Slide 1 (intro)

Good morning prof.
We are Group 53.

Slide 2 (Problem Statement)

So let's dive into the problem statement we have selected for our project.

Many local secondary school students face uncertainty when deciding their future educational and career paths, especially those who choose to embark on the polytechnic path where courses are more specialised.

As a result, their decisions about which polytechnic courses to pursue are often made impulsively and without a clear understanding of their interests and strengths. This uninformed decision-making can lead to dissatisfaction and misalignment between students' chosen fields of study and their true passions or aptitudes.

Slide 3 (Solution)

What if there was a way to help them streamline and refine their decision-making process?

WELCOME TO POLYQUEST

A web application designed to help secondary school students entering polytechnics make more informed decisions about their tertiary educational paths.

The core feature of this application is a comprehensive personality quiz that matches students with suitable polytechnic courses based on their individuality, characteristics, interests, and aptitudes.

This will allow potential polytechnic students to better navigate the myriad of courses available, ensuring their choices align with their true passions and strengths.

Slide 4 (Key Features of the App)

These are the key features of our app

Slide 5 (Use Case Diagram)

This is our use case diagram. Stemming from our main user, secondary students entering polytechnic.

Slide 6 (Feature List)

So these are the main functionalities of our web application which we will demonstrate in the demo later.

Slide 7 (Tech Stack)

This is our tech stack.
Using mainly React for our frontend.
Java SpringBoot for our backend and
PostgreSQL for our database.

Slide 8 (External APIs and Datasets used)

We used OAuth Google for authentication in our project.
We also used datasets from data.gov.sg for the relevant polytechnic course information.

Slide 9 (SWE Practices + Design Patterns)

Now we will talk about software engineering practices and design patterns that we have incorporated in our project.

Slide 10 (SWE Practices)

First, these are the Software Engineering Practices we have applied:

Slide 11 (Documentation - READme)

We added documentation for our project in the Readme file.
This serves as a guide for future developers, enabling long-term consistency and good development

Slide 12 (Documentation - Code Comments)

We also added comments to our codes.
This makes it, easier to read and understand for enhanced collaboration

Slide 13 (Documentation - Swagger UI)

We also used SwaggerUI API Docs

This allows (Frontend and Backend) to collaborate effectively, and enhance future extensibility.
The frontend uses the URLs here to call the backend,
allowing creating, reading, deleting and updating data in the database.

Slide 14 (Good Code Practices 1)

Now moving on to the Good Code Practices we used.
We ensured consistent code style and naming conventions which ensures that the code is readable, maintainable, and understandable.
We used prettier to enforce this.

Slide 15 (Good Code Practices 2)

Also, React enables component reusability, allowing elements to be used across the application effortlessly. Additionally, a composable structure makes the code modular and adaptable, simplifying future updates and expansion. Together, these practices boost both productivity and code quality.

Slide 16 (SCRUM Process)

Polyquest applies SCRUM for iterative, sprint-based development, ensuring continuous improvement. Where we assign roles to someone who sets priorities, another who oversees processes, and the Development Team that completes work each sprint.
Core elements include a Product Backlog, Sprint Backlog, and shippable Increment. Key events are 1-2 week sprints, daily updates via Telegram, and regular Sprint Reviews and Retrospectives. SCRUM values like commitment and openness keep the team adaptive and goal-focused.

Slide 17 (System Design)

Moving on to our project's system design.

Slide 18 (Backend Class Diagram)

This is our class diagram for our backend APIs.
Our backend is structured according to this

Slide 19 (System Architecture)

Polyquest's architecture follows a layered structure—presentation, application logic, and data—ensuring efficient data flow, security, and scalability. Users input data through a React UI, which sends it via an API Gateway to Polyquest's API for processing. The API interacts with PostgreSQL data models and returns recommendations or feedback.
Key benefits:
Separation of Concerns: Clear responsibilities for each layer, enhancing modularity.
Security: The database is secured in a private subnet, accessible only through the API Gateway.
Scalability: Independent scaling of components to manage increased traffic.

Slide 20 (Overview Diagram)

This diagram provides an architectural overview of a system with a frontend, backend, and database interaction. The Student interacts with various components, such as components, hooks, routes, and authentication (Auth). The frontend communicates with the backend via API endpoints using the Facade Pattern to simplify interactions.Backend contains controllers, services, and a database interface, which includes models and factories. The backend implements a Factory + Strategy Pattern to interact with the database and apply strategies for different data processing needs. For the database, A factory pattern is employed to manage connections and operations with the database. This design emphasizes modularity and pattern-based interactions for ease of scalability and maintenance.

Slide 21 (Intro to Design Pattern)

Moving on to design patterns

Slide 22 (Design Patterns we have used)

These are the 2 design patterns that we have used for our project. 1) Strategy and Factory Pattern 2) Model View Controller

Slide 23 (Strategy and Factory Pattern)

This diagram illustrates the use of Strategy and Factory Patterns for database management. Database factory Uses the Factory Pattern to instantiate different database types through a common Database Interface. Database Interface provides a unified way to interact with different databases. Strategy Pattern enables the selection of different database strategies (e.g., SQLite, PostgresSQL) based on specific requirements or contexts. Future Extensions allows easy integration of additional databases in the future by extending the current structure. Hence, this design supports flexibility in database choice and scalability for future database additions.

Slide 24 (MVC)

This diagram illustrates the Model-View-Controller (MVC) architecture. Model interacts with the database to fetch and manage data. View is responsible for the presentation layer, fetching presentation details from the controller. Controller handles user requests, processes data through the model, and sends responses back to the view for display. Data Flow starts from User requests go to the controller, which fetches data from the model, then sends it to the view for rendering, and finally responds to the user. This architecture promotes separation of concerns, making the codebase easier to maintain and scale.

Slide 25 (Traceability in Project Deliverables)

For project traceability, our development was guided by foundational diagrams, ensuring alignment across the development process. Each use case and test case is derived from these diagrams, maintaining a high level of traceability throughout the software development lifecycle.For instance, our diagram links directly to the login functionality, enabling consistent reference from requirements to testing and implementation. This approach helps ensure that each function, like the login feature, is properly documented, tested, and aligned with the overall project objectives.

Slide 26 (Demo)

Follow Demo Script (EXCLUDING INTRODUCTION and SWAGGER.IO Parts)

Slide 27 (Future Developments)

We have a few future developments in mind. First, we will make it easier for students to input, just by uploading their transcript. Next, we will have a sharing feature which generates an image report so students can compare it with their friends via social media. Lastly, We will add more detailed information for courses such as Career paths and projected salary, Courses popularity and the job market.

Slide 28 (END)