Script:

[Introduction] Hello, we are Team Runtime Error. Today, we'll be demo-ing our project on a website application. Our project aims to promote recycling by aiding Singaporean in their recycling efforts. We'll start off with our use uses.

[Use Case Diagram Brief Explanation] This is our use case diagram. Our main use cases includes register for a new account, login, navigate to nearest bin, favourite a location, feedbacks on locations and displaying articles.

[Transition to demo]: Now we'll demonstrate our application.

[Demo: Register and Login] Next I will show how to register an account on the website. You can click the login button in the slide bar and then click the sign up to come to our register page. Here you can type in your name, password, and your email. Our website will check whether your username is unique, the confirm password is identical to your password and whether your email is valid. Here, if the two passwords are not identical and the email is invalid, the error message will show up. If everything is valid, then the website will direct to login page and send a confirmation email to the email. Now we can first login. If your username or password is incorrect then you can not login. We login with the account we created previously and then head to the account page. We can see the confirmation is false. We can go to our email server. This is a fake smtp server to use to mock the email confirmation procedure. The token url will be generated and sent to the email. We copy-paste the url and then go to the website. The flash message will show up if the token is valid. Then we refresh the account page we can see the confirmation is true.

[Demo: Article]
To read up on articles, users can select the articles in the navigation tab. [Direct to articles page] They will be brought to a page where a list of articles will be displayed. Depending on the user's preference, he can select any article to read. For example, the user chooses "What to do with broken lightbulb". [Direct to article 3 page] In this article page, he will be able to view the caption, image, author, date of release and its content. Another feature of the articles page is that users will be able to navigate to other organisational sites for official information by clicking on the icon of the the organisation. In this case, the user can navigate to the NEA websites for any updates. [Direct to NEA website opened in new tab]

[Demo: Navigating to nearest bin]
Now, users need to be able to find a bin of suitable recycling type near them. To do so, users will select "Locate Your Nearest Bin" in the landing page or "Find a bin" in the navigation tab. [Direct to Find a bin page] Based on what type of recyclables the users have, they would select the suitable type in the dropdown on the left, or if they are unsure, they could upload an image of their item on the right which we will

demonstrate later. Then, the user can enter where they want the nearest bin to or check the box on the right for the system to get their current location. Next, users will input their preferred mode of transport. [Click Lighting Waste, enter Raffles City and click transit] [List of nearest bins appears] Now we will demonstrate the uploading of image. User will click on the image button, upload a file and a message of the trash type will be displayed. So choose the Lighting waste option. Now let us try checking the “get current location” box and search. A flash message appears stating our current location and a list of bins nearest to this will appear. Next, user can click on whichever location they want. [Direct to bin details page] So a static map with a marker of the location is shown, with more details of the location. User can then choose to click on the google maps button for specific instructions on how to get there from their chosen location.
Alternatively, user can select “Add to favourites” to bookmark this location.

[Demo: Favourites]

What happens when you forget the postal code or the address of the bin you always go to for recycling? Fret not! This is when the favourites feature comes in! After searching for the specific bins using the “Find a bin” feature, users can simply favourite this location using the add to favourites button [Show Siglap Road for lighting waste]. A successful message indicating favourites is added successfully is displayed. Let us as another location to favourites [Add second hand goods to favourites].

Now we as a user, we might like to view the list of our favourited location. We navigate to the favourites page and click on “Lighting Waste” as category. We can see that Siglap Road, the location with the category of “Lighting Waste” we added to favourites previously is displayed, together with the address. If we navigate to the “Second Hand” category, we would see the other location with the category “Second Hand” displayed.

Sometimes a user may have forgotten whether he or she has already favourited a location. Our application prevents user from adding favourites twice. If a user adds a already favourited location, a message will be flashed to indicate that the location is already added. [Show Add to favourites for siglap road again]

The favourites list is specific to the user. Each user will have a unique favourite’s list based on which location they decide to add as favourites. Let’s say we create another account and not anything to favourites for this user. [Show creation of new account]. For this user, the favourites list will be empty and will not contain the favourited locations of the previous user [Show empty list of favourites]

[Demo: Feedback]

[before presentation, put in feedback with default pic] Moving on, we will be focusing on the use case of providing feedback on a bin location. This is a login-required function, which means should the user not be logged in, he will be redirected to the login page. Upon pressing the feedback icon, we will be redirected to the feedback page, where we are presented with two choices - to display a list of feedback on a selected location, or to create a feedback of our own. Let’s start with the latter. Here, we should fill up the form with our desired waste category, such as Lighting Waste. We then proceed to select from the list of locations that has been filtered to accommodate our previous request. We cannot proceed without adding our rating and reviews as an error will be displayed. We then give our rating and review, before choosing a picture to upload. Just to note, if the user doesn’t upload any picture, a default picture will be displayed instead. Thereafter, we shall submit our feedback. If successful, we will be redirected to the main page with a “Feedback created successfully” message flashed.

Next, if a user wants to see the feedback given to a specific location, we should select its corresponding waste category and location, before searching and displaying it. As shown, we can now see the feedback given, with their respective ratings and text reviews.

[Good SE practices]

For our application, we incorporated the MVC architecture, where we create models, view and controllers. We shall look at the example of when a user tries to log in. From the view, which is the login interface, when the user clicks the login button, the controller as shown in the picture will capture and process the user input and verify with the data Model. Upon validation, the page will be redirected to the main page.

The Open-Closed Principle states that a module should be open for extension but closed for modification. Our code exemplifies this for our templates, which are the HTML classes. We have the base class, which is the layout html class, and articles, feedback, favorites and login classes extend it. These subclasses serve different purposes, hence they should appear differently from each other on the user interface. This is where the “extension” part of OCP comes in: Each of the subclasses overwrites the body and style of the base class layout html, which remains untouched and hence, closed for modification. One can easily add more subclasses in the future (e.g. discussion forum) while keeping the overall basic format the same by extending the layout class and adding on to its own features, rather than modifying layout html.

The standard layout html is preserved as all classes have the green header and the navigation bar, yet each subclass has different features.

Another principle we have applied in Single Responsibility Principle which states that there should not be more than one reason for a class to change. Take the case of Articles, where it is only responsible for operations made on adding and deleting articles. On the other hand, article is is only responsible for changing the attributes of the actual article such as adding a new attribute like article viewership. Hence, each class has a different single responsibility.