Group 08: Developer Diary – Road Trip Application

Week 1: Initial Planning and API Research

1 – Major Decision: Pivoting the Idea

- **What Happened:**
  Initially, our application was focused on flight planning using existing APIs. However, we discovered that none of the available APIs offered free plans suitable for our project scope, as those APIs do not provide details on future flights.
- **Discussion & Decision:**
  After a brainstorming session, we redefined our app to focus on road trip planning using weather information and itinerary planning, utilising OpenAI for itinerary generation.

2 – Design Decision: Choosing Framework and Language

- **What Happened:**
  We needed to decide on the tech stack for our road trip application. The primary requirements were a quick and easy setup process and a responsive frontend for user interaction.
- **Discussion & Decision:**
  We chose Spring Boot with Java because the team had more experience with it, allowing faster development. React was chosen because of our team was familiar with it and because React's component-based architecture simplifies building a dynamic and interactive UI.

Week 2: Designing Core Features

3 – Design Decision: Handling API Integration Challenges

- **What Happened:**
  One of our key features required obtaining weather data for a specific address entered by the user. This process involved two steps:

  1. Using a geocoding API to convert the address into coordinates.
  2. Passing those coordinates to the OpenWeather API to retrieve weather information.

  Initially, we planned to use the Google Geocoding API. However, we encountered account setup issues, so our project account was not verified in time to access the API.
- **Discussion & Decision:**
  While waiting for Google account approval, we switched to the OpenWeatherMap's Geocoding API as a temporary solution. It offered similar functionality and an affordable free tier, allowing us to keep development on track. Once our Google account was active, we integrated the Google Geocoding API because it offered better accuracy and compatibility with other Google Maps features used in the app.

4 – Major Decision: UI Design Challenges

- **What Happened:**
  We needed to create an intuitive and user-friendly interface for the road trip application that incorporated features such as location entry, itinerary generation, and map visualization.

- **Discussion & Decision:**
  To structure the UI effectively, we opted for a three-panel layout with a focus on clarity and accessibility. At the top, users can input locations using a simple text box. The central panel hosts the Google Map view, showing all added locations as pins. At the bottom of the right panel, users can click a button labelled *"Generate Itinerary"* to retrieve a detailed trip plan.

Week 3: Iteration and Testing

5 – Design Decision: Debugging Weather Data Issues

- **What Happened:**
  If a user entered a misspelled location, the app would crash. The OpenWeather API returned an error, but we had not accounted for it in the code.

- **Discussion & Decision:**
  We added error handling to display a user-friendly message: *"Invalid Address"*. We also extended error handling for all our API calls.

6 – Major Decision: Increasing Project Complexity

- **What Happened:**
  As the project progressed, we realized that typing in location names manually could lead to errors (e.g., misspellings) and a frustrating user experience.

- **Discussion & Decision:**
  To enhance usability and add a more polished feel to the application, we considered integrating location auto-complete functionality. Additionally, incorporating Google Maps directly into the interface would provide a visual context for the entered locations, improving user interaction and engagement.

Week 4: Finalisation and Presentation

7 – Major Decision: Finishing the Project and Preparing for Submission

- **What Happened:**
  As we approached the project deadline, our focus shifted from active development to finalizing the application, testing its features, and preparing documentation for submission.

- **Discussion & Decision:**
  This phase involved ensuring the app was functional, user-friendly, and well-documented for presentation. We conducted a thorough round of testing, focusing on edge cases like invalid location entries and handling API failures (e.g., weather, or geocoding APIs). Everyone was given a task to prepare our group to submit the project.

The role of Generative AI in our project

Generative AI was a crucial tool during the early stages of our project and throughout development. We initially used OpenAI to brainstorm project ideas, generate sample code, and create setup instructions. While AI provided inspiration and technical support, we had to adapt its outputs to suit our specific needs.

Using Generative AI for Project Ideas:

Prompt Used:
"Suggest some project ideas for a 4-week development project that integrates APIs and a database."

- **AI Output:** The suggestions included simple apps like a travel planner dashboard, a workout tracker, or a recipe finder. While helpful for brainstorming, most of these ideas were either too basic or lacked the complexity to showcase our skills.
- **Reflection:** The AI outputs sparked discussion, but we ultimately decided to conduct additional research. Through this, we conceptualized our road trip planning app, combining weather data, Google Maps, and itinerary generation with generative AI to add depth and uniqueness. This step underscored the value of supplementing AI suggestions with independent research to develop well-rounded ideas.

Using Generative AI for Sample Code and Technical Support:

Prompt Used:
"Provide sample code for Spring Boot Java for integrating the OpenAI's API."

- **AI Output:** The AI provided a useful starting point for setting up API integrations. However, some examples were incorrect, requiring some adjustments to correct make the call to the API
- **Reflection:** AI accelerated the coding process by offering templates and suggestions, but careful validation and debugging were necessary to make the code production-ready. This taught us to treat generative AI as a supplementary tool rather than a definitive source of solutions.

Using Generative AI for Setup Instructions:

Prompt Used:
"Give me a README with how to download the necessary components for a Spring Boot Java application with Maven"

- **AI Output:** The AI provided a step-by-step guide, setting up the Spring Boot environment with Maven dependencies. The detail given was comprehensive as it also provided troubleshooting instructions as well. This was tested and we also added alternative setups.
- **Reflection:** AI can serve as an excellent starting point to develop from.