Hackathon Project Phases Template

Project Title:

AutoSage App Using Gemini Flash: Intelligent Vehicle Insights with Gemini Flash.

Team Name:

Auto Genius

Team Members:

- Ravirala Siddarth
- Chalamalla HarshaVardhan Reddy
- Tandasa Parthiv
- Nanagiri Prateek Sai Raj
- Gulgoth Rithwik

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered vehicle expert tool using Gemini Flash to help users compare and analyze vehicle specifications.

Key Points:

1. **Problem Statement:**
    - Many users struggle to find reliable, up-to-date information about four-wheelers before making a purchase decision.
2. **Proposed Solution:**
    - An AI-powered application using **Gemini Flash** to provide **real-time vehicle specifications and comparisons.**
3. **Target Users:**
    - **Vehicle buyers** looking for specifications and comparisons.
    - **Eco-conscious consumers** searching for hybrid and electric vehicle options.
4. **Expected Outcome:**
    - A functional **AI-powered vehicle information app** that provides insights based on real-time data .

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the AutoSage App.

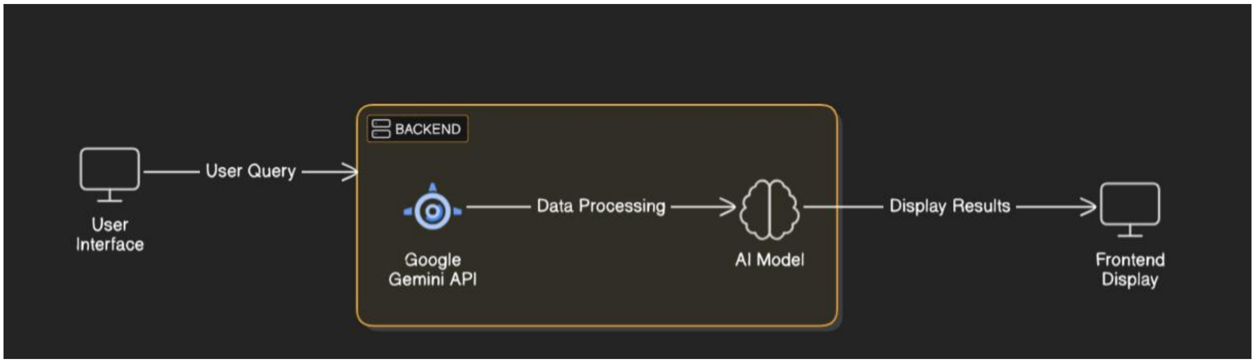
Key Points:

1. **Technical Requirements:**
   - Programming Language: **Python**
   - Backend: **Google Gemini Flash API**
   - Frontend: **HTML,CSS,JAVASCRIPT**
   - Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**
   - Ability to **fetch vehicle details** using Gemini Flash API.
   - Display **specifications and comparisons** in an intuitive UI.
3. **Constraints & Challenges:**
   - Ensuring real-time updates from **Gemini API**.
   - Handling **API rate limits** and optimizing API calls.
   - Providing a **smooth UI experience** with Javascript

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**
   - User enters vehicle-name or vehicle models
   - Query is processed using **Google Gemini API**.
   - AI model fetches and processes the data.
   - The frontend displays **vehicle details and comparisons**.
2. **User Flow:**
   - Step 1: User enters a Vehicle-Names
   - Step 2: The backend **calls the Gemini Flash API** to retrieve vehicle data.
   - Step 3: The app processes the data and **displays results** in an easy-to-read in a tabular format about specification.
3. **UI/UX Considerations:**
   - **Minimalist, user-friendly interface** for seamless navigation.
   - **Filters for price, mileage, and features**.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	🔴 High	5 hours (Day 1)	End of Day 1	Siddarth	Google API Key, Python.	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	3 hours (Day 1)	End of Day 1	Rithwik Parthiv	API response format finalized	Basic input fields
Sprint 2	Vehicle Search & Comparison	🔴 High	3 hours (Day 2)	Mid-Day 2	Harsha	API response, UI elements ready	Comparison functionality with filters
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	Siddarth Prateek	UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	Siddarth Parthiv	API response, UI layout completed	Responsive, better user experience
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(🔴 High Priority) Set up the **environment** & install dependencies.
(🔴 High Priority) Integrate **Google Gemini API**.
(🟡 Medium Priority) Build a **basic UI with input fields**.

Sprint 2 – Core Features & Debugging (Day 2)

(🔴 High Priority) Implement **search & comparison functionalities**.
(🔴 High Priority) Debug API issues & handle **errors**.

Sprint 3 – Testing, Enhancements & Submission (Day 2)

(🟡 Medium Priority) Test API responses, refine UI, & fix UI bugs.
(🟢 Low Priority) Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

Implement core features of the AutoSage App.

Key Points:

Project Goal: To create a web application that allows users to compare the specifications of two different vehicles side-by-side.

Technology Stack Used:

Frontend: HTML, CSS, JavaScript

Backend: Flask (Python web framework)

Programming Language: Python

Development Process:

Designed and implemented the user interface (HTML, CSS) for inputting vehicle names and displaying comparison results.

Developed the Flask backend to handle user input, retrieve vehicle data from a local dataset, and render the comparison results.

Implemented JavaScript functionality for dynamic UI updates, recent search suggestions, and user interactions.

Focused on creating a clean, responsive, and user-friendly design.

Challenges & Fixes: (Since we didn't use an API, let's focus on UI/UX challenges)

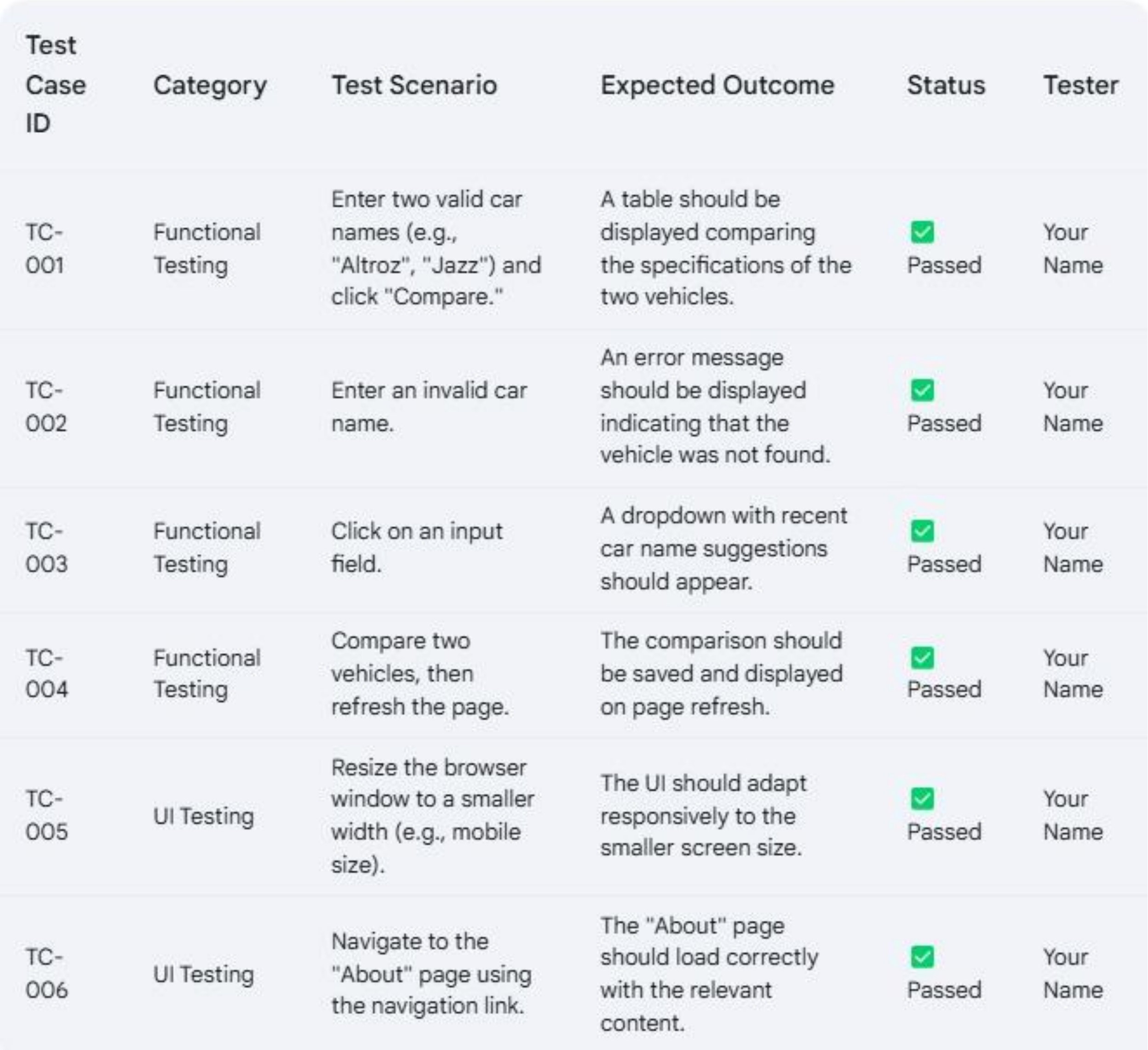
Challenge: Making the layout adapt well to different screen sizes.

Fix: Implemented responsive design using CSS media queries to adjust the layout for smaller screens.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the AutoSage vehicle comparison app functions correctly and provides a smooth user experience.



Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Enter two valid car names (e.g., "Altroz", "Jazz") and click "Compare."	A table should be displayed comparing the specifications of the two vehicles.	✅ Passed	Your Name
TC-002	Functional Testing	Enter an invalid car name.	An error message should be displayed indicating that the vehicle was not found.	✅ Passed	Your Name
TC-003	Functional Testing	Click on an input field.	A dropdown with recent car name suggestions should appear.	✅ Passed	Your Name
TC-004	Functional Testing	Compare two vehicles, then refresh the page.	The comparison should be saved and displayed on page refresh.	✅ Passed	Your Name
TC-005	UI Testing	Resize the browser window to a smaller width (e.g., mobile size).	The UI should adapt responsively to the smaller screen size.	✅ Passed	Your Name
TC-006	UI Testing	Navigate to the "About" page using the navigation link.	The "About" page should load correctly with the relevant content.	✅ Passed	Your Name

Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**