Demo Script

Flow:
Intro
App overview, problem and purpose
Expected users
Main features overview (Use case models, Sequence Diagram, System Architecture)
Design Pattern
Design Principles
Live Demo
Testing
Good SE practices

Presentation Section: Intro, App overview, problem and purpose , Expected users
Presenter: Eldrick

Good Morning professor and classmates. My name is Eldrick. This is Amabel, Aloysius, Kai Seong, Harish and Zhi Heng,
and together we are one start.

Next slide

Today we will be presenting to you our app, justJio, a multi purpose party app that can satisfy all your needs and rid you of all your hassles.

Next slide

So just imagine, your friend Steve's birthday is coming up. You're excited to start planning for his birthday party! You start listing all the things a party should have.

Next slide

Food and drinks, games, the guest list? It would be pretty hard to coordinate with everyone going. And what happens if everyone brought food to the party to share. The calculations to split the bill of the after party would be such a headache.

Next slide

But Fret not, for justJio can help you do all that, reducing the time needed for you to prepare for the party and making your party experience so much better.

Next slide

We predict that most users of our app will be teenagers, young adults and normal working adults. This party app can be used for any occasion, and as such this app is suited for all.

So let me hand my time over to Amabel, who would be diving into the main features of our app.

Presentation Section: Main features overview (Use case models, Sequence Diagram, System Architecture)
Presenter: Amabel

Next slide

Now, I'll be sharing more about the 5 main features of our app. Firstly, we have create room. It is a feature that allows the host to add all participating members of an event to a shared room, where the event details are easily accessible. Once a room is created, the host has the option of adding more members subsequently if needed, and also make use of the split bill function which will be further elaborated on later.

Next, we also have party snacks. For this feature, we made use of the Google API. Users can input their current location to search for nearby shops like supermarkets that sell party snacks.

Similarly, we have party games. If you ever run out of ideas for games to play at a party, this feature displays a list of classic games with rules on how they are to be played.

Furthermore, we also have a spin the wheel function that would definitely come in handy when playing games in a big group at a party.

As mentioned earlier, we have a split bill feature. This is especially useful for parties with a large group, where payments to many different people tend to get messy. This will be further elaborated on later.

Lastly, we have a transaction history feature. This is a simple function that allows users to view their past transactions, which includes who they have paid and who they have received money from, as well as the respective amounts.

Next slide

This is an overview of our app with all the use cases. So the general flow is that every user would have to first register when using our app for the first time. After registering for an account, users will then be able to log in using only their usernames and passwords if they wish to use the app subsequently. And only when a user has logged in, can he access all of our other app features, including pay bill, party games, party snacks, transaction history and the many room features.

Next slide

I'll now delve deeper into 2 of the features mentioned earlier and explain more details using our sequence diagram. First, for our create room feature, when a user clicks on the create room button on the home page, he will be directed to a page where he will be prompted to input details of the event. This includes the name, date, time, location of the event, as well as the users to be invited. If any of the inputs are invalid, our app will show an error message and prompt them to edit their inputs. Once a room is created successfully, our app will display a message to inform the host that the room has been created successfully and our system will send an invitation to the users who are invited.

Next slide

Next, the split bill feature. When a user clicks on the split bill button in the room, they will first be prompted to select the list of payers. This is useful in situations where not everyone in the group is required to pay for a particular item. The user can then key in the name of the item they bought and its cost. Our app will then automatically do the calculation and split the bill accordingly, before prompting the other users the amount to pay to each person.

Next slide

This is our class diagram. As mentioned earlier, when a user first clicks onto the app, a splash screen will be displayed before going to the login page. After the user logs in, he will be brought to the home page, which is where he gets to access the other features. All of our features make use of 3 control classes in total, which are user manager, room manager and transaction manager.

Next slide

This is the system architecture for our application. For the boundary classes which are mainly the UIs, we coded it using React Native. For the control classes, we used the Go fiber framework and for the entities, we stored them in a mySQL cloud database.

Presentation Section: Design Patterns + Principles
Presenter: Harish + Kai Seong

Next slide

Ok, thank you Amabel, Ok so let's look at some of the design patterns our team used for the core of the app. So we used a model view controller system. So the user would view the app through the screen classes and interact using the reducer classes. This allows for high cohesion and low coupling meaning that multiple members in our team could work on the app simultaneously without interference and the logic of the app was developed relatively fast.

Next slide

For our UI we used a facade pattern
as u can see the home page allows the user to access all the core functionalities mentioned by amabel. it allows for loose coupling and the need to fetch data only when necessary.

Next slide

For our controller classes , we used a strategy pattern allowing for encapsulation and the need to select an appropriate reducer only at run time. This also means we can scale it for future developments

Next slide

Thank you Harish. So now I'll be going through the design principles we applied into the architecture of our app. Firstly, we obeyed the principle of least knowledge by storing client and server code separately. The sole way for the both of them to communicate and transfer data was through HTTP REST APIs. This way, it allowed us to abstract away the implementation details of both sides and it would be easier to add new features to both of them as they are in an isolated environment.

Next slide

Next up, we also followed the DRY principle which stands for Dont Repeat Yourself. We identified and refactored commonly used blocks of code into components to increase the modularity, maintainability and readability of our codebase.

Next slide

Finally, our app fulfills the first of SOLID principles - SIngle Responsibility Principle. We separated handlers/controllers in the backend according to specific functionalities. This helped us to debug issues easily as we can test each of them individually.

I'll now pass my time to Zhi Heng for the live demo of the app.

Presentation Section: Live Demo
Presenter: Zhi Heng

Now I will be doing a Live Demo of the app. Assuming I am using the application for the first time, if I do try to login now, as you can see, it will throw an error about user does not exist in the system.

Sign Up

So now, I will proceed to move on to the sign up page. So in this page,

Fill Up Details (Register)
Username: John
Phone Number: 99889988
Email: 123abc
Password: John
Confirm Password: John

If I do input these details inside, you can see here that it throws various errors as we want to ensure that our user does provide valid information and secure passwords to secure their information with us and not get hacked.

So now, I will proceed to change it to a more suitable choice.

If you see here, if i put 123@abc, it still does not work, until i put like john@gmail.com. Then the error will disappear.

Next for passwords, since it is too short, I will need to lengthen it. Even after increasing it, you can see that it still needs us to input the upper and lower letters, numbers and symbols.

So now I will modify the password to be John12345! And same for the confirm password section.

If we do not type a correct password, it will show that passwords do not match.

After rectifying all these errors, we should be able to register successfully and now login to the page.

Login

If we do enter a wrong password by accident, it will tell us as well.

After logging into the page, we are now presented to our Home Page.

Create Room
So now as a user who just signed up, he will want to let's say create a celebration for the end of exams at the end of this month.

He will click on this icon to create the room.

Fill in the following details (Create Room)
Name of Event: End of Exams
Date: 31/11/2022
Time: 08:00pm
Venue: NTU Hall 10
Invitees: ZhiHeng(myself), Kaiseong, Eldrick, joshua

But however mark may not be registered, so when trying to create the room, it will show that mark does not exist.

So if we remove, we will be able to create the room successfully.

Invite User
To follow up, if we want to invite the person after adding the person into the room, we can go back to the same room, by clicking on it, and

Click invite with the plus icon. And let's say want to invite another person who have registered maybe aloysius. Then it will successfully send the invitation.

(TRANSITION TO LOG IN TO ALOYSIUS ACCOUNT)

Join Room
In order for one to accept the invitation, he will need to look under the room invitations.

Party Games
After accepting, he or she may be the other person in charge of hosting the party and assist him or her by looking at the party games that can be played at the game and plan before hand.

Party Snacks
If let's say there are not enough food or drinks at the party, the person can go click on the hamburger icon which shows the nearest party snacks it has to their current location. In this case we are at NTU, so it will show Prime Supermarket. If I were to type another location so that the person can buy beforehand, he can type on it and search.

In this case, he stays near sentosa. Can just type sentosa and type in the location.

If want to change back his current location, he can just tap on this icon on the right to sync back.

After buying the groceries and drinks, he will want to split the bill evenly among all the members who went for the party.

Split Bill
He will go to the party list and select or de-select members to split the bill.

key in the following bill details
Bill Name: Drinks
Amount: tp

If he accidentally types wrong, it will throw an error. Because it is not a number. This amount will be computed to divide the amount and split to all the users evenly.

Amount: 20

After that, you can see it displays split bill successfully.

You can see from the home screen that it shows the amount each person owes to that person.

So now I will login to my personal account to pay him accordingly.

(TRANSITION TO LOG IN TO NEW ACCOUNT CREATED)

Pay Bill
Using this tick icon, I will then press to pay the person accordingly.

Transaction History
And if you can see here it shows red which means it deducts money and paid the person successfully.

And now I will pass the time back to Aloysius to talk more about how we ensure our application works properly.

Presentation Section: Testing
Presenter: Aloysius

To ensure proper functionality of our app, we have employed several testing approaches. One of the approaches adopted was Control Flow testing which is a form of White Box testing. Here is the Control Flow Graph of the user registration page of our app. The user clicks on the sign up button from the login page and input the respective credentials. If any one of the fields are left blank or is invalid, the user will be prompted to fill in or re-enter the field in question. Upon meeting all requirements, the account will be created successfully and the user will be redirected back to the login page.

Next slide

This is the positive test scenario of the registration page where the fields were non-empty and met all the criteria. The test result was a pass as the expected output matched the actual output.

Next slide

This is the negative test scenario for the same page where we chose one input to be invalid, and kept everything else as valid. The expected output would be that the user will be prompted to re-enter the invalid field, which matched the expected output making the test result a pass.

Presentation Section: Good SE Practices
Presenter: Eldrick

Next slide

Thank you Aloysius. To sum up what we have done for our project, I will now be going through some good software engineering practices that we applied while working on our app. Firstly, we frequently refactored our code, refining them to improve code cleanliness, readability and reusability.

Next is API documentation. We noted down detailed guidelines on API endpoints to allow easy integration between front end and back end components.

Thirdly we used GitHub as a platform to track contributions and also store our code changes so that we can fall back on working code if the app were to break.

Fourthly, we worked in pairs to code the front end of the project in order to develop common ownership of code, improving refactoring of code and sharing knowledge to debug and fix broken code.

Next slide

Lastly, we planned ahead to deliver a set amount of work by a certain time. As you can see, we noted down what we needed to do by the end of every 2 weeks. First finishing the functional and non-functional requirements and eventually moving on to finishing our diagrams and lastly finalizing our app and testing it out.

Next slide

And that comes to the end of our presentation. Thank you so much for listening and hope you like our product!