**A PROJECT ON**

****

**SignLingo: Sign Language Translator**

****

**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & TCHNOLOGY**

**Submitted by:**

|  |  |  |
| --- | --- | --- |
| **Ambuj Tripathi** |  | **2017459** |
| **Saatwik Semwal** |  | **2017538** |
| **Sandeep Singh** |  | **2017547** |
| **Suryansh Makhloga** |  | **2017563** |

***Under the Guidance of***

**Mr. Vivek Tomar**

**Assistant Professor**

**Project Team ID: MP23CST024**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**June-2024**



**CANDIDATE’S DECLARATION**

We hereby certify that the work is being presented in the Project Report entitled **“SignLingo : Sign Language Translator”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering and submitted to the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun is an authentic record of my work carried out during a period from **August-2023 to May-2024** under the supervision of **Mr. Vivek Tomar, Assistant Professor** , Department of Computer Science and Engineering, Graphic Era (Deemed to be University).

The matter presented in this dissertation has not been submitted by me/us for the award of any other degree of this or any other Institute/University.

|  |  |
| --- | --- |
| **Ambuj Tripathi** | **2017459** |
| **Saatwik Semwal** | **2017538** |
| **Sandeep Singh** | **2017547** |
| **Suryansh Makhloga** | **2017563** |

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

**Signature** **Signature**

**Supervisor** **Head of the Department**

**External Viva**

**Name of the Examiners: Signature with Date**

1.

2.

**Abstract**

Sign language is key for the deaf or hearing-impaired, but traditional reliance on interpreters presents limitations in both availability and precision. Historical methods for sign translation faced unique challenges from the visual-spatial aspects of sign languages, often failing to accurately recognize gestures or adapt to varying contexts and necessitating instant translation. This caused constraints in usability and hindered the application on diverse platforms such as mobile devices due to high computational requirements. This research introduces a sophisticated framework for sign language recognition, integrating the Mediapipe Landmark system with a combination of machine learning models—Long Short-Term Memory networks, Convolutional Neural Networks, and Random Forest—to enhance the precision of detecting hand signs and movements in real-time and across different settings. The study outlines the processes for data gathering, preprocessing, and classifying gestures and static signs, advancing communication technologies for those with hearing impairment.

**Keywords**: Static Gestures, Dynamic Gestures, Cross-Model Comparison, Assistive Technologies, Classification

**Acknowledgement**

Any achievement, be it scholastic or otherwise does not depend solely on individual effort but on the guidance, encouragement, and cooperation of intellectuals, elders, and friends. Several personalities in their capacity have helped me in carrying out this project work.

Our sincere thanks to project guide **Mr. Vivek Tomar, Assistant Professor,** Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable guidance and support throughout the course of project work and for being a constant source of inspiration.

We extend our thanks to **Prof. (Dr.) Guru Prasad M.S.**, Project Coordinator, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), for his valuable suggestions throughout all the phases of the Project Work.

We are extremely grateful to **Prof. (Dr.) D. P. Singh**, HOD of the Computer Science and Engineering Department, Graphic Era (Deemed to be University), for his moral support and encouragement.

We thank the **management of Graphic Era (Deemed to be University)** for the support throughout the course of our Bachelor’s Degree and for all the facilities they have provided.

Last, but certainly not least we thank all teaching and non-teaching staff of Graphic Era (Deemed to be University) for guiding us on the right path. Most importantly we wish to thank our parents for their support and encouragement.

Ambuj Tripathi 2017459

Saatwik Semwal 2017538

Sandeep Singh 2017547

Suryansh Makhloga 2017563

**Table of Contents**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Contents** | **Page No.** | | | | | | Abstract | i | | | | | | Acknowledgement | ii | | | | | | Table of Contents | iii | | | | | | List of Tables | vi | | | | | | List of Figures | ix | | | | | | **Chapter 1 Introduction** | **1-3** | | | | | | 1.1 Project Introduction | 1 | | | | | | 1.2 Problem Statement | 2 | | | | | | 1.3 Objectives | 3 | | | | | | **Chapter 2**  **Literature Survey/ Background** | **4-5** | | | | | |  |  | | | | | | **Chapter 3 Software Design**   |  | | --- | | 3.1 Architecture Overview | | 3.2 Use Case Diagram | | 3.3 Data Flow Diagram  3.4 Component Design  3.4.1 Frontend (React.js)  3.4.2 Backend (Node.js, Express.js)  3.4.3 Key Modules  3.4.4 Machine Learning Integration  3.5 Testing Strategy | | |  | | | | | | | **Chapter 4**  **Requirements and Methodology** | | | **13-16** | | | | | | | 4.1 Requirements | | |  | | 4.1.1 Hardware Requirements | | |  | | 4.1.2 Software Requirements | | |  | | 4.2 Methodology | | |  | |  | | |  | | **Chapter 5**  **Coding /Code Templates** | | | **21-25** | | | 5.1 Detect Page | | |  | | | 5.2 Dashboard Code  5.3 Model Training  5.4 Login/Logout’ | | |  | | **Chapter 6**  **Testing** | | |  | | 6.1 Unit Testing | | | **41** | | | | | 6.1.1 User Authentication | | |  | | | | | 6.1.2 Gesture Recognition  6.2 Integration Testing  6.2.1 User Profile Management  6.2.2 Authentication and User Data  6.3 Functional Testing  6.3.1 Translational Accuracy  6.3.2 User Interface  6.4 Usability Testing  6.4.1 User Experience  6.5 Regression Testing  6.5.1 Feature Updates  6.5.2 Bug Fixes  **Chapter 7 Result and Discussion**  7.1 Smooth interface creation using React.js  7.2 Dashboard  7.3 User Authentication  7.4 Tutorial  7.5 User Data  7.6 Understanding of Mediapipe Based Model  **Chapter 8 Conclusion and future work**  **Details of Research Publication**  **References** | | |  | | | | |  | | |  | | | | |  |
|  | **List of Tables**   |  |  |  | | --- | --- | --- | | **TABLE No.** | **TITLE** | **PAGE No.** | | 4.1.1 | Hardware Requirements | 20 | | 4.1.2 | Software Requirements | 20 |   **List of Figures**   |  |  |  | | --- | --- | --- | | **FIGURE No.** | **TITLE** | **PAGE No.** | | 3.1 | Use case diagram of SLR- WebApp | 14 | | 3.2 | Data Flow Diagram of SLR- WebApp | 15 | | 4.1 | Working of Mediapipe | 21 | | 7.1 | Website Interface of the project | 32 | | 7.2 | User Dashboard | 33 | | 7.3 | User Login Page | 34 | | 7.4 | Tutorial for learning sign and gestures | 34 | | 7.5 | User Data Stored at Firebase | 35 | | 7.6 | SSD Architecture | 36 | | 7.7 | MobileNet Architecture | 37 | |  |

**Chapter 1**

**Introduction**

In the following sections, a brief introduction and the problem statement for the work has been included.

* 1. **Project Introduction**

In our swiftly evolving technological landscape, the imperative for inclusive and innovative solutions has never been more pronounced, particularly in addressing the communication needs of individuals with disabilities, specifically those who are deaf or hard of hearing. The prevalence of spoken language creates significant barriers for this community, necessitating the development of a crucial tool – a computer-based Sign Language Translator.

Despite remarkable strides in voice recognition technology, a glaring gap exists in sign language recognition and translation. Our project stands at the forefront of bridging this communication divide, with a focus on empowering individuals with hearing impairments and the wider population through a state-of-the-art Sign Language Translator. Integrating cutting-edge computer vision, machine learning, and gesture recognition algorithms, our innovative system enables users to convey messages through sign language gestures, offering real-time translation with exceptional precision.

This multifaceted technical approach encompasses gesture detection, sophisticated natural language processing, and a commitment to user-friendly interfaces. Real-time processing facilitates instantaneous communication between sign language users and those relying on spoken or written language.

Beyond immediate impact, the project aligns with a broader study dedicated to American Sign Language (ASL), aiming to create an advanced model for recognizing fingerspelling-based hand gestures. Acknowledging ASL's complexity and its pivotal role for Deaf and Hard of Hearing (DHH) individuals globally, our approach employs the latest technology, including Google's Mediapipe. By integrating a Random Forest Classifier, LSTM, and Convolutional Neural Network (CNN), our model captures intricate hand-shape and trajectory features, ensuring robust recognition in challenging real-world conditions.

By harnessing technology, our project champions the causes of communication, understanding, and inclusion, paving the way for a more accessible and interconnected world.

* 1. **Problem Statement**

The problem at hand is the absence of a readily available, commercial solution for sign language recognition and translation, which hinders effective communication for individuals who primarily use sign language as their means of expression. While there have been promising developments in the field of computer vision and machine learning, there is a critical need for a robust, user-friendly, and widely accessible Sign Language Translator that can accurately interpret and display sign language gestures in real time.

Our major project aims to address this issue by developing an innovative computer-based solution that empowers both individuals with hearing impairments and those without to engage in seamless and meaningful communication. In our proposed system, users will be able to convey messages through sign language gestures, which will be captured by their camera and processed by our technology. The system will then detect and translate these gestures into a comprehensible form, allowing the intended message to be displayed to the user.

By creating such a Sign Language Translator, we aspire to foster greater inclusivity and accessibility in the digital age, enabling differently-abled individuals to communicate effectively with the broader population. This project not only aligns with the principles of technological advancement but also embodies the spirit of social responsibility, ensuring that no one is left behind in the digital era.

* 1. **Objectives**

The proposed work objectives are as follows:

**1. Develop a Robust Gesture Detection System:** The primary objective of this project is to create a reliable gesture detection system that accurately identifies and tracks sign language gestures using computer vision techniques.

**2. Implement Accurate Gesture Classification:** - Develop and train a CNN-LSTM model for real-time classification of sign language gestures, integrating features extracted using MediaPipe Holistic for spatial and temporal feature learning, ensuring high accuracy and performance.

3. Ensure User-Friendly Interaction: Develop an intuitive and user-friendly interface that allows individuals to interact effortlessly with the system using camera-equipped devices.

**4. Optimize for Real-Time Processing:** Engineer the system for real-time processing, enabling instantaneous communication between sign language users and speakers of spoken or written language.

**5. Compatibility to Various Background:** Develop a robust system capable of effectively handling diverse lighting and background conditions during predictions.

**Chapter 2**

**Literature Survey/ Background**

**In the realm of machine learning (ML) and deep learning applications in assistive technologies, there is a persistent drive for advancements to meet the demand for sign language recognition on a global scale. Effective communication hinges on the clarity of the message conveyed, and recent innovations in sensors, hardware, and computer vision have broadened the applications of assistive technologies, particularly in the domain of communication and sign language recognition.**

**Abdulhussein M. Al-Kaabi, Zuhair S. Al-Taie, and Ali H. Mahmood explored edge detection and grayscale techniques to anticipate hand movements and train a predictive model. However, the two-dimensional nature of grayscale images imposes limitations on accurately identifying landmark features crucial for classifying various gestures. Jia Chong, Xiangmin Xu, and Lishan Kang utilized a Leap Motion controller sensor-based approach to predict hand movements, albeit at a significant cost due to the required hardware. Similarly, Laura Rinalduzzi, Giuseppe Vendramin, and Michele Paci employed sensors in gloves to ensure higher precision in hand movement tracking. Despite their accuracy, the practicality and cost-effectiveness were hindered by the hardware requirements and discomfort from constant glove usage.**

**Feature recognition methods based on color and contour extraction, as demonstrated by Harikrishna Haria, Amit Kumar, and Shubham Verma, fell short in predicting intricate hand gestures, focusing instead on simpler tasks like finger counting. Pei Xu implemented convolutional neural networks (CNN) aimed at real-time hand sign detection but lacked a specific focus on sign language. Abhishek B. introduced a 3D CNN algorithm for hand motion recognition; however, its static image-based design limited its applicability to real-time scenarios.**

**Current sign language recognition systems encounter challenges in achieving faster and more accurate tracking while maintaining lightweight systems and adapting to diverse backgrounds and lighting conditions. To tackle these challenges, recent research has centered on hand landmark detection utilizing the Mediapipe library. This approach, as highlighted by researchers like Matthias Grundmann, Tal Remez, and Manfred Eric Mueller , facilitates the creation of hand skeletons in diverse environments, aiming to improve the flexibility and accuracy of machine learning pipelines for sign language recognition on everyday devices.**

**Chapter 3**

**Software Design**

**3.1 Architecture Overview**

The sign language translator application is built using React for the user interface, with integrated LSTM models and MediaPipe for image processing. The architecture is designed to be modular, scalable, and maintainable, ensuring seamless communication between the frontend and the image processing backend. This setup allows for real-time sign language gesture recognition and translation.

**3.2 Use Case Diagram**

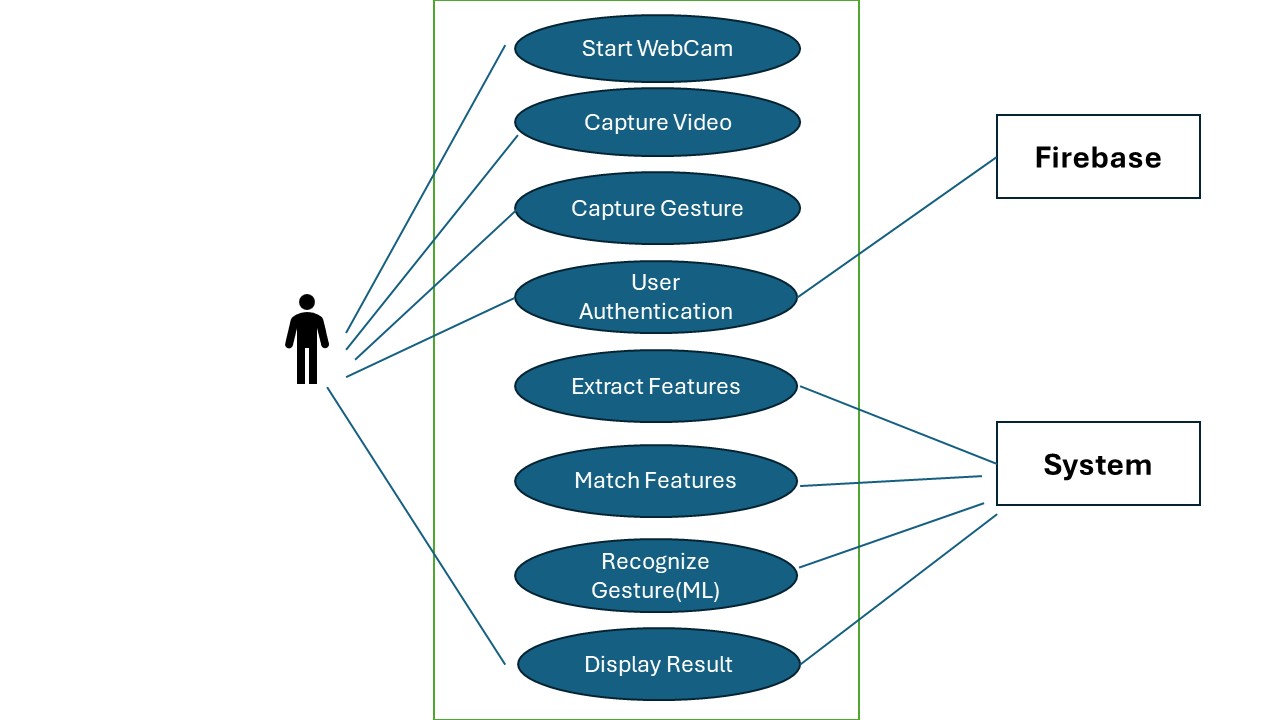
****

Fig-3.1 Use case diagram of SLR-WebApp

This is the proposed system in which actors are driver and the passenger who is using the

Service.

**3.3 Data Flow Diagram**

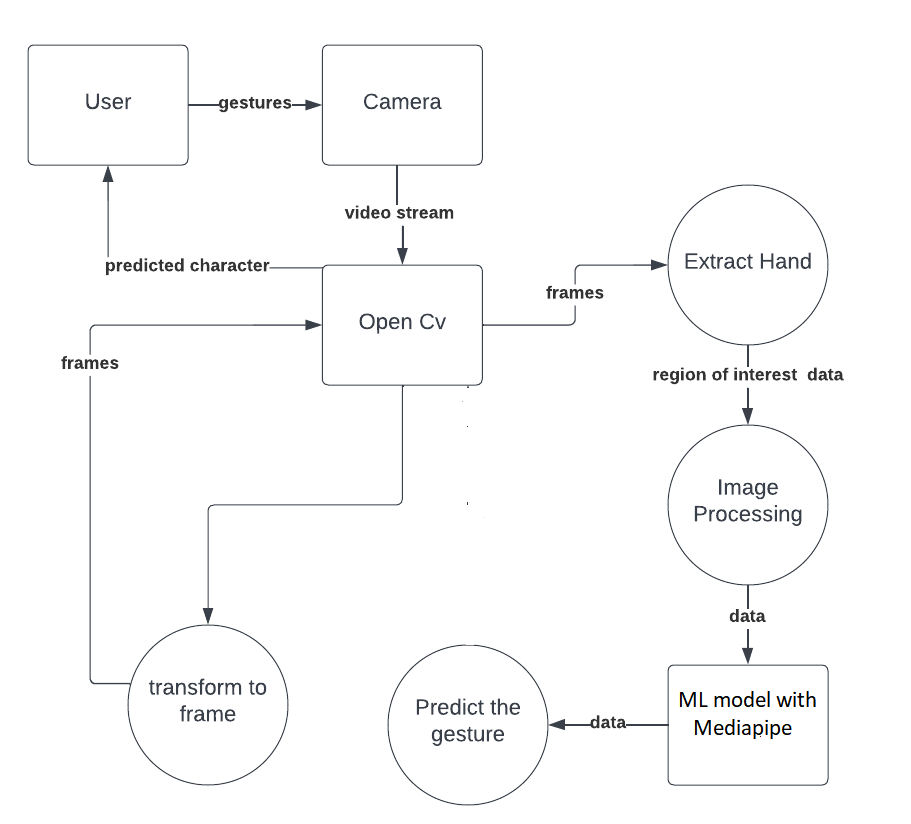


Fig-3.2 Data Flow Diagram of SLR-WebApp

**3.4 Component Design**

* + 1. **Frontend (React.js)**
* **User Interface Components:**
* **Login and Registration Forms:** Interfaces for user authentication, allowing users to sign up and log in.
* **Dashboard:** A central interface where users can access various functionalities of the application, such as starting gesture recognition, viewing their profile, and accessing settings.
* **Gesture Recognition Interface:** The main interface where users can interact with the gesture recognition system. This includes real-time video feed from the device's camera, overlaid with detected gestures.
* **Profile:** Interfaces for users to login and view their activity or logout.
* **State Management:**
* **Redux Toolkit:** Utilized for managing the application's state. Actions and reducers are defined to handle state changes related to user authentication, gesture data, and UI states.
* **APIs and Integration:** Functions to interact with backend services for user authentication, retrieving and updating user data, and sending gesture data for processing.
* **MediaPipe Integration:** Integration with MediaPipe's gesture recognition model to process the real-time video feed and identify gestures.
* **Styling and Responsiveness:**
* **Styled Components:** For consistent styling across the application, ensuring a responsive and user-friendly interface.

**3.4.2 Backend (Node.js, Express.js)**

**Structure:**

* **Server Setup:** The core server setup using Node.js and Express to handle API requests and serve responses.
* **Database:** Firebase, a NoSQL database to store user data, authentication credentials, and gesture data.
* **API Endpoints:**

**User Authentication:**

* **POST /register:** Endpoint to handle user registration.
* **POST /login:** Endpoint to handle user login.
* **GET /logout:** Endpoint to handle user logout.

**Gesture Data Processing:**

* **POST /gestures:** Endpoint to receive gesture data from the frontend and process it using MediaPipe models.
* **Middleware:**
* **Authentication Middleware:** Middleware to protect routes and ensure that only authenticated users can access certain endpoints.
* **Error Handling Middleware:** Middleware to handle errors and send appropriate responses to the frontend.
* **Integration with MediaPipe Models:** Functions to process gesture data using MediaPipe's gesture recognition models. These functions take the input from the frontend, run the MediaPipe models, and return the recognized gestures.
* **Security:**
* **JWT Tokens:** For secure authentication and session management.
* **Encryption:** Secure storage of user passwords using encryption algorithms like bcrypt.

**3.4.3 Key Modules:**

* **Sign Recognition:** It efficiently translates the sign and gestures shown by user in real-time with accuracy bar to show how close is gestures shown by user to the predicted result.
* **User Activity:** Records and display user activity on app. Time spent and gestures practiced by user with frequency are displayed along with ranking in leaderboards.
* **Tutorial:** Keeps showing random signs and gestures with its translation which user can see and practice.
* **Authentication:** Handles user registration, login, password reset, and JWT token management.

**3.4.4 Machine Learning Integration**

**Dataset Preparation:** The dataset of sign language gestures is organized and preprocessed. This involves normalizing the data and splitting it into training, validation, and test sets.

* **CNN (Convolutional Neural Network):** Processes images and extracts spatial features from each frame of sign language gestures. Convolutional layers apply filters to detect edges, textures, and shapes, while max-pooling layers reduce spatial dimensions, preserving essential features and reducing computational complexity. The CNN output is a high-dimensional feature vector representing the spatial characteristics of the hand gestures.
* **LSTM (Long Short-Term Memory):** Analyzes the sequential data from CNN to understand the temporal dynamics of gestures. LSTM layers learn dependencies between consecutive frames, capturing the context and continuity of gestures. By maintaining memory of previous frames, LSTMs predict the current gesture based on both current input and past information.
* **MediaPipe:** Captures hand landmarks and pose estimations in real-time, providing precise coordinates of key points on the hands and body. MediaPipe’s holistic model estimates positions of hands, wrists, and fingers, providing a comprehensive representation of gestures. These landmarks are fed into the CNN for spatial feature extraction and then into the LSTM for temporal analysis. The gesture recognition model is trained using the MediaPipe Model Maker. The process includes defining hyperparameters, setting up training options, and running the training iterations. It includes pretrained palm detection, hand landmark, and gesture embedding model working together.

Together, CNN, LSTM, and MediaPipe form a robust pipeline for translating sign language into text, processing real-time video input, extracting relevant features, and interpreting gestures accurately for efficient sign language translation.

**3.5 Testing Strategy**

* **Unit Testing:** Test individual components and services, such as the React UI components, LSTM processing units, and MediaPipe integration functions.
* **Integration Testing:** Test interactions between different modules and services, ensuring seamless communication between the React frontend, the CNN-LSTM model, and MediaPipe for accurate gesture recognition.
* **End-to-End Testing:** Test the entire workflow from video input through MediaPipe, gesture classification by CNN and LSTM, to the final display of translated text on the UI, ensuring the system works as a cohesive whole.
* **Performance Testing:** Ensure the app performs well under load, maintaining real-time responsiveness and accuracy in gesture recognition, even with multiple users and high volumes of data.

By following this comprehensive testing strategy, the sign language translator can deliver a robust, scalable, and user-friendly experience, ensuring efficient and accurate sign language interpretation.

**Chapter 4**

**Requirements**

**Hardware and Software Requirements**

**4.1 Requirements**

**4.1.1 Hardware Requirements**

|  |  |  |
| --- | --- | --- |
| Sl. No | Name of the Hardware | Specification |
| 1 | Server infrastructure | Latest web server (ex. Node.js webserver ) |
| 2 | Storage | Cloud-Based Database (Firebase) |
| 3 | Security | Firewall and SSL certificates for secure data transmission |
| 4 | GPU | NVIDIA GPU (e.g., NVIDIA GTX 1080 or higher) |

**4.1.2 Software Requirements**

|  |  |  |
| --- | --- | --- |
| Sl. No | Name of the Software | Specification |
| 1 | Development Framework | React, Redux toolkit , ML Algorithms |
| 2 | Programming Languages | JavaScript, Python |
| 3 | Api & Third Party Service’s | MediaPipe, TensorFlow, Google Colab, Google Drive API |
| 4 | Version Control | Git/GitHub |
| 5 | Development Tools | IDE for coding , testing ,  debugging(VS code ) |
| 6. | Browser | Latest version of Google Chrome or Firefox for testing web applications |

**4.2 Methodology**

**Methodology**

Our methodology for the sign language recognition project comprises a meticulous process involving data collection and processing, followed by model training and implementation of three distinct machine learning models: Random Forest, CNN, and LSTM.

**Data Collection and Processing:**

We initiated our study by meticulously gathering pertinent data crucial for training and evaluating the models. Sign language motions were recorded using a high-resolution webcam, with MediaPipe Holistic and OpenCV serving as integral tools for data acquisition. Leveraging the capabilities of MediaPipe Holistic, estimations for pose, hand, and facial landmarks were obtained, essential for accurate sign language interpretation. Real-time data collection entailed capturing predetermined sets of sign language motions, where only the essential components extracted by MediaPipe Holistic were retained to optimize storage space, while discarding redundant video data.

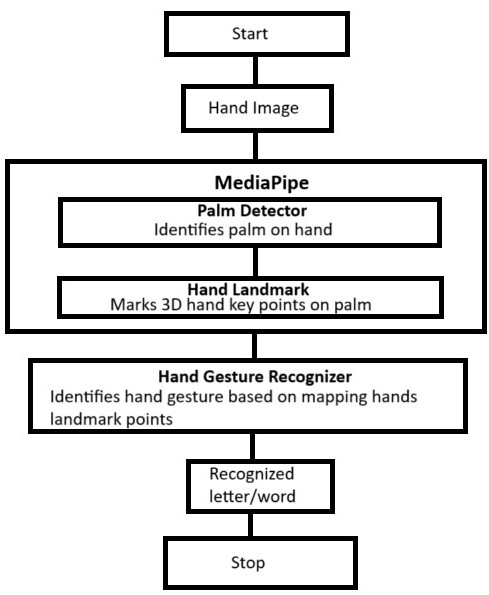


Fig-4.1 Working of Mediapipe

**Model Training:**

Data preprocessing played a pivotal role in preparing the collected data for model training. This involved meticulous steps such as standardizing the array size of video frames and extracting crucial points from each frame. Subsequently, the datasets were partitioned into 80:20 training and testing sets to ensure the models' accuracy and robustness.

* **Random Forest Model:** Leveraging hand landmarks extracted by MediaPipe Holistic, we meticulously normalized the data, utilizing it as features within the dataset. An ensemble learning approach was adopted, employing a collection of decision trees to enhance accuracy and generalization. Hyperparameter tuning was conducted to optimize model performance, followed by evaluation based on accuracy metrics.
* **CNN Model:** The preparation of the dataset involved loading sign language images and encoding their labels numerically. The CNN architecture was meticulously designed, incorporating convolutional and max-pooling layers to facilitate robust feature extraction and classification. Post-compilation with suitable optimizer and loss functions, the model underwent rigorous training on the training data and subsequent evaluation on the testing set to validate its performance.
* **LSTM Model:** LSTM networks were judiciously employed to address challenges associated with capturing long-range dependencies within sequences. Hand key locations extracted using MediaPipe Holistic and OpenCV formed feature arrays concatenated to create input for each frame. The model underwent compilation and training using stochastic gradient descent backpropagation, with subsequent evaluation on a testing set to assess its efficacy in real-time sign language recognition.

**MediaPipe-Based Deep Learning Model:**

Our Sign Language Recognition project employs advanced deep learning techniques within the MediaPipe framework. The model includes palm detection, hand landmark detection, and gesture embedding, each powered by Convolutional Neural Networks (CNNs). For palm detection, we use efficient real-time architectures like Single Shot MultiBox Detector (SSD) or YOLO. The hand landmark model utilizes MobileNet for precise keypoint detection. Gesture classification is achieved using a fully connected network trained with hand embeddings.

Standard backpropagation is applied with appropriate loss functions—localization and classification losses for palm detection, and Mean Squared Error (MSE) for hand landmarks. Data augmentation ensures robustness, and optimization algorithms like Adam enhance training. Transfer learning with pretrained networks and quantization techniques ensures the model's efficiency on mobile and edge devices. These methods ensure our model is accurate and capable of real-time performance.

This systematic, thorough, and comprehensive approach ensured the development of efficient machine learning models for precise sign language recognition, effectively addressing challenges and leveraging the strengths of each model architecture.

**Chapter 5**

**Coding/Code Templates**

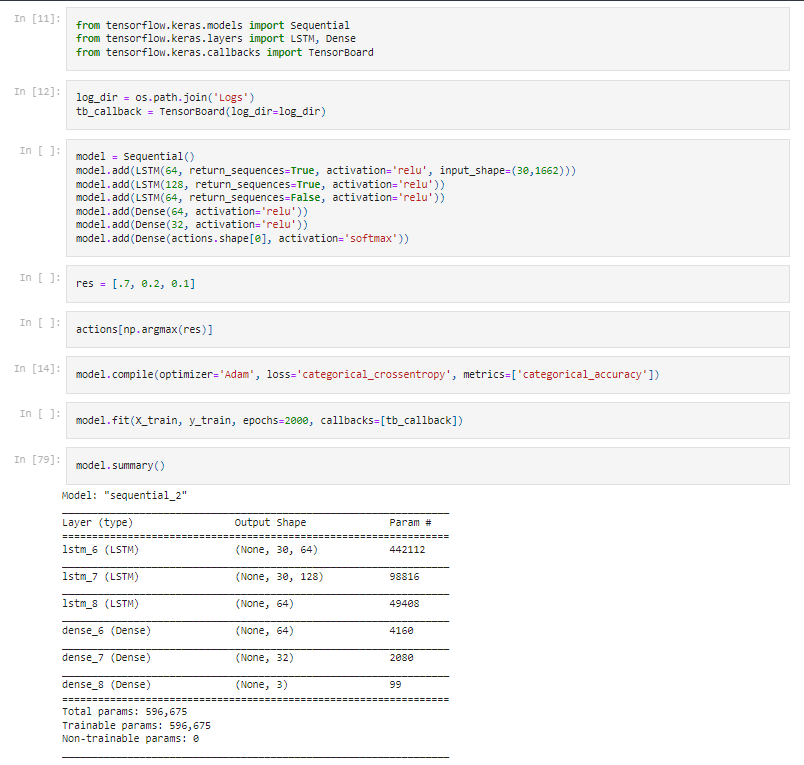
**5.1 Detect Page:**

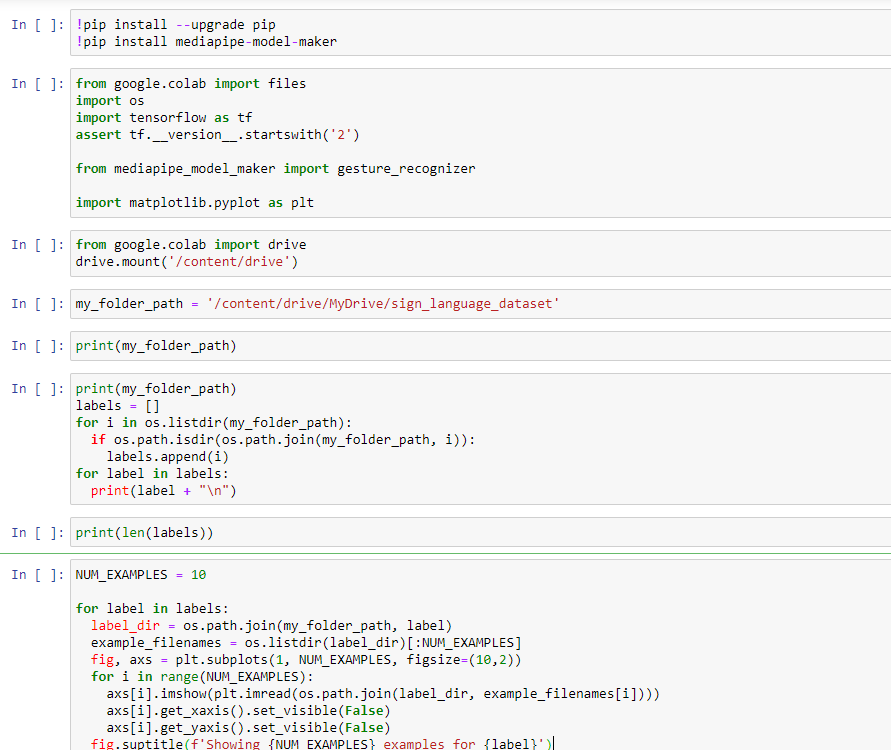
****

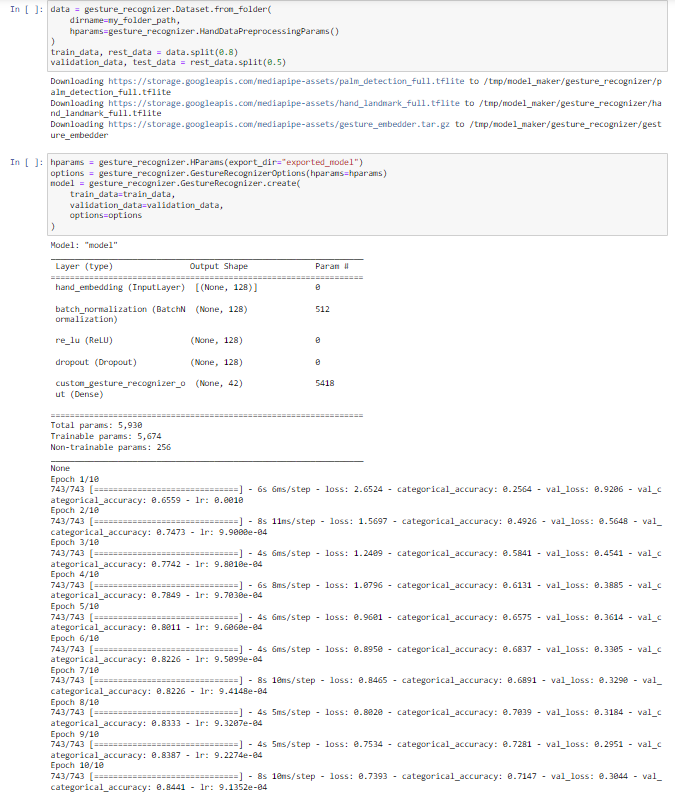
**5.2 Dashboard Code:**

****

**5.3 Model Training**

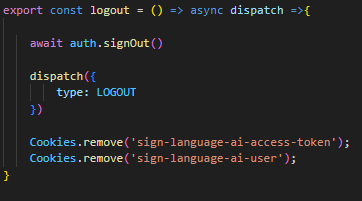


****

****

**5.4 Login/Logout**

****

****

**Chapter 6**

**Testing**

Testing a sign language translator app, integrating React for the user interface, LSTM and Mediapipe for gesture recognition, and Firebase for login authentication, demands a comprehensive approach to ensure its functionality, accuracy, and reliability. Here's a breakdown of various test cases across different testing types:

**6.1 Unit Testing**

**6.1.1 User Authentication**

* Validate user registration with valid data.
* Verify user registration with invalid data (e.g., missing fields, invalid email format).
* Test user login with correct credentials.
* Validate user login with incorrect credentials.
* Ensure password reset functionality works as expected.

**6.1.2 Gesture Recognition**

* Validate trained model accuracy in recognizing various sign gestures.
* Test Mediapipe integration for precise hand landmark detection.

**6.2 Integration Testing**

**6.2.1 User Profile Management**

* Test updating user profile information.
* Verify viewing user profile details.
* Validate profile picture upload functionality.

**6.2.2 Authentication and User Data**

* Test integration between React UI and Firebase authentication for secure login.
* Validate the handling of user data between Firebase and the application.

**6.3 Functional Testing**

**6.3.1 Translation Accuracy**

* Validate the accuracy of sign language translation using deep learning and Mediapipe.
* Test various sign gestures to ensure accurate translation results.

**6.3.2 User Interface**

* Test navigation and usability across different pages and devices.
* Validate form validations and error handling.

**6.4 Usability Testing**

**6.4.1 User Experience**

* Test the overall user experience, ensuring ease of use and clarity in functionality.
* Validate the effectiveness of error messages and prompts.

**6.5 Regression Testing**

**6.5.1 Feature Updates**

* Test existing functionality after adding new features to ensure no regression.
* Validate critical paths to ensure new features do not introduce bugs.

**6.5.2 Bug Fixes**

* Test areas previously affected by bugs to ensure they are resolved.
* Validate related functionalities to ensure bug fixes do not introduce new issues.

By executing these test cases, we ensure that the sign language translator app delivers accurate translations, maintains security and privacy standards, and provides a seamless user experience across devices and gestures.

**Chapter 7**

**Result and Discussion**

**7.1 Smooth interface creation using React.js:**

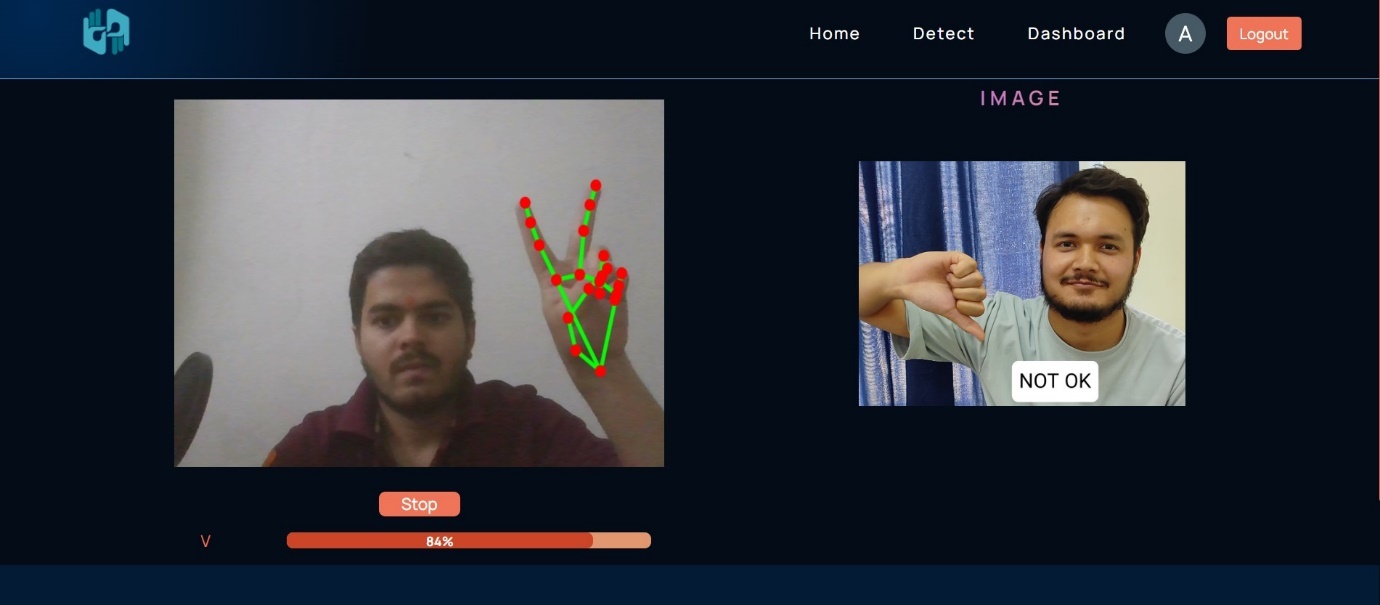


Figure 7.1 Website Interface of the project

The use of React.js in building the interface for the Sign Language Recognition (SLR) application resulted in a highly responsive and smooth user experience. React.js's component-based architecture allowed for modular development, making the interface easy to manage and extend. The dynamic nature of React.js ensured real-time updates and interactions, crucial for the responsiveness required in a real-time SLR system. The virtual DOM mechanism optimized rendering performance, ensuring that the interface remains smooth even with continuous video feed processing. This smooth interaction between the user and the system is vital for maintaining user engagement and providing immediate feedback, prediction on gesture recognition, enhancing the overall usability of the application.

**7.2 Dashboard:**



Figure 7.2 User Dashboard

The dashboard component of the SLR application was designed to provide users with an intuitive overview of their interactions and progress. Utilizing React.js, the dashboard displays a variety of metrics and insights, such as the number of recognized gestures, accuracy rates, and usage statistics. This comprehensive view helps users track their learning progress and identify areas needing improvement. The interactive nature of the dashboard, enabled by React’s reactivity, allows for real-time updates and seamless navigation through different sections. This component plays a crucial role in enhancing user experience by providing actionable insights and promoting continuous learning and engagement with the application.

**7.3 User Authentication:**

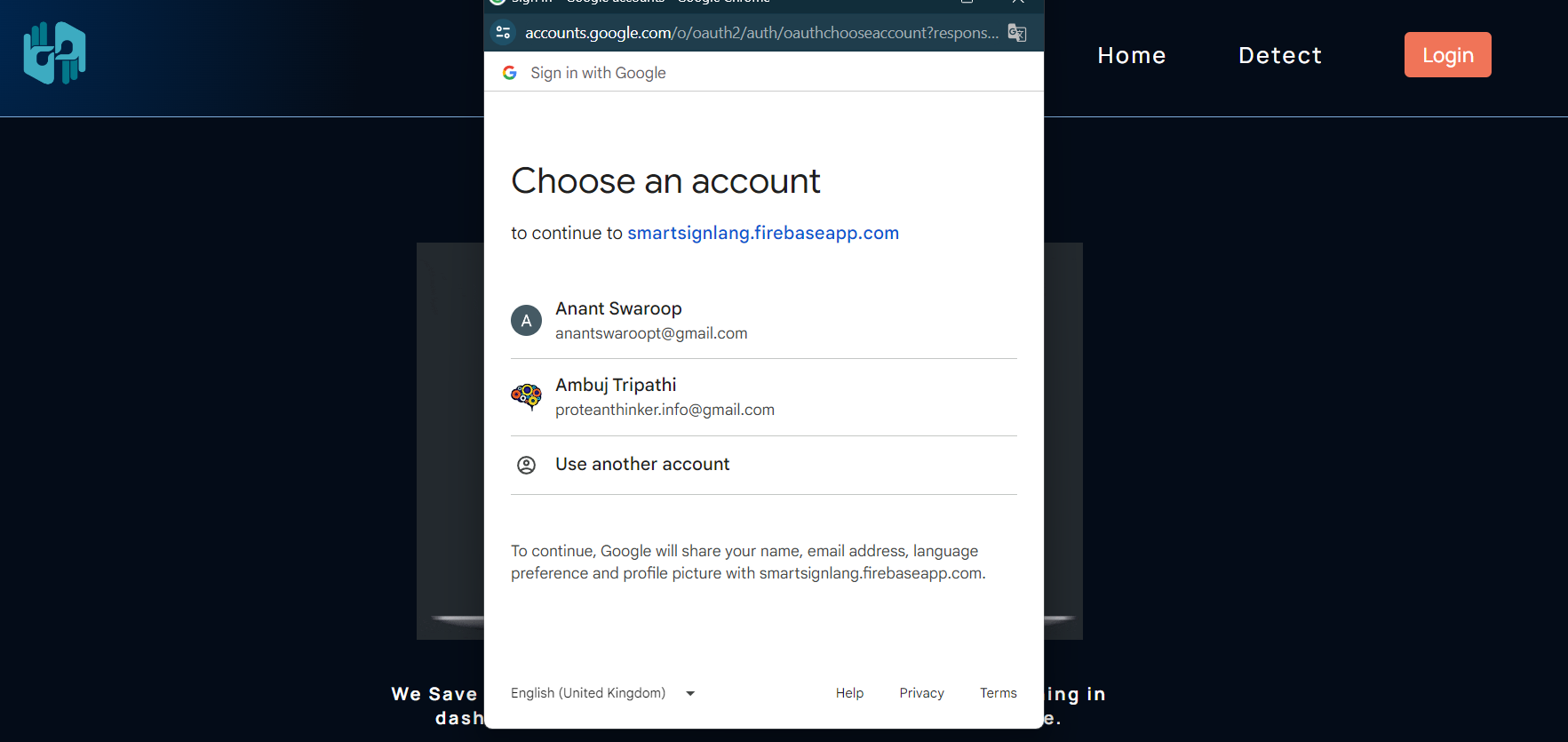
****

Figure 7.3 User Login Page

Our sign language translation application, seamlessly built with React.js for an intuitive user interface and Firebase for robust login authentication, fosters effortless communication and interaction.

**7.4 Tutorial:**

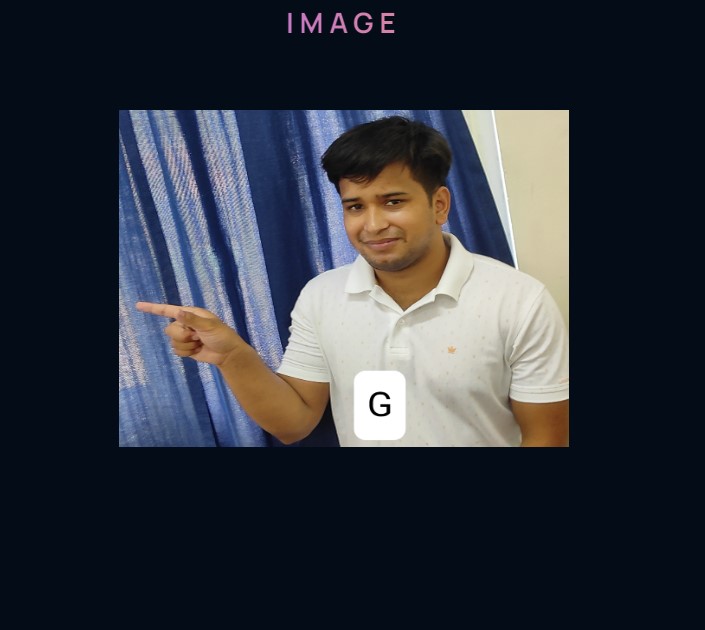


Figure 7.4 Tutorial for learning signs and gestures

The tutorial section is a critical feature designed to aid new users in understanding and effectively using the SLR system. Built with React.js, the tutorial provides step-by-step guidance on how to interact with the application, perform gestures, and understand the feedback provided. Interactive elements and visual aids, such as videos and animations, were integrated to make the learning process engaging and easy to follow. This feature ensures that users can quickly become proficient in using the application, reducing the learning curve and increasing the likelihood of sustained use. The tutorial is continually updated based on user feedback and analytics to ensure it meets the evolving needs of the users.

**7.5 User Data:**

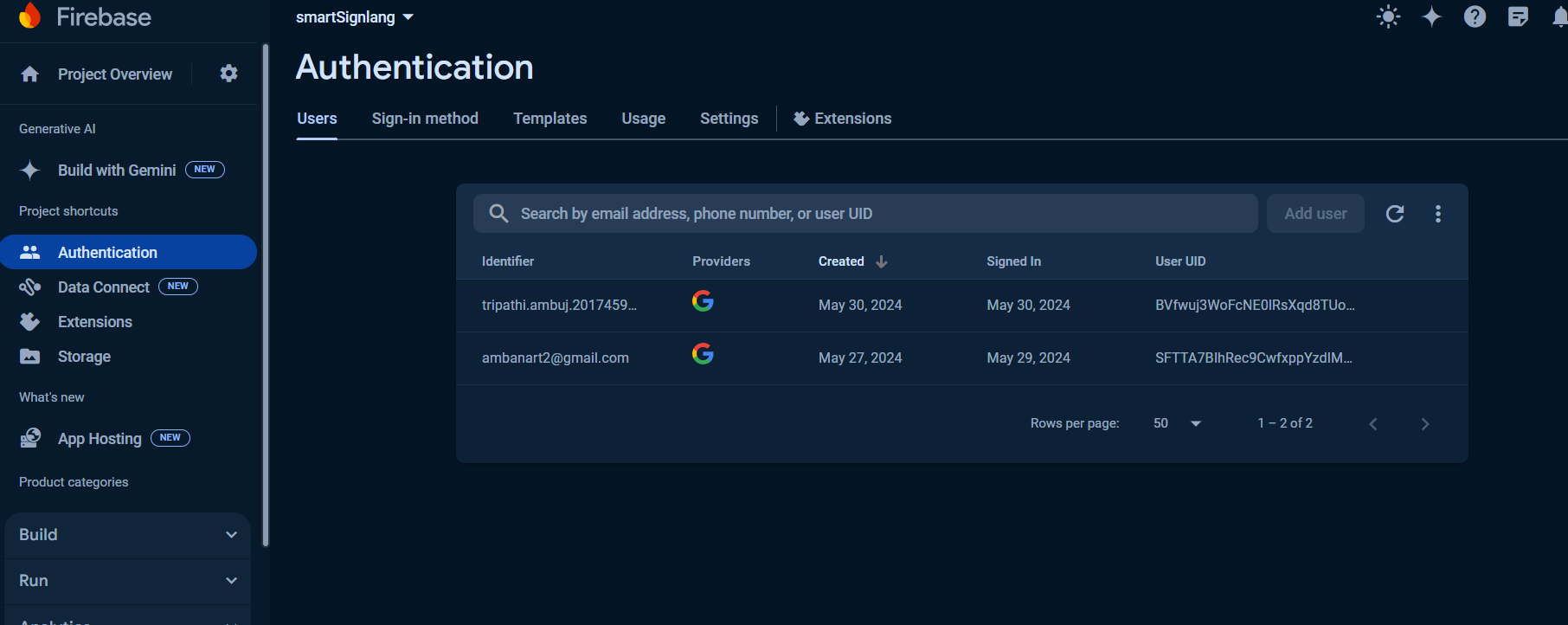


Figure 7.5 User Data stored at Firebase

Firebase was employed to manage user data, providing a robust and scalable backend solution. User data, including login credentials, interaction history, and gesture recognition performance, is securely stored and managed in Firebase. This integration allows for seamless synchronization between the frontend and backend, ensuring that user data is always up-to-date and accessible across different devices. The use of Firebase also enhances security and reliability, as it handles authentication and database management efficiently.

**7.6 Understanding of Mediapipe based Model**

Smart fare prediction is beneficial in carpooling apps for several reasons, enhancing user experience, and improving the overall efficiency of the service.

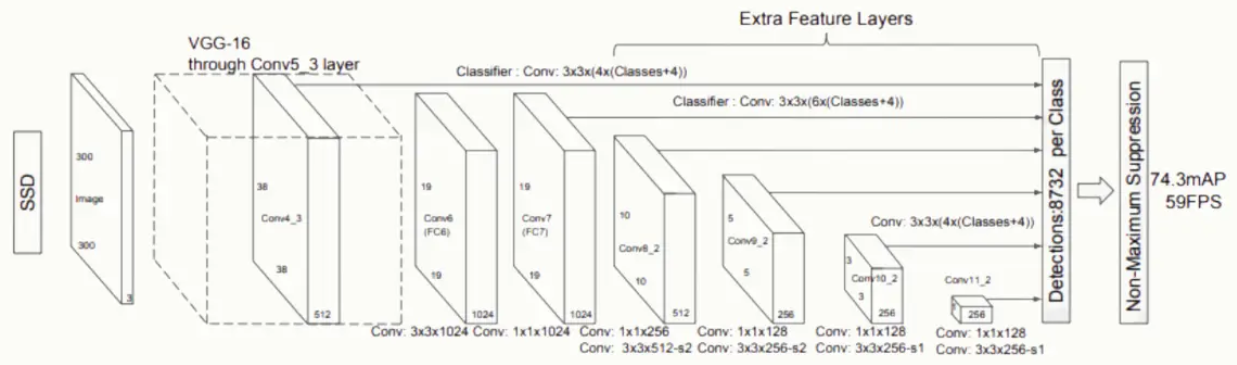
The gesture recognition model finally used in this project utilizes several key stages, each designed for specific tasks within the overall system. These stages are powered by state-of-the-art Convolutional Neural Networks (CNNs), and the process includes palm detection, hand landmark detection, and gesture embedding. 

Figure 7.6 SSD Architecture

For palm detection, the model employs architectures such as Single Shot MultiBox Detector (SSD) or YOLO (You Only Look Once). These models are optimized for real-time performance and can detect objects (in this case, palms) in a single forward pass through the network. CNNs are used to extract features from the input images, and localization and classification loss functions are applied during training to improve the accuracy of bounding box predictions and object classification.

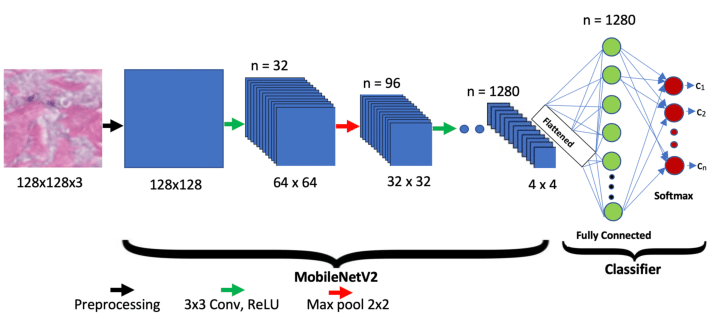


Figure 7.7 MobileNet Architecture

In the hand landmark detection stage, specialized regression networks like MobileNet are used for efficient and accurate keypoint detection. These networks predict the coordinates of key points (landmarks) on the hand. CNNs are again utilized to identify and predict the location of these key points, with the Mean Squared Error (MSE) loss function employed to minimize the difference between predicted and actual landmark positions.

Finally, the gesture embedding and recognition stage involves a fully connected (dense) network that processes the hand embeddings generated by the landmark model. This network classifies gestures based on these embeddings. Techniques such as backpropagation are used for training the neural networks, and optimization algorithms like Adam are implemented to enhance training efficiency.

Extensive data augmentation is used throughout the process to ensure robustness to variations in input data. The combination of these architectures and techniques ensures that the model is capable of accurately and efficiently recognizing sign language gestures in real-time. By following these principles and utilizing advanced deep learning techniques, the project aims to provide a robust and efficient solution for real-time gesture recognition.

**Chapter 8**

**Conclusion and Future Work**

The Sign Language Recognition project explores LSTM and CNN efficiency and finally leverages advanced deep learning techniques within the MediaPipe framework to create an efficient and accurate real-time gesture recognition system. By integrating multiple pre-trained models for palm detection, hand landmark detection, and gesture embedding, the project ensures robust and precise gesture recognition. The system is designed to be user-friendly, providing real-time feedback and comprehensive statistics, enhancing the accessibility of communication for the hearing and speech impaired.

The backend, developed using Node.js, ensures reliable data handling and model processing, while the frontend, built with React Native, offers an intuitive interface for users. The training process is streamlined through Google Colab, making use of TensorFlow and MediaPipe’s pre-trained models. This setup not only accelerates development but also ensures high accuracy and performance, crucial for real-time applications.

**Future Work**

**1. Improved Model Accuracy and Speed:**

* **Optimization**: Further optimize the model to enhance both accuracy and speed, making it more efficient for real-time applications on mobile and edge devices.
* **Advanced Algorithms**: Explore more advanced deep learning algorithms and architectures to improve gesture recognition capabilities.

**2. Dataset Expansion**:

* **Diverse Gestures**: Expand the dataset to include a wider variety of sign language gestures, covering more comprehensive vocabulary.
* **Multilingual Support**: Incorporate gestures from different sign languages to support a global user base.

**3. User Authentication and Profile Management:**

* **Enhanced Security:** Implement multi-factor authentication to enhance security for user accounts.
* **Personalization:** Allow users to personalize their profiles, adjusting settings to improve their user experience.

**4. Integration with Wearable Devices:**

* **Smart Glasses and Watches:** Extend the application’s capabilities to integrate with wearable devices, allowing for more versatile usage scenarios.

**5. Community and Feedback:**

* **User Feedback:** Establish a feedback loop with users to continuously improve the system based on real-world usage.
* **Community Building:** Create a community platform for users to share experiences, tips, and improvements.

The future work plan of our project are as follows:

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Work Description** | **Duration in Days** |
| **1.** | Dataset Expansion | **20-25** |
| **2.** | Community and Feedback | **5-6** |

**Details of Research Publication**

|  |  |  |
| --- | --- | --- |
|  | **Project Team ID** | MP23CST024 |
| 2. | **Name of Supervisor** | Mr. Vivek Tomar |
| 3. | **Title of the paper** | A NOVEL APPROACH FOR SIGN LANGUAGE RECOGNITION USING MEDIAPIPE WITH ML MODELS |
| 4. | **Authors Name** | 1. Ambuj Tripathi 2. Saatwik Semwal 3. Sandeep Singh 4. Suryansh Makhloga 5. Vivek Tomar |
|  | **Percentage of plagiarism**  **(Check in Turnitin,**  **10 words)** | 9% |
| 6. | **Status of the research paper** | * Published/ Presented * Registered for conference * Accepted * Communicated * Not-Communicated |
| 7. | **Scopus Indexed**  **(mark the appropriate one)** | * Journal * Conference |
| Signature of students                                                         Signature of Supervisor | | |

**References**

[1] Alnaim N, Abbod M. Mini gesture detection using neural networks algorithms. In: Eleventh International Conference on Machine Vision (ICMV 2018). vol. 11041. SPIE; 2019. p. 547-55.

[2] Al-Obodi, A. H., Al-Hanine, A. M., Al-Harbi, K. N., Al-Dawas, M. S., & Al-Shargabi, A. A. (2017). A Saudi Sign Language Recognition System Based on Convolutional Neural Networks. \*Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia.\*

[3] Bayazit, et al. (2009). Real-Time Motion-Based Gesture Recognition Using the GPU. In: \*MVA, Citeseer,\* p. 9-12.

[4] Chava Sri Varshini, et al. (2020). Sign Language Recognition. \*International Journal of Engineering Research Technology (IJERT), 09\*(05).

[5] Chavan, S., et al. (2021). Convolutional Neural Network Hand Gesture Recognition for American Sign Language. In: \*2021 IEEE International Conference on Electro Information Technology (EIT), IEEE,\* p. 188-92.

[6] Chen, Zh., et al. (2014). Real-Time Hand Gesture Recognition Using Finger Segmentation. \*The Scientific World Journal, 2014.\*

[7] Huang, H., et al. (2019). Hand Gesture Recognition with Skin Detection and Deep Learning Method. In: \*Journal of Physics: Conference Series, 1213,\* IOP Publishing, p. 022001.

[8] Islam, M. Z., et al. (2019). Static Hand Gesture Recognition Using Convolutional Neural Network with Data Augmentation. In: \*2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), IEEE,\* p. 324-9.

[9] Juneja, S., et al. (2021). Computer Vision-Enabled Character Recognition of Hand Gestures for Patients with Hearing and Speaking Disability. \*Mobile Information Systems, 2021.\*

[10] Nivedita, S., Ramyapriya, Y., et al. (2022). Sign Language Recognition System Using Machine Learning.

[11] Perdana, I. P. I., Putra, et al. (2021). Classification of Sign Language Numbers Using the CNN Method. \*JITTER: Jurnal Ilmiah Teknologi dan Komputer, 2\*(3), 485-93.

[12] Pigou, L., et al. (2014). Sign Language Recognition Using Convolutional Neural Networks. \*European Conference on Computer Vision, 6-12 September 2014.\*

[13] Rajan, G. S., Nagarajan, et al. (2019). Interpretation and Translation of American Sign Language for Hearing Impaired Individuals Using Image Processing. \*International Journal of Recent Technology and Engineering (IJRTE), 8\*(4), 415–420.

[14] Suk, H., et al. (2010). Hand Gesture Recognition Based on Dynamic Bayesian Network Framework. \*Pattern Recognition, 43\*(9), 3059-3072.

[15] Vazquez Lopez, I. (2017). Hand Gesture Recognition for Sign Language Transcription.

[16] Zhang, Fan, et al. (2020). "Mediapipe Hands: On-Device Real-Time Hand Tracking." \*arXiv preprint arXiv:2006.10214.\*