**ISL Speech Conversion System For Medical Emergencies :**

**Project Code Specification File**

***1. Project Overview***

***Title:* ISL Speech Conversion System For Medical Emergencies**

***Objective:***  
To develop an intelligent system that detects and recognizes Indian Sign Language (ISL) gestures in real time using deep learning and converts them into Hindi speech for improved accessibility and emergency communication.

***Description:***  
This project integrates MediaPipe Holistic for gesture landmark extraction and LSTM (Long Short-Term Memory) neural networks for sequential pattern recognition. The trained model recognizes gestures corresponding to emergency or health-related actions such as *help, blood, pain, bone\_break,* and *fever*. The recognized gesture is then converted into Hindi speech using Google Text-to-Speech (gTTS) and played aloud in real time, creating an assistive tool for communication in critical scenarios.

***2. System Configuration***

| Category | Specification |
| --- | --- |
| Processor | Intel Core i5 or higher |
| RAM | Minimum 8 GB |
| Hard Disk | Minimum 20 GB free space |
| GPU (Optional) | NVIDIA GPU for faster model training |
| Operating System | Windows 10 / 11, macOS, or Linux |
| Programming Language | Python 3.10 or higher |
| Framework | TensorFlow / Keras |
| Computer Vision Library | MediaPipe, OpenCV |
| IDE / Notebook | Jupyter Notebook / Google Colab |
| Deployment Tool | Streamlit Interface (for live camera detection) |
| Audio Output | gTTS + playsound for Hindi speech synthesis |

***3. Libraries and Dependencies***

| Library / Package | Purpose / Description |
| --- | --- |
| tensorflow | Deep learning framework for LSTM model creation and training |
| mediapipe | Real-time pose, hand, and face landmark detection |
| opencv-python | Camera feed handling, frame processing, and visualization |
| numpy | Numerical operations and data manipulation |
| matplotlib | Visualization of training accuracy and sample outputs |
| scikit-learn | Dataset splitting, accuracy calculation, and metrics |
| gtts | Google Text-to-Speech conversion for Hindi audio |
| playsound | Playback of generated speech files |
| os, uuid, threading, time | File handling, parallel speech playback, and timing control |
| streamlit | Interactive interface for starting real-time gesture detection |

***4. Model Architecture Details***

| Layer Type | Details / Parameters |
| --- | --- |
| Input Layer | Sequence of 30 frames, each containing 1,662 keypoints (pose, hand, and face landmarks) |
| LSTM Layer 1 | 128 units, return\_sequences=True, activation='tanh' |
| Dropout | 0.3 |
| LSTM Layer 2 | 128 units, return\_sequences=True, activation='tanh' |
| Dropout | 0.3 |
| LSTM Layer 3 | 64 units, return\_sequences=False, activation='tanh' |
| Dropout | 0.3 |
| Dense Layer 1 | 128 neurons, ReLU activation |
| Dropout | 0.3 |
| Dense Layer 2 | 64 neurons, ReLU activation |
| Output Layer | 5 neurons (softmax activation for gesture classification) |
| Optimizer | Adam |
| Loss Function | Categorical Crossentropy |
| Evaluation Metric | Categorical Accuracy |

***5. Dataset Information***

| Parameter | Description |
| --- | --- |
| Dataset Source | Custom dataset recorded using webcam and MediaPipe Holistic |
| Classes / Actions | 5 — help, blood, pain, bone\_break, fever |
| Input Format | NumPy arrays of extracted keypoints per frame |
| Frames per Sequence | 30 frames per gesture sequence |
| Number of Sequences per Class | 50 sequences (base) + augmented data |
| Augmentation | Noise addition, random scaling, shifting for dataset expansion |
| Train-Test Split | 70% training, 30% testing |

***6. Execution Flow***

1. Import Required Libraries
2. Initialize MediaPipe Holistic model
3. Capture video frames using OpenCV
4. Extract and save landmarks as NumPy arrays
5. Organize dataset into classes and sequences
6. Augment data for better generalization
7. Build and train LSTM model on extracted keypoints
8. Evaluate model performance (accuracy, confusion matrix)
9. Save trained model (action.h5)
10. Load model for real-time testing
11. Perform live gesture detection through webcam
12. Convert recognized gesture to Hindi audio using gTTS and play sound

***7. Output and Evaluation***

| Metric | Description / Value |
| --- | --- |
| Training Accuracy | ~95% |
| Testing Accuracy | ~90% |
| Confusion Matrix | Evaluates class-wise gesture recognition accuracy |
| Prediction Output | Predicted gesture name and confidence score |
| Visualization | Live camera feed with landmarks and prediction overlay |
| Speech Output | Real-time Hindi audio corresponding to detected gesture |
| Interface | Streamlit-based "Start Camera" button and display feed |

***8. Future Scope***

* Expand the dataset to include more ISL words and complex sentences.
* Integrate transformer-based architectures or CNN-LSTM hybrids for better temporal modeling.
* Build a mobile app using *TensorFlow Lite* for on-device gesture translation.
* Incorporate multilingual TTS support for broader regional language communication.
* Develop an emergency communication assistant for healthcare or accessibility use cases.

***9. Conclusion***

The project demonstrates a practical AI-driven ISL-to-speech communication system that combines computer vision and deep learning.  
By leveraging MediaPipe for landmark extraction and LSTM for temporal gesture modeling, the system achieves high accuracy in gesture recognition.  
The integration of gTTS for Hindi audio output makes it especially valuable for medical emergencies and accessibility tools.  
Overall, this real-time system bridges communication gaps for individuals relying on sign language.