Manasi

First slide: Greetings

Good morning TA and everyone. My name is Manasi, and my team members are Mayukhi, Glynis, Chen Jie, Glenn and Yunjia. We are team C5 for lab group SCSD, and we will be presenting on our project, SpotEase.

Next slide: Table of contents, Our team

This is our table of contents for today. Before we dive into the content, let us briefly introduce our work distribution. Our team worked mainly on 2 sides of the project. We have Mayukhi, Glynis and Yunjia work on the front-end while Chen Jie, Glenn and myself worked on the back-end.

Next slide: Problem statement and key functionalities

Moving on to our problem statement. Have you ever travelled to a location and realised there are no car parks nearby? Or driven all the way to a carpark, just to find it completely full? This is a common and frustrating experience for many drivers. The lack of real-time information about parking availability often leads to wasted time and increased fuel consumption, which in our economy is a no-no.

Next slide: Problem statement and key functionalities

That's where our application, SpotEase, comes in. SpotEase is built to tackle parking challenges in Singapore's dense urban environment. SpotEase aims to make parking hassle-free by providing real-time updates on parking availability, helping users reduce search time and optimize navigation.

Our target users of this application include daily commuters, casual drivers, and delivery or ride-hailing drivers who face challenges finding parking in Singapore's high-demand areas.

Key functionalities of our application include that it is customized according to the weather — such as recommending sheltered parking during rainy days — to aid users in selecting their desired car park.

To include, our application allows users to save their parking location, so they can easily find their car later in case they forget where they parked. There is also a search history feature which stores previously selected destinations, and users can directly click on previously selected destinations to find car parks in the area again.

yunjia

Next slide: Tech Stack Overview

Thanks manasi. Next, let's move on to the tech stack overview.

Next slide: Tech Stack Overview

SpotEase is a mobile application built with React Native, a JavaScript framework for developing cross-platform mobile apps with native-like performance. The app uses Expo's command-line interface to streamline development. MongoDB serves as the backend database, storing the car park details, profiles and search history. It enables efficient data management and retrieval. SpotEase also integrates with external APIs such as OneMap

API, NEA 2-hour Weather Forecast API and Carpark Availability API from data.gov.sg. The Carpark Availability API returns the number of available lots, total lots and timestamp from the server side.

The list of dependencies for SpotEase can be found in the ‘package.json’ file within the source code.

Next slide: System design components
Now, let’s move on to the System Design Components. In this section, we’ll go through the key design elements that make up SpotEase. We'll walk through various diagrams and explain how they fit together.

Next slide: Use Case Diagram
The use case diagram for SpotEase illustrates the interactions between users and the system, highlighting key functionalities of the application. There are two primary types of users: Guests and Authenticated Users. Both types of users can perform functions like search destinations and get navigation. Authenticated users have additional access to features such as login, register account, forget password, and the ability to view their usage history.

Next Slide: System architecture Diagram
For our system architecture diagram, we used a 3-layered design. At the top, we have the presentation layer, where boundary classes are responsible for our user interface. This layer includes several screens such as loginpage, UserMapView. Each of these components serves as a boundary that facilitates communication between the user and the system. Control classes which handle the main business logic, for example search manager helps process search queries and car park manager which manages car park availability and related details. Lastly, we have our entity class, MongoDB, which serves as the main database in the data access layer. It is responsible for storing persistent data..

mayukhi
Next Slide: Class Diagram
Thanks Yun Jia. I will now be sharing our class diagram. Just like in our system design we have separated them into 3 class objects. For example, our login page boundary class, responsible for displaying open boxes of data fields of email and password. Once the user clicks on the login button, the control object authenticatemanager will use the function validatecredentials to process the user request by calling the database to validate. And if successful, login.js will redirect to the main page.

Next Slide: Sequence Diagram - Login&Guest
Based on the use case diagram shown before, I'll be going through a few of the sequence diagrams. Starting with the login/guest sequence, users can either log in with credentials or choose to continue as a guest. When a user logs in, the system first passes their credentials to the AuthenticationMgr, which then communicates with the database to validate their credential. If verified, it loads the map interface. If credentials are invalid, an error message is displayed. For guest users, the map interface loads directly.

Next Slide: Sequence Diagram - SearchDestination
Next, in the search destination sequence, the user enters a destination, and the search manager retrieves and displays relevant search results. Upon selecting a destination, the system fetches nearby car park details through the database and its geolocation information through OneMaps API. It then calculates distances of each nearby car park and displays the car parks as markers on the map. The user can then view details of the car parks or filter to sort the display of the car park lost based on distance, sheltered parking or weather recommendations. Once a car park is selected, the system prepares to display the route.

Next Slide: Sequence Diagram - GetNavigation
Next, in the get navigation sequence, the navigation manager sends the selected destination to the OneMaps API to retrieve route information. Once received, it displays the route from the user's location to the chosen car park, providing easy navigation guidance.

glynis
Next Slide: Testing
Thanks Mayukhi. Next, we will be moving on to black box testing.

Next Slide: Black box testing - Login
For the login use case, we used black box testing to verify the system behaviour. The main input parameters for this feature are **email** and **password**, both of which can either be valid or invalid. Since the inputs are discrete, we designed our test cases to systematically check different combinations of valid and invalid values.

To isolate issues effectively, we ensured that for each invalid test case, only one input parameter was invalid at a time. For instance:

- In Test ID 2b, we tested an invalid email format.
- In Test ID 2c, we tested an unregistered email.
- In Test ID 2d, we tested a valid email with an invalid password.

This approach helps to pinpoint which parameter causes the login failure, making debugging more efficient.

We also tested boundary scenarios like empty fields in Test ID 2a, where both email and password were left blank, and the system correctly displayed an error.

Finally, in Test ID 2e, we tested the positive path where both input fields were valid. The system responded appropriately by logging in the user, loading the application home page, and displaying the map.

All test cases passed, showing that the login function handles input validation and error messaging as expected."

Next Slide: Software Engineering Practices & Design Patterns

Now, we move on to talk about Software Engineering Practices and Design Patterns we applied.

Next Slide: Software Engineering Practices & Design Patterns
For software engineering practices, we mainly followed incremental development where we were doing specification, development and validation at the same time. One example is that when we are working on a function, the others are coming up with the next function's specification and at the same time validating the functionality of another function.

We also followed the 3 layered architecture, where we split the program into 3 main parts: the presentation or the frontend, app logic or the backend and persistent data or in our case, our user database.

Same Slide: Design Patterns
We used a MVC design pattern while coding the app.
We have the view which is our app frontend and UI,
the controller being the functions responsible for fetching,calculating and filtering the carpark and other data,
and the model being our database where we store user data and map data every time the user selects a route, or log in/register.

We also tried using the single responsibility principle where each class is responsible for a specific task ensuring better modularity.

Next slide: project management
Our team maintained effective project management through regular weekly meetings, where we discussed upcoming tasks, reviewed progress, and planned our next steps. These meetings helped ensure everyone was aligned and could contribute effectively to different parts of the project.

For version control, we used GitHub and worked mainly on the main branch, updating it periodically with new features and fixes. To prevent data loss or accidental overwrites, we made it a point to regularly back up our code, either through manual backups or by saving copies in separate folders. Although we did not use separate branches, we communicated clearly about who was working on what to avoid merge conflicts and maintain smooth collaboration.

This combination of consistent communication and disciplined code management helped keep the project organized and on track.

Next Slide: Live Demo
Now Glenn and Chen jie will give a demo of our app.

Live Demo - Glenn & CJ

Open the app, start from login page

Allowing Location
Accept the permission to allow location access

Guest

1. User click continue as guest
2. Explain that history does not work for guest user
3. Show notification that ‘user is not authenticated’ when guest user goes history page

Register account

1. Enter a valid email address
2. Choose a password and confirm the password
3. Test cases:
    a. Input an invalid email address
    b. Use passwords that do not match
    c. Use a weak password

User Login

1. User logs in using their email and password
2. Show beforehome page with search history previously saved
3. Test case:
    a. Forget password and reset password

Search Destination

1. Enter location you wish to go
2. Select the destination
3. Apply the filters
    a. Show the change when distance radius filter is adjusted
    b. Show what happens when weather parking recommendation is turned on
    c. Show what happens when sheltered parking is turned on
    d. Show that multiple filters can be selected
4. Select a carpark
5. Navigate
    a. Show video from slides

Search History

1. Show the past searches
2. Clear history
3. Test case:
    a. Show the past searches in beforeHome

Parking Location

1. Upload a picture from gallery
2. Add description
3. Save location
4. View saved picture
5. Clear saved location
6. Test case:
    a. Log out and log in again to see the saved picture again
    b. Look at the saved picture from beforeHome page