HackathonProjectPhasesTemplate

ProjectTitle:

Gesture-Based Human-Computer Interaction System

TeamName:

GestureX

Team Members:

- Soha Nabi
- Ruchita Kommagani
- Reguri Sarayu
- Musku Bhuvana Reddy

Phase-1: Brainstorming &Ideation

Objective:

Develop a **gesture-based Human-Computer Interaction (HCI) system** that enhances public kiosk interactions by enabling **touchless navigation** at airports, museums, malls, and other high-traffic areas.

KeyPoints:

1. **ProblemStatement:**

- Traditional touchscreen kiosks require physical contact, which can spread germs and pose hygiene concerns.
- Differently-abled individuals may face challenges using conventional touchscreen interfaces.
- Users need a faster, more intuitive, and accessible way to interact with public kiosk

2. **ProposedSolution:**

A touchless, AI-powered HCI system that enables:

- **Intuitive hand gestures** like pointing, swiping, and fist-clenching for seamless navigation.
- **AI-powered predictive gestures** to anticipate user intent and enhance responsiveness.
- **Multilingual gesture recognition**, adapting to cultural and regional variations in gestures
- **Real-time feedback** through visual and voice prompts for an interactive user experience.
- **Customizable gestures** to support accessibility for differently-abled users.

3. **TargetUsers:**

- Airport & Transit Passengers – Quickly access flight/train details without touching screens.
- Museum & Exhibition Visitors – Explore digital exhibits through interactive gestures
- Mall & Retail Shoppers – Browse store directories and offers hands-free.
- Differently-Abled Users – Benefit from an accessible, touch-free interface.
- Healthcare & Public Spaces – Minimize physical contact in high-traffic areas.

4. **ExpectedOutcome:**

- Reduces physical contact, improving **hygiene and safety**.
- Enhances **accessibility and inclusivity** with customizable gestures.
- Delivers **fast, responsive, and intelligent** touchless interactions..

Phase-2: Requirement Analysis

KeyPoints:

1. **TechnicalRequirements:**

- Programming Language: Python, JavaScript
- Backend: AI-powered Gesture Recognition Model (MediaPipe, OpenCV, or TensorFlow)
- Frontend: React.js or Streamlit for interactive UI
- Hardware: Depth-sensing camera

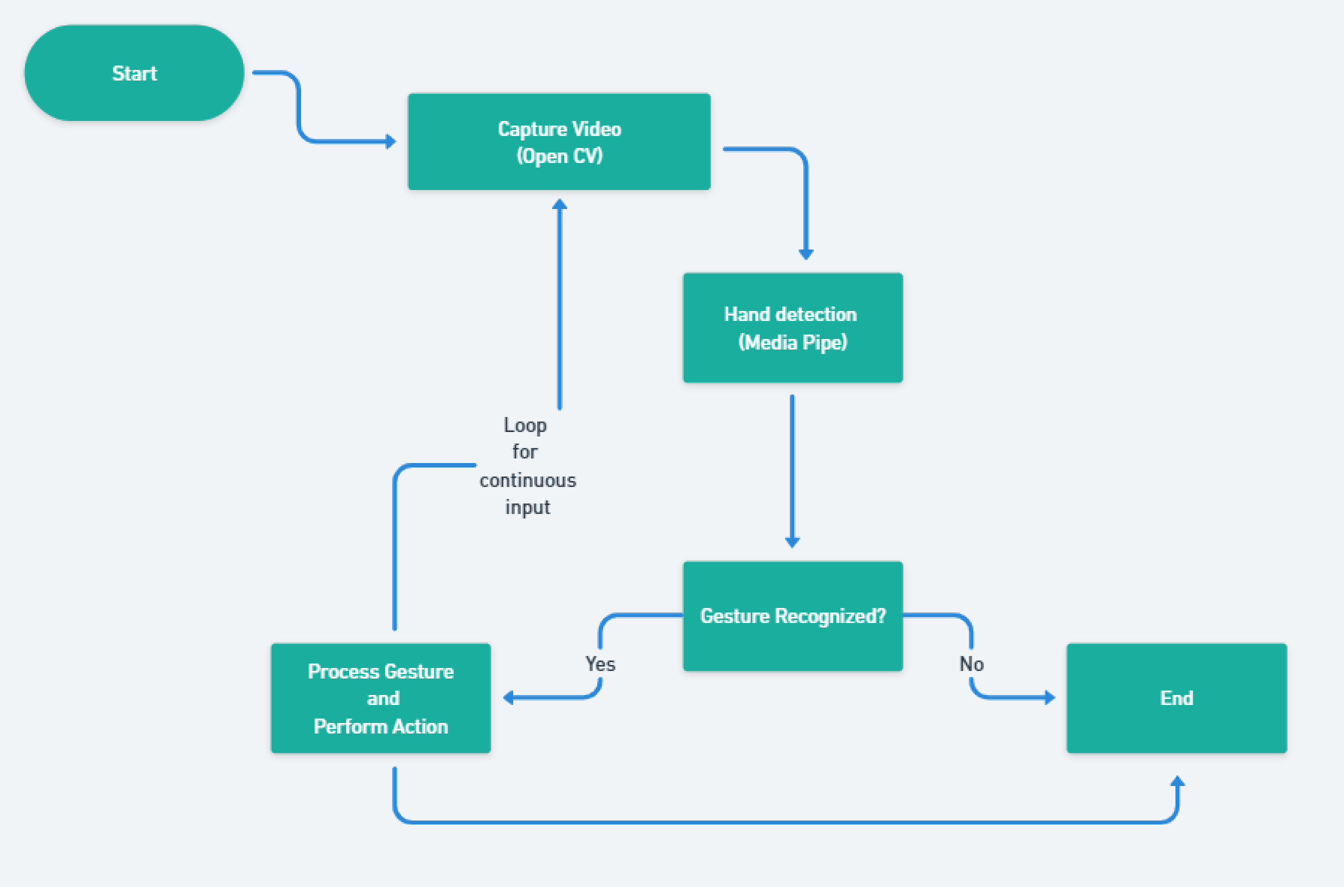
2. **Requirements:**

- Hand Gesture Recognition: Detect gestures like pointing, swiping, and fist-clenching.
- Touchless Navigation: Enable seamless browsing and selection of options.
- AI-powered Predictive Gestures: Anticipate user intent for faster interaction.
- Multilingual Gesture Support: Adapt to cultural variations in gestures.
- Accessibility Features: Customizable gestures for differently-abled users.
- Real-Time Feedback: Provide visual and voice-based response cues.

3. **Constraints&Challenges:**

- **Ensuring high gesture recognition accuracy in different lighting conditions.**
- **Optimizing processing speed for real-time interaction.**
- **Handling different user gestures and variations across demographics.**
- **Integrating AI-driven gesture prediction without latency issues.**

Phase-3: ProjectDesign



KeyPoints:

1. **SystemArchitecture:**

◈ **User Interaction:** User performs hand gestures (e.g., swipe, point, pinch) in front of a camera.

◈ **Gesture Recognition:** The system captures the hand movement using a **depth-sensing camera** (Intel RealSense/Leap Motion).

◈ **AI Processing:**

- AI model (built using **MediaPipe, OpenCV, or TensorFlow**) detects and interprets gestures.
- Recognized gestures are mapped to specific UI actions (e.g., scrolling, selecting).

◈ **Backend Processing:** The system translates gestures into commands and interacts with the UI elements.

◈ **Frontend Display:** The application provides real-time feedback via **visual indicators or voice prompts**.

2. **UserFlow:**

☑ **Step 1:** User approaches the kiosk/screen and positions their hand in front of the camera.
☑ **Step 2:** The system detects the hand and prompts interaction options.
☑ **Step 3:** User performs gestures (e.g., swipe to browse, pinch to zoom, fist-clench to select).
☑ **Step 4:** AI processes the gestures and executes the corresponding action.
☑ **Step 5:** The system provides real-time feedback (highlighting selections, confirming actions).

✅**Step 6:** The interaction concludes when the user confirms or exits the interface.

3. **UI/UXConsiderations:**

🎨**Intuitive & Responsive Design:**
✓ Large icons and text for better visibility.
✓ Simple gesture-based menu navigation.
✓ Touchless controls with **animated gesture guidance**.

🌍**Accessibility & Inclusivity:**
✓ Customizable gestures for users with mobility limitations.
✓ Multilingual gesture interpretation for different regions.

🎭**Visual & Audio Feedback:**
✓ Highlighting UI elements as gestures are recognized.
✓ Voice prompts for additional assistance

Phase-4: Project Planning(Agile Methodologies)

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint	Environment	High	6 hours	End of	Member 1	Google API Key,	API connection

1	Setup & API Integration		(Day 1)	Day 1		Python, Streamlit setup	established & working
Sprint 1	Frontend UI Development	Medium	2 hours (Day 1)	End of Day 1	Member 2	API response format finalized	Basic UI with input fields
Sprint 2	Vehicle Search & Comparison	High	3 hours (Day 2)	Mid-Day 2	Member 1 & 2	API response, UI elements ready	Search functionality with filters
Sprint 2	Error Handling & Debugging	High	1.5 hours (Day 2)	Mid-Day 2	Member 1 & 4	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	Medium	1.5 hours (Day 2)	Mid-Day 2	Member 2 & 3	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

SprintPlanningwithPriorities

Sprint 1 – Setup & Integration (Day 1)

- (● High Priority) Set up the development environment & install dependencies.
- (● High Priority) Integrate gesture recognition model & connect with camera input.
- (□ Medium Priority) Develop a basic UI for gesture-based interactions.

Sprint2–CoreFeatures&Debugging(Day2)

- (● High Priority) Implement gesture recognition for navigation (swipe, select, zoom).
- (● High Priority) Debug recognition issues & optimize response time..

Sprint3–Testing,Enhancements&Submission(Day2)

- (□ Medium Priority) Test gesture accuracy, refine UI, & improve responsiveness.
- (□ Low Priority) Final demo preparation & documentation for submission.

Phase-5: Project Development

KeyPoints:

1. **TechnologyStackUsed:**
    - Frontend: Tkinter/PyQt (for local interfaces) or Web (React/Streamlit)
    - Backend: Python with MediaPipe, OpenCV, TensorFlow for gesture recognition
    - Programming Language: Python
    - Hardware: Depth-sensing camera (Leap Motion, Intel RealSense, or Webcam)

2. **DevelopmentProcess:**

   ○ **Gesture Recognition Module:**
   - Train AI model to detect and classify gestures (e.g., swipe, pinch, fist-clench).
   - Use MediaPipe Hand Tracking for real-time gesture analysis.

   ○ **Gesture-to-Command Mapping:**
   - Define gesture actions (e.g., swiping = scrolling, pointing = selecting).
   - Implement mappings using OpenCV and NumPy.

   ○ **UI & Interaction Layer:**
   - Develop an intuitive touchless interface for kiosks.
   - Integrate real-time feedback (highlighting elements on detection).

   ○ **Performance Optimization:**
   - Optimize gesture detection with frame skipping to reduce lag.
   - Use multi-threading for parallel processing of camera input.

3. **Challenges&Fixes:**

- ● **Challenge:** High latency in recognizing gestures.
  ☑ **Fix:** Reduce frame processing rate and use efficient models like **MobileNetV2**.
- ● **Challenge:** False positive detections.
  ☑ **Fix:** Implement **gesture confirmation logic** (e.g., hold gesture for 1 sec).
- ● **Challenge:** Difficulty recognizing gestures in low light.
  ☑ **Fix:** Adjust camera contrast and integrate **infrared sensing** for improved detection.

Phase-6: Functional & Performance Testing

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	**Functional Testing**	**Query "Best budget cars under ₹10 lakh"**	**Relevant budget cars should be displayed.**	☑ **Passed**	**Tester 1**
TC-002	**Functional Testing**	**Query "Motorcycle maintenance tips for**	**Seasonal tips should be provided.**	☑ **Passed**	**Tester 2**

		winter"			
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	❌ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	🚀 Deployed	DevOps

Final Submission

1. **ProjectReportBasedonthetemplates**
2. **DemoVideo(3-5Minutes)**
3. **GitHub/CodeRepositoryLink**
4. **Presentation**