using the dropdown list select any date.	The selected month's payslip will populate the tables below.	Pass	-

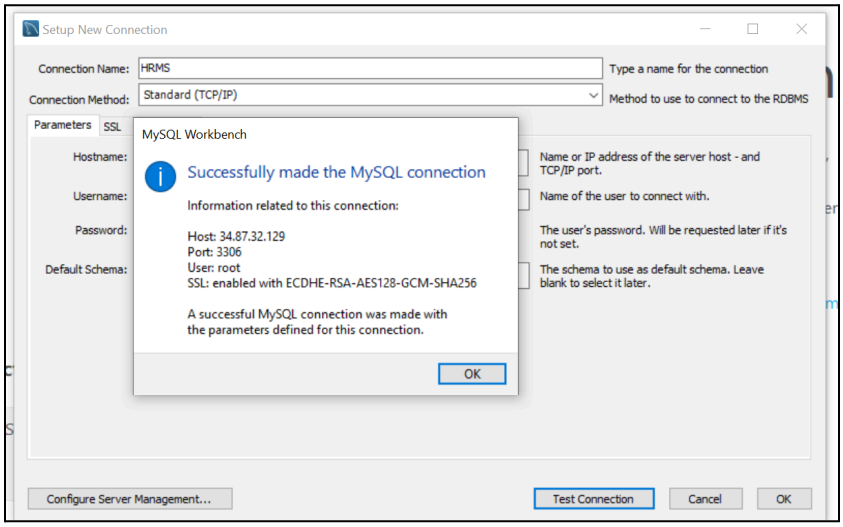
Design

Database (Google Cloud SQL)

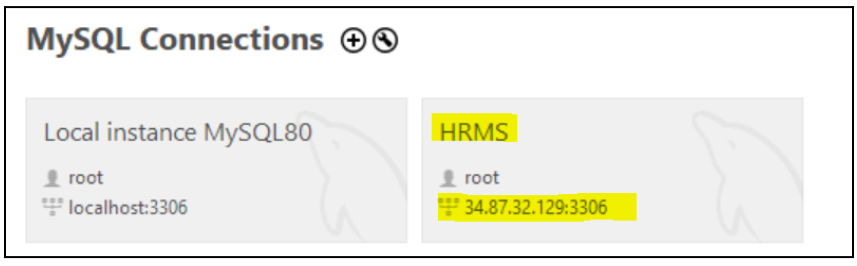
The database for our HRMS Application is hosted on Google Cloud (https://cloud.google.com/). This is done by creating a SQL database instance on Google Cloud. Afterwards, a connection is established in MySQL Workbench by keying in the Public IP address given in Google Cloud. We used MySQL Workbench as we are more familiar with it. After establishing a connection, we created a new schema called “HRMS” in MySQL Workbench, and the SQL statements that we have written are executed in order to create the tables required in our database.
(See Appendix C for All SQL statements written)



(Database instance created on Google Cloud)

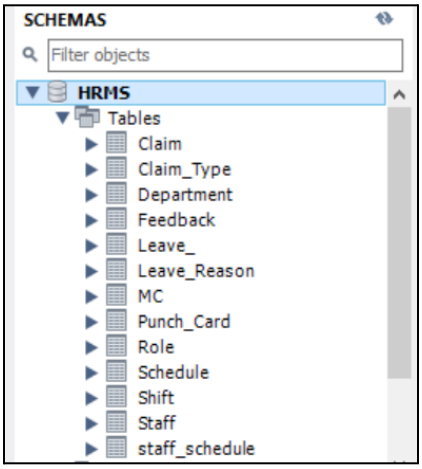


(Establishing connection from MySQL Workbench)



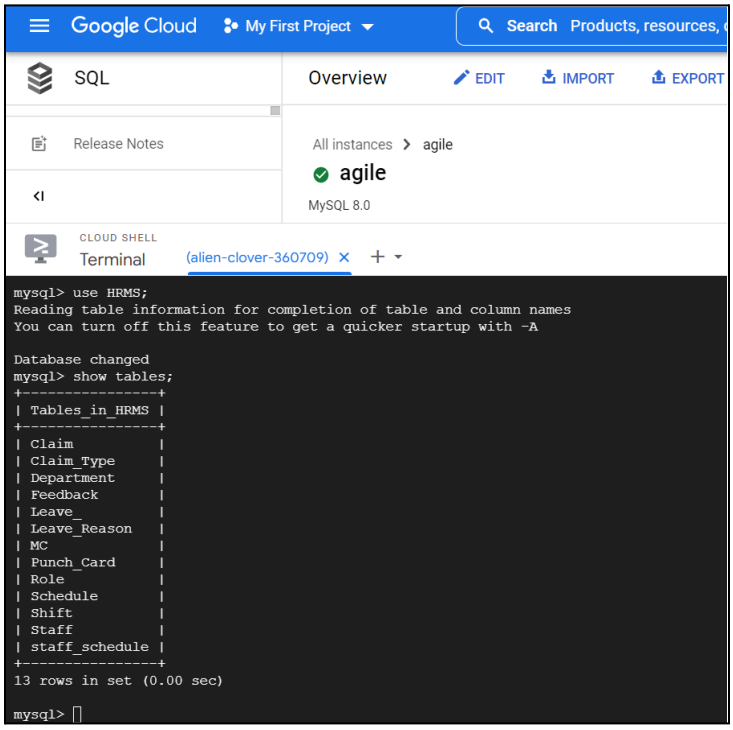
(Connection Established)

```
CREATE table Staff (
    Staff_id INT NOT NULL auto_increment,
    Dept_id INT NOT NULL,
    Role_id INT NOT NULL,
    Staff_name VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL,
    address VARCHAR(50) NOT NULL,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL,
    PRIMARY KEY (Staff_id),
    FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id),
    FOREIGN KEY (Role_id) REFERENCES Role(Role_id)
);
```

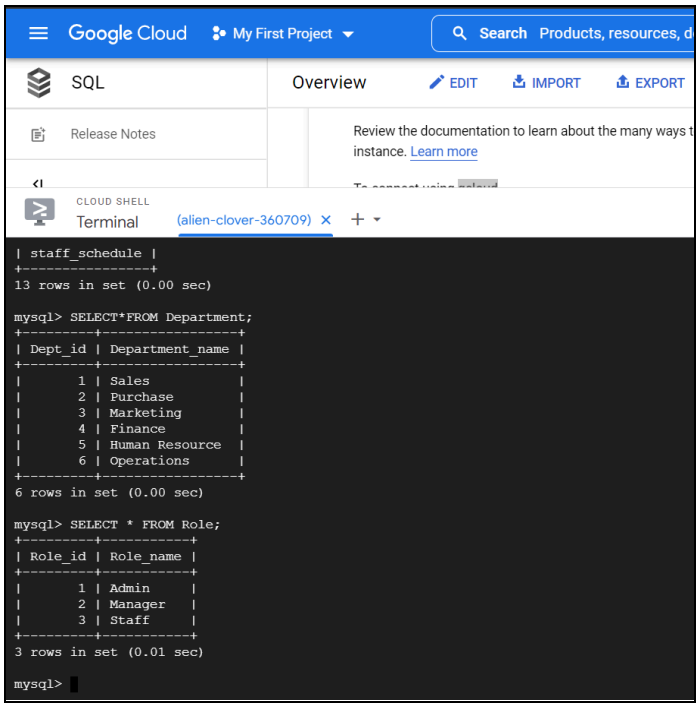


(Left: SQL statement to create Staff table, Right: All tables created in MySQL Workbench)

To verify that the connection is successful, we used the CLI in google cloud to run the “show tables” command. We were able to see that the tables that we have created in our MySQL Workbench have also been created in the database hosted on Google Cloud. We could also see the data we had inserted using a SELECT statement.



(Tables in the hosted database)



(Rows inserted into the Department and Role tables)

Below show the code used to establish a connection to the database using node.js. The public IP address of the database is used as the host, and the name and password of the database is specified. Adding a console.log() statement to print out information regarding the connection has allowed us to verify that a connection was successfully established.

```
// create connection to database hosted on
// Public IP: 34.87.32.129
const db = mysql.createConnection ({
    host: "34.87.32.129",
    user: "root",
    password: "RYsCu397",
    database: "HRMS"
});

// connect to database
db.connect((err) => {
  if (err) {
    throw err;
  }
  console.log("Connected to database");
  console.log(db);
});
global.db = db;
```

```
      _paused: false
    },
    [Symbol(kCapture)]: false
  },
  _connectCalled: true,
  state: 'connected',
  threadId: 5093,
  [Symbol(kCapture)]: false
}
```

User Interface (UI)

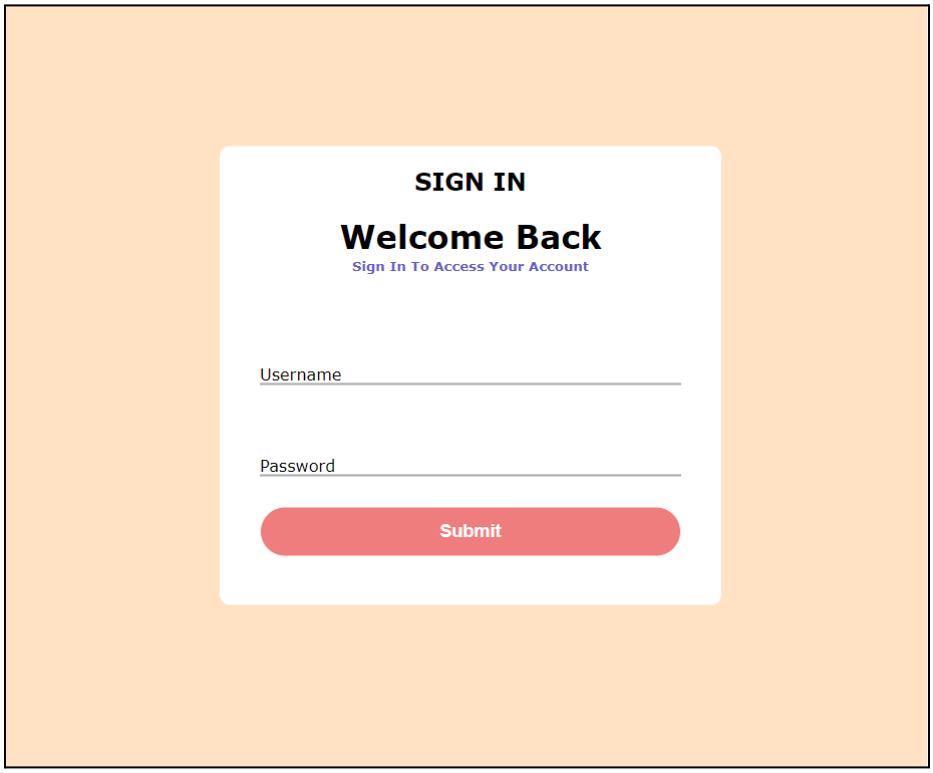
A user interface is whatever the user interacts with to the functionalities provided by a service or product. A good user interface is one where it is effective, easy to understand, and most importantly functional.

Our aim is to provide a simple and minimalistic user interface that will allow easy usage and navigation for our users. It should allow our users to carry out actions (E.g. submit a new claim) without any confusion.

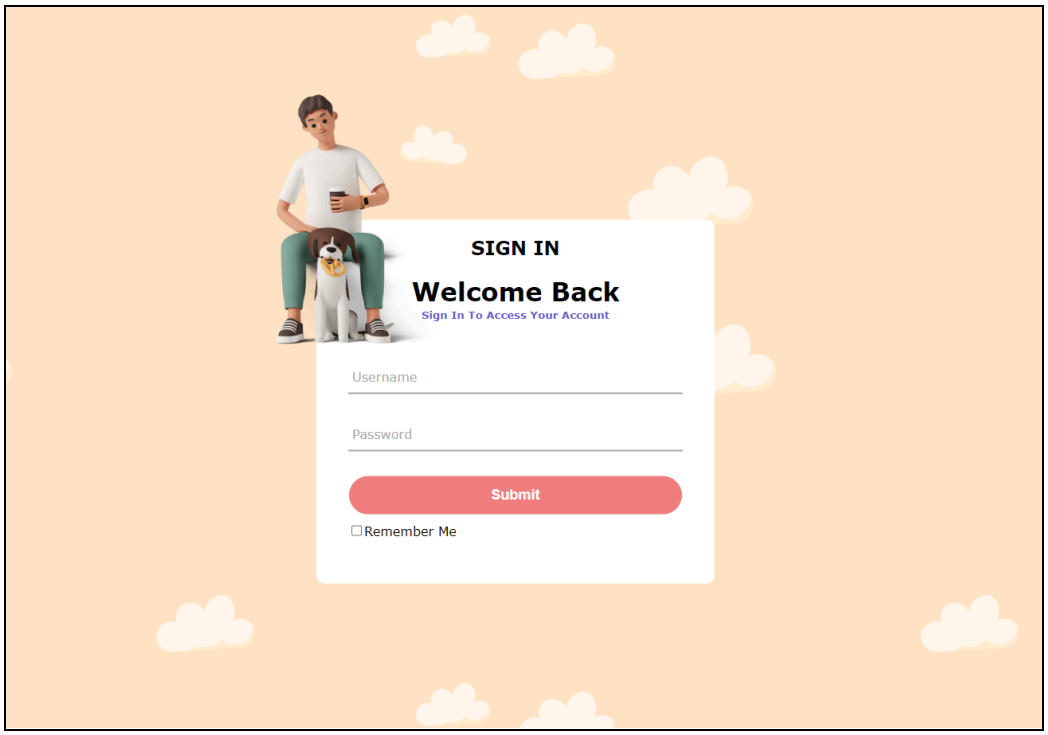
Below shows the UI for each page of our application, as well as the iterations we have gone through to achieve the final product.

Login Page

The login page is the landing page of our application. This page allows users to login to the application to access other functionalities provided by the HRMS application. After reviewing the first iteration, users have feedback that they would like to add images that are in line with the company name ('The Smurfy Gang'). After several iterations with different pictures used, we arrived at the final iteration which the users were happy with; a picture of a boy with a dog, with clouds in the background.



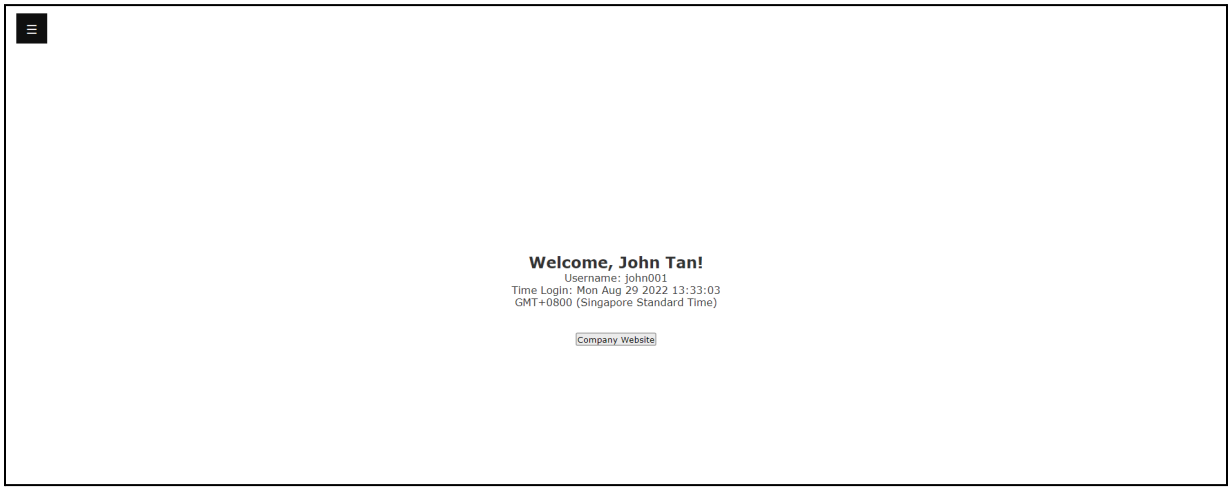
(first iteration)



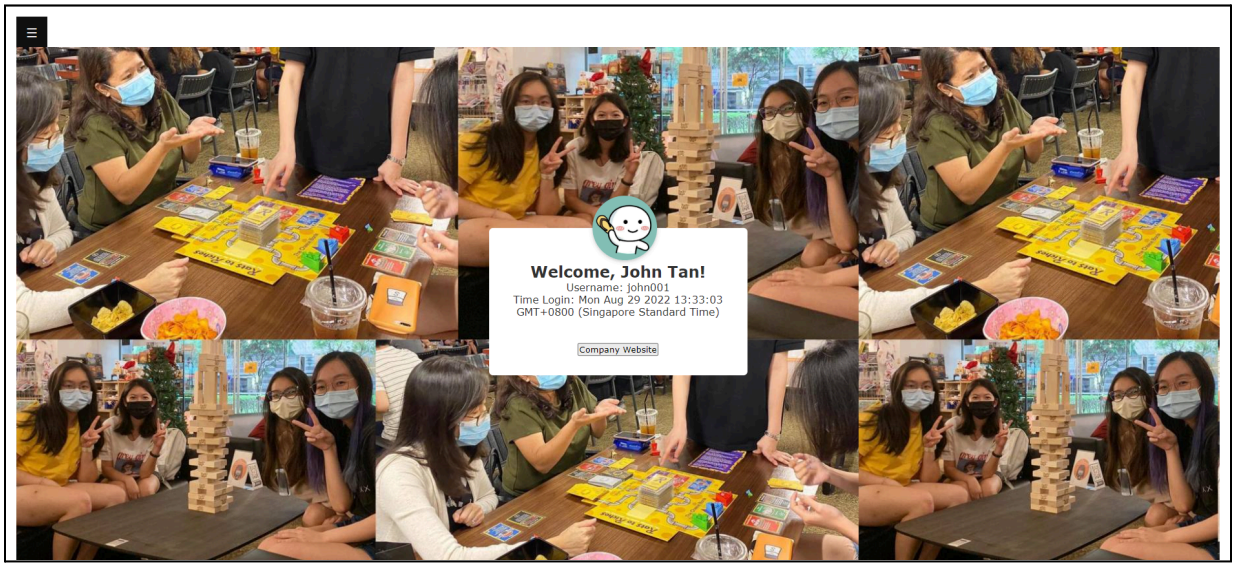
(final iteration)

Homepage

The homepage is displayed right after a successful login. The homepage shows the user's name, username, and the time of login. In the first iteration, we kept to a plain design. However, users feedback that it was way too simple, and they requested for us to add in some pictures as the background. Several iterations later, we used an image of the employees taken during a company event, which the users were satisfied with.



(first iteration)



(final iteration)

Navigation Bar

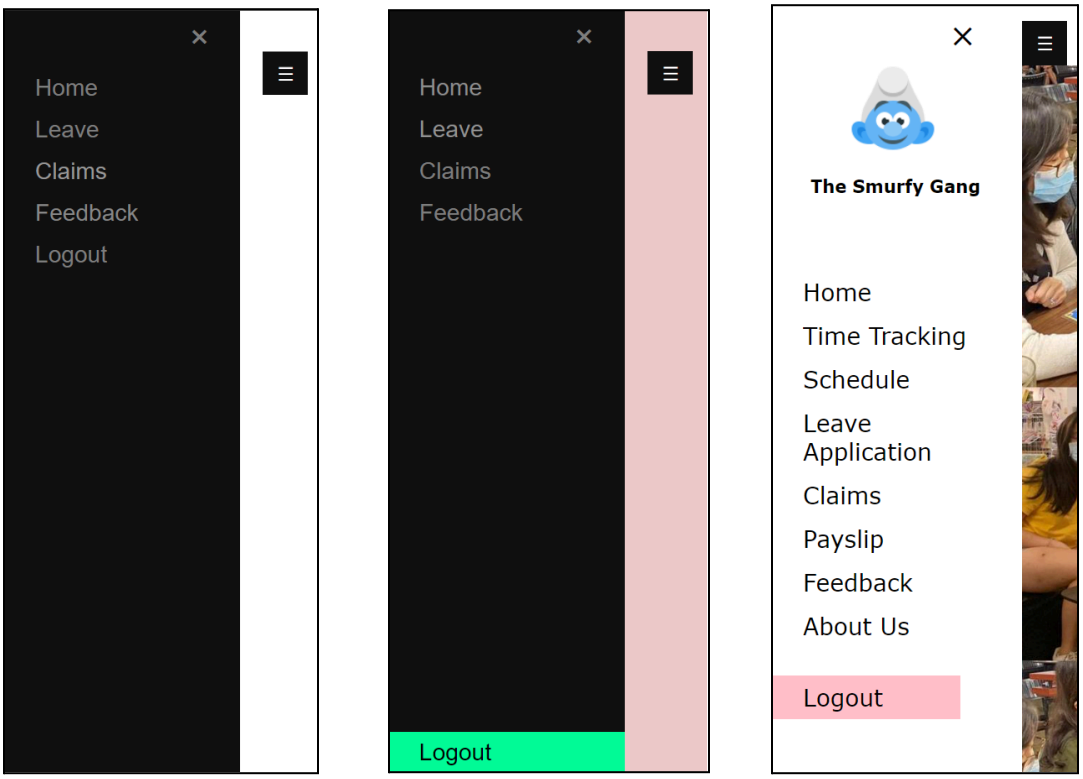
The navigation bar can be brought out by clicking on the menu icon shown below. This menu icon is found on every page except the login page.

Upon clicking the Menu icon, the side-navigation bar is brought out with a sliding motion.

In the first iteration, the “logout” button was located below the “Feedback” tab. However, users have feedback that it is difficult to locate the “logout” button as it blends in with the other tabs. Users also hope to display their company logo and name at the top of the navigation bar.

Therefore, we went through several iterations, and ended up with a side-navigation bar that looks entirely different from the one in the first iteration. The background of the bar is changed to white, with black text. The company’s name and logo is displayed at the top of the navigation bar, and the “logout” button is positioned lower down, with a pink background and black text. The black text will turn light grey upon mouse hover.

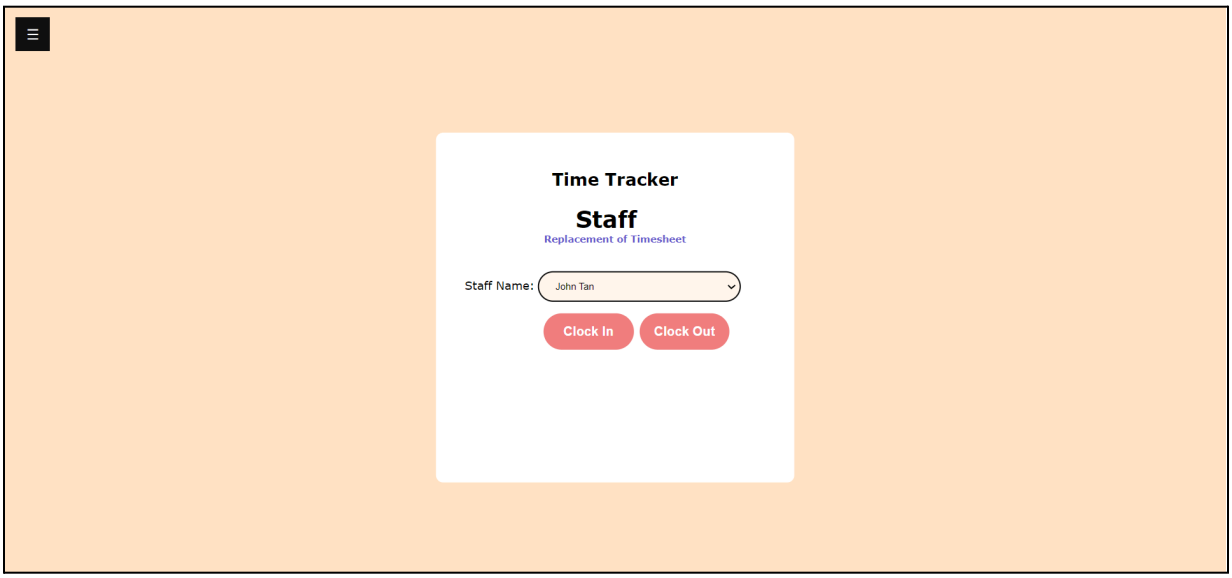
Below shows several iterations of the navigation bar we went though.



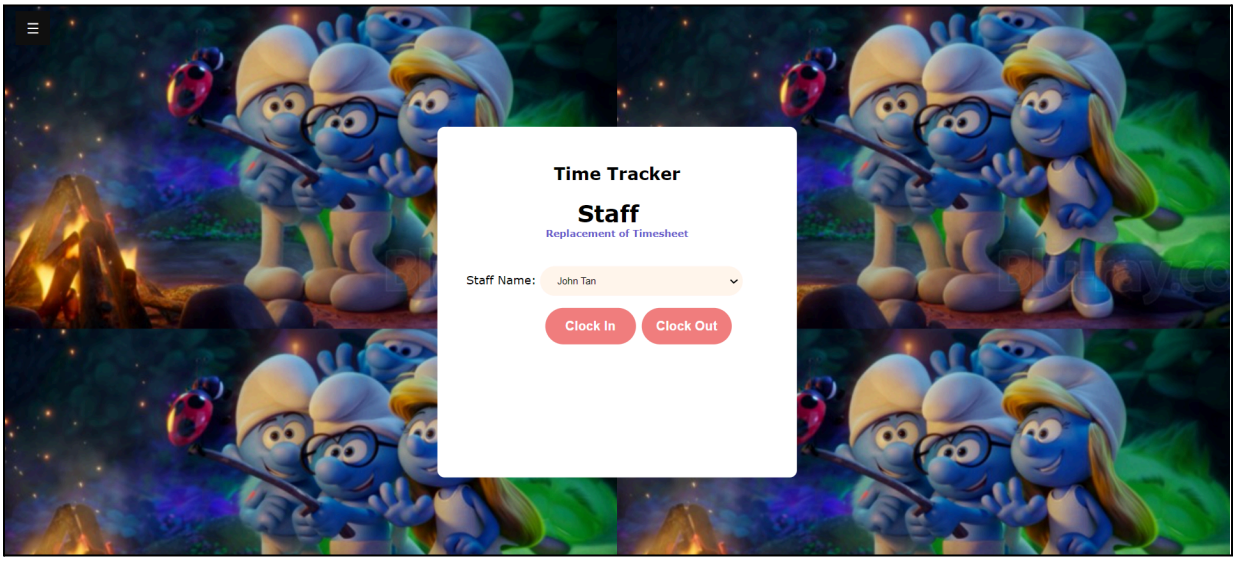
(left to right: first to last iteration)

Time Tracking (Clock in/out)

The time tracking page allows users to clock in when they start their shift, and clock out when their shift ends. In the first few iterations, we used a coloured background to achieve the “minimalistic” look. However, users feedback that since this page would be accessed by staff daily, they would like a more fancy background. Hence, we used a smurf background in the final iteration which the users liked.



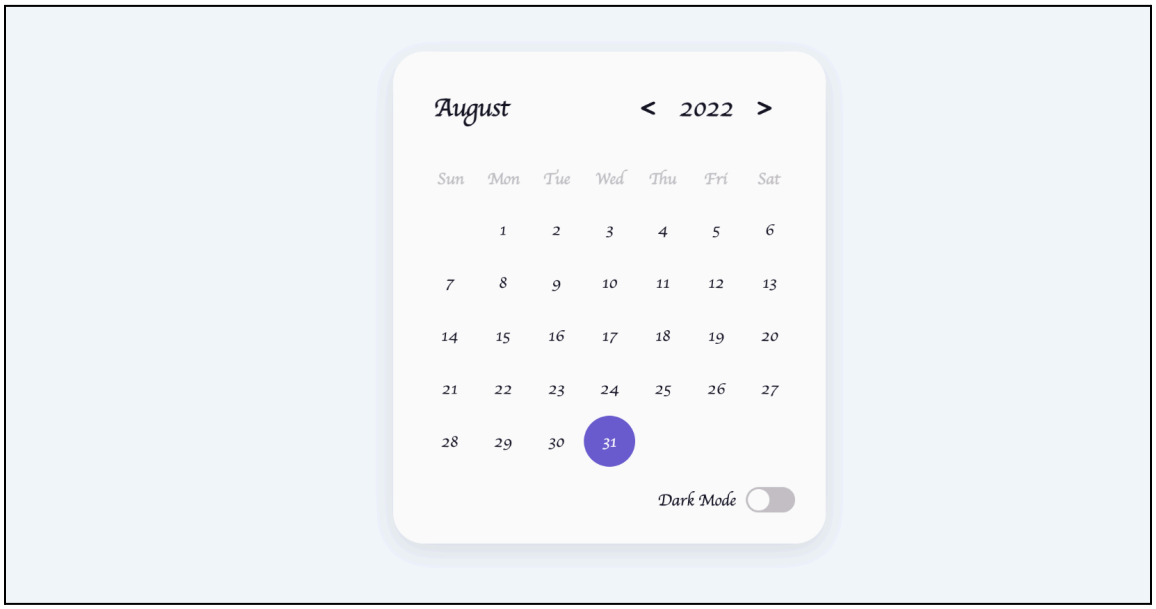
(Second Iteration)



(Final Iteration)

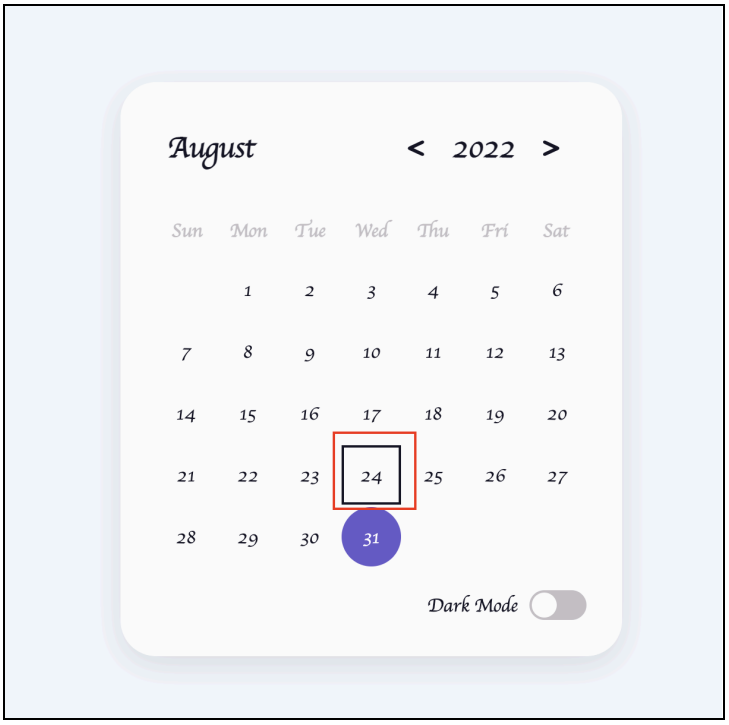
Schedule

The schedule page is a draft of how the Schedule webpage will look like. Our team did not follow the one stated in our mid-term report as stakeholders have emphasised that it is not their top priority.

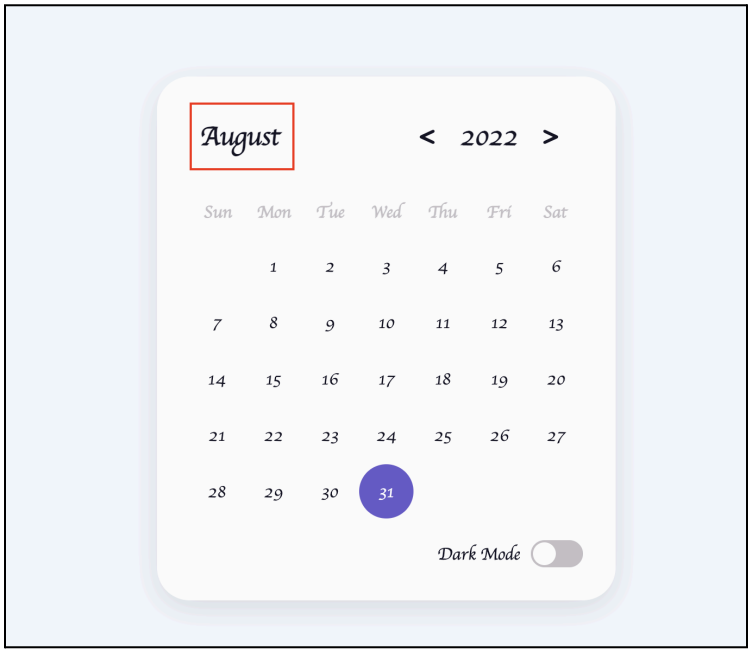


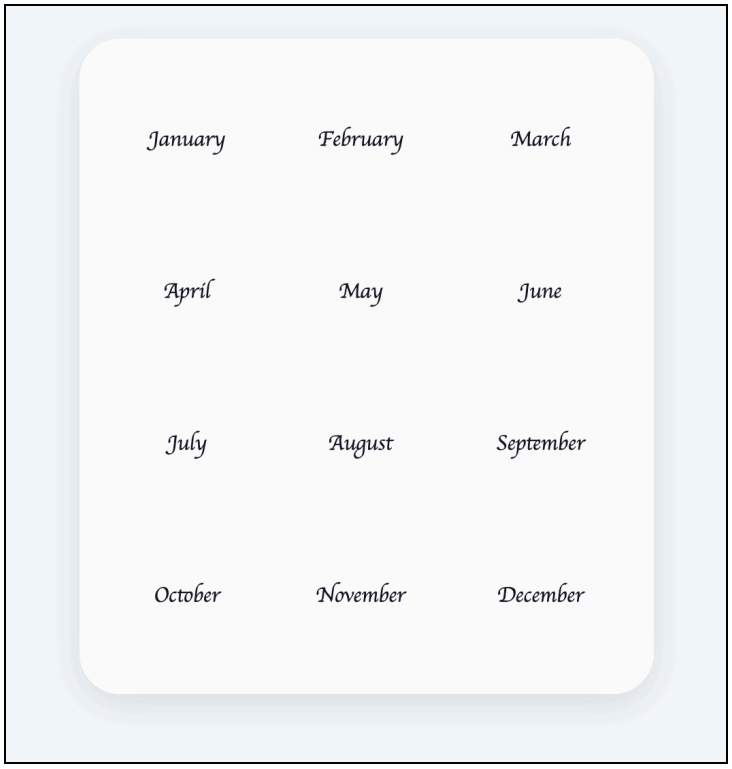
As you can see, today's date is 31st August 2022. Therefore, it is highlighted in the above screenshot with a circle.

This is the light mode version of the calendar. When the user hovers over a desired date, it will be highlighted with a square (as shown below - 24th August 2022)



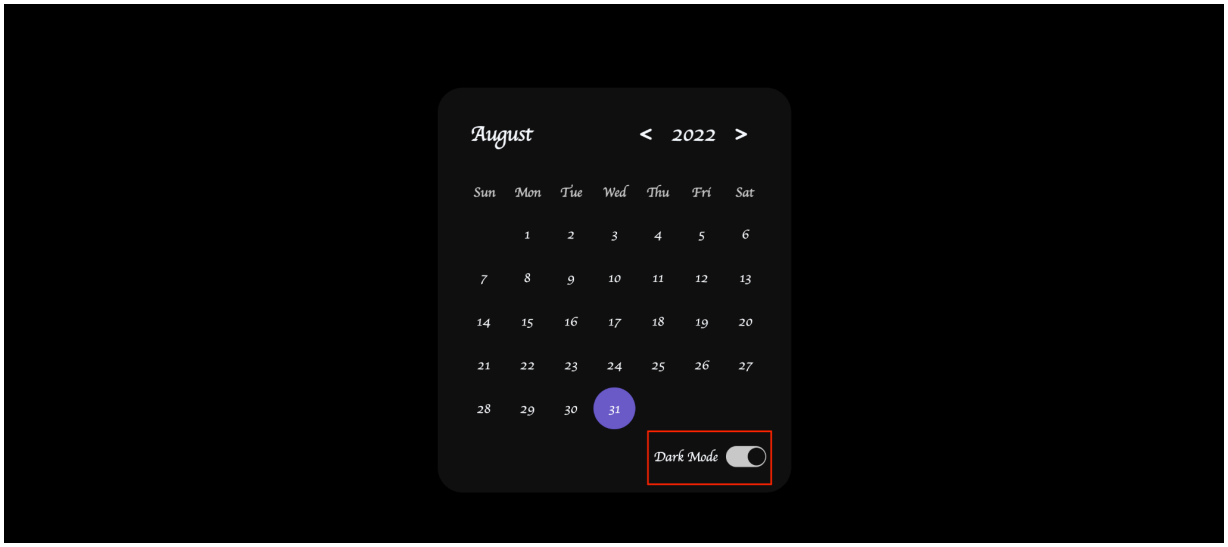
When the user clicks on the month, ‘August’, he will be able to select the desired month. Any of the month, from January to December.





(First Iteration - light mode)

The dark mode has the same functions as the light mode. However, the colour changes from light blue to black.



(First Iteration - dark mode)

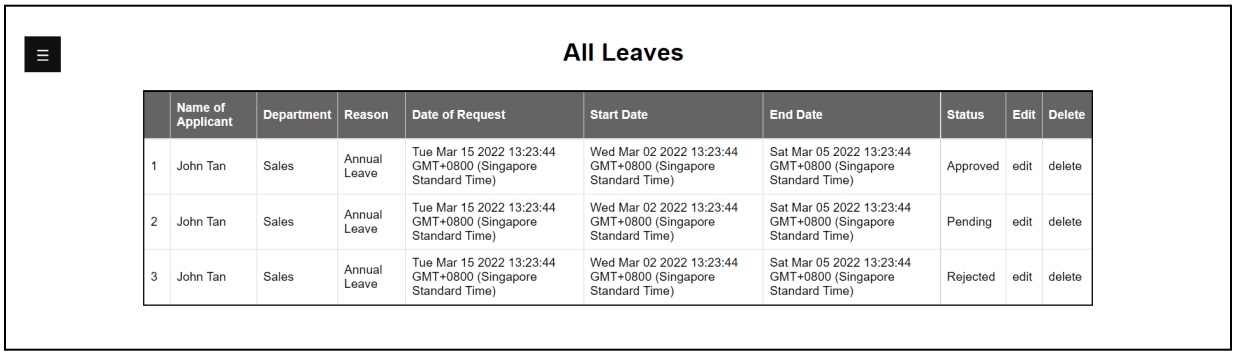
As per our prototype, there is a button for the navigation of the months. Therefore, we changed the layout such that we are no longer navigating the year but the months. This is the light mode version. The same goes for second iteration-dark mode.

(Final Iteration - light mode)

The layout of the final product for maintenance is still an ongoing discussion so we will focus more on the other functionalities. Such as, leave and claims.

Leave (All Leaves)

The All Leave page can be accessed by clicking on the “Leave” tab on the navigation bar. This page displays all leave applications submitted by the logged in user. In the first iteration of the all leaves page, the table displays the Name of Applicant, Department, Date of request, Start date, end date, status, and the last two columns are for the edit and delete function.

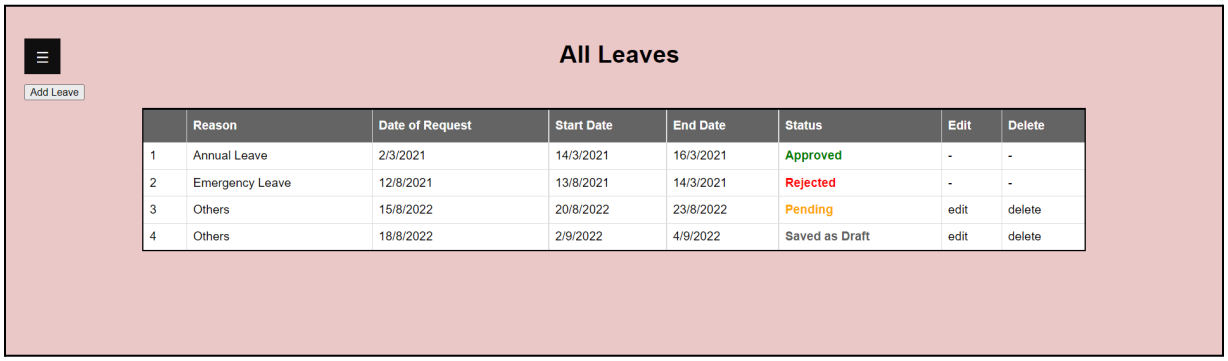


All Leaves

	Name of Applicant	Department	Reason	Date of Request	Start Date	End Date	Status	Edit	Delete
1	John Tan	Sales	Annual Leave	Tue Mar 15 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Wed Mar 02 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Sat Mar 05 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Approved	edit	delete
2	John Tan	Sales	Annual Leave	Tue Mar 15 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Wed Mar 02 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Sat Mar 05 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Pending	edit	delete
3	John Tan	Sales	Annual Leave	Tue Mar 15 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Wed Mar 02 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Sat Mar 05 2022 13:23:44 GMT+0800 (Singapore Standard Time)	Rejected	edit	delete

(First Iteration)

After several iterations, we arrived at the final iteration of the all leaves page. The “Name of Applicant" and “Department” columns have been removed as those were not information that was necessary to the user. All dates have been formatted to dd/m/yyyy format to provide better readability. The status of each leave application is also displayed in different colours (Red = rejected, Green = Approved, Yellow = Pending). The edit and delete button only shows on the rows where the status is “Pending”.



All Leaves

Add Leave

	Reason	Date of Request	Start Date	End Date	Status	Edit	Delete
1	Annual Leave	2/3/2021	14/3/2021	16/3/2021	Approved	-	-
2	Emergency Leave	12/8/2021	13/8/2021	14/3/2021	Rejected	-	-
3	Others	15/8/2022	20/8/2022	23/8/2022	Pending	edit	delete
4	Others	18/8/2022	2/9/2022	4/9/2022	Saved as Draft	edit	delete

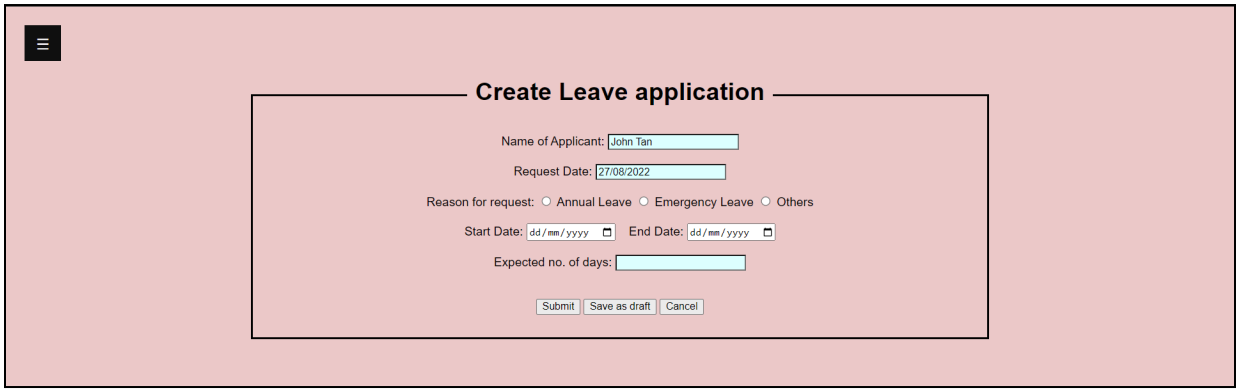
(Final Iteration)

Leave (Create Leave)

Clicking on the “Add Leave” button on the All Leaves page will direct users to the Create Leave Application page where a new leave application can be created. The users appreciated the simplicity shown in the first iteration. However, we felt that it was way too simple. Hence, we suggested changing the background colour which the users agreed with. They had also requested for us to add a “Cancel” and “Save as Draft” button.

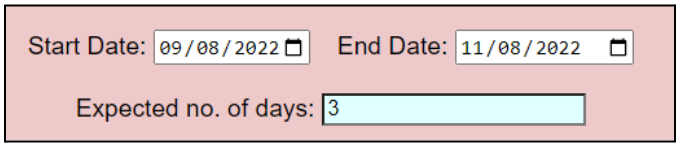
(First Iteration)

The “Name of Applicant” is automatically populated with the name of the current logged in user, while the “Request Date” field is automatically populated with the current date.



(Final Iteration)

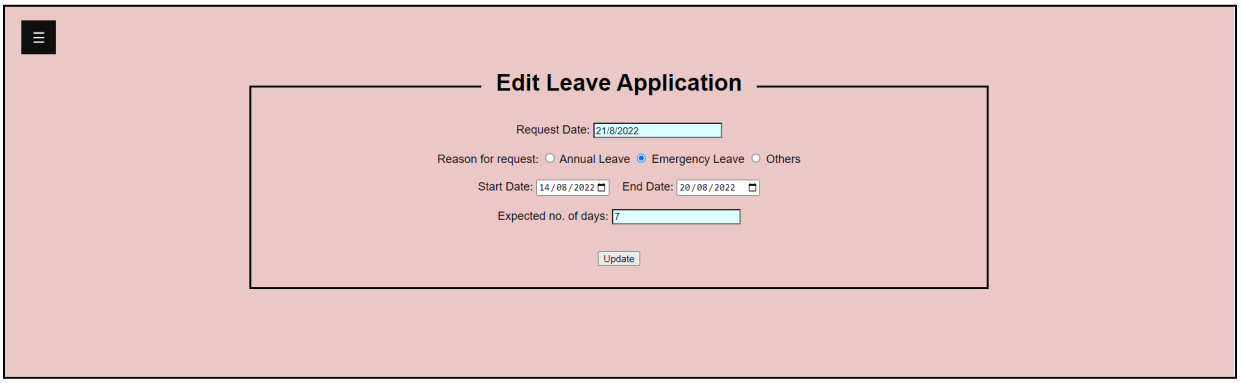
The “expected number of days” field will automatically calculate the number of days of leaves based on the leave start date and end date that has been selected by the user.



Leave (Edit Leave)

4	Emergency Leave	21/8/2022	14/8/2022	20/8/2022	Pending	edit	delete

Clicking on the “edit” button in the All Leaves page will direct users to the “Edit Leave Application” page where details of the leave application can be modified. In the “Edit Leave Application” page, fields are pre-populated with data retrieved from the database.



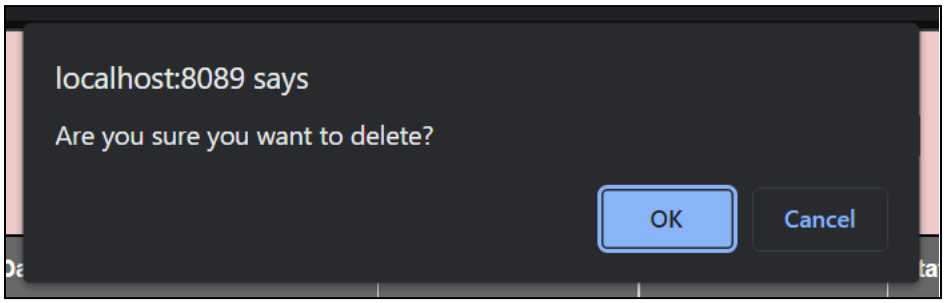
After modifying any of the information, clicking on the update button will direct the user back to the “All Leaves” page, with the change that has just been made being reflected.

4	Emergency Leave	21/8/2022	14/8/2022	18/8/2022	Pending	edit	delete

Leave (Delete Leave)

3	Annual Leave	11/8/2022	2/8/2022	6/8/2022	Pending	edit	delete

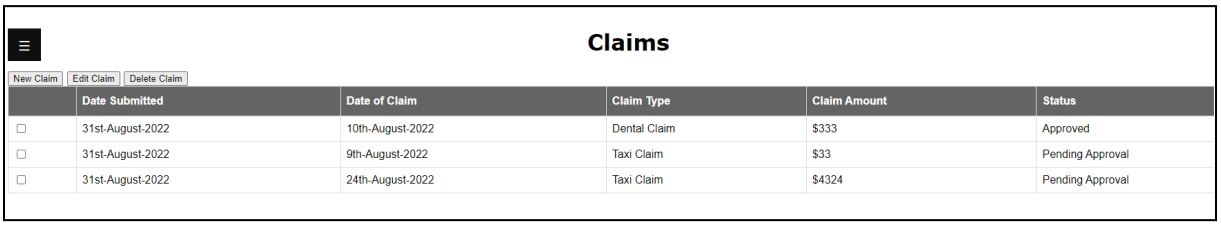
Leave applications can be deleted by clicking on the “delete” button. In the first iteration, the leave application will automatically be deleted after the delete button was clicked. However, users have feedback that a confirmation message would be good as it would prevent accidental deletions if one were to press the button on accident. Therefore, we added code to display a confirmation message as shown below.



Clicking on the “OK” button will delete the leave application, while clicking on “Cancel” will cancel the action and close the message box.

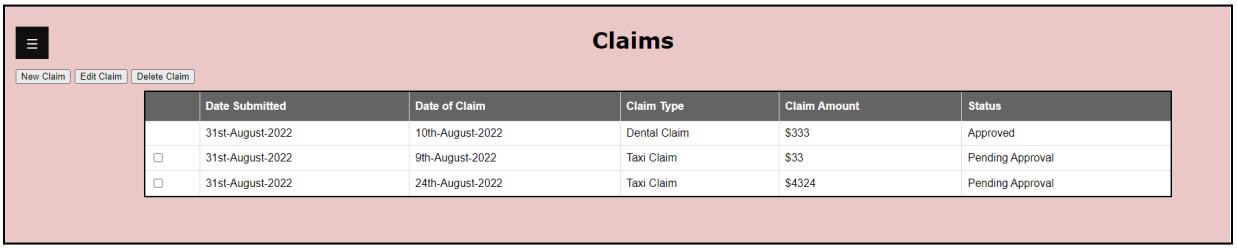
Claims (All Claims)

The All Claims page can be accessed by clicking on the “Claims” tab on the navigation bar. This page displays all claims applications submitted by the logged in user. In the first iteration all the rows with claims would have a checkbox in the first column.



(First Iteration)

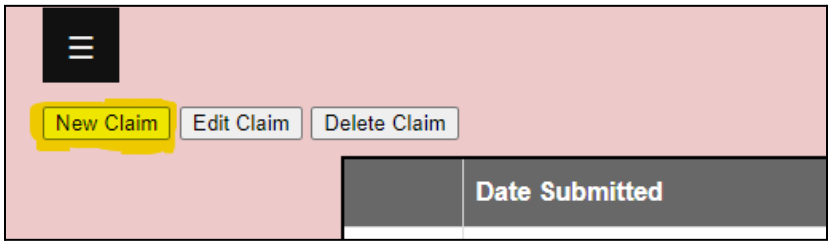
The final iteration of the all claims page. The rows with the status of “Approved” Would not have a checkbox as it can confuse people by making them think they can delete/edit approved claims. Thus we removed them.



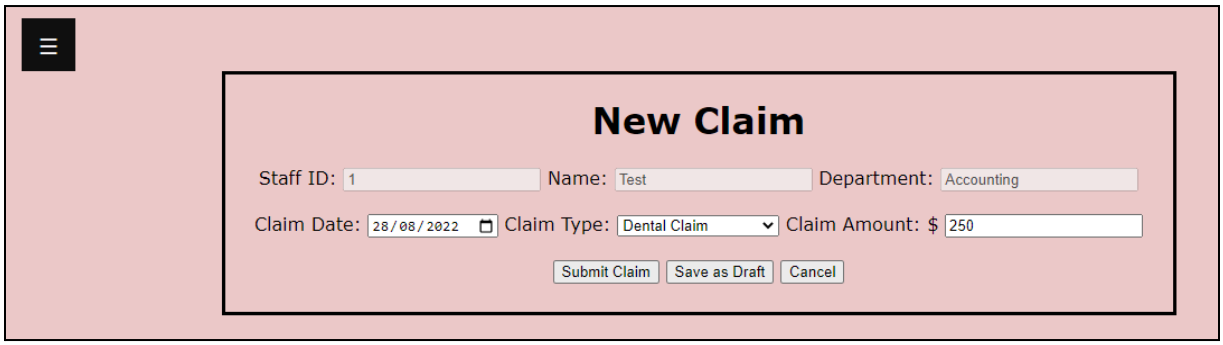
(Final Iteration)

Claims (New Claim)

The new claim page can be accessed by clicking the “New Claim” button on the all claims page.

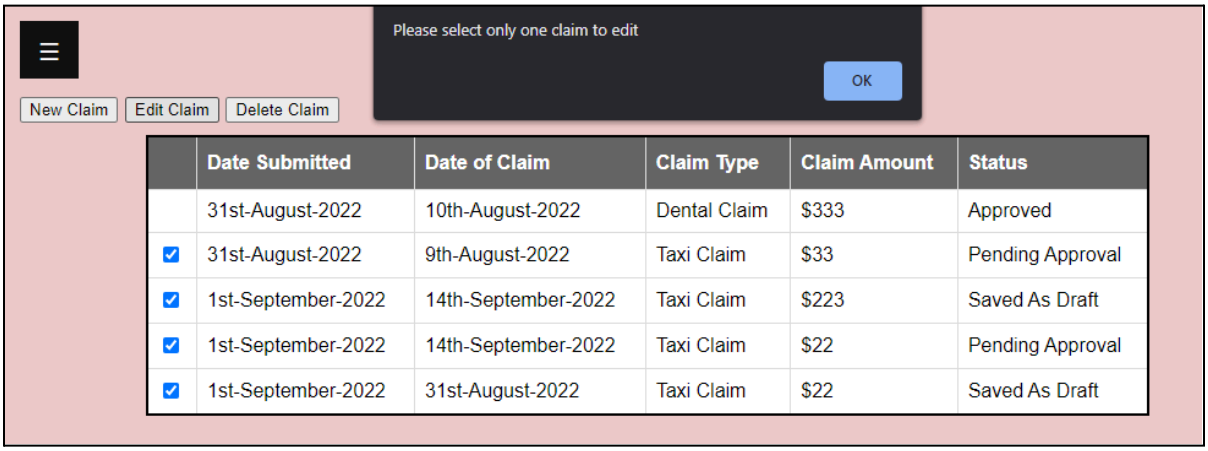


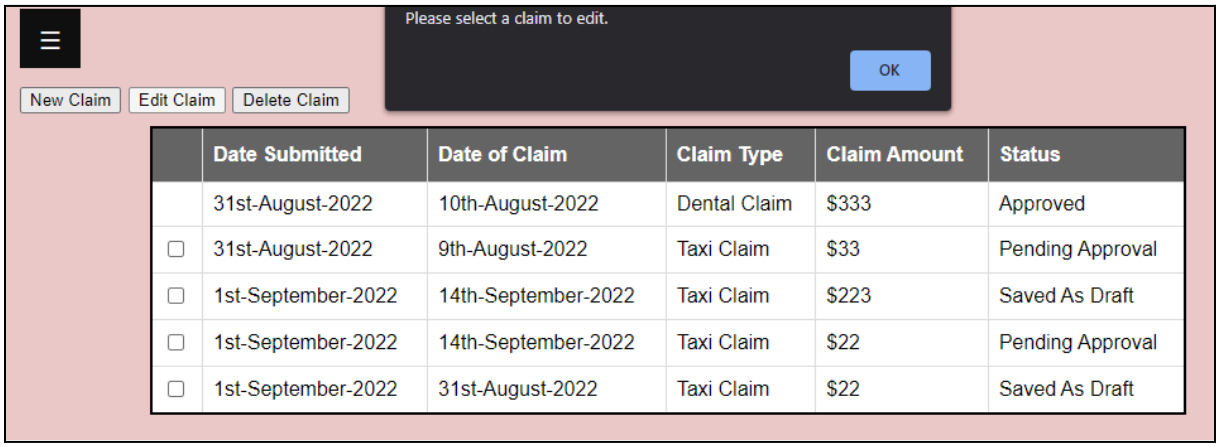
The new claim page will auto-populate the Staff ID, Name and Department fields using data from the database.



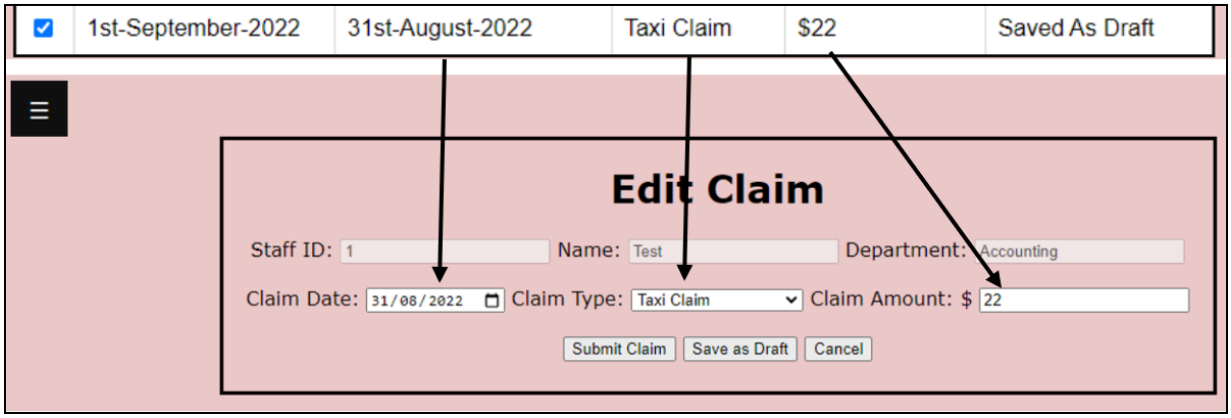
Claims (Edit Claim)

If a user tries to click the “Edit Claim” button with zero or more than one checkbox checked, the system would alert them appropriately.



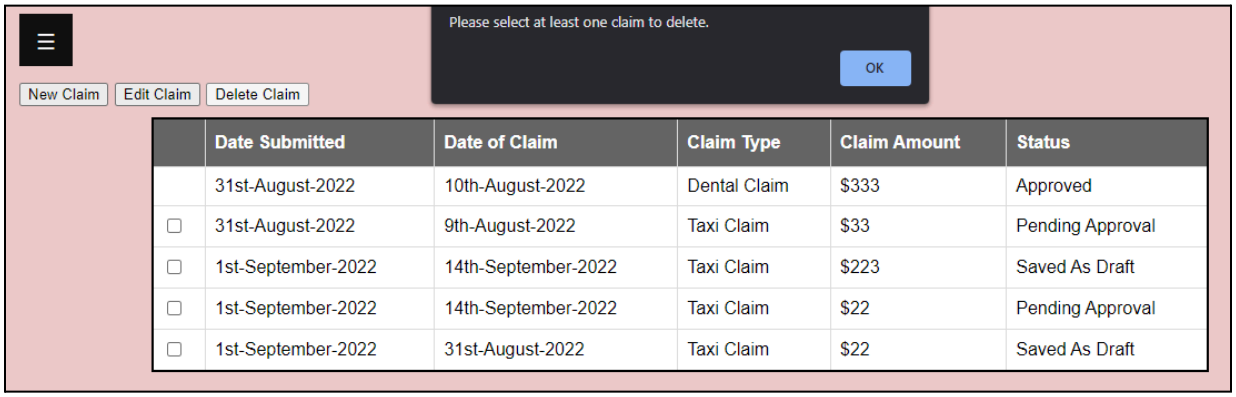


If only one checkbox is selected and the “Edit Claim” button is clicked, it will send them to the edit claim page with all the information populated into the correct fields.



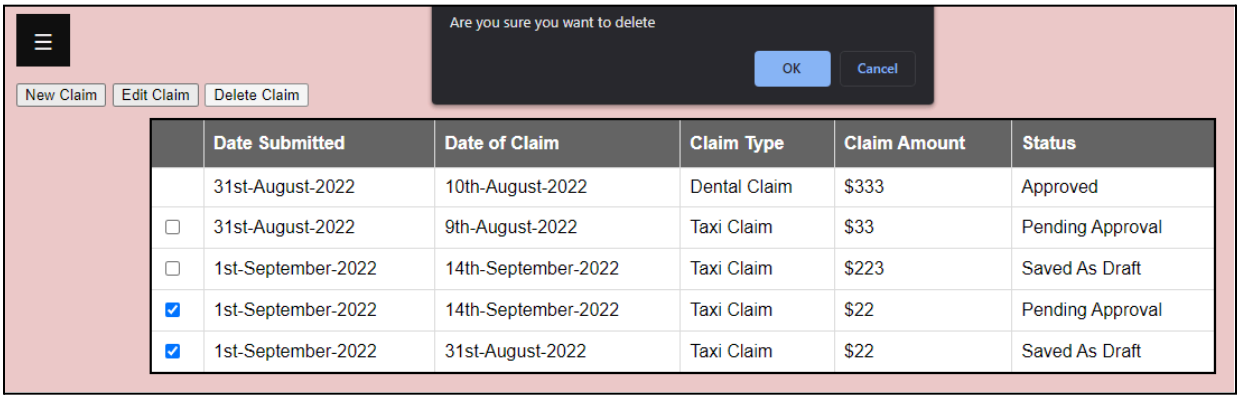
Claims (Delete Claim)

If a user tries to click the “Delete Claim” button with no checkbox checked, the system would have an alert asking them to choose at least one claim to delete.



	Date Submitted	Date of Claim	Claim Type	Claim Amount	Status
	31st-August-2022	10th-August-2022	Dental Claim	$333	Approved
☐	31st-August-2022	9th-August-2022	Taxi Claim	$33	Pending Approval
☐	1st-September-2022	14th-September-2022	Taxi Claim	$223	Saved As Draft
☐	1st-September-2022	14th-September-2022	Taxi Claim	$22	Pending Approval
☐	1st-September-2022	31st-August-2022	Taxi Claim	$22	Saved As Draft

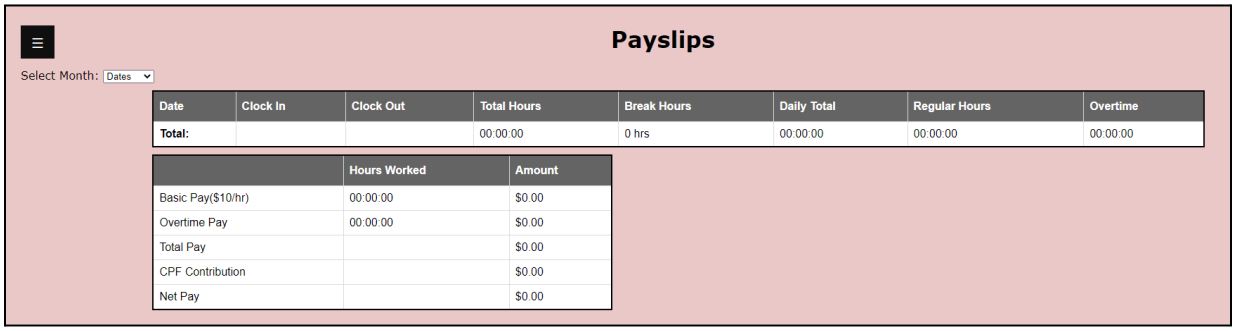
To prevent accidental deletion of claims, we have implemented a confirmation pop-up for users to confirm they want to delete the claim before deleting the selected claims.



	Date Submitted	Date of Claim	Claim Type	Claim Amount	Status
	31st-August-2022	10th-August-2022	Dental Claim	$333	Approved
☐	31st-August-2022	9th-August-2022	Taxi Claim	$33	Pending Approval
☐	1st-September-2022	14th-September-2022	Taxi Claim	$223	Saved As Draft
☑	1st-September-2022	14th-September-2022	Taxi Claim	$22	Pending Approval
☑	1st-September-2022	31st-August-2022	Taxi Claim	$22	Saved As Draft

Payslip

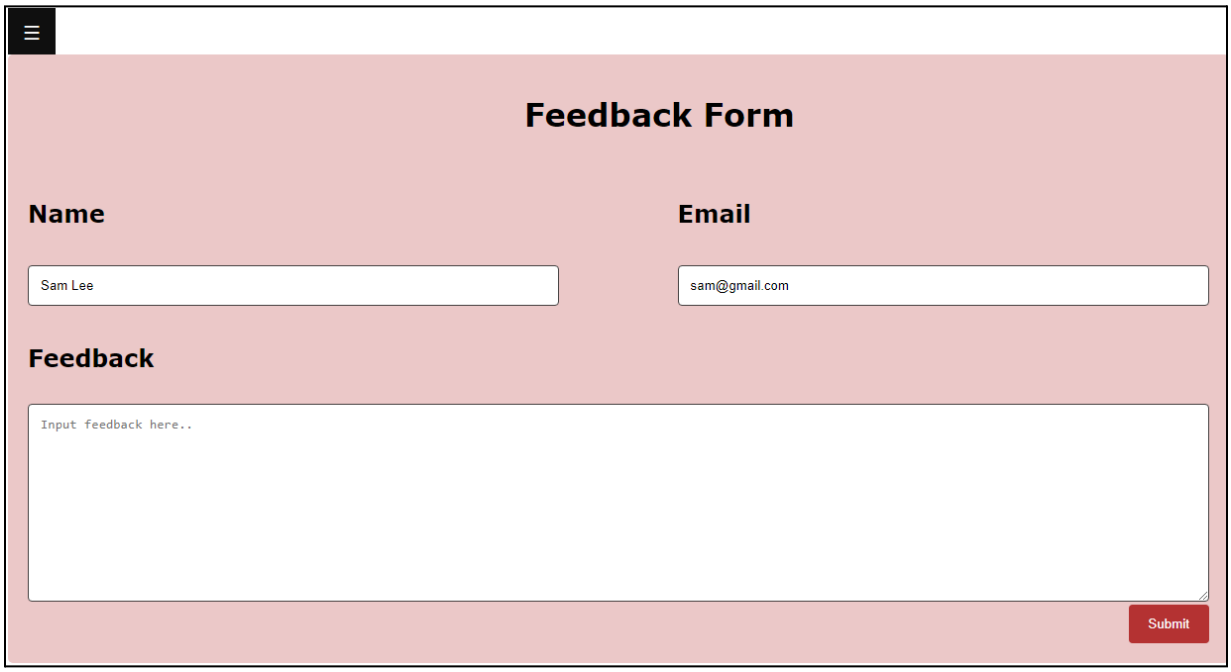
This page calculates the net pay based on the month selected by the user. Calculation of the net pay also takes into consideration the overtime pay and CPF(Central Provident Fund) contribution.



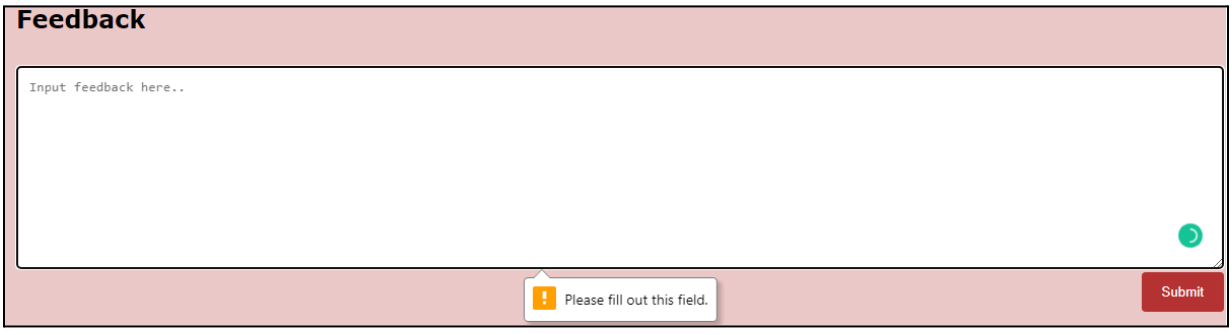
Feedback

The feedback page can be accessed through the “Feedback” tab of the side menu. Users are able to submit any feedback they have regarding the application by filling up the feedback form and clicking submit.

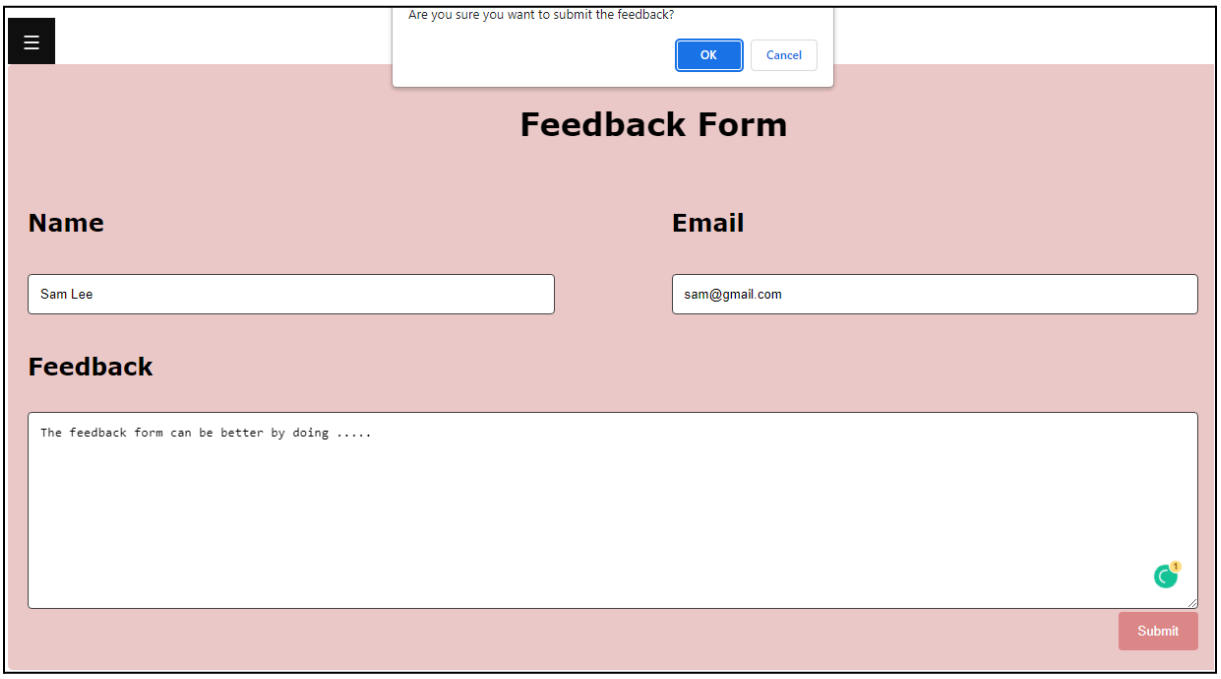
The user name and email is automatically filled and cannot be changed as we do not want the staff to use another staff’s name to submit the feedback.



When submitting the form, the feedback form cannot be empty. If it is empty, the user will be prompted to fill in the field.

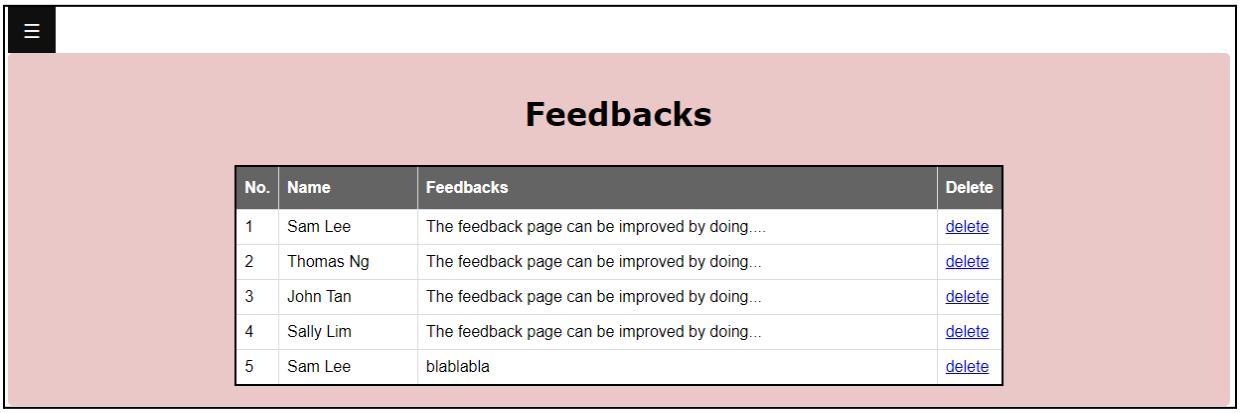


Once the feedback form has been filled, another prompt will pop up to let the user double check the feedback they are about to submit. If the user clicks ok, the feedback will be submitted.

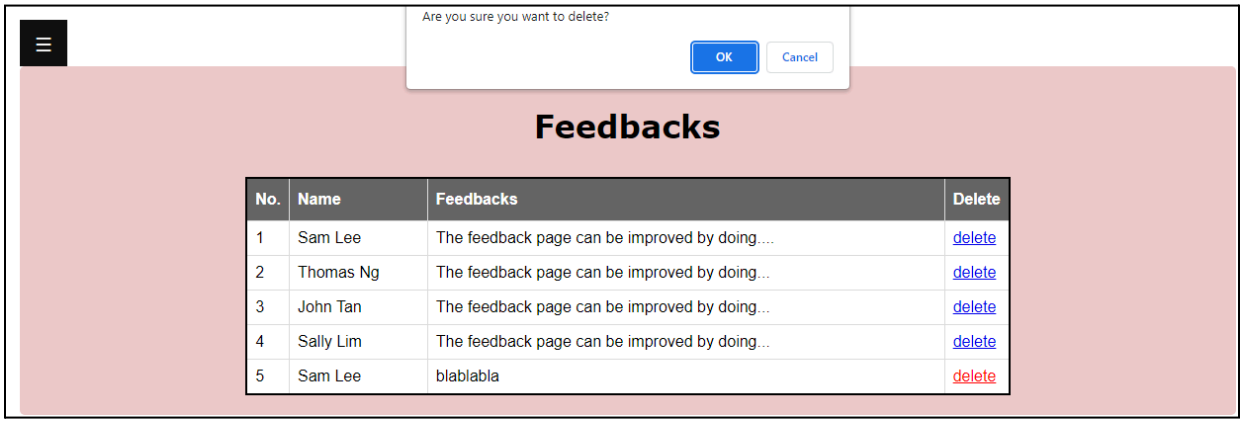


All Feedback

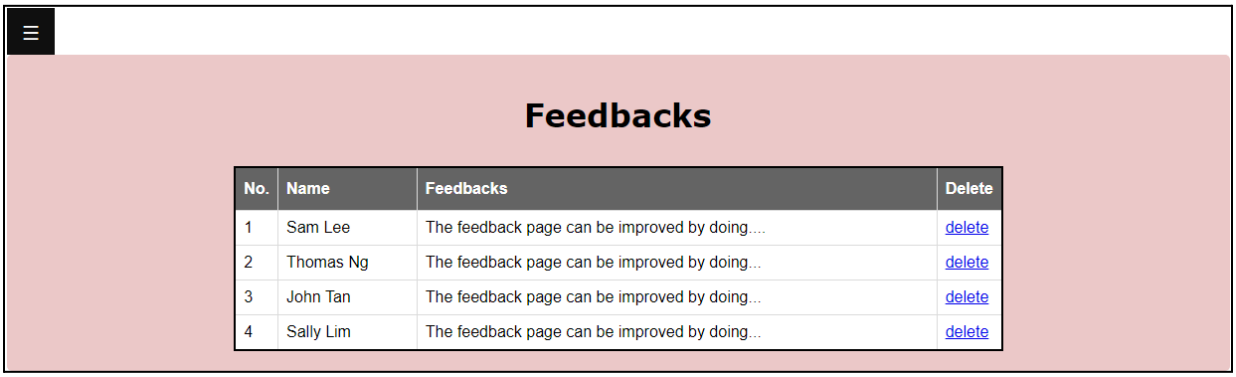
The view feedback page can be accessed through the navigation bar using the “View feedbacks” button. (Only by admins)



The admins will be able to delete unnecessary feedback by clicking on the delete button beside the feedback.

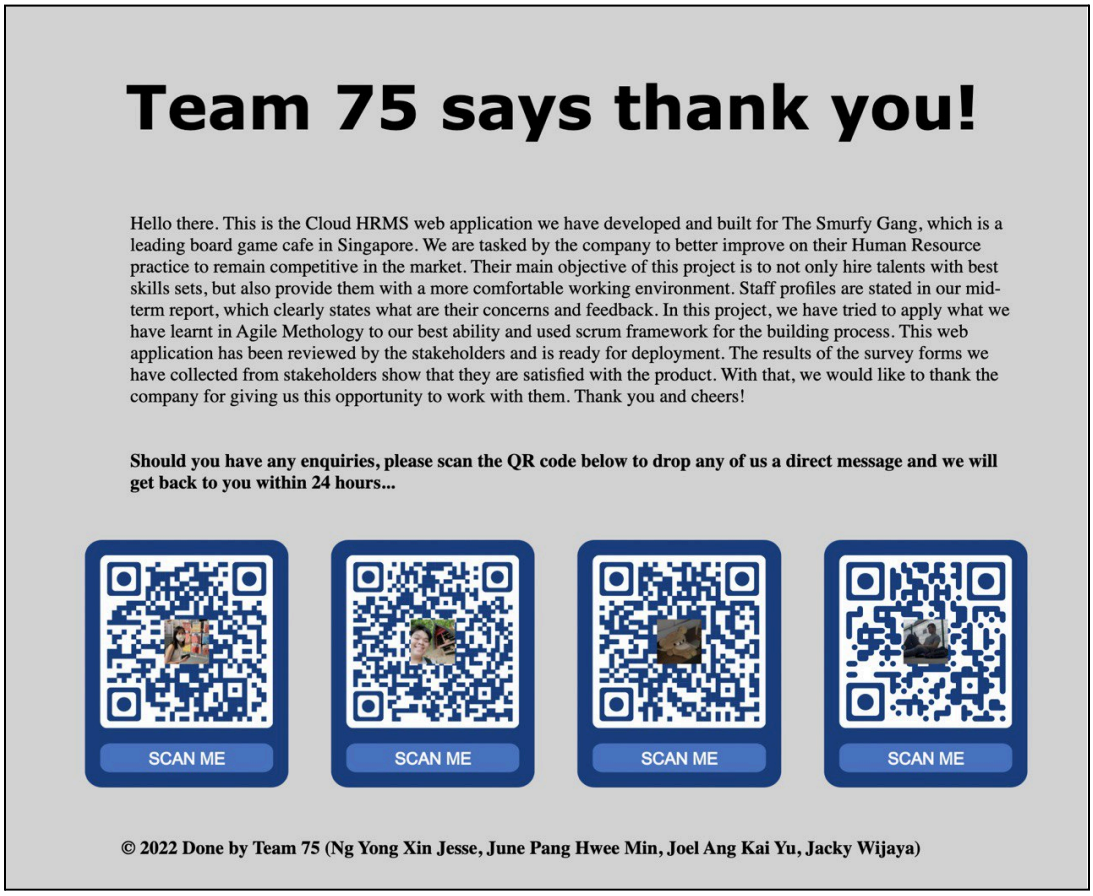


Then, the admin will be prompted if they really want to delete the feedback. If the admin clicks ok, the feedback will be removed from the database.



About Us

Even though we have provided our contact information for maintenance of the web application purpose, our team wanted to show a token of appreciation to the company for this rare opportunity. If they have forgotten or misplaced our contact information. This could be a one stop centre for them to directly contact us via instagram's direct message.



QR Tiger was used to create QR codes that will direct the users to each team member's instagram. We initially proposed the idea of QR code to the stakeholders for time tracking (clock-in and clock-out) but it was turned down as it was not a top priority.

System Development

I. Team Collaboration

As our team is using Agile Methodology for this project, all tasks were split into different parts so that it can be completed in different phases. For example, the login functionality is split into front-end and back-end.

A GitHub Repository was created so that the team is able to work on the code collaboratively. Git is also used as our version control system where we are able to record and track changes made to the code. **(GitHub repository link under Appendix D)**

Our team's web application was built using JavaScript with Node.js, Express.js, HTML, and SQL. They were chosen as all members have experience using them in past projects. Furthermore, JavaScript provides several benefits that we believe will help us in creating a smooth user experience for our users. Those benefits include the ability to create dynamic functions, Cross-Browser Compatibility, ability to use JS libraries and frameworks, and more.

II. Description of the different html pages

PAGE	DESCRIPTION
login.html	Login page that allows users to login to the application using a valid username and password. This is the landing page.
homepage.html	The homepage is displayed right after a successful login. This page displays the logged-in user's name, department, and time of login.
schedule.html	As schedule is still an ongoing discussion with the stakeholders which will be under next deployment, the schedule web page also displays the calendar layout that has buttons for navigation of the months.
allPayslips.html	Displays the selected month's payslip with the list of days worked and the hourly breakdown of each day on one table and another table that calculates their pay for the month.
register.html	Register page for the admin to register a new user by filling in all required fields. (This page is ONLY accessible to users with the "Admin" role)

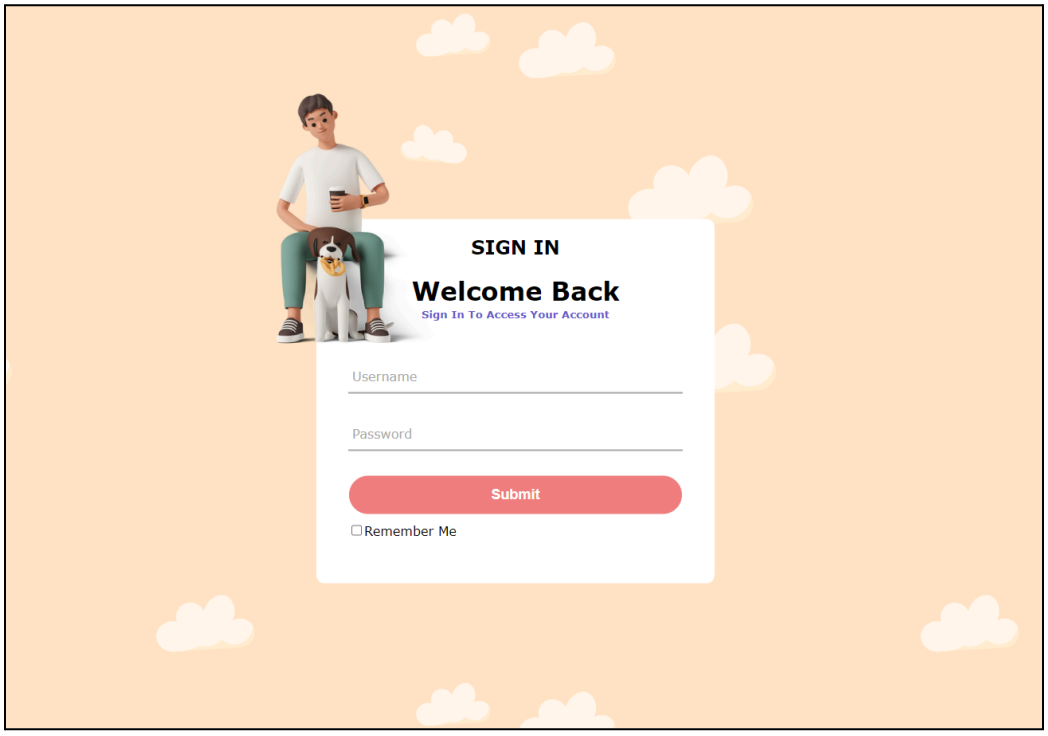
allLeaves.html	Displays all Leave Applications submitted by the logged in user, together with the current status of each leave application.
addLeave.html	Allows the user to create a new leave application by filling in the required fields.
editLeave.html	Clicking on the “edit” button in the “allLeaves” page will bring the user to this page where they can modify the date and reason.
allClaims.html	Displays all Claims Application submitted by the logged in user, together with the current status of the claim. Allows users to choose to create a new claim application, edit and delete from the list of submitted applications.
addClaim.html	Allows the user to create/draft a new claim application by filling in the required fields.
editClaim.html	Allows the user to change the chosen claim’s details to resubmit the claim application. (Users will ONLY be able to choose claim applications that are either “Pending Approval” or “Saved As Draft” to edit.)
feedback.html	Allows the user to submit any feedback they may have about the website.
allFeedbacks.html	Allows the admin to view feedback that has been submitted. (This page is ONLY accessible to users with the “Admin” role)

III. Code Description

All HTML codes can be found in the “views” folder, while all the codes dealing with server side functionality can be found in main.js. Images can be found under public/images.

Login (login.html)

The login page displays a form with the fields “username”, and “password”. This page is displayed whenever the user tries to access any page without logging in (by checking if a session is created).



(login page)

Once the “Submit” button is clicked, the “username” and “password” input will be compared with all the username and passwords stored in the database by using a SELECT statement with a WHERE condition. If the statement returns a result, it means that the username and password is valid, and the session will be created, and users will be redirected to the. If the username and password is invalid, the login page will be displayed again.

```
//GET login page
app.get("/login", function (req, res) {
    req.session.username ? res.render("index.html") : res.render("login.html");
});

app.post('/login', (req, res) => {
    //Check if there is staff with same username and password in database
    let sqlquery = "SELECT * FROM Staff WHERE username = ? AND password = ?";
    let record = [req.body.E_id, req.body.password];
    db.query(sqlquery, record, (err, result) => {
        if (err) {
            res.redirect("/");
        }
        else {
            //Check if there is anything in result
            if (result.length != 0) {
                //Create session
                session = req.session;
                //Add username, staffid and roleid
                session.username = req.body.E_id;
                session.roleid = result[0].Role_id;
                session.staffid = result[0].Staff_id;
                session.loginTime = new Date();
                res.redirect("/");
            }
            else {
                res.redirect("/login");
            }
        }
```

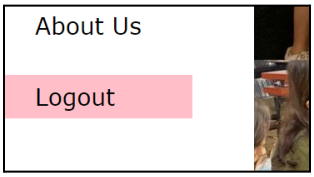
(server side)

```
<h2> SIGN IN</h2>
<h1>Welcome Back</h1>
<h5>Sign In To Access Your Account</h5>
<form action="/login" method="POST" >
    <div class="txt_field">
        <input type="text" name="E_id" placeholder="" required>
        <label>Username</label>
    </div>

    <div class="txt_field">
        <input type="password" name="password" placeholder="" required>
        <label>Password</label>
    </div>

    <input type="submit" style="width: 100%;">

    <div class="checkbox">
        <input type="checkbox" name="remember_me">Remember Me
    </div>

    <img src="/images/boywithdog.png" style="position: absolute; bottom: 350px; right: 350px; width: 40%;">
</form>
```

(code snippet from login.html)

Logout

When the “logout” button on the side menu is clicked, a GET request is made and the session is destroyed. The user is then redirected to the login page.



(logout button)

```
    <a href="/aboutus">About Us</a>
    <a href="/logout" id="logout" class="logout">Logout</a>
</div>
```

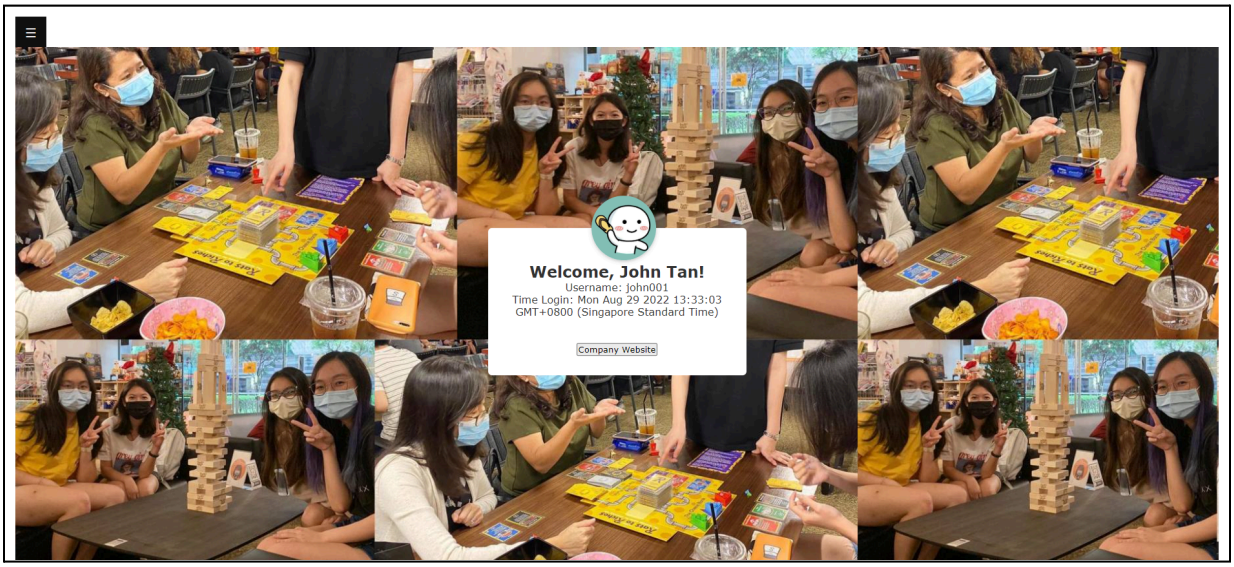
(navigation bar html)

```
app.get('/logout', (req, res) => {
    //Removes session when logging out
    req.session.destroy();
    res.redirect('/');
});
```

(server side - logout GET)

Homepage (index.html)

The homepage displays the logged in user's name, username, and the time of login. This is done by calling a SELECT statement to retrieve the user's name and username where the username matches the username stored in the session. The result of the query is then passed into the homepage to be displayed. The time of login is stored when a session is created upon a successful login.



(Homepage)

```
// GET index
app.get("/",function(req, res) {
  // Use this if else statement to check if user is still logged in
  if (req.session.username) {
    // select username and name
    let sqlquery = "SELECT Staff_name, username FROM Staff WHERE username = ?";

    db.query(sqlquery, req.session.username, (err, result) => {
      if (err) {
        res.redirect("/");
      }
      else {
          //loginTime.setDate(loginTime.getDate() - 1);
          res.render("index.html", { results: result});
      }
    });
  }
  else
      res.redirect("/login");
});
```

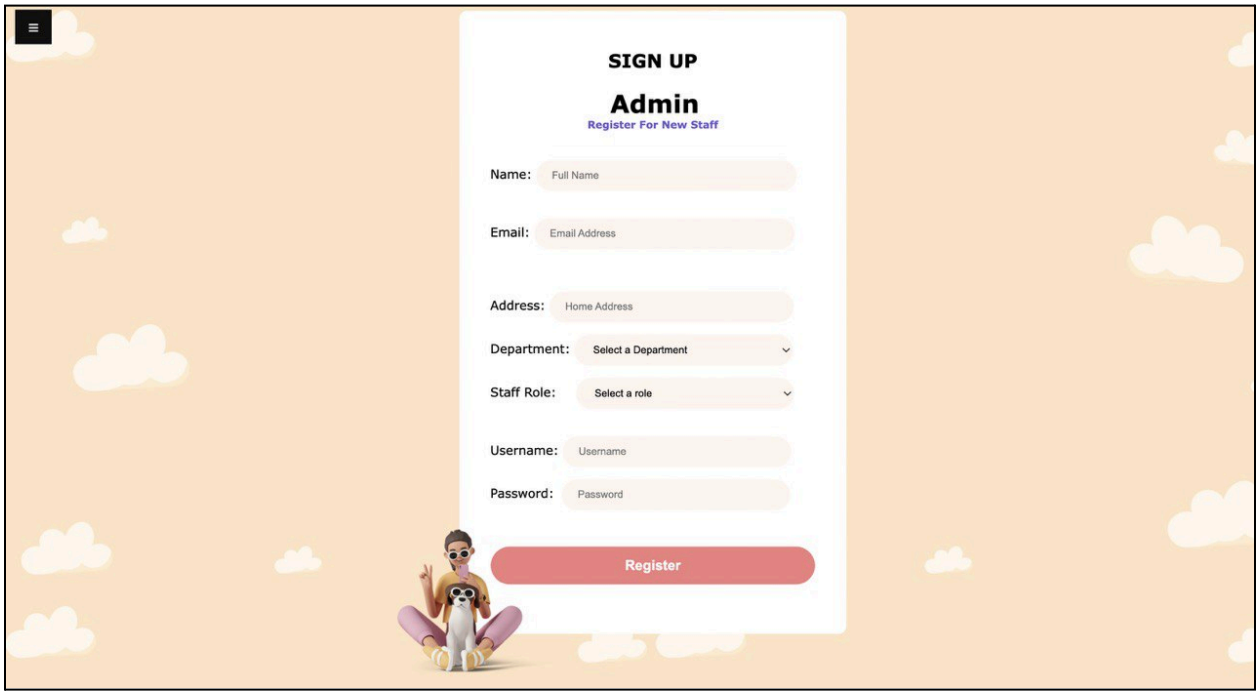
(server side - GET index.html)

```
<div class = "container">
  <% results.forEach(function(result){ %>
  <div class = "popup">
      <img class = "fill" src = "images/avatar.png" width="100%" height="100%">
      <h2>Welcome, <%= result.Staff_name %>!</h2>
      <p>Username: <%= result.username %></p>
      <p>Time Login: <%= session.loginTime %></p>
      <br></br>
      <button type = "button" id = "myButton1">Company Website</button>
  </div>
  <% }) %>
</div>
```

(code snippet from index.html)

Register (register.html)

Several SELECT statements are used to retrieve records from the department and role table to be displayed in the register page, “Department” and “Staff Role” field.



(Register Page)

```
// GET Register page
app.get("/register", function (req, res) {
    if (req.session.username) {
        //check if roles are admin
        if (req.session.roleid == 1) {
            let sqldepartmentquery = "Select * from department";
            let sqlrolequery = "Select * from role";
            db.query(sqldepartmentquery, (err, departmentResult) => {
                if (err) res.redirect("/");
                else {
                    db.query(sqlrolequery, (err, roleResult) => {
                        if (err) res.redirect("/");
                        else {
                            res.render("register.html", { availableDepartment: departmentResult, availableRole: roleResult });
                        }
                    });
                }
            });
        }
        else
            res.redirect("/");
    }
    else
        res.redirect("/login");
});
```

(server side - GET register.html)

```
<h5>Register For New Staff</h5>
<form method="POST" action="/registered">

    <label>Name: </label>
    <input id="name" type="text" name="name" value="" placeholder="Full Name" required /> <br></br>

    <label>Email: </label>
    <input id="email" type="email" name="email" value="" placeholder="Email Address" required /> </b

    <label>Address: </label>
    <input id="address" type="text" name="address" value="" placeholder="Home Address" required /> <

    <label>Department: </label>
    <select id="dept" name="dept" style = "width: 67%; padding: 12px 20px; margin: 8px 0; box-sizing
        <option id = "dept" value="">Select a Department</option>
        <% availableDepartment.forEach(function(department){ %>
        <option value=<%= department.Dept_id %>><%= department.Department_name%></option>
        <% }) %>
    </select>

    <label>Staff Role: </label>
    <select id="role" name="role" style="width: 67%; padding: 12px 20px; margin: 8px 0; box-sizing:
        <option value="">Select A role</option>
        <% availableRole.forEach(function(role){ %>
        <option value=<%= role.Role_id %>><%= role.Role_name%></option>
        <% }) %>
    </select> <br></br>
```

(code snippet from register.html)

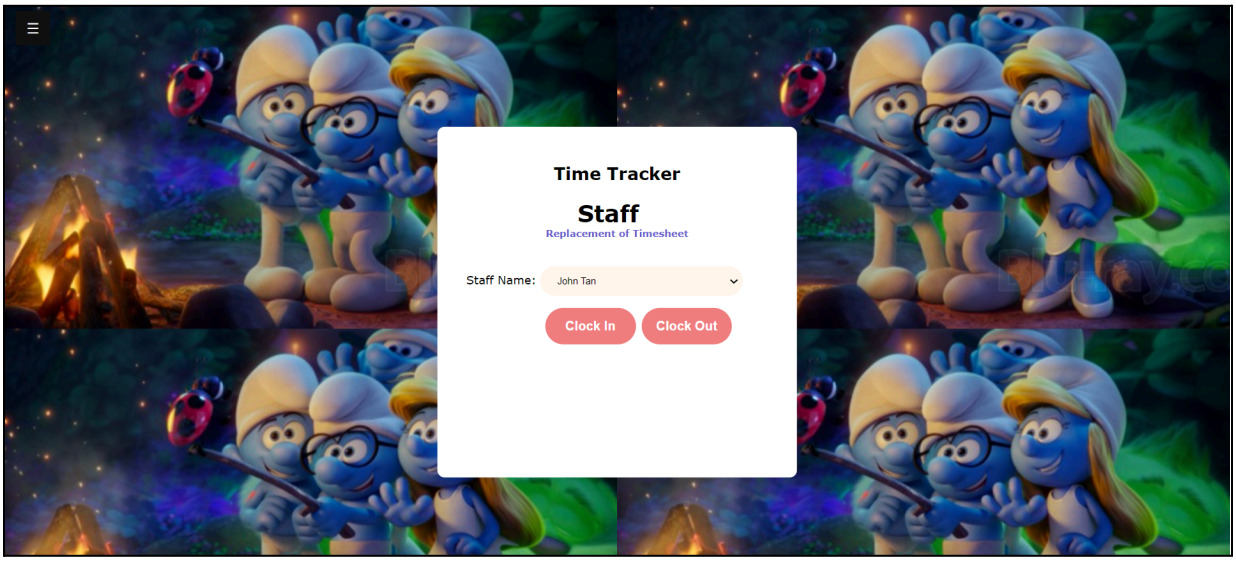
When the “Submit” button is clicked, a post request is made, and an INSERT statement is used to insert the new record into the database.

```
app.post("/registered", function (req, res) {
    // saving data in database
    let sqlquery = "Insert into staff (Dept_id, Role_id, Staff_name, email, address, username, password) values (?,?,?,?,?,?,?)";
    let newrecord = [req.body.dept, req.body.role, req.body.name, req.body.email, req.body.address, req.body.username, req.body.password];
    db.query(sqlquery, newrecord, (err, result) => {
        if (err) res.redirect("/");
        else {
            res.send("Record: " + req.body.dept + " " + req.body.role + " " + req.body.name + " " + req.body.email
                + " " + req.body.address + " " + req.body.username + " " + req.body.password);
        }
    });
});
```

(server side - POST)

Time Tracker (timetracker.html)

The timetracker page contains a dropdown with the name of all the staff, to allow them to select their name and clock in or clock out by clicking on either button. A SELECT statement is used to retrieve all staff names and id, and the result of the query is passed into the page when rendering.



(Time Tracker Page)

```
app.get("/timetracker",function(req, res) {
  if (req.session.username) {
    let sqlquery = "SELECT Staff_id, Staff_name FROM Staff";

    db.query(sqlquery, (err, result) => {
      if (err) {
        res.redirect("/");
      }
      else {
          res.render("timetracker.html", { staffs: result });
      }
    });

  }
  else
      res.render("login.html");
});
```

(server side - GET timetracker.html)

```
<% staffs.forEach(function(name){ %>
<option id = "staffname" value = "<%= name.Staff_id %>"><%= name.Staff_name %></option>
<% }) %>
```

(code snippet from timetracker.html)

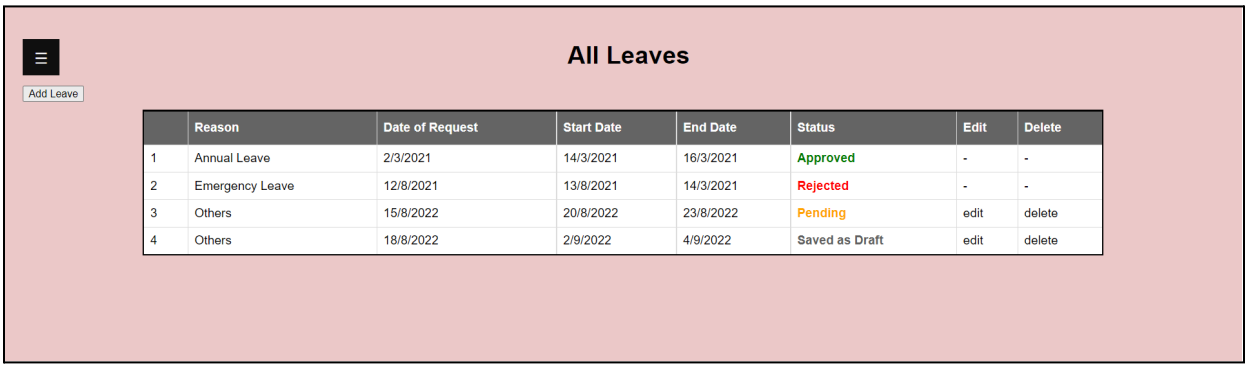
When the “clock in” or “clock out” button is clicked, a post request is sent, and the current time is inserted or updated into the database. An insert statement is used when the “clock in” button is clicked, while an update statement is used when the “clock out” button is clicked, to update the clock out time.

```
// POST timetracked page
app.post("/timetracked", function (req, res) {
    let sqlquery, newrecord;

    // Clock in button pressed
    if (req.body.clockBtn == 'Clock In') {
      sqlquery = "INSERT INTO Punch_Card (Staff_id, date, clock_in_time, clock_out_time, break_hours) VALUES (?,?,?,?,?)";
      newrecord = [req.body.staffName, new Date(), new Date(), "0000-00-00 00:00:00", 1];
    }

    // Clock out button pressed
    else if (req.body.clockBtn == 'Clock Out') {
      sqlquery = "UPDATE Punch_Card SET clock_out_time = ? WHERE clock_out_time = ? AND Staff_id = ?";
      newrecord = [new Date(), "0000-00-00 00:00:00", req.body.staffName];
    }

    db.query(sqlquery, newrecord, (err, result) => {
        if (err) {
          res.redirect("/timetracker");
        }
        else {
          res.redirect("/timetracker");
        }
    });
});
```

(server side - POST)

Leave (allLeaves.html)

For the all leaves page, a select statement is used with a JOIN and WHERE clause to retrieve all the relevant data belonging to the current logged in user.



(All Leaves Page)

```
app.get("/allLeaves", function (req, res) {
  if (req.session.username) {
      let sqlquery =  " SELECT Leave_.Leave_id, Staff.Staff_name, Department.Department_name," +
                      " Leave_Reason.reason, Leave_.date_requested, Leave_.start_date, Leave_.end_date, Leave_.status" +
                      " FROM Leave_" +
                      " JOIN Staff ON Leave_.Staff_id = Staff.Staff_id" +
                      " JOIN Department ON Staff.Dept_id = Department.Dept_id" +
                      " JOIN Leave_Reason ON Leave_.LR_id = Leave_Reason.LR_id" +
                      " WHERE username = ?";

      db.query(sqlquery, req.session.username, (err, result) => {
          if (err) res.redirect("/");
          else {
            res.render("allLeaves.html", { availableLeaves: result });
          }
      });
  }
  else
    res.redirect("/login");
});
```

(server side - GET allLeaves.html)

In the html page, the “date of request”, “start date’ and “end date” which has been passed in from the server side is formatted into dd/mm/yyyy format using JavaScript getdate(), getmonth(), and getyear() functions.

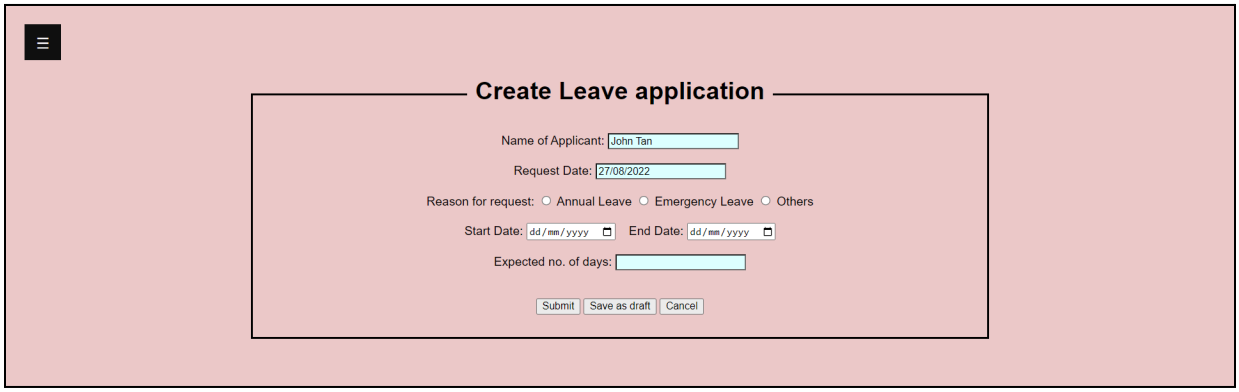
if/ else if/ else statements are also used to display the status in different colours, and to display the “edit” and “delete” buttons only in the rows where the status is “Pending” or “Saved as Draft”.

```
<tbody>
    <% availableLeaves.forEach(function(leave){ %>
    <tr>
        <td></td>
        <!-- <td><%= leave.Staff_name %></td>
        <td><%= leave.Department_name %></td> -->
        <td><%= leave.reason %></td>
        <td><%= (leave.date_requested).getDate(); %>/<%= (leave.date_requested).getMonth() + 1; %>/<%= (leave.date_requested).getFullYear(); %></td>
        <td><%= (leave.start_date).getDate(); %>/<%= (leave.start_date).getMonth() + 1; %>/<%= (leave.start_date).getFullYear(); %></td>
        <td><%= (leave.end_date).getDate(); %>/<%= (leave.end_date).getMonth() + 1 ;%>/<%= (leave.end_date).getFullYear(); %></td>

        <% if (leave.status == 'Approved') {%>
        <td style="color: green;"><b><%= leave.status %></b></td>

        <% } else if (leave.status == 'Rejected') {%>
        <td style="color: red;"><b><%= leave.status %></b></td>

        <% } else if (leave.status == 'Saved as Draft') {%>
        <td style="color: rgb(97, 97, 97);"><b><%= leave.status %></b></td>

        <% } else {%>
        <td style="color: orange;"><b><%= leave.status %></b></td>
        <% } %>

        <!-- Only pending appllication can be edited and deleted -->
        <% if (leave.status == 'Pending' || leave.status == 'Saved as Draft') {%>
        <td>
            <a href="/editLeave?id=<%= leave.Leave_id %>">
                edit
            </a>
        </td>
```

(code snippet from allLeaves.html)

Create Leave (addLeave.html)

A SELECT statement is used to select the name of the logged in user to be displayed in the “Create Leave Application” page. For the “Request Date” field, the current date is displayed in dd/mm/yyyy format. The formatting is done using JavaScript functions. A JavaScript function is also used to calculate the number of days between the “Start Date” and “End Date” selected by the user.



(Create Leave Page)

```
app.get("/addLeave",function(req, res) {
    if (req.session.username) {
        let sqlquery = "SELECT Staff_name FROM Staff WHERE username = ?";

        db.query(sqlquery, req.session.username, (err, result) => {
            if (err) res.redirect("/allLeaves");
            else {
                res.render("addLeave.html", { name: result });
            }
        });
    }
    else
        res.redirect("/login");
});
```

(server side - GET addLeave.html)

```
<form method="POST" action="/addLeave" style="text-align: center;">
  <% name.forEach(function(result){ %>
  <label> Name of Applicant: </label>
  <input type="text" name="applicant" value="<%= result.Staff_name %>" readonly/><br /><br />
  <% }) %>

  <label> Request Date: </label>
  <input type="text" name="requestdate" id="requestdate" readonly/><br /><br />
```

(code snippet from addLeave.html)

```
// get and display today's date in "dd/mm/yyyy" format
const today = new Date().toJSON().slice(0,10).split('-').reverse().join('/');
document.getElementById('requestdate').value = today;

// calculates how many days are between "startdate" and "enddate" chosen
function calcDays() {
    var start = new Date(document.getElementById("startdate").value);
    var end = new Date(document.getElementById("enddate").value);

    return parseInt((end - start) / (24 * 3600 * 1000)) + 1;
}

// set the value of numdays field
function numofdays() {
  if (document.getElementById("enddate")) {
      document.getElementById("numdays2").value = calcDays();
  }
}
```

(code snippet from addLeave.html)

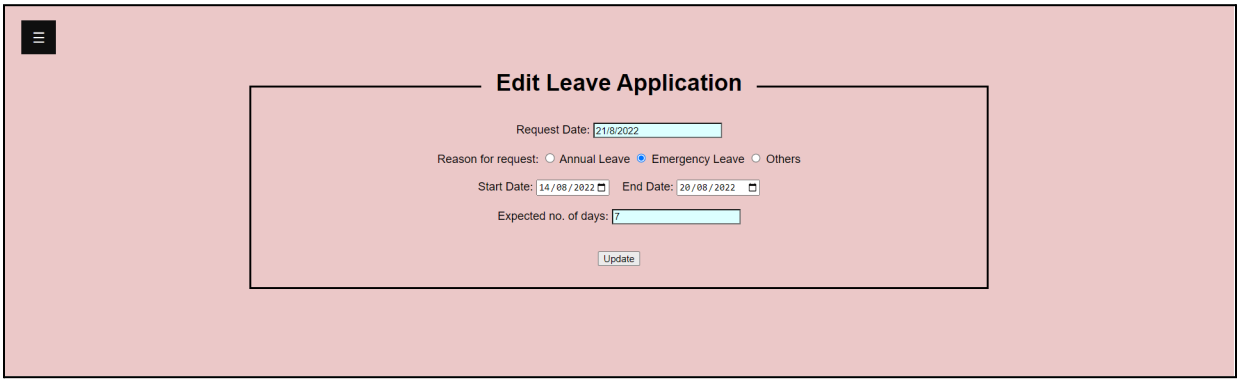
After the user clicks on the “Submit” or “Save as Draft” button, a post request is sent, and the values are inserted into the database. If the “Submit” button is clicked, the status is saved as “Pending”. If the “Save as Draft” button is clicked, the status would be saved as “Saved as Draft”.

```
// POST addLeave page
app.post("/addLeave", function (req, res) {
    if (req.session.username) {
        // set request date to today's date
        const requestdate = new Date();

        var statusVal;
        if (req.body.newLeaveBtn == 'Submit') statusVal = 'Pending';
        else if (req.body.newLeaveBtn == 'Save as draft') statusVal = 'Saved as Draft';

        let sqlquery = "INSERT INTO Leave_ (Staff_id, LR_id, date_requested, start_date, end_date, status) VALUES (?,?,?,?,?,?)";
        let temprecord = [session.roleid, req.body.requestreason, requestdate, req.body.startdate, req.body.enddate, statusVal]

        // execute sql query
        db.query(sqlquery, temprecord, (err, result) => {
            (err) ? res.redirect("/addLeave") : res.redirect("/allLeaves");
        });
    }

    else
        res.redirect("/login");
})
```

(server side - POST)

Modify Leave (editLeave.html)

A SELECT statement is used to retrieve all relevant information to pre-populate the edit page with data previously submitted. In the html file, if/ else if/ else statements are used to check the correct radio button for the “reason for request” according to the value retrieved from the database.



(Edit Leave page)

```
// GET editLeave page
app.get("/editLeave", function (req, res) {
  if (req.session.username) {
      let sqlquery =  " SELECT Leave_.Leave_id, Staff.Staff_name, Department.Department_name," +
                      " Leave_Reason.reason, Leave_.date_requested, Leave_.start_date, Leave_.end_date, Leave_.status" +
                      " FROM Leave_" +
                      " JOIN Staff ON Leave_.Staff_id = Staff.Staff_id" +
                      " JOIN Department ON Staff.Dept_id = Department.Dept_id" +
                      " JOIN Leave_Reason ON Leave_.LR_id = Leave_Reason.LR_id" +
                      " WHERE Leave_id = ?";

      db.query(sqlquery, req.query.id, (err, result) => {
          if (err) {
              res.redirect("/allLeaves");
          }
          else {
              res.render("editLeave.html", { availableLeaves: result });
          }
      });
  }

  else
      res.redirect("/login");
});
```

(server side - GET editLeave.html)

```
<label>Reason for request: </label>
<% if (leave.reason == 'Annual Leave') {%>
<input type="radio" id="requestreason" name="requestreason" value="1" checked required/>
<label for="Annual Leave">Annual Leave</label>
<input type="radio" id="requestreason" name="requestreason" value="2" />
<label for="Emergency Leave">Emergency Leave</label>
<input type="radio" id="requestreason" name="requestreason" value="3" />
<label for="Others">Others</label><br><br>

<% } else if (leave.reason == 'Emergency Leave') {%>
  <input type="radio" id="requestreason" name="requestreason" value="1" required/>
  <label for="Annual Leave">Annual Leave</label>
  <input type="radio" id="requestreason" name="requestreason" value="2" checked />
  <label for="Emergency Leave">Emergency Leave</label>
  <input type="radio" id="requestreason" name="requestreason" value="3" />
  <label for="Others">Others</label><br><br>
```

```
<% } else {%>
  <input type="radio" id="requestreason" name="requestreason" value="1" required/>
  <label for="Annual Leave">Annual Leave</label>
  <input type="radio" id="requestreason" name="requestreason" value="2" />
  <label for="Emergency Leave">Emergency Leave</label>
  <input type="radio" id="requestreason" name="requestreason" value="3" checked />
  <label for="Others">Others</label><br><br>
<% } %>

<label for="startdate">Start Date: </label>
<input type="date" name="startdate" id="startdate" value="<%= leave.start_date %>" >
  
<label for="enddate">End Date: </label>
<input type="date" name="enddate" id="enddate" value="<%= leave.end_date %>" ><br><br>
```

(code snippets from editLeave.html)

Once the “Update” button is clicked, a POST request is sent, and an UPDATE statement is used to update the database with the new values.

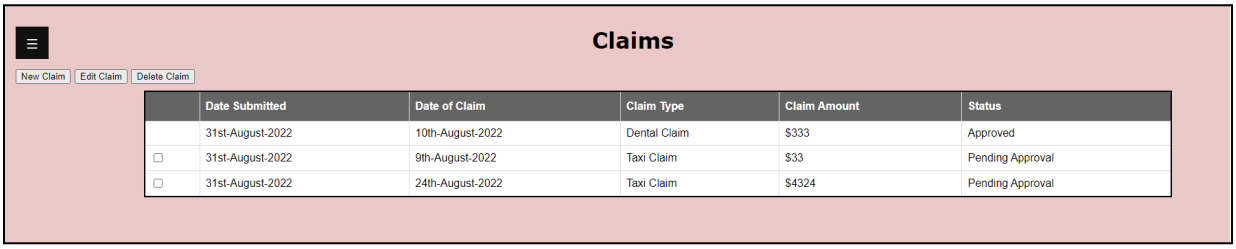
```
app.post("/editLeave", function (req, res) {
    if (req.session.username) {
        let updateLeave = "UPDATE Leave_ SET LR_id = ?, start_date = ?, end_date = ? WHERE Leave_id = ?";
        var leaveParam = [req.body.requestreason, req.body.startdate, req.body.enddate, req.body.leaveid]

        db.query(updateLeave, leaveParam, function (err, result) {
            if (err) {
                res.redirect("/allLeaves");
            }
            else {
                res.redirect("/allLeaves");
            }
        })
    }

    else
        res.redirect("/login");
})
```

(server side - POST)

Claims (allClaims.html)
When you load the claims page a SELECT query will be called to populate the rows in the table below.

When clicking either “Edit Claim” or “Delete Claim” the javascript will check if there are no boxes checked, or in the case for “Edit Claim” button being clicked more than one box checked, if any of them are true a alert pop-up message will appear else the correct html file will be rendered with the information from a SELECT query.

For delete it will loop through all the checked checkboxes and for each one a DELETE query would be made for each of them based on their claim id.



(Claims Page)

```
document.getElementById("editClaim").addEventListener("click", function (event) {
    var checkboxes = document.getElementsByClassName("checkme");
    var numIsChecked = 0;
    for (let i = 0; i < (checkboxes.length - 1); i++) {
        if (checkboxes[i].checked)
            numIsChecked++;
    }
    if (numIsChecked == 0) {
        alert("Please select a claim to edit.");
        event.preventDefault()
    }
    if (numIsChecked > 1) {
        alert("Please select only one claim to edit");
        event.preventDefault()
    }
});

document.getElementById("deleteClaim").addEventListener("click", function (event) {
    var checkboxes = document.getElementsByClassName("checkme");
    var isChecked = false;
    for (let i = 0; i < (checkboxes.length - 1); i++) {
        if (checkboxes[i].checked)
            isChecked = true;
    }
    if (!isChecked) {
        alert("Please select at least one claim to delete.");
        event.preventDefault();
    }
    else {
        if (!confirm("Are you sure you want to delete")) {
            event.preventDefault();
        }
    }
});
```

(js for allClaims.html)

```
//GET Claims Page
app.get("/allClaims", function (req, res) {
    if (req.session.username){
        let sqlquery = "SELECT Claim.claim_id, DATE_FORMAT(Claim.date_submitted, '%D-%M-%Y') AS date_submitted," +
            " DATE_FORMAT(Claim.date_of_claim, '%D-%M-%Y') AS date_of_claim, Claim_Type.type, Claim.claim_amount, Claim.status" +
            " FROM Claim" +
            " JOIN Staff ON Claim.Staff_id = Staff.Staff_id" +
            " JOIN Claim_Type ON Claim.CT_id = Claim_Type.CT_id" +
            " WHERE Staff.Staff_id = ?";
        let staffid = [req.session.staffid];
        db.query(sqlquery, staffid, (err, result) => {
            if (err)
            {
                res.redirect("/");
            }
            else
                res.render("allClaims.html", { availableClaims: result });
        });
    }
    else
        res.redirect("/login");
});
```

(GET allClaims.html)

```
app.post("/allClaims", function (req, res) {
    let buttonValue = req.body.claimButtons;
    if (buttonValue == 'New Claim' || buttonValue == 'Edit Claim') {
        let sqlquery = "SELECT Staff.Staff_id, Staff.Staff_name, Department.Department_name" +
            " From Staff" +
            " JOIN Department ON Staff.Dept_id = Department.Dept_id" +
            " WHERE Staff_id = ?";
        let staffid = [req.session.staffid];
        db.query(sqlquery, staffid, (err, staffDetail) => {
            if (err) res.redirect("/allClaims");
            else {
                let sqlclaimquery = "Select * from Claim_type";
                db.query(sqlclaimquery, (err, claimType) => {
                    if (err) res.redirect("/allClaims");
                    else {
                        if (buttonValue == 'New Claim') {
                            res.render('newClaim.html', { staffDetails: staffDetail, claimTypes: claimType });
                        }
                        if (buttonValue == 'Edit Claim') {
                            let sqlclaimdetailquery = "SELECT Claim.claim_id, DATE_FORMAT(Claim.date_of_claim, '%Y-%m-%d') AS date_of_claim," +
                                                      " Claim.CT_id, Claim.claim_amount" +
                                                      " FROM Claim" +
                                                      " WHERE Claim.claim_id = ?";
                            let claimsid = [req.body.checkbox[0]];
                            db.query(sqlclaimdetailquery, claimsid, (err, claimDetail,) => {
                                if (err) {
                                    res.redirect("/allClaims");
                                }
                                else {
                                    res.render('editClaim.html', { staffDetails: staffDetail, claimTypes: claimType, claimDetails: claimDetail })
                                }
                            });

                        }
                    }
                });
            }
        });
    }
```

(POST allClaims.html(1))

```
    else if (buttonValue == 'Delete Claim') {
        //Get all checked checkboxes
        let claimsid = req.body.checkbox;
        let sqlquery = "DELETE FROM Claim WHERE Claim_id = ? and Staff_id = ? "
        //Loop Through claimsid to delete selected claims
        for (let i in claimsid) {
            let newrecord = [claimsid[parseInt(i)], req.session.staffid];
            db.query(sqlquery, newrecord, (err, result) => {
                if (err) {
                    res.redirect("/");
                }
            });
        }
        res.redirect("/allClaims");
    }
    else {
        res.redirect("/allClaims");
    }
});
```

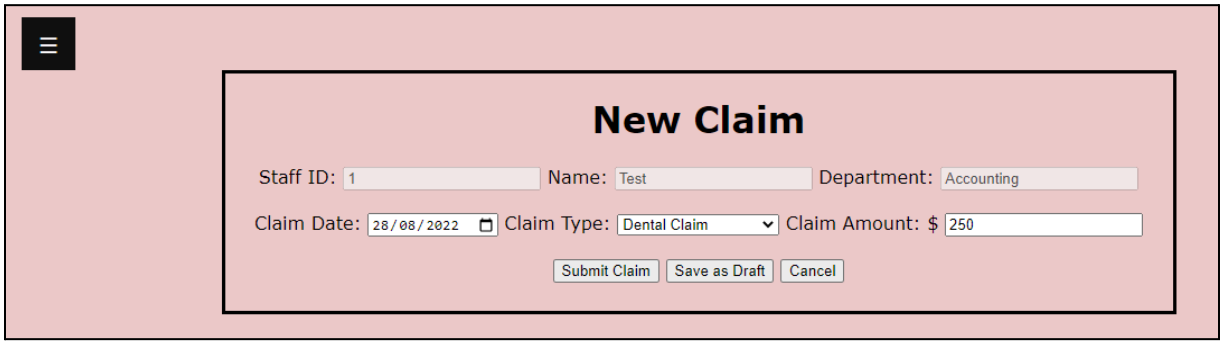
(POST allClaims.html(2))

Add Claim (newClaim.html)

When a “New Claim” button is clicked a SELECT query will be called to fill in the details for Staff ID, Name, Department and the list of Claim Type.

When either “Submit Claim” or “Save as Draft” is clicked the form will do a validation check to see if the fields Claim Date, Claim Type and Claim Amount is filled or it will not submit the form, else it will submit the form and set the status as either “Pending Approval” or “Saved as Draft” based on which button is clicked.

If the “Cancel” button is clicked the form will not be submitted and will go back to the allClaims page.



(New Claim Page)

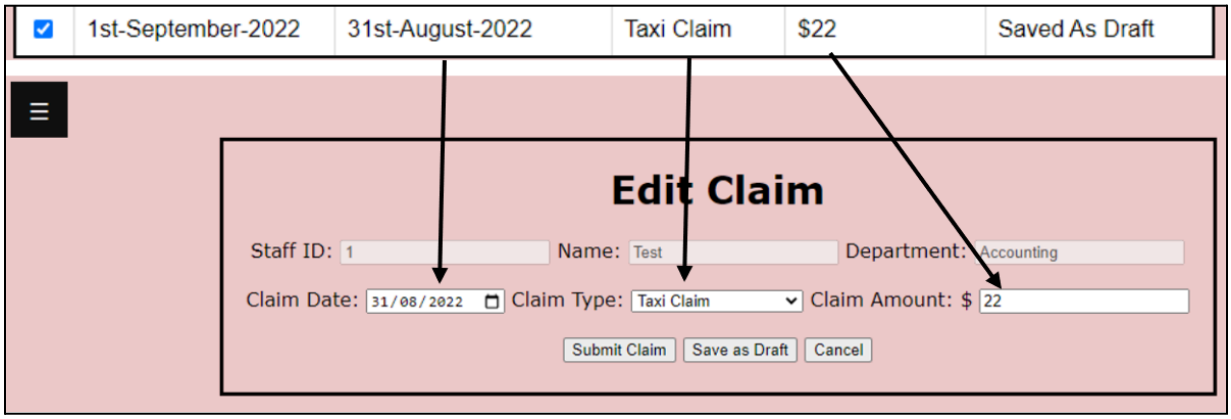
```
app.post("/newClaims", function (req, res) {
    const dateOfClaim = req.body.date + " 00:00:00";

    let date_ob = new Date();
    let date = date_ob.getDate();
    let month = date_ob.getMonth() + 1;
    let year = date_ob.getFullYear();
    let fullDate = year + "-" + month + "-" + date + " 00:00:00";
    let values;
    let sqlquery = "INSERT INTO Claim (CT_id, Staff_id, date_submitted, claim_amount, date_of_claim, status) values (?,?,?,?,?,?)";

    if (req.body.newClaimButtons == "Submit Claim") {
        values = [req.body.claimType, req.body.staffID, fullDate, req.body.claimAmount, dateOfClaim, "Pending Approval"];
    }
    if (req.body.newClaimButtons == "Save as Draft") {
        values = [req.body.claimType, req.body.staffID, fullDate, req.body.claimAmount, dateOfClaim, "Saved As Draft"];
    }
    if (req.body.newClaimButtons == "Cancel") {
        res.redirect("/allClaims");
        return;
    }
    db.query(sqlquery, values, (err, result) => {
        if (err) {
            res.redirect("/");
        }
        else {
            res.redirect("/allClaims");
        }
    });
});
```

(POST newClaim.html)

Modify Claim (editClaim.html)

When a claim is selected to be edited, a SELECT query based on the claim id will be used to populate the fields in the edit claim page.

When either “Submit Claim” or “Saved as Draft” button is clicked a validation would run and check if all fields are filled, else the form would do a UPDATE query based on the claim id and update the claim’s information and depending on which button is click the status would update to either “Pending Approval” or “Saved as Draft”.



(Edit Claim Page)

```
app.post("/editClaims", function (req, res) {
    const dateOfClaim = req.body.date + " 00:00:00";

    let date_ob = new Date();
    let date = date_ob.getDate();
    let month = date_ob.getMonth() + 1;
    let year = date_ob.getFullYear();
    let fullDate = year + "-" + month + "-" + date + " 00:00:00";
    let values;
    let sqlquery = "UPDATE Claim SET CT_id = ?, date_submitted = ?, claim_amount = ?, date_of_claim = ?, status = ? " +
                   "WHERE claim_id = ?";

    if (req.body.newClaimButtons == "Submit Claim") {
        values = [req.body.claimType, fullDate, req.body.claimAmount, dateOfClaim, "Pending Approval", req.body.claim_id];
    }
    if (req.body.newClaimButtons == "Save as Draft") {
        values = [req.body.claimType, fullDate, req.body.claimAmount, dateOfClaim, "Saved As Draft", req.body.claim_id];
    }
    db.query(sqlquery, values, (err, result) => {
        if (err) {
            res.redirect("/allClaims");
        }
        else {
            res.redirect("/allClaims");
        }
    });
});
```

(POST editClaim.html)

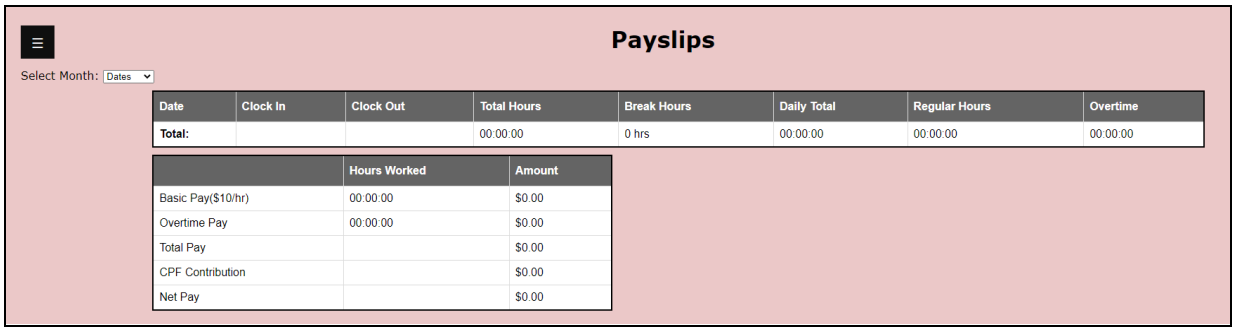
View Payslips(allPayslips.html)

When payslips page is called a SELECT query to fill the select month dropdown list. When selecting a date to view the payslip the table will update the rows using a SELECT query based on the staff id and the selected date.

The second table will calculate the pay using javascript based on the total of the top table.

```
function calculatePay() {
    let totalBasicPayHours = document.getElementById("basicPayHours");
    var TBPH = totalBasicPayHours.innerHTML.split(':');
    var TBPHAmount = ((Number(TBPH[0])) + (Number(TBPH[1]) * (1 / 60)) + (Number(TBPH[2]) * (1 / 3600))) * 10;
    document.getElementById("basicPayAmount").innerHTML = "$" + TBPHAmount.toFixed(2);

    let totalOvertimeHours = document.getElementById("overtimePayHours");
    var TOPH = totalOvertimeHours.innerHTML.split(':');
    var TOPHAmount = ((Number(TOPH[0])) + (Number(TOPH[1]) * (1 / 60)) + (Number(TOPH[2]) * (1 / 3600))) * 10;
    document.getElementById("overtimePayAmount").innerHTML = "$" + TOPHAmount.toFixed(2);

    var totalPayAmount = TBPHAmount + TOPHAmount;
    document.getElementById("totalPayAmount").innerHTML = "$" + totalPayAmount.toFixed(2);

    var CPFAmount = totalPayAmount * 0.2;
    if (CPFAmount == 0)
        document.getElementById("CPFAmount").innerHTML = "$" + CPFAmount.toFixed(2);
    else
        document.getElementById("CPFAmount").innerHTML = "$" + CPFAmount.toFixed(2);

    var netPayAmount = totalPayAmount - CPFAmount;
    document.getElementById("netPayAmount").innerHTML = "$" + netPayAmount.toFixed(2);

}

function convert(sec_num) {
    var hours = Math.floor(sec_num / 3600);
    var minutes = Math.floor((sec_num - (hours * 3600)) / 60);
    var seconds = sec_num - (hours * 3600) - (minutes * 60);
    if (hours < 10) { hours = "0" + hours; }
    if (minutes < 10) { minutes = "0" + minutes; }
    if (seconds < 10) { seconds = "0" + seconds; }
    return hours + ':' + minutes + ':' + seconds;
}
```

(js for calculating the second table in allPayslips.html)



(Payslips Page)

```
app.get("/allPayslips", function (req, res) {

    if (req.session.username) {
        let values = [req.session.staffid];
        sqlquery2 = "SELECT DISTINCT(DATE_FORMAT(date, '%m %Y')) AS Date FROM punch_card WHERE Staff_id = ? ORDER BY date ASC"
        db.query(sqlquery2, values, (err, DDLDates) => {
            if (err) {
                res.redirect("/");
            }
            else {
                res.render("allPayslips.html", { allPunchCards: [], allDropdownDates: DDLDates });
            }
        });
    }
    else
        res.render("login.html");

});
```

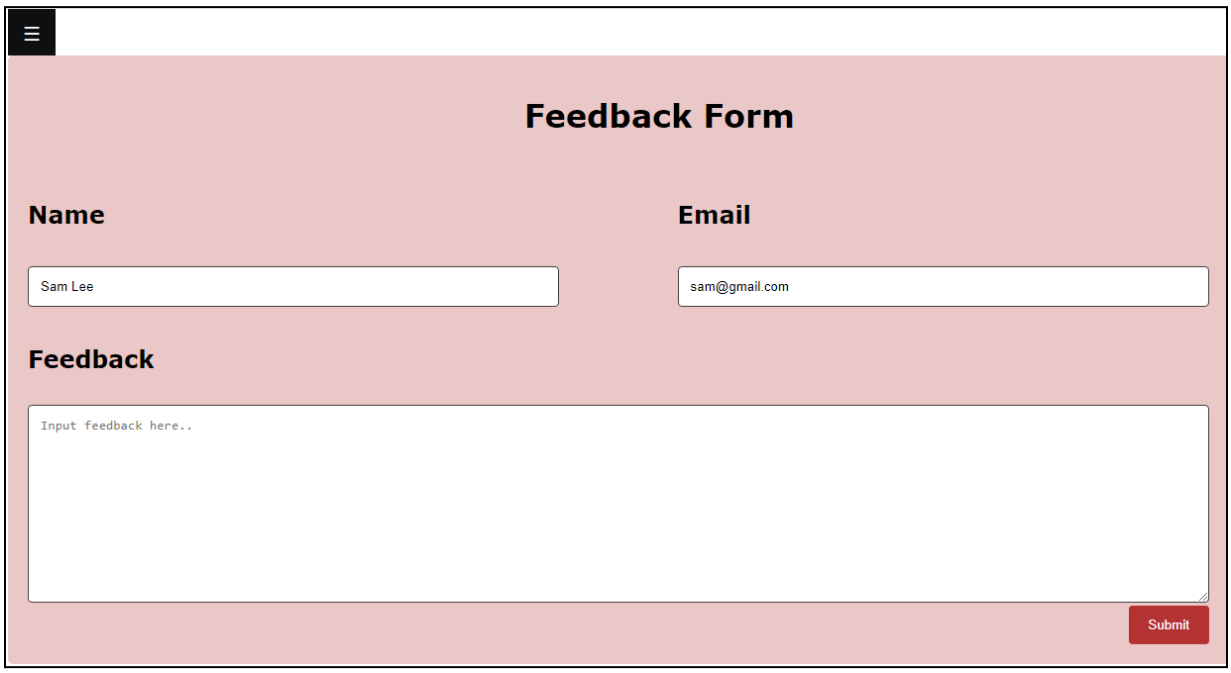
(GET allPayslips.html)

```
app.post("/allPayslips", function (req, res) {

    if (req.session.username) {
        let MY = req.body.DDLDates.trim().split(/\s+/);
        let values = [req.session.staffid, MY[0], MY[1]];
        let sqlquery1 = "SELECT PC_id, date, CAST(clock_in_time AS TIME) AS CIT, CAST(clock_out_time AS TIME) AS COT," +
                        " break_hours, TIMEDIFF(clock_out_time,clock_in_time) AS TotalHours" +
                        " FROM punch_card" +
                        " WHERE staff_id = ? AND MONTH(date) = ? AND YEAR(date) = ?" +
                        " ORDER BY date ASC";
        db.query(sqlquery1, values, (err, PCDates) => {
            if (err) {
                res.redirect("/");
            }
            else {
                sqlquery2 = "SELECT DISTINCT(DATE_FORMAT(date, '%m %Y')) AS Date FROM punch_card WHERE Staff_id = ? ORDER BY date ASC"
                db.query(sqlquery2, values, (err, DDLDates) => {
                    if (err) {
                        res.redirect("/");
                    }
                    else {
                        let sqlquery3 = "SELECT SEC_TO_TIME(sum(TIME_TO_SEC(clock_out_time) - TIME_TO_SEC(clock_in_time))) AS TotalTimeDiff," +
                                        " SUM(break_hours) AS TotalBreakHours, FROM punch_card WHERE MONTH(date) = ?"
                        res.render("allPayslips.html", { allPunchCards: PCDates, allDropdownDates: DDLDates });
                    }
                });
            }
        });
    }
    else
        res.render("login.html");

});
```

(POST allPayslips.html)

Submit Feedback (feedback.html)

The feedback form has the name and email automatically filled in by using a SELECT query to retrieve the data from SQL. To submit the feedback, an INSERT query will be used to submit the user's name, email and feedback.



(Feedback form)

```
app.get("/feedback", function (req, res) {

    if (req.session.username) {
        let sqlquery = "Select * FROM staff WHERE Staff_id = ?";
        let staffid = [req.session.staffid];

        db.query(sqlquery, staffid, (err, result) => {
            if (err) {
                res.redirect("/");
            }
            else
                res.render("feedback.html", { staffDetails: result });
        });
    }
    else
        res.render("login.html");

});
```

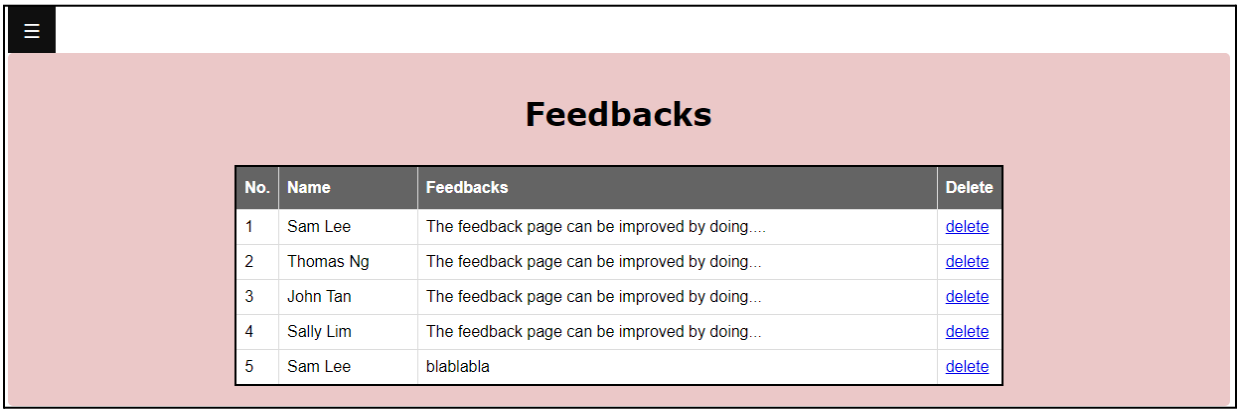
(GET feedback.html)

```
app.post("/feedback_submitted", function (req, res) {
    // saving data in database
    if (req.session.username) {
        let sqlquery = "Insert into feedback (name, email, feedback) values (?,?,?)";
        let newrecord = [req.body.name, req.body.email, req.body.feedback];
        db.query(sqlquery, newrecord, (err, result) => {
            if (err) res.redirect("/");
            else {
                let messages = "Feedback has been submitted";
                res.redirect("/feedback");
            }
        });
    }
    else
        res.render("login.html");
});
```

(POST feedback_submitted)

View Feedbacks (allFeedbacks.html)

The view feedback page shows all the feedback that has been submitted to the database. This is done by using a SELECT query to retrieve all the feedback and names from the database. To delete the feedback from the database, a DELETE query is used based on the Feedback_id submitted to the database.



(View feedback page)

```
app.get("/allFeedbacks", function (req, res) {

    if (req.session.username) {
        let sqlquery = "Select * FROM feedback";

        db.query(sqlquery, (err, result) => {
            if (err) {
                res.redirect("/");
            }
            else
                res.render("allFeedbacks.html", { allFeedbacks: result });
        });
    }
    else
        res.render("login.html");

});
```

(GET allFeedbacks.html)

```
app.get("/deleteFeedback", function (req, res) {
    if (req.session.username) {
        let sqlquery = "DELETE FROM Feedback WHERE Feedback_id = ?";

        db.query(sqlquery, req.query.id, (err, result) => {
            res.redirect("/allFeedbacks");
        });
    }

    else
        res.redirect("/login");
});
```

(GET deleteFeedback)

```
<td>
    <a href="/deleteFeedback?id=<%= feedbacks.Feedback_id %>"
      onclick="return confirm('Are you sure you want to delete the feedback?')">
       delete
    </a>
    </td>
```

(Delete Feedback from HTML)

About Us (aboutus.html)

The design of the web page follows the “minimalist” style that we wanted to achieve for this project. The copyright footer is for credibility purposes, to show that this is done by us and they should not be afraid that scanning the QR code would be untrustworthy. Such that it directs them to a scammer’s website.

CSS was used for styling to achieve a more organised web page.

Code snippets from aboutus.html:

```
<style>

    @import url('https://fonts.googleapis.com/css?family=Allura|Josefin+Sans');

    *
    {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        background-color: lightgrey;



    }

    body
    {
        background: pink;
        background-image: url("images/panda,png");
        font-family: 'Verdana, Geneva, Tahoma, sans-serif';
    }

    .wrapper
    {
        margin-top: 10%;

    }
```

```
.wrapper h1
{
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 52px;
    margin-bottom: 60px;
    text-align: center;
    margin-top: -50px;
    margin-right: 150px;


}

/* team 75 wordings */
.team
{
    display: flex;
    text-align: center;
    width: auto;
    justify-content: center;
    width: 70%;
    margin-left: 50px;
    position: relative;
}
```

```css
.team .team_member h3
{
    color: pink;
    font-size: 26px;
    margin-top: 50px;

    text-align: center;


}

    /* text-align:end; */
    .team .team_member p.role
    {
        margin-left: 420px;
    }


.wrapper p
{
    width: 50%;
    text-align: left;
    margin-left: 320px;

}


.team_member p
{
    right: 550px;
}
```

```css
.copyright h4
{
    text-align: center;
    margin-left: 150px;
}
```

On the server side, the about us page is rendered if the user is logged in. Otherwise, the login page will be rendered.

```
// GET about us page
app.get("/aboutus", function (req, res) {
    req.session.username ? res.render("aboutus.html") : res.render("login.html");
});
```

(server side - GET about us page)

Analysis

Info-tech is a well-known HRMS mobile application in Singapore.

It is more robust, as it has:

1. GPS to help manage attendance using face recognition features to ensure staff clock in at the right time and place.
2. The functions include a leave application form for manager's approval.
3. Claim form that allows users to snap a picture of the receipts for manager's approval
4. Payslips are fully compliant with the CPF board, IRAS and MOM.
5. Salary payments via bank or GIRO are automated

	STRENGTH	**WEAKNESS**
Our HRMS web application	It is robust and secure, just like Info-tech as it gets rid of book-keeping and tedious manual work. It has the leave application form for manager's approval and claim form that allows users to upload file(s) such as travel123.png. Their payroll is auto-calculated and salary is automated that follows the organisation's pay day (every 7th of the month).	We do not have: (1) The GPS and face recognition function to ensure the staff is at the workplace when they clock in and clock out of their shifts. The GPS and face recognition ensure staff do not clock in or out on behalf of their colleagues and the data collected is accurate. We will discuss with the stakeholders again for the second deployment.

(Above screenshot describes our analysis)

Reference: https://www.youtube.com/watch?v=ki13YjOkjdA

During the scrum review, we handed the customer satisfaction survey to our stakeholders and most of them rated our final product a 7 to 9 out of 10. They liked how the web pages are easy to understand and navigate to different web pages but feedback to us that the design could be further improved and synchronised such that it fits the entire application nicely.

Customer Satisfaction Survey:
https://docs.google.com/forms/d/e/1FAIpQLSeIxRMyNFnzdJqnffqvodmY6aiEXtrKiZgh83sBoDZ4vLOwog/viewform?usp=sf_link

(All survey question and responses can be found under **Appendix E**)

Evaluation

As we are a four member team, and our application consists of 9 main components (login/logout, homepage, time tracker , register, leave application, claims, payslip, feedback, about us), we had to divide the tasks such that each member has to work on multiple functions. If any member encounters issues due to unfamiliarity with the code, they are to voice it out immediately to the team so that another member with more experience can provide guidance, or take over the task. This is to ensure that no one will struggle silently, and we can make full use of the time we have.

Throughout the development of the application, we made sure to constantly get feedback from our stakeholders so that we are able to produce an application that most satisfies the requirements of the stakeholders.

Overall, the HRMS application we have made is satisfactory. We managed to implement most of the functionalities that we had mentioned in our mid-term proposal, and even managed to spot several components that we failed to take into consideration for in our mid-term proposal.

On the other hand, there are definitely areas of our application which we can improve on. For example, the login page currently redisplays whenever an invalid username and/or password is entered. This can be improved by displaying an error message whenever a wrong username or password is entered.

Functionalities that we wish to implement in the future includes:

- Back-end functionality of the schedule page (allow users to select working dates through the calendar).
- Implement GPS checking such that users can only click on “Check in” and “Check out” when their GPS location matches the workplace location.
- Implement functionality that allows the admin to reply to feedback submitted by staff, so that the staff can read the reply and be assured that their feedback is taken seriously.

SWOT Analysis

SWOT Analysis	
Strengths	Weaknesses
• Our application has a simple user interface that is easy to use and navigate even for users who are not as tech-savvy. • As the HRMS application we have created is aimed toward employees of 'The Smurfy Gang', it is easier to customise the application according to their liking. for example, using images of a cute and playful theme, which is in line with the name of the company.	• We did not implement error messages wherever a wrong action is done. For example, no error message when invalid password is entered. This might cause confusion for users who are not familiar with the system. • Since the application would be used for almost all HR processes, if there were to be a service disruption, or if the cloud server is down, users would not be able to use the application.
Opportunities	Threats
• Processes that were once manual are now all automated. For example, calculation of payslip. As mentioned in the mid-term proposal, this can help to cut costs on hiring manpower to manage HR processes, and make data more accessible since they are all stored in the cloud.	• There are already many HRMS applications currently available in the market. Hence, it will not be easy for our application to stand out among the rest.

Conclusion

Overall, the HRMS web application created by our team is able to fulfil the purpose of making HR operations more efficient though different functions such as viewing payslip, applying for leave and claims, scheduling of work schedule.

With this HRMS application in place, our stakeholders, The Smurfy Gang, will benefit greatly as most of their manual processes such as calculating payroll, can be automated through the application. This not only brings convenience to both the staff and the management, it will also reduce the workload of the managers who were in charge of these processes previously. This allows more time and freedom to explore other business prospects to improve their business.

Throughout this project, one of the bigger challenges we faced as a team was poor planning and lack of communication initially, which led to us having to rush through the first quarter of the coding process. When our team first started coding the application, we only decided on which tasks to work on, but did not specifically assign each task to a member. Hence, no one worked on it until later on where we all thought someone was working on it when in reality, no one was doing it. That caused us to be behind schedule and had to rush. However, it was a great learning opportunity for us as we did not make the same mistake anymore. After that mistake, we made sure to update each other frequently, and everyone took the initiative to take up more tasks after finishing the task on hand.

We were also too ambitious when we created our high fidelity prototype previously. After we started the actual coding process, we realised that some components were beyond our coding knowledge. So we had to take extra time to bridge the gap. We also overestimated the amount of time we had, as we did not take into consideration the time required to write the report on top of coding and testing of the application. We overcame this by setting our priorities clearly. The team agreed to focus more on the functionality rather than the look of our application. This is so that we can make sure everything works smoothly before spending the remaining time on making it look nice.

Although we were not able to produce the exact result that we had in mind, most members did their part to put the application together, and the team managed to create a decent application that we are all satisfied with, including the stakeholders.

Individual Reflection

Bibliography

1. Society for Human Resource Management (SHRM). (2015). Workforce Analytics: A Critical Evaluation. Retrieved August 20, 2022, from https://www.shrm.org/ResourcesAndTools/business-solutions/Documents/Organizational%20Staff%20Size.pdf

2. What is Scrum? Scrum.org. (n.d.). Retrieved August 25, 2022, from https://www.scrum.org/resources/what-is-scrum

3. Scrum sprint: Product management framework. Infinity. (n.d.). Retrieved August 25, 2022, from https://startinfinity.com/product-management-framework/scrum-sprint

4. System testing. GeeksforGeeks. (2022, June 9). Retrieved August 28, 2022, from https://www.geeksforgeeks.org/system-testing/

5. Admin. (2022, March 18). Why JavaScript Is The Most Demanded IT Skill. FusionReactor. Retrieved September 1, 2022, from https://www.fusion-reactor.com/general-interest/why-javascript-is-the-most-demanded-it-skill/

6. INFO-TECH SYSTEMS INTEGRATORS PTE LTD. (2022). Info-Tech Singapore: Cloud HR & Payroll Software. Retrieved September 1, 2022, from https://www.youtube.com/watch?v=ki13YjOkjdA.

Appendices

Appendix A: Scrum meeting logs

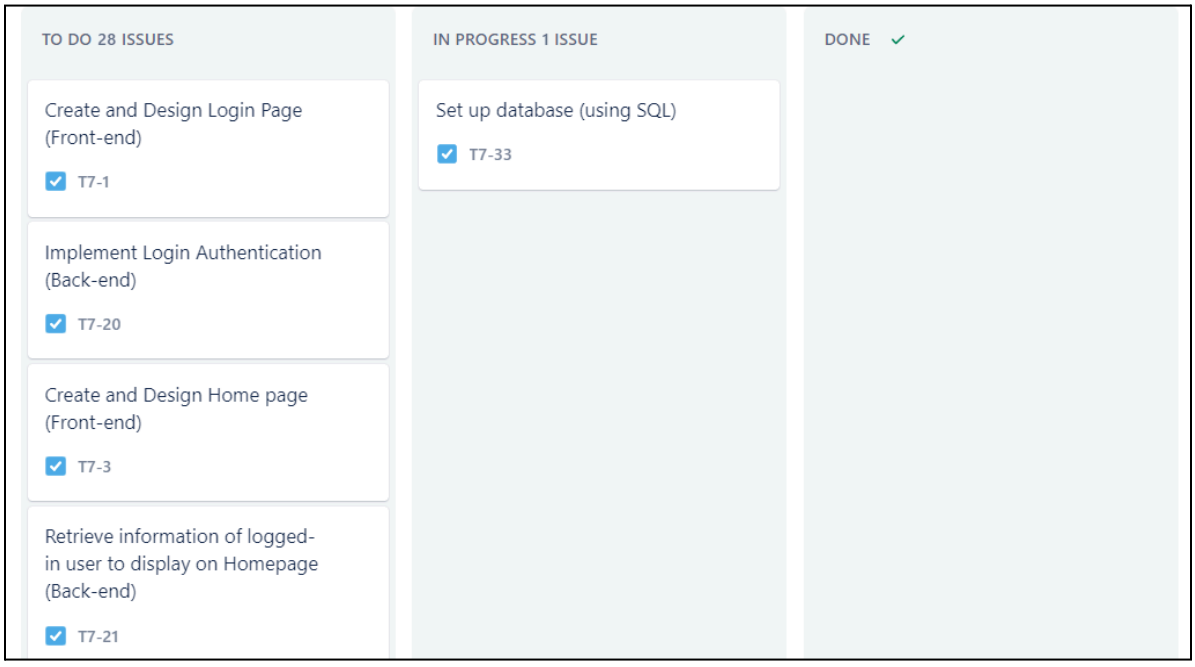
Below shows the full meeting log of all the scrum meetings held by the team.

Scrum meeting 1 (13 July 2022)

In this meeting, we appointed two members as the scrum master and the product owner respectively.

As the product owner is responsible for the product backlog and prioritising of tasks, she has to first look at the prototype we have previously designed and created. Only then can she pen down the required tasks for each function. After a team discussion, she listed the required tasks and included them in the Kanban Board under “To Do”. We then agreed to work towards our end product using SQL, JavaScript with node.js and express.js, together with HTML and CSS .

Deliverables for the next meeting are (1) to create a GitHub repository so that everyone in the team can work on the code collaboratively, and (2) to write SQL statements so as to create the required database tables, according to the ERD diagram we had designed previously.



(Snippet of Kanban board as of 13 July 2022)

Scrum meeting 2 (20 July 2022)

In this meeting, the team members updated each other on the tasks that are completed:

1. GitHub Repository created with base code added.
2. SQL statements to create the database tables written.

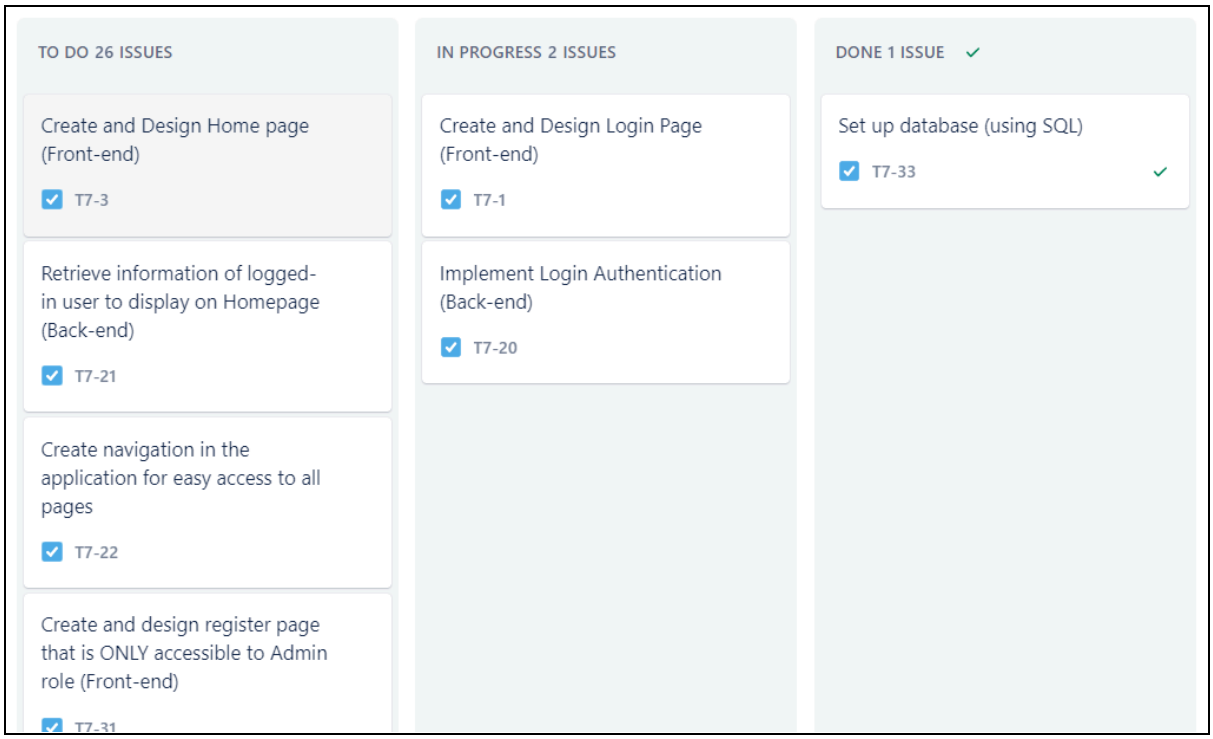
With the GitHub Repository in place, the team leader made sure that every member is able to clone the repository into their IDE and knows how to commit changes made to the code.

The team also inspected the SQL statements written with the database tables at the side for reference, and agreed that the task can be moved from “In progress” to the “Done” section of our Kanban Board.

Our scrum master has marked the login functionality as the highest priority because it is the landing page of our Cloud HRMS application. Therefore, the team agreed to work on the login functionality before diving into other functionalities;

Assignment of tasks:

Member 1	**Member 2**	**Member 3 & 4**
Create and Design Login Page html	Implement Login Authentication (server-side functionality) main.js, mySQL	Work on Final Report



(Snippet of Kanban Board as of 20 July 2022)

Scrum meeting 3 (27 July 2022)

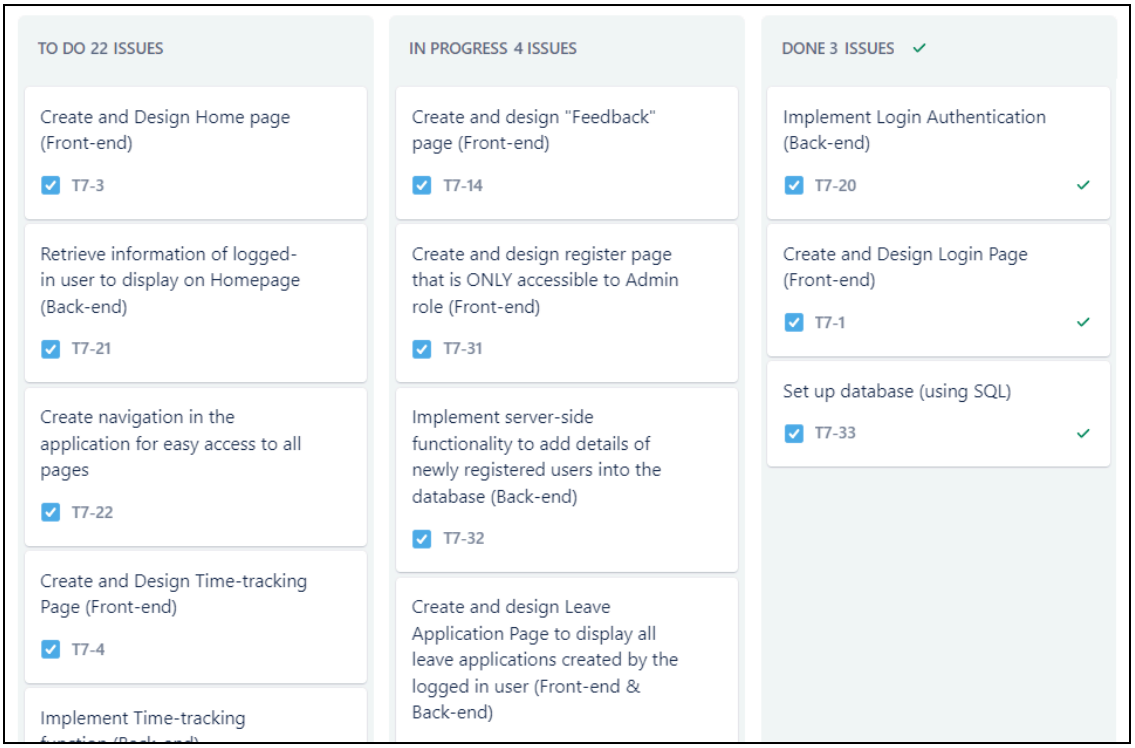
Tasks Completed:

- Login page created with HTML
    - Client-side (front-end)

- Login authentication implemented.
    - Server-side (back-end)
    - Users can log into the web application by keying in a username and password that matches our database records.

With the login page and login functionality fully implemented, the team has agreed to start working on the register page next, so that new user accounts can be created. As mentioned in our mid-term report, this register page is only accessible by users with the 'Admin' role.

The other deliverables for next week, apart from 'register' webpage, are a web page that displays all the 'leave application' submitted by the users, and the 'feedback' form.



(Snippet of Kanban Board as of 27 July 2022)

Scrum meeting 4 (3 August 2022)

Tasks Completed:

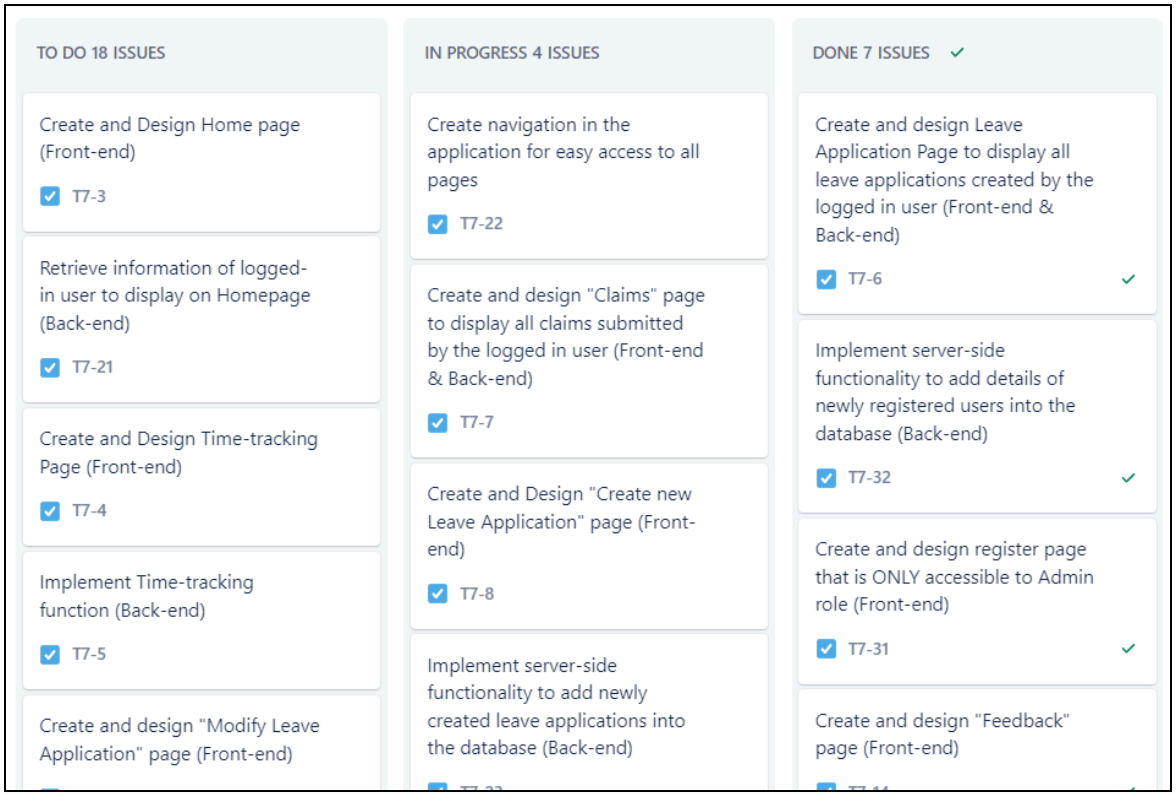
- Register web page (Client-Server side)
    - Admin can register for new user

- 'All Leave' web page (Client-Server side)
    - User can view all leave applications that they have submitted

- Feedback Form (Client-side)
    - User can submit the feedback form

With more web pages added into our web application, the team realised that navigation can be a problem after user login. Hence, we added a side navigation bar for easy and proper navigation around the entire application.

In addition, our team will mark the 'Claims' web page and the 'Add Leave' web page 'IN PROGRESS' and work on the client-server side.



(Snippet of Kanban Board as of 3 August 2022)

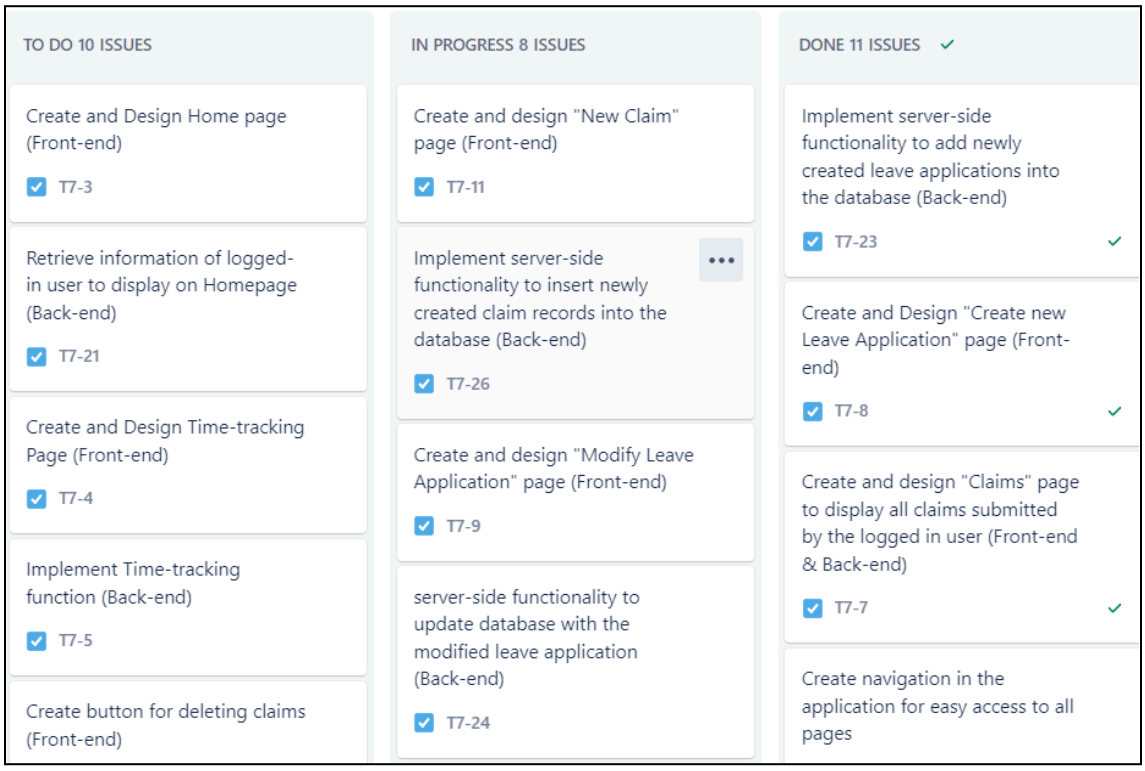
Scrum meeting 5 (10 August 2022)

Tasks Completed:

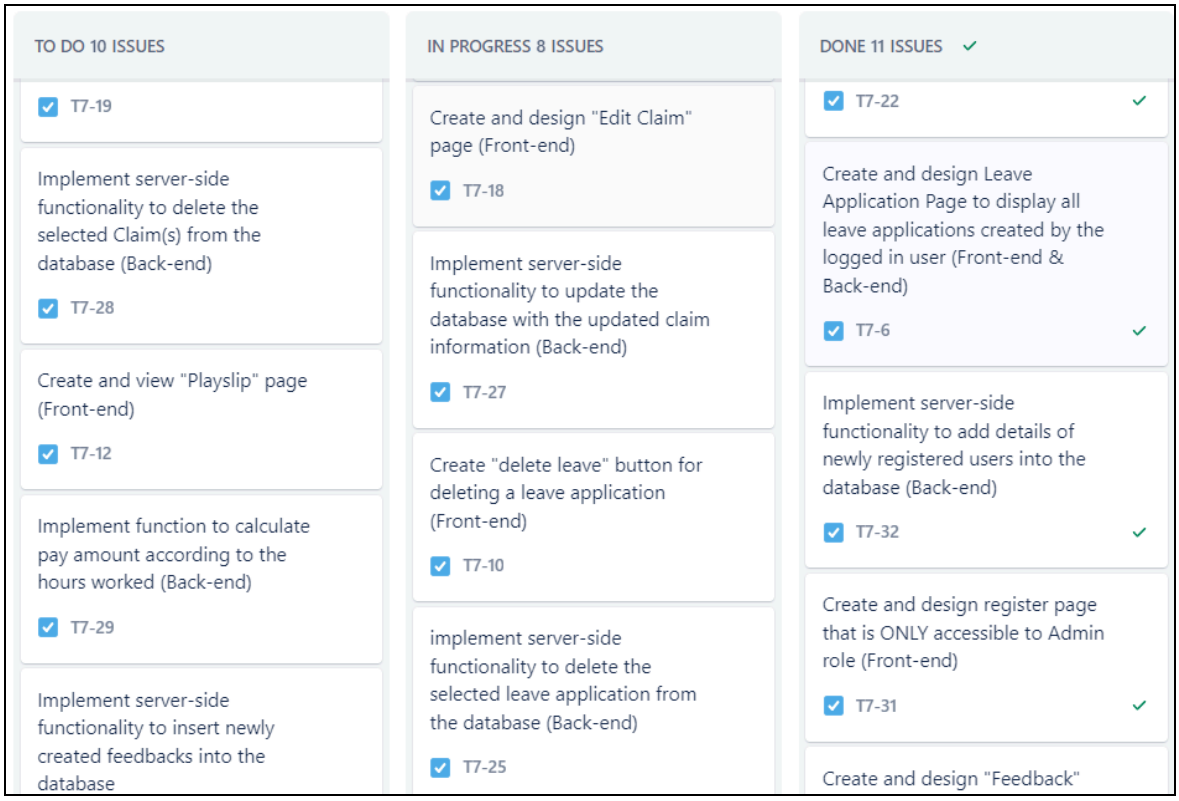
- A side-menu bar for easy navigation to different web pages.
- 'All Claim' webpage (client-server side)
    - Users can view all the claims they have submitted
- 'Add Leave' web page (client-server side)
    - Users can add a new leave application, and view it in the 'All Leaves' web page

With the 'All Claims' webpage displaying all the claims submitted by the logged in user, the next function our team has to work on is the "Add Claim" function. The "Edit Claim" function can be worked on if time allows.

The user is now also able to use the "Add Leave" function, so the next function should allow the user to edit and delete leave applications that have been added. The "delete leave application" should be worked on if time allows.



(Snippet of Kanban Board as of 10 August 2022) (1)



(Snippet of Kanban Board as of 10 August 2022) (2)

Scrum meeting 6 (17 August 2022)

Tasks Completed:

- ‘Add Claim’ web page (client-server side)
    - Users can add new claim (example, for reimbursement)

- ‘Edit Claim’ web page (client-server side)
    - Users can edit submitted claim

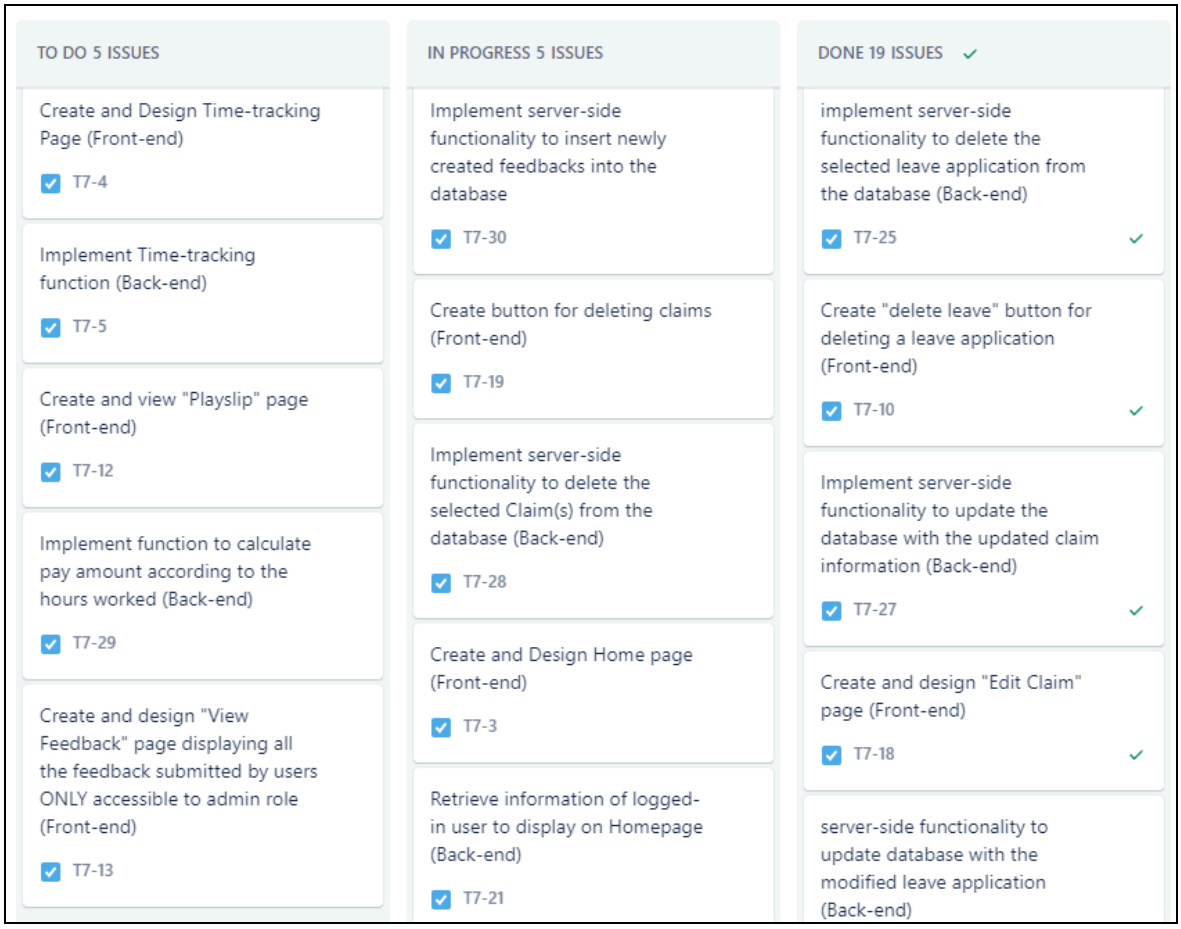
- ‘Edit Leave’ web page (client-server side)
    - Users can edit a leave application they have submitted for the manager’s approval.

- ‘Delete Leave’ web page (client-server side)
    - Users can delete a leave application they have submitted.

With the functionalities for the ‘Claims’, and ‘Leave’ almost completed, our team decided to move on to the homepage. This homepage will be displayed right after a successful user login and it should display the user’s information such as their name, username, and the time of login.

Deliverables for the next meeting:

1. Delete Claim (client-server side)
2. Homepage (client-server side)
3. Feedback form (server-side)
4. Schedule Page (client-side)



(Snippet of Kanban Board as of 17 August 2022)

Scrum meeting 7 (24 August 2022)

Tasks Completed:

- Feedback form (Server-side)
    - Users can submit a feedback form successfully upon clicking the submit button.

- 'Delete Claim' web page (Client-server side)
    - Users can delete claim(s) that were submitted for manager's approval.

- 'Homepage'' web page (Client-server side)
    - Upon successful login, the user is redirected to the homepage.
    - The homepage will display user's information such as their name, username, and the time of login.

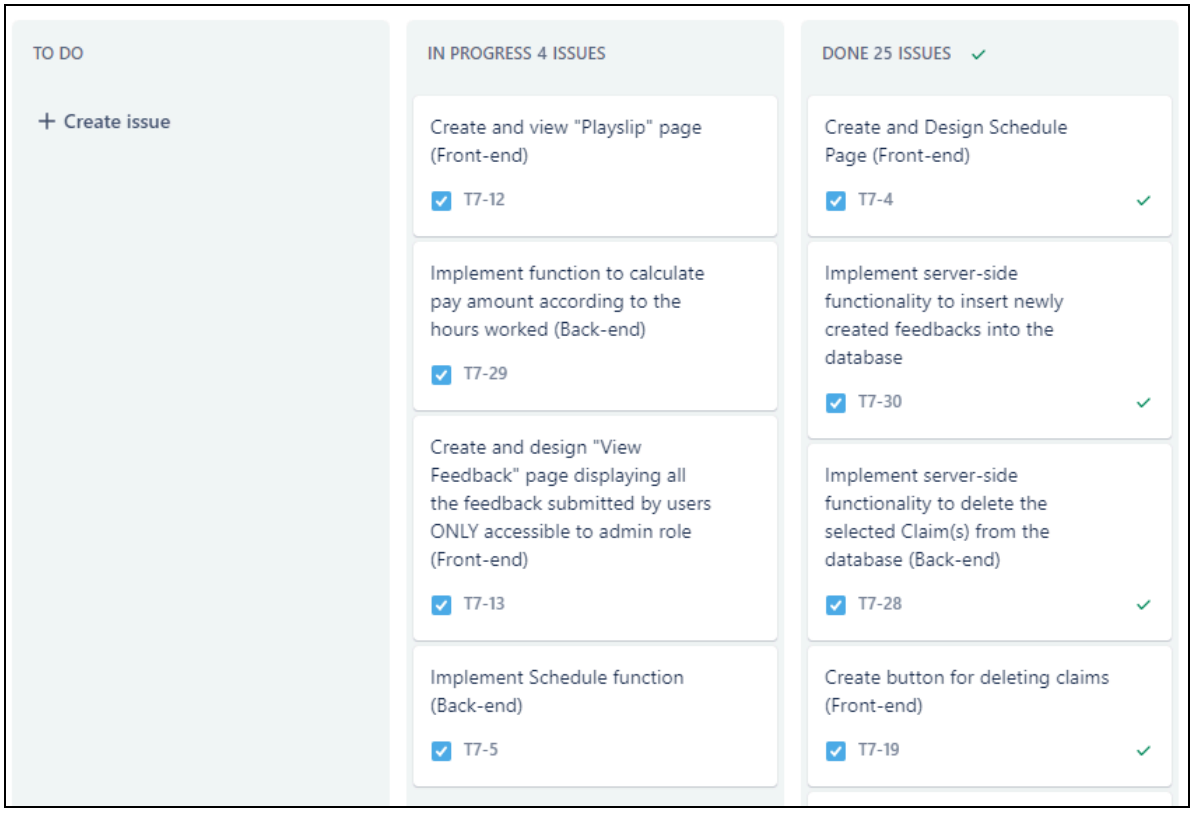
- 'Schedule'' web page (Client-server side)
    - Users can view the 'Schedule page'
    - This page will display a calendar

With the deadline closeby, our team wants to finish up what is still on our kanban board 'IN PROGRESS' section so that we can move it to the 'DONE' section and work on our last few deliverables after the next scrum meeting.

Team members who have free time on hand are to start unit testing for the components that have already been implemented before the scrum review with the stakeholders.

Deliverables for the next meeting:

1. Payslip Page (Client-server side)
2. View Feedback for ADMIN (Client-server side)
3. Unit testing.



(Snippet of Kanban Board as of 24 August 2022)

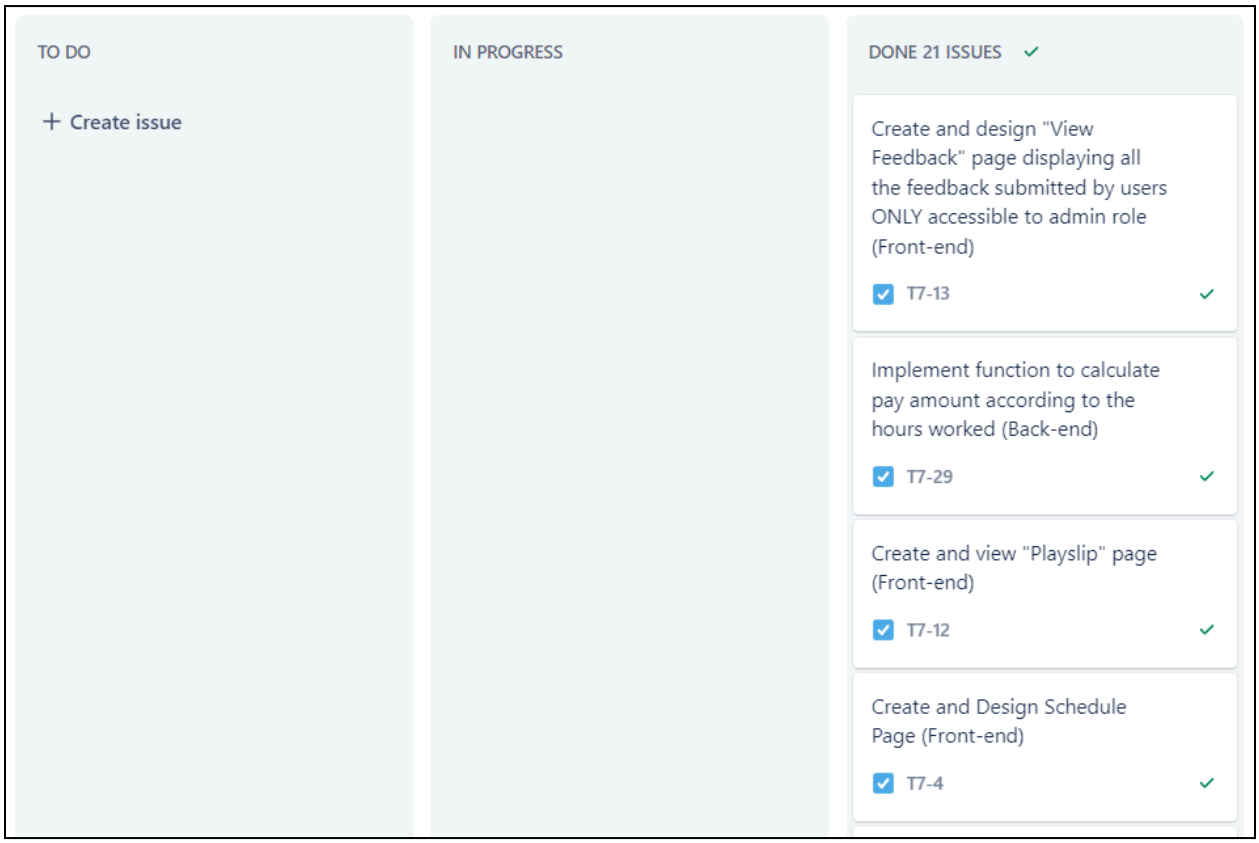
Scrum meeting 8 (31 August 2022)

Tasks Completed:

- Payslip Page (Client-server side)
- View Feedback for ADMIN (Client-server side)
- Unit testing.

In this meeting, the team has confirmed that all of the tasks that are needed to be worked on have all been completed. A check was done by comparing each task under the “Done” section of the kanban board to the actual application to confirm that they are indeed implemented.

With that, each member is tasked to do unit testing on the components that they have implemented, where they note down the details such as test details, expected results, status and so on. This is carried out before the integration stage by each member.

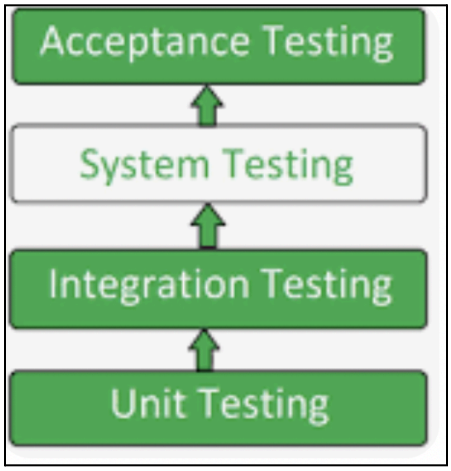


(Snippet of Kanban Board as of 31 August 2022)

Scrum meeting 9 (3 September 2022) - Final meeting

In this meeting, the team updated each other on the results of the unit testing with our test case templates we have filled up. After which, we integrated our codes together to see if all functionality and components worked as intended. Only then, our team can carry out system testing.

System testing is a black-box testing that has both functional and non-functional testing. It is usually carried out by the testers, known as the Quality Assurance (QA) or Quality Checking (QC) team. However, because we only have 4 members on the team, we will be doing the system testing by ourselves.



(Source: GeeksForGeeks: System Testing https://www.geeksforgeeks.org/system-testing/)

This step of system testing is to verify that the product meets the requirements mentioned in the requirement document, which defines what is needed from the product, like its purpose and goals.

As we want the best for 'The Smurfy Gang', our team wants to touch up on our codes with proper code formatting before finalising the final report with the remaining time we have left.

Appendix B: Register Page (Low and medium fidelity prototype)

Low-Fidelity prototype of Register Page

Medium-Fidelity prototype of Register Page

Appendix C: SQL Statements

SQL CREATE TABLE statements

CREATE TABLE Statement used to create tables, with the data type and primary key specified. FOREIGN KEY constraint is also used to specify the relationship between multiple tables. There are 13 tables in total.

```
CREATE table Department (
    Dept_id INT NOT NULL auto_increment,
    Department_name VARCHAR(50),

    PRIMARY KEY (Dept_id)
);

CREATE table Role (
    Role_id INT NOT NULL auto_increment,
    Role_name VARCHAR(50),

    PRIMARY KEY (Role_id)
);
```

```
CREATE table Staff (
    Staff_id INT NOT NULL auto_increment,
    Dept_id INT NOT NULL,
    Role_id INT NOT NULL,
    Staff_name VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL,
    address VARCHAR(50) NOT NULL,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL,

    PRIMARY KEY (Staff_id),
    FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id),
    FOREIGN KEY (Role_id) REFERENCES Role(Role_id)
);
```

```sql
CREATE table Claim_Type (
    CT_id INT NOT NULL auto_increment,
    type VARCHAR(50),

    PRIMARY KEY (CT_id)
);

CREATE table Claim (
    Claim_id INT NOT NULL auto_increment,
    CT_id INT NOT NULL,
    Staff_id INT NOT NULL,
    date_submitted DATETIME,
    claim_amount DOUBLE,
    date_of_claim DATETIME,
    status VARCHAR(50),

    PRIMARY KEY (Claim_id),
    FOREIGN KEY (CT_id) REFERENCES Claim_Type(CT_id),
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id)
);
```

```sql
CREATE table Punch_Card (
    PC_id INT NOT NULL auto_increment,
    Staff_id INT NOT NULL,
    date DATETIME,
    clock_in_time DATETIME,
    clock_out_time DATETIME,
    break_hours DOUBLE,

    PRIMARY KEY (PC_id),
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id)
);

CREATE table Feedback (
    Feedback_id INT NOT NULL auto_increment,
    name VARCHAR(50),
    email VARCHAR(50),
    feedback VARCHAR(250),

    PRIMARY KEY (Feedback_id)
);
```

```sql
CREATE table Leave_Reason (
    LR_id INT NOT NULL auto_increment,
    reason VARCHAR(50),

    PRIMARY KEY (LR_id)
);

CREATE table Leave_ (
    Leave_id INT NOT NULL auto_increment,
    Staff_id INT NOT NULL,
    LR_id INT NOT NULL,
    date_requested DATETIME,
    start_date DATETIME,
    end_date DATETIME,
    status VARCHAR(50),

    PRIMARY KEY (Leave_id),
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id),
    FOREIGN KEY (LR_id) REFERENCES Leave_Reason(LR_id)
);
```

```sql
CREATE table MC (
    MC_id INT NOT NULL auto_increment,
    Staff_id INT NOT NULL,
    MC_type VARCHAR(50),
    date_submitted DATETIME,
    start_date DATETIME,
    end_date DATETIME,

    PRIMARY KEY (MC_id),
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id)
);

CREATE table Shift (
    Shift_id INT NOT NULL auto_increment,
    timing VARCHAR(50),

    PRIMARY KEY (Shift_id)
);
```

```sql
CREATE table Schedule (
    Schedule_id INT NOT NULL auto_increment,
    Staff_id INT NOT NULL,
    Shift_id INT NOT NULL,
    date DATETIME,
    status VARCHAR(50),

    PRIMARY KEY (Schedule_id),
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id),
    FOREIGN KEY (Shift_id) REFERENCES Shift(Shift_id)
);

CREATE table staff_schedule (
    Staff_id INT NOT NULL,
    Schedule_id INT NOT NULL,

    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id),
    FOREIGN KEY (Schedule_id) REFERENCES Schedule(Schedule_id)
);
```

SQL INSERT statements

Insert statements are written to insert data into tables that require manual insertion of data. Those tables are Department, Role, Leave_Reason, and Claim_Type. These tables require manual insertion of data as they contain predefined data that cannot (and should not) be modified through the application.

```sql
INSERT INTO Department (Department_name)
Values ('Sales'), ('Purchase'), ('Marketing'), ('Finance'), ('Human Resource'), ('Operations');

INSERT INTO Role (Role_name)
Values ('Admin'), ('Manager'), ('Staff');

INSERT INTO Leave_Reason (reason)
VALUES ('Annual Leave'), ('Emergency Leave'), ('Others');

INSERT INTO Claim_Type (type)
VALUES ('Dental Claim'), ('Taxi Claim'), ('Medical Claim');
```

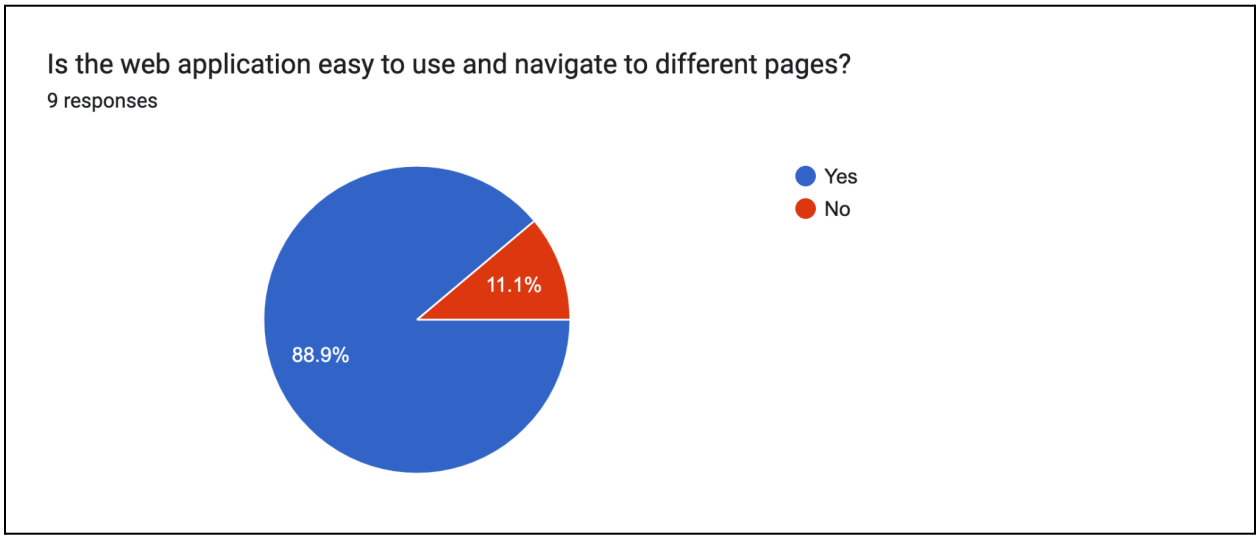
Appendix D: GitHub Repository

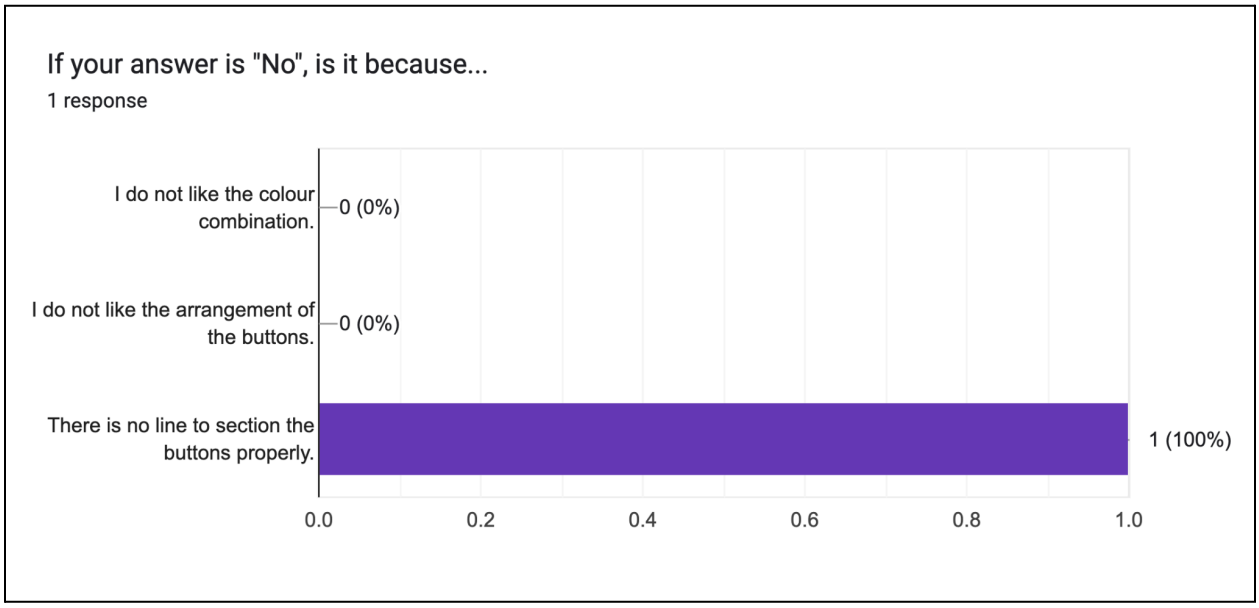
Link to GitHub Repository: https://github.com/Joel-Ang/HRMS-Web-Application

Below shows an overview of all the commits and changes made to the GitHub Repository. We are unable to show the overview from the beginning as the maximum period that can be selected is 1 month.

Appendix E: Survey Questions and Responses

Customer Satisfaction Survey link for the scrum review:
https://docs.google.com/forms/d/e/1FAIpQLSeIxRMyNFnzdJqnffqvodmY6aiEXtrKiZgh83sBoDZ4vLOwog/viewform?usp=sf_link





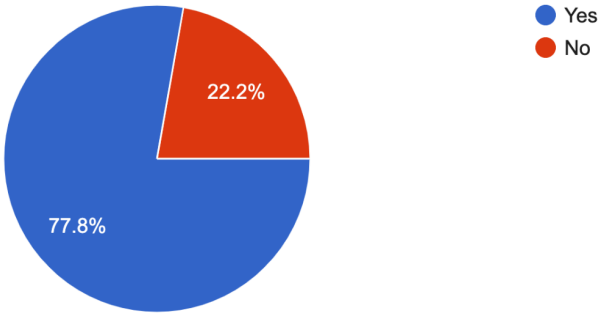
If you have chosen 'Other', please let us know why is that so.

1 response

I choose Yes

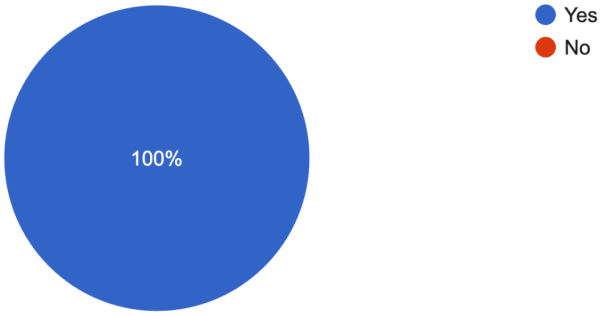
Is the home page's user interface simple to understand?

9 responses



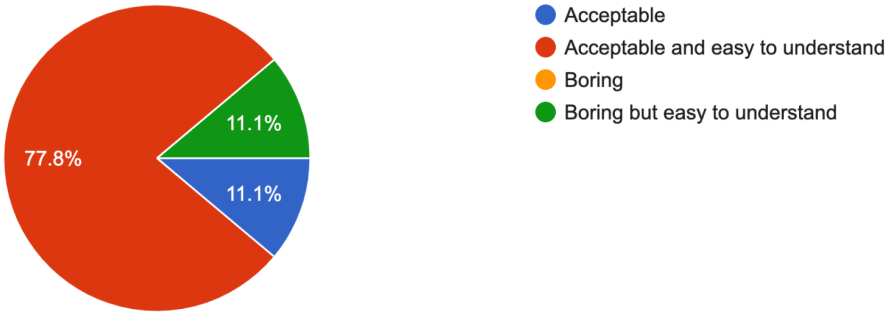
Is the user interface simple to understand what you are required input? Can you tell that this is meant for the Admin Users from the screenshot below?

9 responses



Is the design too simple to your liking? Is the user interface simple to understand?

9 responses



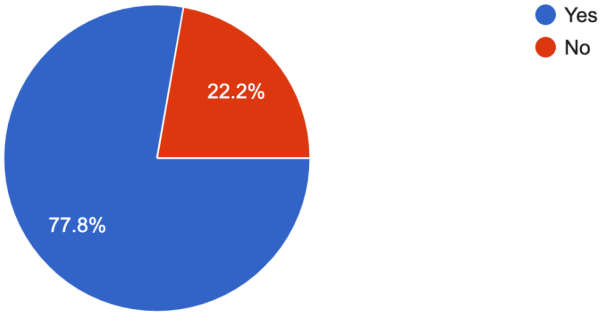
If you have chosen 'Other', please share with us your view.

0 responses

No responses yet for this question.

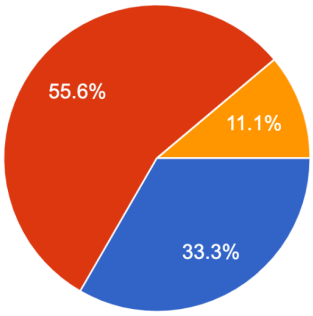
This is the first draft of the schedule page. Do you like this layout ? (do note that this is not the final product, because this will be implemented during ...ond deployment as requested by the stakeholders)

9 responses



This is the second draft of the schedule page. Do you like this layout better?

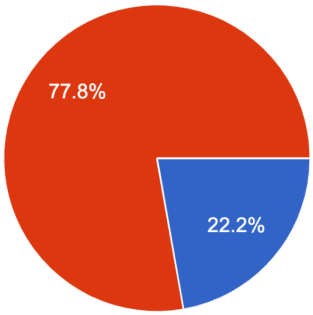
9 responses



- Draft 1, from the previous question, is way better
- This (Draft 2) is way better
- I do not like both of them

Is the table of 'All Leaves' simple to understand? Is the colour coding for 'Approved', 'Pending' and 'Rejected' okay? (We used traffic light colours)

9 responses



- Easy to understand
- Easy to understand & Colour coding is okay
- Not easy to understand
- Colour coding is bad, I want other colours.

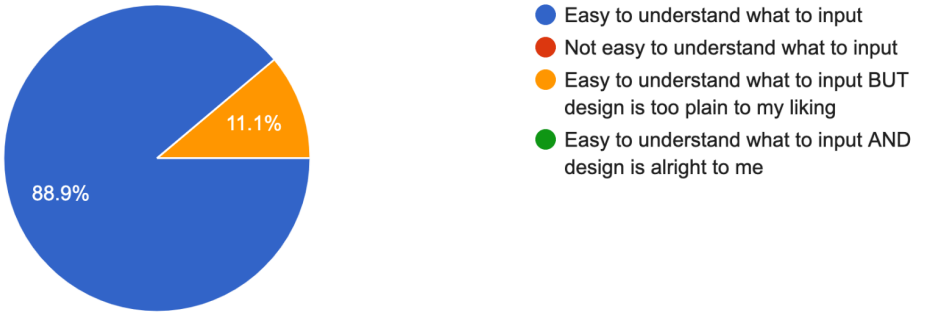
If your answer is 'Other', please share with us your view on 'All Leaves' page.

0 responses

No responses yet for this question.

Is the user interface simple to understand? Do you know what to input? Is the design too simple to your liking?

9 responses



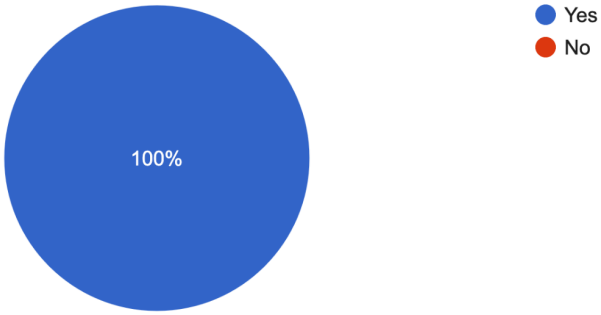
If your answer is 'Other', please share with us your view.

0 responses

No responses yet for this question.

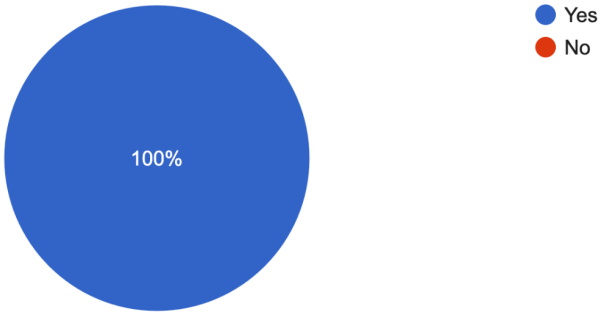
Is the user interface simple to understand?

9 responses



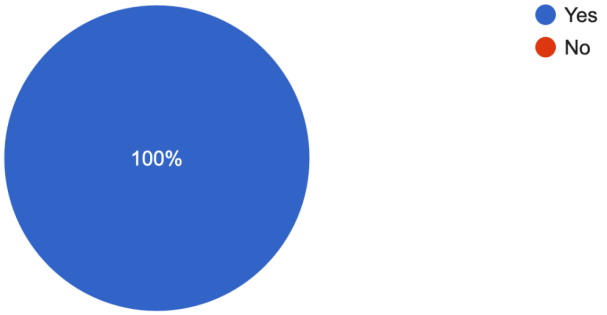
Would you prefer the system to prompt you for confirmation of deletion?

9 responses



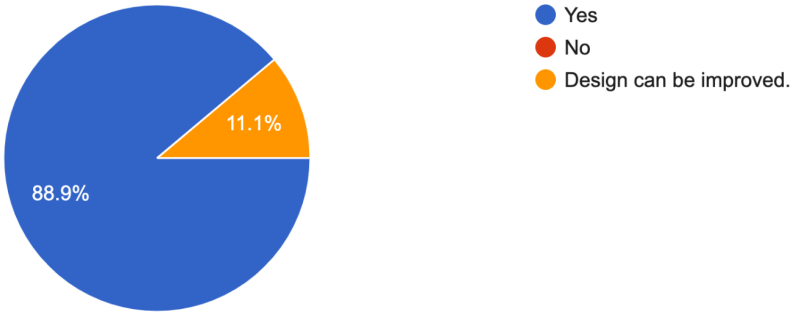
Is the table for 'All Claims' simple to understand?

9 responses



Is the reimbursement form simple to understand?

9 responses



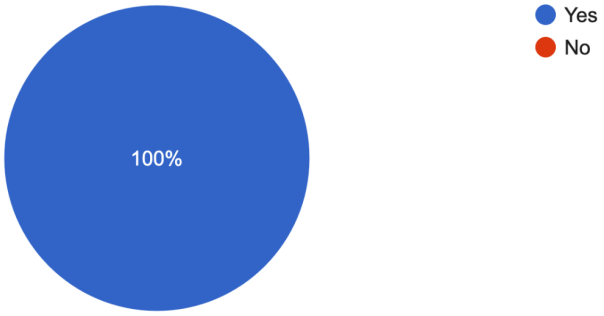
If you have chosen 'Other', please share with us your view.

1 response

More like the other pages would be nice.

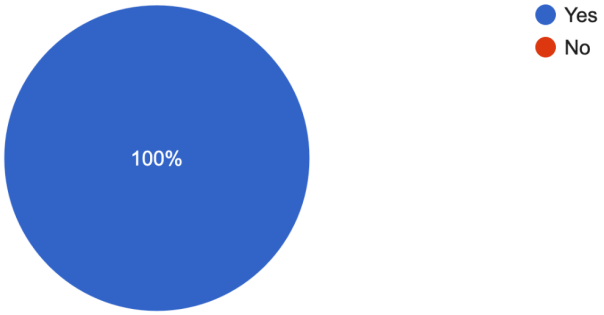
Is the feedback form simple to understand?

9 responses



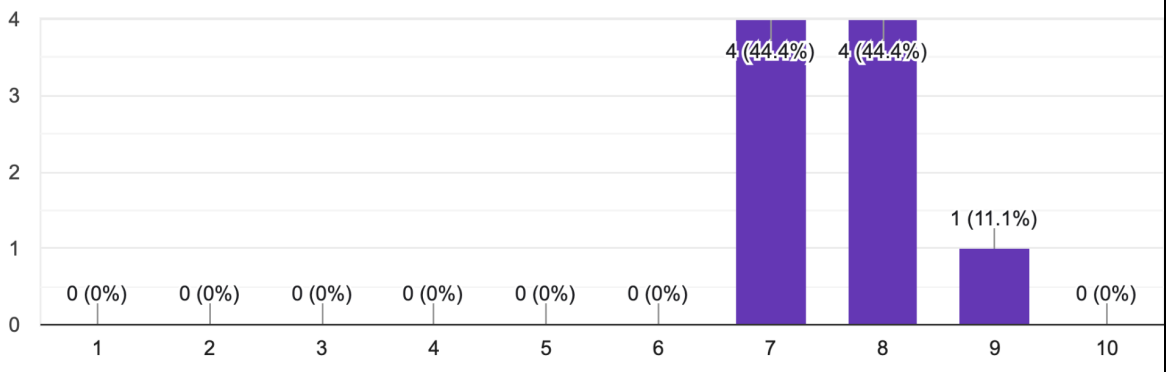
Is the payslip page simple to understand?

9 responses



With that, how would you rate the Cloud HRMS web application we have implemented?

9 responses



We want the best for your organisation. Do leave us a feedback on how we can improve. Thank you and cheers!

3 responses

Have all the forms have similar designs

can use more aesthetic fonts

Should improve more on the design such as colour and layout