Good morning Professor and everyone. My name is Jing Jie, and my team members are Yi Feng, Jumana, Damien, Jing Woon and Hao Ren.
We will be presenting on our project, CarparkGoWhere.

<Next Slide : Table of content>

This is the table of contents for today. We will cover our team, our project problem statement, our solution carparkgowhere, the system design components, traceability, our swe practices followed by the design patterns and lastly the live demo.

<Next Slide : Introduction to our team>
Our team worked mainly on 2 sides of the project, we have Jing Woon and Jumana working on the front-end while Damien, Yi Feng, Hao Ren and myself work on the back-end.

<Next Slide : Introduction to our problem statement>
Moving on to our problem statement. Have you ever travelled to a location and realised there are no Car parks nearby and when you drive to the nearest car park, it is already full? You then have to decide whether to wait for a spot at the current car park, potentially facing a long wait time, or to take a chance on the next nearest one, risking the possibility that just two more minutes of waiting would have secured you a spot.
With high vehicle ownership numbers in singapore, Singapore faces a significant parking problem, particularly in densely populated areas such as the Central Business District area and even at typical HDB car parks after working hours.

<Next Slide : Introduction to our project solution>
To attempt to resolve this issue, Team 46 aims to develop a website that allows users to find the nearest, cheapest or closest car park based on their destination. Our team created a website that assists the user in choosing the car parks based on the user's destination, price and distance of the user's location to the car park, and provides the fastest route to the selected car park. Our website provides more accurate real-time data and a streamlined experience, our platform delivers enhanced features to help users reach their destination more effectively than existing solutions.

Our website aims to reduce the hassle and cost of users having to drive to the car parks to check the price rates. Arriving at car parks only to find no available spaces and facing long wait times and then having to circle around to find alternate car parks.

Next I pass my time to Yi Feng.

<Next Slide: Use Case Diagram - explain main functionality>
Thank you Jing Jie.
Now I will talk about the use case diagram for our project. Once the user opens the website, the system displays a login page, prompting the User to input his username and password to sign

in, users are required to register an account if they want to login or they may continue as a guest account. Users are also able to change their password if they forget their password or when they want to change to a new password.

Once the user login through a registered account or using a guest account , the system will transition into the map interface. The system will display a list of recent parking history if there is any and users are able to select the recently selected car parks from the history.

The user may search for their destination using a destination name or postal code in the search bar. Once a destination is set, the system queries real-time parking data from HDB Carpark API for parking lot spaces near the specified destination. The system displays car park markers around the destination in the map interface and shows a list of car parks on a sidebar list on the map interface with details like carpark name , price rate , lots availability and distance from user location to car park. Users are also able to filter the sidebar list based on shortest distance , cheapest car park and highest car park lots available.

<Next slide: Front end & Back end>

For front end , we use CSS for styling and layout of web pages and EJS for generating dynamic HTML on the server side.

For back end, we use node js to run JavaScript code on the server side , node mailer to send emails from server-side and mongo DB as a database to store user data and user history.

<Next slide: APIs>

For the APIs that we are using , we use one map api for the routing , searching and also for the map interface. For the car park details we use HDB carpark api to get details like coordinates, name and lots availability of the car park.

Next , I will be passing my time to Damien

<Next slide: System Architecture Diagram>

Thank you Yi Feng, I will now be sharing about the architecture of the system. We use a 3-layered design where we have boundary classes that are responsible for our UI like login, register and carparkRoutes and interacting with OneMaps and HDB API. Control classes which handle the main business logic, for example SearchManager helps to query for nearby car parks at the destination and userManagerr will populate the sidebars for users to select. Lastly, we have our entity classes that act as our Data access Layer to store carpark history and user details.

<Next slide: Class Diagram>

Just like in our system design we have separated them into 3 class objects. For example our loginPage boundary class, responsible for displaying open boxes of data fields of email and password for users to fill in and login button to submit the form. Once the user clicks on the login button, the control object Login.js will use the function authenticate to process the user request by calling the database.js which will then validate at the database. And if successful login.js will redirect to the main page.

<Next slide: Sequence Diagram - Login>
Thank you Damien. 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 validates their credentials with the database. If verified, it loads the map interface along with recent parking history. If credentials are invalid, an error message is displayed. For guest users, the map interface loads directly, without access to parking history.

<Next slide: Sequence Diagram - Search>
Next, in the Search and Select 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 HDB Carpark API and its geolocation information through Onemaps API. It then calculates distances of each of the nearby car parks 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 carpark list based on distance, price and parking lots availability. Once a car park is selected, the system prepares to display the route.

<Next slide: Sequence Diagram - Route>
So, in the GetRoute 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. So, now we will be demonstrating our website.

Next I will be passing the time to Jing woon.

<Next slide: Black box testing - Login>
Thank you Jumana.
For the login use case , we use black box testing. Our input parameters such as email and password are only discrete values, since each parameter is either valid or invalid. For invalid cases , only 1 input parameter is invalid per test case as it helps in isolating which input causes

the failure. This approach can simplify debugging, as it’s clear whether the issue is due to the username or password being incorrect.

<Next Slide: Software Engineering Practices>
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.

<Next Slide: Design Patterns>
We used a MVC design pattern while coding the web app. We have the view which is our webpage 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.

I will now pass the time to Haoren who will demo our web app.

<Next Slide: Live Demo>
(One person will be acting as a user , and one other person narrating)

Begin from login page
<User Click on Continue as guest>
Guest:

1. <There is no parking history for guest account>
2. Search bar: Jurong point
3. Select any car park
4. <show in settings there is no parking history>

Allowing Location
Accept the permission to allow location access

Register:
<User click Register>
System validation for email and password

Invalid Flow:

1. Email: Show that if email is registered, cannot register same email
2. Email: sc2006project46@ (Wrong Email format)
3. Password: Car@123 (less than 8 characters)
4. Password need to follow requirement
5. Password: Carpark@123 Re-enter Password: Carpark@1234 (password not match)
6. Email: Null or Password: Null (Email or password empty)

Valid Flow:

1. Email: sc2006project46@gmail.com , Password: Carpark@123 , Re-enter Password: Carpark@123
2. <User click register>

As part of our non-functional requirement, the password is stored on MongoDB in a hashed form.

Forget Password:

Invalid Flow:

1. haoren@ (wrong format)

Valid Flow:

1. haoren@hotmail.co.uk
2. <User click Submit>

Reset Password:

Invalid Flow:

1. OTP: test123 (invalid OTP)
2. < Show OTP from Gmail Account>
3. Enter Valid OTP
4. Password : Carpark1234 (missing symbol)
5. Validation for Password is the same as register page

Valid Flow:

1. OTP: OTP from gmail , New PW: Carpark@1234, Cfm Pw: Carpark@1234
2. <User click Submit>

<System transition into Success Reset Password Page> where user can login , sign up , or use guest account

<User Login with new password>
Email:haoren@hotmail.co.uk , Password : **********

Login:

System validation for email and password

Invalid Flow:

1. Email: haoren@ (Wrong Email format)
2. Email: testtest@gmail.com (No user with that email)
3. Password: Carpark@ (wrong pw)

Valid Flow:

1. Email: haoren@hotmail.co.uk , Pw: **********
2. <User Click Sign in>

SearchNSelect Destination:

Invalid Flow:

1. Search Bar: 991854 (couldn't find postal code location)

Valid Flow:

1. <User will be searching by postal code>
2. Search Bar: 560106
3. <System will show nearby car park markers around the user's search location>
4. <User can click on different car park name to view individual car park detail>
5. Select any carpark
6. <Selected Car park will be saved into database and display in parking history under settings>

7. <User will be searching by destination name>
8. Search Bar: Tanjong pagar plaza

9. <System will show nearby car park markers around the user’s search location>
10. <User can click on different car park name to view individual car park detail>
11. <User Click on different carpark name to view individual car park detail>
12. Select any carpark
13. <Selected Car park will be saved into database and display in parking history under settings>

<Now we will be showing the function of filters>

Filters:

1. Search Bar: Tanjong Pagar Plaza
2. Filter by distance
3. <System sorts the nearby car park list by ascending order where shortest distance from user location to car park is shown first>
4. <User click on all car park to show distance is sorted>
5. Filter by price
6. <System sorts the nearby car park list by ascending order where lowest price is shown first>
7. <User click on all car park to show price rate is sorted>
8. Filter by Lots
9. <System sorts the nearby car park list by descending order where highest number of car park lots is shown first>
10. <User click on all car park to show lots availability is sorted>

Recent parking history:

1. <User delete one of the saved history>
2. <User then Logs out>
3. <User log back in>
4. Email: haoren@hotmail.co.uk , Password : *********
5. <System will only show one less saved history>

<Show settings , display in different language>

<User Logs Out>

Thats it for our presentation. Thank you for listening.

<Return to presentation slide>