**Hackathon Project Phases Template** for the **ProvisionAI :** project.

# **Hackathon Project Phases Template**

## **Project Title:**

### ProVisionAI: Unleashing the Power of Gemini Vision for Image Annotation

## **Team Name:**

The Blind Guy

## **Team Members:**

* Nishchala
* Keerthana
* Manasvi
* Ravi
* Nitvik

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

### **Objective:**

**Develop an AI-powered assistance tool that helps visually impaired users by identifying objects and reading text aloud.**

### **Key Points:**

#### **Problem Statement:**

* **Blind and visually impaired individuals struggle to identify objects, read text, and navigate daily tasks independently.**
* **Existing solutions are either expensive (NVIDIA Jetson or Google Cloud Vision API), inaccurate (Google Assistant or Siri), or difficult to use (OpenCV or TensorFlow).**

#### **Proposed Solution:**

* **An AI-powered application that capture images using a webcam, describe them using AI-generated content, and allow users to interact with it via a Flask web interface.**
* **Uses OpenCV for image capturing, Gemini’s 1.5 flash model (through google.generativeai library) for content generation about the image description, pyttsx3 for Text-to-Speech, speech\_recognition for post requests from the user, Flask Web Application for the interface, Image from PIL library for managing, opening and manipulating images, os to get access to the operating system for making directories, subprocessor for accessing other python scripts(dumbo.py) , jsonify for converting data to JSON format.**

#### **Target Users:**

* **Visually impaired individuals needing assistance in object recognition and text reading.**
* **Organizations and caregivers supporting blind users.**

#### **Expected Outcome:**

* **A functional AI-powered assistance tool that helps visually impaired users recognize objects and read text aloud.**

# **Phase-2: Requirement Analysis**

### **Objective:**

**Define the technical and functional requirements for the Blind Assistance App.**

### **Key Points:**

#### **Technical Requirements:**

* **Programming Language: Python, JavaScript**
* **Libraries: OpenCV (image processing), pyttsx3 (text-to-speech), speech\_recognition (speech-to-text), google.generativeai (AI content generation), PIL (image handling), Flask (web framework), os (file system operations), subprocess (run external processes), time (timing control), logging (error logging), jsonify (JSON responses for Flask).**
* **Hardware: Webcam (for capturing images)**
* **Database: Not required initially**

#### **Functional Requirements:**

* **Capture images using a webcam and process them only when the scene changes.**
* **Generate and describe the contents of captured images using AI (Gemini model) in real-time.**
* **Convert the generated descriptions of the image and user queries into speech output for the user.**

#### **Constraints & Challenges:**

* **Accurate scene change detection and image capture in real-time.**
* **Ensuring reliable speech recognition for user commands.**
* **Seamless real-time integration of image description, speech output, and query handling.**

# 

# **Phase-3: Project Design**

### **Objective:**

**Develop the architecture and user flow of the application.**

### **Key Points:**

#### **System Architecture:**

1. **The user clicks a button to capture an image via the webcam.**
2. **The image is processed using OpenCV for object detection.**
3. **The captured image is analyzed using the Gemini AI model to generate a natural description of the scene.**
4. **The generated description is converted into speech using pyttsx3.**
5. **The app reads out the image description, and users can ask follow-up questions, which are processed and answered based on the image.**

#### **User Flow:**

* **Step 1: User presses a button to capture an image via the webcam.**
* **Step 2: The website processes the image, detects scene changes, and captures the frame.**
* **Step 3: The captured image is analyzed, and a description is generated using the Gemini AI model.**
* **Step 4: The app reads aloud the generated image description, and the user can ask follow-up questions about the image.**

#### **UI/UX Considerations:**

* **Minimalist UI with a large “Capture” button for accessibility.**
* **Simple audio feedback through voice interaction through pyttsx3, which reads out the image description and answers user questions.**

# **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 & library installation** | **🔴 High** | **2 hours** | **Day 1 (Morning)** | **Ravi** | **Python, OpenCV,** | **All dependencies installed** |
| **Sprint 1** | **Capture image via webcam** | **🔴 High** | **2 hours** | **Day 1 (Afternoon)** | **Keerthana** | **OpenCV setup completed** | **Image successfully captured** |
| **Sprint 1** | **Implement Image analysis and description generation** | **🔴 High** | **2 hours** | **Day 1 (Afternoon)** | **Manasvi** | **OpenCV + gemini-1.5-flash** | **Description for the image taken is generated and read out succesfully** |
| **Sprint 1** | **User Input and Interaction** | **🔴 High** | **4 hours** | **Day 1 (Afternoon)** | **Nishchala** | **speech\_recognition+recognizer.listen()+recognizer.recognize\_google()** | **User can ask questions via speech, and app responds correctly** |
| **Sprint 1** | **Error handling & optimizations** | **🔴 High** | **2 hours** | **Day 1 (Evening)** | **Nitvik** | **Debug logs available** | **Faster, more accurate output** |
| **Sprint 2** | **Final testing & UI improvements** | **🟢 Low** | **2 hours** | **Day 2** | **Entire Team** | **All modules integrated** | **Fully functional website** |

# 

# **Phase-5: Project Development**

### **Objective:**

**Implement core features of the Blind Assistance App.**

### **Key Points:**

#### **Technology Stack Used:**

* **Frontend: HTML, CSS, JavaScript**
* **Backend: Flask, OpenCV, pyttsx3, Google Gemini AI**
* **Programming Language: Python**

#### **Development Process:**

* **Implement API key authentication and integrate OpenCV for image processing.**
* **Develop image capture and description generation functionalities using AI (Google Gemini).**
* **Optimize the text-to-speech feature for improved clarity and user interaction.**

#### **Challenges & Fixes:**

|  |  |
| --- | --- |
| **Challenge** | **Fix** |
| **Blurry or low-quality image capture** | **Apply image preprocessing (grayscale & sharpening)** |
| **Delayed image description or processing** | **Optimize image capture and processing pipeline using efficient scene change detection logic** |
| **Voice commands not recognized correctly** | **Improve voice recognition by adjusting ambient noise handling and using better microphone input tuning** |

# 

# 

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

### **Objective:**

**Ensure that the Blind Assistance App works as expected.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| **TC-001** | **Functional Testing** | **Capture image and generate a description** | **Image description should be generated** | **✅ Passed** | **Ravi** |
| **TC-002** | **Functional Testing** | **Answer questions about the image** | **Questions should be answered accurately** | **⚠ Needs Optimization** | **Nishchala** |
| **TC-003** | **Performance Testing** | **Text-to-Speech response time (fast)** | **Speech output should be near-instant** | **⚠ Needs Optimization** | **Ravi** |
| **TC-004** | **Bug Fixes & Improvements** | **Bug Fixes & Improvements** | **Fixed description generation errors** | **✅ Fixed** | **Keerthana** |
| **TC-005** | **Final Validation** | **Ensure voice interaction works seamlessly** | **Clear audio output and voice recognition** | **❌ Failed - Needs Improvements** | **Nitvik** |
| **TC-006** | **Deployment Testing** | **Host the website using Flask** | **website should be accessible via Flask locally** | **🚀 Deployed** | **Nitvik** |

# **Final Submission**

### **Deliverables:**

**✅ Project Report – Based on templates  
✅ Demo Video (3-5 Minutes)  
✅ GitHub/Code Repository Link  
✅ Final Presentation**