Project Title:

Basho: The Gemini Landmark Description App

Team Name:

Brogrammers

Team Members:

- Soham Chitimal
- Majari Teja Srinivas

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered landmark identification and description tool using Gemini, enabling users to receive detailed, structured, and multilingual descriptions of cultural and historical landmarks by simply uploading an image.

Key Points:

Problem Statement:

- Travelers often struggle to find comprehensive and reliable information about landmarks when exploring new places.
- Tour guides and history enthusiasts need quick, accessible insights into architectural significance and historical events.
- There is a need for a multilingual and inclusive tool to provide landmark information to users from different backgrounds.

Proposed Solution:

- An AI-powered application that allows users to upload images of landmarks and receive structured descriptions, including history, architectural details, and fun facts.
- Integrated multilingual support to cater to global and diverse travelers.
- Features deep conversation capabilities for users to ask follow-up questions about the landmark.

Target Users:

- Travelers exploring new cities.
- Tour guides looking for quick landmark details.
- History and architecture enthusiasts.

- Students researching historical sites.

Expected Outcome:

- A user-friendly AI-powered application that provides instant, structured, and rich insights into global landmarks.

Phase-2: Requirement Analysis

Objective:
Define the technical and functional requirements for the Basho App.

Technical Requirements:

- **Programming Language:** JavaScript (React.js for frontend, Node.js for backend)
- **Backend:** Google Gemini API, MongoDB for image storage
- **Frontend:** React.js with a clean, responsive UI along with Tailwind CSS
- **Database:** MongoDB for metadata and images (Local storage also suitable)

Functional Requirements:

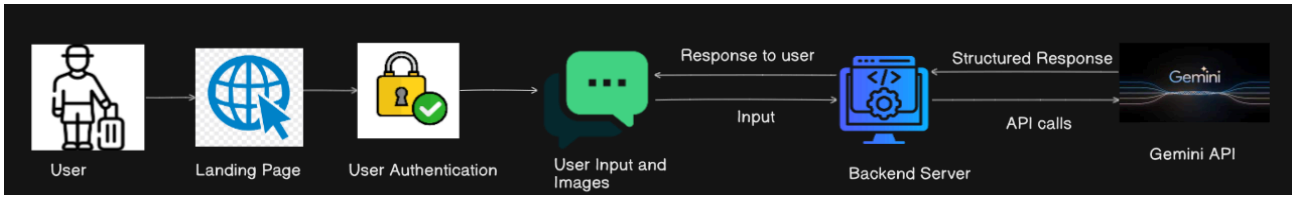
- Accept multiple images for landmark identification.
- Process images and generate structured descriptions.
- Provide multilingual support.
- Enable deep conversational interaction for detailed inquiries.

Constraints & Challenges:

- Ensuring accurate and fast image recognition.
- Handling multiple image uploads efficiently.
- Implementing seamless multilingual capabilities.

Phase-3: Project Design

Objective:
Develop the system architecture and user flow of the application.



System Architecture:

1. User uploads an image of a landmark.
2. The backend processes the image using Google Gemini Flash API.
3. AI generates structured descriptions covering historical, architectural, and cultural details.
4. The app displays results in a user-friendly interface, with follow-up queries enabled.

User Flow:

1. **Step 1:** User uploads an image and enters a brief prompt (voluntarily can give context).
2. **Step 2:** AI processes the image and fetches relevant data.
3. **Step 3:** The app presents structured descriptions in multiple languages.
4. **Step 4:** Users can ask further questions for deeper insights.

UI/UX Considerations:

- Simple, aesthetic and intuitive interface with easy image upload.
- Responsive design for mobile and desktop use.

Phase-4: Project Planning (Agile Methodologies)

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Frontend Environment Setup	🔴 High	2 hours (Day 1)	Mid Day 1	Teja	React , Tailwind CSS , Vite	Working frontend setup
Sprint 1	Backend environment Setup	🔴 High	2 hours (Day 1)	Mid Day 1	Soham	Express , NodeJS , Gemini	Backend Server Setup
Sprint 2	Frontend UI Development	🟡 Medium	4 hours (Day 1 & 2)	End of Day 1	Teja	UI elements and components	UI pages with components setup
Sprint 2	API testing and server upgrade	🔴 High	6 hours(Day 1 & 2)	Mid-Day 2	Soham , Teja	Gemini API , Express , MongoDB	Stable Backend Server
Sprint 3	Testing & UI Enhancements	🟡 Medium	3hours (Day 2)	Mid-Day 2	Soham , Teja	API response, UI layout completed	Responsive UI, UX improvement
Sprint 3	Final Presentation & Deployment	🟢 Low	2 hour (Day 2)	End of Day 2	Soham , Teja	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 in queries**.

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 the core functionalities of the Basho App.

Technology Stack Used:

- **Frontend:** React.js with Tailwind CSS
- **Backend:** Node.js with Express.js
- **AI Model:** Google Gemini API
- **Database:** Local Storage(for storing images and metadata)

Development Process:

1. Implement image upload and processing.
2. Integrate AI for landmark description generation.
3. Develop multilingual support for user accessibility.
4. Enhance UI for structured and interactive user responses.

Challenges & Fixes:

- **Challenge:** Ensuring accurate landmark identification.
  Fix: Optimize AI queries and train for better recognition.
- **Challenge:** Managing large image uploads efficiently.
  Fix: Implement image compression before processing.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the Basho App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Query with images of Charminar and Taj Mahal	Should differentiate and describe both	✅ Passed	Entire Team
TC-002	Functional Testing	Deep conversation by storing various contexts	Images stored and contextual conversation	✅ Passed	Entire Team
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	⚠ Needs Optimization	Entire Team
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	⚠ Needs Optimization	Entire Team
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	✅ Fixed	Entire Team
TC-006	Deployment Testing	Host the app using Netlify and Vercel	App should be accessible online.	❌Backend Deployement Failed	Entire Team
TC-007	Deep Talk	Check if deeptalk is working	Deeper conversations should be supported	⚠ Needs Optimization	Entire Team
TC-008	Frontend Deployment	Frontend Netlfiy Deployment	Frontend part accessible online	✅ Fixed	Entire Team