**Hackathon Project Phases Template** for the **Vision Tagger AI** project.

# **Hackathon Project Phases Template**

## **Project Title:**

**Vision Tagger AI using Gemini Vision Pro**

## **Team Name:**

Phantom Force

## **Team Members:**

* Nanda Nagulavancha
* Sai Deekshith Konda
* Mahesh Babu
* Lahari Sadanala

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

Develop an advanced AI-driven image tagging system that leverages Google's Gemini Vision Pro to generate descriptive tags and structured JSON metadata for images in real time.

### **Key Points:**

1. **Problem Statement:**
   * Manually tagging and categorizing images is time-consuming and inefficient.
   * Businesses and content creators need an automated system to generate accurate, context-aware metadata for images.
2. **Proposed Solution:**
   * A web-based application using Gemini Vision Pro to analyze images and provide meaningful tags.
   * The system generates detailed JSON metadata, including object recognition, scene descriptions, and context-based attributes.
3. **Target Users:**
   * Photographers, designers, and content creators.
   * E-commerce platforms needing automatic product categorization.
   * Data scientists and researchers managing large image datasets.
4. **Expected Outcome:**
   * A functional AI-powered image tagging tool that simplifies image organization and improves searchability.

## **Phase-2: Requirement Analysis**

### **Objective:**

Define the technical and functional requirements for the VisionTagger AI.

### **Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python, Java Script**
   * Backend: **Google Gemini Flash API, Flask**
   * Frontend: **HTML, Tailwind CSS**
   * Database: **Firebase for image metadata storage and retrieval**
2. **Functional Requirements:**
   * Upload images for AI-powered tagging.
   * Generate JSON metadata with detailed descriptions.
   * Allow users to edit and export metadata.
   * Provide a searchable and filterable interface for tagged images.
3. **Constraints & Challenges:**
   * Storing and Optimizing real-time processing for large image datasets.Handling API rate limits and optimizing API calls.
   * Managing API rate limits and response latency.
   * Ensuring high accuracy in generated tags.

## **Phase-3: Project Design**

### **Objective:**

Develop the architecture and user flow of the application.

### **Key Points:**

1. **System Architecture:**
   * Users upload an image via the web interface.
   * The image is sent to the backend for **Gemini Vision Pro processing**.
   * AI generates and returns descriptive **tags** and **metadata**.
   * The metadata is stored in Firebase and displayed to users.
2. **User Flow:**
   * **Step 1:** User uploads an image.
   * **Step 2:** The backend sends the image to Gemini Vision Pro.
   * **Step 3:** AI processes the image and returns metadata.
   * **Step 4:** The metadata is displayed and stored for future use.
3. **UI/UX Considerations:**
   * Clean, intuitive interface for easy navigation.
   * Image upload functionality.
   * JSON file editor

## 

## **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 | 6 hours (Day 1) | End of Day 1 | Nanda, Deekshith | Google API Key, Python, Flask setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Lahari, Mahesh | API response format finalized | Basic UI with input fields |
| Sprint 2 | Image Upload and metadata generator | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Deekshith | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | Deekshith | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Mahesh | API response, UI layout completed | Responsive UI, 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 image metadata extraction and JSON generation.  
 **(🔴 High Priority)** Debug API issues & improve error handling in image processing.

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

**(🟡 Medium Priority)** Test API responses, refine UI, and fix UI-related issues.  
 **(🟢 Low Priority)** Final project deployment and demo preparation.

## **Phase-5: Project Development**

### **Objective:**

### Implement core features of the VisionTagger AI application.

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** HTML,CSS,JavaScript
   * **Backend:** Google Gemini Vision Pro API
   * **Programming Language:** Python, Flask
2. **Development Process:**
   * Implement **API key authentication** and **Gemini Vision API integration**.
   * Develop **image metadata extraction** and **JSON metadata generation** Optimize
3. **Challenges & Fixes:**
   * **Challenge:** Inconsistent metadata extraction for some image formats.  
      **Fix:** Preprocess images to standardize formats before sending requests.

## **Phase-6: Functional & Performance Testing**

### **Objective:**

Ensure that the VissionTagger AI works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Upload an image and generate metadata tags | |  | | --- | | Relevant tags should be extracted and displayed |  |  | | --- | |  | | ✅ Passed | Deekshith ,Lahari |
| TC-002 | Functional Testing | Generate JSON metadata for an image | JSON should include accurate and structured data | ✅ Passed | Nanda,Mahesh |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | ⚠ Needs Optimization | Deekshith |
| TC-004 | Bug Fixes & Improvements | Fixed missing metadata for certain images | Data extraction should be accurate | ✅ Fixed | Nanda |

## **Final Submission**

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