**Hackathon Project Phases Template** that ensures students can complete it efficiently while covering all six phases. The template is structured to capture essential information without being time-consuming.

# Hackathon Project Phases Template

**Project Title:**

Gesture Based Human Computer Interaction System using OpenCv, Mediapipe and palm's text -bison 001

**Team Name:**

Innoventures

**Team Members:**

* Thada Jyothirmai
* Sahithya Adavelly
* Bantu Varsha
* Bandari Navyasri

## Phase-1: Brainstorming & Ideation

**Objective:**

* Identify the problem statement.
* Define the purpose and impact of the project.

**Key Points:**

1. **Problem Statement:** Traditional input methods, such as keyboards, mice, and touchscreens, can be limiting in various scenarios where hands-free interaction is preferred or necessary. Existing digital drawing tools require additional hardware like styluses, and adjusting screen brightness manually can be cumbersome. Moreover, people with disabilities may struggle with standard input devices.

Thus, there is a need for a **gesture-based human-computer interaction system** that enables users to perform tasks like **drawing, screen brightness adjustment, and canvas clearing** using only hand gestures.

1. **Proposed Solution:** This project introduces a **gesture-controlled interface** using **computer vision (OpenCV) and hand tracking (Mediapipe)** to enable hands-free interaction. The system will:

* **Allow users to draw on a virtual canvas** using their right-hand index finger.
* **Adjust screen brightness** dynamically based on left-hand finger gestures.
* **Clear the drawing canvas** when a left-hand shake gesture is detected.

The solution provides a more **intuitive, accessible, and interactive computing experience**, enhancing usability for various applications.

3. **Target Users:**

* **Artists & Designers:** Provides a hands-free way to sketch digitally.
* **People with Disabilities:** Enhances accessibility by replacing touch-based controls.
* **Educators & Presenters:** Enables interactive, touch-free whiteboarding.
* **Gamers & Tech Enthusiasts:** Offers an engaging way to interact with digital content.
* **General Users:** Simplifies tasks like brightness control without physical buttons.

4. **Expected Outcome:**

* A **fully functional real-time gesture-based system** for drawing and brightness control.
* A **smooth and interactive interface** with high accuracy in gesture recognition.
* **Increased accessibility** for users with mobility impairments.
* A **touch-free, convenient way** to interact with digital devices.

## Phase-2: Requirement Analysis

**Objective:**

● Define technical and functional requirements.

**Key Points:**

1. **Technical Requirements:** This project utilizes **computer vision and real-time gesture recognition**.

**Languages & Frameworks:**

* **Python** – Primary programming language for implementation.
* **OpenCV** – For real-time video capture and image processing.
* **Mediapipe** – For hand tracking and gesture recognition.
* **NumPy** – For mathematical operations (e.g., distance calculations).
* **Screen Brightness Control (sbc)** – To adjust screen brightness based on hand gestures.

**Tools & Dependencies:**

* **Webcam** – Required for real-time hand detection.
* **Collections (Deque)** – Used to track hand movement history for shake detection.

**Installation of Dependencies:**

pip install opencv-python mediapipe numpy screen-brightness-control

1. **Functional Requirements:**

**Hand Gesture Recognition:**

* Identify **left** and **right** hands separately.
* Detect key landmarks (fingertips, palm, wrist) to interpret gestures.

**Right-Hand Functionality: Drawing Mode**

* Track the **index fingertip** position in real-time.
* Allow users to **draw** on a virtual canvas by moving their index finger.
* Display the drawn content over the live camera feed.

**Left-Hand Functionality: Brightness Control**

* Measure the **distance between thumb and index finger** to adjust brightness dynamically.
* Map gesture-based brightness control to screen brightness settings.
* Display **brightness percentage** on the screen.

**Shake Gesture (Canvas Reset)**

* Detect **rapid hand movements** using a **deque-based motion tracker**.
* If movement surpasses a threshold, **clear the drawing canvas**.
* Provide a **visual indication** when the shake gesture is detected.

**User Interaction & Exit Mechanism**

* Provide **visual feedback** ("Drawing...", "Brightness: 50%") for gesture recognition.
* Allow users to **exit** the application by pressing **‘q’** on the keyboard.

1. **Constraints & Challenges:**

**1. Environmental & Hardware Limitations**

* **Lighting conditions** affect hand detection accuracy.
* **Low-resolution webcams** may reduce tracking precision.

**2. Gesture Recognition Limitations**

* Incorrect hand positioning might cause **misclassification** of gestures.
* The system may need **fine-tuning** to distinguish similar hand movements.

**3. Performance & Real-Time Processing**

* The application must **run at 30+ FPS** to ensure smooth real-time tracking.
* High **CPU usage** could be an issue on **low-end devices**.

**4. Compatibility Issues**

* **Screen brightness control** may **not work on macOS/Linux** due to system restrictions.
* Different webcams have **varying field-of-view (FOV)**, impacting hand detection accuracy.

**5. User Adaptation & Learning Curve**

* Users may need **time to get accustomed** to the gestures.
* On-screen **instructions and visual cues** can help improve usability.

## Phase-3: Project Design

**Objective:**

● Create the architecture and user flow.

**Key Points:**

1. **User Flow:** The user interacts with the system in the following way:

**Launch the Application:**

* The webcam starts capturing video.
* Hand detection begins.

**Gesture-Based Interactions:**

* **Right Hand:**
* Move the **index finger** to start **drawing**.
* The **virtual ink** follows finger movements.
* **Left Hand:**
* Adjust brightness by **pinching thumb and index finger** closer/farther.
* Shake the hand to **clear the canvas**.

**Real-Time Feedback:**

* Drawing actions appear as **white strokes** on the screen.
* Brightness levels update dynamically with **percentage text overlay**.

**Exit the Program:**

* Press **‘q’** to close the application.

1. **System Architecture Diagram:**

**FlowChart:**



1. **UI/UX Considerations:**

Since this is a **gesture-based interface**, the focus is on **visual feedback** rather than traditional UI elements.

**Key UI Features:**

✅ **Real-time feedback:** Display **"Drawing..."** when using the right hand.  
✅ **Brightness Control Display:** Show **brightness percentage (%)** at the top.  
✅ **Smooth UI Overlay:** Blend drawings on a **semi-transparent layer** for clarity.  
✅ **Minimal Distractions:** No extra buttons; gestures control everything.

**Wireframe Representation:**



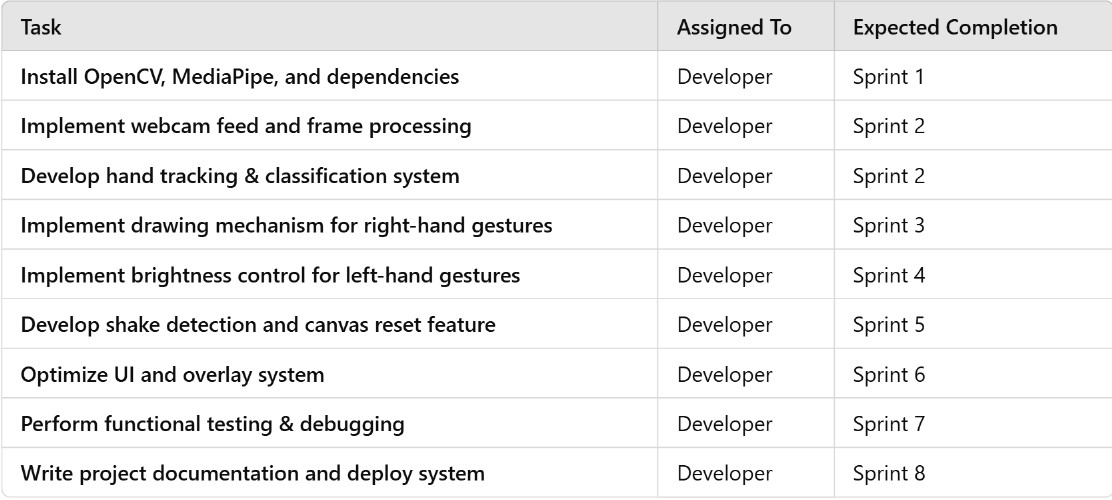
## Phase-4: Project Planning (Agile Methodologies)

**Objective:**

● **Break down the tasks** into manageable units using **Agile methodologies**.



* **Assign tasks** to team members for efficient collaboration.



**Key Points:**

* **Sprint Planning:**
* **Research & Data Collection**: Gather datasets and explore different gesture recognition techniques.
* **Model Development**: Implement OpenCV & MediaPipe for hand tracking and gesture recognition.
* **Integration with Palm's text-bison-001**: Connect recognized gestures to text-based commands.
* **User Interface & Interaction**: Develop a frontend to display recognized gestures and system responses.
* **Testing & Debugging**: Evaluate system performance, accuracy, and real-time responsiveness.
* **Task Allocation:**
* **Jyothrma**i: Implement OpenCV & MediaPipe for gesture tracking, integrate with the frontend.
* **Sahithya**: Work on training & optimizing the model, improving recognition accuracy.Handle Palm's text-bison-001 integration for text-based interactions.
* **Varsha**: Develop a user-friendly UI and implement real-time feedback.
* **Navyasri**: Conduct testing, bug fixing, and ensure smooth system performance.
* **Timeline & Milestones:**

| **Task** | **Deadline** | **Milestone** |
| --- | --- | --- |
| Research & Data Collection | 1 Week | Finalize gesture dataset & framework setup |
| Model Development | 2 Weeks | Working prototype with basic gesture recognition |
| Integration with Palm's API | 1 Week | Text-based interaction working with gestures |
| UI Development | 1 Week | Functional UI displaying recognized gestures |
| Testing & Debugging | 1 Week | Refined system with improved accuracy |

## Phase-5: Project Development

**Objective:**

Implement the core functionalities of the Gesture-Based Human-Computer Interaction System.

Integrate OpenCV for video capture, MediaPipe for hand tracking, and system controls for brightness adjustment and drawing.

Debug and refine the project to ensure smooth performance

**Key Points:**

1. **Technology Stack Used:**

* **Programming Language:** Python
* **Libraries & Tools:**
* **OpenCV:** Video capture and image processing
* **MediaPipe:** Hand tracking and gesture recognition
* **NumPy:** Mathematical operations for hand movement detection
* **Screen-Brightness-Control (sbc):** Adjusting brightness based on gestures
* **Deque (collections module):** Storing recent hand positions for shake detection
* **OpenCV Drawing Tools:** Implementing the virtual drawing feature

1. **Development Process:**

**1.Setting Up the Development Environment:**

* Installed required libraries (OpenCV, MediaPipe, NumPy, etc.).
* Tested webcam functionality to ensure proper video capture.

**2.Implementing Hand Detection:**

* Used MediaPipe’s Hand module to track hand landmarks.
* Converted video frames to RGB for processing.
* Extracted thumb, index finger, and wrist positions for gesture detection.

**3.Implementing Gesture Recognition & Actions:**

* **Drawing Gesture (Right Hand):**
* Tracked index finger movement and drew lines on a virtual canvas using cv2.line().
* **Brightness Control (Left Hand):**
* Calculated the distance between thumb and index finger.
* Adjusted screen brightness dynamically using sbc.set\_brightness().
* **Shake-to-Clear Gesture:**
* Stored recent wrist positions in a deque.
* Cleared the canvas if the total movement exceeded a predefined threshold.

**4.Combining UI Elements:**

* Merged live video feed and virtual drawing canvas using cv2.addWeighted().
* Displayed gesture statuses (e.g., "Drawing...", "Brightness: 70%") using cv2.putText().

**5.Testing & Debugging:**

* Tested under different lighting conditions to improve gesture responsiveness.
* Adjusted shake sensitivity for accurate canvas clearing.
* Optimized real-time performance to reduce lag during continuous drawing.

3.**Challenges & Fixes:**

* **Hand tracking accuracy was low in dim lighting:** Fixed by improving lighting conditions and using a high-resolution webcam.
* **Brightness control was too sensitive:** Adjusted the range of np.interp() for smoother transitions.
* **Shake detection triggered unintentionally:** Increased the deque size and movement threshold.
* **Canvas and video feed misalignment:** Fixed using cv2.flip().
* **Performance lag during drawing:** Optimized cv2.line() rendering and limited frame processing.

## Phase-6: Functional & Performance Testing

**Objective:**

● Ensure the project works as expected.

**Key Points:**

1. **Test Cases Executed:**

* Hand Detection Accuracy
* Drawing Functionality
* Brightness Control
* Shake Detection & Canvas Clearing
* Performance & Responsiveness

**Bug Fixes & Improvements:**

* Adjusted confidence threshold for better hand detection.
* Refined interpolation for smooth brightness control.
* Increased shake detection threshold to avoid accidental clearing.
* Optimized frame processing to reduce lag.

2.**Final Validation:**

* Ensured all features work as expected.
* Verified real-time interaction and accuracy.
* Conducted user testing for usability.

3.**Deployment (if applicable):** **https://drive.google.com/file/d/1YvYrxuFiDuw6k1jeDzMN21gIqmKOSzws/view?usp=drive\_link**

## Final Submission

1. **Project Report Based on the templates-https://gamma.app/docs/3uldoenvjkw26fa**
2. **Demo Video (3-5 Minutes) -https://drive.google.com/file/d/1YvYrxuFiDuw6k1jeDzMN21gIqmKOSzws/view?usp=drive\_link**
3. **GitHub/Code Repository Link - https://github.com/sahithyasahii/AI-Hackthon-2025**
4. **Presentation - https://gamma.app/docs/3uldoenvjkw26fa**