NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE SYSTEM

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

1. Mogalathurthi N S V Devi Parvathi – 21B01A05A7
2. Mathi Sai Divya Sri – 21B01A05A1
3. Neelapala Manasa – 21B01A05B8
4. Kodali Sravani - 21B01A0575
5. Koppada Naga Lakshmi – 22B05A0507

***Under the esteemed guidance of* Mr.**
P. Sunil
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, accredited by NBA, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202
2024 – 2025

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, accredited by NBA, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled,’ ***NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE SYSTEM’*** *is being submitted by* ***Mogalathurthi N S V Devi Parvathi, Mathi Sai Divya Sri, Neelapala Manasa, Kodali Sravani, Koppada Naga Lakshmi*** *bearing the* ***Regd. No. 21B01A05A7, 21B01A05A1, 21B01A05B8, 21B01A0575, 22B05A0507*** *in partial fulfilment of the requirements for the award of the degree of “****Bachelor of Technology*** *in* ***Computer Science & Engineering****” is a record of Bonafide work carried out by her under my guidance and supervision during the academic year* ***2024–2025*** *and it has been found worthy of acceptance according to the requirements of the university.*

Internal Guide **Head of the Department**

External Examiner

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout the course of this seminar. We take this opportunity to express our gratitude to all those who have helped us in this seminar.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju**, Chairman of SVES, for his constant support on each and every progressive work of mine.

We wish to express our sincere thanks to **Dr. G. Srinivasa Rao**, Principal of SVECW for being a source of inspiration and constant encouragement.

We wish to express our sincere thanks to **Prof P. Venkata Rama Raju**, Vice-Principal of SVECW for being a source of inspirational and constant encouragement.

We wish to express our sincere thanks to **Dr. P. Srinivasa Raju,** Director Students Affairs & Admin for SVES Group of Institutions for being a source of inspirational and constant encouragement.

We wish to place our deep sense of gratitude to **Dr. P. Kiran Sree**, Head of the Department of Computer Science & Engineering for his valuable pieces of advice in completing this seminar successfully.

We are deeply thankful to our project coordinator **Dr. P. R. Sudha Rani**, Professor for her indispensable guidance and unwavering support throughout our project's completion. Her expertise and dedication have been invaluable to our success.

We are deeply indebted and sincere thanks to our PRC members **Dr. A. Seenu, Dr. T. Gayathri**, **Mr. Y. Ramu** for their valuable advice in completing this project successfully.

Our deep sense of gratitude and sincere thanks to **Mr. P. Sunil**, Assistant Professor for his unflinching devotion and valuable suggestions throughout my project work.

Project Associates:

1. M N S V Devi Parvathi - 21B01A05A7
2. Mathi Sai Divya Sri – 21B01A05A1
3. Neelapala Manasa- 21B01A05B8
4. Kodali Sravani - 21B01A0575
5. Koppada Naga Lakshmi – 22B05A0507

ABSTRACT

Effective communication remains a significant challenge for individuals who rely on sign language, particularly when interacting with non-signers. This project presents an NLP and Blender-Based Sign Language Animation System that converts spoken or written English text into Indian Sign Language (ISL) animations in real time. Leveraging Natural Language Processing (NLP), the system extracts essential keywords from the input, maps them to corresponding 3D sign animations, and applies fingerspelling when direct signs are unavailable to ensure clarity and completeness. The animations, developed using Blender, are dynamically merged into a single video to represent the full input meaningfully. Additional features such as adjustable playback speed and video downloading enhance accessibility and user experience. The system integrates speech-to-text conversion, tense detection, synonym handling, and video rendering to support effective communication for the deaf and hard-of-hearing community. By combining intelligent language processing with 3D animation and user-centric features, this system offers a practical, inclusive, and educational tool for learning and using Indian Sign Language.

TABLE OF CONTENTS

Table of Contents	Page No
1. INTRODUCTION	01-03
1.1 Background and Motivation	02-02
1.2 Need for Sign Language Animation	02-03
1.3 Objectives of the Project	03-03
2. SYSTEM ANALYSIS	04-07
2.1 Existing System and Limitations	05-05
2.2 Proposed System	05-06
2.3 Feasibility Study	06-07
3. SYSTEM REQUIREMENTS	08-13
3.1 Hardware Requirements	09-10
3.2 Software Requirements	10-10
3.3 Functional and Non-Functional Requirements	11-13
4. SYSTEM DESIGN	14- 21
4.1 System Architecture	15-16
4.2 Data Flow Diagram	16-17
4.3 UML Diagrams	17-21
5. SYSTEM IMPLEMENTATION	22-36
5.1 Text Processing Using NLP	23-26
5.2 Synonym Mapping and Word Recognition	26-27
5.3 Animation Generation in Blender	27-33
5.4 Video Generation and Control Features	33-35
5.5 History and Favourites Features	35-36

6. TESTING AND EVALUATION	37-41
6.1 Test Cases	38-39
6.2 System Performance Analysis	39-41
7. CONCLUSION	42-43
8. BIBLIOGRAPHY	44-45
9. APPENDIX	46-57

List of figures

Figure	Page No
4.1.1 Work flow	17
4.1.2 Class diagram	19
4.1.3 Sequence diagram	20
5.0.1 Project execution process	23
5.0.2 Web application interface	24
5.0.3 Sign up page	24
5.0.4 login page	25
5.0.5 Home page	25
5.1.1 Input text	26
5.1.2 Input text is processed in this format	27
5.2.1 Animation being played for the corresponding text	27
5.2.2 Finger spelling	28
5.3.1 dataset created using blender	29
5.3.2 Contact page	29
5.3.3 Importing model into blender	30

5.3.4 Setting file location	31
5.3.5 3D model	31
5.3.6 Animation timeline with keyframes	32
5.3.7 Animation Rendering	33
5.4.1 Video Download Option	34
5.4.2 Video Speed Control Dropdown	34
5.5.1 History Page showing past translated inputs with date, keywords, and video links	35
5.5.2 Favourites Page displaying saved animations for quick access and reuse	36
6.1.1 Test cases to check application working	38
6.1.2 Test case - 01	38
6.1.3 Test case - 02	39
6.1.4 Test case - 03	39
6.1.5 Test case - 04	40
6.1.6 Test case - 05	40
6.1.7 Test case - 06	41
6.1.8 Test case - 07	41
9.0.1 Natural Language Processing	48
9.0.2 Blender 4.3	50

1. INTRODUCTION

1.1 Background and Motivation

Language is the foundation of human communication, enabling people to exchange ideas, express emotions, and collaborate effectively. For individuals with hearing and speech impairments, sign language serves as an essential mode of communication. However, widespread adoption of sign language is hindered by a lack of awareness, training, and accessibility tools. This gap creates communication barriers, especially in education, employment, and social settings.

Existing methods of sign language interpretation rely heavily on human interpreters, which may not always be available or practical. Some digital solutions exist, but they often lack real-time adaptability and scalability. With recent advancements in Natural Language Processing (NLP), speech recognition, and 3D animation, it is now feasible to develop an automated system that converts text and speech into Indian Sign Language (ISL) animations.

This project, NLP and Blender-Based Sign Language Animation System, is designed to address these challenges by providing a real-time, interactive, and scalable solution for ISL translation. By leveraging NLP for text processing and Blender for 3D animation generation, this system enhances accessibility and promotes inclusivity by enabling individuals to communicate effectively using ISL.

1.2 Need for Sign Language Animation

The need for an automated sign language animation system arises from the following challenges and limitations:

- **Limited ISL Proficiency:** A significant portion of the population, including educators, healthcare professionals, and service providers, lacks familiarity with ISL, making effective communication difficult.
- **Dependence on Human Interpreters:** Manual sign language interpretation is not always feasible due to availability constraints and associated costs.
- **Lack of Real-Time Solutions:** Most existing tools focus on static representations of sign language rather than dynamic, real-time animations that adapt to user input.
- **Educational and Accessibility Gaps:** The absence of interactive ISL learning tools hinders effective education for both the hearing-impaired and individuals learning sign language.

- **Technological Advancements:** Recent developments in Natural Language Processing and 3D animation enable the creation of highly accurate, automated ISL translation systems.
- **Promoting Inclusivity:** Providing an easy-to-use, scalable, and web-based platform ensures that ISL users and non-signers can communicate seamlessly in various domains such as education, workplaces, healthcare, and public services.

1.3 Objectives of the Project

The primary objectives of this project are:

- To Develop a system that converts both spoken and written English input into Indian Sign Language (ISL) animations in real time.
- To Apply Natural Language Processing (NLP) techniques for analyzing, extracting, and interpreting meaningful keywords from user input.
- To Integrate Blender to generate accurate and realistic 3D sign language animations with proper gesture representation.
- To design a scalable, interactive, and user-friendly web-based platform that facilitates ISL communication and learning.
- To Implement a fallback system that uses fingerspelling when predefined ISL signs are unavailable.
- To incorporate playback speed control and video download functionality for better accessibility and user convenience.
- To Implement user-specific features such as storing animation history and allowing users to mark animations as favourites
- To create a solution that can be extended to assistive technologies, educational environments, and digital accessibility tools.

By combining language processing with 3D animation and interactive features, this system promotes effective and inclusive communication, enabling individuals with hearing impairments to engage more seamlessly with the wider community. Its implementation represents a significant step toward bridging accessibility gaps and fostering a digitally inclusive society.

2. SYSTEM ANALYSIS

2.1 Existing System and Limitations

The current methods for sign language translation are largely dependent on human interpreters or pre-recorded sign videos. These approaches, while effective in controlled settings, present several challenges in terms of accessibility, scalability, and real-time adaptability.

Limitations of the Existing System:

- **Dependence on Human Interpreters:** The availability of qualified sign language interpreters is limited, which makes real-time communication difficult in many everyday scenarios.

- **Lack of Automation:** Most existing solutions are based on static sign images or predefined videos, which cannot dynamically adapt to new or complex phrases.

- **Limited Digital Tools:** Some platforms offer basic text-to-sign features, but they typically lack real-time animation, NLP-based processing, or support for sentence-level input.

- **High Cost and Poor Scalability:** Manual interpretation and static video libraries can be expensive and are not feasible for continuous or large-scale use.

- **Inaccessibility in Remote Areas:** Many regions do not have reliable access to interpreters or digital solutions, leaving the hearing-impaired community underserved.

2.2 Proposed System

To address the limitations of existing systems, the NLP and Blender-Based Sign Language Animation System introduces an automated, real-time solution for translating both spoken and written English into Indian Sign Language (ISL) animations. This system is designed to be scalable, accessible, and user-friendly, aiming to bridge the communication gap for the deaf and hard-of-hearing community.

Key Features of the Proposed System:

- **Real-Time Text and Speech Processing**: The system accepts both typed and spoken input and converts it instantly into ISL animations.

- **Natural Language Processing (NLP)**: NLP techniques such as tokenization, POS tagging, lemmatization, tense detection, and synonym mapping are employed to extract meaningful keywords.
- **3D Animation Using Blender:** Blender is used to create realistic, gesture-accurate sign animations for each word or letter.
- **Synonym Mapping for Flexibility**: A custom synonym dictionary, along with WordNet, is integrated to find alternate ISL-compatible words when direct matches are unavailable.
- **Finger Spelling for Non-Mapped Words:** For words without ISL equivalents, the system generates animations for each letter to ensure complete translation.
- **Web-Based Platform:** The application is built on Django and accessible via a browser, making it device-independent and easy to use.
- **Video Merging and Rendering:** Individual animations are automatically merged into a single cohesive video using FFmpeg.
- **Playback Speed Control:** Users can adjust the speed of the animation playback for better learning and accessibility.
- **Download Option:** Users can download the generated animation videos for offline use or reference.
- **User Authentication with Personalization:** The system allows users to register, log in, and manage their own animation history and favorites.

This system not only enhances digital accessibility but also provides a learning platform for those wishing to understand or teach ISL through interactive, visual means.

2.3 Feasibility Study

To evaluate the practicality and effectiveness of the proposed system, a feasibility study has been conducted considering technical, economic, and operational factors.

Technical Feasibility

- **Implementation of NLP:** The system employs Natural Language Processing techniques for analyzing and interpreting both text and speech inputs.
- **3D Animation Rendering:** Blender is used to generate smooth and realistic animations of ISL gestures, ensuring accurate sign representation.

- **Web-Based Deployment:** The application is deployed on a web platform, requiring only a standard browser, making it widely accessible without the need for additional software. **Economic Feasibility**

- **Cost-Effective Solution:** The system leverages open-source technologies such as Blender, Django, and Python-based NLP libraries, significantly reducing development and deployment costs.

- **Scalability:** Unlike manual interpretation services, the system supports multiple users simultaneously without incurring additional costs or resource overhead.

Operational Feasibility

- **Ease of Use:** The user interface is designed to be intuitive and user-friendly, requiring minimal technical knowledge for interaction.

- **Accessibility and Utility:** The system benefits a wide range of users, including individuals with hearing impairments, educators, students, and ISL learners.

- **Real-World Applications:** This solution has practical use in educational institutions, workplaces, healthcare, government services, and other public or private environments that require inclusive communication tools.

The NLP and Blender-Based Sign Language Animation System addresses the key limitations of traditional approaches by offering an automated, real-time, and scalable method for ISL translation. Through the integration of NLP for language processing and Blender for animation, it promotes accessibility, inclusivity, and meaningful communication across diverse user groups.

3. SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

The hardware requirements define the minimum and recommended system specifications needed for smooth execution of the project. These specifications ensure that the NLP processing, speech recognition, and 3D animation rendering function without performance bottlenecks.

Minimum Requirements:

The minimum specifications are sufficient for basic functionality and testing of the system.

- Processor: Intel Core i3 or AMD equivalent → Capable of handling basic NLP and web-based applications.
- RAM: 4GB → Sufficient for running the web application and small-scale NLP processing.
- Storage: 10GB free space → Required to store necessary animation files and datasets.
- GPU: Integrated Graphics → Supports basic rendering of sign language animations.
- Operating System: Windows 10 / Ubuntu 18.04 or later → Ensures compatibility with the required software.

Recommended Requirements:

For optimal performance in real-time animation processing, higher specifications are suggested.

- Processor: Intel Core i5 or higher / AMD Ryzen 5 or higher → Faster text processing and animation rendering.
- RAM: 8GB or more → Handles larger inputs efficiently and supports multitasking.
- Storage: 20GB free space → Required for storing a larger animation dataset.
- GPU: Dedicated GPU (NVIDIA GTX 1050 or higher) → Improves animation playback and real-time rendering in Blender.

- Operating System: Windows 10/11, Ubuntu 20.04 or later → Ensures smooth execution and compatibility with dependencies.

3.2 Software Requirements

This section outlines the essential software components and tools required to develop, deploy, and operate the NLP and Blender-Based Sign Language Animation System effectively.

- Operating System: Windows, Linux (Ubuntu), or macOS
  → The project is cross-platform and can be executed across all major OS environments.

- Programming Language: Python (version 3.8 or later)
  → Used for implementing NLP logic, backend development, and animation processing.

- Web Framework: Django
  → Powers the backend server, routing, database handling, user management, and dynamic HTML rendering.

- NLP Library: NLTK (Natural Language Toolkit)
  → Facilitates text preprocessing, tokenization, lemmatization, POS tagging, and synonym handling.

- Speech Recognition: WebkitSpeechRecognition API
  → Enables real-time voice input by converting speech to text through the browser.

- 3D Animation Software: Blender 4.3
  → Used to design, animate, and export ISL gesture videos in 3D.

- Database: SQLite
  → Lightweight database for storing user history, favorite records, and animation file mappings.

- Video Merging Tool: FFmpeg
  → Command-line utility used to merge individual gesture animations into a single video.

- Web Browser: Google Chrome, Mozilla Firefox, or Microsoft Edge
  → Required for accessing the web application and interacting with features such as speech input and video playback.

3.3 Functional Requirements

Functional requirements define the core capabilities of the system that contribute to the realtime translation of spoken and written English into Indian Sign Language (ISL) animations.

1. Text Input Processing

- Accepts user-entered English text as input.
- Processes the text using NLP techniques such as tokenization, lemmatization, and stopword removal.
- Identifies meaningful keywords and maps them to corresponding ISL animations.

2. Speech-to-Text Conversion

- Captures spoken input using the WebkitSpeechRecognition API.
- Converts speech into text in real time via the web interface.
- Passes the converted text through the NLP pipeline for animation generation.

3. Sign Language Animation Generation

- Retrieves corresponding animations from a predefined Blender-generated dataset.
- Dynamically renders and displays the animations in real time.
- Uses FFmpeg to merge individual sign animations into a single video output.

4. Synonym Handling

- If a direct sign animation is not available for a word, the system substitutes it with a synonym using a custom synonym dictionary and WordNet.
- Ensures higher accuracy and flexibility in translation.

5. Finger Spelling for Unrecognized Words

- If no ISL gesture or synonym is found, the system spells out the word letter-by-letter using fingerspelling animations.

6. Video Playback and Control

- Allows users to play, pause, and control the playback speed of the generated video.
- Offers a range of speed options (e.g., 0.5x, 1x, 1.5x, 2x) to suit user preferences.

7. Video Download Feature

- Enables users to download the generated ISL animation video for offline use or future reference.

8. User Authentication and Personalization

- Allows users to register, log in, and manage personal accounts securely.

9. History Tracking

- Automatically saves translated inputs, extracted keywords, and generated videos to the user's history.

10. Favorites Management

- Users can mark any previously generated animation as a favorite.
- Favorites can be viewed, added, or removed through a dedicated interface.

3.4 Non-Functional Requirements

These requirements specify the overall performance, usability, and quality standards the system should meet.

1. Performance

- The system should process user input and generate animation results within 2–5 seconds.
- Animation playback must be smooth and lag-free.

2. Scalability

- Capable of handling multiple users concurrently.
- Easily extendable to include more signs, languages, or features.

3. Usability

- The user interface should be intuitive and responsive.

- Users should be able to navigate, input text or speech, and access video results without training.

4. Security

- Authentication system should secure user accounts and protect history/favorites data.

- Prevent unauthorized access to system resources and animation datasets.

5. Maintainability

- Follows modular coding practices for ease of debugging, testing, and enhancement.
- New animations, synonyms, and language modules should be easily integrable.

6. Portability

- The application should run across platforms including Windows, Linux, and macOS with minimal configuration.

The NLP and Blender-Based Sign Language Animation System combines modern web technologies, NLP, speech recognition, and 3D animation to deliver an intelligent, user-friendly platform for Indian Sign Language translation. With real-time processing, video merging, playback features, and personalization, it effectively supports accessible communication for the deaf and hard-of-hearing community.

4. SYSTEM DESIGN

4.1 Introduction to System Design

System design defines the architecture, workflow, and functional structure of the application. It provides a systematic approach for the integration and interaction of various modules to ensure smooth data flow, modular development, and efficient execution.

The NLP and Blender-Based Sign Language Animation System adopts a modular, layered architecture that combines Natural Language Processing (NLP), speech recognition, and 3D animation rendering. The system translates real-time text or speech into Indian Sign Language (ISL) animations through a seamless and interactive web interface.

This design promotes scalability, performance, and user accessibility, while providing features such as synonym handling, fingerspelling, animation merging, playback speed control, download options, user history tracking, and favorites management.

4.2 System Architecture

The system architecture consists of interconnected layers that work collaboratively to convert user input into animated ISL output. Below is an overview of each major component:

User Interface (UI)

→ A web-based front-end allowing users to input text or speech, view animations, control playback speed, and download videos.

Speech Recognition Module

→ Captures spoken input and converts it into text using the WebkitSpeechRecognition API (browser-based).

Natural Language Processing (NLP) Module

→ Processes the text using tokenization, stopword removal, lemmatization, part-of-speech tagging, synonym mapping, and tense detection.

Animation Dataset

→ Stores predefined Blender-generated video animations for ISL words and fingerspelling (a–z).

Blender Animation Renderer

→ Handles generation and exporting of 3D sign language animations for both full words and individual letters.

Video Merger (FFmpeg Tool)

→ Merges multiple animation clips into a single video to represent the entire sentence coherently.

***Backend Server (Django Framework*)**

→ Manages routing, session handling, authentication, user data, and coordination between frontend and processing modules.

SQLite Database

→ Stores user credentials, input history, favorite animations, and synonym mappings.

This architecture ensures modularity, extensibility, and effective resource management throughout the system's lifecycle.

4.3 Workflow of the System

The system follows a stepwise approach to converting text or speech into ISL animations:

1. User Input:

- The user enters text manually or speaks into the microphone.
- The system captures speech using WebkitSpeechRecognition.

2. Preprocessing:

- Speech input is converted into text.
- NLP techniques, including tokenization, stopword removal, lemmatization, and synonym mapping, are applied.

3. Animation Mapping:

- The system searches for predefined ISL animations in the dataset.
- If a direct ISL equivalent is unavailable, the system generates a finger-spelling animation.

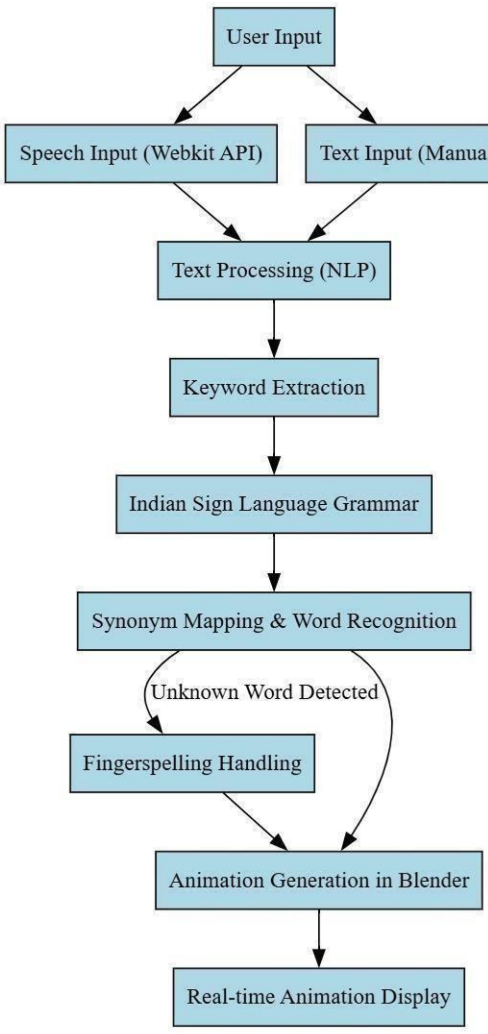
4. Rendering and Display:

- The Blender engine processes the selected animation.
- The rendered animation is displayed in the web interface.

5. User Interaction and Feedback:

- Users can replay animations or modify their input for better accuracy.
- The system allows future enhancements based on user feedback.

This structured workflow ensures real-time, efficient conversion of speech or text into ISL animations while maintaining accuracy.


User Input
Speech Input (Webkit API)
Text Input (Manual)
Text Processing (NLP)
Keyword Extraction
Indian Sign Language Grammar
Synonym Mapping & Word Recognition
Unknown Word Detected
Fingerspelling Handling
Animation Generation in Blender
Real-time Animation Display

4.1.1 Work Flow

4.4 UML Diagrams

Class Diagram

The Class Diagram represents the core structure of the NLP and Blender-Based Sign Language Animation System, detailing key classes and their interactions. It illustrates how different components collaborate to process user input and generate corresponding Indian Sign Language (ISL) animations.

Main Components

1. User Class

- Represents the user interacting with the system by providing input in the form of text or speech.

- If the input is speech, it is processed using speech recognition before further processing.

2. SpeechRecognition Class

- Converts spoken input into text using speech-to-text processing.
- This enables the system to handle both textual and verbal inputs effectively.

3. NLP Processing Class

- Handles Natural Language Processing (NLP) tasks, including extracting key terms and determining if a direct ISL gesture exists for a given word.
- If no direct gesture is found, it further checks for alternative ways to represent the input.

4. SynonymHandler Class

- If a direct ISL gesture is unavailable, this module searches for a synonym that has a corresponding sign language animation.

5. FingerSpelling Class

- When neither a direct gesture nor a synonym is available, the system defaults to finger spelling, where each letter of the word is represented through ISL.

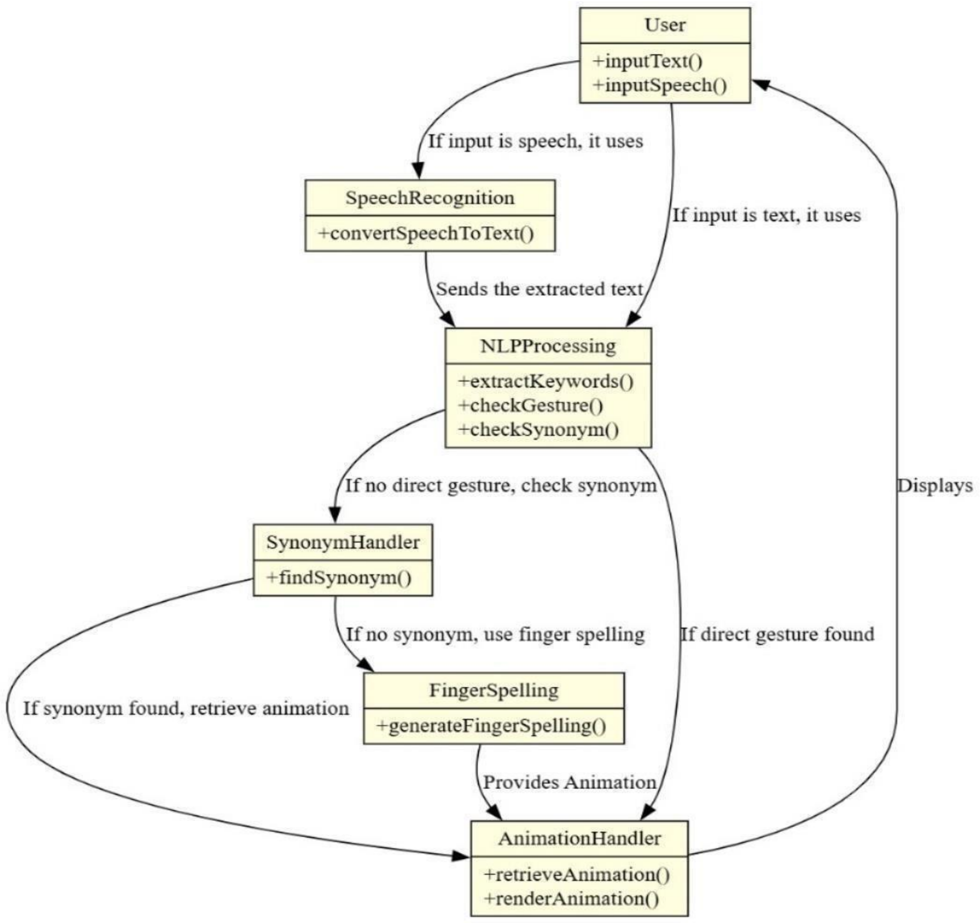
6. AnimationHandler Class

- Retrieves and renders the appropriate animation based on the processed text.
- Ensures smooth display of ISL gestures for effective communication.

Workflow Overview

1. The user provides input (either speech or text).
2. If speech is detected, it is converted into text before processing.
3. The system analyzes the text to identify a matching ISL gesture.
4. If no direct gesture is available, it looks for a synonym; otherwise, it defaults to finger spelling.
5. The appropriate animation is retrieved and displayed to the user.

This structured class-based approach ensures an efficient and scalable system for translating spoken or written language into ISL animations, bridging communication gaps for the hearingimpaired community.



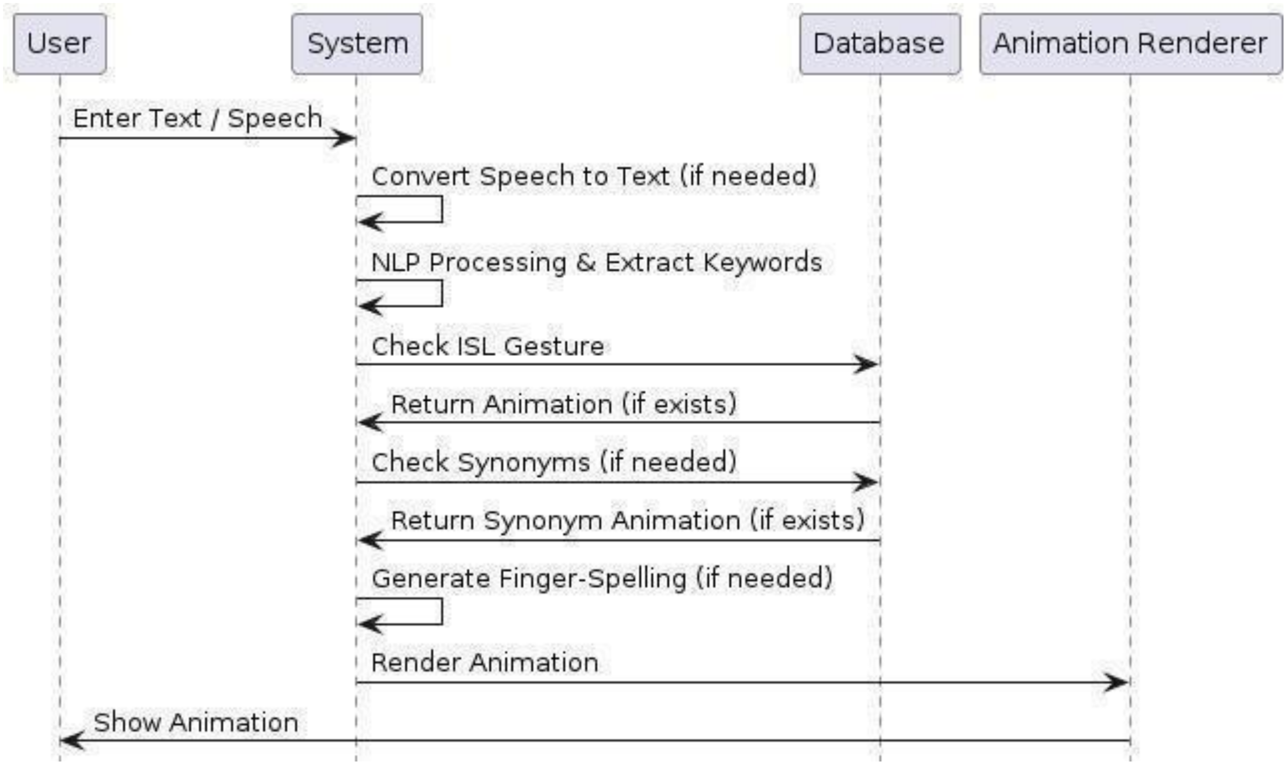
4.1.2 Class Diagram

Sequence Diagram

The Sequence Diagram outlines the step-by-step flow of interaction between system components:

1. The user submits text or speech input.
2. The system processes the input using NLP techniques.
3. The system retrieves the corresponding ISL animation from the dataset.
4. Blender renders and displays the animation.
5. The user views the animated output.

This diagram highlights the logical sequence of operations within the system, ensuring clarity in data processing.


User
System
Database
Animation Renderer
Enter Text / Speech
Convert Speech to Text (if needed)
NLP Processing & Extract Keywords
Check ISL Gesture
Return Animation (if exists)
Check Synonyms (if needed)
Return Synonym Animation (if exists)
Generate Finger-Spelling (if needed)
Render Animation
Show Animation

4.1.3 Sequence Diagram

4.5 System Components

The NLP and Blender-Based Sign Language Animation System is built from multiple integrated components that work in coordination to translate spoken or written input into Indian Sign Language (ISL) animations. Each component plays a vital role in ensuring accurate processing, smooth animation, and user-friendly interaction.

Key Components:

Frontend (Web Interface)

→ Developed using HTML, CSS, and JavaScript, the web interface allows users to enter input, control playback speed, download videos, and navigate between features such as history and favorites.

Backend (Django Framework)

→ Manages all core logic including NLP processing, video merging, user session handling, history tracking, favorites management, and communication between frontend and backend services.

Database (SQLite)

→ Stores user authentication details, text input history, filtered keywords, video paths, and usermarked favorites. Lightweight yet powerful, SQLite supports rapid queries and reliable storage.

Natural Language Processing (NLP) Module

→ Utilizes NLTK and custom synonym mapping to preprocess user input through tokenization, lemmatization, stopword removal, part-of-speech tagging, synonym substitution, and tense detection.

3D Animation Engine (Blender)

→ Animations for ISL gestures (words and fingerspelling letters) are created in Blender, exported as .mp4 clips, and used as the animation dataset.

Video Merging Tool (FFmpeg)

→ After selecting relevant animation clips, FFmpeg merges them into a single cohesive video, improving viewer experience and enabling smooth playback and downloading.

Speech Recognition Module (WebkitSpeechRecognition API)

→ Allows users to provide input through voice. The speech is converted into text within the browser and processed in the same pipeline as typed input.

Each of these components contributes to creating a real-time, personalized, and accessible ISL animation platform, supporting both learning and communication needs for the deaf and hard-ofhearing community.

5. SYSTEM IMPLEMENTATION

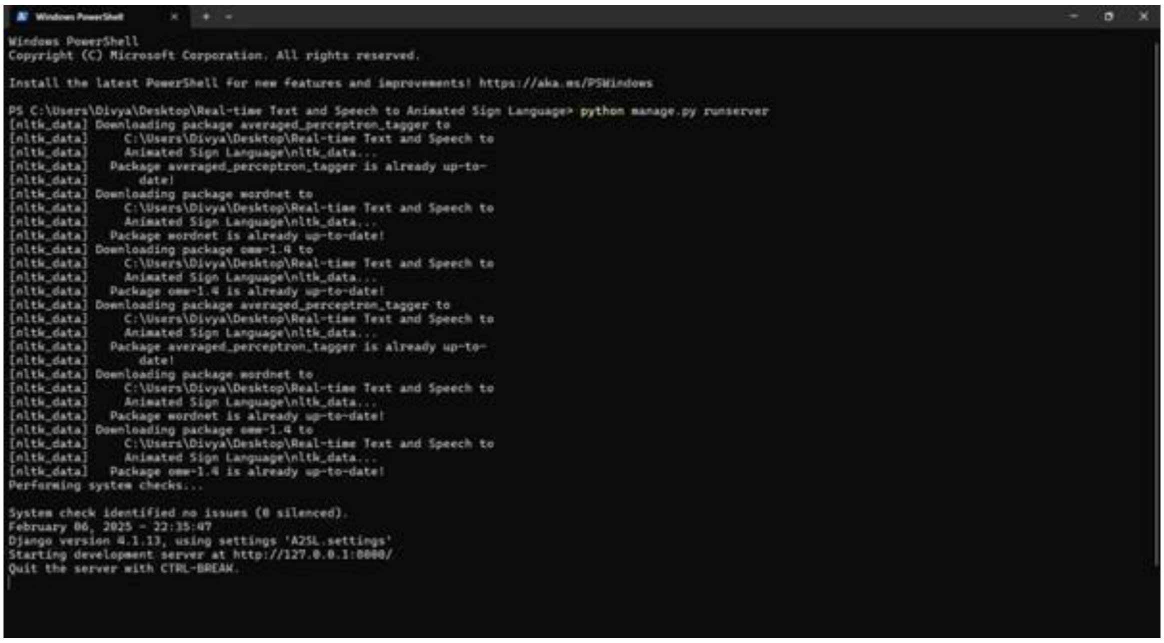
The **NLP and Blender-Based Sign Language Animation System** is developed to bridge the communication gap between individuals who use Indian Sign Language (ISL) and those who rely on spoken or written language. The system enables users to input text or speech, which is then processed through Natural Language Processing (NLP) techniques, mapped to pre-recorded ISL animations, and displayed through a web-based interface using Blenderrendered animations.

The execution of the NLP and Blender-Based Sign Language Animation System begins by setting up the Django development server, which acts as the backend for processing user inputs and rendering animations. The project is implemented as a web-based application, and to run the system, the server must be initialized. This is done by navigating to the project directory in the terminal and executing the command:

python manage.py runserver

This command starts the Django development server, enabling users to access the application through a web browser.

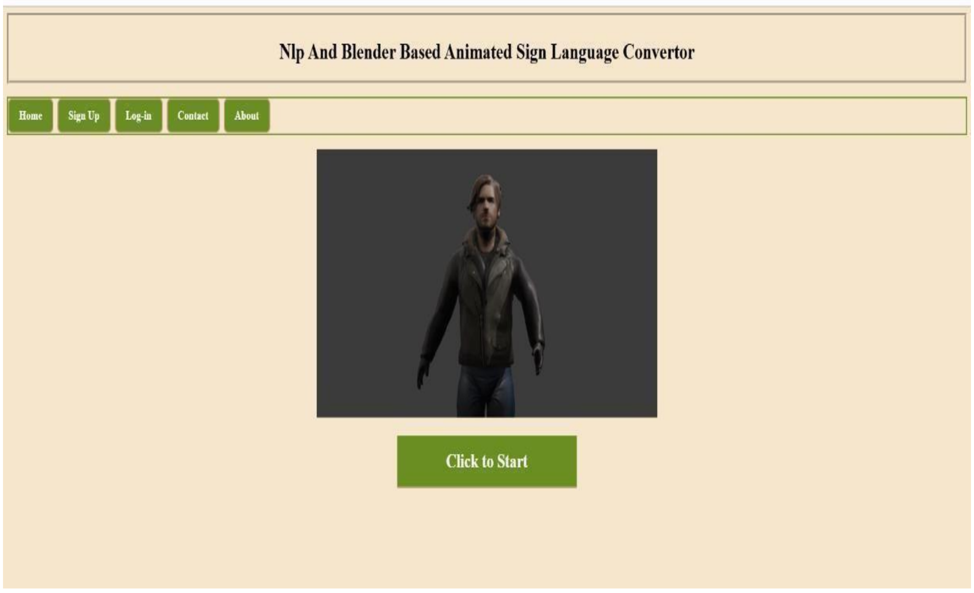
Once the server is running, the terminal displays a local server address, typically http://127.0.0.1:8000/, which serves as the entry point for users to interact with the system. Upon opening this address in a browser, users are directed to the home page, where they can log in or sign up for an account. The authentication system ensures that user credentials are securely managed through an SQLite database.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Divya\Desktop\Real-time Text and Speech to Animated Sign Language> python manage.py runserver
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Divya\Desktop\Real-time Text and Speech to
[nltk_data]     Animated Sign Language\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Divya\Desktop\Real-time Text and Speech to
[nltk_data]     Animated Sign Language\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Divya\Desktop\Real-time Text and Speech to
[nltk_data]     Animated Sign Language\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Divya\Desktop\Real-time Text and Speech to
[nltk_data]     Animated Sign Language\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Divya\Desktop\Real-time Text and Speech to
[nltk_data]     Animated Sign Language\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Divya\Desktop\Real-time Text and Speech to
[nltk_data]     Animated Sign Language\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Performing system checks...

System check identified no issues (0 silenced).
February 06, 2025 - 22:35:47
Django version 4.1.13, using settings 'A2SL.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

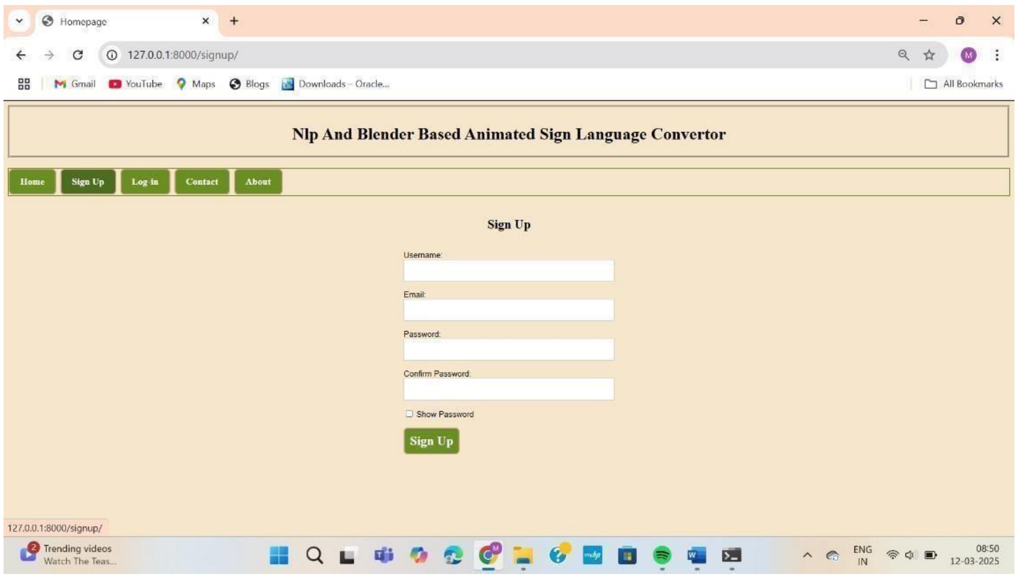
5.0.1 Project Execution process

The home page serves as the main navigation point, providing users with access to essential system functionalities. It gives an overview of how the platform works and directs users to either log in or sign up for an account.

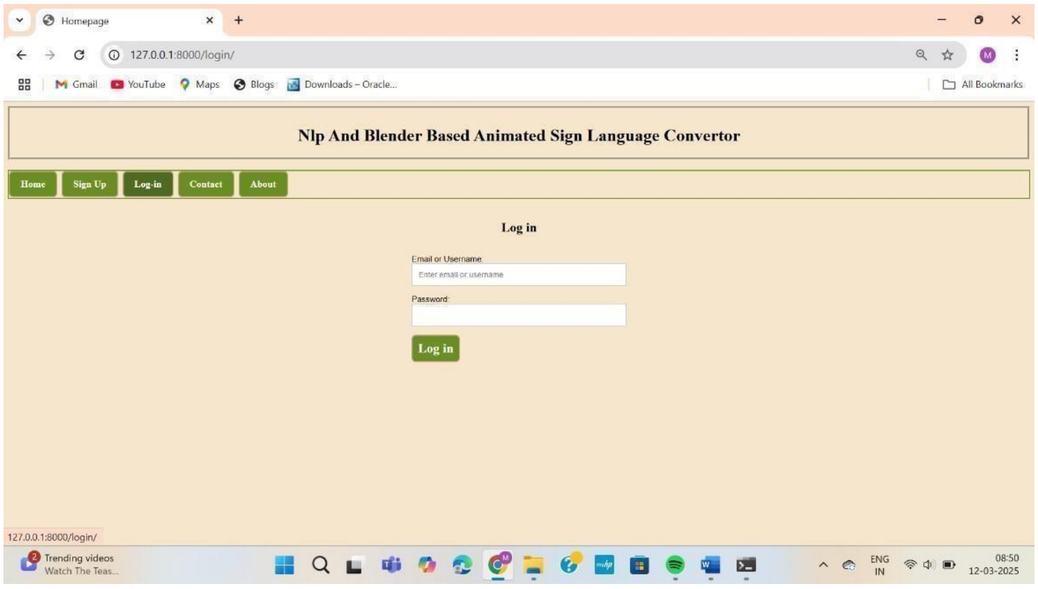


5.0.2 Web Application Interface

For security and personalized interaction, users are required to log in. The login page allows existing users to enter their credentials, while new users must register via the sign-up page, where they provide details such as username, email, and password. These credentials are securely stored in an SQLite database, ensuring authentication and preventing unauthorized access.



5.0.3 Sign up page



Nlp And Blender Based Animated Sign Language Convertor

Log in

Email or Username:

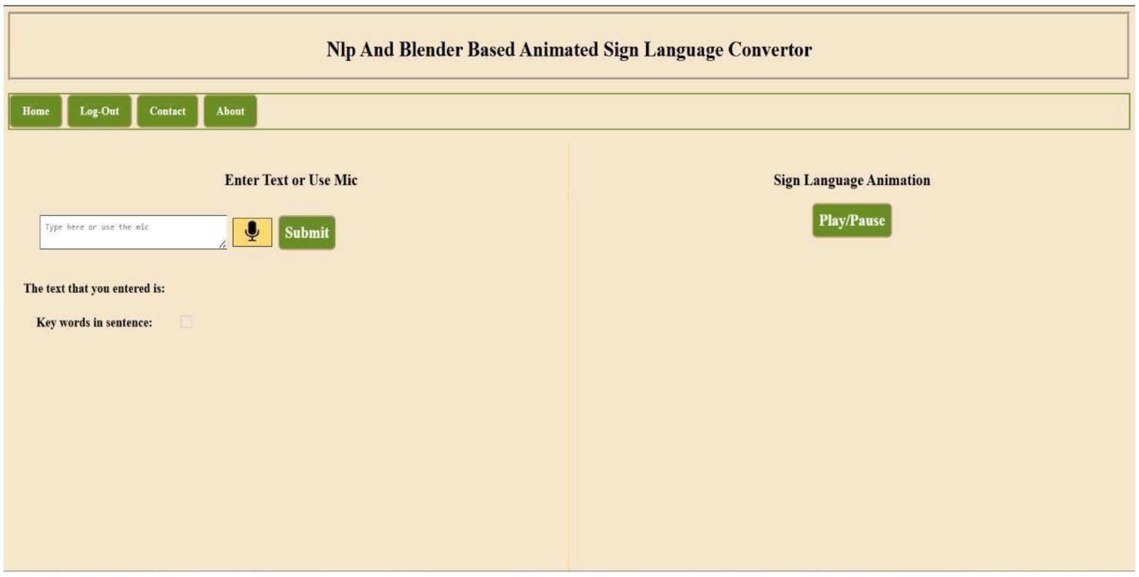
Enter email or username

Password:

5.0.4 Login Page

Upon successful login, users are directed to the start page, which acts as a transition to the core functionalities of the system. Here, users can choose to input text manually or activate speech recognition for real-time processing.

The input section is designed to be user-friendly, allowing users to enter sentences manually in the text input box or use the microphone to convert speech into text through WebkitSpeechRecognition. This ensures accessibility for those who may not be comfortable with typing. Once the input is received, NLP processing breaks it down into individual words, performing tokenization, lemmatization, and synonym mapping to match words with their corresponding ISL animations.



Nlp And Blender Based Animated Sign Language Convertor

Enter Text or Use Mic

Sign Language Animation

The text that you entered is:

Key words in sentence:

5.0.5 Home Page

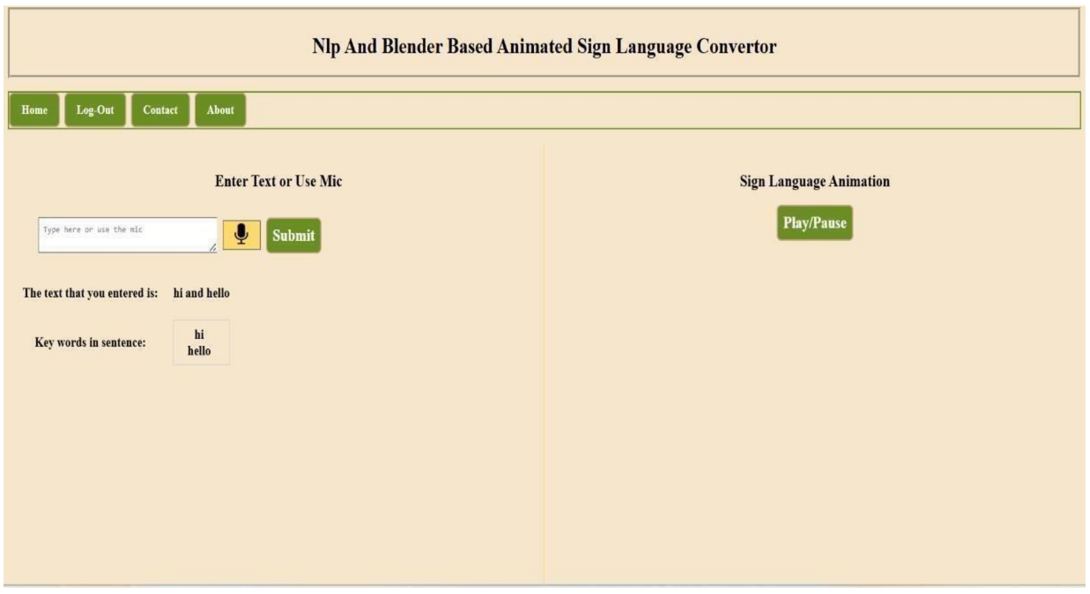
If an animation exists for a given word, it is fetched and displayed in real-time. However, if a word does not have a predefined ISL sign, the system dynamically generates letter-by-letter signing using fingerspelling animations. This ensures that even uncommon words can still be represented in ISL.

The animation playback system is powered by Blender, which renders high-quality 3D animations of ISL gestures. These animations are loaded dynamically based on user input and displayed in a smooth sequence to maintain visual clarity. Users can pause, replay, or slow down animations as needed.

The accuracy of the system depends largely on the dataset of animations. The database currently contains 75 animations, including the 26 alphabets, numbers, and frequently used words. The dataset is continuously expanding to cover more ISL gestures and phrases to improve accessibility and effectiveness.

To ensure a seamless experience, the backend of the system is powered by Django, which manages communication between the user interface, NLP processing, and animation rendering. The Pro Exec page provides insights into system execution, performance, and optimization details. Django efficiently handles user requests, processes text input, retrieves animations, and manages authentication.

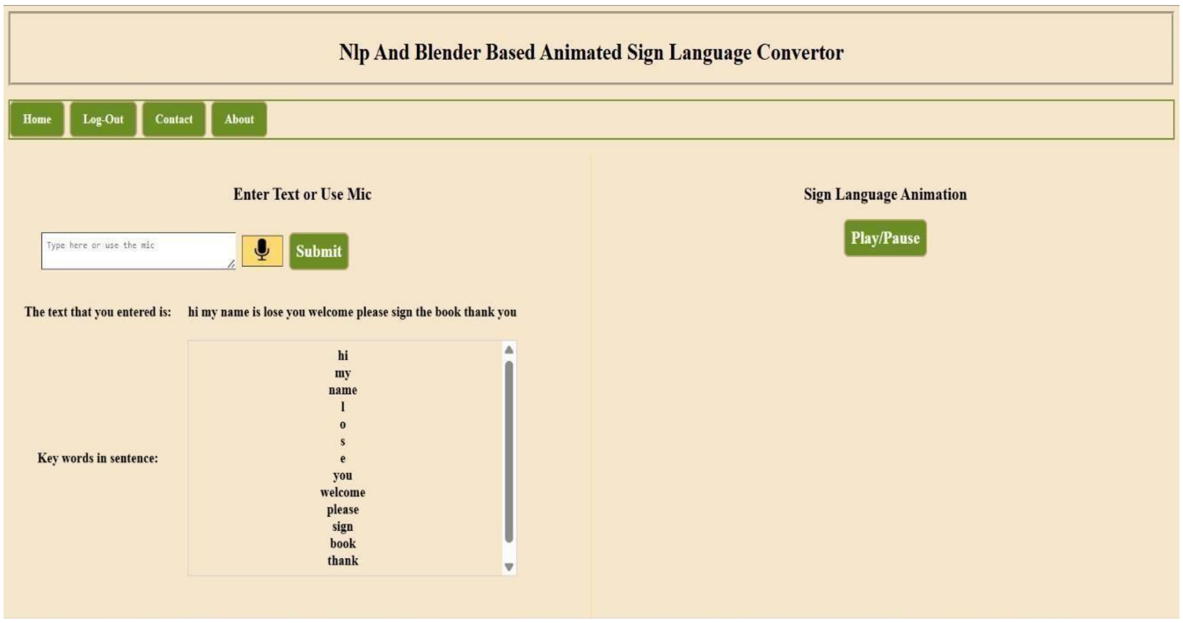
Users who wish to provide feedback or seek assistance can navigate to the contact page, where they can submit inquiries or suggest improvements. This helps in refining the platform based on real-world user experiences.


Nlp And Blender Based Animated Sign Language Convertor

The text that you entered is: hi and hello

Key words in sentence: hi hello

5.1.1 Input Text


Nlp And Blender Based Animated Sign Language Convertor

Enter Text or Use Mic

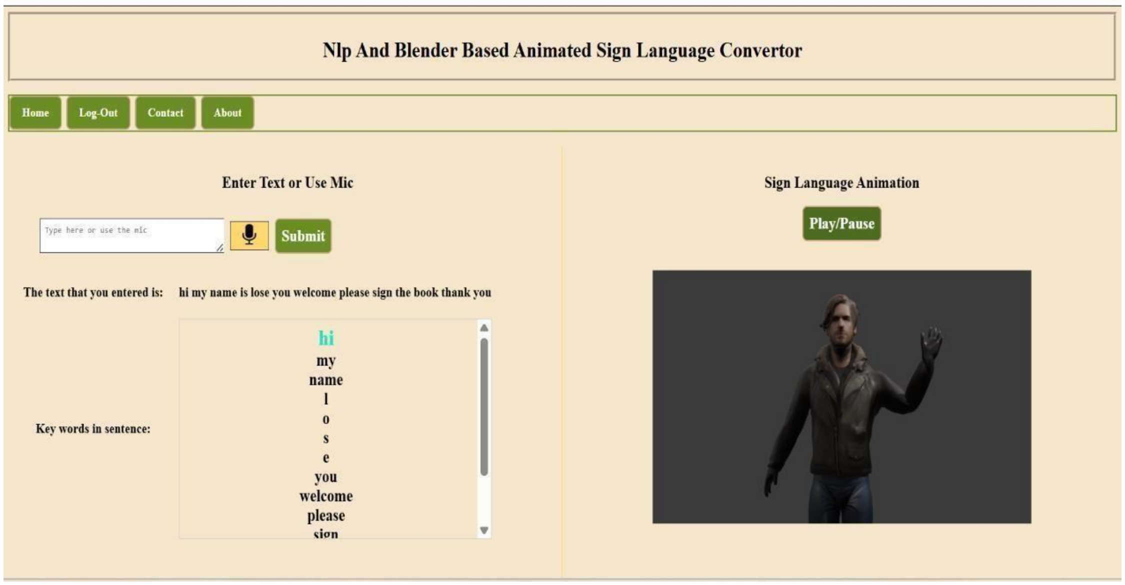
Sign Language Animation

The text that you entered is: hi my name is lose you welcome please sign the book thank you

Key words in sentence: hi my name l o s e you welcome please sign book thank

5.1.2 Input text is processed in this format

The system is designed with scalability in mind, allowing future enhancements such as AIpowered gesture recognition, real-time gesture synthesis, and support for additional languages. The integration of NLP, WebkitSpeechRecognition, and Blender-based animations makes this platform a powerful tool for promoting ISL learning and communication accessibility.


Nlp And Blender Based Animated Sign Language Convertor

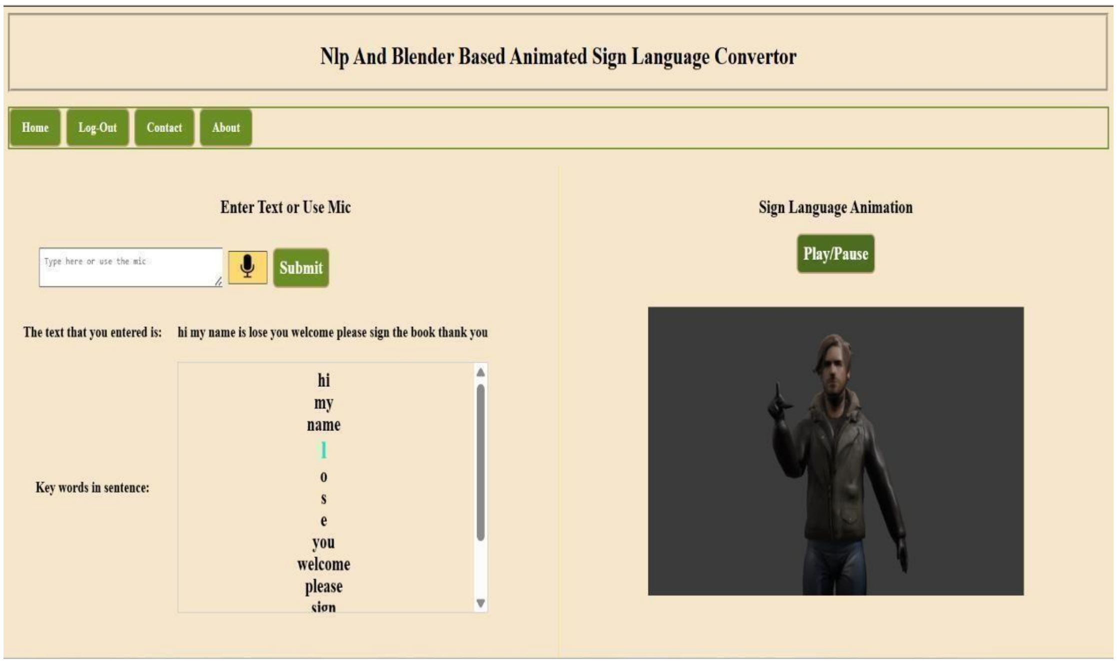
Enter Text or Use Mic

Sign Language Animation

The text that you entered is: hi my name is lose you welcome please sign the book thank you

Key words in sentence: hi my name l o s e you welcome please sign

5.2.1 Animation being played for the corresponding text



The text that you entered is: hi my name is lose you welcome please sign the book thank you

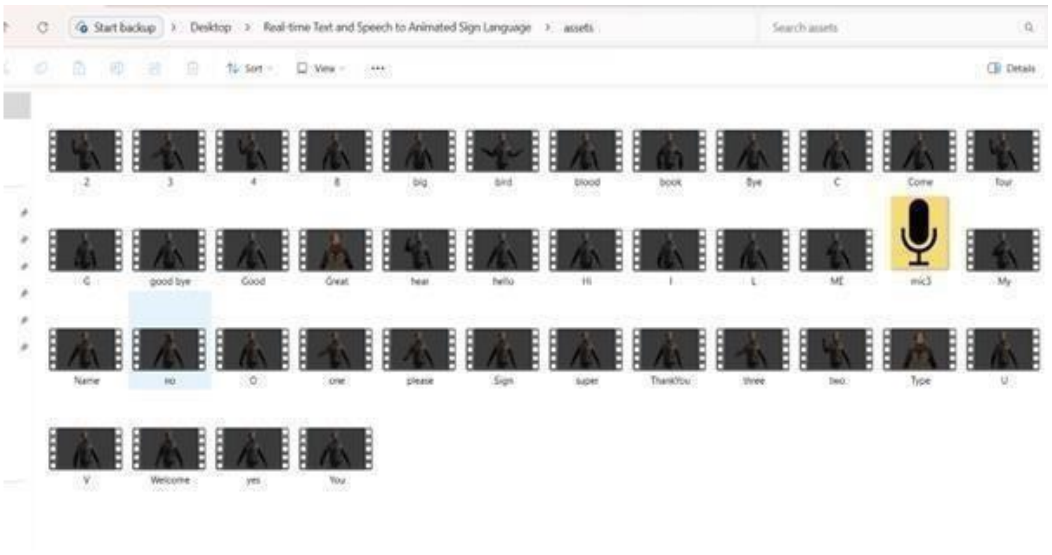
Key words in sentence: hi my name l o s e you welcome please sign

5.2.2 Finger spelling for words with no sign in the dataset

The NLP and Blender-Based Sign Language Animation System ensures that users can receive accurate Indian Sign Language (ISL) animations for both common words and words that do not have a direct ISL sign. The system provides animations by first checking if a word exists in the predefined ISL animation dataset. If the word is found, its corresponding Blenderrendered animation is retrieved and displayed in real time.

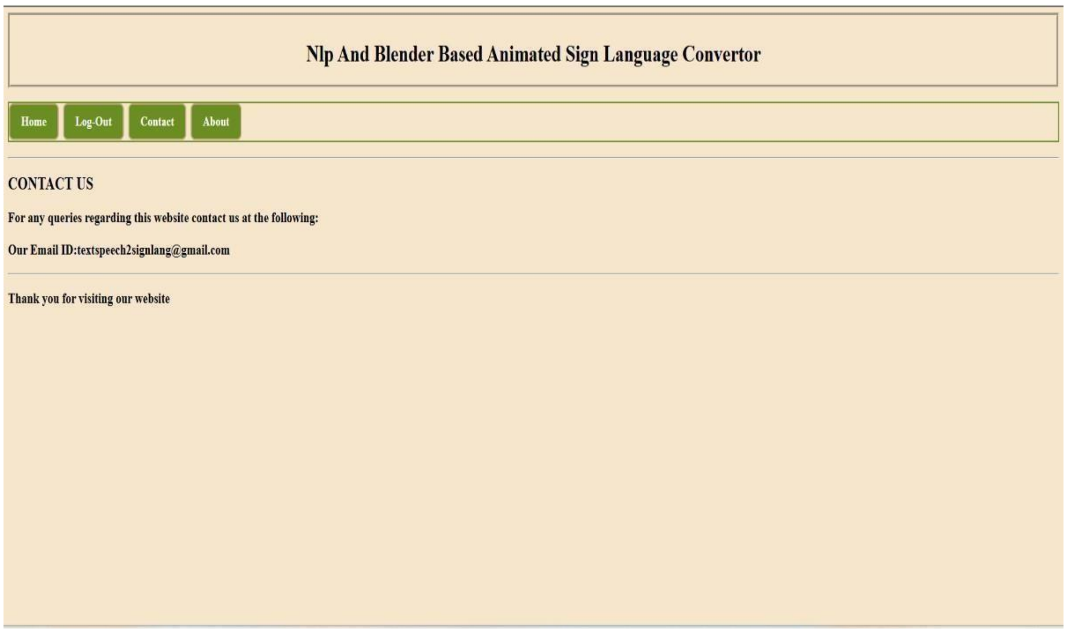
For words that do not have a direct ISL animation, the system automatically switches to finger spelling, where each letter of the word is displayed sequentially using predefined alphabet animations. This approach ensures that every word can be represented, even if it is not explicitly stored in the animation dataset.

The process begins when a user enters a word or phrase. The Natural Language Processing (NLP) module analyses the input and attempts to match it with a stored ISL animation. If a match is found, the animation is fetched and played. However, if no predefined sign is available, the word is broken down into individual letters, and the corresponding finger spelling animations are displayed one by one.



5.3.1 Dataset created using Blender

The contact page serves as a dedicated section where users can seek assistance, provide feedback, or suggest improvements for the NLP and Blender-Based Sign Language Animation System. This page allows users to directly communicate with the development team, ensuring continuous improvements based on real-world usage and feedback.



Nlp And Blender Based Animated Sign Language Convertor

Home Log-Out Contact About

CONTACT US

For any queries regarding this website contact us at the following:

Our Email ID:textspeech2signlang@gmail.com

Thank you for visiting our website

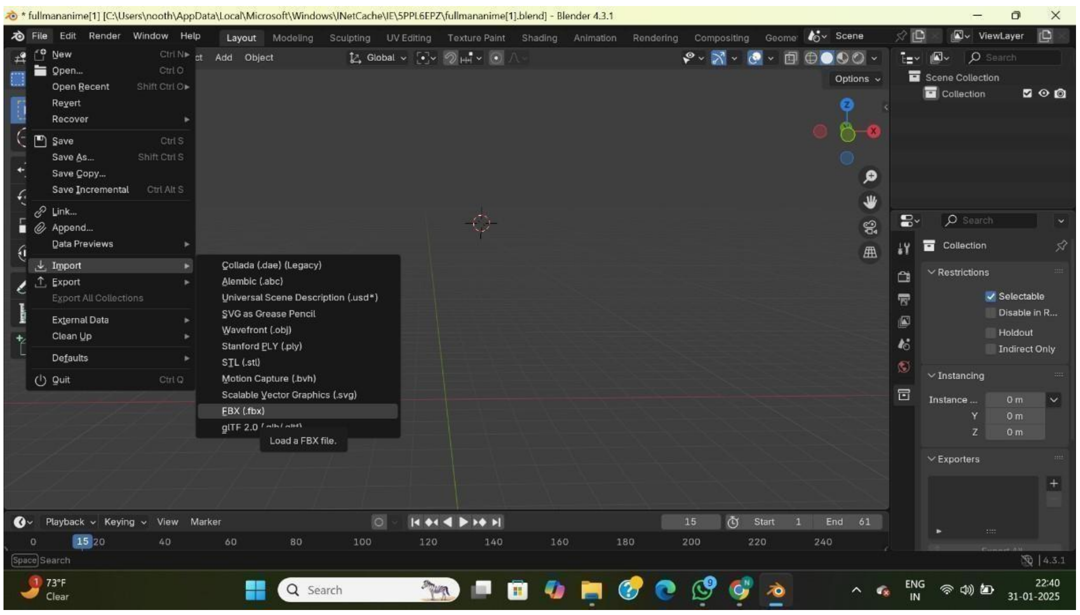
5.3.2 Contact Page

Animation Creation Process:

The NLP and Blender-Based Sign Language Animation System uses Blender to create and render Indian Sign Language (ISL) animations. The animation creation process ensures that the hand gestures are accurate, smooth, and visually clear for effective communication. Each animation is designed to replicate real-life ISL gestures, ensuring accessibility for users. The animation workflow follows several structured steps, from importing assets to rendering the final output, which are then stored in the system for real-time playback.

1. *Importing Assets into Blender*

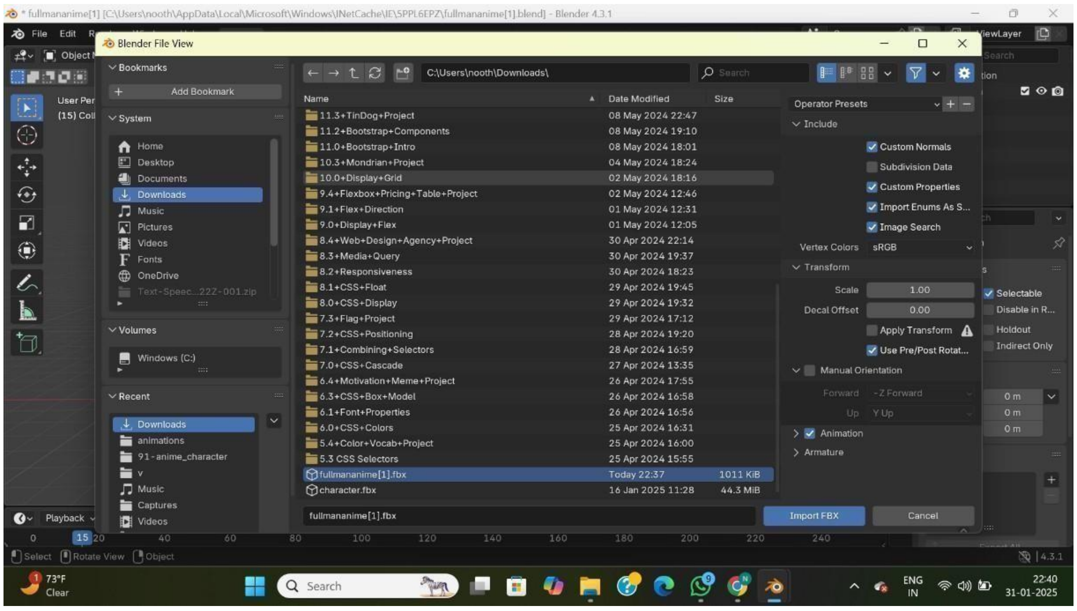
The animation process starts with importing the necessary assets into Blender. This includes 3D hand models, skeletal rigging, and texture files. The model used for animations is designed to be highly detailed and flexible, allowing for precise movement of fingers, wrist, and palm. Blender supports various 3D file formats, enabling seamless integration of assets.



5.3.3 Importing model into Blender

2. *Setting the File Location*

To ensure smooth execution and retrieval of animations, a designated file structure is created. The file location is set in Blender, defining directories for animation sequences, textures, and motion data. Proper file organization ensures that animations are stored correctly and can be easily retrieved by the system when needed.



C:\Users\nooth\Downloads\

Name	Date Modified	Size
11.3+TinDog+Project	08 May 2024 22:47	
11.2+Bootstrap+Components	08 May 2024 19:10	
11.0+Bootstrap+Intro	08 May 2024 18:01	
10.3+Mondrian+Project	04 May 2024 18:24	
10.0+Display+Grid	02 May 2024 18:16	
9.4+Flexbox+Pricing+Table+Project	02 May 2024 12:46	
9.1+Flex+Direction	01 May 2024 12:31	
9.0+Display+Flex	01 May 2024 12:05	
8.4+Web+Design+Agency+Project	30 Apr 2024 22:14	
8.3+Media+Query	30 Apr 2024 19:37	
8.2+Responsiveness	30 Apr 2024 18:23	
8.1+CSS+Float	29 Apr 2024 19:45	
8.0+CSS+Display	29 Apr 2024 19:32	
7.3+Flag+Project	29 Apr 2024 17:12	
7.2+CSS+Positioning	28 Apr 2024 19:20	
7.1+Combining+Selectors	28 Apr 2024 16:59	
7.0+CSS+Cascade	27 Apr 2024 13:35	
6.4+Motivation+Meme+Project	26 Apr 2024 17:55	
6.3+CSS+Box+Model	26 Apr 2024 16:58	
6.1+Font+Properties	26 Apr 2024 16:56	
6.0+CSS+Colors	25 Apr 2024 16:31	
5.4+Color+Vocab+Project	25 Apr 2024 16:00	
5.3 CSS Selectors	25 Apr 2024 15:55	
fullmananime[1].fbx	Today 22:37	1011 KiB
character.fbx	16 Jan 2025 11:28	44.3 MiB

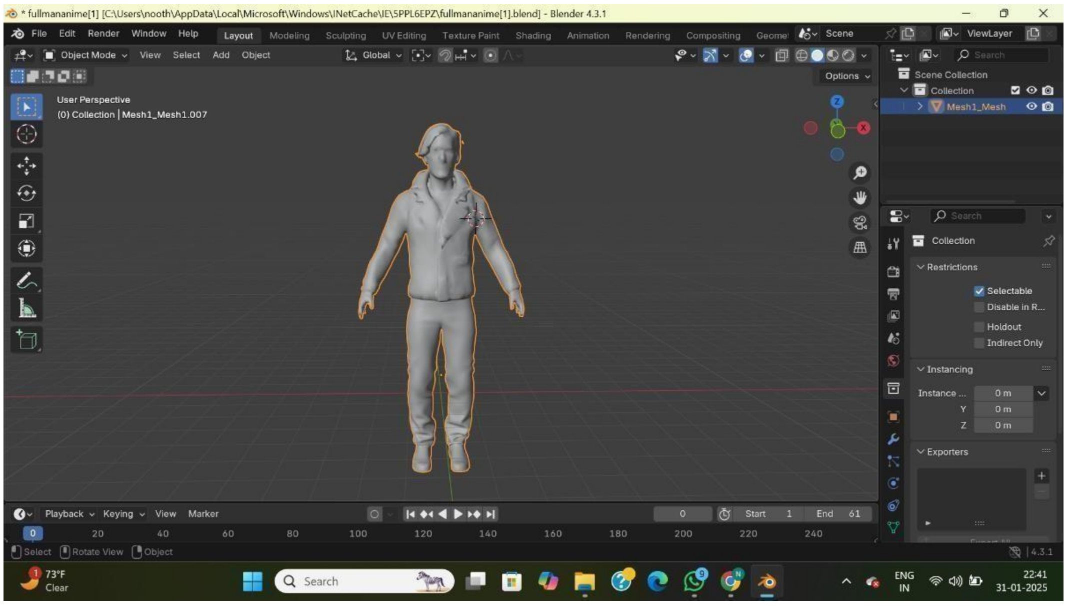
fullmananime[1].fbx

Include: Custom Normals ✓; Subdivision Data; Custom Properties ✓; Import Enums As S... ✓; Image Search ✓; Vertex Colors sRGB
Transform: Scale 1.00; Decal Offset 0.00; Apply Transform; Use Pre/Post Rotat... ✓; Manual Orientation; Forward -Z Forward; Up Y Up
Animation ✓
Armature

5.3.4 Setting the location for the animation to be downloaded

3. Designing and rigging the 3D Hand Model

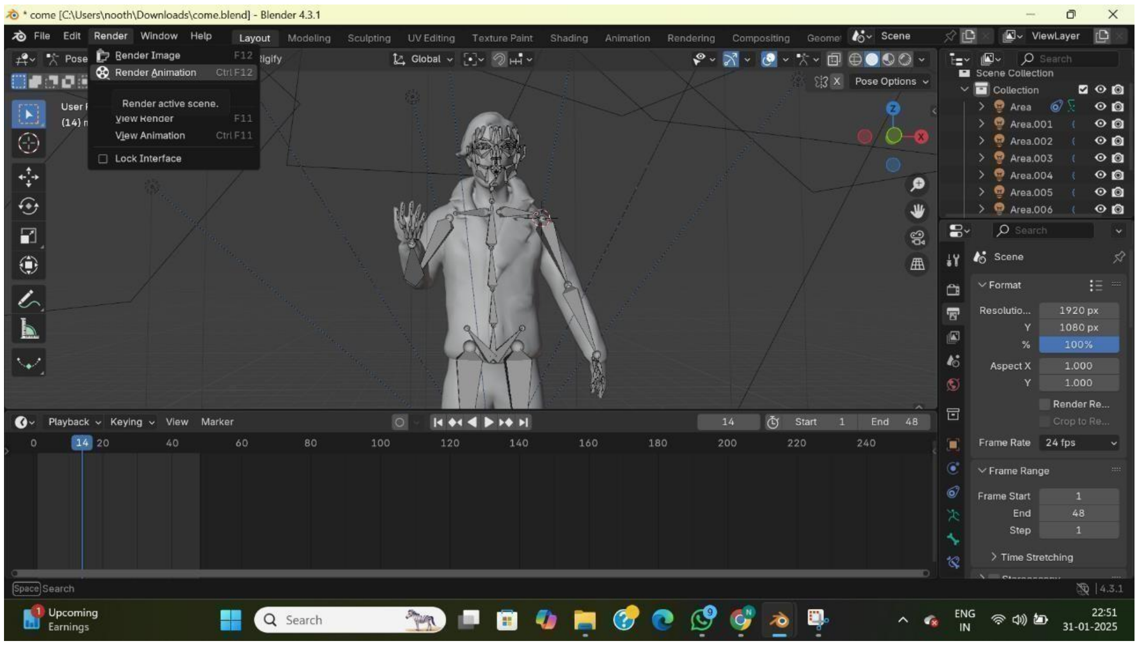
The 3D model of the hand serves as the core element for ISL animations. The model is carefully structured with realistic finger proportions, wrist movement, and articulation points, ensuring that the gestures are natural and easy to interpret. The rigging process involves adding bones and control points that define how different parts of the hand move. Blender's inverse kinematics (IK) constraints allow smooth transitions between different signs, making animations more dynamic.



5.3.5 3D model created for the creation of Animations

4. *Setting Keyframes for Animation*

Once the 3D model is rigged, keyframes are set in Blender's animation timeline to define the movement of each ISL gesture. Keyframes mark the start, midpoint, and end positions of each sign, ensuring smooth hand movements. By adjusting the interpolation between keyframes, the animation achieves fluid motion, preventing robotic or stiff gestures. Each ISL sign is animated frame by frame to maintain accuracy and clarity.

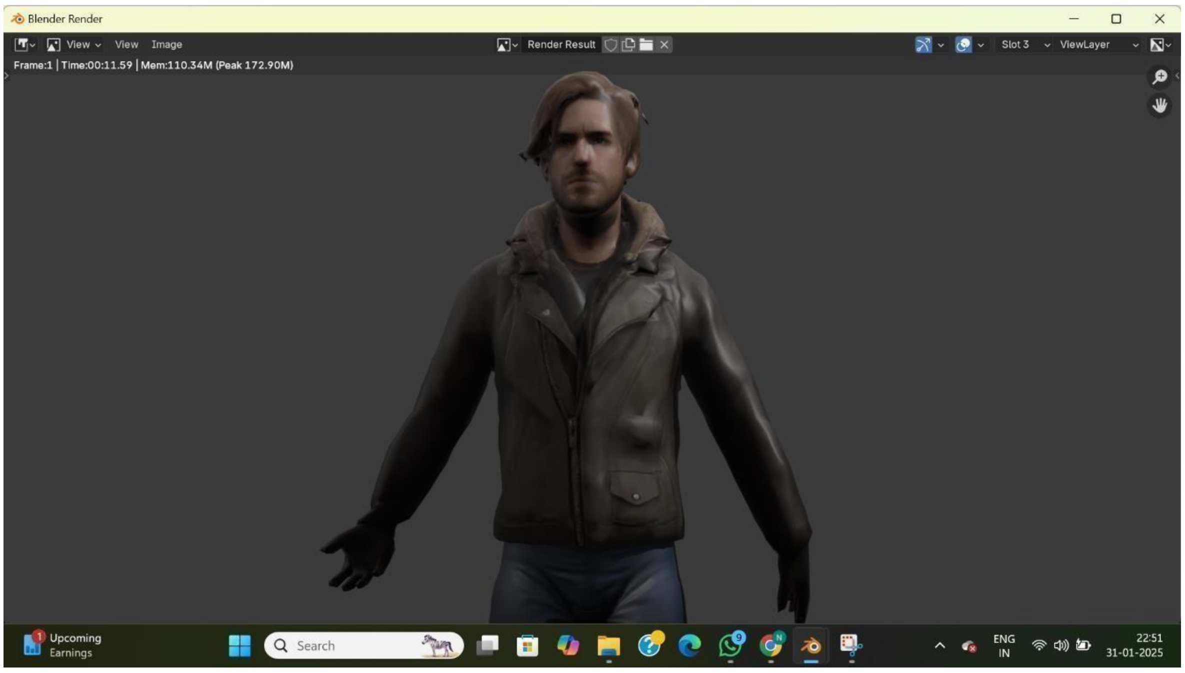


Resolution X 1920 px
Y 1080 px
% 100%
Aspect X 1.000
Y 1.000
Frame Rate 24 fps
Frame Start 1
End 48
Step 1

5.3.6 he animation timeline with keyframes

5. *Rendering the Final Animation Output*

After keyframe adjustments, the rendering process converts the animated hand gestures into MP4 files with optimized frame rate and resolution. The system ensures that each animation plays smoothly without lag, enhancing the user experience. The Blender rendering engine processes the animation sequences, applying lighting, shading, and movement smoothing to enhance realism.



5.3.7 Animation Rendering

Once rendered, the animations are stored in the system dataset, ready for integration. When a user inputs a word, the system searches for the corresponding ISL animation and plays it in real-time. In cases where a word is not available in the dataset, the system generates finger spelling animations dynamically.

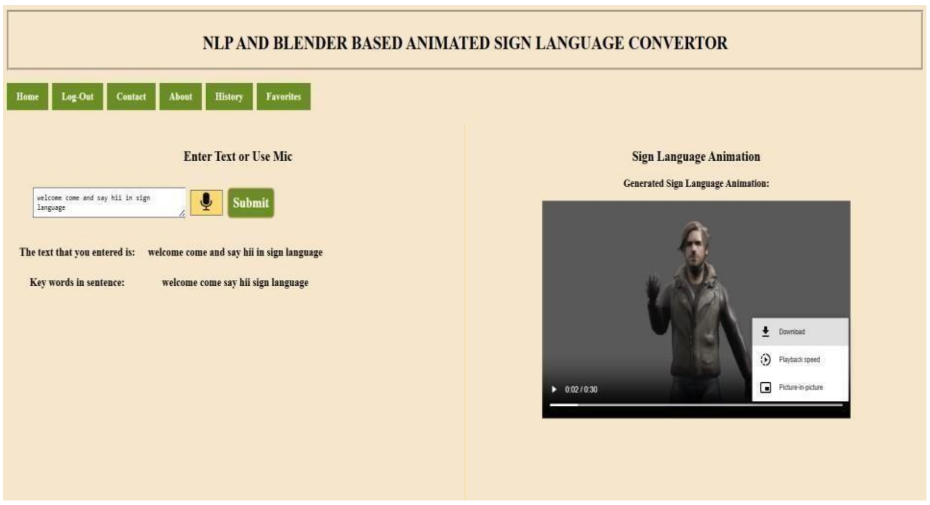
The successful implementation of the NLP and Blender-Based Sign Language Animation System ensures seamless conversion of text and speech into Indian Sign Language (ISL) animations using Natural Language Processing (NLP), WebkitSpeechRecognition, and Blenderrendered gestures. By integrating a structured backend (Django), real-time animation retrieval, and dynamic finger spelling, the system provides an interactive and accessible platform for ISL communication. This implementation sets the foundation for future advancements, such as expanding the animation dataset and enhancing NLP accuracy, making it a scalable solution for bridging communication gaps in the digital space.

5.4 Video Generation and Control Features

To enhance the viewing experience and make the animations user-friendly, additional playback features are incorporated into the system.

Users are provided with a **video speed control dropdown (Fig 5.18)** that allows them to slow down or speed up the animation playback according to their comfort. This is particularly useful for learners who may want to study signs at a slower pace for better understanding. Additionally, a **download button (Fig 5.17)** is integrated, enabling users to save the generated sign language animation videos locally. This feature is helpful for offline learning, teaching purposes, or sharing sign animations with others.

These interactive features make the system more dynamic, flexible, and accessible to a wider audience.


NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE CONVERTOR

Enter Text or Use Mic

welcome come and say hii in sign language

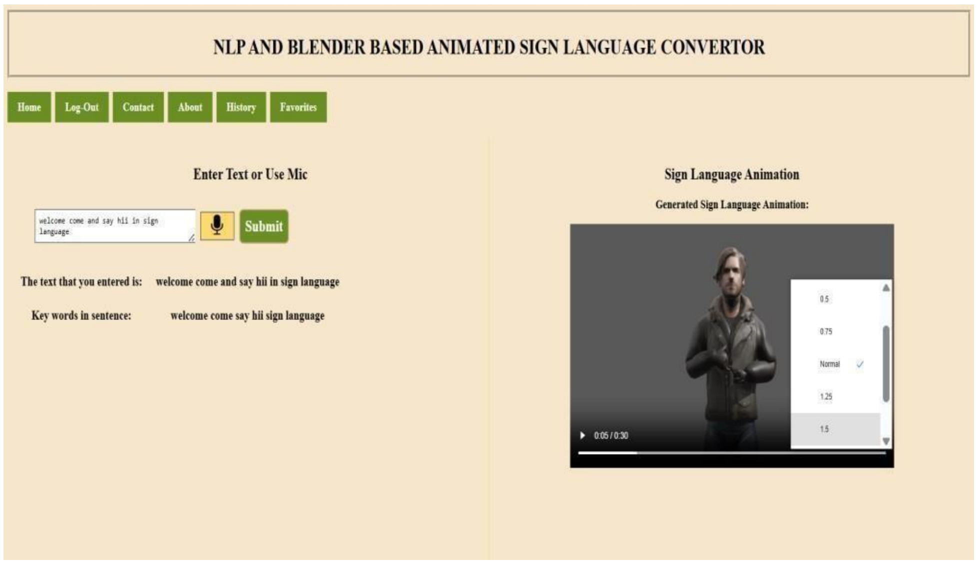
The text that you entered is: welcome come and say hii in sign language

Key words in sentence: welcome come say hii sign language

Sign Language Animation

Generated Sign Language Animation:

5.4.1 download button


NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE CONVERTOR

Enter Text or Use Mic

welcome come and say hii in sign language

The text that you entered is: welcome come and say hii in sign language

Key words in sentence: welcome come say hii sign language

Sign Language Animation

Generated Sign Language Animation:

Playback speed: Normal ✓

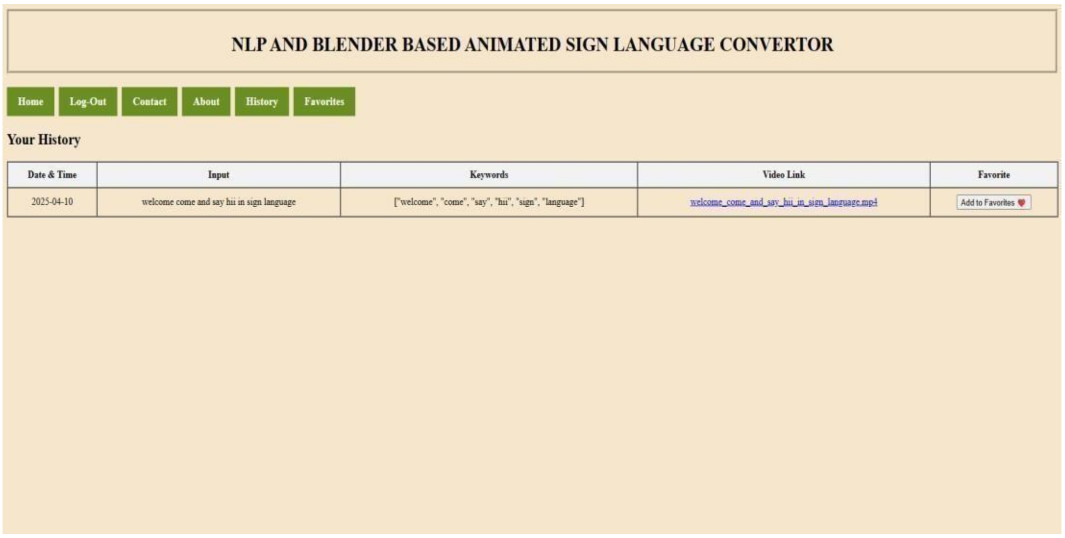
5.4.2 video speed control dropdown

5.5 History and Favorites Features

To enhance personalization and user interaction, the NLP and Blender-Based Sign Language Animation System includes two important features: History and Favorites.

History Tracking

The system automatically maintains a history of all translated inputs made by each user. Every time a user generates an animation, the corresponding input text, extracted keywords, and generated video path are stored and timestamped. This allows users to review their past translations, revisit previously generated sign animations, and avoid reprocessing the same inputs. The History page is accessible through the navigation bar and offers direct links to replay the associated videos.


NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE CONVERTOR

Your History

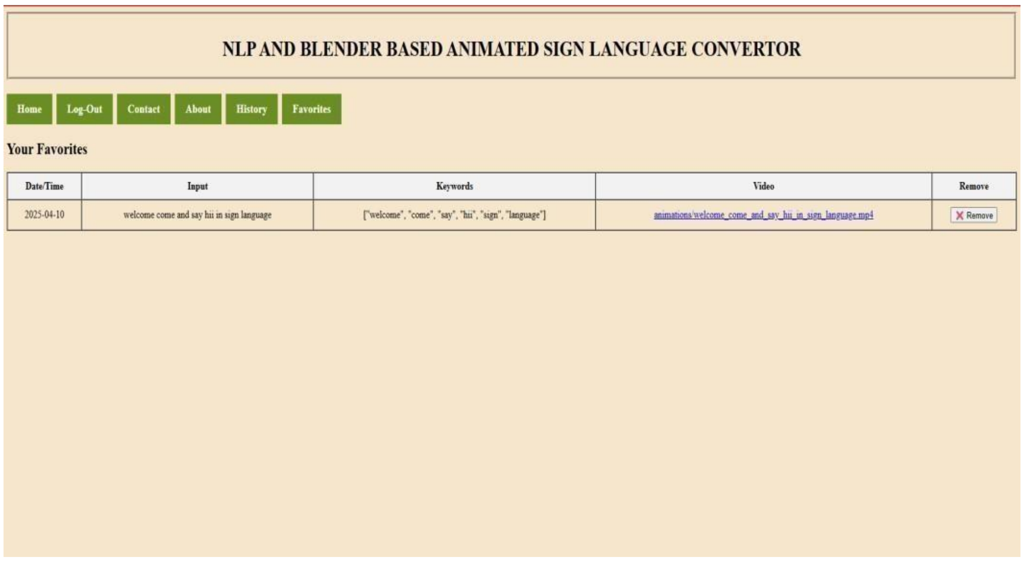
Date & Time	Input	Keywords	Video Link	Favorite
2025-04-10	welcome come and say hii in sign language	["welcome", "come", "say", "hii", "sign", "language"]	welcome_come_and_say_hii_in_sign_language.mp4	Add to Favorites

5.5.1 History Page showing past translated inputs with date, keywords, and video links

Favourites Management

Users can mark any past translation as a favorite for quicker access in the future. On the History page, a simple click allows users to add an entry to the Favorites list, which can later be managed separately under the Favorites tab. This feature is especially useful for frequent learners, educators, and individuals who repeatedly use specific sign language phrases. Users can also remove items from Favorites at any time to keep their list organized.

Both features are user-specific and secured by authentication, ensuring that personal data such as favorites and history logs are only visible to the respective logged-in user. These enhancements make the system more engaging, reusable, and learner-friendly, helping users retain and efficiently manage their learning progress.



NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE CONVERTOR

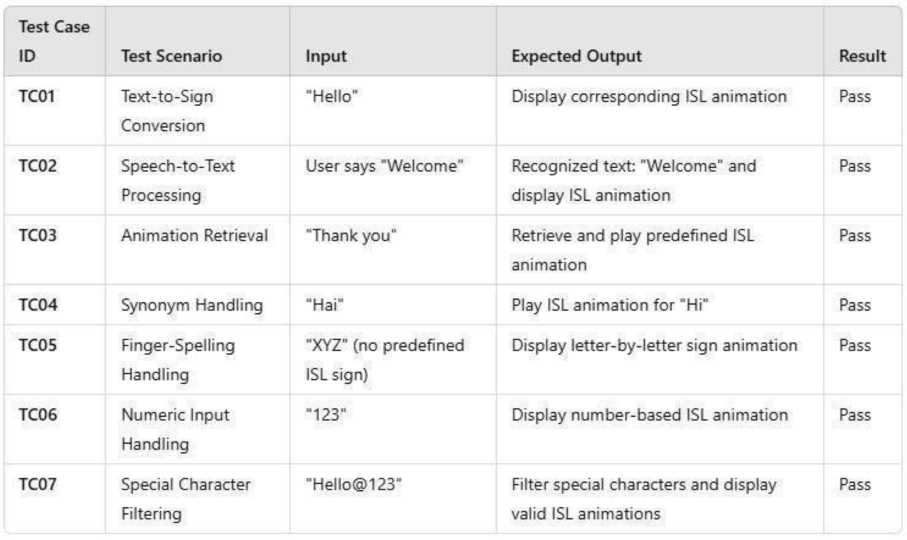
Your Favorites

Date/Time	Input	Keywords	Video	Remove
2025-04-10	welcome come and say hii in sign language	["welcome", "come", "say", "hii", "sign", "language"]	animations/welcome_come_and_say_hii_in_sign_language.mp4	Remove

5.5.2 Favorites Page displaying saved animations for quick access and reuse

6. SYSTEM TESTING

To ensure the accuracy and reliability of the system, multiple test cases were designed and executed. The table below summarizes the key test cases performed:



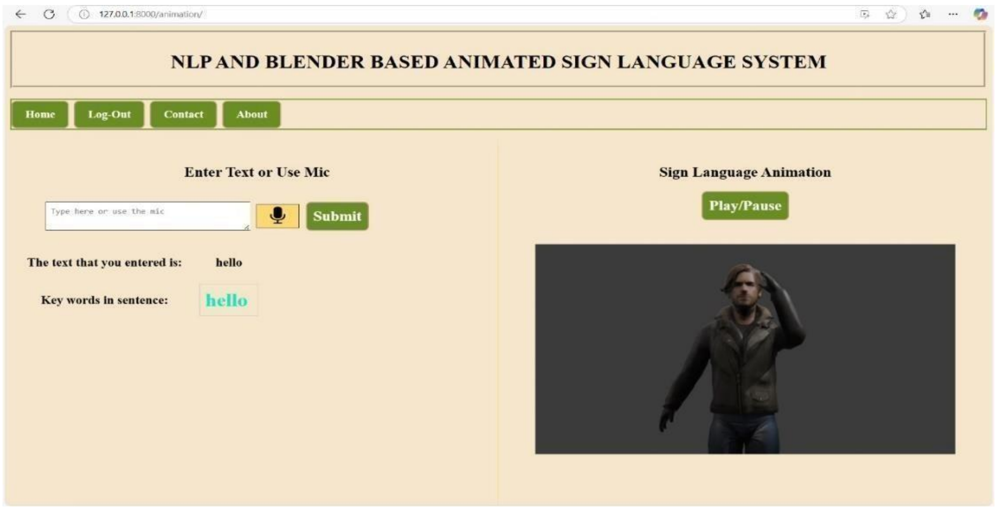
Test Case ID	Test Scenario	Input	Expected Output	Result
TC01	Text-to-Sign Conversion	"Hello"	Display corresponding ISL animation	Pass
TC02	Speech-to-Text Processing	User says "Welcome"	Recognized text: "Welcome" and display ISL animation	Pass
TC03	Animation Retrieval	"Thank you"	Retrieve and play predefined ISL animation	Pass
TC04	Synonym Handling	"Hai"	Play ISL animation for "Hi"	Pass
TC05	Finger-Spelling Handling	"XYZ" (no predefined ISL sign)	Display letter-by-letter sign animation	Pass
TC06	Numeric Input Handling	"123"	Display number-based ISL animation	Pass
TC07	Special Character Filtering	"Hello@123"	Filter special characters and display valid ISL animations	Pass

6.1.1 Test Cases for testing the application

The following test cases were conducted to evaluate the system's functionality, accuracy, and reliability in converting text and speech inputs into Indian Sign Language (ISL) animations.

6.1 Text-to-Sign Conversion

This test validates whether a given text input is correctly converted into its corresponding ISL animation. For example, when the user enters **"Hello"**, the system successfully retrieves and **displays the ISL sign animation for "Hello."**



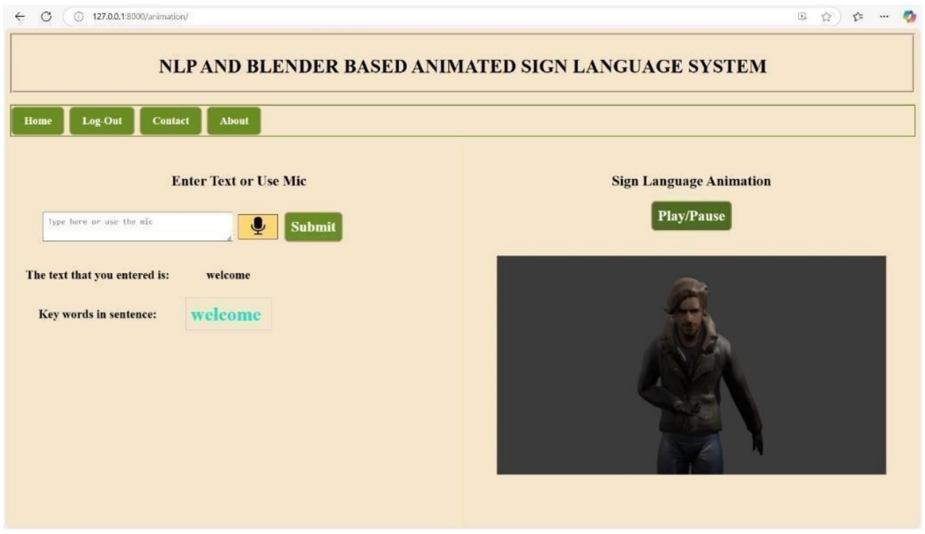
The text that you entered is: hello

Key words in sentence: hello

6.1.2 Test case - 01

6.2 Speech-to-Text Processing

This test ensures that spoken words are accurately transcribed and converted into ISL animations. When the user says **"Welcome"**, the system recognizes the speech, converts it into text, and then plays the appropriate ISL animation.



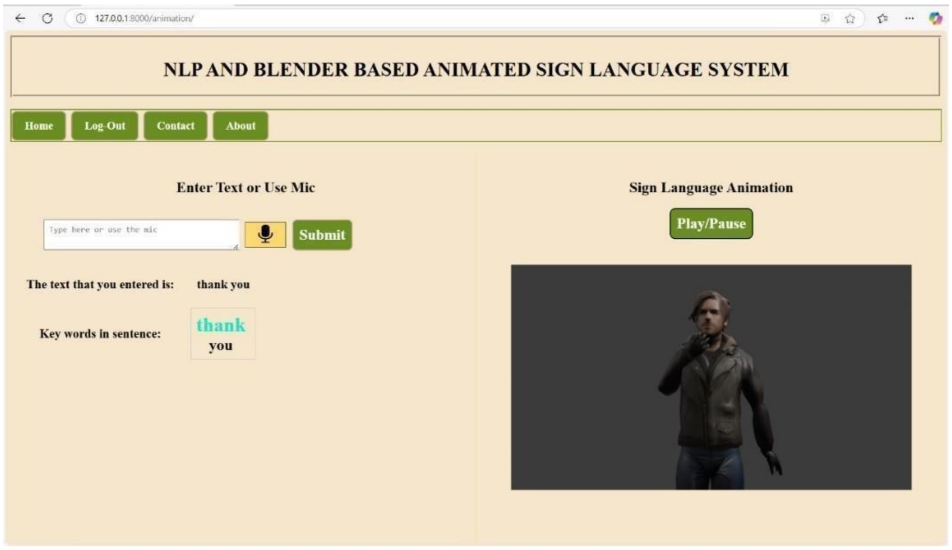
The text that you entered is: welcome

Key words in sentence: welcome

6.1.3 Test case - 02

6.3 Animation Retrieval

This test checks whether the system correctly retrieves stored ISL animations for predefined words. For instance, entering **"Thank you"** triggers the correct animation playback.



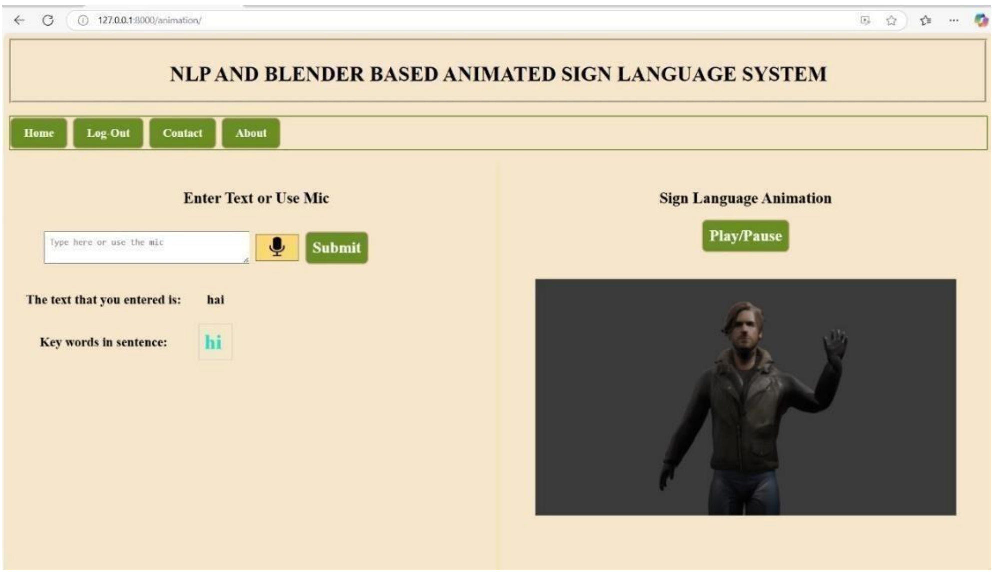
The text that you entered is: thank you

Key words in sentence: thank you

6.1.4 Test case - 03

6.4 Synonym Handling

The system maps synonyms to their appropriate ISL animations. When a user types **"Hai"**, it is recognized as a synonym for **"Hi"**, and the corresponding ISL animation is played.



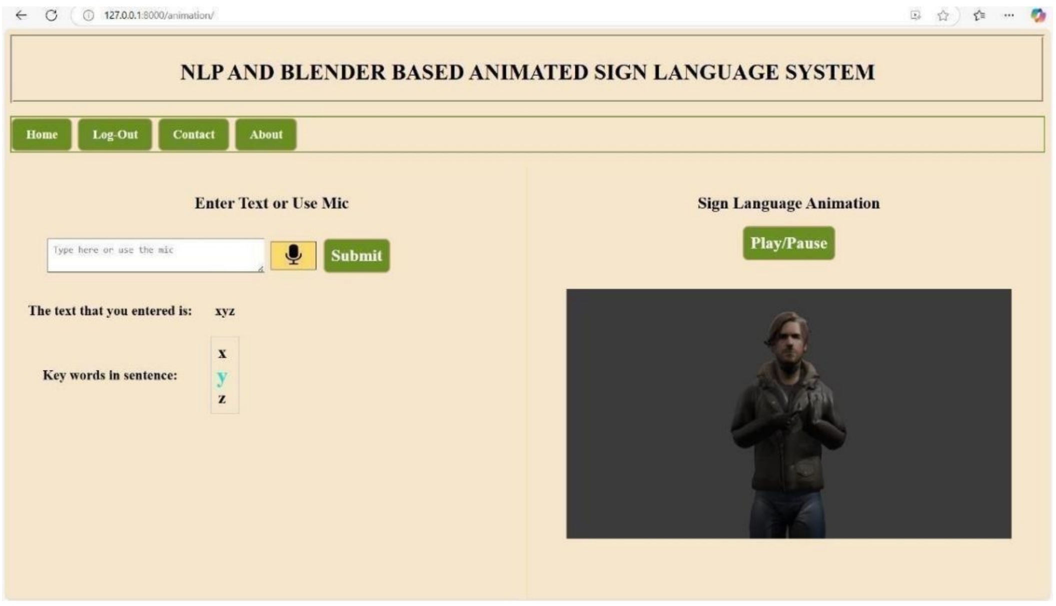
The text that you entered is: hai

Key words in sentence: hi

6.1.5 Test case - 04

6.5 Finger-Spelling Handling

For words that do not have predefined ISL signs, the system performs letter-by-letter animation using finger-spelling. When the user enters **"XYZ"**, the system displays an animation that spells out each letter individually.



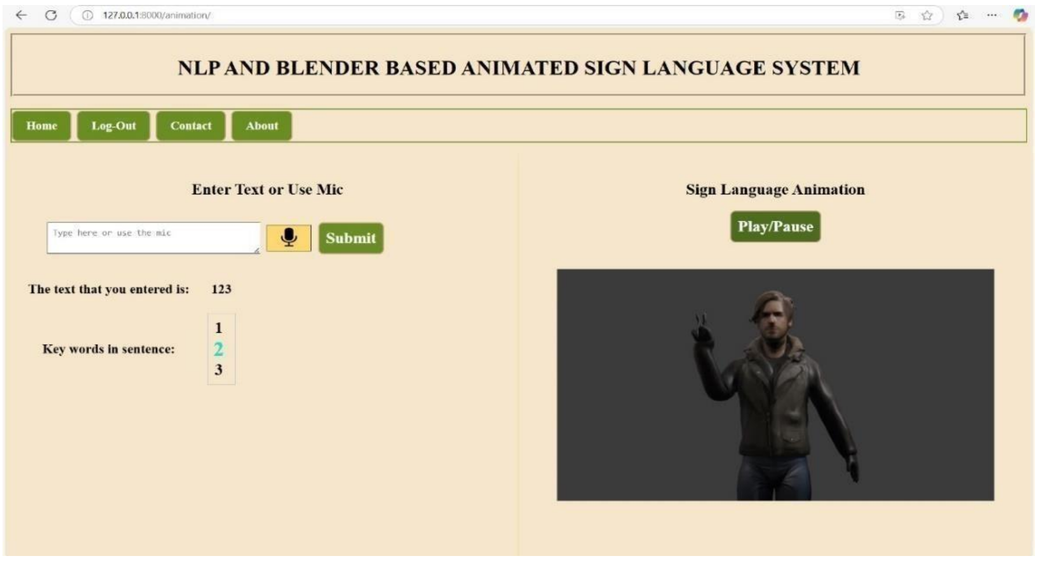
The text that you entered is: xyz

Key words in sentence: x y z

6.1.6 Test case – 05

6.6 Numeric Input Handling

The system correctly interprets numbers and maps them to ISL animations. When the user enters **"123"**, the corresponding number signs are displayed.



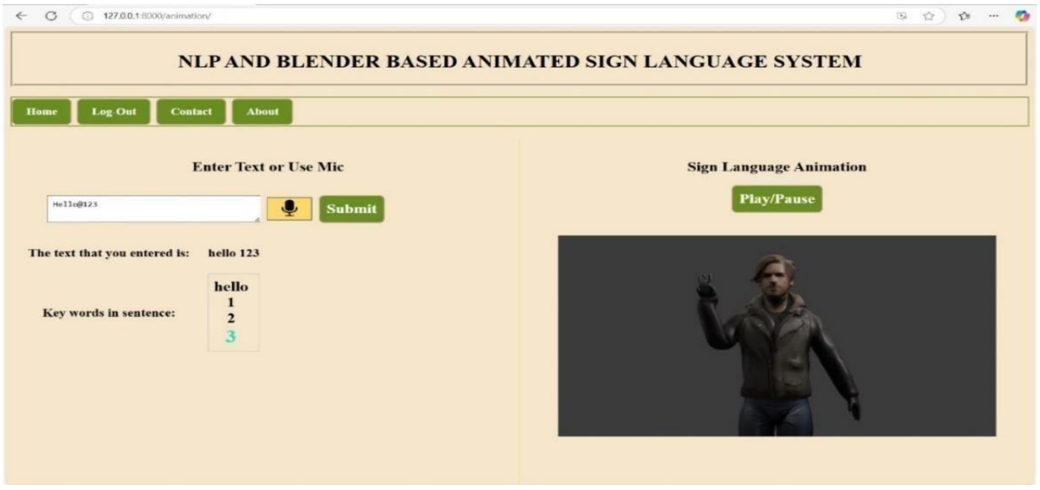
The text that you entered is: 123

Key words in sentence: 1 2 3

6.1.7 Test case - 06

6.7 Special Character Filtering

This test ensures that special characters are ignored, and only valid ISL animations are displayed. For instance, the input **"Hello@123"** filters out **"@"** and plays animations for **"Hello"** and **"123"** only.



The text that you entered is: hello 123

Key words in sentence: hello 1 2 3

6.1.8 Test case – 07

All test cases passed successfully, demonstrating the system's robustness in handling various inputs. The results confirm that the system effectively converts both text and speech into ISL

7. CONCLUSION

The **NLP and Blender-Based Sign Language Animation System** successfully converts text and speech into **Indian Sign Language (ISL) animations**, making communication more accessible for the hearing-impaired community. By integrating **Natural Language Processing (NLP) with Blender-based 3D animations**, the system provides an efficient and interactive way to interpret spoken and written language into sign language gestures.

The project was tested for various scenarios, including **text-to-sign conversion, speech recognition, synonym handling, and special character filtering**. The results confirmed the system's ability to accurately translate common words and phrases into ISL animations. Even in cases where predefined signs were unavailable, the system efficiently handled **fingerspelling representation**, ensuring that communication remained uninterrupted.

This project lays the foundation for a **practical and scalable ISL translation system**. However, future improvements can focus on **expanding the dataset by adding more animations** to cover a broader vocabulary. A well-structured and extensive dataset will improve the system's accuracy and fluency, making it more useful for real-world applications. Additionally, refining animation quality and optimizing system performance can further enhance user experience.

With continued development, this system has the potential to become a valuable tool for **education, accessibility, and inclusive communication**, helping bridge the gap between the hearing and hearing-impaired communities.

8. BIBLIOGRAPHY

- **Research Papers & Articles**
  - "Overview of Sign Language Translation Based on Natural Language Processing" (ResearchGate)
  - "Including Signed Languages in Natural Language Processing" (ACL Anthology)
- **Blender Animation Tutorials**
  - "Sign Language Animations in Blender" (YouTube)
  - "Blender 3D Easy Hand Tutorial - Basic Rigging and Animating" (YouTube)
- **Datasets & Open Source Projects**
  - "Sign Language Processing" (Research Sign)
  - "3D Animation Framework for Sign Language" (ResearchGate)
- **Software & Libraries**
  - Blender – Open-source 3D creation suite
  - NLTK – Natural Language Toolkit for NLP tasks
  - Web Speech API (WebkitSpeechRecognition) – Browser-based speech-to-text API

9. APPENDIX

Natural Language Processing (NLP)

Natural Language Processing (NLP) is a fundamental component in the NLP and Blender-Based Sign Language Animation System, responsible for converting text-based inputs into meaningful Indian Sign Language (ISL) animations. NLP techniques ensure that user input, whether provided through text or speech, is accurately processed and mapped to corresponding sign language gestures. This involves a series of language processing steps, such as tokenization, lemmatization, stopword removal, and synonym mapping, all of which work together to refine input data before animation retrieval.

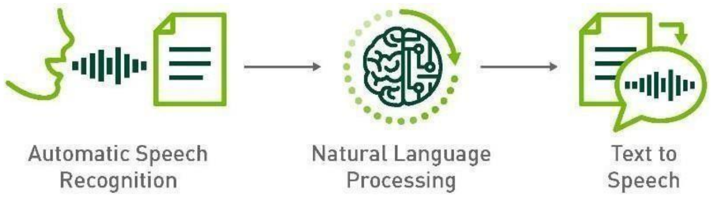
The system first processes user input by removing unnecessary punctuation and special characters, ensuring that only relevant words are considered. Tokenization breaks the input into individual words, which are then analysed to determine their root forms through lemmatization. This helps standardize word variations, making it easier to match words with existing ISL animations. Stopword removal eliminates commonly used words such as "is," "are," and "the," which do not contribute significantly to the sign language gesture. Additionally, the system checks for synonyms in a predefined dictionary to replace words that may not have a direct ISL animation.

For example, if a user inputs "Goodbye, see you later," the system processes the text by removing punctuation, resulting in "Goodbye see you later." The tokenization step then splits it into ["Goodbye," "see," "you," "later"]. Common stopwords like "you" are removed, leaving ["Goodbye," "see," "later"]. If "later" is not available in the animation dataset, the system searches the synonym dictionary and may replace it with a known ISL equivalent. The system then retrieves animations for "Goodbye" and "see," and if "later" has no direct animation, it either maps to a synonym or falls back to finger spelling.

The system integrates NLP with WebkitSpeechRecognition for speech-to-text conversion, ensuring that spoken words can be processed using the same NLP pipeline. When a user speaks into the system, their speech is transcribed into text, which is then analysed, processed, and mapped to ISL animations. The system also utilizes a JSON-based synonym database to enhance word-matching accuracy, reducing the chances of missing animations due to vocabulary variations.

Once the text is fully processed, the system searches for corresponding ISL animations. If an animation is found, it is retrieved and rendered using Blender’s animation engine. If a word lacks an exact animation, the system may use finger spelling, where the letters of the word are individually displayed using ISL gestures. This ensures that the system can handle a wide range of inputs, even when specific words are not present in the dataset.

NLP significantly enhances the system’s ability to understand and process language, making the translation of text and speech into sign language more effective. By leveraging NLP techniques such as synonym mapping, tokenization, and lemmatization, the system can accurately interpret a variety of inputs and provide meaningful sign language translations. Future enhancements could include expanding the synonym database, integrating machine learning models for contextual understanding, and improving animation mapping techniques to cover more complex sentence structures.



9.1.1 Natural Language Processing

Blender (version 4.3)

Blender plays a crucial role in the NLP and Blender-Based Sign Language Animation System, serving as the core animation tool for generating and displaying Indian Sign Language (ISL) gestures. It is an open-source 3D modelling and animation software used to create realistic sign language animations that enhance the accessibility and communication experience for users. The system utilizes Blender to animate hand gestures corresponding to ISL, ensuring accurate and fluid motion representation.

The animation process begins with the creation of 3D hand models in Blender, which are rigged using an armature system. Rigging involves adding a skeleton structure to the model, allowing for precise movement of fingers and hand gestures. Each gesture is designed in accordance with ISL standards to ensure accurate representation of words and letters. Keyframing is used to animate transitions between different hand positions, making the gestures smooth and natural.

To generate animations dynamically, the system maps processed text or speech inputs to predefined Blender animations. When a user inputs a word that has a corresponding ISL gesture, the system fetches the animation file from the database and plays it in the web interface. If a word lacks a direct ISL animation, the system falls back to finger spelling, where individual letter animations are displayed sequentially. This ensures that the system can handle a broad vocabulary, even if certain words do not have predefined animations.

The animation files are stored in a structured format, with each word or letter mapped to a specific Blender animation file. These files are rendered in real-time and integrated into the web application, allowing users to see the animated sign language output corresponding to their input. The rendering process ensures high-quality visualization while optimizing performance for smooth playback.

Blender’s rendering capabilities allow for realistic shading, lighting, and movement, enhancing the clarity and effectiveness of ISL animations. The animations are exported as video files or real-time rendered sequences that can be accessed through the Django web application. By leveraging Blender’s powerful animation tools, the system ensures that sign language gestures are not only accurate but also visually engaging.

Future improvements could include expanding the animation dataset, refining motion capture techniques for more lifelike gestures, and integrating artificial intelligence to automate animation generation for new words. By continuing to enhance the Blender-based animation pipeline, the system can provide a more comprehensive and natural sign language translation experience.

9.1.2 Blender 4.3

Django Framework:

Django serves as the backbone of the NLP and Blender-Based Sign Language Animation System, managing the backend logic, handling user interactions, and ensuring seamless communication between different components of the system. It is a high-level Python web framework that facilitates the development of scalable and maintainable applications. Django's Model-ViewController (MVC) architecture, robust security features, and built-in database management make it an ideal choice for structuring the project.

In this system, Django is responsible for processing user inputs, managing user authentication, and facilitating data flow between the frontend interface, NLP module, and Blender animation renderer. When a user provides text or speech input, Django handles the request, applies natural language processing techniques to extract key words, and retrieves the corresponding Indian Sign Language (ISL) animations from the dataset.

Django's views and templates are used to render web pages dynamically. The views.py file processes user requests and interacts with the NLP module to analyse input text. The templates directory contains HTML pages that structure the user interface, displaying animations and processed outputs in a user-friendly manner. Django's URL routing system allows different functionalities, such as home, login, signup, animation processing, and contact pages, to be accessed smoothly.

A key feature of Django in this project is user authentication and session management. The system allows users to sign up, log in, and securely access the animation platform. Django's builtin authentication mechanisms ensure that user credentials are securely stored and managed.

Django also integrates with SQLite, a lightweight database used in this project primarily for storing user-related data. The database manages user profiles and login credentials, enabling a personalized experience. However, the ISL animations themselves are stored as static files rather than in the database.

Another important aspect of Django's role is managing API requests and handling errors. It ensures that invalid or unsupported text inputs do not cause disruptions in the animation process. Django logs error, handles exceptions, and provides meaningful feedback to users, improving system reliability.

Overall, Django serves as a powerful bridge between the user interface, text processing, and animation rendering, ensuring efficient workflow execution. Its modular design allows for easy scalability, making it possible to expand the system with additional features such as more animations, advanced NLP capabilities, or real-time user interaction enhancements in the future.

SQLite3

SQLite3 is a lightweight, self-contained, and serverless database engine that is widely used for small to medium-scale applications. It is an open-source relational database management system (RDBMS) that stores data in a single file, making it highly portable and easy to use. Unlike other database management systems like MySQL or PostgreSQL, SQLite does not require a separate server to run, making it ideal for applications that need a simple yet efficient way to manage data. In the NLP and Blender-Based Sign Language Animation System, SQLite3 plays a crucial role in storing and managing user data. Since the project involves user authentication, the database maintains records such as usernames, email addresses, and passwords. The Django framework provides built-in support for SQLite3, allowing seamless interaction between the backend and the database.

One of the key advantages of SQLite3 is its lightweight nature and ease of integration. It is wellsuited for applications that do not require high-concurrency database operations. Since SQLite3 stores data in a single .sqlite3 file, it eliminates the complexity of database configuration, making it highly accessible for projects like this one.

The database is primarily used for:

- User authentication (storing login credentials securely).
- Tracking user sessions to allow a smooth experience while using the web application.
- Logging user interactions for potential future enhancements.

Since this project does not involve large-scale data storage for animations (which are stored as files), SQLite3 is a perfect fit due to its simplicity and reliability. The DB Browser for SQLite tool can also be used to inspect the database, modify records, and execute SQL queries visually.

Overall, SQLite3 serves as an efficient and minimalistic database solution that complements the Django backend in ensuring smooth authentication and data storage for users of the NLP and Blender-Based Sign Language Animation System.

WebkitSpeechRecognition

Speech recognition plays a significant role in making applications more interactive and accessible. WebkitSpeechRecognition is a built-in browser API that allows real-time speech-to-text conversion without requiring additional external libraries. This technology is particularly useful in applications where voice commands or spoken input need to be processed efficiently.

In the NLP and Blender-Based Sign Language Animation System, WebkitSpeechRecognition is used to convert spoken words into text, allowing users to provide input through speech instead of typing. This feature enhances accessibility, especially for individuals who may find it easier to speak rather than type.

How WebkitSpeechRecognition Works in the Project:

1. The user clicks on the microphone button on the web interface.
2. The WebkitSpeechRecognition API starts recording the speech.
3. Once the user stops speaking, the API converts the speech into text and displays it in the input field.
4. The text is then processed using Natural Language Processing (NLP) to identify keywords and match them with available sign language animations.
5. If an exact match for the word is found in the predefined animation dataset, the corresponding Blender animation is played. Otherwise, finger-spelling is used to represent the word.

Advantages of Using WebkitSpeechRecognition:

- Real-time conversion: Converts speech to text instantly without noticeable delays.
- No external dependencies: It works directly in the browser without requiring third-party installations.
- Cross-platform support: Works on most modern web browsers that support speech recognition.
- Enhances accessibility: Enables users who may have difficulty typing to interact with the system using voice commands.

Since WebkitSpeechRecognition is a browser-dependent API, its availability and performance may vary across different browsers. Currently, it works best on Google Chrome and Chromium-based browsers. However, fallback options like manual text input are provided to ensure usability across different platforms.

By integrating WebkitSpeechRecognition into the system, this project enables a seamless, voiceenabled interaction, making it more intuitive and user-friendly for individuals looking to convert spoken language into Indian Sign Language (ISL) animations.

Web Development:

The NLP and Blender-Based Sign Language Animation System is implemented as a web-based application to ensure accessibility across different devices and platforms. The front-end of the system is designed using HTML, CSS, and JavaScript, which together provide a structured, visually appealing, and interactive user experience.

Role of HTML, CSS, and JavaScript in the Project

The web application consists of multiple interfaces where users can input text or speech, view animations, and interact with the system. Each of these technologies plays a crucial role in ensuring smooth functionality and responsiveness.

1. HTML (Hypertext Markup Language)
    - HTML serves as the skeleton of the application, defining the structure and layout of web pages.
    - It is used to create elements like text input areas, buttons, video players, and navigation menus.
    - The application includes multiple HTML pages, such as:
        - Home Page: Introduction to the project.
        - Login & Signup Pages: User authentication.
        - Animation Page: Displays the sign language animations. Contact Page: Provides a form for user feedback.
2. CSS (Cascading Style Sheets)

- CSS is responsible for the styling and layout of the application, enhancing the user experience with

  visually appealing designs. o It ensures the website is responsive, meaning it adapts to different screen sizes, such as desktops, tablets, and mobile phones.

- Key design elements include:
  - Color themes that provide a clean and professional look.
  - Animations and transitions to improve visual engagement.
  - Flexbox and Grid layouts for proper alignment of elements.

3. JavaScript

   - JavaScript enables dynamic interactions within the application, allowing users to engage with various features in real time.
   - It is used for:
     - Handling user input events such as clicks, text input, and voice recognition.
     - Controlling speech-to-text conversion using WebkitSpeechRecognition.
     - Managing the real-time playback of animations, ensuring smooth transition between sign gestures.
     - Validating user inputs on login and signup pages.
     - Handling errors, such as displaying messages when an animation for a word is not found.

How Web Technologies Work Together in the Project

- When a user enters text or speech input, JavaScript captures the data and processes it for sign language animation mapping.

- If an animation exists for the word, the system retrieves and plays the corresponding video.

- The front-end elements (HTML & CSS) ensure that the user interface is clear and easy to navigate, while JavaScript dynamically updates the displayed animations.

By integrating HTML, CSS, and JavaScript, this project delivers a seamless, responsive, and accessible user experience, ensuring efficient real-time sign language translation. The web-based approach makes the system lightweight and easy to use without requiring users to install additional software.

This project is a step towards making communication more inclusive and accessible.By integrating Natural Language Processing, Blender animations, and **WebkitSpeechRecognition**, we have built a system that bridges the gap between text, speech, and Indian Sign Language. While the dataset currently includes a limited number of animations, the system demonstrates the potential for **real-time, interactive sign language translation**.

This work is not just about technology but about fostering a world where communication barriers are minimized. The project provides a foundation that can be expanded with **a richer dataset, enhanced NLP techniques, and improved animation capabilities**. As technology advances, solutions like this can contribute significantly to accessibility, education, and interaction for individuals relying on sign language.

With continued improvements, this system can grow into a **more comprehensive and impactful communication tool**. The journey does not end here—rather, it marks the beginning of an evolving effort to create **more inclusive digital interactions**.

Appendix – A

Project Repository Details

Project Title: NLP AND BLENDER BASED ANIMATED SIGN LANGUAGE SYSTEM

Batch: B-5

Batch Members:

1. M N S V Devi Parvathi - 21B01A05A7
2. Mathi Sai Divya Sri – 21B01A05A1
3. Neelapala Manasa- 21B01A05B8
4. Kodali Sravani - 21B01A0575
5. Koppada Naga Lakshmi – 22B05A0507

Department: Computer Science and Engineering

Institution: Shri Vishnu Engineering College for Women

Guide: **P. Sunil (Assistant Professor)**

Submission Date: 11/04/2025

Project Repository Link The complete project files, including source code, documentation, additional resources, are available at the following GitHub repository.

GitHub Repository: https://github.com/NoothanaMogalathurthi/NLP-And-Blender-Based-Animated-Sign-Language-System

For quick access, scan the QR code below: