GenAI Hackathon Project Phases

Project Title:

Gesture-Based Human Interaction System Using OpenCV, MediaPipe, and Palm's Text-Bison-001

Team Name:

Neural Ninjas

Team Members:

- Sudarshan
- Rajan
- Bhushan
- Prabhat

Phase-1: Brainstorming & Ideation

Objective:

Develop a gesture-based human interaction system leveraging OpenCV, MediaPipe, and Palm's text-bison-001 to facilitate touchless communication between users and machines.

Key Points:

1. **Problem Statement:**

   Many people face challenges with traditional input systems, especially in scenarios where touch is not feasible (hospitals, industrial environments, etc.). Gesture-based systems can simplify human-machine interaction and increase accessibility.

2. **Proposed Solution:**
   - An AI-powered application using OpenCV and MediaPipe for real-time hand gesture recognition**.**

- Utilizing Palm's text-bison-001 API for intelligent text generation and response based on recognized gestures.

3. **Target Users:**

   - **Healthcare professionals** working in sterile environments **.**
   - **Factory workers** requiring touchless device control.
   - **Public kiosk users** who prefer touchless interactions (like at airports, ATMs, etc.)..

4. **Expected Outcome:**

   - A working prototype that detects hand gestures and generates meaningful text output using Palm's text-bison-001/Gemini Flash 1.5, displayed on the interface.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the gesture-based human interaction system.

Key Points:

1. **Technical Requirements:**

   - Programming Language: **Python**
   - Backend: **Google Gemini Flash API/Streamlit/MediaPipe**
   - Frontend: **Streamlit Web Framework**
   - Database: **Not required initially (API-based queries)**

2. **Functional Requirements:**

   - Real-time hand **gesture recognition**.
   - Generate text output based on recognized gestures.
   - Display text output through an **intuitive UI**.
   - Ensure **low-latency processing** for seamless interaction.

3. **Constraints & Challenges:**

   - Ensuring accurate gesture recognition.
   - Optimizing API calls to Palm's text-bison-001.

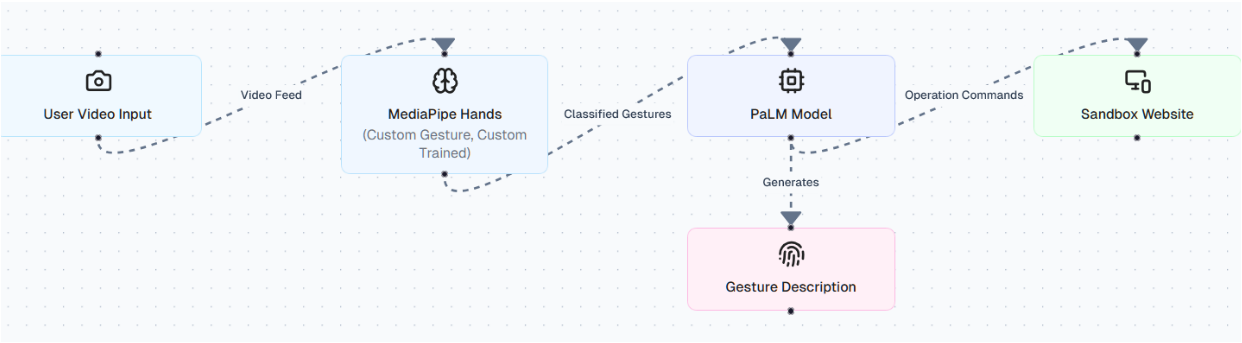
- Maintaining low response time.
- Handling **API rate limits** and optimizing API calls.
- Providing a **smooth UI experience** with Streamlit.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.

Our proposed architecture model



Key Points:

1. **System Architecture:**

    - User performs a hand gesture in front of a webcam.
    - OpenCV captures the video feed.
    - MediaPipe Hands detects and tracks hand landmarks.
    - Gesture classification is based on the motion of landmarks across frames.
    - A buffer stores recent frames to analyze movement patterns.
    - Based on detected movement, a gesture is classified (swipe left, swipe right, select, etc.).
    - The classified gesture is displayed on the screen in real-time.
      .

2. **User Flow:**

    - Step 1: User performs a gesture.
    - Step 2: Backend processes the gesture using MediaPipe Hands.
    - Step 3: Recognized gesture is displayed as text on the interface.**UI/UX**

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	3 hours (Day 1)	End of Day 1	Bhushan	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	Prabhat	API response format finalized	Basic UI with sandbox website to demo
Sprint 2	MediaPipe Hands Setup	🔴 High	3 hours (Day 1)	End of Day 1	Rajan	API response, UI elements ready	Search functionality with filters
Sprint 3	Testing & UI Enhancements	🟡 Medium	3 hours (Day 1)	Mid-Day 2	Sudharshan	Streamlit setup, Python, UI layout	Responsive UI, sandbox website
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	Mid-Day 2	Entire Team	Prototype	Demo-ready documentation

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(🔴 **High Priority)** Set up the development environment (OpenCV, MediaPipe, Flask).
(🔴 **High Priority)** Implement basic hand detection using MediaPipe Hands.
(🟡 **Medium Priority)** Build a basic UI for visualizing detected gestures.

Sprint 2 – Core Features & Debugging (Day 1)

(🔴 **High Priority)** Implement gesture-to-action mapping using MediaPipe classification.
(🔴 **High Priority)** Fix gesture misclassification and improve detection accuracy.

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

(🟡 Medium Priority) Debug gesture inconsistencies and optimize tracking for real-time use.
(🟢 Low Priority) Prepare documentation and presentation, highlighting implementation and challenges.

Phase-5: Project Development

Objective:

Implement core features of the Touchless Kiosk App.

Key Points:

1. **Technology Stack Used:**
   - **Frontend:** Streamlit
   - **Backend:** Google Gemini Flash API
   - **Programming Language:** Python
   -
2. **Development Process:**
   - **Gesture Tracking Implementation**: Started with real-time hand tracking using MediaPipe Hands and OpenCV, ensuring smooth and consistent detection.
   - **Motion Analysis & Classification**: Utilized a frame buffer to analyze hand movements over multiple frames, defining gesture classification logic without relying on LSTMs.
   - **Optimization & Stability**: Adjusted movement thresholds, refined gesture recognition accuracy, and optimized performance to ensure real-time responsiveness.

3. **Challenges & Fixes:**
   - **Inconsistent Data Capture:** Initial attempts to use a dataset for LSTM-based classification led to formatting issues, requiring a shift to a real-time, rule-based approach using MediaPipe alone.

   - **Unstable Gesture Recognition:** Sudden tracking jumps affected accuracy, which was resolved by implementing a frame buffer for smoother gesture transitions.

- **Environmental Variability:** Lighting conditions and hand positioning impacted detection reliability, so we fine-tuned thresholds and improved preprocessing for consistent results.

Phase-6: Functional & Performance Testing

Objective:

Ensure that appropriate gestures are reflected on the sandbox website

Test Case ID	Category	Test Scenario	Expected Outcome	Status
TC-001	Functional Testing	Perform "Swipe Left" gesture	UI should navigate left.	⚠ Works, but Sandbox not implemented
TC-002	Functional Testing	Perform "Swipe Right" gesture	UI should navigate right.	⚠ Works, but Sandbox not implemented
TC-003	Performance Testing	Check real-time gesture response time	Response should be under 500ms.	⚠ Needs Optimization
TC-004	Bug Fixes & Improvements	Fix gesture misclassification issues	Gesture detection should be consistent.	✅ Mostly Consistent gesture detection
TC-005	Final Validation	Ensure UI updates correctly on detected gestures	UI elements should reflect gestures accurately..	❌ Failed - UI sandbox website was not implemented
TC-006	Deployment Testing	Test full system on sandbox website	Gestures should work as expected in real-world tests.	❌ Not Deployed

---