**Hackathon Project Phases Template** for the **Gesture-Based Human Interaction system** project.

Hackathon Project Phases Template

# Project Title:

**Gesture-Based Human Interaction System**

# Team Name:

Inno-Vibe

# Team Members:

* Harshath
* Yashwanth
* Shiva
* Bhavana
* Sandhya

# Phase-1: Brainstorming & Ideation

## Objective:

Gesture-based Human-Computer Interaction (HCI) is an emerging field that enables users to communicate with computers using hand gestures instead of traditional input devices like a keyboard or mouse. This system leverages **OpenCV** for image processing, **Media Pipe** for real-time hand tracking, and **Palm's Text-Bison-001** for natural language understanding and command execution.

## Key Points:

1. **Problem Statement:**
   * Traditional human-computer interaction methods, such as keyboards and touchscreens, can be inefficient or inaccessible for users with disabilities.
   * There is a growing need for more intuitive, contactless, and natural ways to interact with digital systems.
2. **Proposed Solution:**
   * Develop a gesture-based interaction system using computer vision and machine learning to recognize hand movements and translate them into commands.
   * Integrate the system with various applications, such as smart homes, virtual reality, and assistive technologies, to enhance user experience.
3. **Target Users:**
   * Individuals with physical disabilities who face challenges using conventional input devices.
   * General users seeking a more intuitive and hands-free interaction with digital devices.
4. **Expected Outcome:**
   * A seamless, user-friendly gesture recognition system that improves accessibility and interaction efficiency in digital environments

# Phase-2: Requirement Analysis

## Objective:

Define the technical and functional requirements for the gesture-based Human-Computer Interaction System

## Key Points:

1. **Technical Requirements:**

**Hardware Requirements:**

* + **Camera/Sensors:** High-resolution RGB camera or depth sensors (e.g., Kinect, LiDAR) for gesture recognition.
  + **Processor:** High-performance CPU/GPU for real-time processing (e.g., Intel i7/i9, NVIDIA RTX).
  + **Memory:** Minimum 8GB RAM (16GB recommended for smooth processing).
  + **Input Devices:** Optional support for touchscreens, microphones, or wearables (e.g., smart gloves).

**Software Requirements:**

* + **Operating System:** Compatible with Windows, Linux, or macOS.
  + **Programming Languages:** Python, C++, or Java for system development.
  + **Machine Learning Frameworks:** TensorFlow, OpenCV, or MediaPipe for gesture recognition.
  + **User Interface (UI):** Developed using HTML, CSS, JavaScript (for web-based applications) or PyQt, Tkinter (for desktop applications).

1. **Functional Requirements:**

 **Gesture Recognition:** Detect and interpret predefined hand gestures.

 **Voice Control (Optional):** Integrate speech recognition for multimodal interaction.

 **Real-time Processing:** System must respond to user inputs within milliseconds.

 **Customization:** Allow users to define and train their own gestures.

 **Error Handling:** Provide feedback for unrecognized or ambiguous gesturesAbility to **fetch vehicle details** using Gemini Flash API.

1. **Constraints & Challenges:**

**Constraints:**

* + **Hardware Limitations:** Requires high-performance cameras and processors for accurate real-time recognition.
  + **Environmental Conditions:** Poor lighting, background noise, or cluttered backgrounds can affect gesture detection accuracy.
  + **User Variability:** Differences in hand size, skin tone, and movement speed may impact recognition consistency.
  + **Limited Gesture Set:** Defining a universal set of gestures that work across different applications and user preferences can be challenging.
  + **Computational Requirements:** High computational power needed for machine learning-based gesture recognition models.

**Challenges:**

* + **Accuracy & Precision:** Ensuring high recognition accuracy while minimizing false positives and negatives.
  + **Latency Issues:** Real-time processing must be optimized to avoid lag in gesture response.
  + **User Adaptability:** Some users may find gesture-based interactions unnatural or difficult to learn.
  + **Security & Privacy:** Preventing unauthorized access and ensuring that the system does not collect sensitive user data.
  + **Integration with Existing Systems:** Compatibility with different operating systems, software, and hardware configurations.

# Phase-3: Project Design

## Objective:

## Key Points:

1. **System Architecture:**
   * The system follows a modular architecture with multiple layers for processing user gestures and responding to commands.

**Input Layer (Data Acquisition)**

* + Captures real-time user gestures via **camera** or **sensor** (e.g., RGB camera, depth sensor, Leap Motion).
  + Preprocesses images to filter noise and improve recognition.

**Processing Layer (Gesture Recognition & Interpretation)**

* + **Feature Extraction:** Identifies keypoints (fingers, palm, hand position) using **OpenCV**, **MediaPipe**, or **TensorFlow**.

**Machine Learning Model:** Classifies gestures using a trained **Neural Network (CNN, LSTM)** or traditional computer vision techniques.

* + **Decision Module:** Maps recognized gestures to corresponding actions (e.g., swiping, clicking, zooming).

**Application Layer (User Interface & System Interaction)**

* + Converts gesture inputs into system commands (e.g., controlling a smart device, navigating a UI).
  + Provides feedback via **visual cues, sound, or haptic feedback** to enhance user experience.
  + Allows **gesture customization** where users can train and define their own gestures.

**Integration Layer (API & Communication Layer)**

* + Connects with third-party applications, **IoT devices, AR/VR environments, or smart home systems** via APIs.

1. **User Flow:**

**Step 1: User Initialization**

* + The user **launches the application** on a desktop, mobile, or AR/VR device.
  + A **calibration step** may be required to adjust camera/sensor settings for optimal recognition.

**Step 2: Gesture Input & Processing**

* + The user **performs a gesture** in front of the camera/sensor.
  + The system **captures the gesture**, processes the image, and extracts key features.
  + The **AI model classifies** the gesture and determines the corresponding action.

**Step 3: System Response & Feedback**

* + The application **executes the mapped action** (e.g., opening an app, scrolling, controlling a device).
  + The system provides **visual/audio/haptic feedback** to confirm recognition.

**Step 4: Gesture Customization (Optional)**

* + Users can **define and train** custom gestures using a built-in training module.
  + The system updates its gesture database and refines recognition accuracy over time.

**Step 5: Continuous Learning & Adaptation**

* + The system **improves accuracy** through adaptive learning based on user behavior.
  + Periodic **updates** enhance recognition capabilities and introduce new features.

**Simplified diagram:**

* + **📷 Camera/Sensor → 🎯 Feature Extraction → 🤖 AI-Based Gesture Recognition → 🔄 Command Mapping → 💻 Application Response**

# 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 | Harshath | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡  Medium | 2 hours  (Day 1) | End of Day 1 | Bhavana | API response format finalized | Basic UI with input fields |
| Sprint 2 | Vehicle Search & Comparison | 🔴 High | 3 hours  (Day 2) | Mid-Day 2 | Shiva & Yashwanth | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours  (Day 2) | Mid-Day 2 | Sandhya | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡  Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Bhavana & Harshath | API response, UI layout completed | Responsive UI, better user experience |
| Sprint 3 | Final Presentation & Deployment | 🟢 Low | 1 hour  (Day 2) | End of Day 2 | Harshath  Yashwanth  Bhavana  Shiva  Sandhya | Working prototype | Demo-ready project |

## Sprint Planning with Priorities

**Sprint 1 – Setup & Integration (Day 1)**

**(**🔴 **High Priority)** Set up the **environment** & requirement gathering

**(**🔴 **High Priority)** Integrate **Google studio, lot**

**(**🟡 **Medium Priority)** Build a **basic source code 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. Ensure **real-time responsiveness** with minimal latency.

**(**🟢 **Low Priority)** Final **demo preparation & deployment**

# Phase-5: Project Development

## Objective:

Implement core features of theGesture-Based Human-Computer Interaction System

|  |  |
| --- | --- |
|  |  |

## Key Points:

1. **Technology Stack Used:**

**Frontend & User Interface (UI):**

* + Build a visual feedback system using Streamlit, Flask Web UI, or ReactJS.
  + Display detected gestures, interpreted commands, and system responses.

Streamlit

**Backend & API Development:**

Create a Flask/Fast API-based backend for processing gestures and triggering commands.

* + Integrate cloud services (e.g., Google Cloud AI) for enhanced NLP capabilities.
  + Google Gemini Flash API

**Programming Language:** Python

1. **Development Process:**

**Hand Tracking & Gesture Recognition:**

Use MediaPipe for detecting hand landmarks and recognizing gestures.

* + Implement OpenCV for real-time video capture and image preprocessing.

**NLP & AI Integration:**

* + Connect Palm’s Text-Bison-001 for gesture-to-text conversion and intelligent command execution.
  + Develop a gesture-to-action mapping mechanism for different applications.
  + Implement API key authentication and Gemini API integration.
  + Develop vehicle comparison and maintenance tips logic.
  + Optimize search queries for performance and relevance.

1. **Challenges & Fixes:**

**Challenge:**

Delayed API response times.

**Fix:** Implement **caching** to store frequently queried results.

Deploy on local systems, edge devices, or cloud platforms.

Integrate with IoT, smart home systems, and virtual assistance

# Phase-6: Functional & Performance Testing

## Objective:

Ensure that the **Gesture-Based Human Interaction System** works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Query "Hand gesture 1” | Relevant budget cars should be displayed. | ✅ Passed | Bhavana & Shiva |
| TC-002 | Functional Testing | Query "Hand gesture 2" | Seasonal tips should be provided. | ✅ Passed | Harshath |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | Yashwanth &Sandhya |
| 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 |

**Conclusion**

The project scope covers end-to-end development, from gesture recognition to AI-driven command execution, ensuring a robust, scalable, and real-world deployable system. It has broad applications in accessibility, automation, and interactive computing, making human-computer interaction more intuitive and natural.

# Final Submission

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