**GenAI Hackathon**

**TITLE:** Gesture-Based Human-Computer Interaction System using OpenCV, MediaPipe and Palm's text-bison-001

**TEAM : POINT BEAKERS**

**Team Members :**

Narendra Bandi (22B81A6625)

V Teja Vigneshwar (22B81A6653)

Y B Sai Prasad(22B81A6637)

J R Vishweshwar Sai(22B81A6633)

S Sriniketh Reddy (22B81A6651)

**Phase-1: Brainstorming & Ideation**

**Problem Statement:**

The Gesture-Based Human-Computer Interaction System leverages real-time hand gesture recognition to enable users to interact with computers through intuitive hand movements. Utilizing computer vision techniques with OpenCV and MediaPipe, the system detects and interprets a variety of gestures, such as thumbs up, fist, open hand, and more. Integrated with a generative AI model, it provides descriptive narratives for recognized gestures, enhancing user experience.

A user-friendly Streamlit interface facilitates easy interaction and visualization, making this system ideal for touchless control, interactive gaming, and assistive technologies.

**Research Existing Solutions :**

**Microsoft Kinect:** Uses depth sensors and computer vision for motion tracking, but is hardware-dependent.

**Leap Motion:** High-precision infrared-based hand tracking, but requires a dedicated sensor.

**Google MediaPipe:** A lightweight and efficient hand tracking solution using only a webcam.

**Gesture-Controlled Smart Devices (e.g., Samsung's Air Gesture):** Limited to predefined gestures and specific devices.

While some of these solutions offer robust gesture recognition, they often require specialized hardware. Our approach aims to provide a software-only, AI-enhanced solution using standard webcams and generative AI to enhance interaction.

**Proposed Solution :**

1. **Hand Gesture Recognition :** Utilizes OpenCV and MediaPipe for real-time hand gesture recognition, recognizes gestures like thumbs up, fist, open hand, and pointing.  
   **Integration with AI Model:** Uses Palm’s text-bison-001 to generate descriptions for recognized .
2. **Touchless Interaction:** Enables intuitive human-computer interaction without physical contact.Reduces germ transmission, ensuring hygiene in public spaces.
3. **Application in Public Kiosks :** Implemented in airports, museums, malls, and other public terminals.Allows users to navigate menus, select options, and retrieve information.
4. **Gesture-Based Controls:**  
    a) Open Hand → Scroll through content.

b) Fist → Select an option.

c) Pointing → Highlight choices.

1. **User-Friendly Interface:** Developed with Streamlit for an interactive and accessible UI.
2. Provides real-time visual feedback for seamless user experience.
3. **Accessibility and Inclusivity:**Facilitates a more inclusive digital interaction environment.

**Target Users :**

The Gesture-Based Human-Computer Interaction System is designed for a broad range of users across different domains. The key target user groups include:

1. Accessibility & Assistive Technology Users

* Physically challenged individuals who struggle with traditional input devices like keyboards and mice.
* People with mobility impairments who can benefit from touchless control.
* Elderly users who may find gesture-based interfaces easier to use than small buttons or touchscreens.

2. Professionals & Remote Workers

* Presenters and educators who want to control slides or presentations using hand gestures.
* Developers and engineers working on human-computer interaction (HCI) innovations.

3. Gamers & Entertainment Industry

* Gamers looking for an immersive experience through motion-based controls.
* VR/AR enthusiasts who need better interaction mechanisms without controllers.
* Content creators who want gesture-based scene control during live streaming.

4. Healthcare & Medical Fields

* Surgeons and doctors who need a touchless interface for viewing medical records in sterile environments.
* Therapists and rehabilitation experts using gesture recognition for motor skill improvement.

5. Public & Industrial Applications

* Retail & Kiosks: Touchless interaction in public places (ATMs, ticket machines, digital signboards).
* Automotive Industry: Gesture controls for in-car navigation and infotainment systems.
* Factories & Warehouses: Hands-free control of machinery and robotic systems.

**Expected Outcome:**

1. Real-Time Gesture Recognition

* Accurate detection and classification of hand gestures using OpenCV and MediaPipe.
* Support for a variety of gestures (e.g., thumbs up, fist, open hand, swipe, etc.).
* Smooth and responsive gesture tracking with minimal latency.

2. AI-Powered Descriptive Feedback

* Integration of a generative AI model to interpret gestures and provide meaningful responses.
* Context-aware descriptions that enhance user engagement.

3. User-Friendly Streamlit Interface

* Interactive dashboard to visualize detected gestures in real time.
* Customization options for gesture-based commands.
* Simple and intuitive design for ease of use.

4. Enhanced Human-Computer Interaction

* Touchless control for accessibility, gaming, smart devices, and automation.
* Potential integration with third-party applications (e.g., smart home systems, presentations, etc.).
* Improved usability for individuals with physical disabilities.

5. Scalability & Future Enhancements

* Modular design, allowing expansion with additional gestures or AI improvements.
* Potential for multi-hand or full-body gesture tracking in future updates.
* Compatibility with various devices (laptops, desktops, IoT systems).

**Phase-2: Requirement Analysis**

**1)Software Requirements**

Programming Language**:** Python

Libraries & Frameworks:

* OpenCV (for real-time video processing)
* MediaPipe (for hand tracking and gesture detection)
* Streamlit (for UI development)
* TensorFlow/PyTorch
* OpenAI API / LLM model

**2)Hardware Requirements**

Computer with at least:

* Processor: Intel i5 or equivalent
* Standard webcam for gesture detection.
* RAM: 8GB or higher (for smooth execution).

**3) Functional Requirements**

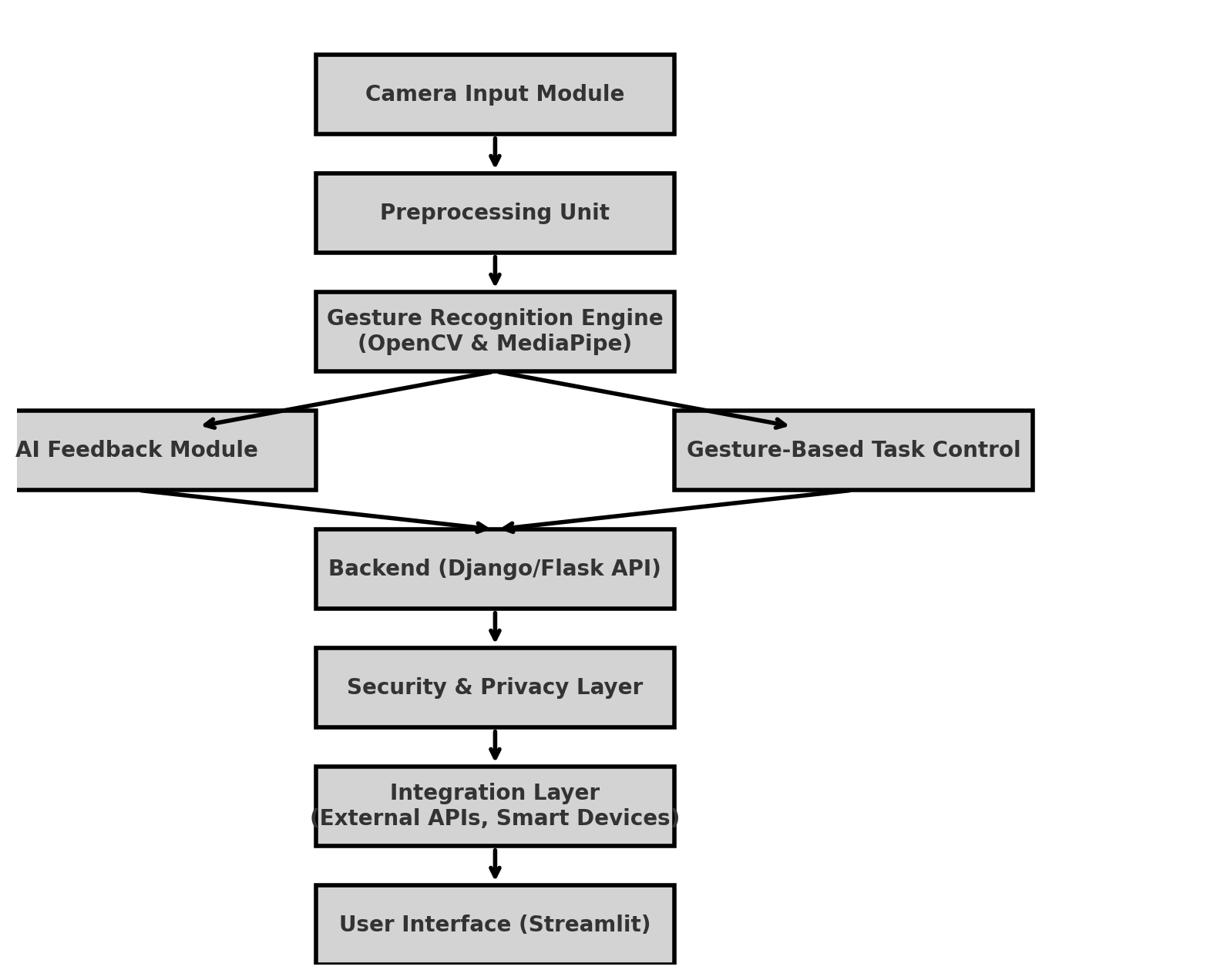
* User Registration & Authentication
* Real-Time Gesture Recognition
* AI-Powered Descriptive Feedback
* User Interface (Streamlit Dashboard)
* Gesture-Based Task Control
* Integration & Compatibility
* Security & Privacy

**4) Performance Requirements**

1. Real-time response time (<100ms delay in gesture recognition).
2. High accuracy in gesture detection (~90%+ under good lighting conditions).
3. Low resource consumption for smooth performance on mid-range devices.

**Phase-3: Project Design**

**Project Design**



**Flow of the Model:**

**User Interface (Streamlit)**

**Security & Privacy Layer**

**Backend (Django/Flask API)**

**AI Descriptive Feedback**

**Gesture-Based Task Control**

**Gesture Recognition (OpenCV & MediaPipe)**

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

**1. Sprint Planning**

* Define product backlog with key features (gesture recognition, AI feedback, UI).
* Break down features into user stories and assign story points.
* Plan 2-week sprints with achievable tasks.

**2. Iterative Development (Sprints)**

Each sprint includes:

* **Sprint Goal:** Implement specific feature(s).
* **Daily Standups:** Short updates on progress and blockers.
* **Continuous Integration & Testing:** Regular testing of gesture recognition, UI, and AI feedback.

**3. Key Sprints Breakdown**

* **Sprint 1:** Set up project, UI prototype, basic gesture recognition.
* **Sprint 2:** Integrate MediaPipe/OpenCV for gesture tracking.
* **Sprint 3:** Implement AI-powered descriptive feedback.
* **Sprint 4:** Enable gesture-based task control & UI improvements.
* **Sprint 5:** Backend integration (Django/Flask), database setup.
* **Sprint 6:** Security enhancements, API support, and performance optimization.
* **Sprint 7:** Testing, debugging, and documentation.
* **Sprint 8:** Final deployment and user feedback.

**4. Continuous Feedback & Improvements**

* Gather feedback after each sprint.
* Make necessary refinements in future iterations.
* Conduct a retrospective to improve the next sprint.

**Phase-5: Project Development**

**Objective:**

Develop and integrate all components of the **Gesture-Based Human-Computer Interaction System**.

**Key Points:**

1. **Technology Stack Used:**
   * Programming: Python (AI & Backend).
   * Frameworks: Django (Backend), Streamlit (Frontend).
   * AI/ML: OpenCV, MediaPipe (Hand Tracking), Generative AI (for descriptive feedback).
2. **Development Process:**
   * **Step 1:** Set up the Django backend with necessary API endpoints.
   * **Step 2:** Implement gesture recognition using OpenCV & MediaPipe.
   * **Step 3:** Develop the Streamlit UI to display real-time webcam feed and detected gestures.
   * **Step 4:** Integrate AI-powered descriptive feedback for recognized gestures.
   * **Step 5:** Implement gesture-based task control (e.g., slide navigation, media control).
   * **Step 6:** Ensure security & privacy by processing video locally and encrypting sensitive data.
   * **Step 7:** Optimize performance and test across different environments.
3. **Challenges & Fixes:**
   * **Latency in real-time gesture recognition** → Optimized model processing and reduced unnecessary computations.
   * **Lighting & background interference affecting detection** → Applied adaptive thresholding and background filtering.
   * **Ensuring AI-generated feedback is context-aware** → Fine-tuned AI model with better prompt engineering.
   * **Seamless communication between frontend and backend** → Used WebSockets for real-time updates.

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

**Objective:**

To ensure the **Gesture-Based Human-Computer Interaction System** functions accurately, performs efficiently, and meets all project requirements.

**Key Points:**

1. **Test Cases Executed:**
   1. **Gesture Recognition Accuracy:** Verify detection of different hand gestures (thumbs up, fist, swipe, open hand, etc.).
   2. **Real-Time Performance:** Measure system response time and latency in gesture recognition.
   3. **AI Feedback Validity:** Test if AI-generated descriptions align with recognized gestures.
   4. **UI Functionality:** Ensure Streamlit interface updates in real-time without delays.
   5. **Backend API Testing:** Validate Django API responses for gesture classification and AI feedback.
   6. **Cross-Platform Compatibility:** Test system on various devices (Windows, Mac, Linux) with different webcams.
2. **Bug Fixes & Improvements:**
   1. Fixed delayed gesture recognition by optimizing MediaPipe model parameters.
   2. Resolved background noise interference with adaptive filtering.
   3. Improved gesture-to-command mapping for more intuitive control.
   4. Optimized AI feedback prompts for better contextual descriptions.
3. **Final Validation:**
   1. Cross-check with initial functional and non-functional requirements.
   2. Ensure security and privacy measures are enforced.
   3. Conduct user testing for feedback on usability and interaction.
4. **Deployment (if applicable):**
   1. Host on a cloud platform (AWS, Google Cloud, or Streamlit Sharing).
   2. Configure API endpoints for scalable access.
   3. Implement monitoring & logging for performance tracking.