COMBINED SCRIPT

Slide 3: The Problem

As Singaporeans start to realise the importance of being environmentally friendly, more people are now interested in recycling their used items. However, while the National Environmental Agency website has a list of recycling bins in Singapore, they are inconvenient and difficult to locate, discouraging people from recycling. In fact, according to NEA, the domestic recycling rate decreased from 22% in 2018 to 17% in 2019. Hence, to promote recycling and make it more convenient, we created the SG!Recycle application, which will make recycling simpler than ever.

Slide 4: Key Features of Our App

The SG!Recycle Application combines Recycling and Convenience, which were once disparate. By displaying the top 10 Recycling Bins at the user's fingertips, it greatly enhances the ease of recycling used items.

So what sets us apart from other applications already available in the market? We have an additional feature only made possible with modern technology: a Recyclable Classifier. Leveraging Machine Learning Algorithms to classify objects, SG!Recycle helps the user to determine if his object to be disposed is recyclable. This further reduces the Time Complexity — I mean time taken that the user would have needed to check if his item was recyclable.

These two main functions make the entire experience of recycling significantly more convenient, from start to finish. This application can easily be downloaded on the user's mobile device, to be used anytime, anywhere.

Slide 5: Our Development Process

From the problem statement, we thought from a user perspective to figure out what he or she would want from an optimal recycling application. Afterwards, we atomised these user requirements to get a fitting description of the system and its relevant use cases.

Slide 6/7: Functional Requirements

Firstly, we derived our functional requirements. Our functional requirements are mainly split into two portions -- that of the location finder and that of the item scanner.

For the location finder, the application must essentially find the 10 closest recycling bins from the location detected or inputted.

For the item scanner, the application must firstly inform the user on the type of items recyclable then allow the user to scan and determine if his item is suitable for recycling.

Slide 8: Non-Functional Requirements
Next, we derived our non-functional requirements. We included requirements mainly based on system reliability and minimum load.

Slide 9: Use Case Diagram
From our functional requirements, we identified the actors and various sub functionalities to derive 6 use cases. In particular, because our system made use of multiple external APIs, we had various actors apart from our system user.

Slide 10/11: Class Diagram
From the requirements and use case model that we generated, we analysed the requirements and derived multiple classes to produce our system model. We derived both the conceptual model which describes the system structure and dynamic model which describes the system behaviour. The challenge here that we faced was how to stereotype the different classes. For that, we carefully split them with user-interactive Boundary Classes, logic realizing Control Classes, and information wielding Entity Classes.

Slide 12/13/14: Sequence Diagram
Following this, we derived a sequence diagram to determine how we will fulfill the processes defined in the use cases and which objects to perform a certain process.

For the location finder, we identified 2 boundary classes, 2 control classes, and 1 entity class.
For the itemScanner, we identified the 2 boundary classes and one control class.

Slide 15: System Architecture
The system architecture made us reflect carefully on how our application was supposed to be structured. Initially, we went with a 3-tier architecture with Presentation, App Logic and Persistent Data layers.
However, we felt that was too vague since our application involved leveraging on external APIs such as Google Maps and the National Environmental Agency.

Hence, we extended our architecture to 4 tiers to include an External API Interface tier, where the classes in there communicated with the APIs used.

Slide 16: Design Patterns
Once the system architecture was done, we started deliberating on suitable design patterns to include in the development of our application.

We made use of the single responsibility principle to modularise our code

In particular, we made sure that every class was only responsible for one function. For instance, the Distance Calculator class was only responsible for calculating the distance to the various bins, and the Recyclable Detector class was only responsible for classifying items into recyclable and non-recyclable.
This reduces the coupling of the classes, and allows us to make changes to any functionalities easily. EMPHASIZE This allows us to implement additional upgrades that we will mention later.

Slide 17: Design Patterns
The Principle of Least Knowledge was also applied in our application, where we ensured that Classes only communicated with their immediate "friends", but not "friends of friends". This was to ensure that unit testing and debugging was made much simpler, narrowing the scope of search for bugs greatly.

Slide 18: Distance calculator in depth
Now, zooming into the implementation of one particular use case…

For the use case RETURN NEAREST RECYCLING BINS, we implemented it using the DISTANCE CALCULATOR CLASS.

For the implementation of this functionality, firstly, the Distance Calculator functions by taking in the user's current location in Latitude and Longitude format, regardless of auto detection or manual user input.

Using this, the Calculator compares this user location with all available recycling bin locations stored in an Entity Class, BinLocations, storing each recycling bin coordinate along with its distance from the user within a HashMap.

Next, the final HashMap of all recycling bin locations along with their distances from users, is sorted based on the closest distance to the user.

Finally, the first 10 keys in the HashMap are passed into the external Google API to drop pins on the User Interface Map.

Slide 19: Control Flow Testing
The final priority before coding out the application was to determine the test cases that would be used to evaluate the application's efficacy, especially for the case of locating the nearest bins to the user.

Firstly for the distance calculator method we described above, control flow testing was used on calculating the distance of bins to the user. This was because the execution paths of these

functionalities were very clear and expected outputs were the most obvious, allowing potential bugs to be found easily through Control Flow Testing.

Slide 20: Boundary Values and Discrete Values Testing
Next, for the manual detection of user locations, Boundary Value and Equivalence Class testing came in especially handy. By comparing against the requirements specification, we could detect software bugs or find out if the software was not ready to receive certain input.
Hence, the team had to figure how to minimize the number of valid and ???invalid postal code inputs from the user, before passing the input to the Google Maps API to search for the location. This required research in the Postal Code system of Singapore, as well as some thought into possible invalid values.

Slide 21: Future Extensions
Of course, there's no such thing as a perfect application. If we had more time, here are some functions that we hope to implement. Firstly, for the Item Classifier, we can include more materials. For instance, we hope to include locations of E-Waste Recycling Bin, as well as be able to detect and classify E-Waste. We also hope to implement a Google Maps function to guide the user to the recycling bin he wishes to go.

Slide 22: Transition
This is Owen, who is new to recycling. He wishes to start recycling, but is unsure of where to find a recycling bin, or what he can recycle.

Fortunately, he has downloaded the SG!Recycle app, that will aid him in his quest to become environmentally friendly.

Slide 23: Demo Home Page
Owen opens up the app, and on the home page, he can scroll through these elements. Each elements represents a type of recyclables, and clicking on it provides him with additional information on that type of recyclable.
Now that he is roughly aware of the items recyclable, he wonders about where he can find the nearest recycling bin to deposit his items.

Slide 24: Demo Locations Permissions
Unsure, he simply clicks on the "FIND LOCATION" tab. SG!Recycle will first request for location permissions from the user. Owen clicks on "Allow" so that the application can obtain his current location automatically. In the case that the user does not wish to reveal his current location, the user can choose to use manual detection of location instead, which we will demonstrate later.

Slide 25: Demo Location Finder
Once Owen's current location is detected, the system sends the coordinates to the DISTANCE CALCULATOR ALGORITHM, which we mentioned previously. The distance calculator immediately checks the user location against the recycling bins in singapore, and returns the coordinates of the top 10 bins with the nearest location.

On the map, we can see the map zoomed-in onto Owen's current location, with red pins indicating the 10 bins closest to him. If he wishes to, he can zoom in the map further.

Slide 26: Demo Location Finder
HOWEVER, what happens if Owen's location cannot be detected, or he wants to search in advance for another location? Not to worry, he can simply manually input the postal code by clicking the postal tab on the toolbar. Once he sends a valid postal code in, our application will similarly send his coordinates to the DISTANCE CALCULATOR, which will once again return the nearest bins.

Slide 27: Demo Location Finder Error
In the event that the system is unable to detect Owen's current location automatically or Owen keyed in an invalid postal code, the corresponding error messages will be displayed on the respective user interface, as shown here.

Slide 28: Demo Camera Permissions
Now, after Owen knows where to go to recycle his items, what if he is still unsure as to whether his item is recyclable? To Verify, he can simply click on the "ITEM VERIFIER". Once again, the application will request the user's permission to allow camera access. Unfortunately, if this permission is not allowed, the user will be unable to use this particular function.

Slide 29: Demo Recyclable Detector
When permission is accepted, Owen sees a camera interface where he can scan the items he wants to recycle. He places the items in front of the camera, and the ITEM DETECTOR then detects the material type of the object and sends it to the ITEM CLASSIFIER algorithm. If the object's material is suitable for recycling, the classifier will classify it as recyclable and then inform the user that his item is good for recycling.

Slide 30
"THIS APP IS SO COOL!! I think I can really make good use of this app to recycle more in the future!! I no longer have to spend ages walking around trying to find recycling bins. Plus, it's so convenient to be able to check the recyclability of the items, I don't have to worry about contaminating recycling bins with non-recyclables anymore. I should introduce this app to my friends and we can all use it together!!"

Total time: 12 Min speaking time