Good afternoon, we are team Pineapple. Our team consists of 6 members, Jin Min, Nicholas, Wei hong, Haozheng, Jia Ying and myself Potala.
Let’s dive into the content of our presentation.

(next)

Here is what we will be covering today.

We will start with a brief overview of our project. Next, we will present our use case model providing insights into the functionality and user interactions. We will then dive into the software engineering practices, system design and traceability that we practiced to ensure the quality of our project. And finally, we will conclude with the live demo of our web application.

1. Overview

Before we dive into the details of our web application. Have you been in a situation where you were running late, desperately searching for a parking spot, only to discover that the carpark is completely full?

(next)

Our project centers around the development of a web application to make the lives of drivers more convenient and efficient. This application allows users to reach for a location and in real time displays a list of nearby carparks within a 500 meter radius. What sets us apart is that we also provide information about carpark availability, helping users plan their journeys. Users can also favourite their frequently visited carparks and access navigation directly from our app.

Our primary target audience for our application is of course, drivers who will benefit from the convenience and time-saving features our application offers.

(next)

2. Use Case Model

This is our use case diagram. Let's begin by identifying the 6 key actors. Here we have user, device gps module, google map api, system, data.gov api and ura api. And a total of 17 use cases. I'll pass my time to jia ying who will be talking about the software engineering practices.

(next)

3. Software Engineering Practices

In our software engineering project, we followed a structured SDLC approach while integrating the principles of Extreme Programming (XP) with incremental development. So we first started off with requirements elicitation, followed by the analysis and system design, before implementing our software in terms of coding.

When we implement the software, we broke down the project into smaller, functional components. Each component, represented by user stories, went through development iterations, adhering to XP's iterative approach.This combination allowed us to efficiently manage the project's complexity, adapt to evolving requirements, and ensure the timely delivery of valuable features.
(next)

During our implementation, we also kept to a few object design principles,
We ensured that our followed the single responsibility and dependency injection principle, making our code easier to maintain, test, as well as allow reusability
(next)

4. System Design

We use a layered architecture system design, separating it into 3 layers ; namely the boundary , which are all our interfaces , the control layer, containing all our handler classes and API, as well as our entity layer, with all the entity classes containing data. This makes it easy for us to add in new classes according to their functionality, and ensures the ease of maintenance of our software

(next)

As for the design patterns, we kept to a facade pattern and observer pattern. We reduced the dependencies between our classes, and only linked then when necessary. Hence when a class needs to be changed, only a few will be affected. Enhancing our reusability as well as extensibility

(next)

5. Traceability

Throughout the SDLC of our product, we want to ensure that there is good traceability. Firstly, we ensured that our requirements can be verifiable, traceable and unambiguous. We do it by atomizing our requirements. Here's an example of one of our functional requirements: the search feature. **(next)** During the requirements elicitation phase, we broke down the requirements by asking ourselves how we want the users to be able to search for a location, and what response we want from the system when we do the search. Doing so allows us to break down our requirement into simpler parts which ensures that as we go along with the design and implementation of the application, we can verify and trace that we are following it accordingly. **(next)** We can then design our application, identifying relevant classes that will be responsible for fulfilling the requirements.

(next)

With the requirements defined, we then refer to our use case model and come up with the relevant description and **(next)** sequence diagrams so that the requirements can be met. This is followed by the implementation of the application **(next)**, where our team "translates" what we have worked out in the design phase into the actual implementation via code.

(next)

Finally, after the application is implemented, we test them according to the requirements that we have defined. We have gone through black box and white box testing where we test different inputs into our system and also the control flow testing. Here in these screenshots are our test cases and results of the black box testing for our search feature. From the requirements defined earlier, we tested our application and were able to verify that all the requirements are working accordingly - that the search feature returns us results that were required such as the carpark address and availability, among other things.

As you can see, from the start to the finish, we ensured that there is good traceability in our project, and that our requirements are verifiable, traceable and unambiguous.

6. Live Demo

1. Enter app
2. Search
    1. Gibberish
    2. "Nanyang CC"
    3. "310256"
3. Sort
    1. By descending availability
    2. By ascending distance
4. Select carpark (from search results)
5. Navigate
6. Favourite carpark
7. Go to favourites page

8. Click on carpark
9. Unfavourite carpark
10. Back to favourites page