

Table of Content

Development Plan	4
Key Features	6
Search and Filter	6
Search Bar and Filter	6
Map View	6
Challenges	7
Booking	8
Booking Form and Availability Calendar	8
Booking Management	9
User Messaging System	9
Challenges	9
Additional Notes and Future Implementations	10
Authentication	11
User Authentication and Profile Management	11
Challenges	12
Administrator Access (Facility Owner Perspective)	13
Facility Management and Booking Management	13
Limitation	14
Messaging	15
Notifications and Reminders	15
Messaging	15
Challenges	15
Limitations	16
Dashboards	17
Performance Metrics	17
Recent Applications and Daily Booking	17
Challenges	17
Limitations	17
Responsive Design	18
Main Page	18
Facility View Page	19
Map View Page	20
Message Page	21
Canceled Features	23
Challenges While deploying	23
Improvement of the Website	24
1. Use of Local Storage for Storing Usernames	24
2. Lack of Type Checking	24
3. Inefficient Data Handling	25
4. Other known issues	25
Testing	26
Unit Testing	26

End to End Testing	31
User Testing	34
Tech Stack	40
Frontend Development	41
Framework and Bundler	41
UI Components and Styling	41
JavaScript for Logic and Communication	41
Key Packages	41
Backend Development and Database	41
Node.js and Express.js	41
Key Packages	42
Database	42
Version Control	42
Software Engineering Practices	44
Overall development progress	44
Frontend	44
Transition to React.js	44
Libraries	44
Responsive Design	44
Backend	45
Choosing Node.js	45
Learning Key Concepts	45
Refactoring with Routers	45
Activity Diagram	46
User flow Map	46
Activity diagram	46
CI / CD	48
Comment Minimally, But Sufficiently	48
Removing Dead Code	49
Sample account	51
Instructions	51

Project Introduction

Team Name

Peeka Book

Proposed Level of Achievement

Apollo 11

Motivation

As students at NUS, we have used REBOKS to book facilities for our CCA and hall activities. However, the user experience was not satisfactory. As freshmen not familiar with the university campus, there was no location information provided for the facilities, making it difficult to find the correct venue for our CCA activities and sometimes leading to incorrect bookings. We hope to build a new website that not only improves the functionality of REBOKS but also allows for booking facilities outside the campus.

Aim

We hope to develop software that enables people to share facilities together without clashing time and the booking process of the facility can be more convenient and friendly to all-age.

User stories

1. As a student who lives beside a basketball court, I want to book the court on weekends so that I can play basketball with my friend.
2. As new residents live beside this area, I would like to search the available facilities around my area and perhaps use the facility.
3. As a user that is always busy, I want the software to remind me of the upcoming booking so that I would not miss it.
4. As a user that finished using the facility, I want to share my experience on the software so that it may be helpful for others.
5. As an owner of a facility, I wish to share this facility with others and perhaps make profits from it.
6. As a user not familiar with modern technology, I wish the website is easy to use and can use different features without obstacles.

Development Plan

Week 1 13 May – 19 May	18May (Sat) – Mission Control 1 • formulate a more comprehensive version of the project, write out the details of each features • clarify unclear idea about the project, make sure both of us understand the project purpose and direction are the same • design each webpage to make our development easier and clear during the development stage • learn git and some frontend Tech stack
Week 2 20 May – 26 May	25 May (Sat) – Mission Control 2 • learn about backend and database Tech stack, know something about software engineering principle • Learn about the integration with frontend, backend and database • frontend development: create basic web page for our project ○ Login page & create account page & email verification page(ZH) ○ Main page(search page)(ZH) ○ My bookings page(Collin) ○ My facilities page(Collin) ○ Facilities profile page(for user)(ZH) ○ Facilities profile page(for owner)(ZH) ○ Send booking application page (ZH) ○ Create facility identity page(ZH)
Week 3 27 May – 2Jun	1Jun (Sat) – Mission Control 3 • Integrate frontend with backend • Web Page navigation(Collin) • Database design(ZH) • Backend development: develop main feature ○ Searching(ZH) ■ Main page and retrieve correspond facility by the name ■ Update the webpage ○ Booking (Co) ■ Submit booking application ■ Swap booking ■ Contact the owner(optional) ○ User(ZH) ■ Sign up ■ Edit user profile information ■ Email verification ○ facility management (Collin) ■ Approve or reject the booking application ■ Edit the data of the facility • Writing README ○ Introduction, motivation, Aim ○ Key features ○ Completed and uncompleted features description • Deploy project to internet(ZH) • Project poster and video
Week 4 3 Jun – 9 Jun	3Jun (Mon) – MileStones 1 deadline 2pm 8Jun (Sat) – Mission Control 4 • Finish and complete all core features ○ Users system(edit, submit)

	○ Map search features ■ List nearby facilities ■ Click on the pin button show the detail of the facility ○ Advanced filter function ■ Filter by location, type, price range and distance ○ Booking application management ■ Transfer image through form ● Learn about software testing
Week 4 3 Jun – 9 Jun	3Jun (Mon) – Milestones 1 deadline 2pm 8Jun (Sat) – Mission Control 4 ● Finish and complete all core features ○ Users system(edit, submit) ○ Map search features ■ List nearby facilities ■ Click on the pin button show the detail of the facility ○ Advanced filter function ■ Filter by location, type, price range and distance ○ Booking application management ■ Transfer image through form
Week 5 10 Jun -16 Jun	Implement rest of the important features ● Reviews and social ● Booking management ○ Delete, approve, swap
Week 6 17 Jun – 23 Jun	Implement rest of the important features ● UI/UX design ● Software testing
Week 7 24 Jun – 30 Jun	● Documentation writing ● Deploying into internet ● Test case writing ● Advanced filter function
Week 8 1 Jul – 7 Jul	1Jul (Mon) – Milestones 2 deadline 2pm ● Implement extension(contact) ● Fixing some other not implemented small features ● Dashboard
Week 9 8 Jul – 14 Jul	● Continue implementing extension ● Clean up code
Week 10 15 Jul – 21 Jul	● Complete test about product ● Usability testing integration testing ● Fixing bugs ● UI / UX design
Week 11 22 Jul – 28 Jul	● Poster and video
Week 12 29 Jul – 4 Aug	29 Jul (Mon) – Milestones 3 deadline 2pm

Key Features

Search and Filter

- Search Bar allows user to input keywords to find facilities
- Filter Panel: provides options to filter search results by service type, price range and distance.
- Map View: show service locations on the map near the user or relative to the user's search criteria. The map shows the precise location of each facility. User can select each location for more detailed information

Search Bar and Filter

1. **Searching:** Allows users to input keywords to search for facilities. After clicking the enter key or 'go' button, only the facilities related to the keyword will be displayed. The search functionality currently supports keywords based on facility names.
2. **Advanced Filters:** Beside the search bar, there is a filter icon. Clicking the filter icon will pop up a modal, allowing users to choose more advanced filter options for the desired facility.
 - a. **Filter Options Include:**
 - i. Type
 - ii. Distance
 - iii. Price Range

Limitations

- only facility name can be search by keyword

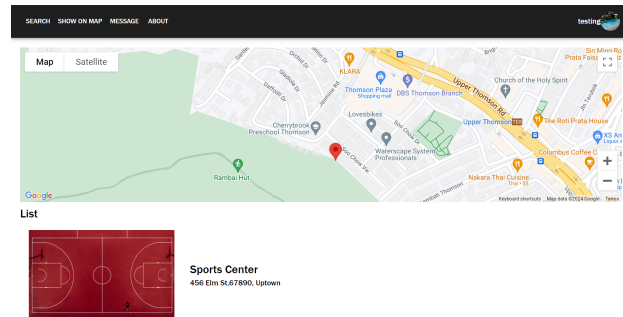
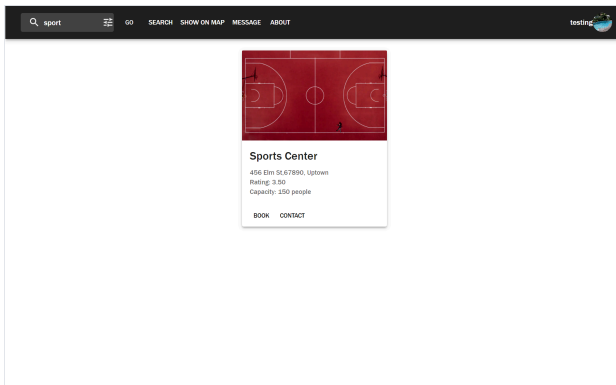
Map View

1. **Location Visualization:** The Map View feature allows users to visualize facility locations directly on a map using a map API. The markers on the map represent the location of facilities provided by the facilities owners.
2. **User Interaction**
 - a. **Navigating the Map:** Users can interact with the map to explore nearby facilities. As they pan or zoom the map, the list and markers update to reflect only those facilities currently within the map's viewport.
 - b. **Selecting a Facility:**
 - i. **Clicking a Marker:** Users can click on a marker on the map to view brief information about the facility. This brief information helps users quickly identify if the facility meets their needs.
 - ii. **Clicking the List:** Alternatively, users can click on a facility from the list below the map. This action will highlight the corresponding marker on the map for easy identification.

- c. **Detailed Information:** Clicking on either the marker or the list item will bring up a modal with detailed information about the selected facility. This modal includes comprehensive details that help users make informed decisions.
- d. **Booking a Facility:** Within the detailed information modal, users have the option to click the "Book" button to proceed with booking the facility to ensure a smooth booking process.

Special Implementation

The Map View feature is designed to display only the locations (pinpoints/markers) of facilities that are currently visible in the list below the map. As users interact with the map, the view bounds are calculated dynamically and only the facilities whose markers are within the visible area of the map will be shown. This dynamic display enhances user interaction, allowing users to focus on and easily locate the facilities they are interested in by correlating the map view with the list below.



Challenges

The most difficult challenge was figuring out how to use the Google Maps API, as there are various ways to use it, such as with plain HTML, React, and other methods. Some tutorials used specific tech stacks, and we had to figure out on our own how to properly use the Google Maps API. The filter panel icon also gave us trouble because the alignment was not correct, and we spent a lot of time figuring out how to center it.

Booking

- Availability Calendar: display available dates and times for booking facility
- Booking Form: collect necessary details from applicants such as personal information and specific requirements from the users.
- Booking Submission: send booking requests to the owner of the facility for approval.
- Booking Management: display a list of upcoming events for users to view, swap or delete bookings
- Reviews and rating: allow users to leave detailed reviews and star ratings for services under each facility profile page so that other users can have a reference.
- User Messaging System: enable users to message facility owners for inquiries or additional requests.

Booking Form and Availability Calendar

1. **Information provided by owners:** Our booking form provides essential details about the facility, such as address, name, capacity, and type, ensuring users can confirm their selection accurately.
2. **Information provided by users:** Users should provide necessary information to complete the booking process. The providing information procedure include different sections:
 - a. **Date Picker:** Users can select current or future dates. Past dates are disabled to prevent inappropriate bookings.
 - b. **Time Slot Picker:**
 - i. **Interactive Table:** Users choose time slots by interacting with a clickable table.
 - ii. **Color Coding:**
 1. Green: Available time slots.
 2. Red: Already booked time slots that are not selectable.
 - iii. **Dynamic Display:** Selected time slots are shown dynamically below the table for easy reference.
 - iv. **Continuous Selection Requirement:** Users must select continuous time slots to maintain coherent bookings. Non-continuous selections are automatically cleared.
 - c. **Other Required Information:**
 - i. Number of people
 - ii. Purpose of the booking
 - iii. Reason for the application
3. **Form Submission:**
 - a. **Submission to the Owner:** Once users fill out the required information, they submit the form to the owner for review.
 - b. **Approval Process:** The owner reviews the submission based on availability and other considerations before approving or denying the booking.
4. **Other Features:**

- a. **Dynamic Fee Calculation:** Total fees are calculated in real-time based on the selected time slot so that users can check the total costs and make sure it does not exceed their budgets.

Booking Management

- 1. **Future Bookings:** After the owner accepts applications, they are displayed on the management page under the "Future Bookings" section. Users can **view** the details of their bookings, **swap** time slots with other people, or **cancel** their bookings.
- 2. **Past Bookings:** For bookings that have already happened, users can rate or delete the activities. When users click the rate button, a modal will pop up where they can give a **rating** out of 5 stars and write a **comment**.
- 3. **Swap Booking:** The swap booking extension allows users to anonymously exchange their booking slots with others if their schedules change. Users can list their booking availability for swapping and specify their preferred new dates and times. The system matches users with compatible swap requests without revealing their identities. Once a match is found, both parties are notified and asked to confirm the swap. The swapping process is completed only after both users approve the exchange.

User Messaging System

- 1. **Inquire Information:** users can directly message the facility owner via the messaging page to inquire about or request additional requirements.
- 2. **Confirmation notification:** Those approved applications will also notify users through the messaging system.

25

9101112131415

16171819202122

23242526272829

30

TODAY

25

Tue

08:00

09:00

10:00

11:00

12:00

13:00

14:00

15:00

16:00

17:00

18:00

19:00

13:00 - 15:00

Number of People

3

Purpose

☒ Personal

☐ Organization

☐ Other

Reason

for training

\$100

SUBMIT

RESET

booking

Future Booking

No	Venue/Facility	Date	Time	Actions
1	ABCDEFGHJKLMNOPQRSTUVWXYZ	27/6/2024	15:00:00 - 19:00:00	DETAILSWAPCANCEL

History

No	Venue/Facility	Date	Time	Actions
1	Conference Room	13/6/2024	16:00:00 - 17:00:00	RATECANCEL
2	ABCDEFGHJKLMNOPQRSTUVWXYZ	14/6/2024	12:00:00 - 16:00:00	RATECANCEL
3	ABCDEFGHJKLMNOPQRSTUVWXYZ	24/6/2024	19:00:00 - 20:00:00	RATECANCEL
4	ABCDEFGHJKLMNOPQRSTUVWXYZ	24/6/2024	16:00:00 - 19:00:00	RATECANCEL

Challenges

Most of the challenges we encountered involved understanding the component documentation and how to use it properly. The time slot uses a package called Scheduler, which is written in TypeScript. Since we don't have any experience with TypeScript, sometimes we don't know the syntax of the parameters, leading to some buggy behaviors.

Additional Notes and Future Implementations

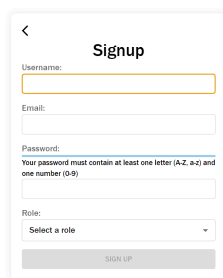
We are not planning to add a payment system right now, so the fees shown will just be for display. The swap booking feature is more complicated than we thought and won't be ready in this milestone, but we will work on it for future updates. We will also add a messaging system in the next milestone to let users and owners talk directly through the platform and also for the confirmation message.

Authentication

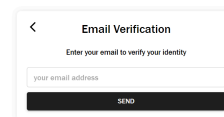
- User profile sign-up login: enable account creation with a username, password and email verification. Users can log in the webpage using their username and password.
- Password Security: ensure passwords meet security standards.
- Profile Management: allow users to upload a profile picture and edit personal information.
- Password Recovery: when a user forgot a password there will be email verification sent to the user with the new password. The password is only available for 5 minutes and after that the send password will not be usable.

User Authentication and Profile Management

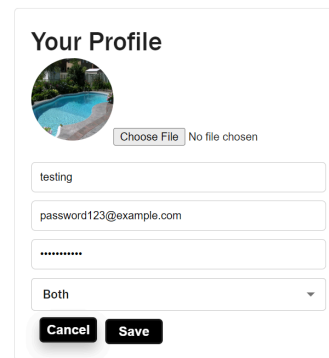
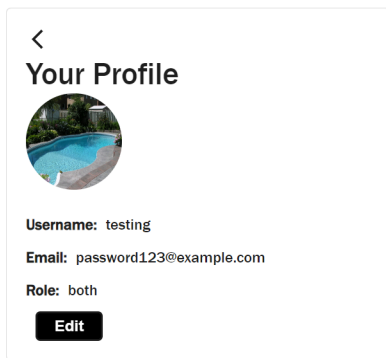
1. **Login:** Upon being directed to the login page, users are required to log in using their username and password.
2. **Sign Up:** If users do not have an account, they can select the "Sign Up" option to create one. The sign-up process involves completing basic information to create an account. Once the account is successfully created, users will be redirected to the login page to log in.
 - a. **Basic Information:** user should provide a unique username, password, email, and what is their role for using the web application.
 - b. **Password Strength:** The system ensures passwords meet security standards, users password should be more than 6 characters and no more than 12 characters. password must also include alphabet and numbers.
3. **Password Recovery:** For users who have forgotten their password, a verification email will be sent to the user. The email includes a new reset password so that users can use the password to log in their account.
4. **User Profile Management:** Once logged in, users can access their settings by clicking on their user avatar located at the top right corner of the web page and selecting the "Edit Profile" option. In this section, users can:
 - a. Update their unique username
 - b. Upload a new profile picture
 - c. If users choose not to save their changes, they can cancel the edit, and their profile information will remain unchanged.



A mobile app signup form titled "Signup" with a back arrow. It contains four input fields: "Username:", "Email:", "Password:", and "Role:". The "Password:" field has a hint: "Your password must contain at least one letter (A-Z, a-z) and one number (0-9)". The "Role:" field is a dropdown menu with the text "Select a role". At the bottom is a "SIGN UP" button.



A mobile app email verification form titled "Email Verification" with a back arrow. It contains a single input field labeled "your email address" and a "SEND" button at the bottom.



Challenges

Since we started trying to connect to the database for user authentication, the challenge was how to connect to the database and configure it. Although it took some time, it was quite smooth compared to other challenges and didn't take as long. Another challenge was the password recovery method, as it required configuring Node Mailer and Mailslurp to test if the emails were received. We also used Multer to process image uploads, and it was quite hard to figure out how to set up the image upload at first and which folder to place the images in.

Administrator Access (Facility Owner Perspective)

- **Facility management:** The owners of a facility can create the identity of the facility by providing basic information of the facility such name, address, pricing, remarks and profile picture to let others discover their facility. The Information of the facility can be edited afterwards if needed. For the ratings of the facilities, the owner can choose to delete or reply for the reviews.
- **Booking Management:** allow administrators to view, approve or reject booking. Owners can view the booking slots of the day to see who is using the facility.

Facility Management and Booking Management

1. Facility

- Create Facility Identity:** Owners can create facility identity by providing name, address, pricing, capacity, remarks, and a profile picture to help others discover their facility.
 - Pin Precise Location:** Owners can pin down the location of the facility on a map to let users know the precise location.
 - Auto Searching Address:** After the owner enters the address, the map will auto-search for the facility and automatically pin down the location if the address is found.
- Edit Facility Information:** Owners can edit the facility's information, update the facility's profile picture or edit remarks related to the facility.
- Ratings management:** Owners can delete reviews or reply to reviews.

2. Booking

- Approve/Reject Applications:** Owners can view, approve, or reject booking applications.
- Current User Display:** A facility card at the top of the page shows the current users of the facility, making it easy for the owner to see who is using it.
- Recent/Daily Booking Slots:** Owners can check out the booking slots for the day to see who's using the facility. There's also a table with recent bookings so owners can quickly get a snapshot of the latest activity and operations.

<



ABCDEFGHIJKLMNOPQRSTUVWXYZ

123 Street, 12345, Region

Rating: 4.00

Capacity: 100 people

Fees/hour: \$50.00

Type: Basketball Court

Current User

testing



Applications

User	Type	Reason	Operation
 testing	personal	123 jalan	DETAIL APPROVE REJECT

Rating

 testing

★★★★★

 jane_smith

★★★★☆

Excellent service!

Very good, but room for improvement.

< back

Select your facility page picture

[Choose File](#) Backyardpool.jpg



Name:

Pool

Unit & Street:

23A Kent Ridge Road

Postcode:

23145

Region:

Singapore

[Map](#) [Satellite](#)



Limitation

For the reply message function, we think it's just as tricky as the messaging system itself. We'll get it done by the next milestone.

Messaging

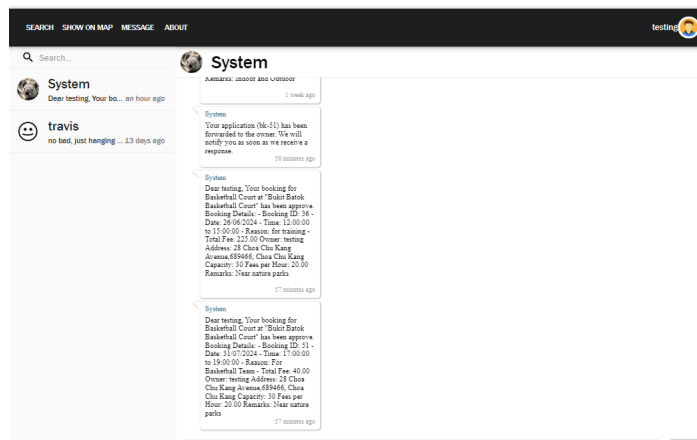
- Notification and reminders: the upcoming bookings, any changes like cancellations or swap will inform the user using the messaging system.
- Messaging: users can send a message to the owner to know for the more details of the facility information, after booking the booking information will also be sent to the user for reference.

Notifications and Reminders

1. **Booking Application Submission:** Users will receive a message containing their booking application details after submitting their applications. This message will include the booking number.
2. **Application Status Update:** Users will receive a message with detailed information about their application once the owner has taken action (approved or rejected). This message will include: Booking ID, Date, Start time to end time, Reason, Total fees, Owner of the facilities, Address, Remarks.

Messaging

1. **Inquiries:** Users can send messages to the owner to get more details about the facility.
2. **Sending Messages:**
 - Users can send messages to other persons and the messages will be received and rendered immediately.
 - Users can utilize this feature to inquire more information about the facility.
 - By clicking the contact button on the main page or facility information page, users will be redirected to the message page where the dialog box will be directed to the owner of the facility.
 - Users can also search for a username to start a conversation with them.



Challenges

Implementing messaging features has been tough, mainly because of the complexities of Socket.IO and networking. Figuring out the difference between socket and io and understanding how events work has been confusing. Since we don't have a strong background in networking, we've had to rely on tutorials and a lot of trial and error to get things working.

Limitations

- Due to the package we are using, it is not compatible with line breaks, so the messages will be messy. However, the line breaks are actually stored in the database and can be fixed by implementing the dialog ourselves. Due to time limitations, we have not yet implemented this fix.

Dashboards

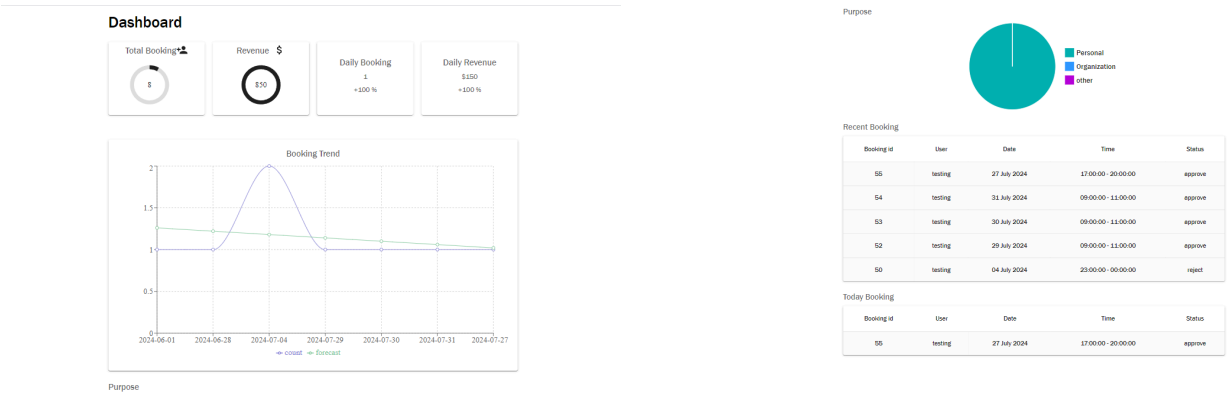
- Performance Metrics: display booking trends, revenue reports, user engagement statistics, occupancy and ratings.
- Recent Applications: This table shows the latest facility applications, giving owners a quick overview of recent activity.
- Daily Booking: This table displays bookings for the current day, so owners can see who is scheduled to use the facility today.

Performance Metrics

- **Data Trends:** Each facility has a chart displaying trends over time to help you forecast booking counts and predict revenue.
- **Total Metrics:** View total bookings, total revenue, daily bookings, and daily revenue.
- **Purpose Distribution:** See the distribution of booking purposes to understand how your facility is being used.

Recent Applications and Daily Booking

- **Recent Applications:** A table showing the latest facility applications, providing a quick overview regardless of their status.
- **Daily Booking:** A table with booking details for the current day, including timeslots and user information, so owners can see who is using the facility and when.



Challenges

Making a forecast for the booking trend is a bit tricky since we are not familiar with regression analysis. Therefore, we decided to use a JavaScript regression library to accomplish it. We used linear regression to indicate the trend of the bookings. However, due to the limited data, the accuracy may not be as high as we would like.

Limitations

The data chart might not have accurate plotting as it needs to include bookings for every day, even on days with no bookings. Therefore, we decided to insert data only for the days when bookings occurred.

Responsive Design

To ensure our platform is user-friendly across all devices, we've implemented a responsive design. Some of the pages have similar implementation design so we will only take out one to describe. While we aim to make every page fully responsive, some areas are still in progress.

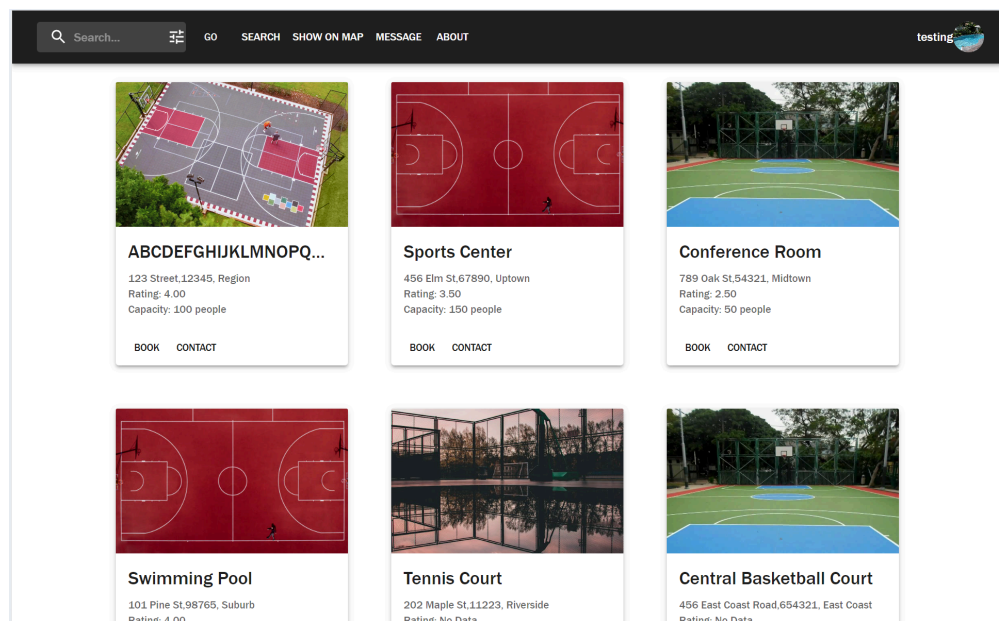
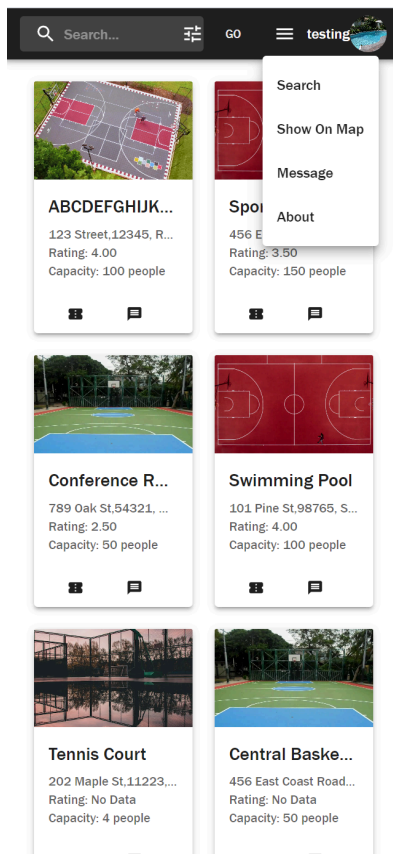
Main Page

- **Menu Options:**

- On large screens, the menu displays full words for options like "Search," "Show on Map," "Message," and "About."
- On smaller screens, these options are condensed into a pop-up menu that appears when the user clicks an icon. This keeps the page tidy and improves user interaction.

- **Buttons:**

- The "Booking" and "Contact" buttons adapt based on screen size. On smaller screens, these buttons display as icons. On larger screens, the buttons display the full text.
- This approach ensures the interface remains clean and easy to use, regardless of screen size, providing a seamless user experience.



Facility View Page

To enhance user experience across different devices, the Facility View page also has been designed to adapt to various screen sizes:

- **Information Cards:**
 - On larger screens, information cards are displayed side by side, making it easy for users to view all details at a glance.
 - On smaller screens, the cards are displayed in an up-and-down layout to improve readability and user interaction.
- **Comment Section:**
 - On larger screens, comments are displayed two per row, allowing users to see more comments simultaneously.
 - On smaller screens, comments are displayed one per row to ensure each comment is easily readable.
- **Other Adjustments:**
 - Button spacing and gaps between comments are adjusted using viewport ratios to maintain a flexible and visually appealing layout across all devices.





ABCDEFGHIJKLMN**OPQRSTUVWXYZ**

123 Street,12345, Region
Rating: 4.00
Capacity: 100 people
Fees/hour: \$50.00
Type: Basketball Court

OWNER

testing



BOOK

CONTACT

Rating



testing★★★★★
Excellent service!



jane_smith★★★★☆
Very good, but room for improvement.



Kong Zi
★★★★☆





ABCDEFGHIJKLMN**OPQRSTUVWXYZ**

123 Street,12345, Region
Rating: 4.00
Capacity: 100 people
Fees/hour: \$50.00
Type: Basketball Court

OWNER

testing



BOOK

CONTACT

Rating



testing★★★★★
Excellent service!



jane_smith★★★★☆
Very good, but room for improvement.



Kong



Map View Page

The Map View page allows users to search for facilities using a map interface. To ensure a seamless experience across all devices, we've implemented a responsive design

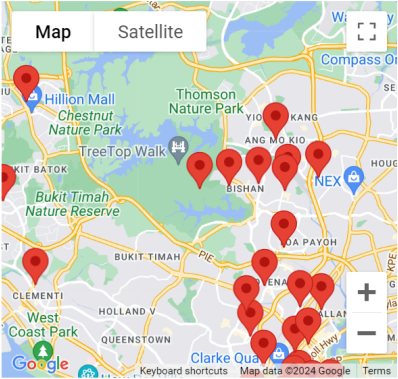
- **Component Sizing:**
 - Image sizes, font sizes, and other components are adjusted using viewport size, allowing them to automatically resize based on screen dimensions.
- **Facility Details Display:**
 - On larger screens, both the image and the complete details of the facility are displayed, providing a comprehensive view.
 - On smaller screens, only the name and address of the facility are displayed to avoid clutter and make it easier for users to browse multiple facilities at once.

Regardless of screen size, users can click on the list below or directly on the map to view more detailed information about a facility.

testing

Map

Satellite



List

Sports Center

456 Elm St,67890, Uptown

Conference Room

789 Oak St,54321, Midtown

Swimming Pool

101 Pine St,98765, Suburb

Tennis Court

202 Maple St,11223, Riverside

East Coast Basketball Court

789 West Avenue,987654, West

Westside Basketball Court

101 North Street,135792, North

Northside Basketball Court

202 South Road,97531, South

Southside Basketball Court

SEARCH

SHOW ON MAP

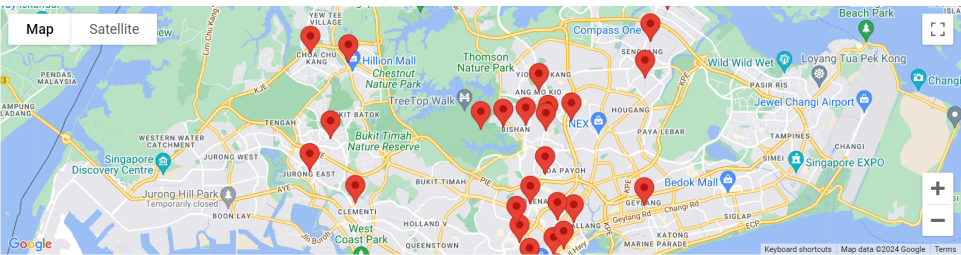
MESSAGE

ABOUT

testing

Map

Satellite



List



Sports Center

456 Elm St,67890, Uptown



Conference Room

789 Oak St,54321, Midtown

Message Page

We’ve made the message page more user-friendly by designing it to adapt to different screen sizes. Here’s how it works:

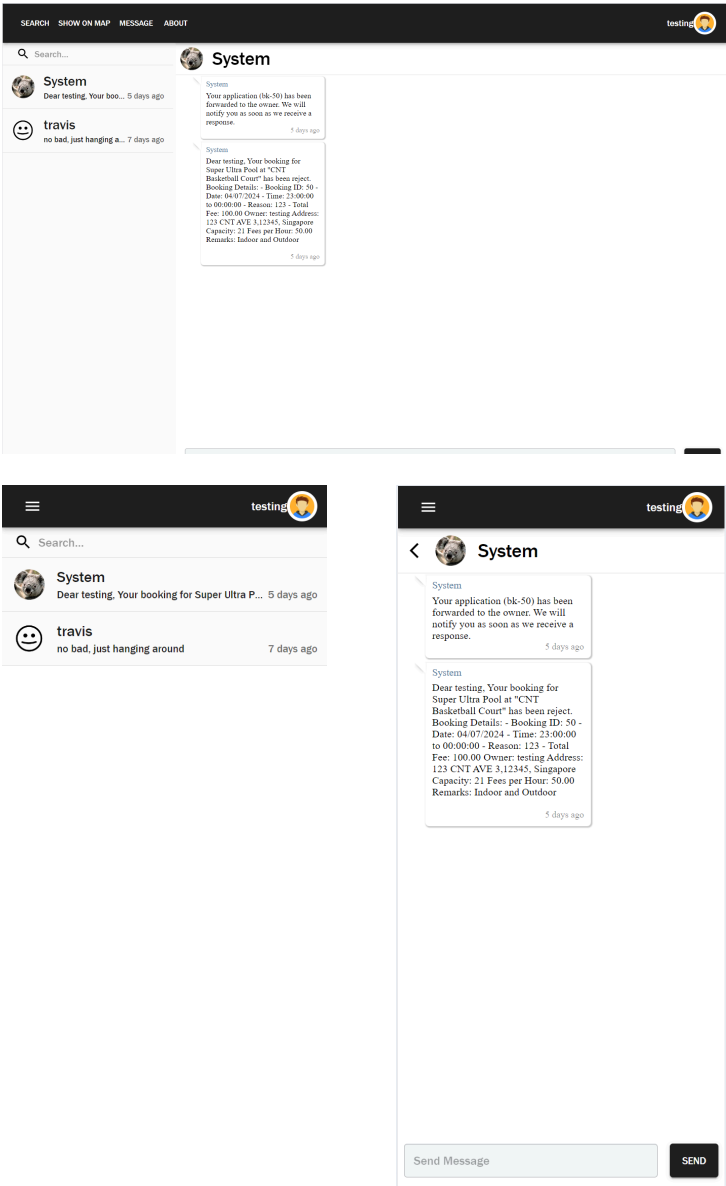
Chat List and Chat Content

Larger Screens

- **Side-by-Side Layout:** On wider screens, you’ll see the chat list and chat content on the same page.

Smaller Screens

- **Full-Screen Chat List:** On smaller screens, the chat list takes up the whole screen. When you select a chat, the chat content shows up and takes over the screen.
- **Back Button:** There’s a back icon to help you navigate back to the chat list after you’re done messaging.



Challenges

The most difficult part of responsive design is looking through the documentation about the components. We have to adjust the design step by step and check if it is convenient for users. We are still unable to fully utilize the props that MUI provides because they are too complicated to use. We hope that in the next milestone, we can better utilize these props to improve our web page design.

Canceled Features

- **Event Creation [canceled]**: This feature for administrators to create and manage events.
- **Send Invitation [canceled]**: This feature to share bookings and invite friends via a link or email.

We've decided these features aren't needed, so we're focusing on what matters most.

Challenges While deploying

We chose to use Amazon AWS RDS for our database because the free trial of our last provider ended. It took some time to set up the settings, and we got stuck on the public access part of the database. Next, we wanted to put our backend code on AWS EC2. We had a lot of trouble setting up the virtual machine. Even after setting it up, we couldn't get any response from the server. Finally, we found out that we mixed up the HTTP and HTTPS protocols. Since our website on Netlify uses HTTPS, we couldn't access the server API with HTTP. So, we had to use Render to deploy our backend code.

Improvement of the Website

1. Use of Local Storage for Storing Usernames

We discovered a significant security issue with how usernames are currently stored. Our web page username is saved in localStorage to keep track of users across different pages. This approach has some risks because the data in localStorage can be easily accessed and changed by anyone using JavaScript. This could lead to data being stolen or altered without permission. Additionally, localStorage doesn't have a built-in way to automatically clear out old data, which can cause outdated or insecure information to persist. Furthermore, not all web browsers handle localStorage consistently; some might delete this data unexpectedly, which can disrupt the user's experience. Moreover, the message features will not be able to differentiate sender and receiver on the same computer if it is loaded in the same browser.

```
useEffect(() => {
  const myUsername = localStorage.getItem('username') || 'testing';
  fetch(`${import.meta.env.VITE_SERVER_URL}/user/${myUsername}`)
    .then((res) => res.json())
    .then((data) => {
      setInitialInfo(data);
      setUsername(data.username);
      setEmail(data.email);
      setPassword(data.passwordHash);
      setImage(data.user_image);
      setRole(data.role);
      setId(data.id);
      setPreview(`${data.user_image}`);
    });
}, []);
```

Proposed Solution

To enhance security and address these vulnerabilities, we propose using cookies to store user identification data instead of localStorage. Cookies provide better control over data management, including expiration and access restrictions. We recommend encrypting the data stored in cookies to protect it from being read or altered by unauthorized parties. This encryption ensures that even if the data is accessed, it remains unreadable without the decryption key.

Using cookies also allows for the implementation of secure attributes. The HttpOnly attribute prevents client-side scripts from accessing the cookie, thereby protecting it from cross-site scripting (XSS) attacks. The Secure attribute ensures that cookies are only transmitted over HTTPS, safeguarding the data during transmission. The SameSite attribute can be configured to prevent cookies from being sent in cross-site requests, reducing the risk of cross-site request forgery (CSRF) attacks.

2. Lack of Type Checking

Our website uses JavaScript, which doesn't have built-in type checking. This can lead to errors with data types that aren't caught until the code runs, causing bugs and unexpected issues.

Although React offers a way to check the types of props passed to components, we haven't used this feature because we weren't familiar with it when we started the project.

Proposed Solution

To improve code quality and make future development easier, we recommend using TypeScript. TypeScript adds type checking to JavaScript, helping catch errors early in the development process. This reduces bugs and makes the codebase easier to maintain and extend. If switching to TypeScript isn't an option, we can start using prop-type checking in our React components. This will help ensure that the data being passed around is of the expected type, making our project more reliable and easier to work with.

3. Inefficient Data Handling

Our current setup does not efficiently manage large amounts of data retrieved from the server. On the main page, all facilities are loaded at once, which can put a heavy load on the server and slow down the website, especially as the number of facilities increases.

Proposed Solution

To improve performance and user experience, we recommend implementing data fetching limits and pagination. This approach involves retrieving only a subset of data at a time and allowing users to navigate through the rest using pages. Additionally, we can implement lazy loading for images and data, ensuring that only the content currently in view is loaded. These techniques can reduce server strain, speed up page load times, and provide a smoother experience for users as they interact with the site.

4. Other known issues

- a. **Messaging System Time Zone:** We've noticed that the messaging system displays timestamps in UTC, which can be confusing for users. We've implemented a temporary fix that adjusts the timestamps to the local time zone upon page refresh. This issue seems to be related to how the messaging package handles time formats.
- b. **Duplicate Broadcast Messages:** Some users have reported receiving the same broadcast message multiple times. We're aware of this bug but haven't pinpointed the exact cause yet.
- c. **Swap Process in Booking Application:** The process for swapping bookings is currently too complicated. It allows multiple swaps and even self-swaps, which can be confusing. We're working on making this feature more user-friendly and straightforward.

Testing

Unit Testing

No	Page	Component	Test Case	Pass/Fail
1	Login Page		render the Login Form properly	Pass
			autofocus on the first input	Pass
			successful login	Pass
			able to handle successful login	Pass
			able to handle unsuccessful login	Pass
2	Sign Up Page		render the signup form properly	Pass
			shows password strength messages	Pass
			able to handle form submission	Pass
			able to handle duplicate username	Pass
			able to handle navigation back	Pass
3	Main Page		fetches and displays facilities on mount	Pass
			able to search for facility	Pass
			able to navigate to booking page	Pass
			fetches facilities on empty search	Pass
4	User Profile Page		renders UserProfilePage component properly	Pass
			able to handle edit button click	Pass
			able to handle form submission	Pass
5	Not Found Page		renders the 404 not found message properly	Pass
6	Email Verification Page		renders the component and displays initial state properly	Pass
			able to handles form submission correctly	Pass
			displays error message when user not found	Pass
			displays generic error message on fetch failure	Pass
7	Facility Owner Page		renders the component and displays loading initially properly	Pass
			fetches and displays facility and ratings data	Pass

			navigates back to booking management page on back button click	Pass
8	Facility User Page		fetches and displays facility details and ratings	Pass
			navigates to booking page on book button click	Pass
			logs contact page message on contact button click	Pass
			navigates back on back button click	Pass
9	Create Facility Page		renders the component and checks initial state	Pass
			able to handle input changes	Pass
			able to handles form submission with success	Pass
			able to handles form submission with error	Pass
			able to handles form reset	Pass
			able to handles back button click	Pass
10	Booking Page		renders correctly	Pass
			able to handles drawer open state	Pass
			able to updates view based on URL param	Pass
			changes current view when button in sidebar is clicked	Pass
		Booking	should render future and history bookings correctly	Pass
			should display error message on fetch failure	Pass
			should delete a booking when cancel button is clicked	Pass
		Facility	fetches and displays facilities	Pass
			displays an error message when the fetch fails	Pass
			navigates to create facility page when button is clicked	Pass
11	Booking Application Page		fetches data and displays FacilityCard and BookingForm with correct data	Pass
			handles back button click	Pass
Backend Code Test				
1		checkIdentity	existing and correct username and password	Pass

			existing and incorrect username or password	Pass
			not existing user	Pass
2		createBookingApplication	created only one application and recorded in database	Pass
			status should be pending	Pass
			other type cannot insert	Pass
3		createFacility	should insert facility information and address	Pass
			should throw an error if the query fails	Pass
4		createRateComment	should insert a rating for the given booking ID and data	Pass
			should throw an error if no matching booking ID is found	Pass
			should throw an error if the query fails	Pass
5		filterFacility	partial restrictions	Pass
			type, distance, priceRange validation	Pass
6		getAverageRatingByld	should return the average rating for the given facility ID	Pass
			should return an empty array if no rating is found	Pass
			should throw an error if the query fails	Pass
7		getBookingDetail	should return booking detail for a given bookingId	Pass
			should return undefined for a non-existing bookingId	Pass
8		getBookingList	gives back detail according to username	Pass
			gives back approve application	Pass
9		getEventsByld	should return events for the given facility ID	Pass
			should return an empty array if no events are found	Pass
			should return an error if the query fails	Pass
10		getFacilityList	should return array of facility by given username	Pass
			should return an empty array if no facility found	Pass
11		getPendingBooking	should return an array of pending booking with the given facility id	Pass

			should return an empty array if no pending booking found	Pass
			should throw an error if query fail	Pass
12		getRatingsById	should return ratings with user info for the given facility id	Pass
			should return an empty array if no ratings are found	Pass
			should throw an error if the query fails	Pass
13		getUser	should return user detail information by the given username	Pass
			should return undefined if user not found	Pass
			should throw an error if duplicate usernames are found	Pass
			should throw error if the query fail	Pass
14		getUserByEmail	should return username for the given email	Pass
			should return undefined if email not found	Pass
			should throw error if the query fail	Pass
15		searching	by bound	Pass
			by id	Pass
			give back one object	Pass
16		searchFacilityByName	should return facilities with their average ratings for a given keyword	Pass
			should return an empty array if no facilities are found	Pass
			should throw an error if the query fails	Pass
17		sendMail	received with new password	Pass
			received email	Pass
18		signup	throw error while duplicate username	Pass
			record user in database	Pass
19		swapBooking	should update the booking timeslot for the given booking id	Pass
			should indicate no row effected if the booking id not found	Pass

			should throw error if the query fail	Pass
20		updateBookingStatus	should update the booking status for the given booking id	Pass
			should indicate no row effected if the booking id not found	Pass
			should throw error if the query fail	Pass
21		updateFacilityProfile	should update the facility information for the given id and data	Pass
			should throw error if the update facility information fail	Pass
			should throw error if the update facility address fail	Pass
22		updatePassword	should update the password for the given password	Pass
			should return result of no column effected for the given email and new password	Pass
			should throw error if the query fail	Pass
23		updateUser	throw error when username is duplicate	Pass
			image name should update	Pass

End to End Testing

No	Tested part	scenario	steps	Pass/Fail	Comment
1	Login Page, Sign up Page	sign up properly	user fill all section and gives unique username	pass	
			user click submit button		
			user redirect to login page		
		sign up duplicate name	User sign up for duplicate name	pass	
			show duplicate message and change duplicated username		
			display duplicate message		
		forgot password	user have existing account	pass	
			click on the forgot password link		
			enter the email for the account and redirect to main page		
			user will receive email include the reset password		
			user use the reset password login to the website		
		wrong account credential	user login using wrong username and password	pass	
			display wrong password or username message		
		correct account credential	user login using correct username and password	pass	
			redirect to main page		
2	Main Page	user search a facility keyword using searchBar	user click the search bar and input some keyword	pass	
			the results should relevant to the keyword (name, owner name, address)		
			user input empty string, results should be all facilities		
3	Show On Map	user search facility using the map and the list below	user navigate to show on map page	pass	
			user scroll on the map and the marker show the precise location		
			user click on the marker show information window that include the information of that particular facility		
			user select one facility from the list and also show the information of the facility		
			user click the book button and redirect to booking page		
4	Booking application Page	standard booking procedure	user select one facility from main page	pass	
			user select date		
			user can only select continuos timeslot		
			user fill the rest of the application form		
			user submit the form and the redirect to main page		

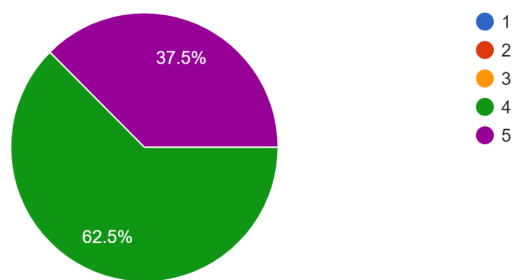
		user partially fill all information and cannot submit	user not select date can submit (auto select date today)	pass	
			user not select time cannot submit		
			user not filling number of people cannot submit		
			user not select the purpose cannot submit		
			user not filling reason cannot submit		
5	Booking Page	user give rating for the application and the section will disappear	user click the rate button and give comments	pass	
			user submit rating and comments		
			the application disappears		
		user delete the past booking	user click the cancel button	pass	
			the applications should disappear immediately		
6	Facility Page	user create a facility and automatically pin the latitude and longitude	user navigate to faciltiy page	pass	
			user click create facility button and be redirected to creating page		
			user upload a photo of the facility		
			user fill the name and the address		
			the map should automatically pin down the address if it exist		
			user fill the rest of the application form		
			user submit the form and redirect to last page if success		
		not complete facility information	user not submit photo can submit	pass	
			user not fill address cannot submit		
			user not pin location can submit		unable to test pin with cypress
			user not fill type, fees and capacity cannot submit		
			user not fill remarks can submit		
		user approve application	user submit application of his own facility	pass	
			user navigate to faciltiy page		
			click view button to the owner view of facility		
			click the approve button of the application		
			application should disappears		
		user check reject the application	user submit application of his own facility	pass	
			user navigate to faciltiy page		
			click view button to the owner view of facility		
			click the reject button of the application		
			application should disappears		
7	Edit Profile Page	user update information and save	user click on edit profile and edit information	pass	
			user update the name and the image		
			user click save button		
			the latest user information will be displayed		

		user update information but cancel	user click on edit profile and edit information	pass	
			user update the name and the image		
			user click cancel button		
			the user information will remains as before		
8	Message Page	user click the contact button and send message to the owner	user navigate to main page	pass	
			user click contact button		
			user redirected to message page and the dialog should be the owner		
			user send message to the owner		
			message should send successfully		
			the message sent show on the message box		
		user search for other user and start conversation	user navigate to message page	pass	
			user search the username of the person		
			the username display should only display the username that match the input		
			user click on username and start conversation		
9	Login Page/ logout features	user logout	navigate to the page the have navigation bar(main page, map view page, message page, about page)	Pass	
			click the user avatar and click logout button		
			should be redirect to login page		

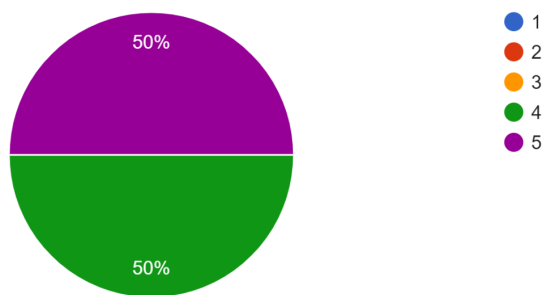
User Testing

1. Account Sign-Up and Log In

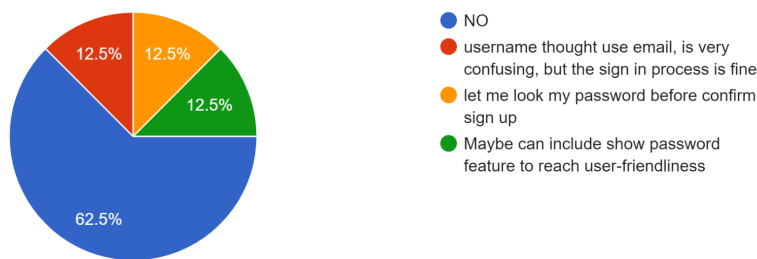
On a scale of 1 to 5, rate the overall experience of the sign-up process.
8 responses



On a scale of 1 to 5, rate the overall experience of the login process.
8 responses



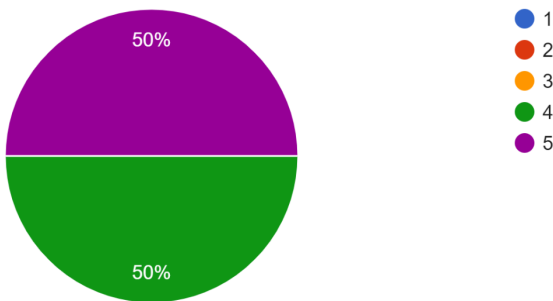
Were there any difficulties or points of confusion during the sign-up or login process? If so, please describe.
8 responses



2. Searching

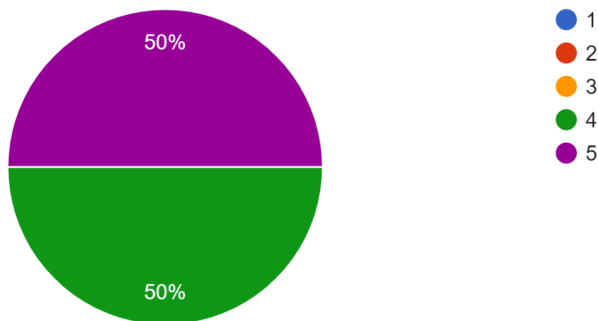
On a scale of 1 to 5, rate the map view searching experience. Was the map easy to navigate and use? Did the map accurately display nearby facilities?

8 responses



On a scale of 1 to 5, rate the experience of searching by keyword. Was the search bar easy to locate and use? Did the search results match your expectations?

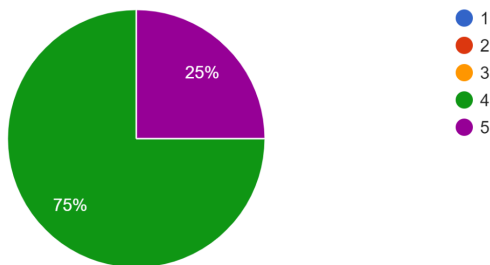
8 responses



3. Booking

On a scale of 1 to 5, rate the experience of the booking process. Is the information needed evident? Is the date and time picker easy to use?

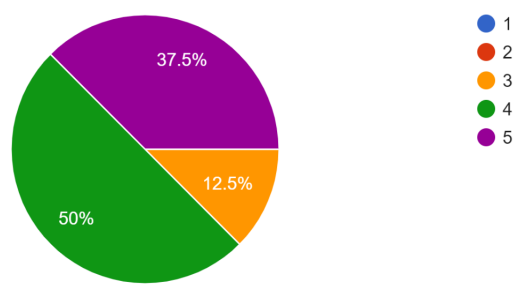
8 responses



4. Messaging

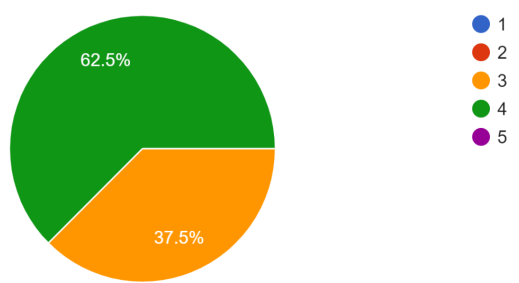
On a scale of 1 to 5, rate the experience of the messaging process. Did the contact button redirect you to the owner dialog? Is the display of the message good? Is finding the user experience seamless?

8 responses



On a scale of 1 to 5, rate the experience of the messaging process. Did the contact button redirect you to the owner dialog? Is the display of the message good? Is finding the user experience seamless?

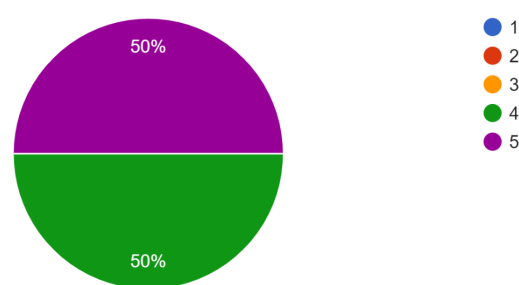
8 responses



5. Facility

On a scale of 1 to 5, rate the experience of the creation process. Does the map auto-pin the address accurately? Is the needed information clear? Does th...icture of the facility display the preview properly?

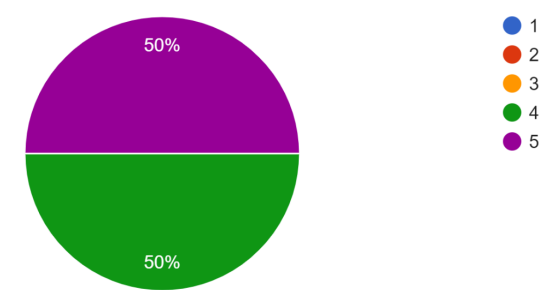
8 responses



6. Navigation

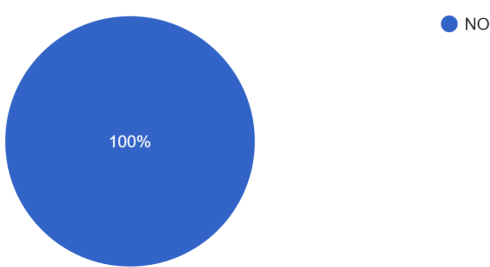
On a scale of 1 to 5, rate the navigation experience.

8 responses



Was there any place that was hard to navigate or confusing? If so, please describe.

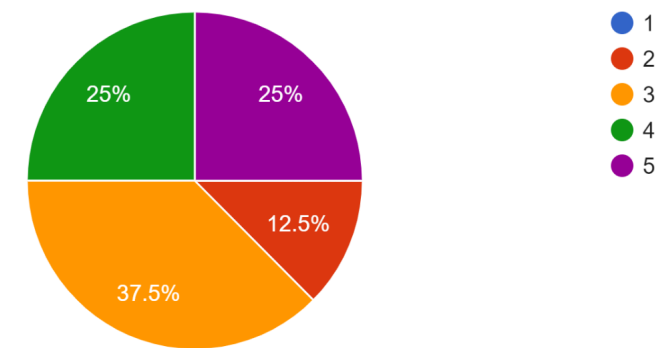
8 responses



7. UI Design

On a scale of 1 to 5, rate the overall UI design of our webpage.

8 responses



Which page is the worst and needs the most improvement? Please explain why.

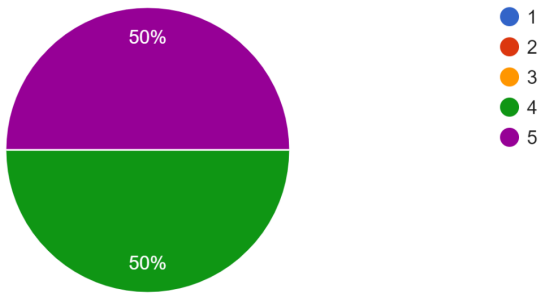
8 responses

I think all is fine
nothing
the message of the system, the way of notification message should to be more user friendly like giving a box interface such as shopee.
the page a little bit empty
Maybe can include colourful UI
message in pc need to scroll down n type
Message, came out two duplicate chat bubble
The bug I met in messaging page, I sent one massage and the system auto duplicate into five but overall is ok

8. Additional Tasks and Question

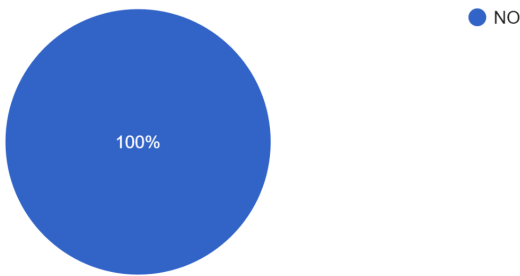
On a scale of 1 to 5, rate the mobile responsiveness of the website.

8 responses



Are there any features or pages that do not work well on mobile? If so, please describe.

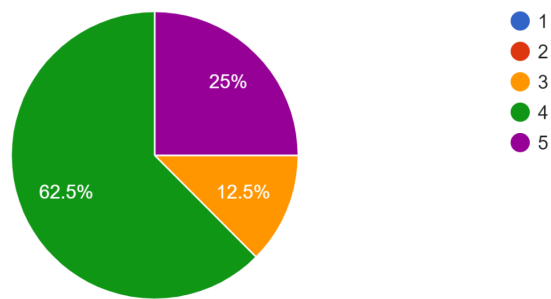
8 responses



9. General Feedback

On a scale of 1 to 5, rate your overall satisfaction with the website.

8 responses

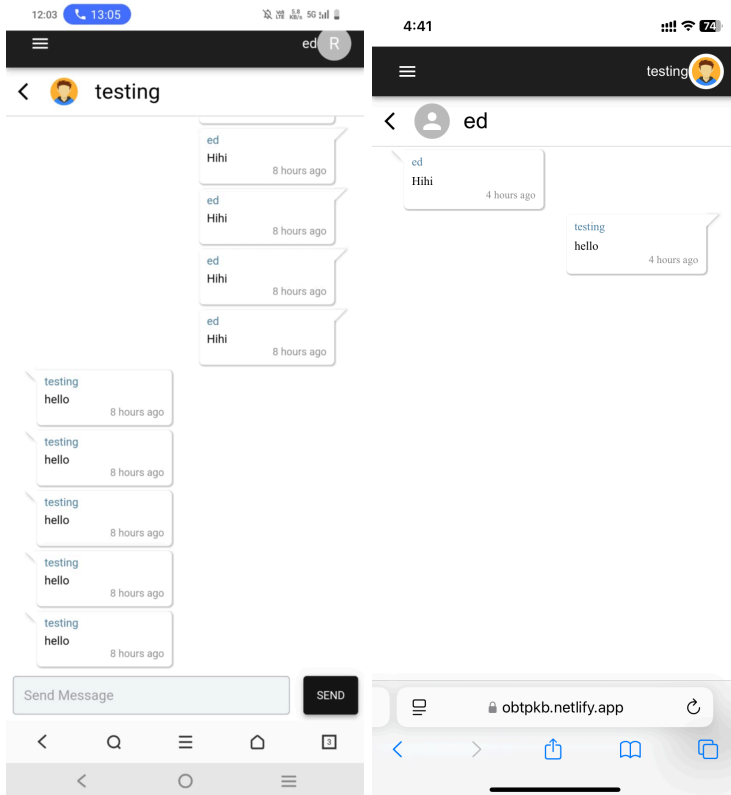


What did you like the most about the website? What did you like the least about the website? Do you have any additional comments or suggestions for improvement?

8 responses

I like all equally, no suggestion, good job
convenient and easy
The page transition is super smooth, and the buttons are very user friendly. What I don't like is that it doesn't automatically log out of the system. I don't have to log in again when I close the webpage and undo it.
the responsiveness is good no lacking but can add some image because the page is little bit empty
This websites fully deploy as a professional booking system. No
very simple to use
None
The most is every facility is connect to the map and the map function is interesting. The least is the the messaging page because of the bug. After debugging is ok

Bug feedback from the user testing



During our testing, we discovered a bug in our messaging system. This bug causes some users to send or receive messages two to five times on their screens, while everything appears normal on our end. Additionally, the system's time display occasionally shows errors, displaying UTC time instead of the user's local time zone.

Tech Stack

- Frontend Development: HTML, CSS, Vite, React JS, MUI, JavaScript
- Backend Development: Node JS, Express JS, JavaScript
- Database: MySQL
- Testing: [Vitest](#), React Testing Library, jest dom, Cypress
- Version Control: git, GitHub
- Poster & video: Canva, Adobe Premiere pro

Frontend Development

Framework and Bundler

For our frontend, we're using React.js. To make things easier and faster, we have chosen Vite as our bundler. It helps us manage the project smoothly and reloads pages super quickly, which is a big plus.

UI Components and Styling

We're also using the MUI library. It gives us a ton of ready-to-use components, so we don't have to build everything from scratch. We can pick what we need and tweak it to fit our style. MUI offers different ways to style components—through props, themes, or even regular CSS modules like in typical React projects.

JavaScript for Logic and Communication

For basic logic, like checking the continuous time slot selection, we rely on JavaScript on the frontend. We also use JavaScript to connect with the server, whether it's to post data or fetch the info we need.

Key Packages

There are a few important packages we'd like to mention that have been incredibly helpful during our development process:

- **Day.js and Moment.js:** We use these for handling date and time. They allow us to generate formatted date-time strings and easily change the format of dates and times as needed.
- **React Scheduler:** This package provides a template for the timeslot, which we can modify to create a time selection feature. It has been essential for customizing how our timeslots behave.
- **React Router DOM:** To enable navigation between different pages on our website, we use React Router DOM. This makes it easy to set up routes and handle navigation smoothly.

Backend Development and Database

Node.js and Express.js

For our backend, we're using Node.js as our JavaScript environment and Express.js as our framework to create the server. We chose Express because it's simple and scalable, making it perfect for our needs.

Key Packages

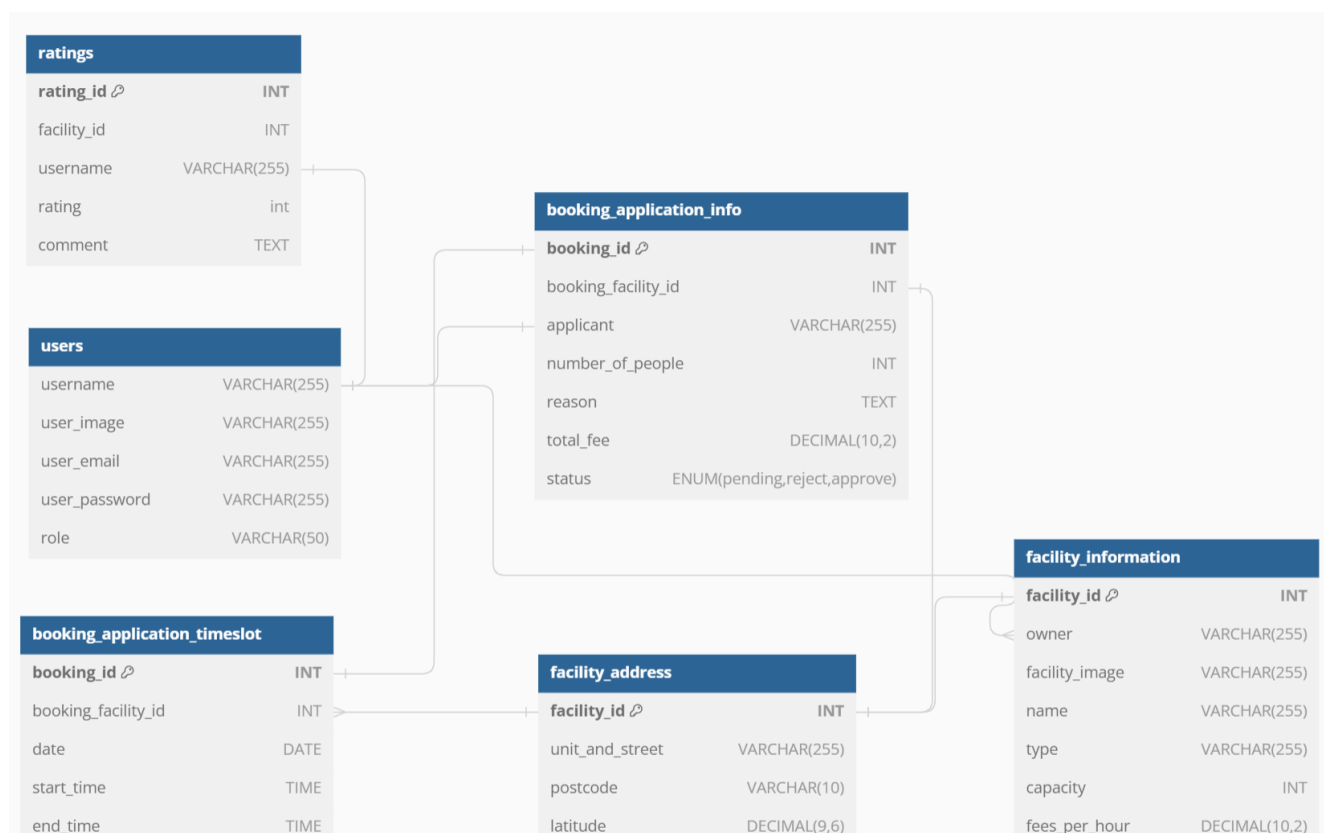
There are several important packages that have greatly aided our backend development:

- **CORS:** This package allows our frontend and backend to communicate across different domains during development.
- **Multer:** We use Multer to handle file uploads, such as images.
- **Socket.IO:** Socket.IO is essential for our project as it provides real-time, bidirectional communication between your application's clients and server, enabling instant updates and interactive features.
- **Other useful packages:** Other packages like nodemailer, dotenv and more have also been very useful in our project.

Database

We use a relational database to store our data because it fits our needs and keeps everything organized. The database we choose is MySQL. Our queries are pretty straightforward, but we do have some view tables to make things easier.

- **Data Storage:** We only store text data (words) in the database, not images. Storing images in the database could slow down our queries if we end up with a lot of data. Instead, we'll store images in the cloud and fetch them when needed.
- **Database design:**



Version Control

For version control, we use Git and GitHub to organize our workflow. Each of us has our own branch and we work on our parts separately. After we finish our part, we push our code to GitHub and create a pull request to merge it into the main branch. Once that's done, we pull the latest code from the main branch to keep the project up to date so we can continue working on the next part

Software Engineering Practices

- Agile/iterative process
- Unified Modeling Language (UML), user flow diagrams, activity diagrams
- Continuous integration (CI) / continuous deployment (CD)
- 'KISS' principle
- Comment Minimally, But Sufficiently
- Delete dead code

Overall development progress

Our project follows an agile process. Before we start implementing features, we plan and design them carefully. After development, we test each feature to make sure it works, either manually or with automated testing. Our approach is iterative: first, we build the basic structure of our webpage without fancy styling. We focus on making sure the essential features work well. Then, we refine the project. This means we improve how the code is written to increase its readability and extensibility. We also use libraries to make the webpage better for users.

Frontend

We start our web development project with no prior development experience. Therefore, we first learn about fundamental frontend tools such as HTML, CSS, and JavaScript. Our early development stages did not involve any frameworks; we relied solely on these basic technologies to build our project from the ground up.

Transition to React.js

Recognizing the potential for a more efficient development process, we decided to switch to using the React.js framework. The reasons for this transition were manifold:

- **Ease of Development:** React.js simplifies complex tasks and offers a more structured approach to coding.
- **Organized Codebase:** React.js allows for the separation of code into different components, enhancing readability and maintainability.
- **Extensive Ecosystem:** The framework's rich ecosystem and the vast number of available resources can greatly accelerate development progress.

Libraries

As we delved deeper into styling, we discovered Material-UI (MUI) as a valuable resource. MUI provided essential and basic components that seamlessly integrated into our project. This not only standardized the appearance of our application but also reduced the amount of custom styling needed. Our primary task became connecting these pre-built components to our application's functionality.

Responsive Design

Understanding the importance of accessibility and usability, we committed to making our webpage responsive. This ensures that our application can be easily used across a variety of devices, including desktops, tablets, and smartphones. Our goal is to create a convenient and user-friendly platform that caters to users of all ages and adapts to the diverse range of devices they use for daily operations.

Backend

Our journey into backend development was met with several challenges. Initially, we lacked an understanding of the purpose and functionalities of backend development, as well as which framework would be most suitable for our needs.

Choosing Node.js

After some research, we chose Node.js for our backend development due to its popularity and ease of learning for beginners. Node.js is known for its efficiency and scalability, making it a fitting choice for our project.

Learning Key Concepts

As we progressed, we learned several fundamental concepts crucial to backend development:

- **Form Data and JSON Data Transfer:** We explored how to handle and transfer data between the frontend and backend, focusing on both form data and JSON data formats.
- **Middleware:** We understood the importance of middleware in handling requests and responses, performing various operations like authentication and error handling.
- **CORS (Cross-Origin Resource Sharing):** We learned to manage CORS policies to control how resources are shared between different origins, ensuring secure and smooth interactions.

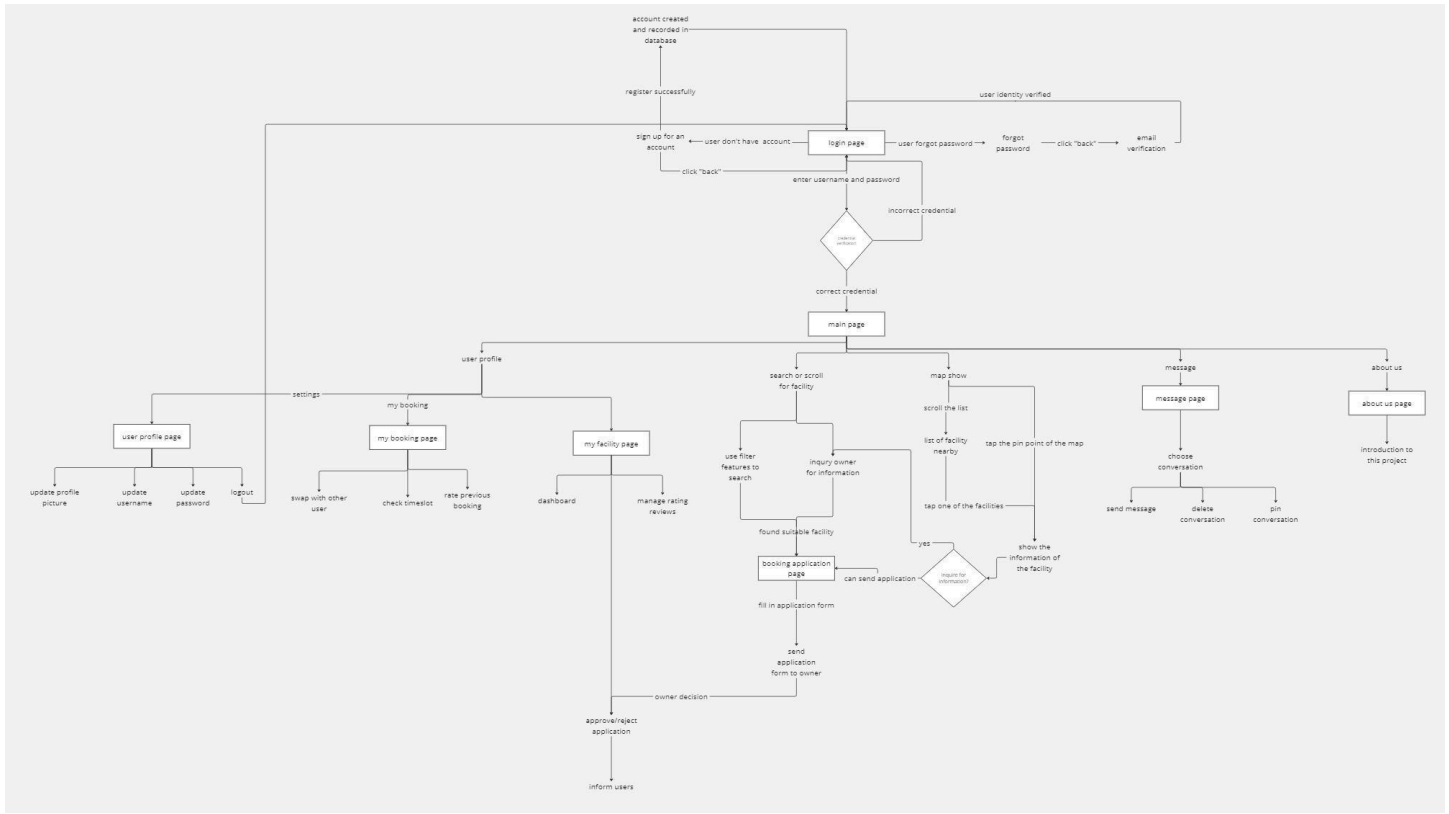
Refactoring with Routers

Initially, our server-side code was lengthy and difficult to manage, with all operations written in a single file. This approach quickly proved unsustainable as our project grew in complexity. To address this, we learned about routing and the importance of separating server code into modular components. By using routers, we were able to:

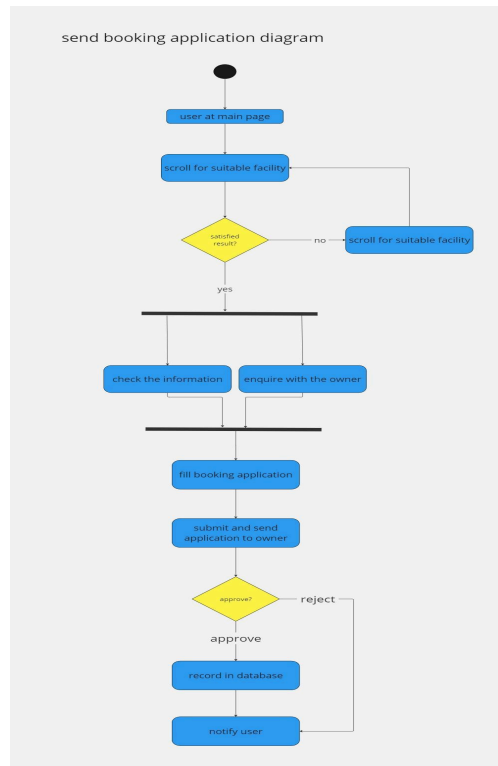
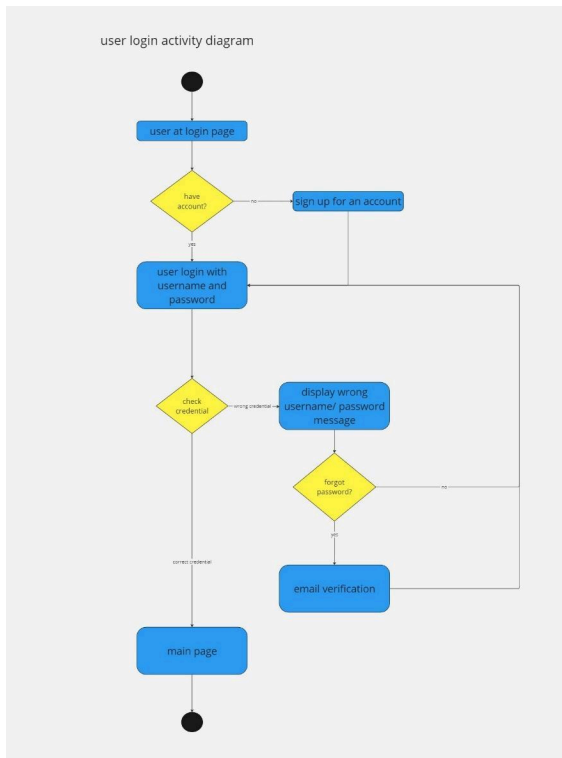
- **Improve Code Organization:** Each file was designated to handle specific tasks, making the codebase cleaner and easier to navigate.
- **Enhance Maintainability:** With a modular structure, it became easier to update, debug, and maintain the code.

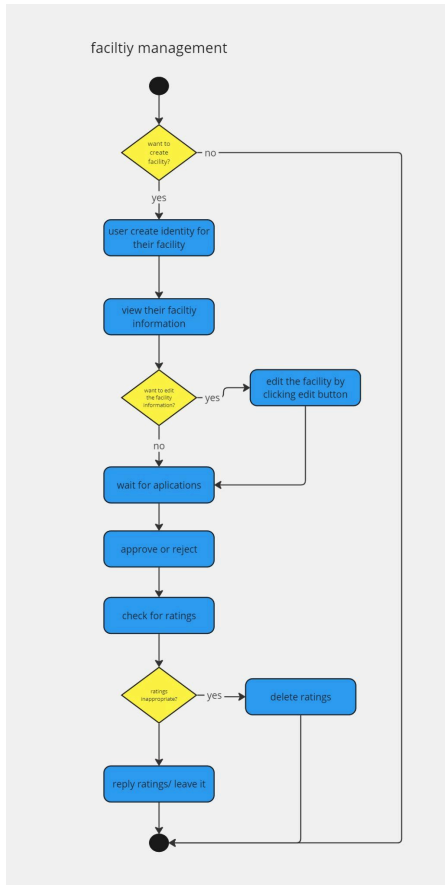
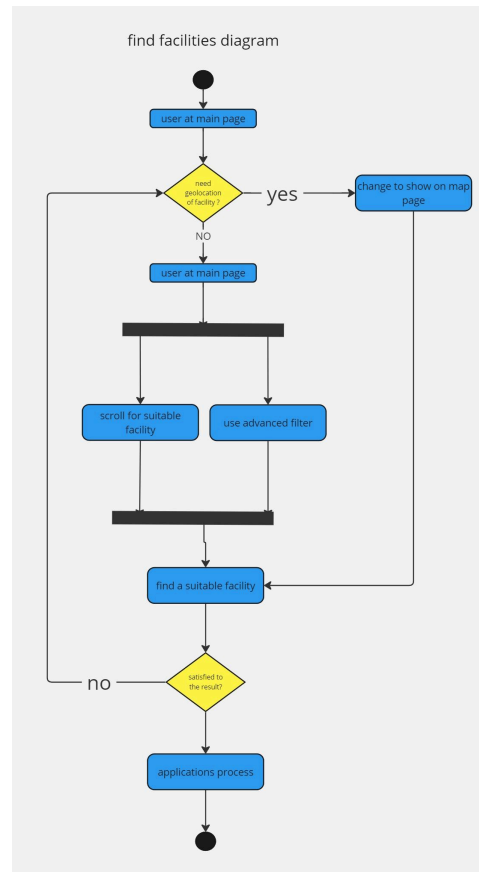
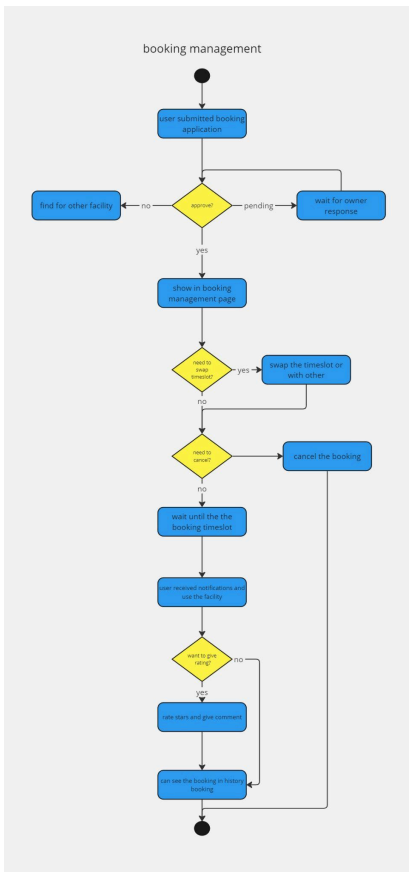
Activity Diagram

User flow Map



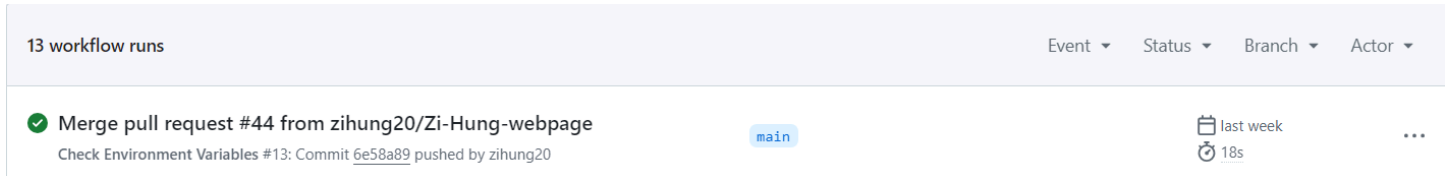
Activity diagram





CI / CD

For Continuous Integration (CI), we use pull requests to ensure code quality before merging changes into the main branch. When a pull request is made, a YAML file will validate that environment variables are correctly set up, given that we don't use a real API during development. We rely on GitHub Actions to handle the CI process, running tests and checks to confirm that everything is functioning correctly before merging.



For Continuous Deployment (CD), we use Netlify, which integrates with our GitHub repository. Netlify automatically deploys our website whenever changes are pushed to the main branch. This setup streamlines the deployment process, as Netlify handles updates and deployments automatically, removing the need for manual intervention.

Comment Minimally, But Sufficiently

When collaborating with other team members, it's crucial to maintain a balance between writing sufficient comments for understanding and avoiding excessive comments that hinder readability. Our guideline is to comment code only when necessary to enhance clarity. Here are the principles we follow

Self-Explanatory Code:

- Code that is clear and evident from the function or variable names should not have additional comments.

```
const handleReset = () => {
  setFormData({
    owner: myUsername,
    facilityPicture: null,
    facilityName: '',
    unitAndStreet: '',
    postcode: '',
    region: '',
    latitude: '',
    longitude: '',
    facilityType: '',
    capacity: '',
    fees: '',
    remarks: '',
  });
  setImagePreview('');
  setError(null);
  setSuccess(false);
};

const handleSubmit = async (e) => {
  e.preventDefault();
  const form = new FormData();
  for (const key in formData) {
    form.append(key, formData[key]);
  }

  try {
    const response = await fetch(
      `${import.meta.env.VITE_SERVER_URL}/facility`,
      {
        method: 'POST',
        body: form,
      }
    );
    const result = await response.json();
    if (result.success) {
      setSuccess(true);
      navigate('/bookingManagementPage/facility');
    } else {
      throw new Error(result.message || 'Failed to create facility');
    }
  } catch (error) {
    setError(error.message);
    setSuccess(false);
  }
};

const checkPassword = (password) => {
  let strength = 0;
  if (password.match(/[a-zA-Z]/)) {
    strength += 1;
  }
  if (password.match(/[0-9]/)) {
    strength += 1;
  }
  if (password.length == 0) {
    setMessage(
      'Your password must contain at least one letter (A-Z, a-z) and one number (0-9)',
    );
  } else if (password.length < 6) {
    setMessage('Minimum number of characters is 6');
  } else if (password.length > 12) {
    setMessage('Maximum number of characters is 12');
  } else if (
    password.length >= 6 &&
    password.length <= 12 &&
    strength !== 2
  ) {
    setMessage('Password is not good enough');
  } else if (
    password.length >= 6 &&
    password.length <= 12 &&
    strength === 2
  ) {
    setMessage('Good password');
  }

  setStrength(strength * 50);
};
```

The function names `handleReset`, `handleSubmit` and `checkPassword` clearly indicate their purposes, so no comments are needed to explain what these functions do.

Complex Logic:

- For complex or non-intuitive logic, comments are necessary to explain what the code does.

```
//geocoding: address -> lat lng
useEffect(() => {
  const unitAndStreet = formData.unitAndStreet;
  const postcode = formData.postcode;
  const region = formData.region;
  clearTimeout(id);
  setId(
    setTimeout(() => {
      if (unitAndStreet !== '' && postcode !== '' && region !== '') {
        console.log('fully Check');
        const fullAddress = unitAndStreet + ', ' + postcode + ', ' + region;
        fromAddress(fullAddress)
          .then(({ results }) => {
            const { lat, lng } = results[0].geometry.location;
            console.log(lat, lng);
            setCoordinate(lat, lng);
          })
          .catch(console.error);
      }
    }, 1000),
  );
}, [formData.unitAndStreet, formData.postcode, formData.region]);
```

```
//edit mode auto input all the data
useEffect(() => {
  if (data !== undefined) {
    setImagePreview(data.facilityPicture);
    setFormData((pre) => ({
      ...pre,
      ...data,
    }));
  }
}, [data]);
```

Here, the `useEffect` hook performs an operation after the address information is complete, converting it to latitude and longitude. Comments are added to explain this logic. Consider a React component where the logic differentiates between edit and create modes based on the presence of data. We use the `useEffect` hook to handle this logic, and we add comments to make it clear.

Removing Dead Code

During the development process, it is common to use sample data for testing purposes. Once the code is verified to be working correctly, this sample data may be forgotten and left within the codebase. However, since the code will automatically update the sample data with data from the database, the sample data becomes redundant and is no longer needed.

```
const sampleData = [
  {
    event_id: 1,
    title: 'Event 1',
    start: new Date('2024/6/11 09:00'),
    end: new Date('2024/6/11 10:00'),
    color: 'red',
  },
  {
    event_id: 2,
    title: 'Event 2',
    start: new Date('2024/6/11 10:00'),
    end: new Date('2024/6/11 11:00'),
    color: 'red',
  },
];

function Table({ initialDate, setForm, facilityId }) {
  const [force, forceUpdate] = useState(1);
  const [startTime, setStartTime] = useState(dayjs());
  const [endTime, setEndTime] = useState(dayjs().add(2, 'hour'));
  const [data, setData] = useState(sampleData);
  const [viewStart, setViewStart] = useState(8);
  const [viewEnd, setViewEnd] = useState(20);
```

```
function Table({ initialDate, setForm, facilityId }) {
  const [force, forceUpdate] = useState(1);
  const [startTime, setStartTime] = useState(dayjs());
  const [endTime, setEndTime] = useState(dayjs().add(2, 'hour'));
  const [data, setData] = useState([]);
  const [viewStart, setViewStart] = useState(8);
  const [viewEnd, setViewEnd] = useState(20);
```

delete unuse code  zihung20 committed 3 weeks ago	243e873  
remove unused code  tanyqcollin committed last month	033d65a  

Our GitHub commit history often includes commits related to cleaning up the code. These commits are part of our ongoing efforts to remove dead code and keep the codebase clean and efficient.

Important link and documents

1. [Project poster](#)
2. [Project video](#)
3. [Project log](#)

Sample account

Username: testing / travis

Password: password123 / travis123

*server will take more than one minute to response with inactivity, please wait for a while after click any operation

Instructions

1. **Login Page** Access our web page using this [link](#), you will redirect to Login Page. Use the provided sample account to log in to the website. Please note that the login process may take a while to respond.
2. **Forgot Your Password** If you forgot your password, there is a hyperlink to reset it. Clicking the link will redirect you to a page where you can enter your email address for password recovery. After entering your email address, click the "Send" button to receive a recovery email. The email will contain a new password.
3. **Sign Up** For new users, there is a hyperlink to sign up. Clicking the link will redirect you to a sign-up page where you can create a new account. You can enter a username, email, and password without using your real information, and select the role for the account. However, if you use a fake email account, you will not be able to receive the password recovery email. After filling in the details, click the "Sign Up" button to complete the process.
4. **Main Page** After logging in from the login page, you will be redirected to the main page. The main page features a search bar and a list of facilities. You can use the search bar to find facilities by entering keywords, which will filter the list accordingly. For example, if you search for "hall," the results will display facilities that include the keyword "hall." You can also use the advanced filter to search for facilities; the options change dynamically according to the database data. You can scroll for the facility you want and click on the card. It will redirect you to another page.
5. **Facility Information Page** On this page, you can see detailed information about the facility, including its full address, owner, and rating. The ratings will show the stars that users have given and their comments about the facility. You can use these reviews as a reference to decide whether to book the facility or not.

6. **Show on Map** You can navigate to the "Show on Map" page to visualize the location of available facilities. The facility list at the bottom dynamically changes according to the bounds of the map. Click on the marker or the list to see more information about the facility. You can choose to book the facility by clicking the "Book" button.
7. **Booking Process** To book a facility, click the "Book" button, which will redirect you to a page containing the facility's current information and a form to fill out. You will need to select the starting date, starting time, ending time, number of people, and purpose. The "Reason" input box is optional. Once you have completed all fields, click the "Submit" button to submit your application. If you wish to clear the form data, click the "Reset" button to clear the form. After successful submission, it will redirect you to the main page. Please wait patiently.
8. **Create a Facility** This page allows users to upload their facility information to our database, making it visible to others. The form includes fields for the facility name, picture, address, type, capacity, fee per hour, and remarks. Additionally, a map API is integrated into the page, enabling users to precisely pinpoint the facility's location to prevent errors. Users can click on the map to pin the facility's location, which will display the longitude and latitude and save them to our database. After successful submission, it will redirect you to the last page. Please wait patiently.
9. **Check Booking** The booking you made will be under the booking section (click on your profile picture if you are on the main page). Future bookings are those with dates and times in the future; you can choose to swap the time slot, delete, or check the details of the booking application. For past booking applications, you can choose to rate the booking or delete (cancel) the booking history.
10. **Facilities Management** In the Facilities section, you can create and manage your facilities. To add a new facility, upload a photo and provide the facility's name, address, capacity, fee rate, and any additional remarks. Once the submission is complete, you will be redirected to the previous page. Please allow a moment for the process to finish. By clicking on a facility, you can view detailed data related to its bookings, including total revenue, total bookings, daily revenue, and daily bookings. This data is presented alongside visual charts for better analysis. Additionally, the dashboard includes a table of today's bookings so you can see how many users are scheduled to use your facilities.
11. **Edit Profile** You can change your profile picture after logging into the web application. Choose a picture you like and upload it through the website. It will be recorded in our database and fetched from the internet. You can also change other information like your username and password if you want to.
12. **Messaging** You can chat with facility owners or other users to get more info about their experiences. Just type in the username of the person you want to reach or click the contact button on the main page. Then, type your message and hit send to start the conversation.