Real-Time Communication System Powered by AI for Specially Abled

1.Introduction

1.1Project Overview

The **Real-Time Communication System Powered by AI for Specially Abled** is designed to assist individuals with disabilities by using artificial intelligence to bridge communication gaps. This system focuses on real-time translation of different forms of communication, such as converting sign language into text or speech, speech into text, and text into speech. The use of AI technologies like image recognition, natural language processing, and speech recognition ensures that communication is seamless and accessible, empowering users to interact more effectively in social, educational, or professional settings.

By leveraging cutting-edge AI, the system aims to improve the quality of life for specially-abled individuals by enabling independent communication. Whether it's translating sign language for those with hearing impairments or providing text-to-speech for individuals with speech disabilities, the platform is tailored to meet various accessibility needs. With real-time processing and customizable features, the system promotes inclusivity and supports diverse communication requirements, making it a valuable tool for a more accessible and connected world.

1.2 Project Objective

The objective is to develop a communication platform that utilizes artificial intelligence to enable seamless, real-time interaction for individuals with disabilities. The system aims to break down communication barriers for people with hearing, speech, or visual impairments by converting various forms of communication, such as sign language, speech, and text, into accessible formats. By leveraging AI technologies like speech-to-text, text-to-speech, and sign language recognition, the system ensures that specially-abled individuals can engage in everyday conversations and activities more independently and confidently.

This project seeks to empower specially-abled individuals by providing them with a tool that facilitates real-time, efficient communication, fostering inclusivity and enhancing social participation. The system aims to promote accessibility and autonomy, enabling users to interact in diverse environments such as educational, professional, and social settings, while improving their overall quality of life.

2. Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently

executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

Activity 1: Define Problem Statement

Problem Statement: A specially-abled individual faces significant challenges in communication due to disabilities such as hearing, speech, or visual impairments. Traditional communication methods often fail to meet their needs, leading to feelings of isolation and dependency. The challenge lies in creating a real-time communication system that accurately translates various forms of communication (e.g., sign language, speech, and text) into accessible formats. Despite advancements in assistive technologies, uncertainty remains about how to effectively combine AI models to deliver seamless, real-time translations that cater to diverse communication needs across different disabilities.

Ref. template: Click Here
Blueberry yield prediction Problem Statement Report: Click Here

Activity 2: Project Proposal (Proposed Solution)

The proposed project aims to leverage advanced AI and machine learning techniques to develop a real-time communication system that addresses the needs of specially-abled individuals. By utilizing a combination of speech-to-text, text-to-speech, and sign language recognition, the system will accurately translate various communication forms into accessible formats. The project will analyze diverse data inputs, such as voice patterns, gestures, and text, to create a predictive model that provides real-time, seamless translations. By deploying machine learning algorithms and natural language processing, the system will enhance communication efficiency, ensuring inclusivity and accessibility for users across different disabilities. The system will be designed to adapt to various environments, from social to professional settings, ensuring that specially-abled individuals can engage independently and effectively.

Ref. template: Click Here
Blueberry yield prediction Project Proposal Report: Click Here

Activity 3: Initial Project Planning

Initial Project Planning involves outlining key objectives, defining scope, and identifying the yield prediction. It encompasses setting timelines, allocating resources, and determining the overall project strategy. During this phase, the team establishes a clear understanding of the dataset, formulates goals for analysis, and plans the workflow for data processing. Effective initial planning lays the foundation for a systematic and well-executed project, ensuring successful outcomes.

Ref. template: Click Here
Blueberry yield prediction Initial Project Planning Report: Click Here

3. Data Collection and Preprocessing Phase

The Data Collection and Preprocessing Phase involves executing a plan to gather relevant data from Kaggle, ensuring data quality through verification and addressing missing values. Preprocessing tasks include cleaning, encoding, and organizing the dataset for subsequent exploratory analysis and machine learning model development.

Activity 1: Data Collection Plan, Raw Data Sources Identified

The dataset for the **Real-Time Communication System Powered by AI for Specially Abled** is sourced from publicly available datasets on platforms like Kaggle, along with additional datasets focused on sign language recognition, speech recognition, and text-to-speech conversions. These datasets include image data for sign language gestures, The goal is to gather diverse data types, covering a wide range of disabilities, to build a comprehensive system that can accurately handle various forms of communication.

Ref. template: Click Here
Blueberry yield prediction Raw Data Sources Report: Click Here

Activity 2: Data Quality Report

This includes addressing missing values, correcting data inconsistencies, and handling outliers, especially in image and audio data. For example, images or videos with poor resolution or incomplete gestures will be filtered out, and audio samples with background noise will be pre-processed to ensure clearer data for model training.

Ref. template: Click Here
Blueberry yield prediction Data Quality Report: Click Here

Activity 3: Data Exploration and Preprocessing

Involve examining the dataset for patterns in gestures, speech, and text. Key steps will include analyzing the distribution of gesture images, identifying patterns in speech-to-text conversions, and detecting any correlations between input methods (e.g., sign language gestures and text translations). Preprocessing will involve tasks such as resizing or augmenting images for gesture recognition, normalizing audio for speech recognition, and encoding textual data for natural language processing.

Ref. template: Click Here
Blueberry yield prediction Data Exploration and Preprocessing Report: Click Here

4. Model Development Phase

The Model Development Phase entails crafting a predictive model for loan approval. It encompasses strategic feature selection, evaluating and selecting models initiating training with code, and rigorously validating and assessing model performance for informed decision-making in the lending process.

Activity 1: Initial Model Training Code, Model Validation and Evaluation Report

The **Initial Model Training Code** for the **Real-Time Communication System Powered by AI for Specially Abled** will focus on training a **Convolutional Neural Network (CNN)** for image-based tasks, specifically for **sign language recognition**. The code will implement a CNN model that takes in images of sign language gestures as input, processes them through multiple convolutional layers, and outputs the corresponding sign language text or translation.The model will be trained on a labeled dataset containing images of sign language gestures. Common frameworks like **TensorFlow** or **PyTorch** will be used to implement the CNN model. Hyperparameters like learning rate, number of layers, and batch size will be tuned to optimize the model's performance.

Ref. template: Click Here

Blueberry yield Model Development Phase Template: CLick Here

Activity 2:Model Selection Report

The **Model Selection Report** for the **Real-Time Communication System Powered by AI for Specially Abled** focuses on the rationale behind choosing **Convolutional Neural Networks (CNN)** for image-based tasks like **sign language recognition**. This choice is driven by the nature of the problem, the advantages CNNs offer in image classification, and their ability to handle complex patterns in visual data.

Ref. template: Click Here
Blueberry yield Model Selection Report :Click Here

5. Model Optimization and Tuning Phase

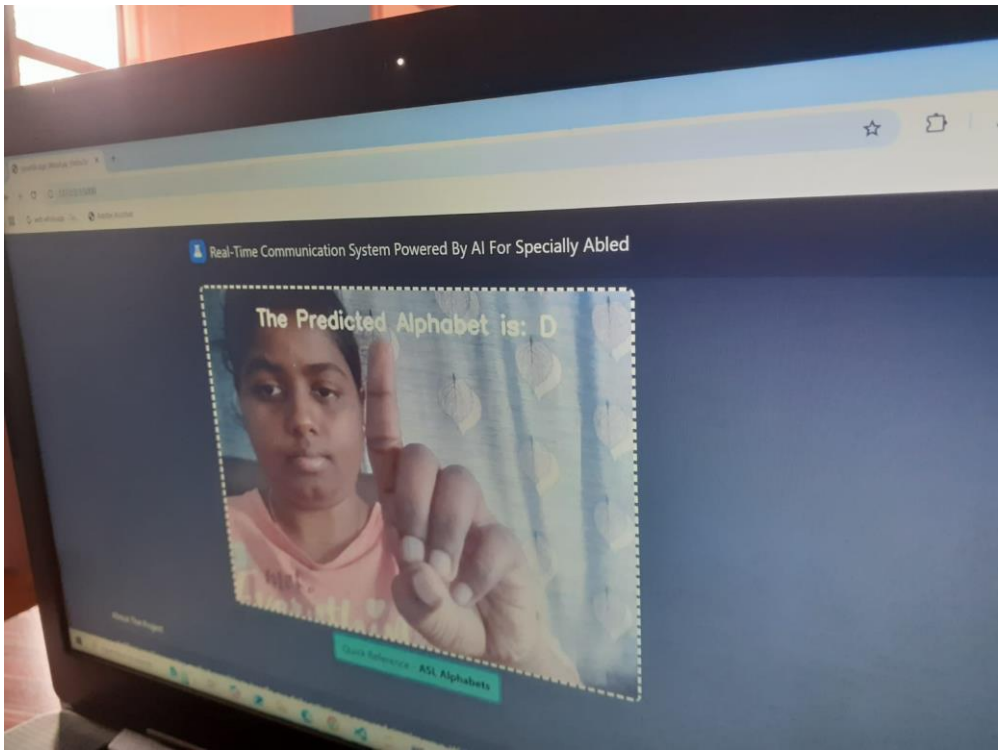
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

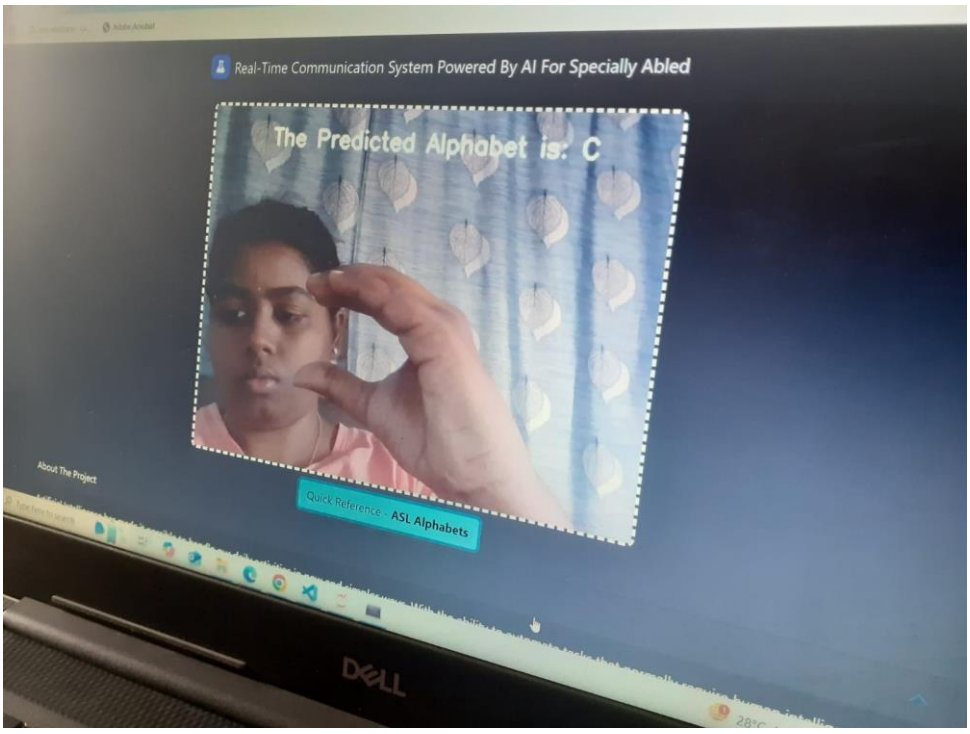
Activity 1: Hyperparameter Tuning Documentation

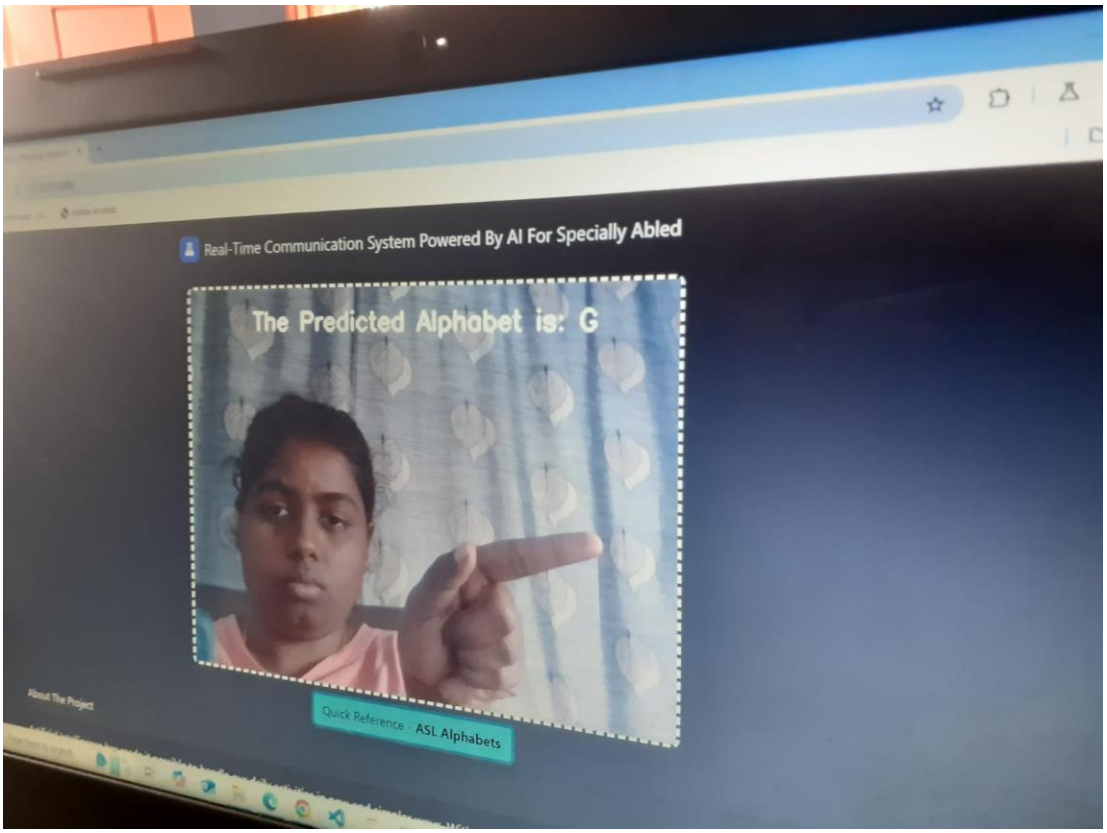
The **Convolutional Neural Network (CNN)** was fine-tuned through a rigorous hyperparameter optimization process. The selected hyperparameters, including learning rate, batch size, and the number of epochs, contributed significantly to the model's superior performance in recognizing sign language gestures. The final CNN model was chosen for its high accuracy, low loss, and strong generalization capabilities, making it the optimal choice for real-time communication systems for specially abled individuals.

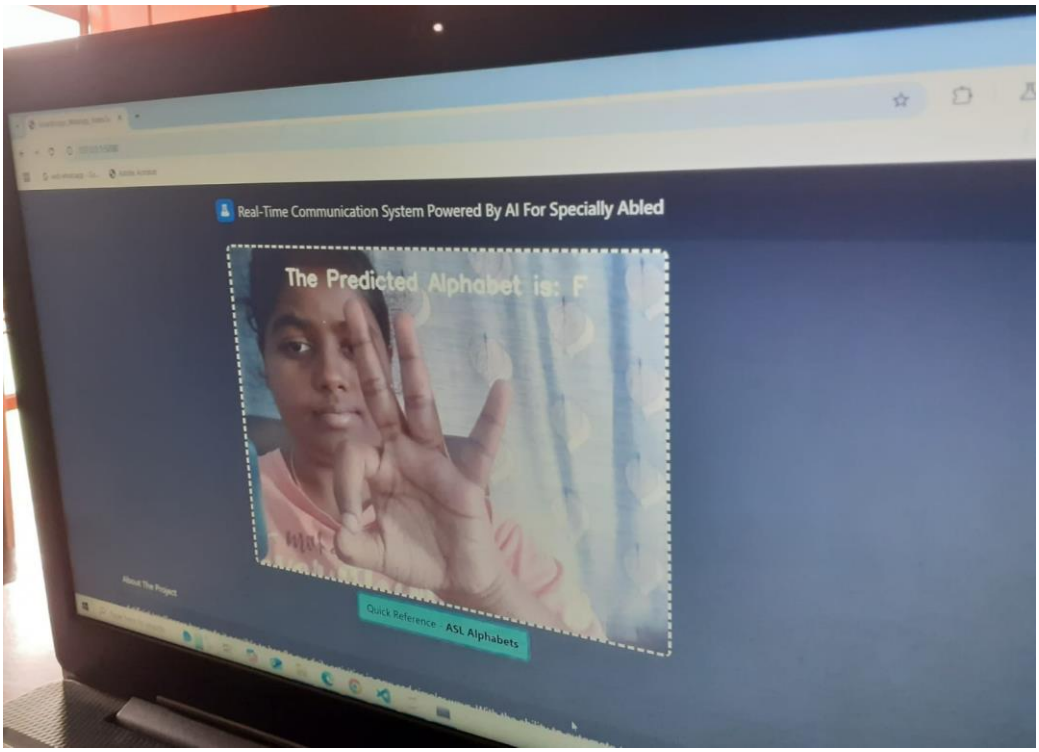
Ref. template: Click Here
Blueberry yield Hyperparameter Tuning Report: Click Here

6.RESULT


Real-Time Communication System Powered By AI For Specially Abled
The Predicted Alphabet is: D
Quick Reference - ASL Alphabets


Real-Time Communication System Powered By AI For Specially Abled
The Predicted Alphabet is: C
About The Project
Quick Reference - ASL Alphabets
DELL


Real-Time Communication System Powered By AI For Specially Abled
The Predicted Alphabet is: G
About The Project
Quick Reference - ASL Alphabets


Real-Time Communication System Powered By AI For Specially Abled
The Predicted Alphabet is: F
About The Project
Quick Reference - ASL Alphabets

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Improved Communication for Specially Abled Individuals**: The real-time communication system powered by AI can bridge communication gaps for people with hearing, speech, or cognitive impairments, facilitating smoother interactions..
2. **Accessibility**: Enhances accessibility by enabling specially abled individuals to communicate more effectively using voice recognition, sign language recognition, or text-to-speech functionalities.
3. **Personalization**: AI-based systems can be tailored to the specific needs of the user, adapting to their preferences and unique communication styles.
4. **Real-Time Translation**: The system can provide instant translation or conversion of sign language into text or voice, improving the speed and effectiveness of communication.
5. **Independence for Specially Abled Individuals**: It empowers users by enhancing their ability to communicate independently in social and professional settings, reducing reliance on caregivers.

DISADVANTAGES:

1. **Data Dependency**: The system's effectiveness depends on accurate data, such as a well-trained model for speech or sign language recognition, which may not be readily available for all languages or dialects.
2. **Technical Complexity:** Developing and maintaining such systems requires high technical expertise in AI, machine learning, and real-time data processing, which can make implementation challenging.
3. **Privacy and Security Concerns**: Real-time communication systems often involve sensitive personal data, raising concerns about data privacy, security, and unauthorized access.
4. **Model and Recognition Accuracy**: The system’s performance might be inconsistent, especially in noisy environments or when dealing with uncommon speech or sign language variations, leading to potential communication breakdowns.

8.CONCLUSION

► In conclusion, the proposed Real-Time Communication System powered by AI for specially abled individuals offers a transformative solution to enhance communication accessibility. By integrating advanced AI technologies such as speech recognition, sign language interpretation, and real-time translation, the system empowers specially abled individuals to communicate more effectively and independently.

► The project's key components—data collection, model training, real-time processing, and deployment—form a solid foundation for a responsive and accurate communication system. Additionally, the inclusion of personalized features and adaptive learning ensures that the system meets the diverse needs of users, making it both practical and user-friendly.

► By enhancing communication for specially abled individuals, this system not only fosters inclusivity and independence but also contributes to societal well-being. Its impact extends beyond individual users, supporting broader social integration and providing innovative solutions to everyday communication challenges. The holistic approach of the Real-Time Communication System paves the way for a more accessible and inclusive future.

\

9.FUTURE SCOPE

1. **Integration with Wearable Devices**: Future systems could integrate with wearable devices like smart glasses or watches to provide seamless communication support for specially abled individuals in real-time, without the need for additional hardware.
2. **Enhanced Multi-Language and Multi-Dialect Support**: The system could expand its capabilities to support a wider variety of languages, dialects, and regional sign languages, making it more inclusive for global use.
3. **Emotion and Tone Recognition**: Future versions of the system could incorporate emotion and tone recognition to better understand the context of communication, improving the system's ability to respond appropriately in sensitive conversations.
4. **AI-Driven Personalized Learning**: The system could adapt over time to better serve individual needs, learning from user interactions to provide more accurate translations and communication aids tailored to the user's style.
5. **Real-Time Translation for Group Communication**: Expanding the system to support real-time group communication with multiple participants, offering translation services across different communication methods (e.g., speech, text, and sign language) in a single conversation.
6. **Integration with Virtual Reality (VR)**: AI-powered communication tools could be integrated with VR platforms, creating immersive environments where specially abled individuals can communicate and interact naturally in virtual spaces.
7. **AI-Powered Assistive Chatbots**: Future advancements could allow the creation of virtual assistants or chatbots that assist specially abled individuals by providing on-demand help for various tasks, such as navigation, social interaction, or learning support.
8. **Collaboration with Smart Home Systems**: The system could be integrated with smart home devices, allowing specially abled individuals to control their environment using AI-driven communication systems for greater independence and ease of living.
9. **Global Awareness and Advocacy**: By integrating with social media platforms and online communities, the system could promote awareness, advocacy, and education on issues faced by specially abled individuals, fostering a global conversation on accessibility.
10. **Improved Speech Recognition and Accessibility for Diverse Disabilities**: Continuous improvements in speech recognition and AI modeling could address a wider range of disabilities, such as cognitive impairments, enhancing the system’s effectiveness for users with different needs.

10.APPENDIX

10.1 SOURCE CODE:

Index.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
   <title>SmartBridge_WebApp_VideoTemplate</title>
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
   <link rel="stylesheet" href="{{ url_for('static', filename='assets/css/index.css') }}">

</head>

<body style="background: rgb(39,43,48);">
   <nav class="navbar navbar-light navbar-expand-md py-3" style="background: #212529;">
      <div class="container">
         <div></div><a class="navbar-brand d-flex align-items-center" href="#"><span
               class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-content-center align-
items-center me-2 bs-icon"><i
                  class="fas fa-flask"></i></span><span style="color: rgb(255,255,255);">Real-Time
Communication
               System Powered By AI&amp;nbsp;For Specially Abled</span></a>
         <div></div>
      </div>
   </nav>
   <section>
      <div class="d-flex flex-column justify-content-center align-items-center">
         <div class="d-flex flex-column justify-content-center align-items-center" id="div-video-feed"
            style="width: 640px;height: 480px;margin: 10px;min-height: 480px;min-width:
640px;border-radius: 10px;border: 4px dashed rgb(255,255,255) ;">
            <img src="{{ url_for('video_feed') }}" style="width: 100%;height: 100%;color:
rgb(255,255,255);text-align: center;font-size: 20px;"
    alt="Camera Access Not Provided!">

         </div>
      </div>
      <div class="d-flex flex-column justify-content-center align-items-center" style="margin-bottom:
10px;"><button
            class="btn btn-info" type="button" data-bs-target="#modal-1" data-bs-
toggle="modal">Quick Reference
```

```
            -<strong> ASL Alphabets</strong></button></div>
    </section>
    <section>
        <div class="container">
            <div class="accordion text-white" role="tablist" id="accordion-1">
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <h2 class="accordion-header" role="tab"><button class="accordion-button" data-bs-
toggle="collapse"
                        data-bs-target="#accordion-1 .item-1" aria-expanded="true"
                        aria-controls="accordion-1 .item-1"
                        style="background: rgb(39,43,48);color: rgb(255,255,255);">About The
Project</button></h2>
                <div class="accordion-collapse collapse show item-1" role="tabpanel" data-bs-
parent="#accordion-1">
                    <div class="accordion-body">
                        <p class="mb-0">Artificial Intelligence has made it possible to handle our daily
activities
                            in new and simpler ways. With the ability to automate tasks that normally require
human
                            intelligence, such as speech and voice recognition, visual perception, predictive
text
                            functionality, decision-making, and a variety of other tasks, AI can assist people
with
                            disabilities by significantly improving their ability to get around and participate in
                            daily activities.<br><br>Currently, Sign Recognition is available <strong>only
for
                                alphabets A-I</strong> and not for J-Z, since J-Z alphabets also require
Gesture
                            Recognition for them to be able to be predicted correctly to a certain degree of
                            accuracy.</p>
                    </div>
                </div>
            </div>
            <div class="accordion-item" style="background: rgb(33,37,41);">
                <h2 class="accordion-header" role="tab"><button class="accordion-button collapsed"
                    data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-
expanded="false"
                    aria-controls="accordion-1 .item-2"
                    style="background: rgb(39,43,48);color: rgb(231,241,255);">Developed
By</button></h2>
                <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-
parent="#accordion-1">
                    <div class="accordion-body">
                        <p class="mb-0">Students at Vaagdevi Engineering College
                            <br><br>1. <strong>Tejaswi Kodithala</strong> 21UK1A0507<br>2.
                            <strong>Madhavi Tanniru</strong> 21UK1A0509<br>3. <strong>Sai Pravalika
Mugada</strong> 21UK1A0539
                            <br>4. <strong>Amulya Dadi</strong> 21UK1A0564
                        </p>
                    </div>
```

```
                </div>
            </div>
        </div>
    </div>
</section>
<div class="modal fade" role="dialog" tabindex="-1" id="modal-1">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">American Sign Language - Alphabets</h4><button
type="button"
                    class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body"><img src="{{ url_for('static',
filename='assets/img/ASL_Alphabets.png') }}" width="100%"></div>
            <div class="modal-footer"><button class="btn btn-secondary" type="button"
                    data-bs-dismiss="modal">Close</button></div>
        </div>
    </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

App.py

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask, Response, render_template
import time
import threading

class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        if not self.video.isOpened():
            print("Error: Unable to access the camera.")
            exit()
        self.model = load_model('SLDmodel.h5')  # Load your trained model
        self.index = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']  # List of characters for prediction
        self.y = None
        self.last_prediction_time = time.time()

    def __del__(self):
        self.video.release()

    def get_frame(self):
        ret, frame = self.video.read()
        if not ret:
            print("Failed to grab frame")
            return None
        frame = cv2.resize(frame, (640, 480))  # Resize the frame
        copy = frame.copy()  # Define Region of Interest (ROI) for sign language
        copy = copy[150:150+200, 50:50+200]  # Save the ROI as an image and make predictions

        current_time = time.time()
        if current_time - self.last_prediction_time >= 1.0:  # 1 second delay
            cv2.imwrite('image.jpg', copy)
            copy_img = image.load_img('image.jpg', target_size=(64, 64))  # Resize image to match
model input
            x = image.img_to_array(copy_img)
            x = np.expand_dims(x, axis=0)
            pred = np.argmax(self.model.predict(x), axis=1)
            self.y = pred[0]
            self.last_prediction_time = current_time

        cv2.putText(frame, f'The Predicted Alphabet is: {self.index[self.y] if self.y is not None else ""}',
(100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 3)
```

```
        ret, jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        if frame is None:
            print("No frame captured.")
            break
        yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    video = Video()
    return Response(gen(video), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(debug=True)
```

CODE SNIPPETS

DATA COLLECTION

Importing necessary libraries

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D,Dense, Dropout, Flatten
from keras.preprocessing import image

from keras.models import load_model
import numpy as np
import cv2
from keras.optimizers import Adam
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

Configure the ImageDataGenerator for the training & testing set with augmentations

```
train_datagen = ImageDataGenerator(
    rescale=1./255,         # Rescale pixel values (0-255) to (0-1)
    shear_range=0.2,         # Apply shear transformation
    zoom_range=0.2,          # Apply zoom transformation
    horizontal_flip=True,    # Randomly flip images horizontally
    vertical_flip=False
)
test_datagen = ImageDataGenerator(rescale=1./255)
```

DATA PREPROCESSING

```
train_generator = train_datagen.flow_from_directory(
    'Dataset/training_set',
    target_size=(64,64),
    batch_size=128,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    'Dataset/test_set',
    target_size=(64,64),
    batch_size=128,
    class_mode='categorical'
)
```

```
Found 15750 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

```
print("Len x-train : ", len(train_generator))
print("Len x-test : ", len(test_generator))
```

```
Len x-train :  124
Len x-test :  18
```

```
# The Class Indices in Training Dataset
train_generator.class_indices
```

```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Initializing and adding layers

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(9, activation='softmax'))
```

compling the model

```
optimizer = Adam(learning_rate=0.0001)  # Lower learning rate
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

visualization

```
model.summary()
```

Model: "sequential_12"

Layer (type)	Output Shape	Param #
conv2d_33 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_33 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_34 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_34 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_35 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_35 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten_9 (Flatten)	(None, 4608)	0
dense_15 (Dense)	(None, 128)	589,952
dense_16 (Dense)	(None, 9)	1,161

Total params: 684,361 (2.61 MB)

Trainable params: 684,361 (2.61 MB)

Non-trainable params: 0 (0.00 B)

Training the Model

```
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=test_generator
)
```

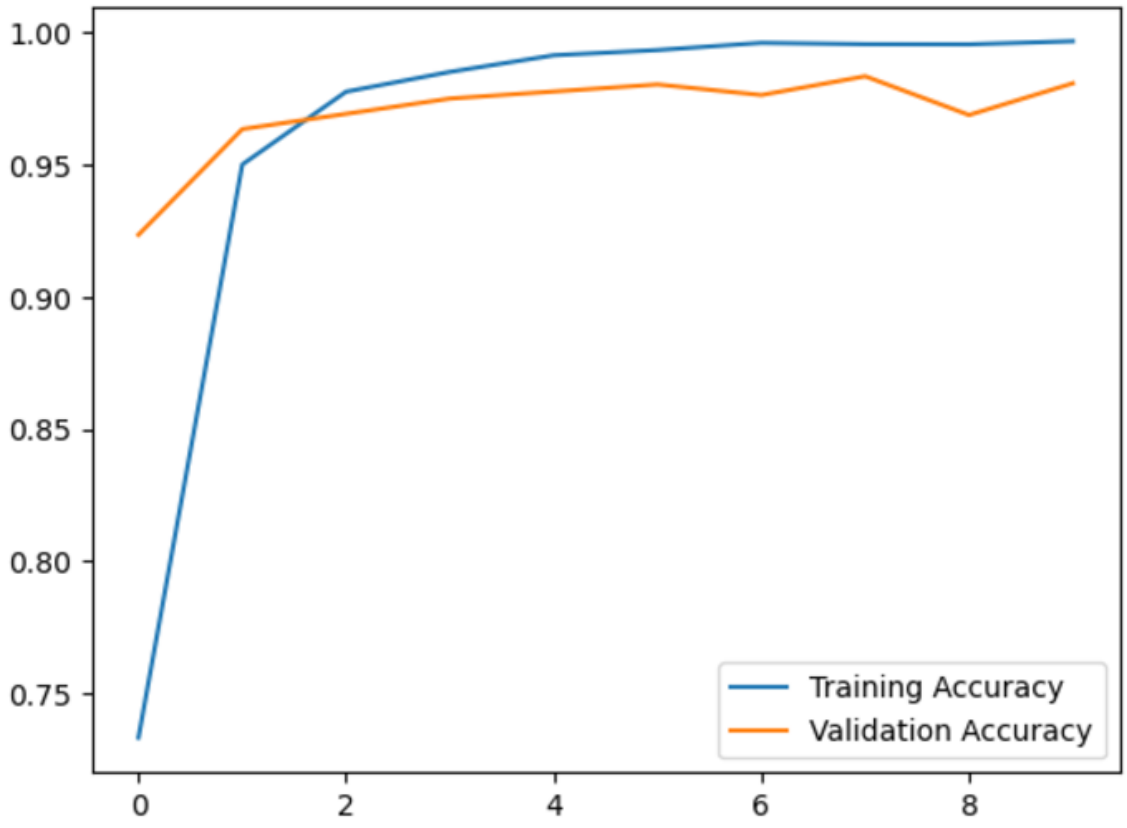
```
Epoch 1/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 89s 681ms/step - accuracy: 0.5330 - loss: 1.5709 - val_accuracy: 0.9236 - val_loss: 0.2684
Epoch 2/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 75s 600ms/step - accuracy: 0.9380 - loss: 0.2158 - val_accuracy: 0.9636 - val_loss: 0.1724
Epoch 3/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 73s 587ms/step - accuracy: 0.9727 - loss: 0.0966 - val_accuracy: 0.9693 - val_loss: 0.1197
Epoch 4/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 76s 608ms/step - accuracy: 0.9828 - loss: 0.0582 - val_accuracy: 0.9751 - val_loss: 0.1098
Epoch 5/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 72s 583ms/step - accuracy: 0.9909 - loss: 0.0348 - val_accuracy: 0.9778 - val_loss: 0.1084
Epoch 6/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 73s 590ms/step - accuracy: 0.9940 - loss: 0.0249 - val_accuracy: 0.9804 - val_loss: 0.0941
Epoch 7/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 72s 582ms/step - accuracy: 0.9955 - loss: 0.0173 - val_accuracy: 0.9764 - val_loss: 0.1248
Epoch 8/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 73s 590ms/step - accuracy: 0.9958 - loss: 0.0141 - val_accuracy: 0.9836 - val_loss: 0.0684
Epoch 9/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 79s 634ms/step - accuracy: 0.9960 - loss: 0.0149 - val_accuracy: 0.9689 - val_loss: 0.1356
Epoch 10/10
124/124 ━━━━━━━━━━━━━━━━━━━━ 74s 596ms/step - accuracy: 0.9945 - loss: 0.0160 - val_accuracy: 0.9809 - val_loss: 0.0887
```

Evaluting Training Accuracy

```
val_loss, val_accuracy = model.evaluate(train_generator)
print(f"Validation Loss: {val_loss}, Validation Accuracy: {val_accuracy}")
```

```
124/124 ━━━━━━━━━━━━━━━━━━━━ 74s 597ms/step - accuracy: 0.9986 - loss: 0.0066
Validation Loss: 0.0061272624880075455, Validation Accuracy: 0.9987936615943909
```

```
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.show()
```



saving the model

```
model.save("SLDmodel.h5")
```

loading the saved model

```
model=load_model("SLDmodel.h5")
```

Testing the model

```
from skimage.transform import resize
def detect(frame):
    img = image.img_to_array(frame)
    img = resize(img, (64, 64))
    img = np.expand_dims(img, axis=0)

    # Normalize if the pixel values are greater than 1 (for model compatibility)
    if np.max(img) > 1:
        img = img / 255.0

    # Make prediction
    pred = np.argmax(model.predict(img), axis=1)

    # Define the output labels
    op = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']

    # Print the predicted letter
    print("The predicted letter is ", op[pred[0]])
    print(f"Shape of a single image: {img.shape}")
```

```
frame = cv2.imread(r"C:\Users\VENU\Desktop\MAJOR PROJECT\RTCS(Github)\Dataset\test_set\G\1.png")
if frame is None:
    print("Error loading image!")
else:
    data = detect(frame)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step
The predicted letter is  G
Shape of a single image: (1, 64, 64, 3)
```

```
t3=cv2.imread(r"C:\Users\VENU\Desktop\MAJOR PROJECT\RTCS(Github)\Dataset\test_set\A\1.png")
data=detect(t3)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
The predicted letter is  A
Shape of a single image: (1, 64, 64, 3)
```

```
t4=cv2.imread(r"C:\Users\VENU\Desktop\MAJOR PROJECT\RTCS(Github)\Dataset\test_set\C\1.png")
data=detect(t4)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
The predicted letter is  C
Shape of a single image: (1, 64, 64, 3)
```

10.2 Github & project Demo Link:

Github link: https://github.com/TejaswiKodithala/Major-Project