Project Report: Red Bus Mobile App Testing Tool

Objective:

The goal was to create a tool that uses a multimodal Large Language Model (LLM) to generate detailed testing instructions for the Red Bus mobile app, focusing specifically on its bus selection feature. The tool was designed to analyze screenshots and contextual inputs, producing comprehensive test cases to validate the app's functionality.

Scope:

Core Features Covered:

- **Source, Destination, and Date Selection:** Ensures the user can choose where they're going, where they're starting, and when.
- **Bus Selection:** Validates the display and selection process of available buses.
- **Seat Selection:** Checks that the user can pick their seat on the selected bus.
- **Pick-up and Drop-off Point Selection:** Ensures accurate selection and display of journey start and end points.

Bonus Features:

- **Offers:** Highlights any discounts or promotions available.
- **Filters:** Validates sorting buses by time, price, or other criteria.
- **Bus Information:** Ensures correct display of bus details such as amenities, photos, and user reviews.

User Flow and Prompting Strategy

User Flow:

1. **Landing on the App:**
    - **Action:** User opens the Red Bus app and lands on the home screen.

- **Prompt Consideration:** The prompt was designed to recognize the initial state of the app, ensuring that test cases would cover the user's options from the home screen (e.g., selecting source, destination, and date).

2. **Source, Destination, and Date Selection:**
   - **Action:** User selects the source, destination, and travel date.
   - **Prompt Consideration:** The prompt accounts for verifying the correct input and confirmation of these details, generating specific test cases to ensure selections are saved and displayed correctly.
3. **Bus Selection:**
   - **Action:** User views available buses and selects one for booking.
   - **Prompt Consideration:** The prompt was designed to validate the display of bus options based on user inputs, ensuring that selecting a bus correctly advances to the next steps.
4. **Seat Selection:**
   - **Action:** User selects a seat from the available options.
   - **Prompt Consideration:** Test cases were generated to verify accurate seat availability and selection.
5. **Pick-up and Drop-off Point Selection:**
   - **Action:** User selects pick-up and drop-off points.
   - **Prompt Consideration:** The prompt includes test cases to confirm that pick-up and drop-off options are correctly displayed and selections are properly registered.
6. **Bonus Features:**
   - **Action:** User applies filters, views bus information, or checks available offers.
   - **Prompt Consideration:** Test cases were generated to validate the correct application of filters, accurate display of bus details, and proper display of available offers.

Prompting Strategy:

1. **Initial Setup:**
   - **Capture Screenshots:** Screenshots were taken at each critical interaction point within the app.
   - **Contextual Inputs:** Context for each screenshot was defined to guide the LLM in understanding the flow.
2. **Prompt Design:**
   - **Segmentation:** The prompt was segmented based on core functionalities (e.g., Source/Destination/Date Selection) and bonus features (e.g., Filters, Offers).
   - **Multi-shot Prompting:** Used for complex scenarios to ensure detailed and accurate test cases.
3. **Detailed Test Case Generation:**
   - **Incorporation of Industry Standards:** Structured to include ID, Scenario, Description, Pre-conditions, Steps, Data, Expected Results, and Comments.
   - **Iterative Refinement:** The prompt was refined iteratively for clarity and accuracy.
4. **A/B Testing Experience:**
   - **Product Management Perspective:** My experience in product A/B testing allowed me to approach this project with a product management mindset. Understanding the pain points during application testing helped in developing prompts that anticipated real-world issues, leading to a more robust and user-centered tool.
5. **Final Output and Feedback:**
   - **Validation:** The generated test cases were reviewed for completeness and accuracy.
   - **User Interaction:** The tool allows users to add comments and status (Pass/Fail) to each test case, providing a dynamic and interactive final output.

Tech Stack and Implementation

LLM Used:

- **Google's Gemini-1.5-Flash:**
    - **Reason for Selection:** Chosen for its robust multimodal capabilities, allowing it to process both text and image inputs effectively. This LLM was best suited for generating the detailed and contextually accurate test cases required for this project.

Alternative LLMs Considered:

- **OpenAI's GPT-4:**
    - **Why Not Used:** Although powerful, GPT-4 lacks native multimodal integration without additional customization, making it less suitable for this project.
- **Cohere and other Open Source LLMs:**
    - **Why Not Used:** While viable for text-based tasks, they don’t support multimodal processing as effectively as Google’s Gemini, requiring additional workarounds.

Backend Implementation:

- **Flask:**
    - **Reason for Selection:** I have prior experience with Flask, making it a natural first choice. It provides a simple yet powerful framework for handling HTTP requests and integrating with the LLM.
    - **Implementation Details:** Flask was used to handle front-end inputs, process the screenshots and contextual data, and render the LLM-generated test cases.

Frontend Implementation:

- **HTML/CSS:**
    - **User Interface:** The front-end was kept simple to focus on functionality. A basic HTML form was used to upload screenshots and input context, with CSS for styling.

    - **Final Output:** The generated test cases are displayed in a formatted, readable manner with options for the user to input the status (Pass/Fail) and comments.

Integration Considerations:

- **Why LangChain Wasn’t Used:** While LangChain provides advanced capabilities for chaining multiple LLMs and tools together, the project's scope didn’t require its complexity. The task was straightforward enough to be handled by a single LLM integrated through Flask.

Conclusion:

This project successfully demonstrated the ability to use a multimodal LLM to generate comprehensive test cases for a mobile app based on screenshots and contextual inputs. The combination of a well-structured prompt, a robust LLM, and a simple yet effective backend setup resulted in a tool that meets the project's objectives. My experience in A/B testing and product management played a crucial role in anticipating the needs and challenges of the testing process, ensuring that the tool is both practical and user-friendly.