Script
Jolynn: 1 - 10
Melv and adrian: 11 - 15 (test cases)
Pearlina: 17 - 19
Ryan:
Win: 22 - 28 + conclusion

1	Hi everyone we are group Orange1
2	This is the flow of our presentation.
3	Firstly, our app introduction.
4	Have you ever been out and about minding your own business when you suddenly realise your phone is almost out of battery? And you don't know where to charge your phone? Well introducing to you…
5	Wattway! Which is an app that helps you locate charging locations closest to you in instantaneous real time!
6	This is our use case diagram. When we were planning, these were the main functions of our app :..... Our target audience are people who have a phone and forget to bring their portable chargers.
7	For the functional requirement, the app will display nearby charging locations and its relevant details in a 5km radius. It will also search for locations and display charging locations in the searched area.
8	To allow our database of existing charging locations to keep growing, the app also allows you to add new charging locations, report incorrectly described or damaged charging locations and vote on pending charging locations and the reports for verification.
9	To incentivise users to help expand our database, users can earn points through voting and contributing to the shared database and use the points to redeem various rewards.
10	Now we will move on to the demonstration……..
11	To begin, our user will first attempt to register themselves with WattWay. Under the signup page, they will fill up the required information such as username and password. <<Sign up>> For a base test case, we will demonstrate that a duplicate username will not be accepted. <<Attempt to sign up with "adrian">> As you can see from the database, the username "adrian" has already been taken, and hence cannot be used again for a sign up.

	We can verify this with the database as shown. <<Show database>> Next, we will reattempt a sign up with “winfred”. As you can see this will go through and he will be added to the database as a new user. <<Show success + refresh database>> Moving on, we will demonstrate a login function with “adrian” as previously shown in the database. We will first show a failed login due to a wrong password. As you can see from the database, his password does not match what I have typed in. <<Login fail with wrong password>> Next, we will finally login successfully with the correct password. <<Login success>>
12	Upon logging in, we are presented with maps queried from Google maps using our own API key. From here, we can click this button to bring us to our current location, which is in SCSE. <<Click on button>> A user can then look for the nearest charging port location based on these markers. Green markers indicate a confirmed location, where at least 20 votes have been casted in favor of that location, while orange markers indicate pending locations, where it is still waiting for more votes for it to become confirmed. Additionally, a user may search for a nearby location to view charging ports in that vicinity to aid them in their day’s planning. For our example, our user wishes to go to Orchard. <<Go to search page and search for orchard>> The redirected map will display nearby charging locations for the user to look at again.
13	Finally, let’s first move back to our current location. **<<Back button to go back>>** We can first view a green marker which indicates a confirmed location. <Click on green marker>> From here, we can see the details of the location, such as a description, the number of ports available, what kind of ports and cable heads are available, whether there is a locker provided and the status of the location. Next, we can also view a pending location notated by an orange marker. Similarly, we can view the details of the charging ports. Additionally, we are able to vote on these locations with upvotes and downvotes. <<Play with voting buttons>> Do take note our voting system changes a pending location to a confirmed location (So orange marker to green marker) **automatically** upon receiving more than 20 upvotes. Conversely, if it receives more than 10 downvotes, it will be removed from the system.
14	Next, we will demonstrate the rewards page of our app.(Click on ALL to show the

	deals) The rewards page is where users can make use of their points gained from activities in our app, such as sending a new report for a new charging location or to report an existing port as damaged or incorrectly described. On the rewards page, the user can first check the tier that they are in by clicking on the rewards drop down button. It will show the user's current tier based on their lifetime points gained from the app, as well as the current points that the user currently has. Tiers are split into Gold with more than 20000 lifetime points, Silver with more than 10000 lifetime points, Bronze with more than 5000 lifetime points and no current tier for users with less than 5000 lifetime points. <<Display User points>> Based on the current points that the user has, the user is able to use the points to redeem the deals available on our application. The user is able to filter the rewards available in our application by clicking the filter dropdown button. "All" will show all available deals in the app, while other filters such as travel will only show deals related to the filter, as you can see on the phone. << Display all deals >> << Display deals by filter>> Next, the user is able to select the deal the user wishes to redeem. Let's take an example where the user wishes to redeem this deal for 100 Grab points, which cost 1000 points. The user can simply click on the "redeem" button, and the user will be able to successfully redeem the deal, given that the user has sufficient points currently for the deal. After successful redemption of the deal, the deal will appear as redeemed on the screen and the user is unable to redeem the same deal again even if the user has sufficient points (Click to show to redeem the same deal again, should show error message). Successful redemption of the deal will also update the user's current points available in the app (Click to show current updated points - Should minus 1000 points).
15	Moving on, we have the activities page where the user is able to either submit a report to add a new location or to submit a report on an existing charging location to report damaged or incorrectly described ports. Firstly, discover a new charging location. The user can click on the dropdown and a form will appear for the user to fill up. The information required to fill in would include where the new charging location is, the number of ports available as the capacity of the location, the different port types available, the types of wires available, if any, as well as whether the charging ports are lockable or not. Once the user fills in all of the information, the user can click on the submit button to submit the report. <Fill up form and click submit> Upon successful submission of the report, the user will be rewarded with 50 points that will be added to the user's account immediately (Close the dropdown and go to the rewards page to show change in points). The submitted report will also be added to our database immediately (Show the changes in charging locations database). Following that, the user is also able to report a port by clicking the dropdown for it. A similar form will appear which requires the user to fill up. Starting off by stating whether it will be a damaged or incorrect report. The user will then need to indicate the location ID of the charging location, which can be found by clicking on

	the marker of the charging location on the map, as well as the new capacity of the location. The user will then select the currently available port types as well as wire type if any and indicate whether it is lockable or not. Lastly, the user should provide a short description for the report, for example in a damaged report, the user can indicate " There are 2 ports damaged leaving only 8 available." After filling up the form, the user can click the submit button to submit the report. (CLick Submit). Successful submission will likewise give the user 50 points and the report will be added to the database (Show the changes in points in rewards and changes in reports database). The damage report can also be seen under the location details of the location marker on the map. Under the database, we can see the report that has been just submitted, and the details are as follows. As you can see, the details that we put into the form are translated accurately into the backend. Under the type of cables available column, we stored the data in a 6 digit number. First number is whether it is lockable or not, 0 being lockable, 1 being cannot be locked. Next digit represents whether there is a USB port, third digit represents whether there is a USB-C port, fourth digit represents whether there is a USB-C cable, fifth digit represents whether there is a lightning cable and the final digit represents whether there is a micro-USB wire. It is important to note that each user only has 3 activity quotas per day, where each report submitted will deduct one quota from the user, to prevent multiple submissions just to gain points. Lastly, we have our profile page, which will display some miscellaneous information such as the general settings for users to change their username and passwords, the history activities on the app such as deals redeemed, our app privacy policy, some information about the app, some common FAQs for users whom might not be very familiar with the app yet, and lastly a logout button for the user to log out once they are done with the app. And that is the end of our application live demo (Log Out and proceed to the next part of the presentation….)

16	Moving on to software concepts and designs, we made use of …
17	SOLID Principles. We identified 2 principles, Single Responsibility principle and the open closed principle. We applied the Single Responsibility Principle to our frontend where each page has a certain responsibility. For example, the login page is purely used for logging in, the sign up page is for signing up. We also applied SRP in our backend. For instance, locations class only has functionalities related to charging locations, Action class only deals with reports and the User class only handles user details and functions. On the other hand, we applied the Open Closed principle in our frontend. When a new function needs to be added, a new functionality class can be added without modifying the existing classes.
18	Moving on, to our architectural pattern.

	Initially we used the 3-Layered Architecture pattern to separate our app into distinct layers, each handling a specific set of responsibilities. This separation helps in organising, managing, and maintaining the app. We first identified the entity, boundary and control classes. Then we indicated the interfaces which interact with the model indirectly to display and modify data, responding to user input and present information. For the Application Managers, they manage user input, control the application flow, and act as an intermediary between the Interface and Database. Lastly the database manages the data, rules, and logic of the application. It responds to requests for information, processes data, and defines how data is manipulated and stored. However after beginning the coding process, we realised that our architectural pattern is not as feasible, thus we came up with a…
19	more detailed version which includes the fastapi class that allows the frontend to communicate with the backend. It mainly contains the GETTER and POST functions which allows the frontend to retrieve info from and send info to backend. We also included the database helper functions which allows our backend classes to interact with the database. By breaking down our system architecture into smaller and more focused modules, it allows us to abstract away the implementation details of different modules, making our code more loosely coupled and higher cohesion, thus allowing us to code and test individually, which makes the progress of our code faster.
20	Thirdly, We also applied the principle of least knowledge which states that a module or component should only interact with a limited set of other modules, and only in a well-defined way. By limiting access or information on a need-to-know basis, each unit should only have limited knowledge about other units. This is seen in the backend programming where our manager classes are initiated only with the information they require for their respective functions, instead of the whole range of data initiated with our main user class. This reduces the risk of unauthorised access, data breaches, and other security vulnerabilities, minimising the potential harm that can be caused by misuse or exploitation
21	Lastly, we applied the DRY principle to reduce repetition of information and duplicates by avoiding repetitive code, logic, or data. This also helped with the maintenance and consistency as we reduce the effort needed to make changes or fixes. When designing our database helper functions, we realised we were constantly reusing the code to parse through our database with minimal changes. Hence, we abstracted out the logic and created a function solely for iterating through our database, and extracting data based on filters passed through the function. This not only reduces the amount of repeated code in our program, but also allows us to easily implement more functions in the future that requires similar functionality. ~~We also made use of Abstraction and Encapsulation to isolate common logic.~~

22	Moving on to traceability of the getting the location of our app
23	We first present to you the sequence diagram of getting nearby charging locations and the intended output on the app
24	For frontier nd, this is our code to get location. We used a google api to present the google map and created a GoogleMapController object to allow us to call for the longitude and latitude of locations to navigate the map. We set the initial camera position as NTU and there is a button that allows us to get the nearby charging locations from the backend database. How this is done is…

25	The frontend will first interact with the backend through FAST api calls. It is done through parsing the API URL with its details with the query parameters
26	Next, The FastAPI module receives the requests from frontend and passes it to the backend using the different backend functions available. It is also able to return messages back to frontend for frontend to use.
27	The backend code processes the data passed and it will do its logic checks and operations, as well as pass it to the database for it to be processed and stored. The bottom image shows how the database and details of each charging location are stored.
28	Finally, all the info will be returned back to the frontend through the api response and frontend uses this information to display the nearby charging locations on its interface.
29	From this project, we would like to conclude that…
30	Through wattway, we learnt that software engineering all boils down to these 3 concepts. Usability, in ensuring ease of usage for our users Performance, to ensure that the app works efficiently and effectively during runtime Expandability, ensuring that the app is loosely coupled for ease of adding new functions in the future.
31	With that, we thank you for listening to our presentation.