

**AI Powered Virtual Calculator using Computer Vision and Generative Model**

**A Minor Project Report**

Submitted in partial fulfillment of requirement of the Degree  
of

**BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE &  
ENGINEERING**

**BY:-**

Punit Pawar – EN22CS303038

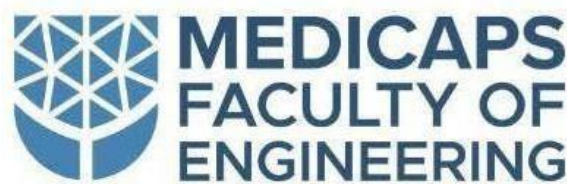
Rounak Pathekar – EN22CS303046

Yashasvi Bakore – EN22CS303059

**Under the Guidance of:**

Dr. Hare Ram Jha

Dr. Manish Korde



**Department of Computer Science & Engineering  
Faculty of Engineering  
MEDICAPS UNIVERSITY, INDORE- 453331**

**Jan-June 2025**

## **REPORT APPROVAL**

The project work “AI POWERED VIRTUAL CALCULATOR USING COMPUTER VISION AND GENERATIVE MODEL” is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approve any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation:

Affiliation:

External Examiner

Name:

Designation

Affiliation

## **DECLARATION**

We hereby declare that the project entitled "**AI Powered Virtual Calculator using Computer Vision and Generative Model**" submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science & Engineering, completed under the supervision of Prof. Hare Ram Jha and Prof. Manish Korde, Faculty of Engineering, Medi-Caps University, Indore is an authentic work.

Further, we declare that the content of this Project work, in full or in parts, has neither been taken from any other source nor been submitted to any other Institute or University for the award of any degree or diploma.

**Punit Pawar (EN22CS303038)**

**Rounak Pathekar (EN22CS303046)**

**Yashasvi Bakore (EN22CS303059)**

## **CERTIFICATE**

This is to certify that the project entitled "**AI Powered Virtual Calculator using Computer Vision and Generative Model**" submitted by **Punit Pawar (EN22CS303038), Rounak Pathekar (EN22CS303046), and Yashasvi Bakore (EN22CS303059)** in partial fulfillment for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** is a record of the work carried out by them under our supervision and that the work has not formed the basis of the award of any other degree elsewhere.

---

**Dr. Hare Ram Jha Sir**

Department of Computer Science & Engineering  
MEDICAPS University, Indore

---

**Dr. Manish Korde Sir**

Department of Computer Science & Engineering  
MEDICAPS University, Indore

---

**Dr. Ratnesh Litoriya Sir**

Head of the Department  
Computer Science & Engineering  
Medi-Caps University, Indore

## **ACKNOWLEDGEMENT**

We would like to express our gratitude to the Honorable Chancellor, Shri R. C. Mittal, and the Vice Chancellor, Prof. (Dr.) D. K. Patnaik, for their invaluable support and encouragement. We also thank Prof. (Dr.) Pramod S. Nair, Dean, Faculty of Engineering, and Dr. Ratnesh Litoriya, Head of the Department of Computer Science & Engineering, for facilitating this project.

Our sincere thanks go to our guides, **Dr. Hare Ram Jha** and **Dr. Manish Korde**, whose expertise and dedication inspired and guided us throughout the development of this project.

We extend our appreciation to our peers and all those who directly or indirectly supported us in completing this project.

**Punit Pawar (EN22CS303038)**

**Rounak Pathekar (EN22CS303046)**

**Yashasvi Bakore (EN22CS303059)**

## **ABSTRACT**

In an era of touchless interaction and AI integration, our project introduces an **AI Powered Virtual Calculator** that leverages **Computer Vision** and **Generative Models** for real-time mathematical computation using hand gestures. This system utilizes **OpenCV** and **MediaPipe** to detect and interpret gestures drawn in the air, transforming them into mathematical expressions.

Captured gestures are rendered onto a virtual canvas and saved as an image, which is then processed by the **Google Gemini API** to extract, understand, and solve the expression. This combination ensures a hands-free, accessible, and intelligent tool ideal for education, accessibility, and smart environments.

The backend is built using **Flask**, while the frontend offers a clean interface for live feedback and results. This work contributes to the field of human-computer interaction (HCI) and assistive AI technologies.

**Keywords:** Hand Gesture Recognition, Computer Vision, OpenCV, MediaPipe, Generative AI, Google Gemini API, Virtual Calculator, Deep Learning, Flask, Human-Computer Interaction.

## **KEYWORDS**

- Computer Vision
- Hand Gesture Recognition
- Generative Model
- Google Gemini API
- Deep learning
- OpenCV
- Mediapipe
- Generative AI
- Flask
- Canvas
- JavaScript
- CSS
- HTML
- Python
- Webcam
- Hand Detection
- Mathematical Computation
- Dotenv
- Human Computer Interaction

## **ABBREVIATIONS**

<b>Abbreviation</b>	<b>Full Form</b>
AI	Artificial Intelligence
CV	Computer Vision
HCI	Human-Computer Interaction
API	Application Programming Interface
GUI	Graphical User Interface
ML	Machine Learning
DL	Deep Learning
IDE	Integrated Development Environment
ROI	Region of Interest
JPG/JPEG	Joint Photographic Experts Group
RGB	Red Green Blue (Color Space)
CSS	Cascading Style Sheets
JS	JavaScript
HTML	HyperText Markup Language
UI	User Interface
SDK	Software Development Kit
JSON	JavaScript Object Notation
API Key	Application Programming Interface Key
Flask	(No abbreviation – it's a Python web framework)
OpenCV	Open Source Computer Vision Library
MP (MediaPipe)	MediaPipe (Google's CV framework)



## **Table of Contents**

<b>Chapters</b>	<b>Contents</b>	<b>Page No.</b>
	Report Approval	i
	Declaration	ii
	Certificate	iii
	Acknowledgement	iv
	Abstract	v
	Keywords	vi
	List of figures	vii
	Abbreviations	viii
Chapter 1	Introduction	1
	1.1 Introduction	1
	1.2 Literature Review	2
	1.3 Objectives	3
	1.4 Significance	3
	1.5 Research Design	4
	1.6 Source of Data	5
Chapter 2	REQUIREMENTS SPECIFICATION	8
	2.1 User Characteristics	8
	2.2 Functional Requirements	8
	2.3 Dependencies	8
	2.4 Performance Requirements	8
	2.5 Hardware Requirements	9
	2.6 Constraints & Assumptions	9
Chapter 3	DESIGN	10
	3.1 Algorithm	10
	3.2 Function Oriented Design for procedural approach	10

	3.3 <b>System Design</b>	11
	3.3.1 Data Flow Diagrams (Level 0, Level1)	11
	3.3.2 Activity Diagram	13
	3.3.3 Flow Chart Diagram	14
	3.3.4 Class Diagram	15
	3.3.5 Sequence diagram	15
Chapter 4	Implementation, Testing, and Maintenance	16
	4.1 Introduction to Languages, IDE's, Tools and Technologies used for Implementation	16
	4.2 Testing Techniques and Test Plans	17
Chapter 5	Results and Discussions	19
	5.1 User interface Representation	19
	5.2 Brief description of various modules of the system	20
	5.3 Snapshots of system with brief details	
	5.4 Back end Representation	23
Chapter 6	Summary and Conclusions	25
Chapter 7	Future scope	26
	Appendix	27
	Reference	28

# CHAPTER 1 : INTRODUCTION

## **1.1 INTRODUCTION:-**

In the fast-evolving landscape of artificial intelligence and human-computer interaction (HCI), new paradigms are emerging that redefine how users interface with technology. One such paradigm is the use of gesture-based interaction, particularly hand gestures, which offers an intuitive and natural way of communicating with machines. Traditional input devices such as keyboards, mice, and even voice commands are giving way to more human-centric, touchless interfaces.

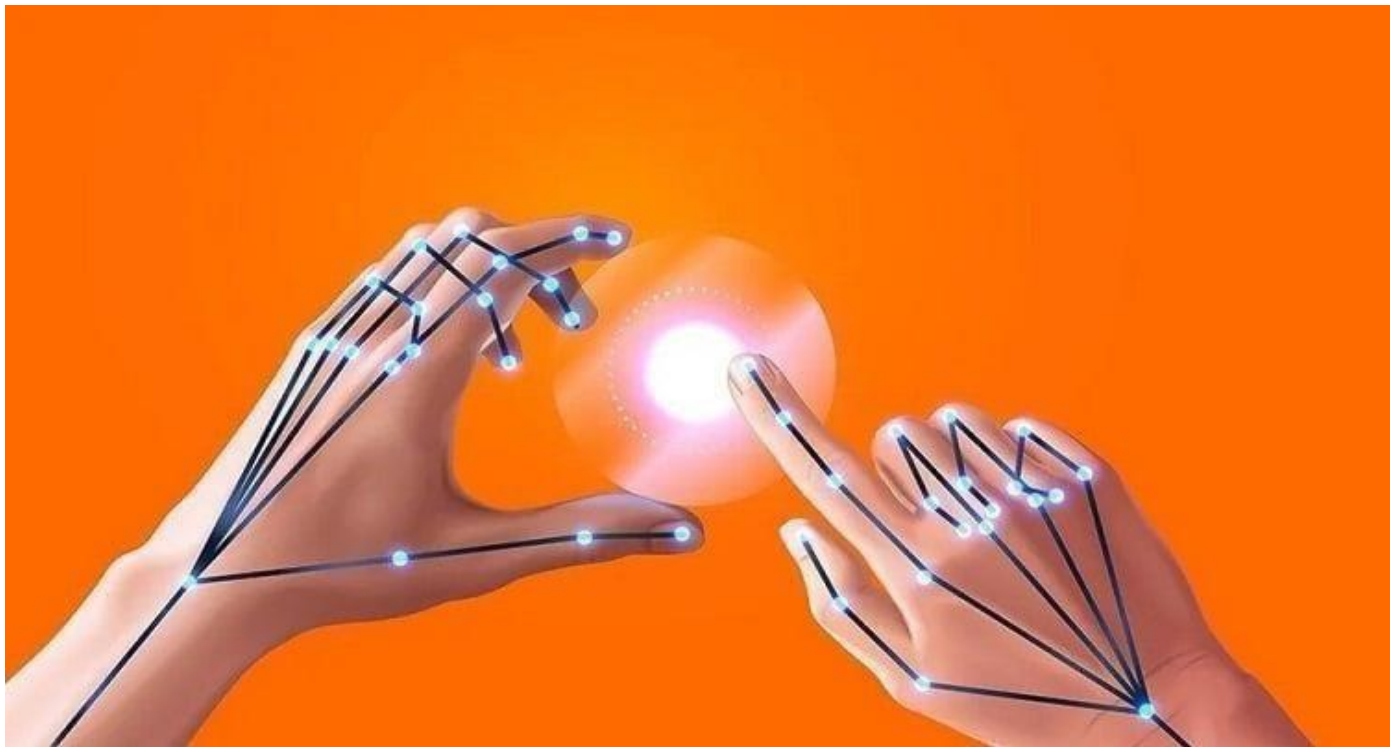
The **AI Powered Virtual Calculator using Computer Vision and Generative Model** aims to integrate computer vision, deep learning, and generative AI to create a seamless, intelligent, and accessible method for performing mathematical calculations. The core idea is to allow users to draw numbers and operations in the air using hand gestures, which are then interpreted as mathematical expressions and solved in real time.

This innovation is particularly significant in contexts where touch-based interaction is limited or undesirable, such as during the COVID-19 pandemic, or for individuals with physical impairments. Moreover, by leveraging AI capabilities through the Google Gemini API, the calculator transcends basic arithmetic to solve advanced mathematical problems, enhancing its utility in educational and technical domains.

The virtual calculator operates through a multi-stage process: gesture detection, drawing interpretation, image capture, and AI-driven solution generation. Hand gestures are recognized using MediaPipe, interpreted through OpenCV, and the canvas output is analyzed by Gemini to extract and solve mathematical expressions. This integration not only enhances user accessibility but also contributes to the broader research fields of computer vision, machine learning, and multimodal interaction design.

In this report, we explore the motivation behind the project, review related research, outline the system design and technical stack, present implementation strategies, and discuss the results and future potential of the system.

The ultimate goal is to demonstrate that a non-contact, AI-enabled calculator can serve as both a functional tool and a foundation for future advancements in smart education, accessible computing, and intelligent interfaces.



*Figure 1.1- Hand Gesture Recognition*

## **1.2 LITERATURE REVIEW:-**

In recent years, significant progress has been made in the fields of computer vision and artificial intelligence, particularly in enhancing human-computer interaction through intuitive and non-contact methods such as gesture recognition. Traditionally, gesture-based systems relied on physical devices like sensor gloves or infrared markers, which were expensive and cumbersome. However, with the advent of advanced computer vision tools like OpenCV and Google's MediaPipe, it has become possible to build gesture recognition systems that operate in real-time using standard webcams. These frameworks allow for precise tracking of hand landmarks and gesture classification without additional hardware, making them ideal for developing accessible and scalable interaction models.

Simultaneously, the emergence of powerful generative AI models such as OpenAI's GPT and Google Gemini has revolutionized the way machines understand and respond to human input. These models can now process multimodal data—including images—and extract meaningful context from handwritten or sketched mathematical expressions. They not only solve the equations but also provide step-by-step reasoning behind their answers, closely resembling how a human would approach the problem.

The integration of gesture recognition and generative AI has opened new possibilities in educational and assistive technologies. Our project builds on these foundational technologies to develop a virtual calculator that allows users to draw equations in the air and receive instant, AI-generated solutions. This novel approach eliminates the need for touchscreens or physical input devices, promoting hygienic, hands-free interaction while showcasing the practical power of real-time computer vision and generative intelligence.

### **1.3 OBJECTIVES:-**

The primary goal of this project is to develop a touchless, AI-powered virtual calculator that recognizes hand gestures and solves mathematical expressions using generative AI. The specific objectives include:

1. To design a system capable of capturing real-time hand gestures using computer vision techniques.
2. To implement a virtual drawing canvas that enables users to draw mathematical symbols or expressions in the air.
3. To utilize MediaPipe for accurate hand landmark detection and gesture classification.
4. To integrate OpenCV for image processing and canvas rendering functionalities.
5. To develop an intuitive, gesture-based interface that eliminates the need for traditional input methods like mouse or keyboard.
6. To capture the drawn expression as an image and send it to the Google Gemini API for interpretation and problem solving.
7. To display AI-generated solutions with logical steps in a user-friendly web interface.
8. To ensure the system operates in real-time, offering quick recognition and response for seamless interaction.
9. To build a lightweight, scalable application using Flask as the backend and HTML/CSS/JS for the frontend.
10. To promote accessibility by creating a hands-free calculator beneficial for educational, assistive, and smart device contexts.

### **1.4 SIGNIFICANCES:-**

The development of an AI Powered Virtual Calculator using Computer Vision and Generative Models holds substantial significance in both technological and societal contexts:

#### **1. Enhanced Human-Computer Interaction (HCI):**

By allowing users to perform calculations through hand gestures, the project exemplifies a natural and intuitive mode of communication between humans and machines, reducing reliance on traditional input devices.

#### **2. Touchless and Hygienic Interface:**

In post-pandemic times, touchless technologies are increasingly preferred in public and educational environments. This project offers a clean, contact-free method for performing computations, ideal for classrooms, kiosks, and healthcare settings.

#### **3. Accessibility and Inclusion:**

The system benefits individuals with physical disabilities or motor limitations who may struggle with using a mouse, keyboard, or touchscreen. It empowers them with an alternative, voice-free, touch-free computing interface.

#### **4. Educational Applications:**

Students can engage in interactive learning by solving math problems with gestures, making education more dynamic and engaging. The use of AI-generated step-by-step solutions aids conceptual understanding.

### 5. **Innovation in Assistive Technology:**

By integrating generative AI and computer vision, this project showcases how modern AI can be repurposed for assistive tools that go beyond conventional applications.

### 6. **Demonstration of Multidisciplinary Integration:**

The project successfully combines skills from machine learning, computer vision, natural language processing, and web development, demonstrating the power of interdisciplinary collaboration.

### 7. **Scalability and Real-World Potential:**

The solution is lightweight and scalable, making it suitable for integration into smart devices, educational tablets, and future HCI systems.

## **1.5 RESEARCH DESIGN:-**

The research design for the project “AI Powered Virtual Calculator using Computer Vision and Generative Model” is structured to guide the systematic development and implementation of a gesture-based, AI-integrated calculator. The design emphasizes the integration of real-time computer vision with generative AI to provide a smooth, interactive, and intelligent user experience.

### 1. Research Type

The project follows an applied research methodology with a focus on technological innovation and practical implementation. It leverages experimental and iterative design to prototype, test, and refine the system.

### 2. Development Approach

The system development was conducted using the Agile methodology, allowing for continuous feedback, modular progress, and iterative improvement. This approach made it easier to integrate complex components like gesture detection and AI response handling in stages.

### 3. System Workflow

The core workflow of the project includes the following stages:

- Real-time hand gesture tracking using MediaPipe and OpenCV
- Rendering gesture-based input onto a virtual canvas
- Capturing and saving the canvas as an image
- Sending the image to Google Gemini API for interpretation and computation
- Receiving and displaying AI-generated solutions via Flask and web interface

### 4. Tools and Technologies Used

- Programming Language: Python
- Vision Libraries: OpenCV, MediaPipe
- Web Framework: Flask
- AI Model: Google Gemini API (Gemini 1.5 Pro)

- Frontend: HTML, CSS, JavaScript
- Environment: Jupyter Notebook, VS Code
- Image Handling: Pillow, NumPy

## 5. Data Collection and Testing

Gesture data was collected via webcam in real-time using MediaPipe's landmark detection. Testing was done in various lighting conditions and backgrounds to ensure system robustness. Expression recognition accuracy and AI output quality were assessed based on clarity and correctness of solutions.

## 6. Validation Techniques

- Manual validation of gesture recognition accuracy
- Output validation by comparing Gemini solutions with expected mathematical results
- Usability testing among peers for interface feedback

## 7. Ethics and Safety Considerations

The project does not store user data or use personally identifiable information. It ensures touchless interaction for hygiene and accessibility. The use of generative AI is limited to solving mathematical expressions for educational purposes.

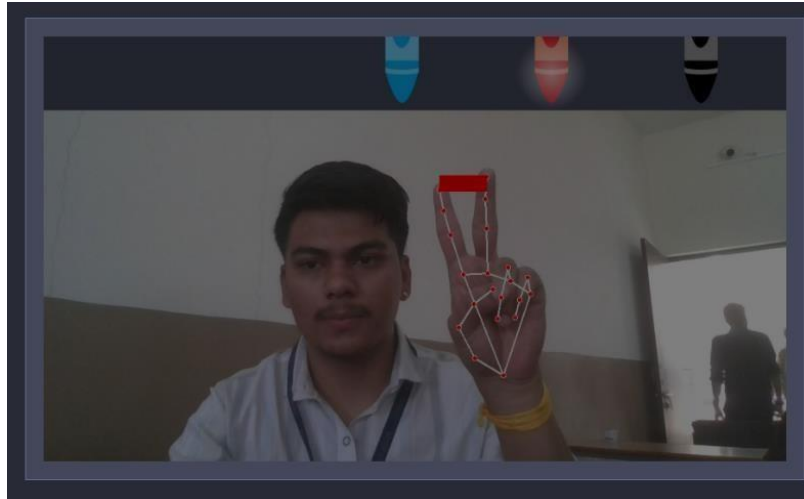
### **1.6 SOURCE OF DATA:-**

The data used in this project is entirely user-generated and captured in real-time through computer vision and gesture tracking components. Unlike conventional projects that rely on static or pre-collected datasets, our system is dynamic and interactive—collecting data on-the-fly during user interaction.

#### **1. Primary Data Source – Webcam Input**

The core data originates from a standard webcam, which continuously captures video frames of the user's hand movements. These frames are processed in real-time using the MediaPipe library to extract hand landmarks, specifically 21 key points that define the shape, orientation, and motion of the hand.

This landmark data enables the system to interpret gestures such as drawing with a single finger, selecting options using two fingers, or pinching to signal capture. The movement of the fingertip across the frame generates trace lines on a virtual canvas, simulating freehand mathematical drawing.

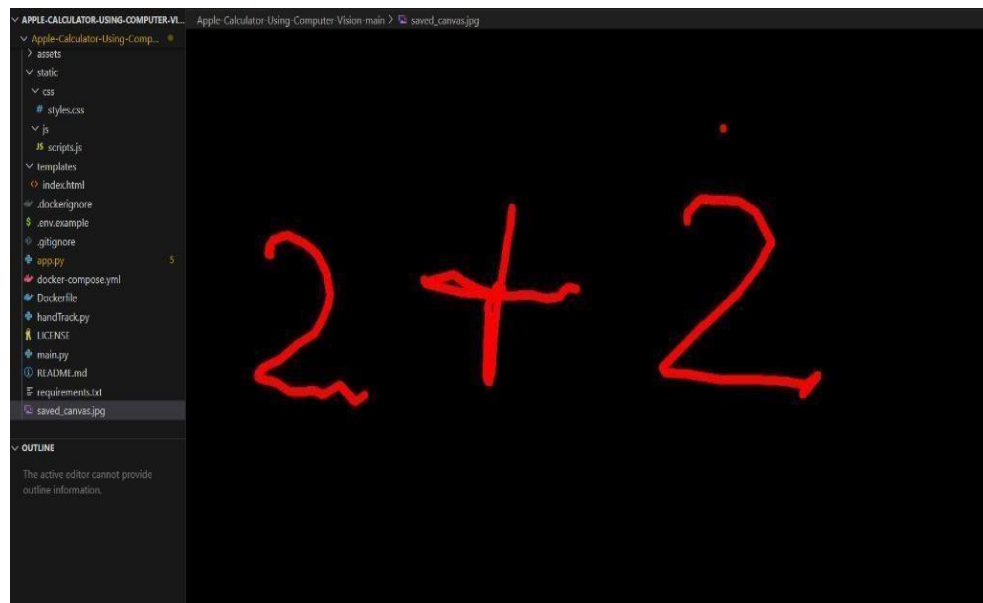


*Figure - 1.2 Webcam Input*

## 2. Virtual Canvas Data

As users draw equations using their hand in the air, the system renders those movements on a virtual whiteboard or canvas using OpenCV. This generated canvas serves as a visual representation of the mathematical expression and is periodically saved as an image file (e.g., saved\_canvas.jpg) when a “pinch” gesture is detected.

These image files represent the secondary form of data—the equation the user intends to solve—captured in a format suitable for image-based processing and inference.



*Figure - 1.3 Virtual Canvas Data*

## 3. Secondary Data Source – Google Gemini API

Once an image of the handwritten or drawn expression is captured, it is sent as input to the Gemini API (part of Google’s Generative AI services). This API acts as an external intelligent data processor. It extracts content from the image using its embedded vision model, interprets the mathematical context, and generates an answer along with step-by-step reasoning. The response is received in JSON/text format and displayed back to the user on the web interface.



#### **4. No Use of Pre-existing Datasets**

This project does not rely on any third-party gesture recognition or handwriting datasets. All data is captured from live user interaction, which ensures authenticity, variability, and real-time adaptability.

#### **5. Data Privacy and Ethics**

All data is handled in memory without being permanently stored or transmitted beyond the API call. No personal images or identity data are saved, making the system privacy-respecting and ethically compliant.

## CHAPTER 2: REQUIREMENT SPECIFICATION

### 2.1 USER CHARACTERISTICS:-

The primary users of this AI-powered virtual calculator are students, educators, and individuals with physical disabilities who benefit from touch-free, intelligent computing systems. Additionally, users with interests in human-computer interaction, smart classrooms, or AI-based education tools will find this application relevant.

Users are expected to have a basic understanding of mathematical symbols and simple hand gesture movements. Although no programming knowledge is required, a familiarity with virtual interfaces and gesture-based devices enhances user experience. The interface has been designed with usability in mind, ensuring that it is accessible to both tech-savvy individuals and novices alike.

### 2.2 FUNCTIONAL REQUIREMENTS:-

- **Gesture Recognition:** Detect and classify hand gestures using the webcam and convert them into drawing strokes.
- **Canvas Interaction:** Display a live drawing canvas that captures user input.
- **Image Capture:** Automatically save the canvas as an image upon completing a gesture or expression.
- **AI Integration:** Use Google Gemini API to interpret and solve the captured mathematical expression.
- **Live Feedback:** Provide real-time video feed and solution results on a web interface.
- **Multimodal Access:** Support future extension to include voice commands or text input.

### 2.3 DEPENDENCIES:-

- **MediaPipe:** For real-time hand tracking.
- **OpenCV:** For image processing and canvas rendering.
- **Flask:** To create the web application.
- **Google Generative AI (Gemini):** For analyzing and solving mathematical expressions.
- **JavaScript, HTML, CSS:** For frontend development.

### 2.4 PERFORMANCE REQUIREMENTS:-

- **Real-Time Gesture Response:** System must process hand movement and render it on canvas with minimal lag.
- **Low Latency AI Response:** The solution retrieval from Gemini API should not exceed 2–3 seconds.
- **Frame Rate:** Video feed should maintain a minimum of 20 FPS for seamless interaction.

- **Accuracy:** Gesture recognition and interpretation should achieve over 90% reliability under good lighting.

## 2.5 **HARDWARE REQUIREMENTS:-**

- **Camera:** Webcam with at least 720p resolution.
- **Processor:** Minimum Intel i5 or equivalent.
- **Memory:** 8 GB RAM or higher recommended.
- **Storage:** Minimum 2 GB free for dependencies and image storage.

## 2.6 **CONSTRAINTS AND ASSUMPTIONS:-**

- Requires a stable internet connection to interact with the Gemini API.
- System performance may vary with lighting conditions and webcam quality.
- AI responses are dependent on the quality of image captured; blurred or unclear drawings may result in incorrect output.
- Users are assumed to gesture clearly within the camera frame.

## CHAPTER 3: DESIGN

### 3.1 ALGORITHM:-

The virtual calculator is driven by a gesture-detection and drawing algorithm that processes video input and maps it to either gesture control or mathematical symbol rendering.

#### High-Level Steps:

1. Capture real-time video using OpenCV.
2. Detect hand landmarks using MediaPipe.
3. Check gesture conditions:
  - Two fingers up → Select mode
  - One finger up → Drawing mode
  - Thumb and index together → Trigger solve
4. In Drawing Mode:
  - Track index finger position.
  - Draw on canvas with detected coordinates.
5. When trigger detected:
  - Save canvas as image.
  - Send to Flask server for AI-based solving.

### 3.2 FUNCTIONAL ORIENTED DESIGN FOR PROCEDURAL APPROACH :-

Below are the key functions:

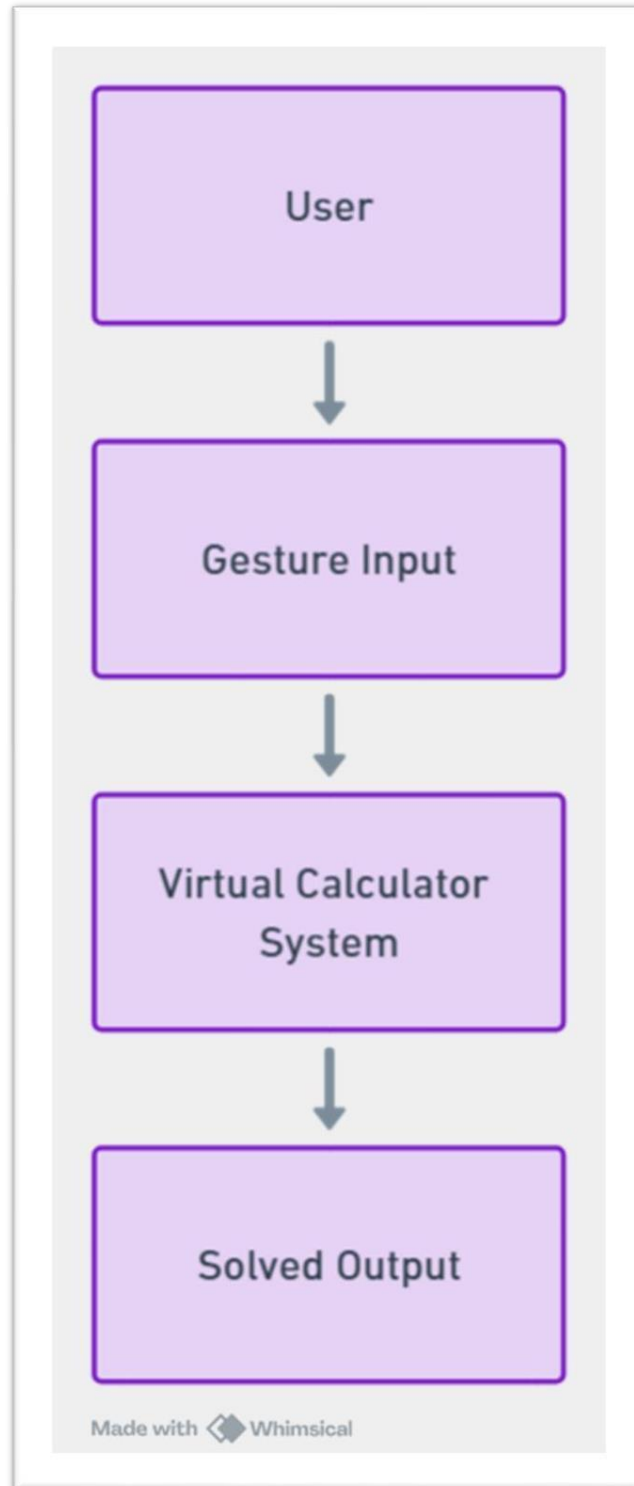
- detectGesture() → Interprets hand landmark positions
- drawOnCanvas() → Draws strokes on virtual whiteboard
- saveCanvas() → Captures current canvas to local storage
- sendToGeminiAPI() → Sends canvas image to Gemini AI
- parseGeminiResponse() → Extracts solution and renders output

Each function handles a single responsibility to follow procedural modularity.

### 3.3 SYSTEM DESIGN :-

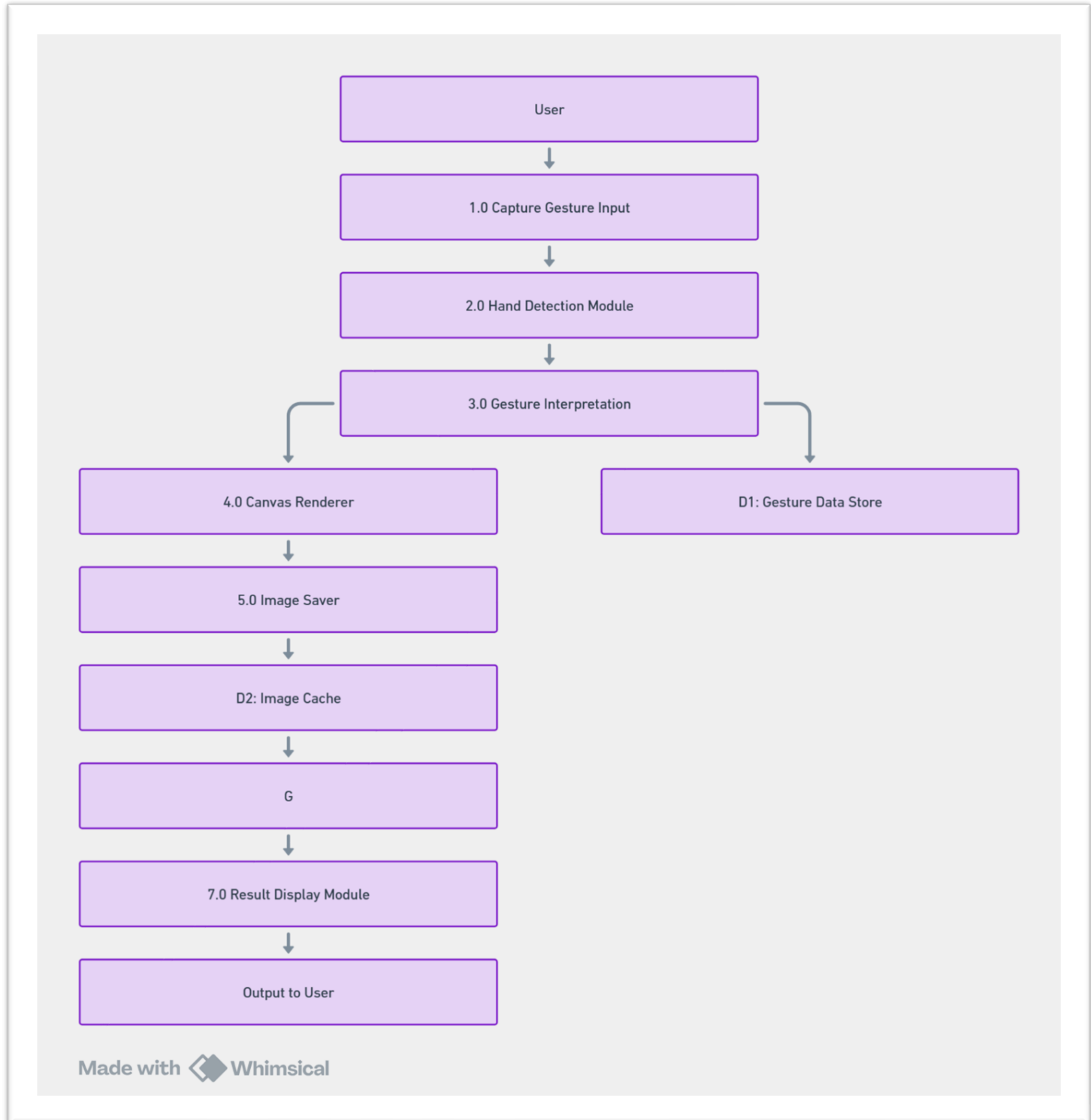
### 3.3.1 Data Flow Diagrams :

- Level 0 DFD: High-Level Process - This level shows the system as a single process and the interaction with external entities.



*Figure 3.1- Data Flow Diagram (level 0)*

- Level 1 DFD: Internal Breakdown - Level 1 expands the central process into more detailed subprocesses that explain how gesture processing, drawing, solving, and displaying are handled.



**Figure 3.2- Data Flow Diagram (level 1)**

3.3.2 Activity Diagram –

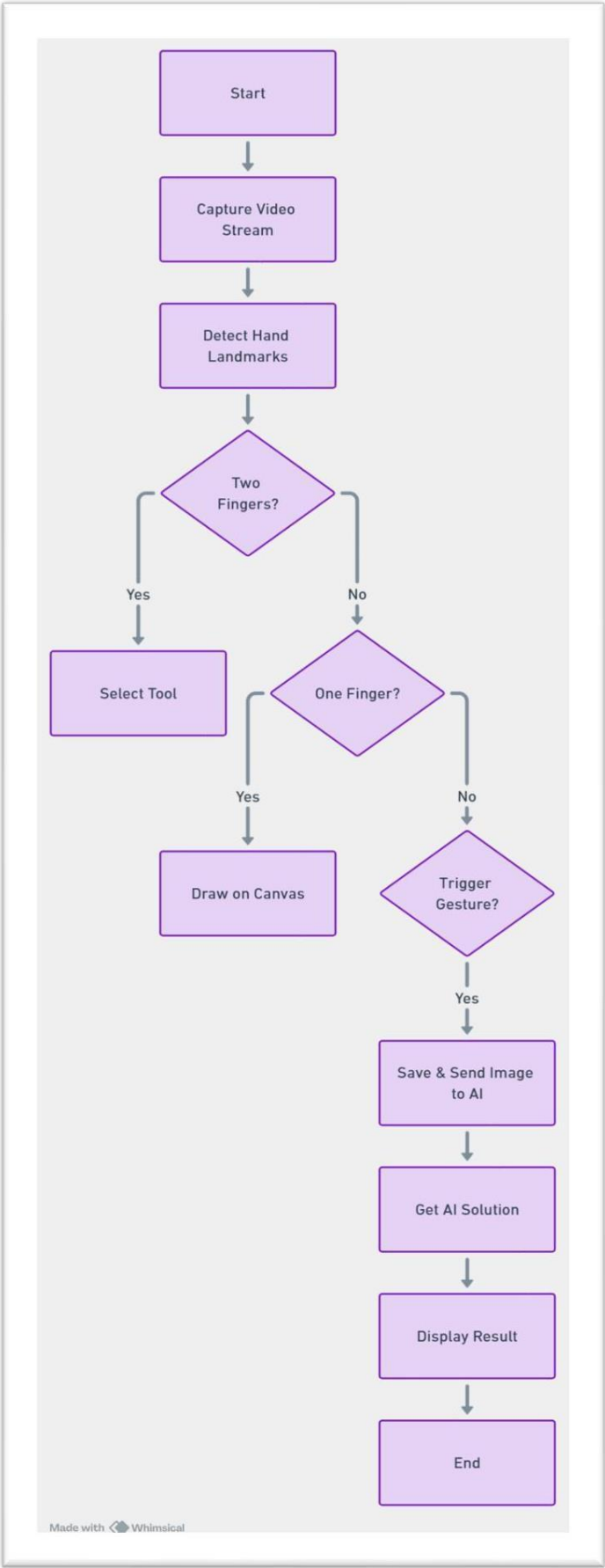
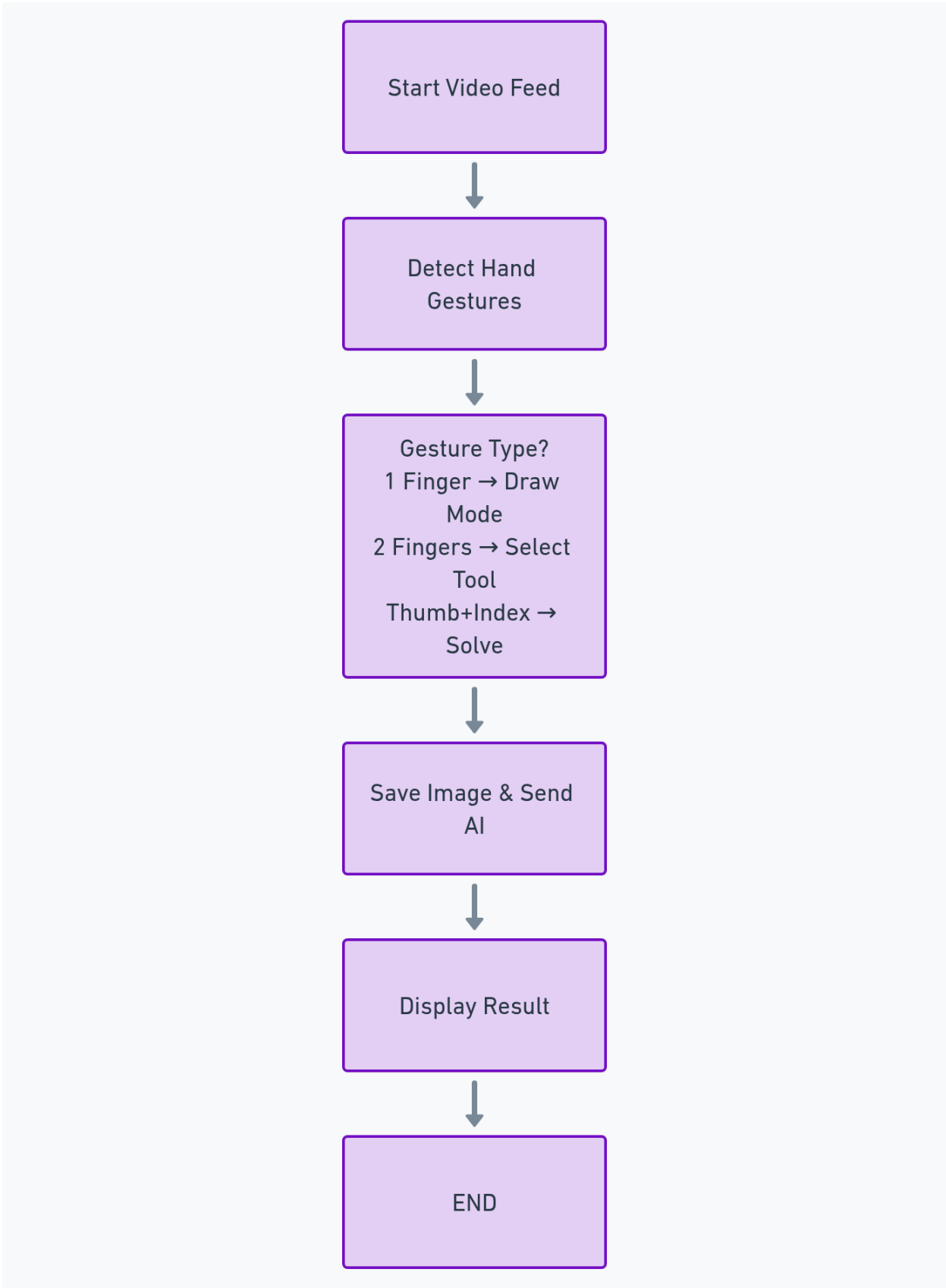


Figure 3.3 - Activity Diagram

**3.3.3 Flow Chart –**



*Figure 3.4 - Flow Chart*



3.3.4 Class Diagram -

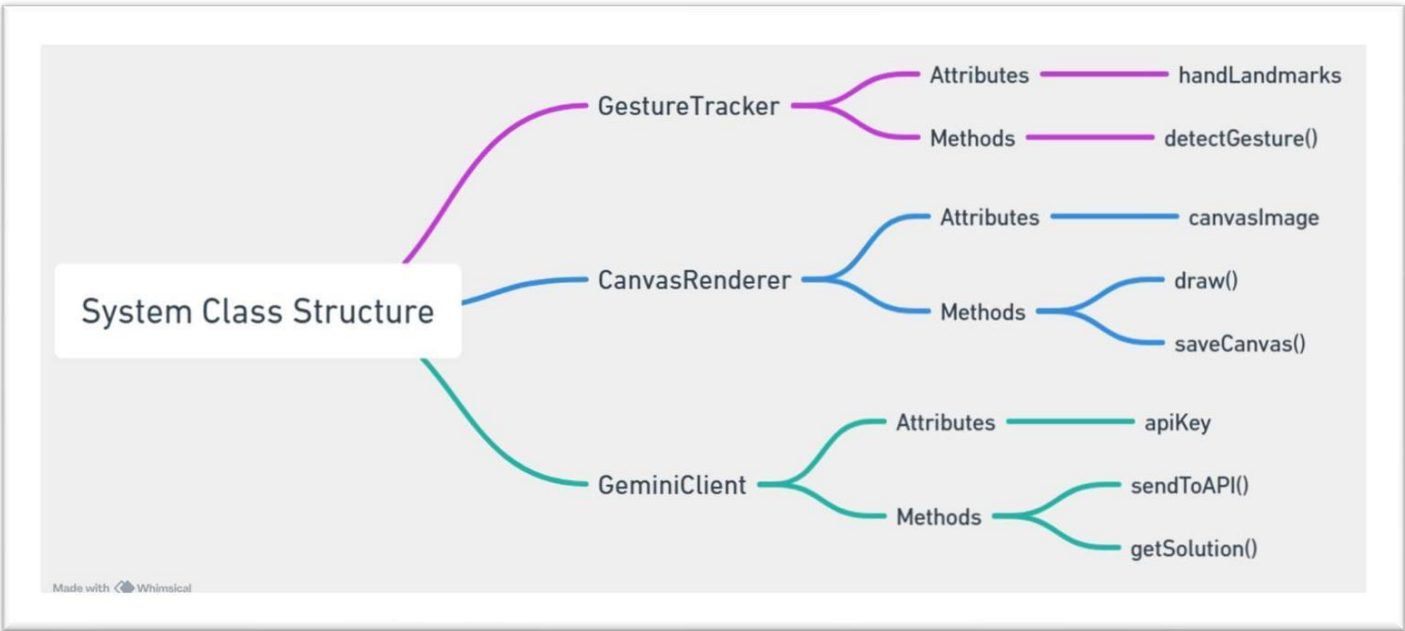


Figure 3.5 - Class Diagram

3.3.5 Sequence Diagram -

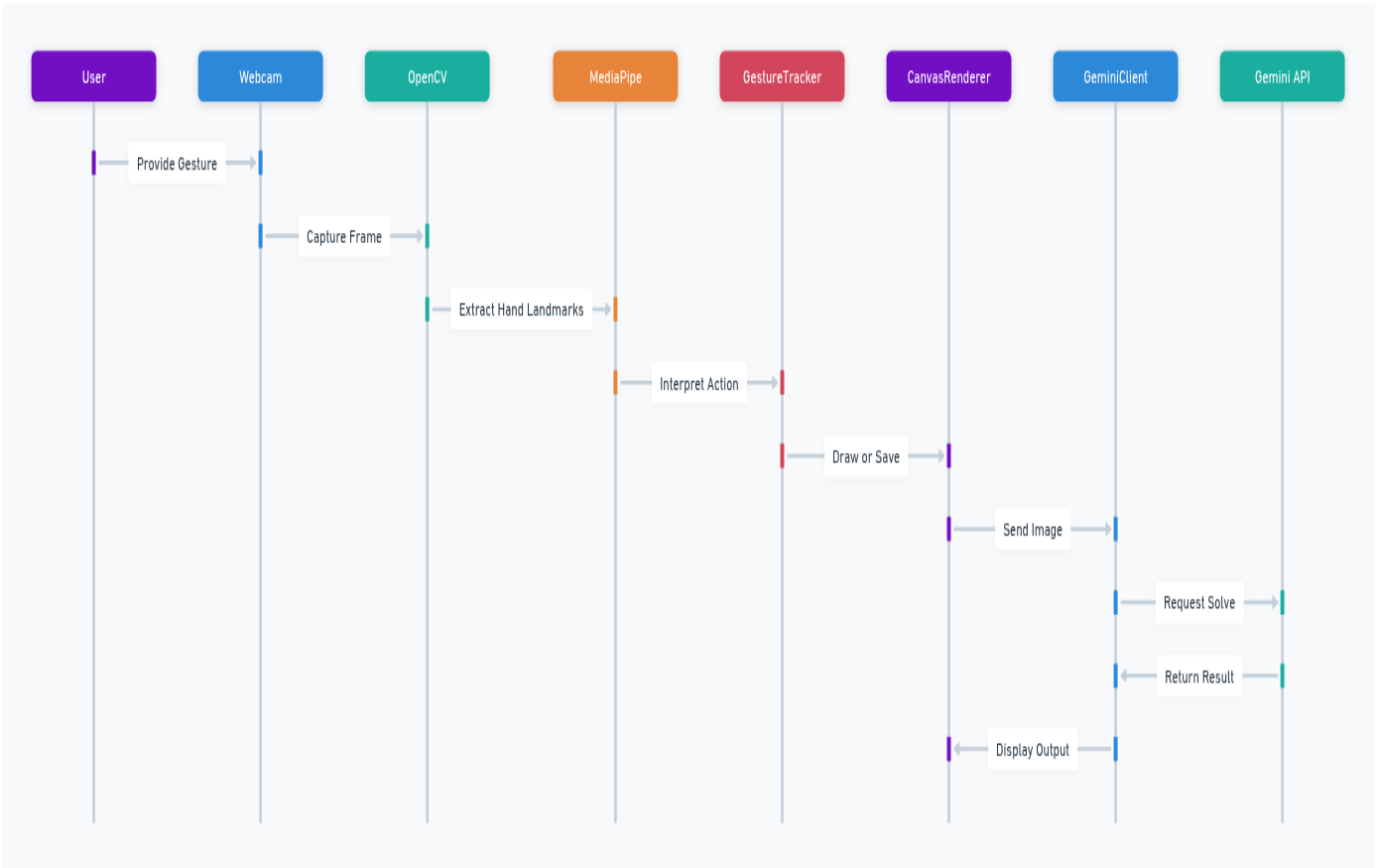


Figure 3.6 - Sequence Diagram

## **CHAPTER 4 : IMPLEMENTAION , TESTING AND MAINTENANCE**

### **4.1 INTRODUCTION TO LANGUAGES , IDEs , TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION :-**

The development of the AI Powered Virtual Calculator demanded a robust combination of programming languages, development environments, and supporting tools to ensure seamless functionality, modular design, and real-time interaction. Below is an overview of the technologies that made the implementation effective and efficient.

#### **1. Programming Languages Python-**

Python was the primary language for the backend implementation due to its simplicity, rich set of libraries (OpenCV, MediaPipe), and strong community support for computer vision and AI.

#### **JavaScript-**

JavaScript was used to enhance frontend interactivity and to handle dynamic updates (such as fetching results from the Flask backend).

#### **HTML & CSS-**

HTML was used for creating the basic structure of the frontend, while CSS was employed for styling and layout. Together, they provided a clean and responsive user interface for the virtual calculator.

#### **2. Development Tools & IDEs Visual Studio Code (VS Code) -**

The team utilized Visual Studio Code as the primary Integrated Development Environment (IDE) due to its support for Python, HTML, CSS, and JavaScript along with real-time extensions and Git integration.

#### **GitHub -**

GitHub was used for version control and collaborative development. It ensured that code contributions from multiple team members were efficiently managed and tracked.

#### **3. Libraries and Frameworks OpenCV (Open Source Computer Vision Library) -**

Used extensively for real-time video processing, drawing gestures on a virtual canvas, and capturing snapshots for AI processing.

#### **MediaPipe (by Google) -**

MediaPipe provided robust hand-tracking capabilities. It was used to detect and classify hand landmarks, essential for interpreting gestures.

#### **Flask -**

A lightweight Python web framework used to handle backend services, serve video frames to the web interface, and connect with external APIs such as Google Gemini.

### **Google Generative AI (Gemini API) -**

This API powered the AI component of the calculator by interpreting the handwritten gesture images and solving the recognized mathematical expressions using natural language reasoning,

### **4. Other Technologies NumPy -**

Used for numerical operations and matrix manipulations when processing image data.

### **PIL (Python Imaging Library) -**

Used to convert OpenCV images into formats suitable for API communication and display.

### **Dotenv -**

Employed for secure management of sensitive credentials like API keys by storing them in a .env file.

## **4.2 TESTING TECHNIQUES AND TEST PLANS:-**

The reliability and accuracy of the AI Powered Virtual Calculator were validated using structured testing strategies, covering both unit-level and system-wide functionalities. The following describes the testing techniques employed and the detailed plan used during development.

### **Testing Techniques :-**

#### **1. Unit Testing**

- Each function or module was independently tested. Key areas included:
- Hand landmark detection via MediaPipe
- Drawing recognition and canvas updates
- Gemini API integration and result parsing

#### **2. Integration Testing**

Modules such as gesture recognition, canvas rendering, and backend communication were tested together to ensure data passed seamlessly between them.

#### **3. Functional Testing**

Real-world test cases were used to evaluate system functionalities such as:

- Detecting specific hand gestures
- Drawing readable mathematical expressions
- Successfully retrieving and displaying AI-generated solutions

#### 4. Performance Testing

Assessed the responsiveness of the real-time video processing and the average time taken by the Gemini API to return results. The system maintained real-time rendering at an average of 20–25 FPS.

#### 5. Usability Testing

Conducted with fellow classmates to observe the ease of use, UI feedback clarity, and intuitiveness of hand gesture interactions.

#### Test Plan:-

Test Case ID	Description	Input	Expected Output	Status
TC01	Hand gesture detection initialization	OpenCV + webcam stream	Accurate hand tracking with MediaPipe	Pass
TC02	Drawing using index finger	Hand in draw mode	Canvas draws as finger moves	Pass
TC03	Switching between tools (eraser/pen)	Gesture with two fingers	Tool switches visually and functionally	Pass
TC04	Saving canvas to file	'Save' gesture detected	saved_canvas.jpg created in local storage	Pass
TC05	AI solving via Gemini API	saved_canvas.jpg	Gemini returns LaTeX-formatted solution	Pass
TC06	Invalid/unclear drawing	Incomplete math expression	Gemini prompts for clarification or error	Pass
TC07	API key missing or invalid	.env misconfigured	Flask logs error; frontend handles gracefully	Pass
TC08	Real-time frame lag under load	Multiple fast gestures	Minimal frame drops, handled via threading	Pass

*Table 4.1 - Tests performance status of project*

## CHAPTER 5: RESULTS AND DISCUSSION

### 5.1 USER INTERFACE REPRESENTATION:-

The user interface (UI) of the AI Powered Virtual Calculator is designed to be intuitive, responsive, and userfriendly. It bridges the interaction between the user's physical gestures and the system's intelligent backend that interprets and solves mathematical problems.

#### 1. Interface Components - A) Live Camera Feed Window

A real-time video stream from the webcam is displayed, showing the user's hand gestures. It serves as the main interaction area where the user draws numbers or symbols in the air.

#### B) Gesture Drawing Canvas

An overlay on top of the video feed that captures index-finger motion for drawing. It acts as the virtual writing pad for users to sketch expressions.

#### C) Tool Indicator Panel

Displays the currently selected tool:

- Pen (default)
- Eraser
- Save & Solve trigger The panel updates based on gesture recognition to indicate mode changes.

#### D) Color and Thickness Selector (Optional in future versions)

Allows dynamic customization of pen color or stroke thickness via gesture toggles or UI buttons.

#### E) Solution Display Area

Once the canvas image is processed by the backend, the Gemini API response is rendered in a text area or popup beneath the drawing feed. The solution includes both the expression interpretation and the step-by-step result.

### 2. Workflow Overview -

- User positions hand in front of the camera.
- System detects gesture and enters drawing or selection mode.
- Expression is drawn on the air-canvas.
- Specific gesture triggers saving of the drawing.
- The system sends the image to Gemini API and receives a response

### 3. Visual Representation (Annexure Suggestion) -

You can attach screenshots in the annexure for each of the following:

- Initial webcam interface with hand detection
- Canvas while drawing an equation

- Tool switch gesture being recognized
- Saved drawing preview
- Gemini-generated solution being displayed

## **5.2 BRIEF DESCRIPTION OF VARIOUS MODULES OF THE SYSTEM :-**

The AI Powered Virtual Calculator system is divided into modular components, each responsible for a specific function. This modular structure enhances scalability, maintainability, and collaboration during development.

### **1. Hand Detection Module (handTrack.py) -**

- Uses Google's MediaPipe framework to identify hand landmarks in real-time.
- Tracks finger positions and recognizes gestures (e.g., index finger up = draw mode, two fingers = selection mode).
- Outputs coordinates used to draw on a virtual canvas.

### **2. Drawing Canvas Module (main.py) -**

- Initializes the OpenCV video stream and overlays a blank canvas on the webcam feed.
- Tracks finger movement to draw strokes corresponding to mathematical expressions.
- Detects gesture combinations to toggle between drawing, erasing, and saving.

### **3. Tool Selection and Gesture Interpretation -**

- Detects specific hand poses to:
- Switch between pen and eraser.
- Trigger saving of the canvas when thumb and index finger are brought close.
- Displays current mode on the interface for visual feedback.

### **4. Image Processing and Save Handler -**

- Captures the canvas as an image (saved\_canvas.jpg) when a save gesture is made.
- Prepares the image for further processing or transmission by resizing, converting, or enhancing contrast (if needed)

### **5. Backend Processing & API Communication (app.py) -**

- Built using Flask to provide a lightweight backend service.
- Receives requests from the frontend to send the saved canvas image to the Gemini API.
- Parses the response and returns a human-readable solution.

### **6. Gemini API Integration -**

- Integrates Google's Generative AI model (Gemini 1.5 Pro) using secure API keys.
- Sends the saved image to the Gemini model with prompts like "Interpret this mathematical image and solve it step-by-step."
- Receives detailed output including the interpreted expression and steps to solve it.

### **7. Frontend Interface (HTML, CSS, JS) -**

- Displays the live video stream with the overlay canvas.

- Provides feedback on current mode/tool and visual status of API results.
- Uses JavaScript (scripts.js) to asynchronously fetch and display AI responses without reloading the page.

## 8. Environment and Configuration Module -

- Uses .env file and the dotenv package to securely store and access sensitive credentials like the Gemini API key.
- Contains requirements.txt to manage Python dependencies and environment setup.

## 5.3 SNAPSHOTS OF SYSTEM WITH BRIEF DETAILS:-

### 1. Home Page (Web Interface)

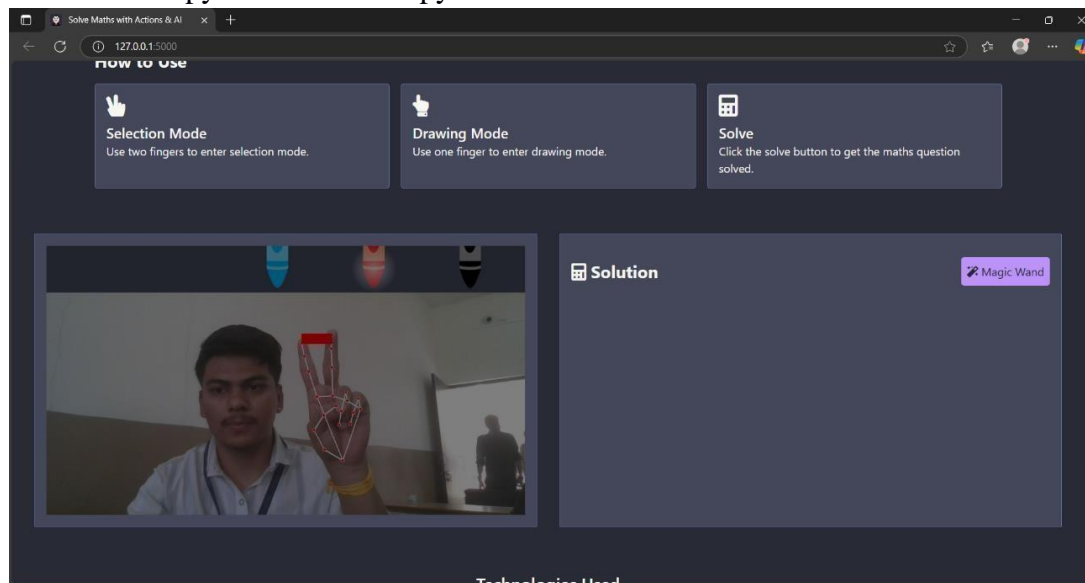
The user first lands on a simple, intuitive web interface served via Flask.

Components:

- Live webcam feed (handled via JavaScript + Flask streaming).
- Virtual drawing canvas where gestures are visualized.
- "Solve" button (optional fallback to gesture-based trigger).
- Output area for AI-generated results (expression + solution).
- Technologies: HTML, CSS (styles.css), JS (scripts.js), Flask for dynamic content.

### 2. Real-Time Hand Gesture Tracking

- Captured via webcam using OpenCV.
- MediaPipe detects hand landmarks (e.g., index finger, thumb).
- Gesture logic determines:
  - Two fingers → select mode
  - One finger → drawing mode
  - Pinch (index + thumb) → save canvas and trigger backend
- Implemented in main.py and handTrack.py



*Figure 5.1 - Real-time hand Gesture tracking*

### 3. Virtual Drawing Canvas

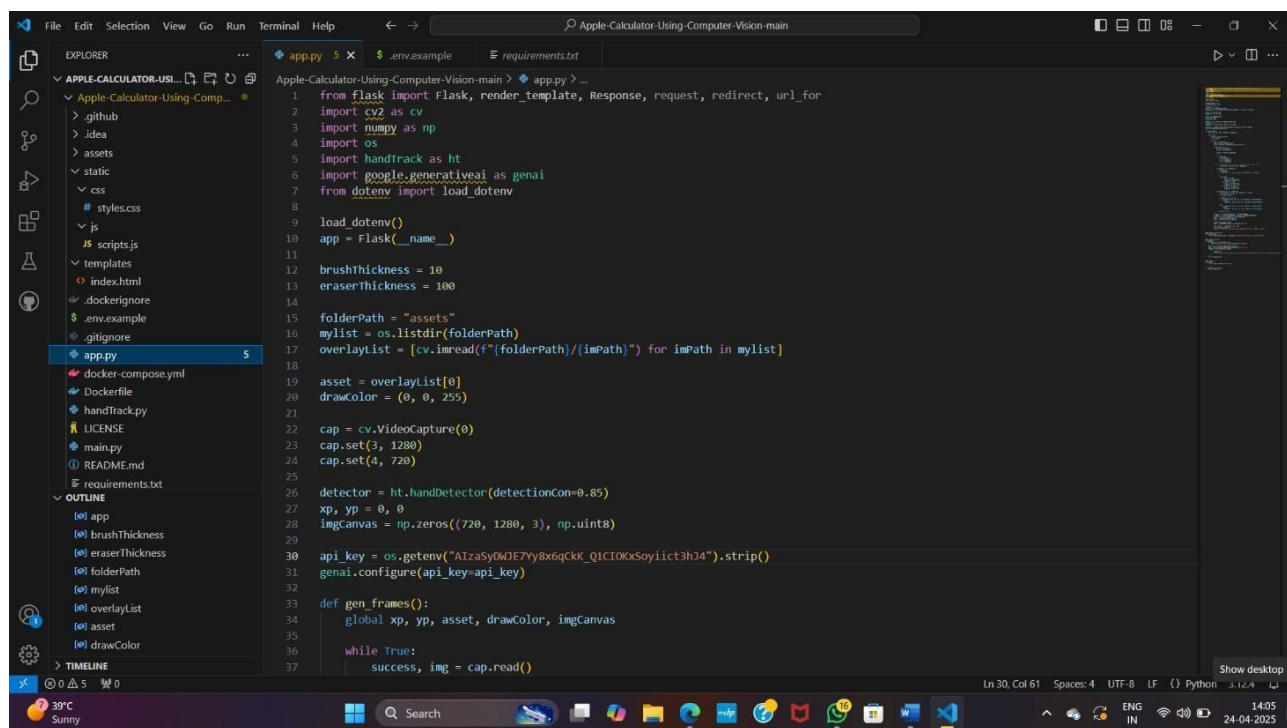
- A whiteboard-like area is overlaid on the video feed.
- Finger movement is used to "write" numbers and math symbols.
- Visual feedback (color, stroke width) shown in real time.

### 4. Canvas Image Capture

- When a save gesture is made:
- Canvas is stored locally as saved\_canvas.jpg.
- Flask backend is called via HTTP to initiate processing.

### 5. Backend Processing (Flask app.py)

- Handles routes like:
- /solve: Receives saved image.
- Communicates with Google Gemini API using an image prompt.
- Uses environment variable (.env) for Gemini API key (secure).
- Sends result back as JSON or HTML response.



```
1 from flask import Flask, render_template, Response, request, redirect, url_for
2 import cv2 as cv
3 import numpy as np
4 import os
5 import handtrack as ht
6 import google.generativeai as genai
7 from dotenv import load_dotenv
8
9 load_dotenv()
10 app = Flask(__name__)
11
12 brushThickness = 10
13 eraserThickness = 100
14
15 folderPath = "assets"
16 mylist = os.listdir(folderPath)
17 overlayList = [cv.imread(f"{folderPath}/{imPath}") for imPath in mylist]
18
19 asset = overlayList[0]
20 drawColor = (0, 0, 255)
21
22 cap = cv.VideoCapture(0)
23 cap.set(3, 1280)
24 cap.set(4, 720)
25
26 detector = ht.handDetector(detectionCon=0.85)
27 xp, yp = 0, 0
28 imgCanvas = np.zeros((720, 1280, 3), np.uint8)
29
30 api_key = os.getenv("AIzaSyDkDE7y8x6gCkK_QICIOkxSoyiict3h74").strip()
31 genai.configure(api_key=api_key)
32
33 def gen_frames():
34     global xp, yp, asset, drawColor, imgCanvas
35
36     while True:
37         success, img = cap.read()
```

*Figure 5.2 – Backend code snippet*

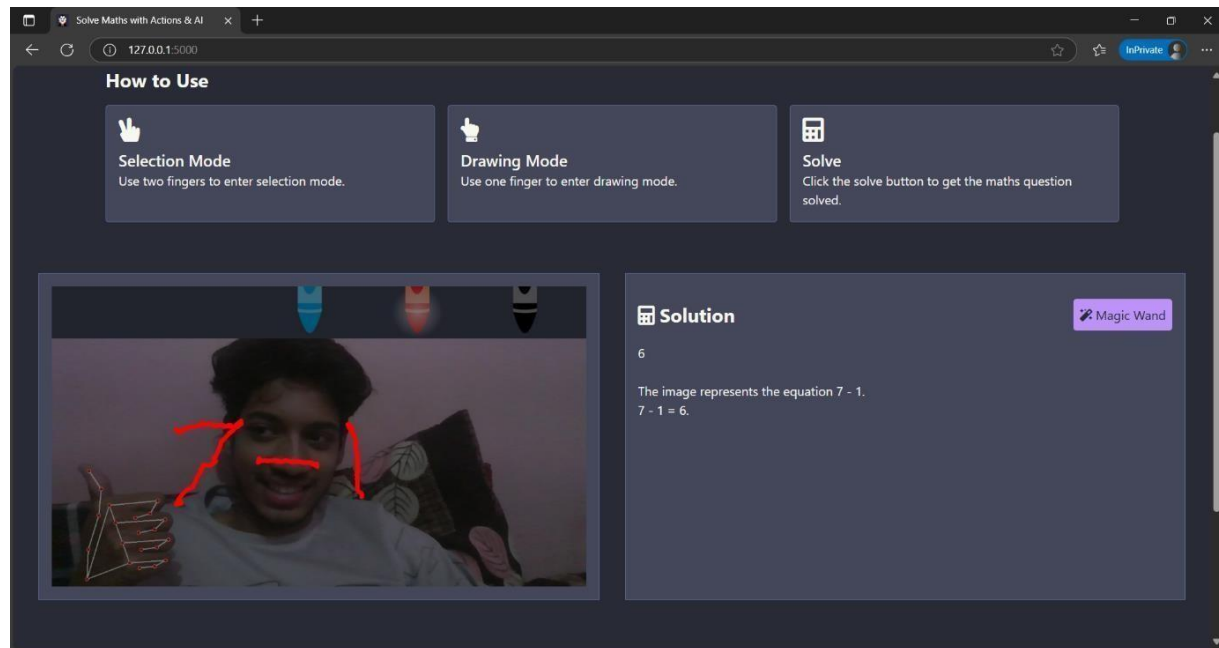
### 6. Google Gemini API

- Receives canvas image as input.
- Prompts it with: "Solve the following hand-written mathematical expression."
- Returns:
- Parsed expression



## 7. Solution Display

- Web interface is dynamically updated using JS and Flask.
- Displays:
  - The interpreted equation (e.g., “ $2x + 3 = 7$ ”)
  - The complete solution steps
  - Final output (e.g., “ $x = 2$ ”)



*Figure 5.3 -Output*

## 5.4 BACK END REPRESENTATION :-

The backend of the AI Powered Virtual Calculator serves as the processing engine that handles gesture recognition, image capturing, API interaction, and solution retrieval. Built primarily with Python and Flask, the backend is lightweight yet robust enough to perform real-time computation and communication.

### 1. Technology Stack :

- Programming Language: Python
- Web Framework: Flask
- External API: Google Gemini Generative AI API
- Libraries Used: OpenCV, MediaPipe, NumPy, Pillow, dotenv

### 2. Core Backend Responsibilities :

- Initialize and manage webcam video stream
- Recognize hand gestures using MediaPipe
- Draw and update gesture strokes on a virtual canvas
- Save the canvas as an image file when the gesture is triggered
- Send the image to the Gemini API and retrieve the result
- Display AI-generated mathematical solutions on the frontend

### 3. Backend File Structure :

#### **main.py:**

- Runs the real-time hand tracking and canvas drawing system using OpenCV.
- Detects gestures, handles tool switching, and saves the canvas image. **handTrack.py:**
- Encapsulates the hand landmark detection logic using MediaPipe.
- Returns finger coordinates and identifies finger status (up/down).

#### **app.py:**

- Hosts the Flask server.
- Defines API endpoints to receive requests from the frontend, send images to the Gemini API, and return the interpreted solution.

#### **.env:**

- Contains sensitive data like the Gemini API key. Loaded securely via the dotenv library.

#### **requirements.txt:**

Lists all Python dependencies needed to run the backend environment.

### 4. API Communication Flow :

1. User performs a “save” gesture to trigger canvas image capture.
2. Flask backend receives a request from JavaScript (AJAX call).
3. The saved canvas image is loaded and sent to the Gemini model with a prompt.
4. Gemini API responds with the interpreted expression and solution.
5. Flask sends this response back to the frontend, which displays it below the video stream.

### 5. Security and Efficiency Considerations :

- Environment Variables: Sensitive keys are stored in .env and not hardcoded.
- Lightweight Framework: Flask ensures fast server response and minimal overhead.
- Asynchronous Handling: JavaScript fetch ensures non-blocking updates on the UI side.
- Error Handling: Logs and response codes are implemented to handle cases such as invalid API keys or network failures.

## CHAPTER 6: SUMMARY AND CONCLUSION

The AI Powered Virtual Calculator using Computer Vision and Generative Model presents an innovative fusion of gesture-based human-computer interaction and cutting-edge AI reasoning. Designed with a user-friendly interface and real-time capabilities, the system empowers users to perform mathematical calculations simply by drawing expressions in the air.

Throughout the development process, various technologies such as OpenCV, MediaPipe, Flask, and the Google Gemini API were integrated to build a robust solution. The project was developed using Python for backend logic and gesture processing, HTML/CSS and JavaScript for frontend interactivity, and Gemini's large language model for intelligent problem-solving.

The system supports a touchless interaction model, making it particularly valuable in educational, assistive, and public settings where contactless operation is desired. The modular design and lightweight implementation also make it easy to scale or expand.

This project demonstrates how artificial intelligence and computer vision can converge to solve everyday tasks in an intuitive way. By transforming hand gestures into meaningful digital input and using generative AI to interpret and solve complex mathematical problems, the virtual calculator redefines how users can interact with computational tools.

Key achievements include:

- Real-time hand gesture recognition with accuracy and fluidity.
- Dynamic canvas for drawing mathematical expressions.
- Seamless integration with Google Gemini for problem-solving.
- A responsive web interface for real-time feedback and results.

The successful implementation of this project not only reflects technical proficiency but also opens doors to future innovations in AI-powered assistive technologies.

## CHAPTER 7: FUTURE SCOPE

While the current version of the AI Powered Virtual Calculator offers a fully functional and interactive system, several enhancements and new features can be incorporated to expand its capabilities and improve user experience. The following points outline the potential future developments:

### 1. Multi-Symbol and Multi-Line Expression Recognition

Extend the gesture detection and canvas logic to support complex mathematical expressions, such as equations involving integrals, summations, and matrices over multiple lines.

### 2. Real-Time Expression Parsing

Introduce real-time optical character recognition (OCR) or gesture classification models to interpret mathematical symbols as the user draws them, allowing instant feedback before submission.

### 3. Voice Command Integration

Integrate voice recognition for tool switching (e.g., “eraser,” “save”) or verbally confirming expressions, making the system more inclusive for users with limited mobility.

### 4. Adaptive AI Feedback

Leverage Gemini’s conversational capabilities to not only solve problems but also guide users step-by-step, making the calculator a teaching assistant for math learners.

### 5. Mobile and Tablet Compatibility

Port the system to mobile platforms using React Native or Flutter, allowing hand gesture detection through front-facing cameras on smartphones or tablets.

### 6. Cloud-Based Model Hosting

Move the backend to cloud infrastructure to enhance scalability and allow multiple users to access the calculator simultaneously via a web app or mobile application.

### 7. User Authentication and History Tracking

Add a login system to store user history and previous problem sets, enabling students to revisit solved equations and track learning progress.

These improvements aim to position the calculator not only as a utility tool but also as a futuristic and intelligent learning companion across platforms and user groups.

# APPENDIX

This appendix includes supplementary materials that support the understanding and technical depth of the AI Powered Virtual Calculator project. The contents provide additional context for the system's implementation and usage.

- **Appendix A – System Architecture Diagram -**

A visual representation of the end-to-end system pipeline:  
User Gesture → MediaPipe → Canvas Drawing → Save Image → Flask API → Gemini Model → Display Result.

- **Appendix B – Gesture Mapping Table -**

Gesture Type	Description	Function Triggered
One Finger (Index)	Drawing Mode	Draws on canvas
Two Fingers Up	Selection Mode	Switch tool (pen/eraser)
Thumb + Index Touch	Trigger Save	Save canvas as image

- **Appendix C – Gemini AI Output Samples -**

1. Input Image: Hand-drawn equation
2. Output: Gemini’s interpretation and step-by-step solution in natural language format.

- **Appendix D – Code Snapshots -**

1. main.py: Drawing loop and gesture handling
2. handTrack.py: Hand landmark detection logic
3. app.py: API request and Gemini model communication
4. scripts.js: AJAX fetch and dynamic result update

## REFERENCES

- [1] Mitra, S., & Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3), 311–324.
- [2] Gao, X., Zhang, Y., & Wang, K. (2020). CNN-based static and dynamic hand gesture recognition. *Pattern Recognition Letters*, 133, 40–47.
- [3] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- [4] Zhang, Z., Cui, Y., & Xu, D. (2019). Real-Time Object Detection with OpenCV in Educational Applications. *International Journal of Emerging Technologies in Learning*, 14(15), 66–77.
- [5] Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
- [6] Google. (2024). Google Gemini API Documentation. Retrieved from <https://ai.google.dev/>
- [7] Wolfram Alpha. (2023). Symbolic Mathematical Solver. Retrieved from <https://www.wolframalpha.com/>
- [8] Gupta, R., Banerjee, S., & Narayanan, A. (2022). Comparative Study of AI-Based Math Solvers. *Journal of Intelligent Systems*, 31(2), 125–134.
- [9] MediaPipe. (2021). Google's MediaPipe Framework. Retrieved from <https://mediapipe.dev/>
- [10] TensorFlow. (2024). Image Classification with TensorFlow. Retrieved from <https://www.tensorflow.org/tutorials/images/classification>