
Project Title:

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

Team Name:

Palm’s Visionaries

Team Members:

- K. Sucharitha
- M. Nandini
- G. Sravya
- Y. Nikitha

Phase-1: Brainstorming & Ideation

Objective:

To develop a real-time gesture-based human-computer interaction (HCI) system using OpenCV, MediaPipe, and Palm’s text-bison-001, enabling users to control digital interfaces through intuitive hand gestures, enhancing accessibility, efficiency, and touchless interaction.

Key Points:

1. **Problem Statement:**
    - Many users struggle to find an intuitive and efficient way to interact with computers using touchless gestures.
    - Existing systems lack accuracy and responsiveness in real-time hand gesture recognition.
    - Users need a reliable solution for seamless Human-Computer Interaction (HCI) using OpenCV, MediaPipe, and Palm's **text-bison-001**.

2. **Proposed Solution:**

    - A gesture-based HCI system using **OpenCV, MediaPipe, and Palm's text-bison-001** to enable real-time hand gesture recognition.
    - The system allows users to interact with digital interfaces touchlessly, enhancing accessibility and user experience.
    - It supports various applications, including virtual mouse control, media navigation, and smart device operations.
    - Optimized for high accuracy, low latency, and adaptability to different environments.
3. **Target Users:**

    - **Gamers & VR/AR Users** – Individuals seeking immersive, gesture-based controls for gaming and virtual environments.
    - **Professionals & Creatives** – Designers, artists, and office workers looking for efficient, hands-free interactions.
    - **Smart Home & IoT Users** – People who want to control smart devices with simple hand gestures.
    - **Healthcare & Assistive Tech Industry** – Medical professionals and organizations developing accessibility solutions.
4. **Expected Outcome:**

    - **Touchless Interaction** – Users can control applications like virtual mouse movements, media navigation, and smart device operations without physical contact.

Phase-2: Requirement Analysis

Objective:

Develop a user-friendly and inclusive system for diverse users, including those with mobility impairments.

Key Points:

1. **Technical Requirements:**

    - Programming Language: **Python**
    - Backend: **Node.js with Express.js for OpenCV2 & MediaPipe**
    - Frontend: **Uses react and websockets to receive gesture based commands React.js (for web version).**
    - Database: **Firebase Firestore or PostgreSQL for storing vehicle data and user preferences**
    - Ability to **fetch vehicle details** using Gemini .
    - Firebase Firestore or PostgreSQL for storing vehicle data and user preferences.
    - **Authentication:** Firebase Authentication or OAuth for secure user login.

2. **Constraints & Challenges:**

    - **AI Limitations** – API rate limits and response delays from Gemini Flash.
    - **Cross-Platform Compatibility** – Smooth performance on iOS & Android.
      UI/UX Optimization – Ensuring a user-friendly experience.

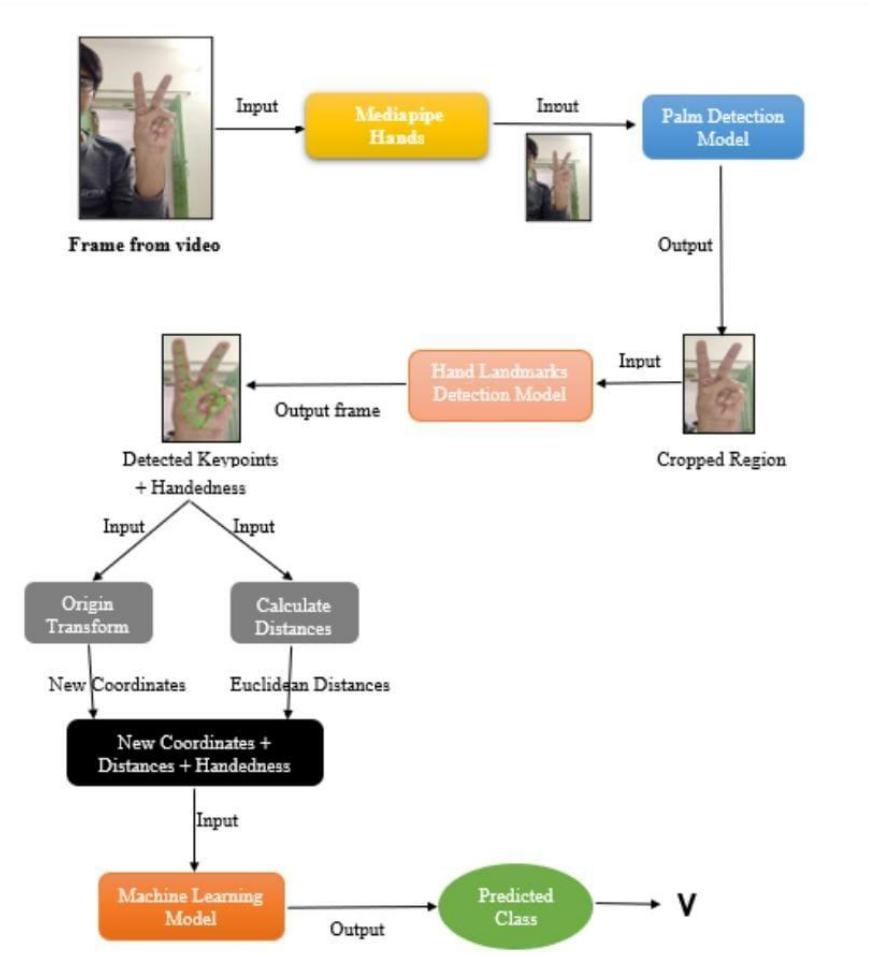
Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.

Key Points:

1. **System Architecture:**



- **User Authentication** – Secure login using Firebase Auth or OAuth.
- **Backend Server** – Node.js with Express.js to handle requests.
- **AI Model Processing** – Gemini Flash processes user queries and generates relevant responses.
- **Vehicle Database Integration** – Connects with automotive APIs (e.g., Carfax, RapidAPI) for specifications..

2. **User Flow:**

- **User Onboarding & Login** – Register/Login via **Firebase Auth/OAuth**.
- **Set Preferences** – Select vehicle type, budget, fuel type, etc.
- **Search Query** – Enter vehicle comparison, reviews, or maintenance queries.
- **Backend Processing** – Sends request to **Google Gemini API** & fetches data.
- **AI Data Processing** – Extracts vehicle details, reviews, and eco-friendly insights.
- **Display Results** – Shows comparisons, specifications, and recommendations.
- **User Actions** – Bookmark vehicles, read reviews, explore alternatives.
- **Dealer & Service Locator** – Find showrooms, service centers, charging stations.
- **Push Notifications** – Alerts for price drops, new models, and maintenance.
- **Exit & Re-Engagement** – User leaves but receives alerts to return.

3. **UI/UX Considerations:**

- **Personalized Suggestions** – AI-based insights for user preferences.
- **Easy Bookmarking & History** – Save searches and favorite vehicles.
- **Accessibility & Notifications** – Voice search, real-time alerts, and feedback messages..

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	MediaPipe & OpenCV	🔴 High	6 hours (Day 1)	End of Day 1	Member 1	Google Command prompt, Python	OpenCV2 & Media Pipe installed
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	Member 2	React.js / Next.js	Basic UI with input fields
Sprint 2	Vehicle Search & Comparison	🔴 High	3 hours (Day 2)	Mid-Day 2	Member 1& 2	OpenCV2 & MediaPipe	Search functionality with filters
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	Member 1&4	UI inputs	Imporved OpenCV2 & MediaPipe
Sprint 3	Testing & UI Enhancements	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	Member 2& 3	Hand, thumb movement	Responsive Movement recognition
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	End of Day 2	Entire Team	Gesture Recognition	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(🔴 **High Priority)** Set up the **environment** & install dependencies.
(🔴 **High Priority)** Integrate **Google Gemini**.
(🟡 **Medium Priority)** Build a **basic UI 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 OpenCV2 & MediaPipe, refine UI, & fix UI bugs.
(🟢 **Low Priority)** Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

Develop a **scalable, AI-powered** vehicle insights app integrating **Google Gemini API** for real-time comparisons, recommendations, and eco-friendly guidance while ensuring **cross-platform compatibility, security, and user-friendly experience**.

Key Points:

1. **Technology Stack Used:**
    - **Frontend: React.js\Next.js\Web Cam**
    - **Backend: OpenCV2 & MediaPipe for Hand tracking**
    - **Programming Language:** Python
2. **Development Process:**
    - **Design & Development** – Create UI/UX, build frontend (React Native/Web), and develop backend with **OpenCV2 & MediaPipe**
    - **Testing & Security** – Conduct functional testing, ensure **GDPR & CCPA compliance**, and optimize performance.
    - **Deployment & Updates** – Launch on **cloud servers & app stores**, gather user feedback, and release improvements.
3. **Challenges & Fixes**
    - **Challenge:** Delayed API response times**.**
      Fix: Implement caching to store frequently queried results.
    - **Challenge**: Limited API calls per minute.
      Fix: Optimize queries to fetch only necessary data.

Phase-6: Functional & Performance Testing

Objective:

Ensure all **Gesture recognition** features, including search, AI recommendations, and comparisons, function correctly across devices. Optimize **OpenCV2 & MediaPipe response times, loading speed, and scalability** for a seamless user experience..

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Wave Gesture recogination	Detects wave gesture logs and gesture.wave and sends data over websockets	✅ Passed	Tester 1
TC-002	Functional Testing	Thumbs-Up gesture recognition	The system detects thumbs-Up and logs it	✅ Passed	Tester 2
TC-003	Performance Testing	Fist gesture Detection	“Gesture.first” should be logged and transmitted	✅ Passed	Tester 3
TC-004	Bug Fixes & Improvements	Partial hand visibility	The system recognize the gesture if enough hand landmarks are visible.	✅ Fixed	Developer
TC-005	Final Validation	Speed Sensitivity	Recognize the gesture consistently regardless of speed	Different speeds	Tester 2
TC-006	Deployment Testing	Poor Lighting Conditions	Detects gestures but may show reduced accuracy.	Performed a wave gestures	DevOps

Final Submission

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