(Script for Presentation)
(Can include Transcript for Demo Video)

You may consider demonstrating traceability from within modelling tools and IDEs.

[Intro]
Good afternoon professors, or to anyone watching this demo.
We are Group 53 from Lab Group TDDA.
In Singapore's fast-paced and competitive society, education is viewed as the primary means to climb the social ladder, making it a top priority.
Given the high stakes of their choices, students often feel intense pressure and uncertainty when deciding on their educational paths. This is especially true for students taking their N and O levels as they prepare for polytechnic, where selecting a course can shape their future.
Our solution, PolyQuest, addresses this challenge by allowing students to enter their academic results and interests and complete a set of personality questions.
Based on this information, PolyQuest just like our name recommends courses that empower students to make more informed decisions for them to embark on their very own poly quest.

Our technologies are built on React for the frontend and Java Spring Boot for our backend.
The applications are built independently to enable continuous development and maintain low coupling between systems. Integration is facilitated through REST API documentation provided via Swagger UI from the Java backend.

[Dallas - Auth]
For my part, my feature was the authentication of the application, so this involves the sign up, sign in, forget password, google login and wrapping the entire application so that there are no unauthorised users. We used Firebase as the library for handling authentication instead of building our own authentication system.

For all of the forms, there is basic validation, such as if the password is too short, or if the fields are missing any required fields.

Shows all the 3 pages and the validation for each page
Shows Google login capabilities

[Yan Yee - Dashboard]
Upon logging in, you will be greeted with a personalised welcome message, You will see your username at the top right. If it's your first time using PolyQuest, You will be encouraged to take a quick quiz to help identify which courses best suit your skills and preferences. The test takes about 5 minutes, with 25 questions. Below is the list of courses that we got from our database which uses the data.gov apis data that is available. Each course is displayed in a card layout, showing essential information such as the course name, institution, number of career prospects

and number of intakes. You can easily get more details by clicking the ‘View More’ button for each course. For example, the first card Diploma in Oral Health, once you press ‘View More’, it will display course details, specific career prospects and key skills you will learn if you take up the course. To make it easier, PolyQuest provides a search bar at the top. You can use it to search for specific courses or specific schools.

[Ze Kang - Academic Result Page]
When a new user logs in to the webpage and hasn’t taken a test yet, they will be greeted by a ‘Let's start' button which will bring them to begin their test. Once clicked, the user is moved to the Academic Result Page. On the Academic Result Page, an array of available subjects and interests are automatically fetched from our backend APIs. These entries are mapped into the page and displayed as dropdown menus for easy selection. Users have the option to choose from a list of subjects and assign grades to each one. Subject selection is flexible as well. Users can add and delete subjects as required but deletions are limited to a minimum of 6 selected subjects to ensure sufficient selection. English and Elementary Mathematics are compulsory and cannot be removed from the list. In addition to subjects, users can also select interests. There’s a requirement to choose at least one interest, but beyond that, users can select as many as they want. These interests are flexible, users can add or remove them at any time, and all selections are saved in an array tied to their unique student ID. Before moving to the next page, there are a few criteria that need to be met: the user must have at least 6 subjects selected, each of these 6 subjects must have a grade assigned to it, user must have at least one interest selected. If any criteria are not met, a toast notification will be displayed for the respective error. Once these criteria are met, the user can click the 'Next page' button. At this point, the student ID, selected subject names and grades are posted to the backend API. Additionally, the student ID and selected interests that are mapped to the interest names are sent as Booleans to the backend API tracking user preferences. After the data is successfully posted, the user is directed to the Test Page, where they can begin the actual test.

[Ze Kang - Test Page]
On this page, users can complete a comprehensive 25-question personality test. This page is designed to understand user preferences and recommend courses based on their responses. After successful submission of academic results and interests, users will be directed to the Test Page. The test has a total of 25 questions spread across 5 pages, with 5 questions per page. The questions are dynamically fetched from our backend API and displayed in individual question boxes. Each user receives a unique test Id, allowing them to take multiple test attempts. Each question offers a 'radio group' with five options, ranging from Strongly Disagree to Strongly Agree. These options are designed to capture a user’s level of agreement on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). These values play an important role in the recommendation calculations performed later. At the top of the page, there’s a stepper component that keeps track of each page completed, helping users see their overall progress. At the bottom, a progress bar shows the completion status of questions on the current page. To

the left and right of the progress bar, users will find the 'Previous' and 'Next' buttons, allowing them to navigate through the pages. Users must complete all 5 questions on each page before advancing. If they try to proceed without answering all questions, a toast notification will appear, prompting them to finish all responses before moving forward. Once all questions on a page are completed, clicking 'Next' will move the user forward and mark that page as completed on the stepper with a tick. Every time a user clicks 'Next' to move to a new page, their current responses are mapped and saved in an array along with the test Id and question Id. This ensures each page's responses are securely stored as they progress. Users can return to previous pages to make any changes to their answers. This flexibility allows them to review and adjust their responses as needed. On the final page, the 'Next' button changes to a 'Finish' button. Once all 25 questions are answered, users can click 'Finish'. Clicking 'Finish' posts the array of user responses, completed with the test Id and question Id to our backend API. A toast notification confirms the successful submission. After this, the user is redirected to the Result Page, where they'll see personalised course recommendations based on their responses.

[Malcolm - Bookmarks]

Now let's talk about the Bookmark feature. For each course card listed in the app, you'll see a bookmark icon at the top right corner of the course card. This lets users save courses they're interested in with a simple click.

The goal here is to help users easily keep track of courses they want to revisit later. Whether they're exploring different programs or narrowing down their options, bookmarking a course adds it to their personal list. This way, all saved courses are in one convenient spot, making it easy to compare and review.

[Yan Yee - Activity]

Entering the Activity page, you can view a list of tests you have completed, including details such as what courses were recommended to you, the date taken, and which sector was recommended to you. You will also be able to view the recommended courses from this by clicking on View More. In the same section, you can delete your past test records by clicking on the trash icon. This allows you to clean up your recommended courses if you no longer wish to keep them or if they no longer match your interests.

[Malcolm - Settings]

Next, let's take a look at the Settings page in our app, where users can manage their account details in three main sections: Personal Info, Academic Results, and Delete Account.

First, in Personal Info, users can easily view their name and email address. If they have a standard login, they'll also see an option to change their password by entering their current one and setting a new one. For those who signed in with Google, there's no need for a change password feature since their login is handled through Google's authentication system.

Moving on to the Academic Results section, where users can input and manage their subject grades. However, it starts off disabled – users need to first attempt the test to enable it. Once that's done, they're free to add, edit, or remove subjects and grades as needed.

Lastly, we have the Delete Account option. This section allows users to permanently delete their account if they wish. For standard login users, a password confirmation is required for added security. But for Google sign-in users, there's no need for that extra step. It's important to note that deleting an account is irreversible, and once done, all user data is removed from our system.

[Malcolm - Feedback]

Now let's look at the Feedback feature. This feature is divided into four categories: User Experience, Test Quality, Recommendation Accuracy, and Overall Satisfaction. Each section allows users to quickly rate different aspects of the app, from navigation ease to recommendation relevance, using a simple scale from 'Bad' to 'Excellent.'

Once users submit their feedback, it helps us identify areas for improvement. If there are any issues or questions, users can easily reach out to us at **rocky@ntu.edu.sg**, as noted at the bottom of the form.

[Carwyn - Swagger]
This is a list of the API endpoints and their respective URLs on Swagger.io.
This allows (Frontend and Backend) to collaborate effectively, and enhance future extensibility.
The frontend uses the URLs here to call the backend,
allowing creating (which is the POST URL), reading (which is the GET URL), deleting (which is the DELETE URL) and updating data (which is the PUT URL) in the database.

[Jerwin - ML]
Now for a basic overview of the ML part.
The code has an entry check every 5 seconds to see if there is a new test. Upon receiving a new test, the Machine Learning Model will get the responses from the test id and convert them into a mean response, which would serve as the recommendation score. Then this recommendation score would be compared to all the courses available. As all the courses have a recommendation score, we use a threshold range so only when the student's score is in this range, it will be recommended to the student. For our ML model, we use Multi-Layered

Perceptron Regressor (MLPRegressor) since it tends to generalise better especially where our features do not have a strict linear relationship with one another and it has high-dimensional patterns. Since we are also dealing with continuous variables (the responses), it is easier to implement this model as it requires less feature engineering as compared to other models.