Good morning everyone. My name is _____, and my team members are _____. Our group name is no idea.

<Next slide>

The product we have designed is a mobile app called MyTripDiary, that enables its users to keep track of their daily commutes, and save time by informing them of faster or cheaper routes and modes of transport.

<Next slide>

Here is the use case diagram for our project. Let's say you've just downloaded the app, registered, and you've reached the home screen. Let's also say that you're preparing to leave the house soon to go to NTU, and would like to find out what routes you can take, along with estimated duration and price.

The first thing you'd want to do is add a trip, which you can later execute. For our app we make a distinction between a Trip, and an Executed Trip. A trip is defined by its start and end point, while an executed trip is defined by start and end points, but also when the trip took place, and the mode of transport used to execute the trip. To add a trip you'll click on "Saved Trips", which will bring you to an empty screen for now. Then you'll click on "Add trip", which will bring you to a screen where you can fill in the trip name, start point, and end point, so the start point would be your home address and the end point is NTU Lee Wee Nam Library Bus stop. Once added, you'll be brought back to the saved trips page which will be show that newly-added trip. When you click on that trip, it will bring you to a screen where you can pick your preferred mode of transport. After you've picked a mode of transport, a price estimate will be shown to you. If you're satisfied with the price, you can click "Execute trip" to save that trip into your account, and you can leave the house to go to NTU. After clicking "execute trip", you can view that executed trip in the "Trip History" page.

Let's say on your way to NTU, you wanted to take a slightly different route for whatever reason, which changed the price of your trip. In that case, from the app's home screen, you can go to "Trip History", click on the recently-executed trip, and edit the price from there.

In the future, if you'd like to execute another trip from your home to NTU Lee Wee Nam Library Bus stop, you don't have to add the trip again, you can simply go to "Saved trips" and execute the trip from there. However, if you want to execute a trip with a different start point or end point, you'll have to add that new trip before executing it.

Once you have at least one executed trip in your account, you can view stats related to your trips, such as total price spent in a given time range.

<Next slide>

So that covered some of our main app functionality, and now we'll proceed to the live demo.
<Live demo>

(This section will be written in terms of what actions the presenter will be doing, and presenter will simply verbalize whatever he's doing)

Registration

- <Begin from the login screen>
- Click "dark theme" and then "light theme"
- Click "Register here"
- Show basic validation for email and password
    - Enter invalid email (without ".com") + valid password
    - Enter valid email + the following passwords
        - Invalid password: jack123 (less than 8 chars)

        - Invalid password: jack1234 (no uppercase)
        - Invalid password: JACK1234 (no lowercase)
        - Invalid password: Jack1234 (no special char)
        - Valid password: Jack123!

Add trip + Execute trip

- <Brought to home screen>
- Click “Saved trips” (“It’s empty for now, so let’s add a trip”)
- Click add trip
- Enter trip name that is too long (cannot exceed 32 characters), enter <Some random address> for start point and NTU Lee Wee Nam Library for end point → error msg; trip name too long → enter shorter trip name and click add trip
- <Brought back to saved trips screen>
- Click on the recently added trip
- Select mode of transport → view the price → click “Execute trip”
- Go to trip history screen to verify the executed trip is saved to the database

Edit executed trip price

- Go to trip history screen, select trip
- Show basic price validation
    - Invalid price: -0.01 (Cannot be negative)
    - Invalid price: 100.01 (Must be <= 100)
    - Valid price: 0 (valid, cos maybe you got a free ride from a friend)
    - Valid price: 100.0
    - Valid price: 2.80

Edit trip name

- Go to saved trips page, edit the trip name, show basic validation (name cannot be > 32 chars)

Analytics

- Show how to show total price in given time period

<Back to slides>

One software engineering practice we applied is modular design, which is breaking the code down into smaller, reusable components or modules. By doing this, the code becomes more flexible, easier to maintain, and easier to test. For example, we had separate modules for database interactions, user interface components, and app logic. Each module would be responsible for a specific aspect of the project, which made it easier to identify and fix problems when they arose.

Another software engineering practice we used was version control. We used Git to collaborate on the code and keep track of changes over time. By using version control, we could work on different features or aspects of the app independently, without worrying about conflicts or overwriting each other's work, as there is always the commit history to fall back on. This leads to a more efficient development process and reduces the risk of introducing bugs or errors.

<Next slide>

For design patterns, we used the Model-View-Controller (MVC) pattern. This pattern separates the code into three distinct layers: the model (which represents the data), the view (which represents the user interface), and the controller (which handles user input and

communicates with the model and view). The MVC pattern was beneficial as it allowed for high cohesion and low coupling. This allowed our group members to work on the app simultaneously with little interference.

Another design pattern we used was strategy design pattern. This pattern separates the functions into different classes, thus supporting low coupling. This also supports the Open-Close Principle, where the app is open to extension in the future, but closed for modification. Single Responsibility is also upheld, where each class supports only one function, making the application very agile.

<Next slide>

<Present traceability of “Execute Trip” use case. Show:

- the class diagrams,
- sequence diagrams,
- good designs applied, how we implemented them
- and testing performed >