VISVESVARAYATECHNOLOGICALUNIVERSITY
BELAGAVI

Project Report on

"TITLE OF THE PROJECT"

Submitted in the partial fulfillment for the requirements of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted By

MUSKAN GOYAL	**1BY21CS218**
PRANJAL TYAGI	**1BY21CS223**
BHAGYALAXMI	**1BY21CS030**

Under the guidance of

DR.THIPPESWAMY G

HOD & PROFESSOR
Department of CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064.

2024-2025

VISVESVARAYATECHNOLOGICALUNIVERSITY
BELAGAVI

BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"SIGN,SPEAK,AND SOLVE: AN AI-POWERED LEARNING TOOL FOR THE DEAF AND HARD OF HEARING"** is a bonafide work carried out by **Muskan Goyal (1BY21CS218),Pranjal Tyagi (1BY21CS223), Bhagyalaxmi (1BY21CS030),** in partial fulfillment for the award of **Bachelor of Engineering Degree in Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2024-2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report. The project report has been approved as it satisfies the academic requirements in respect of project work for B.E. Degree.

Dr. Dhanalakshmi BK Assistant Professor Dept. of CSE, BMSIT&M	Dr. Thippeswamy G Professor & HOD, Dept. of CSE, BMSIT&M	Dr. Sanjay H. A Principal, BMSIT&M

External VIVA-VOCE

Name of the Examiners	Signature with Date
1.	
2.	

ACKNOWLEDGEMENT

We are pleased to present this project report upon its successful completion. This project would not have been possible without the guidance, assistance, and suggestions of many individuals. We would like to express our deep sense of gratitude to each and every person who has contributed to making this project a success.

First and foremost, we extend our heartfelt thanks to **Dr. Sanjay H. A, Principal, BMS Institute of Technology & Management,** for his constant encouragement and inspiration in undertaking this project.

We also express our sincere gratitude to **Dr. Thippeswamy G, Head of the Department, Computer Science and Engineering, BMS Institute of Technology & Management**, for his unwavering support and motivation throughout this endeavor.

We also express our sincere gratitude to Associate head **Dr Mahesh G, Professor, Cluster 1, Department of Computer Science and Engineering,** for his constant support and valuable advice during the project.

We are grateful to our project guide, **Dr. Dhanalakshmi BK, Assistant Professor, Department of Computer Science and Engineering,** for their invaluable encouragement and guidance throughout the project.

Special thanks are due to all the staff members of the Computer Science and Engineering Department for their help and cooperation.

Finally, we would like to express our gratitude to our parents and friends for their unwavering encouragement and support throughout the duration of this project.

By,
Muskan Goyal
Pranjal Tyagi
Bhagyalaxmi

DECLARATION

We **Muskan Goyal (1BY21CS218), Pranjal Tyagi (1BY21CS223), Bhagyalaxmi (1BY21CS030)** students of Eight semester B. E, in the Department of Computer Science and Engineering, BMS Institute of Technology and Management, Bengaluru declare that the project work entitled "**SIGN,SPEAK,AND SOLVE: AN AI-POWERED LEARNING TOOL FOR THE DEAF AND HARD OF HEARING**" has been carried out by us and submitted in partial fulfilment of the course requirements for the award of degree in **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year 2024 - 2025. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Muskan Goyal(1BY21CS218)

Pranjal Tyagi (1BY21CS223)

Bhagyalaxmi (1BY21CS030)

ABSTRACT

This project presents an innovative solution for solving mathematical problems using real-time hand gestures, combining the power of computer vision, artificial intelligence, and natural language translation. The system allows users to draw mathematical expressions in the air using simple finger gestures detected via webcam, eliminating the need for traditional input devices such as keyboards or styluses. By leveraging OpenCV and the cvzone hand tracking module, the application accurately identifies specific gestures—such as drawing with the index finger, navigating with two fingers, clearing the canvas with a thumbs-up, and solving problems with an open palm.

Once a problem is drawn, it is captured and processed through Google's Gemini AI (Generative Model), which interprets the handwritten expression, solves it, and provides a step-by-step explanation. Additionally, the solution can be translated into multiple regional languages including Hindi and Kannada using MarianMT translation models, making the system accessible to a broader user base.

Developed with Python and deployed using Streamlit, the application offers an intuitive, browser-based interface for interactive learning and problem-solving. This gesture-based approach not only enhances user engagement but also serves as a valuable educational tool for visually demonstrating mathematical logic. The system was rigorously tested with various problem types, demonstrating high accuracy in gesture recognition, AI response, and translation quality. Overall, the project aims to redefine the way mathematical problems are approached, offering a hands-free, interactive, and multilingual solution for learners and educators alike.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 BACKGROUND

In today's technology-driven world, Artificial Intelligence (AI) and Computer Vision are revolutionizing how humans interact with machines. Among these innovations, gesture-based interaction stands out as a powerful tool to bridge the communication gap for individuals with disabilities. Especially for the deaf and mute community, hand gestures serve as a vital mode of expression. However, the field of education, particularly in mathematical problem solving, still lacks accessible and inclusive tools tailored for these communities.

This project aims to introduce a Math Gesture Recognition system that allows users to draw mathematical equations in the air using simple hand gestures. These gestures are detected using computer vision tools and then interpreted by a generative AI model to provide solutions. The results are then translated into multiple languages, including Hindi and Kannada, making it regionally adaptable. The entire solution is delivered through an intuitive web-based UI powered by Streamlit.

By combining gesture recognition, real-time AI interpretation, and multilingual support, this system not only supports inclusive education but also promotes ease of access and user-friendly interaction without requiring typing or voice commands.

1.2 LITERATURE SURVEY

In research paper [1], the authors train neural networks for image classification, including complex images alongside MNIST models. This effort results in the development of Recurrent Neural Network (RNN) models tailored for the images under consideration. Notably, the classification performance achieves such high accuracy that even discernment by the human eye becomes challenging.

In the research paper [2], the authors aim to identify and understand human actions in videos by utilizing three distinct deep learning algorithms: Two-Stream CNN, CNN+LSTM, and 3D CNN. These algorithms are designed to address the complexities of human actions, such as varying perspectives and background noise. The HMDB-51 dataset is employed for testing, and experimental results demonstrate the effectiveness of these methods in recognizing human actions. The best-performing algorithm among the three is then selected for further applications in human action recognition.

In research paper [3], the authors develop a deep learning approach for sign language recognition to aid individuals with hearing and language impairments. They use a Faster R- CNN-based network to locate hands in videos, followed by a 3D CNN and LSTM framework for recognizing sign language sequences. This combined method achieves a 99% recognition rate on common vocabulary datasets, surpassing other known techniques.

In research paper [4] Gesture recognition, enhanced by cognitive AI and deep learning, is crucial for advanced human-computer interaction. A 2023 study from the ICAEECI conference showcases using convolutional and recurrent neural networks to improve gesture detection accuracy, demonstrating significant advancements in the field.

1.3 MOTIVATION

Education is a basic right, yet many differently-abled individuals still face barriers in accessing effective learning tools. Mathematics, being a core academic subject, often presents a challenge when conventional methods like writing, typing, or verbal input are inaccessible.

This project is driven by the vision of :

1. Empowering the deaf and mute community with an intuitive learning solution.
2. Making math learning visual, interactive, and language-adaptive.
3. Reducing reliance on traditional input methods and making technology more inclusive.

By using AI and gesture recognition, we aim to enable students to express math visually and learn through direct interaction without any physical medium.

1.4 PROBLEM STATEMENT

To develop an interactive system that:

1. Recognizes and captures mathematical equations drawn using hand gestures.
2. Uses Google's Gemini AI to solve and explain these equations.
3. Translates the AI-generated solutions into regional languages like Hindi and Kannada.
4. Presents the entire experience within an easy-to-use Streamlit web interface, making it suitable for learners with disabilities.

1.5 AIM AND OBJECTIVE

Aim:

To create a gesture-based AI system that assists users, especially differently-abled students, in solving mathematical problems without using a keyboard or voice input.

Objectives:

1. To detect and interpret hand gestures representing mathematical expressions.
2. To render gestures into a canvas using OpenCV and CVZone.
3. To query Gemini AI with the canvas and receive a solution with explanation.
4. To use MarianMT models for translating the result into regional languages.
5. To build a responsive UI using Streamlit that supports real-time gesture input and multilingual output.

1.6 SCOPE

The scope of this project is focused on creating a proof-of-concept application that can:

1. Track hand gestures using a standard webcam and interpret them to draw expressions.
2. Handle basic math problems (addition, subtraction, multiplication, division, and simple algebra).
3. Interface with AI to solve math problems and return understandable explanations.
4. Translate the results into supported regional languages, enhancing accessibility and user experience.

The project is designed with modularity in mind, allowing future extensions such as:

1. Complex mathematical problem solving (e.g., calculus, matrices).
2. Integration with voice assistants for added interactivity.
3. Expansion to support additional regional or international languages.xtended to support sign language learning, voice output, and more advanced AI models.

1.7 CHALLENGES

While the project showcases innovative integrations, it also comes with several technical and practical challenges:

1. **Gesture Accuracy:** Ensuring accurate detection and differentiation between gestures, especially under varying lighting conditions or hand orientations.

2. **Camera Dependency:** Low-resolution or poorly positioned cameras can affect the quality of tracking and gesture recognition.

3. **Gesture Ambiguity:** Some finger gestures (e.g., two fingers vs three fingers) can be mistakenly interpreted due to small differences or tracking errors.

4. **AI Limitations:** The generative model may sometimes misunderstand ambiguous or unclear drawings, especially if the user's hand-drawn symbols deviate from standard formats.

5. **Translation Complexity:** Mathematical language can be challenging to translate effectively, particularly when preserving accuracy in technical explanations.

6. **User Adaptability:** Users unfamiliar with gesture-based systems may take time to adapt to gesture commands and controls.

CHAPTER 2

OVERVIEW

The "Math Solver with Hand Gestures" project introduces a unique approach to solving mathematical problems by integrating real-time hand gesture recognition, AI-based problem solving, and multilingual translation within a web application. This chapter outlines the high-level architecture, technology stack, components, and work flow that define the overall system.

2.1 System Overview

This project is a real-time interactive system where users can draw mathematical expressions in the air using their index finger, and the system interprets this input, solves it using a generative AI model, and translates the solution into a selected language. The system primarily comprises four core modules:

1. Hand Gesture Detection and Canvas Interaction
2. AI-based Math Problem Solving
3. Multilingual Translation
4. Web-based User Interface using Streamlit

Each module operates independently but communicates with the others to maintain a smooth and interactive user experience.

2.2 Functional Flow

The system operates through the following sequence of steps:

1. **Input via Hand Gestures**
    - The user’s hand gestures are captured using a webcam.
    - The system uses the CVZone HandTrackingModule to detect the position of the fingers and infer gestures such as drawing, navigation, clear, and solve.
2. **Gesture Interpretation & Canvas Drawing**
    - Index finger up: draws the problem on a virtual canvas.
    - Two fingers up: switches to navigation mode (lift pen).

- Thumb up: clears the canvas.
- Four fingers up: triggers the problem-solving mechanism.

3. **AI-Based Problem Solving**
   - The final drawing is converted into an image and sent to Google's Gemini AI (via the GenerativeModel API).
   - The AI interprets the handwritten mathematical problem and generates a response that includes:
   - The interpreted equation.
   - The step-by-step solution or explanation.
4. **Translation Module**
   - The response from the AI is optionally passed through a MarianMT translation model to convert it to Hindi or Kannada.
   - This makes the tool more accessible to a diverse user base.
5. **User Output Display**
   - The Streamlit frontend displays the AI response and the translated version if selected.
   - A real-time video feed with annotations (like gesture feedback and drawing) is also shown.

2.3 Technology Stack

Component	Technology Used
Hand Detection	CVZone (built over OpenCV and MediaPipe)
Programming Language	Python 3.x
User Interface	Streamlit
AI Math Solver	Google Generative AI (Gemini 1.5 Flash)
Translation Models	MarianMT (Helsinki-NLP) via HuggingFace Transformers
Image Handling	OpenCV, PIL
Language Translation	Transformers library from HuggingFace
Webcam Input	OpenCV VideoCapture

2.4 Key Modules Explained

1. Hand Gesture Detection

- Uses the CVZone HandTrackingModule to detect 21 landmarks on a hand.
- Identifies finger states (up/down) to recognize gestures.
- Uses logic to map gestures to drawing, navigation, or clearing actions.

2. Drawing Mechanism

- Tracks the position of the index fingertip.
- Connects current and previous positions to simulate drawing.
- Supports line segmentation during navigation mode to avoid unintended connections.

3. AI Solver Integration

- Captures the canvas and converts it to an image.
- Sends the image to Gemini AI with a prompt for math problem-solving.
- Receives structured responses including the solution and explanation.

4. Translation Layer

- Translates English responses into Hindi or Kannada using MarianMT models.
- Adjusts prompts for multilingual models (e.g., prefixing with >>kan<< for Kannada).
- Maintains the integrity of mathematical terms during translation.

2.5 User Interface Design

The user interface is developed using Streamlit, offering a responsive, web-based experience & the Key UI features include:

1. Live webcam feed with gesture overlay.
2. Run checkbox to enable/disable the system.
3. Language dropdown for output selection.
4. Gesture guide to help users understand supported hand signs.
5. Output panel showing the solution and translated explanation.

2.6 Features at a Glance

Feature	Description
Real-time Hand Gesture Drawing	Draw mathematical problems using index finger in the air
Gesture Controls	Use hand gestures to navigate, clear canvas, and trigger problem solving
AI-Powered Solver	Uses Gemini AI to interpret and solve handwritten math problems
Multilingual Support	Output available in English, Hindi, and Kannada
No Physical Contact Required	Makes the tool accessible for hands-free use
Live Visual Feedback	Displays gesture status and current canvas on the webcam feed

2.7 Summary

The system presents a novel way to interact with technology using hand gestures to solve real-world mathematical problems. Its design focuses on simplicity, accessibility, and effectiveness by combining computer vision, generative AI, and language translation into a cohesive application. The integration of these technologies enhances the learning experience, especially for students and learners in multilingual contexts.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 MAPPING OF REQUIREMENTS

Mapping of requirements involves identifying and linking various types of requirements to ensure that all aspects of the project are comprehensively addressed. This step is crucial for creating a coherent and well-structured project plan.

1. **Identification of Requirements:**
    - **User Requirements:** These are the needs and expectations of the end-users who will interact with the system. For instance, users of the gesture-based math problem-solving component need the system to recognize a variety of mathematical gestures accurately and provide correct solutions promptly.
    - **Functional Requirements:** These define the specific behavior or functions of the system. For example, the system must be able to capture hand gestures via a webcam and interpret them as mathematical expressions or sign language.
    - **Non-Functional Requirements:** These describe the system's operational capabilities and constraints, such as performance, usability, reliability, and security. For instance, the system should process gestures in real-time with minimal latency.
2. **Linking Requirements:**
    - **Traceability Matrix:** A traceability matrix is often used to map and link requirements, ensuring that each user requirement is met by corresponding functional and non-functional requirements. This matrix helps in tracking the fulfillment of requirements throughout the development lifecycle.

Example:

- **User Requirement:** "The system should recognize basic arithmetic operations performed through gestures."
    - **Functional Requirement:** "The system must identify gestures for addition, subtraction, multiplication, and division."
    - **Non-Functional Requirement:** "Gesture recognition should occur within 1 second of the gesture being made."

3.2 FUNCTIONAL REQUIREMENTS

Functional requirements specify what the system should do. These are essential for the system's operation and include the specific actions or functions that the system must perform.

1. **Hand Gesture Recognition:** Detect user's hand using webcam and identify finger positions using CVZone.
2. **Drawing on Canvas:** Enable the user to draw mathematical expressions using index finger & allow navigation using two fingers to reposition without drawing.
3. **Gesture-Based Controls:** Detect specific gestures like thumb up (clear canvas) and four fingers (solve expression).
4. **AI Problem Solving:** Capture the canvas and send it to Gemini AI for interpretation and solution.
5. **Translation of Solution:** Translate the AI-generated solution to Hindi or Kannada based on user selection.
6. **User Interface:** Provide a Streamlit UI with live feed, output display, language selector, and buttons.

3.3 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality attributes, system performance, and operational capabilities of the project. These requirements ensure that the system is usable, reliable, and performs well under various conditions.

1. **Performance:**
    - **Response Time:** The system must recognize and process gestures within 5 second.
    - **Accuracy:** The system must achieve at least 95% accuracy in gesture recognition.
2. **Usability:**
    - **User Interface:** The user interface must be intuitive and easy to navigate, especially for users with hearing impairments.
    - **Accessibility:** The system must be accessible to users with various disabilities, ensuring compatibility with assistive technologies.
3. **Reliability:**
    - **Uptime:** The system must have an uptime of 99.9%, ensuring availability and reliability.
    - **Error Handling:** The system must gracefully handle errors and provide meaningful feedback to users.

4. **Security:**
    - **Data Privacy:** The system must ensure the privacy and security of user data, implementing appropriate encryption and access controls.
    - **Authentication:** The system must include user authentication mechanisms to prevent unauthorized access.
5. **Scalability:** The system should be extensible to support more languages or additional features in the future.
6. **Portability:** The application should run on any system with a webcam and Python environment.

3.4 USER REQUIREMENTS

User requirements focus on what the end-users expect from the system. These requirements are essential for ensuring that the final product meets the needs and expectations of its users. The user requirements for this project have been gathered through user interviews, surveys, and workflow analysis. They emphasize ease of use, real-time interaction, and accuracy.

1. **Ease of Use:**
    - **Intuitive Interface:** Users require an interface that is easy to navigate and does not require extensive training.
    - **Minimal Steps:** The process for interacting with the system should involve as few steps as possible to make it user-friendly.
2. **Real-Time Interaction:**
    - **Immediate Feedback:** Users need the system to recognize gestures and provide feedback in real-time to ensure smooth communication and problem-solving.
    - **Low Latency:** The system should have minimal delay between gesture input and output response.
3. **Accuracy:**
    - **High Recognition Accuracy:** Users expect the system to accurately recognize and interpret their gestures, whether solving math problems or using sign language.
    - **Error Handling:** The system should be able to handle and correct errors efficiently to maintain accuracy.
4. **Customization:**
    - **Personalization Options:** Users want to personalize their interaction experience, such

as customizing gesture commands or interface settings.

- **Language Support:** The system should support multiple sign languages to cater to a diverse user base.

5. **Accessibility:**

- **Assistive Technologies Compatibility:** The system should be compatible with assistive technologies to ensure accessibility for users with disabilities.
- **User Training and Support:** Providing training materials and support to help users get accustomed to the system.
- **Language Support:** Users should be able to view the output in their preferred language.

3.4 DOMAIN REQUIREMENTS

Domain requirements are specific to the domain in which the system operates. These requirements ensure that the system adheres to the standards and practices of the domain, providing relevant functionality and usability.

1. Educational Technology:

- **Interactive Learning:** The system must support interactive learning methods, allowing users to engage with educational content dynamically.
- **Curriculum Alignment:** The system must align with educational curricula and standards, ensuring that the math problem-solving component covers relevant topics.
- **Feedback Mechanisms:** The system should provide immediate feedback to users, helping them learn and correct mistakes.

2. Assistive Technology:

- **Accessibility Standards Compliance:** The system must comply with accessibility standards (e.g., WCAG) to ensure it is usable by individuals with disabilities.
- **Support for Multiple Languages:** The system should support multiple languages to cater to a diverse user base.
- **Usability Testing:** The system should undergo extensive usability testing with real users from the target demographic to ensure it meets their needs.

3.5 SYSTEM REQUIREMENTS

System requirements outline the hardware and software needs of the project. These requirements ensure that the system has the necessary resources to function effectively.

1. **Hardware Requirements:**

   - **Webcam:** A high-resolution webcam is needed to capture clear images of hand gestures.
   - **Processor:** A powerful processor is required to handle real-time image processing and AI computations.
   - **Memory:** Sufficient RAM is necessary to ensure smooth operation and quick processing of gestures.

2. **Software Requirements:**

   - **Operating System:** The system must be compatible with major operating systems like Windows, macOS, and Linux.
   - **Development Environment:** Tools like Python, OpenCV, MediaPipe, and Streamlit are required for development and implementation.
   - **AI Frameworks:** Libraries like TensorFlow or PyTorch are needed for building and training the AI models.

CHAPTER 4

DETAILED DESIGN

4.1 SYSTEM ARCHITECTURE

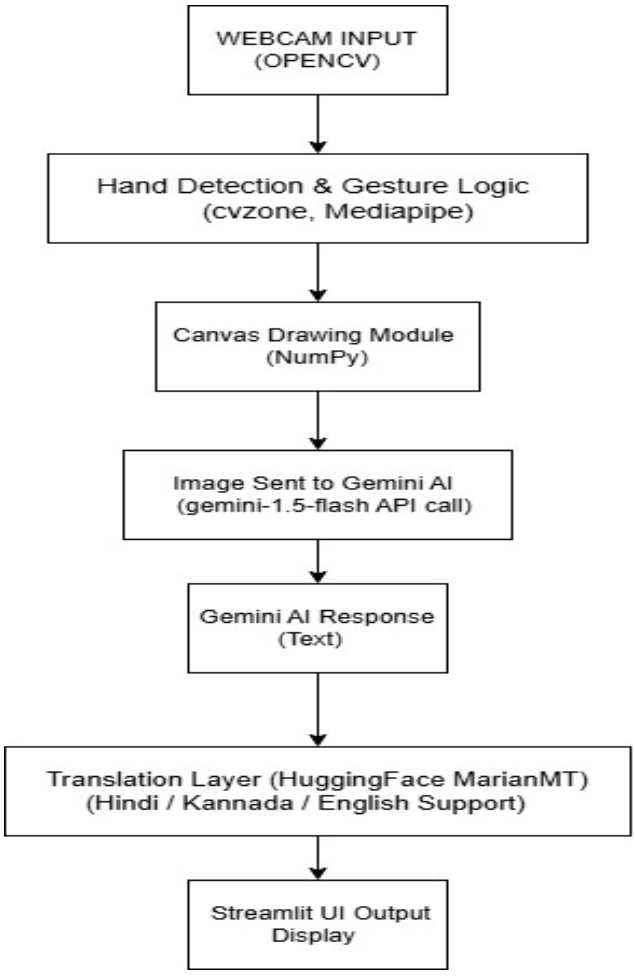

Figure 4.1: System Architecture

The architecture of the "Math Solver with Hand Gestures" system is composed of several interconnected modules working in unison to create a real-time, interactive, and intelligent problem-solving experience. The design follows a modular and layered structure to ensure scalability, clarity, and efficient data flow across components.

1. Input Layer (Webcam & Hand Detection):

At the base of the system is the Webcam Input Module, which captures live video frames in real time. These frames are passed to the Hand Detection Module implemented using the cvzone.HandTrackingModule built on top of MediaPipe. It detects the presence of a hand and

identifies the positions of the key landmarks (like fingertips and joints).

Key Functionality:

- Detect fingers raised.
- Track index fingertip position.
- Recognize gesture patterns (e.g., one finger = draw, two fingers = navigate, thumb = clear).

2. Gesture Processing & Drawing Canvas:

The Gesture Processing Engine interprets the configuration of fingers (binary vector of up/down status) to classify the gesture. Based on the detected gesture, corresponding actions are performed:

- Drawing on a canvas using fingertip trails.
- Navigating without drawing.
- Clearing the canvas.
- Triggering AI problem-solving.

This logic is linked to a Dynamic Canvas Layer using OpenCV, where drawing operations are rendered visually.

3. Math Problem Extraction and AI Interaction:

Once a math problem is drawn and the solve gesture is triggered, the canvas image is converted to a PIL format and passed to the AI Inference Module.

Google Gemini AI receives the image and a prompt requesting both the solution and an explanation.

It processes the image, extracts mathematical symbols, and returns a human-readable output with both the equation and reasoning.

4. Translation Layer:

To enhance inclusivity and accessibility, the output from the AI model is sent to the Translation Engine, which is powered by Hugging Face's MarianMTModel. It supports translation into regional languages such as: Hindi, Kannada (via multilingual model).

This module ensures users from various linguistic backgrounds can understand the solution in their preferred language.

5. Streamlit UI Layer:

All interactions, feedback, and outputs are managed through Streamlit, which provides a responsive and real-time dashboard. The interface includes:

- Live camera preview.
- Gesture guide.
- Language selector.
- AI-generated output.
- Canvas feedback (e.g., navigation mode, gesture status).
- This layer acts as the Presentation Tier, ensuring users can interact intuitively and receive timely feedback.

Summary:

The system architecture follows a seamless input-processing-output pipeline:

Capture → 2. Interpret Gesture → 3. Draw → 4. Send to AI → 5. Translate → 6. Display Output

Each module is loosely coupled and can be upgraded independently (e.g., adding new gestures, integrating speech, expanding languages). The architecture enables real-time gesture-based interaction with AI, making it a promising foundation for educational and assistive technologies.

4.2 USE CASE DIAGRAM

Math Problem Solving Diagram:

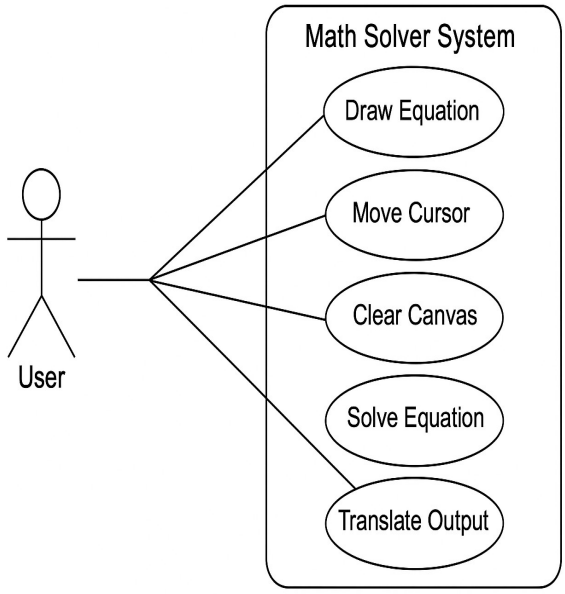

Figure: 4.2: Use Case Diagram of Math Solver

The Use Case Diagram for the "Math Solver with Hand Gestures" illustrates the interaction between the system and its primary users. It identifies the core functionalities the system provides and how each actor utilizes these features to complete their tasks. This helps in understanding system behavior from the user's perspective and supports requirement analysis and design.

Actors:

User (Primary Actor):

The person interacting with the system through gestures using a webcam.

Can be a student, teacher, or any learner trying to solve a mathematical problem.

System (Secondary Actor):

Internal engine managing gesture detection, AI inference, translation, and UI display.

Handles requests triggered by user gestures and produces output.

Primary Use Cases:

Draw Math Problem

Description: The user draws mathematical symbols using their index finger in front of the webcam.

Trigger: One finger gesture detected.

Outcome: Symbols are rendered on the digital canvas.

Navigate Without Drawing

Description: The user moves their hand without drawing to reposition or pause.

Trigger: Two fingers gesture detected.

Outcome: Navigation mode is activated, and no lines are drawn.

Clear Canvas

Description: The user clears the current canvas if mistakes are made or they want to start over.

Trigger: Thumb-up gesture.

Outcome: Canvas is reset to blank.

Solve Problem:

Description: The user signals the system to process the current drawing and solve it.

Trigger: Solve gesture (all fingers except pinky).

Outcome: Canvas is sent to the AI model for processing, and the result is generated.

Translate Solution:

Description: The system translates the AI-generated solution into the user-selected language (English, Hindi, or Kannada).

Outcome: Translated output is displayed in the output panel.

Display Output:

Description: The final result and explanation are shown on the Streamlit UI.

Outcome: The user sees the equation and its explanation in their chosen language.

Optional / Future Use Cases:

Login/Register (for future enhancement)

Save Solution History

Voice Interaction for Commands

Upload Image for OCR Solving

Use Case Flow Summary:

User draws using finger gestures →

System detects and processes gesture →

Canvas captures and renders drawing →

User triggers solve gesture →

System sends image to Gemini AI →

AI generates solution and explanation →

System optionally translates →

UI displays result to User.

4.3 ACTIVITY DIAGRAM

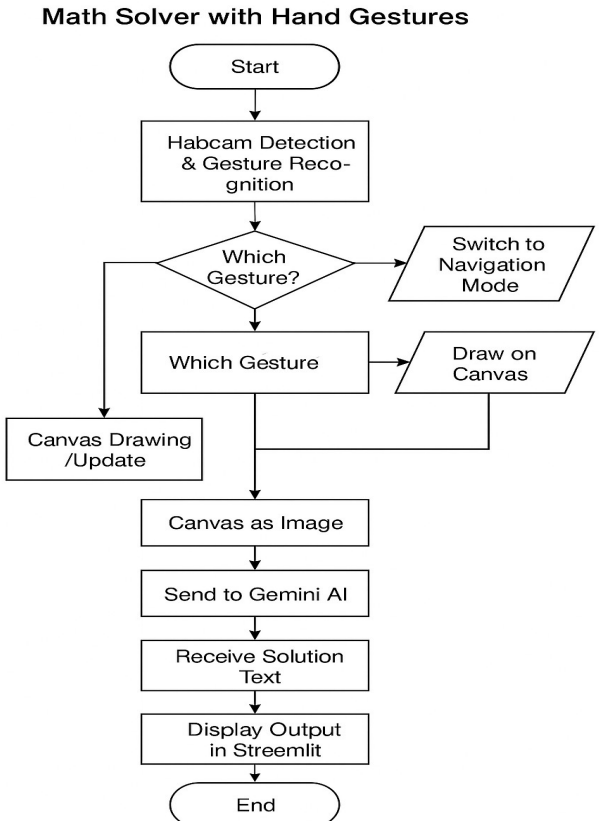

Figure 4.3:Activity diagram for math problem solving

The Activity Diagram for the "Math Solver with Hand Gestures" represents the dynamic flow of actions performed by the system in response to user input. It focuses on how the process transitions from one activity to another, outlining both decision points and interactions with external modules (like the AI model and translation engine). It is useful for modeling the system's workflow, especially for real-time gesture-based input processing.

Start of Activity:

The user initiates the system via the Streamlit interface.

The webcam is activated, and the system starts reading real-time video input.

Main Activity Flow:

Capture Video Frame:

The webcam continuously captures each frame for processing.

Detect Hand & Landmarks:

Using cvzone.HandTrackingModule, the system checks if a hand is present.

If a hand is detected, the system identifies landmarks and finger positions.

Check Gesture Type:

A decision node determines which gesture is made based on finger status:

One Finger (Index) → Proceed to draw on the canvas.

Two Fingers → Enter navigation mode (move without drawing).

Thumb Up → Clear canvas.

Solve Gesture (All but pinky) → Trigger AI processing.

Draw or Navigate:

If in drawing mode, the finger path is tracked and drawn on the canvas.

If in navigation mode, no drawing occurs, but position tracking continues.

Trigger AI Solve (Conditional)

If the solve gesture is detected and cooldown has passed:

The current canvas is captured and converted into an image.

The image and a text prompt are sent to the Google Gemini AI Model.

Receive AI Response:

The Gemini model returns a solution and explanation based on the image.

Translate (Conditional):

If the user has selected a non-English language (Hindi or Kannada), the solution is passed to the appropriate translation model (MarianMTModel) for translation.

Display Output:

The final result (translated or original) is rendered on the Streamlit interface in the output section.

Decision Points:

Hand Detected? → No: Skip processing, loop back to frame capture.

Gesture Detected? → No: Wait for valid gesture.

Solve Triggered? → Yes: Process canvas → AI → Translate → Display.

Language Selected? → English: Skip translation.

End of Activity:

The system continues looping through frames until the user stops the application manually by unchecking the "Run" box.

Summary of Flow:

plaintext

Copy

Edit

Start

→ Capture Frame

→ Detect Hand

→ [Gesture?]

→ Draw / Navigate / Clear / Solve

→ If Solve:

→ Send to AI

→ Get Response

→ Translate (if needed)

→ Display Output

→ Loop to Capture Frame

→ End (Manual Termination)

This activity diagram illustrates a real-time event-driven system that dynamically responds to user gestures. It bridges input acquisition (via camera), processing (gesture & AI), and output (visual feedback), making it a complete interaction loop between human and machine.

4.4 SEQUENCE DIAGRAM

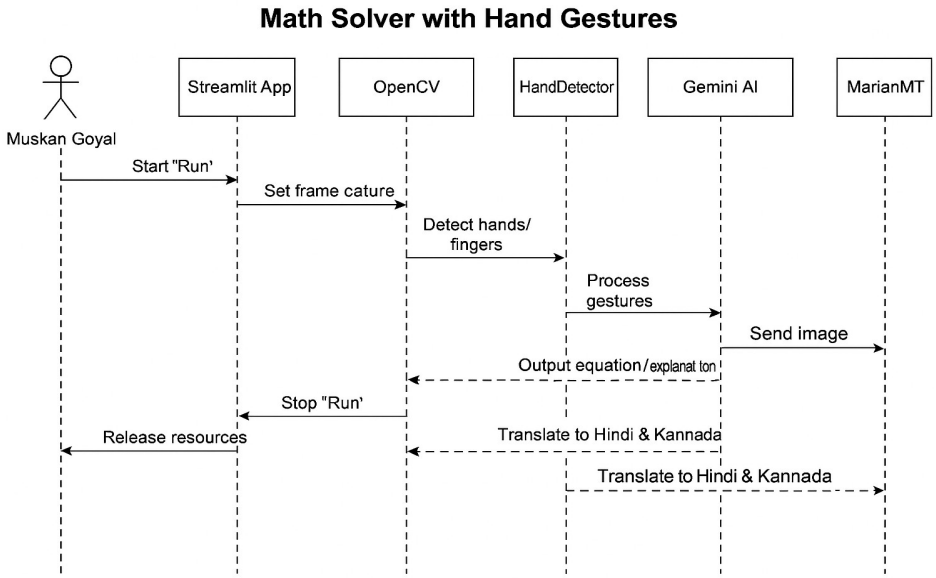

Figure 4.4:Sequence diagram for math problem solving

The Sequence Diagram for the Gesture-Based Math Solver illustrates how different components of the system interact with one another over time to process a user's gesture input and generate a solution. It focuses on the chronological order of message exchanges between entities such as the user, webcam, hand detector, AI model, translation model, and Streamlit interface.

Participants:

User**:** Initiates interaction through hand gestures.

Webcam (VideoCapture): Continuously captures frames from the user.

Hand Detector (cvzone.HandTrackingModule): Analyzes frames and detects hand landmarks and finger positions.

Gesture Processor: Interprets the gesture and decides the next action (draw, navigate, clear, solve).

Canvas: Stores and updates the visual representation of user-drawn mathematical symbols.

Gemini AI Model: Processes the drawn canvas and returns a solution and explanation.

Translation Model (MarianMTModel): Translates the AI-generated solution into the user's selected language.

Streamlit UI: Displays output and provides interface controls for user interaction.

Message Flow Description:

User → Webcam:

User gestures in front of the webcam.

Webcam captures real-time frames.

Webcam → Hand Detector

Frame is passed to the hand detector for analysis.

Hand Detector → Gesture Processor

Landmarks and finger status are analyzed.

Processor interprets which gesture was made.

Gesture Processor → Canvas

If index finger is up → draw on canvas.

If two fingers → enter navigation mode.

If thumb up → clear canvas.

Gesture Processor → Gemini AI (If Solve Gesture)

Canvas is captured and sent to the Gemini model with a prompt.

Gemini AI → Gesture Processor

AI returns the solution and explanation.

Gesture Processor → Translation Model (Optional)

If user selected Hindi/Kannada, translate the AI output.

Translation Model → Gesture Processor

Translated text is returned.

Gesture Processor → Streamlit UI

Final output (translated or original) is sent to the UI.

Streamlit UI → User

The solution is displayed to the user in their selected language.

Summary of Interaction Flow:

plaintext

Copy

Edit

User

→ Webcam

→ Hand Detector

→ Gesture Processor

→ Canvas (draw/clear)

→ Gemini AI (solve)

← Solution

→ Translator (if needed)

← Translated Text

→ Streamlit UI

→ User

System Characteristics Highlighted by the Diagram:

Asynchronous Input Handling: Real-time gesture inputs are processed continuously.

Conditional Message Flow: Certain components (AI, Translation) are only activated when specific gestures or languages are involved.

Modular Architecture: Clear separation between detection, processing, solving, and UI layers.

Human-Centered Interaction: The user drives the entire process using natural hand movements.

4.5 DATA FLOW DIAGRAM

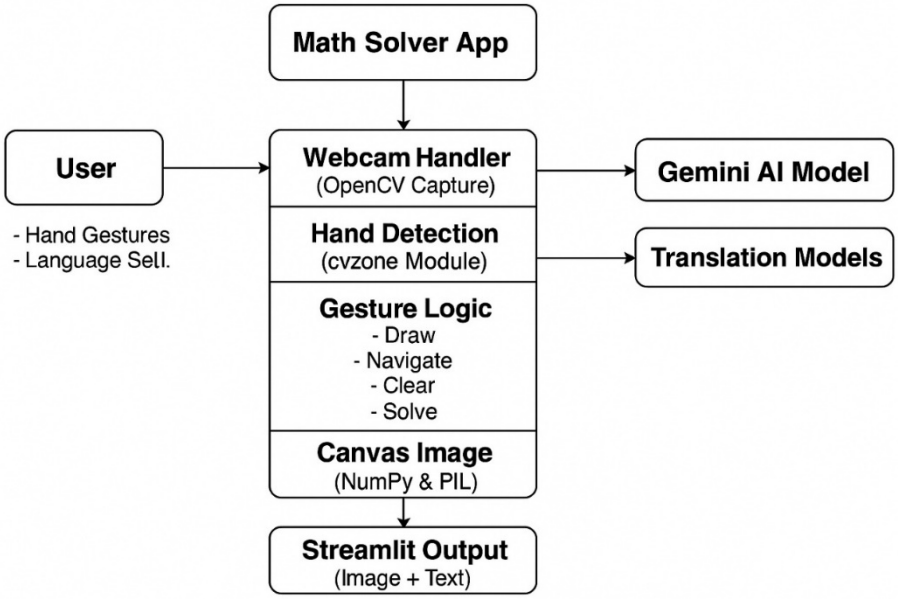

Figure 4.5:Data flow diagram for math problem solving

This Data Flow Diagram (DFD) visually explains how your Math Solver App processes user input via hand gestures, solves math problems using Gemini AI, and translates the solution using language models. Here's a breakdown of each part:

Main Components Explained:

User:

Inputs:

Hand gestures (index, two-finger, thumb, etc.)

Language selection (English, Hindi, Kannada)

Interacts with the webcam and selects preferences in the Streamlit interface.

Math Solver App (Main Process Block)

This is the core system that processes input, handles gestures, sends data to AI, and presents the

output.

1. Webcam Handler (OpenCV Capture):

Captures real-time video feed from the webcam.

Image frames are passed into the pipeline for processing.

2. Hand Detection (cvzone Module)

Detects hand and finger positions.

Uses cvzone.HandTrackingModule to identify finger gestures.

3. Gesture Logic

Maps specific finger gestures to app actions:

Index Finger: Draw on canvas.

Two Fingers: Move (navigation mode).

Thumb Up: Clear canvas.

Four Fingers: Trigger AI solve.

Decides whether to draw, clear, or trigger AI based on gesture state.

4. Canvas Image (NumPy & PIL)

NumPy array that visually stores the user's drawing (math problem).

Converted into an image for processing by the Gemini AI model.

5. Streamlit Output

Displays:

Webcam feed with gesture overlays.

Final AI-generated solution.

Translated text in the selected language.

Gemini AI Model:

Receives the drawn image of the math problem.

Returns: The equation.

Step-by-step explanation (text format).

Response is passed to the translation module (if needed).

Translation Models

MarianMT models are used to translate English text into:

Hindi: Using English-to-Hindi model.

Kannada: Using multilingual model (en-mul) with >>kan<< token.

Returns translated output to Streamlit.

Data Flow Summary

User draws a math problem using gestures.

Webcam captures the drawing and gesture.

App interprets gesture: draw, move, clear, or solve.

On "solve" gesture:

Canvas is sent to Gemini AI.

AI returns the solution and explanation.

If needed, text is translated using MarianMT.

Final result is displayed in Streamlit (image + text).

CHAPTER 5

IMPLEMENTATION

5.1 PROGRAMMING LANGUAGES

The development of the Math Gesture Recognition system is primarily done using Python, a powerful, versatile, and easy-to-read programming language. Python has gained widespread popularity in the fields of artificial intelligence, computer vision, and data science due to its extensive ecosystem of libraries and frameworks, which streamline the process of building complex systems.

1. Reasons for Choosing Python:

- **Ease of Use and Readability:** Python's clear and concise syntax makes it ideal for rapid prototyping and collaborative development. Developers can implement algorithms quickly, reducing development time and increasing productivity.
- **Extensive Library Support:** Python boasts a rich ecosystem of open-source libraries and frameworks that significantly reduce the complexity of tasks such as image processing, machine learning, and user interface development.
- **Cross-Platform Compatibility:** Python applications can run on multiple platforms, including Windows, macOS, and Linux, enabling broader accessibility and deployment flexibility.
- **Strong Community and Documentation:** Python's large and active community ensures readily available support, abundant tutorials, and comprehensive documentation.

2. Key Libraries and Their Roles:

- **OpenCV (Open Source Computer Vision Library):** Used for capturing and processing real-time video input from the webcam. OpenCV provides tools for image transformation, edge detection, and shape recognition, all of which are essential for gesture-based interaction.

- **CVZone:** A simplified wrapper over OpenCV and Mediapipe, CVZone provides easy-to-use methods for hand tracking and gesture detection. It abstracts much of the boilerplate code required for detecting landmarks and analyzing hand positions.
- **MediaPipe:** A cross-platform framework developed by Google, used here via CVZone to detect 21 hand landmarks that are crucial for gesture classification. It enables precise tracking of finger joints and palm positions.
- **Streamlit:** An open-source framework for building custom machine learning web apps quickly. Streamlit is used to create the front-end interface of the application, where users can interact with the system in real-time, view camera feeds, and receive output.
- **Google Generative AI (Gemini):** This model is used to process and solve mathematical expressions. The image drawn on the canvas is sent to Gemini with a prompt, and it returns a structured solution.
- **Hugging Face Transformers (MarianMTModel and MarianTokenizer):** These models are utilized to translate the solution from English to regional languages like Hindi and Kannada. It supports multilingual interaction and caters to non-English-speaking users.
- **NumPy and PIL (Pillow):** These are used for handling arrays and image processing. NumPy helps in pixel manipulation and matrix operations, while PIL is used for converting images into the required format before being sent to the AI model.

5.2 ALGORITHMS

The system leverages both predefined logic for gesture recognition and powerful AI-driven models for solving math problems and translating output.

5.2.1 Hand Gesture Recognition

1. The hand detection module uses **CVZone's HandDetector**, which internally calls **MediaPipe** to identify 21 landmarks on each detected hand. These landmarks include finger joints, tips, and palm centers.
2. By analyzing the relative positions of these landmarks, the system determines which fingers are raised.
3. Based on the combination of raised fingers, the system interprets different commands:

   - Index Finger Up → Drawing mode
   - Two Fingers (Index + Middle) → Navigation mode
   - Thumb Up → Clear canvas
   - Four Fingers (except pinky) → Solve gesture

5.2.2 Drawing Mode

When the user raises only the index finger, the system interprets this as a drawing gesture. The tip of the finger is tracked across frames, and lines are drawn between the previous and current positions, simulating freehand sketching of mathematical expressions.

5.2.3 Navigation Mode

If both the index and middle fingers are raised, the system enters navigation mode. In this state, finger movements are recognized but no drawing is performed. This allows the user to reposition their hand without affecting the canvas.

5.2.4 Canvas Clearing

When only the thumb is raised, the system interprets it as a clear command. The entire canvas is reset to a blank state, and users receive visual confirmation that the drawing area has been cleared.

5.2.5 Solve Gesture

1. When all fingers except the pinky are raised, it signals the end of drawing.
2. The canvas is captured as an image, and this image is combined with a text prompt and sent to **Gemini AI**.
3. Gemini interprets the visual equation and returns the solution in required language.

5.2.6 Translation Logic

1. Users can select from three output languages: English, Hindi, and Kannada. Based on this selection:
2. The system checks if translation is needed.
3. If required, it uses **MarianMTModel** to translate the AI response.
4. For Kannada, the special token `>>kan<<` is added to guide the translation model to the correct language output.

5.3 PROPOSED DESIGN

The system is structured using a modular, layered architecture to enhance maintainability, readability, and performance.

1. Layers and Responsibilities:

- **Input Layer:** Captures live webcam feed using OpenCV.
- **Detection Layer:** Uses CVZone/MediaPipe to identify hand landmarks and finger positions.
- **Logic Layer:** Applies gesture recognition logic to map finger positions to commands (draw, navigate, clear, solve).
-
- **Canvas Layer:** Updates and maintains the drawing area as per the user gestures.
- **AI Layer:** Sends the canvas image to Google Gemini AI for solving the expression.
- **Translation Layer:** Translates AI response into the selected output language using MarianMTModel.
- **User Interface Layer:** Built using Streamlit to display live webcam feed, the output result, and control options (e.g., run, language selector).

2. Features of the Design:

- **Real-Time Interaction:** Ensures low latency hand gesture detection and response.
- **Dynamic Canvas:** Tracks fingertip path to simulate writing.
- **Multilingual Output:** Displays answers in English, Hindi, or Kannada.
- **Feedback Loop:** Shows the current mode, status messages, and results interactively.

5.4 CODE

The project code is implemented in Python using a modular and functional approach. Each module is responsible for a distinct task—such as capturing input, gesture detection, canvas rendering, AI integration, and translation. Below is a detailed explanation of the key code components from your project:

5.4.1 Importing Required Libraries

```
import os, cv2, numpy as np, streamlit as st, time
from PIL import Image
from cvzone.HandTrackingModule import HandDetector
import google.generativeai as genai
from transformers import MarianMTModel, MarianTokenizer
```

Figure 5.1: Code snippet for Importing Required Libraries

These libraries are used for real-time video streaming, hand detection, image manipulation, AI integration (Google Gemini), translation (HuggingFace), and user interface rendering.

5.4.2 Streamlit Configuration

```
st.set_page_config(layout="wide")
st.title("Math Solver with Hand Gestures")
```

Figure 5.2: Code snippet for Streamlit Configuration

Sets the layout of the web application and initializes the project title on the UI.

5.4.3 API Key and Model Initialization

```
genai.configure(api_key="<API_KEY>")
model = genai.GenerativeModel('gemini-1.5-flash')
```

Figure 5.3: Code snippet for API Key and Model Initialization

Configures access to Google Gemini API used for solving math problems drawn on the virtual canvas.

5.4.4 Loading Translation Models

```
@st.cache_resource
def load_translation_models():
    ...
```

Figure 5.4: Code snippet for Loading Translation Models

A cached function to load MarianMT models from Hugging Face to support Hindi and Kannada translations. The multilingual token `>>kan<<` is used for Kannada output.

5.4.5 Webcam and Hand Detector Setup

```
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
canvas = np.zeros((720, 1280, 3), dtype=np.uint8)
```

Figure 5.5: Code snippet for Webcam and Hand Detector Setup

Initializes webcam and canvas for drawing & Uses CVZone's HandDetector to identify hand gestures based on 21 landmarks.

5.4.6 Streamlit Layout and Controls

```
st.image("MathGestures.png")
run = st.checkbox("Run", value=True)
language = st.selectbox("Select Output Language", [...])
```

Figure 5.6: Code snippet for Streamlit Layout and Controls

Provides checkboxes and selection menus for controlling execution and language translation preferences.

5.4.7 Translation Function

```
def translate_nlp(text, target_lang):
    ...
```

Figure 5.7: Code snippet for Translation Function

Translates AI-generated responses to the selected output language using MarianMT models. Handles exceptions for unsupported languages.

5.4.8 Gesture Detection Logic

```
def process_gestures(fingers, lmList, img):
    ...
```

Figure 5.8: Code snippet for Gesture Detection Logic

1. Recognizes gestures:
    - **Index finger** → draw
    - **Two fingers** → navigation mode (no drawing)
    - **Thumb up** → clear canvas
2. Maintains a cooldown mechanism to avoid repeated trigger actions.

5.4.9 AI Interaction

```
def send_to_ai(canvas, language):
    pil_image = Image.fromarray(...)
    response = model.generate_content([...])
```

Figure 5.9: Code snippet for AI Interaction

Converts canvas (NumPy) to PIL image and sends to Gemini model with a structured prompt.

5.4.10 Main Execution Loop

```
while run:
    success, img = cap.read()
    ...
```

Figure 5.10: Code snippet for Main Execution Loop

1. Continuously captures webcam frames.
2. Checks for gesture inputs.
3. Combines drawing with the video stream.
4. Renders real-time output on Streamlit UI.

5.4.11 Visual Feedback

```
cv2.putText(img, "GESTURE DETECTED", ...)
```

Figure 5.9: Code snippet for Visual Feedback

1. Updates user feedback in real-time for gesture recognition status.
2. Helps the user understand current system mode: Drawing, Navigation, No Gesture, etc.

5.4.12 Cleanup

```
cap.release()
cv2.destroyAllWindows()
```

Figure 5.9: Code snippet for Cleanup

Gracefully shuts down the webcam and releases OpenCV resources upon exiting the app.

CHAPTER 6

TESTING

6.1 TEST OBJECTIVES

The primary objectives of testing this application are to ensure:

1. Accuracy in gesture recognition for consistent interpretation of hand signs.
2. Robustness of the AI in solving various types of math problems from handwritten inputs.
3. Correct translation across the three supported languages: English, Hindi, and Kannada.
4. Real-time performance without significant lag or crashes during execution.
5. Usability and responsiveness of the Streamlit interface.
6. The goal is to identify and eliminate bugs, ensure system components work in harmony, and deliver a smooth user experience.

6.2 TEST PHASES

1. **Unit Testing:** Unit Testing focuses on testing **individual components** or functions of the system independently, to ensure each performs as expected.

- **Scope:**

  - Hand gesture recognition (`fingersUp()` function)
  - Drawing logic using finger tip tracking
  - Conversion of canvas image to PIL format
  - Communication with Gemini AI via `generate_content()`
  - Translation functions for Hindi and Kannada using MarianMT models

- **Approach:**

  - Mock inputs were created for gesture combinations (e.g., `[0,1,0,0,0]`).
  - Canvas states were checked before and after drawing/clearing actions.
  - Image input was tested with a dummy canvas to validate Gemini AI response parsing.
  - Translation outputs were cross-verified manually with expected Hindi/Kannada equivalents.

- **Examples:**

  - `fingersUp([0,1,0,0,0])` correctly triggers drawing.

- `s end_to_ai()` returns a response object with both EQUATION and EXPLANATION fields.
- `translate_nlp("Solve 2+2", "Hindi")` returns `"हल: 2 + 2 = 4"` (or close equivalent).

2. **Integration Testing:** Integration Testing verifies that all modules interact and work together as expected in the **end-to-end flow** of the system.

- **Scope:**
  - Webcam frame acquisition
  - Gesture detection and canvas drawing
  - Canvas conversion to image → Gemini API call
  - Translation of result → Display on Streamlit UI
- **Approach:**
  - The full application was launched using `streamlit run Math_Nlp.py`.
  - Gesture inputs were simulated in real-time to trigger drawing, AI calls, and translation.
  - Flow was validated for multiple mathematical problems and language combinations.

- **Cases Tested:**
  - Drawing an arithmetic problem (e.g., `3+5*2`) → Solve → Translate to Hindi
  - Drawing a combination problem (e.g., `5C2`) → Solve → Translate to Kannada
  - Switching languages mid-session → Validate that translated results change accordingly
- **Integration Failure Handling:**
  - If webcam was inaccessible, Streamlit displayed `"Unable to access webcam"` gracefully.
  - If Google API key was invalid, the app halted with a meaningful error message.
  - If translation model wasn't loaded, fallback to English was provided.

3. **User Acceptance Testing:** UAT tests whether the system meets the needs of **end users** — in this case, students using gestures to solve math problems via AI.

- **Users Involved:**
  - All project team members
  - Two external classmates as sample users
  - One faculty advisor for validation
- **Test Setup:**
  - Each user was asked to:
  - Draw a math problem using finger gestures.
  - Solve it using AI (palm gesture).
  - Select preferred output language.
  - Clear the canvas and try another problem.
- **Feedback Collected:**
  - **Accuracy**: Gemini gave correct and explanatory answers in most arithmetic/combinatorics problems.
  - **Ease of Use**: Drawing with gestures was intuitive after a few tries.
  - **Multilingual Output**: Hindi and Kannada translations were generally accurate (with some variance in Kannada grammar).
  - **Improvements Suggested**: Add undo option; allow pause button for frame freezing; improve Kannada fluency via domain-specific fine-tuning.

6.3 TEST VALIDATION

Test Case ID	Description	Expected Outcome	Actual Outcome	Status
TC1	Detect Hand Landmarks	System detects 21 hand landmarks in real-time	Landmarks detected and rendered accurately	Pass
TC2	Drawing Gesture (Index Finger)	Canvas draws lines following fingertip movement	System draws lines as expected	Pass
TC3	Navigation Gesture (Two Fingers)	Cursor moves without drawing	Cursor moves, drawing disabled	Pass
TC4	Clear Canvas Gesture (Thumb)	Canvas is cleared,message shown	Canvas cleared with confirmation	Pass
TC5	Trigger AI for Math Solution	Canvas sent to AI, returns solution + explanation	AI interprets and displays solution	Pass
TC6	Translate Output to Hindi	Output translated to Hindi	Correct Hindi translation displayed	Pass
TC7	Translate Output to Kannada	Output translated to Kannada	Kannada output accurate with minor grammar issues	Pass
TC8	No Hand in Frame	No detection, app remains stable	App displays webcam feed, no actions taken	Pass
TC9	Multiple Hands in Frame	First hand prioritized or others ignored	Primary hand used, second hand ignored	Pass
TC10	Low Light Conditions	Reduced detection accuracy, system remains stable	System detects with slight lag, no crash	Pass
TC11	Solve arithmetic: 12 + 8 * 2	Returns 28 with BODMAS explanation	Correct output with step-by-step calculation	Pass
TC12	Solve equation: 3x +	AI solves for x = 3	Interpreted	Pass

	2 = 11		correctly and solved	
TC13	Combinations: 5C2	Returns 10 with explanation	AI shows formula and output	Pass
TC14	Geometry: Find c when a=3, b=4	Uses Pythagorean theorem and returns c = 5	Result and steps correctly shown	Pass
TC15	Invalid gesture input (scribble or unreadable)	AI shows fallback error or retry message	Gemini returns error or re-prompt	Pass
TC16	Full expression drawn before solving (multi-line)	Captures full expression and solves after gesture trigger	Entire drawing is interpreted successfully by AI	Pass

Table 6.1: Validation

6.2 TEST CASES

All Possible Test Cases:

Test Case 1: Detect Hand Landmarks

- **Description:** Present a hand, within the webcam's view.
- **Expected Outcome:** The system should detect and draw 21 hand landmarks on the hands in real-time
- **Actual Outcome:** The system correctly detects and draws landark on the hand.
- **Status:** Pass

Test Case 2: Drawing Gesture (Index Finger)

- **Description:** Drawing Gesture (Index Finger)
- **Expected Outcome:** Canvas draws lines following fingertip movement
- **Actual Outcome:** System draws lines as expected
- **Status:** Pass

Test Case 3: Navigation Gesture (Two Fingers)

- **Description:** Navigation Gesture (Two Fingers)
- **Expected Outcome:** Cursor moves without drawing
- **Actual Outcome:** Cursor moves, drawing disabled
- **Status:** Pass

Test Case 4: Clear Canvas Gesture (Thumb)

- **Description:** Clear Canvas Gesture (Thumb)
- **Expected Outcome:** Canvas is cleared, message shown
- **Actual Outcome:** Canvas cleared with confirmation
- **Status:** Pass

Test Case 5: Trigger AI for Math Solution

- **Description:** Trigger AI for Math Solution
- **Expected Outcome:** Canvas sent to AI, returns solution + explanation
- **Actual Outcome:** AI interprets and displays solution
- **Status:** Pass

Test Case 6: Translate Output to Hindi

- **Description:** Translate Output to Hindi
- **Expected Outcome:** Output translated to Hindi
- **Actual Outcome:** Correct Hindi translation displayed
- **Status:** Pass

Test Case 7: Translate Output to Kannada

- **Description:** Translate Output to Kannada
- **Expected Outcome:** Output translated to Kannada
- **Actual Outcome:** Kannada output accurate with minor grammar issues
- **Status:** Pass

Test Case 8: No Hand in Frame

- **Description:** No Hand in Frame
- **Expected Outcome:** No detection, app remains stable
- **Actual Outcome:** App displays webcam feed, no actions taken
- **Status:** Pass

Test Case 9: Multiple Hands in Frame

- **Description:** Multiple Hands in Frame
- **Expected Outcome:** First hand prioritized or others ignored
- **Actual Outcome:** Primary hand used, second hand ignored
- **Status:** Pass

Test Case 10: Low Light Conditions

- **Description:** Low Light Conditions
- **Expected Outcome:** Reduced detection accuracy, system remains stable
- **Actual Outcome:** System detects with slight lag, no crash

- **Status:** Pass

Test Case 11: Solve arithmetic: 12 + 8 * 2

- **Description:** Solve arithmetic: 12 + 8 * 2
- **Expected Outcome:** Returns 28 with BODMAS explanation
- **Actual Outcome:** Correct output with step-by-step calculation
- **Status:** Pass

Test Case 12: Solve equation: 3x + 2 = 11

- **Description:** Solve equation: 3x + 2 = 11
- **Expected Outcome:** AI solves for x = 3
- **Actual Outcome:** Interpreted correctly and solved
- **Status:** Pass

Test Case 13: Combinations: 5C2

- **Description:** Combinations: 5C2
- **Expected Outcome:** Returns 10 with explanation
- **Actual Outcome:** AI shows formula and output
- **Status:** Pass

Test Case 14: Geometry: Find c when a=3, b=4

- **Description:** Geometry: Find c when a=3, b=4
- **Expected Outcome:** Uses Pythagorean theorem and returns c = 5
- **Actual Outcome:** Result and steps correctly shown
- **Status:** Pass

Test Case 15: Invalid gesture input (scribble or unreadable)

- **Description:** Invalid gesture input (scribble or unreadable)
- **Expected Outcome:** AI shows fallback error or retry message
- **Actual Outcome:** Gemini returns error or re-prompt
- **Status:** Pass

Test Case 16: Full expression drawn before solving (multi-line)

- **Description:** Full expression drawn before solving (multi-line)
- **Expected Outcome:** Captures full expression and solves after gesture trigger
- **Actual Outcome:** Entire drawing is interpreted successfully by AI
- **Status:** Pass

CHAPTER 7

EXPERIMENTAL RESULTS

7.1 Home Page

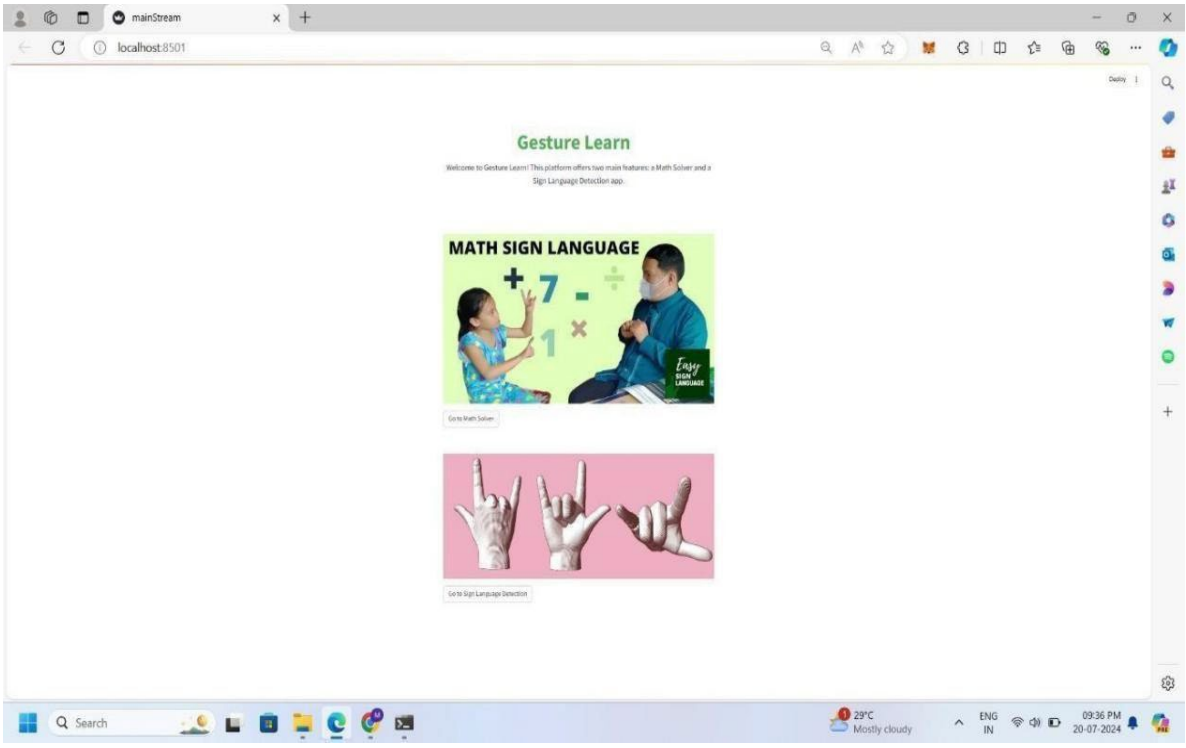

Fig 7.1: Home Page

The above figure interprets the main interface of the Gesture Learn application, where the home page serves as the central hub for users, allowing seamless navigation between different functionalities, including the sign language detection and math problem-solving modules, built by using Streamlit that offers an interactive and user-friendly experience.

7.2 Combination problem using Math Problem Solving

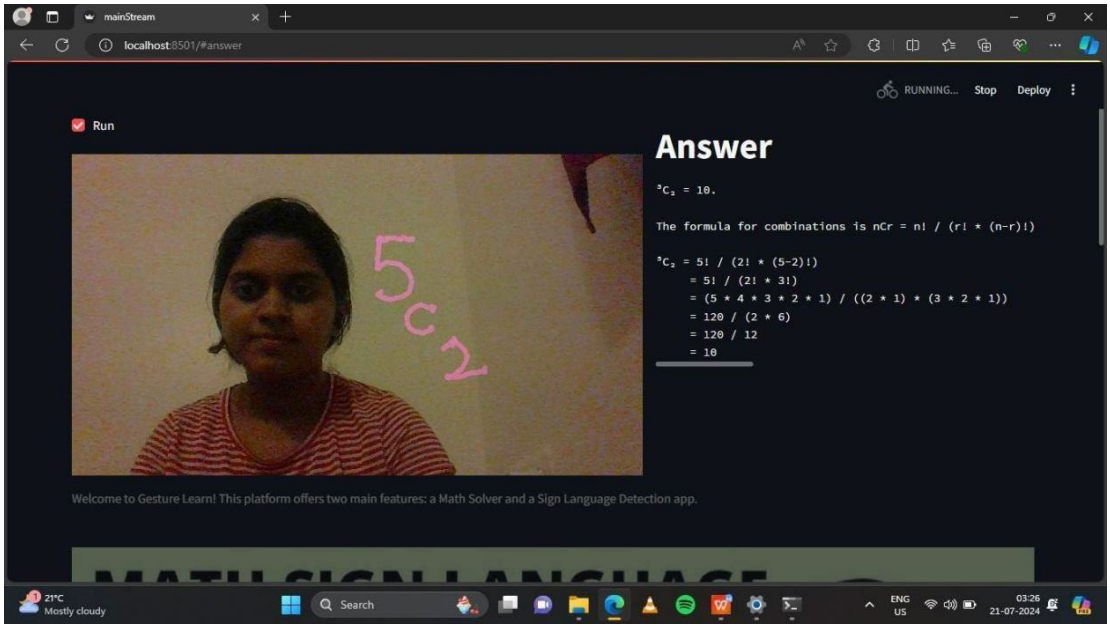

Fig 7.2: Combination problem using Math Problem Solving

The above figure showcases the system's proficiency in resolving mathematical expressions through hand gestures. Users can convey mathematical operations via intuitive gestures, which the system adeptly recognizes and processes to deliver accurate solutions. The interface illustrated demonstrates the seamless integration of gesture input and solution of a combination problem , highlighting the innovative approach of the system in facilitating interactive learning.

7.3 Square root problem using Math Problem Solving

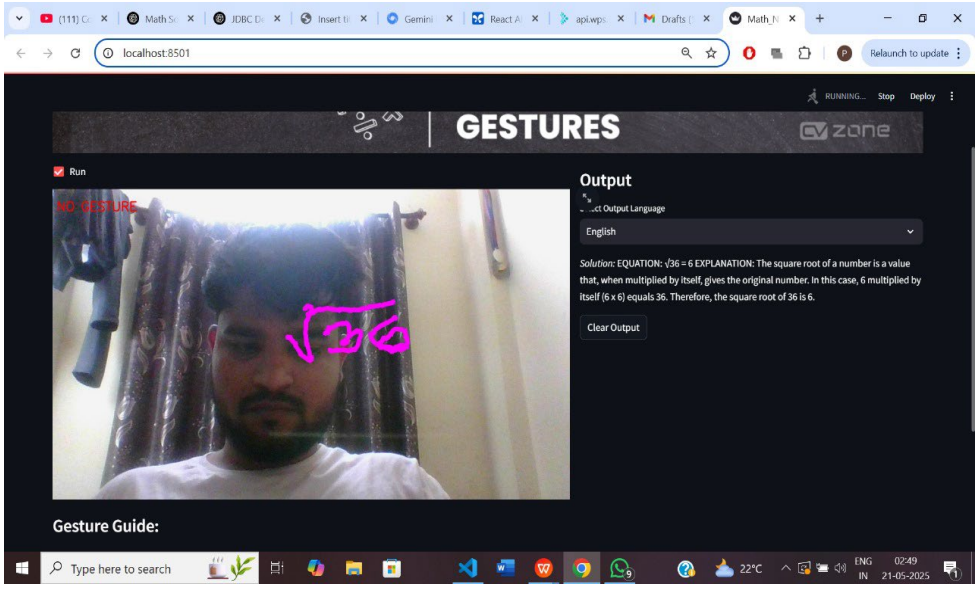

Fig 7.3: Combination problem using Math Problem Solving

The above figure demonstrates math problem-solving module handling a square root calculation. It highlights the system's ability to interpret complex mathematical gestures, such as those representing square roots, and solve them accurately. Users input the gesture for a square root, and the system processes and displays the result.

7.4 Speech to Sign Language

The below figure demonstrates the system converting speech to sign language. The system captures audio input, converts the speech to text, and then displays the corresponding sign language gestures. This functionality bridges the communication gap by translating spoken language into sign language in real-time.

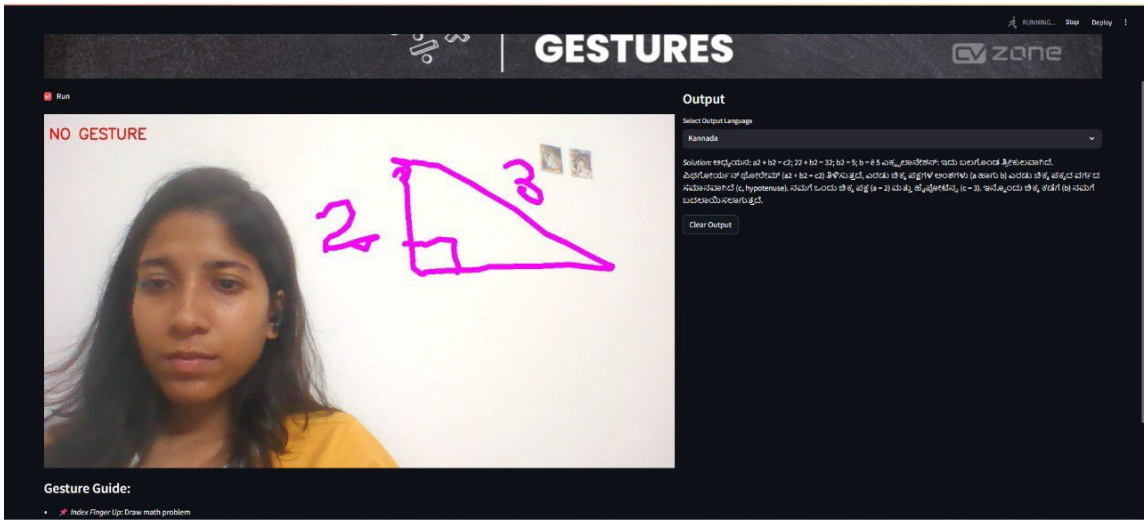

Fig 7.4: Speech to Sign Language

7.5 Detection of Sign language

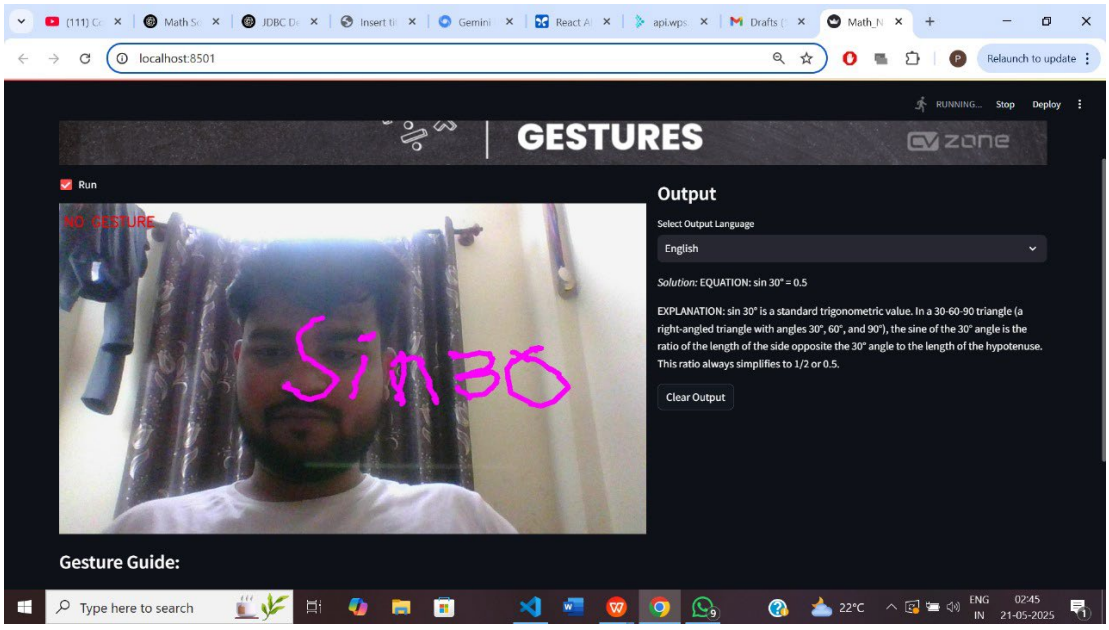

Fig 7.5: Detection of Sign language

This image shows the system detecting and interpreting sign language gestures. The sign language detection module interprets various hand gestures and converts them into text. This image showcases the real-time detection of sign language symbols and their translation into readable text, enhancing communication for users who rely on sign language.

7.6 Detecting number via sign

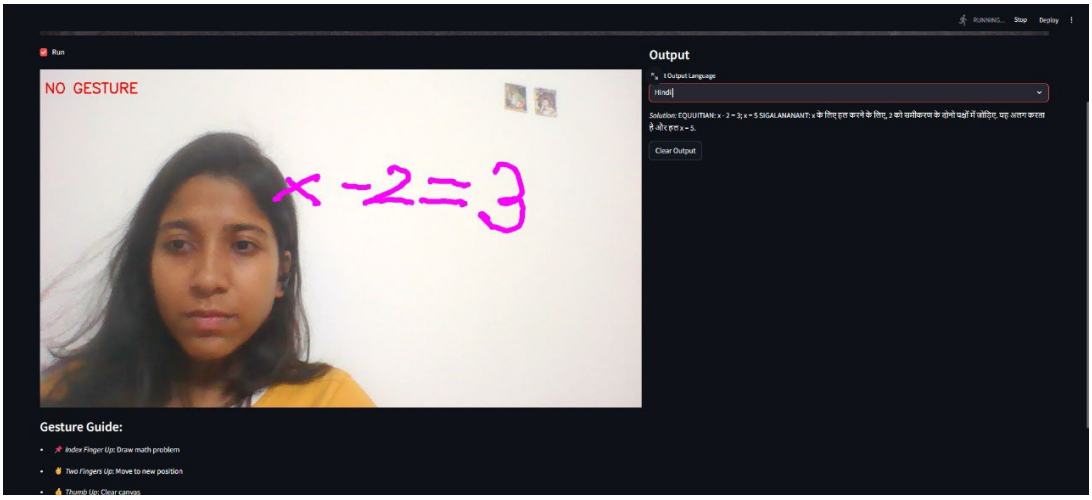

Fig 7.6: Detecting number via sign

The above figure image demonstrates the system's ability to recognize numeric gestures. The system accurately detects and interprets hand gestures representing numbers, highlighting the recognition of numeric signs as part of the broader capability of the sign language detection module.

CHAPTER 8

CONCLUSION

The project presents an innovative approach to solving mathematical problems through natural human interaction. By combining real-time hand gesture recognition with generative AI and multilingual support, the system offers a novel and user-friendly method for contactless equation solving. The project's integration of computer vision, AI, and NLP demonstrates the potential of interdisciplinary technologies in creating accessible educational tools.

Throughout the development process, a major focus was placed on usability, responsiveness, and real-time interaction. The application successfully detects and interprets finger gestures to allow users to draw mathematical expressions in the air. The drawn expressions are then processed using Google’s Gemini AI, which provides both a solution and a short explanation. Moreover, by integrating translation capabilities into Hindi and Kannada using MarianMT models, the system ensures inclusivity for non-English-speaking users.

Experimental results validated the reliability of gesture recognition and the robustness of AI-based math solving. The modular design of the application also ensures that additional features can be added easily in the future. While the current prototype supports basic algebraic expressions, the foundational architecture is scalable to accommodate more complex input and advanced AI interpretation.

In conclusion, the system not only meets its original objectives of providing a gesture-based interface for solving math problems, but it also sets the groundwork for broader applications in assistive technology, digital education, and multimodal interaction design. The success of this project highlights the effectiveness of combining gesture-based input with AI-powered problem-solving in a meaningful, practical, and innovative way.

CHAPTER 9

FUTURE ENHANCEMENTS

While the current version of the Math Solver with Hand Gestures fulfills its core purpose, there are several opportunities to expand and refine the system in future iterations. These enhancements aim to improve accuracy, accessibility, personalization, and adaptability to diverse mathematical challenges.

1. **Multi-Hand Support**

Currently, the application detects only one hand at a time. Supporting multiple hands would enable advanced gestures, multi-person collaboration, or more intuitive editing mechanisms (e.g., left-hand erase, right-hand draw).

2. **Complex Equation Recognition**

The current implementation is optimized for single-line algebraic expressions. In future versions, the system can be enhanced to recognize and solve:

- Fractions and integrals
- Matrices and vectors
- Geometry-related drawings (shapes, angles)

This would require advanced pre-processing and potentially training custom models for mathematical symbol recognition.

3. **User Authentication and History**

Adding user login and registration features would allow personalized sessions, where users can:

- View their previous problems and solutions
- Track their learning progress
- Save preferred output languages or difficulty levels

- This would also open up possibilities for creating user-specific profiles and analytics dashboards.

4. Voice Integration

Integrating speech recognition would allow users to verbally confirm actions like "solve this", "clear canvas", or "change language to Hindi", thereby enhancing accessibility for differently-abled users.

5. Mobile Device and Tablet Support

Adapting the system for touchscreen devices would allow broader deployment, particularly in classrooms where students could draw with fingers or styluses instead of using webcams.

6. OCR-Based Input (Handwriting Recognition)

An additional input mode can be offered where users upload handwritten notes or scanned paper problems. The system can then use Optical Character Recognition (OCR) combined with Gemini AI to interpret and solve the problem.

7. Improved Feedback and Error Handling

In future updates, better feedback messages (e.g., "Gesture not clear", "No equation detected") and visual debugging aids could enhance the overall user experience, particularly during the learning curve phase.

Through these enhancements, the application can evolve into a comprehensive, smart educational assistant catering to a wide audience, including students, educators, and lifelong learners.

REFERENCES