Project Report

1.INTRODUCTION

Project Overview:

The project "ASL Alphabet Image Recognition" aimed to create a machine learning model proficient in precisely identifying hand signs representing the American Sign Language (ASL) alphabet. The overarching objective was to integrate this model into a real-time application, with the ultimate aim of fostering smooth communication between the deaf and hearing communities.

Purpose:

The ASL Alphabet Image Recognition project seeks to construct a machine learning model proficient in precisely identifying American Sign Language (ASL) alphabet hand signs. Through the training of a deep learning system to classify images corresponding to the 26 letters of the English alphabet, as well as symbols for "space," "delete," and "nothing," the project endeavors to narrow the communication divide between the deaf and hearing communities. The primary goal is to devise a real-time application that employs this model to interpret ASL hand signs from live video streams. Ultimately, this technological advancement aims to improve communication accessibility for those who use ASL as their primary language, empowering them to engage more effectively with the broader community. The project's report comprehensively documents the development process, methodologies employed, results obtained, and the potential impact of this innovation on enhancing communication accessibility and inclusivity for both deaf and hearing individuals.

2. LITERATURE SURVEY

Existing problem

Developing a robust ASL Alphabet Image Recognition system encounters significant challenges. The diversity in hand shapes, sizes, and orientations among individuals creates hurdles for precise gesture classification. Real-time processing constraints impede quick recognition in live applications. Sensitivity to fluctuating lighting conditions and backgrounds adversely affects recognition accuracy. Furthermore, ensuring model generalization to accommodate unseen hand gestures and variations poses persistent complexities in achieving dependable and accurate ASL recognition..

References

1. https://towardsdatascience.com/building-and-deploying-an-alphabet-recognition-system-7ab59654c676
2. https://arxiv.org/pdf/1103.0365
3. http://archive.ics.uci.edu/dataset/59/letter+recognition/

Problem Statements:

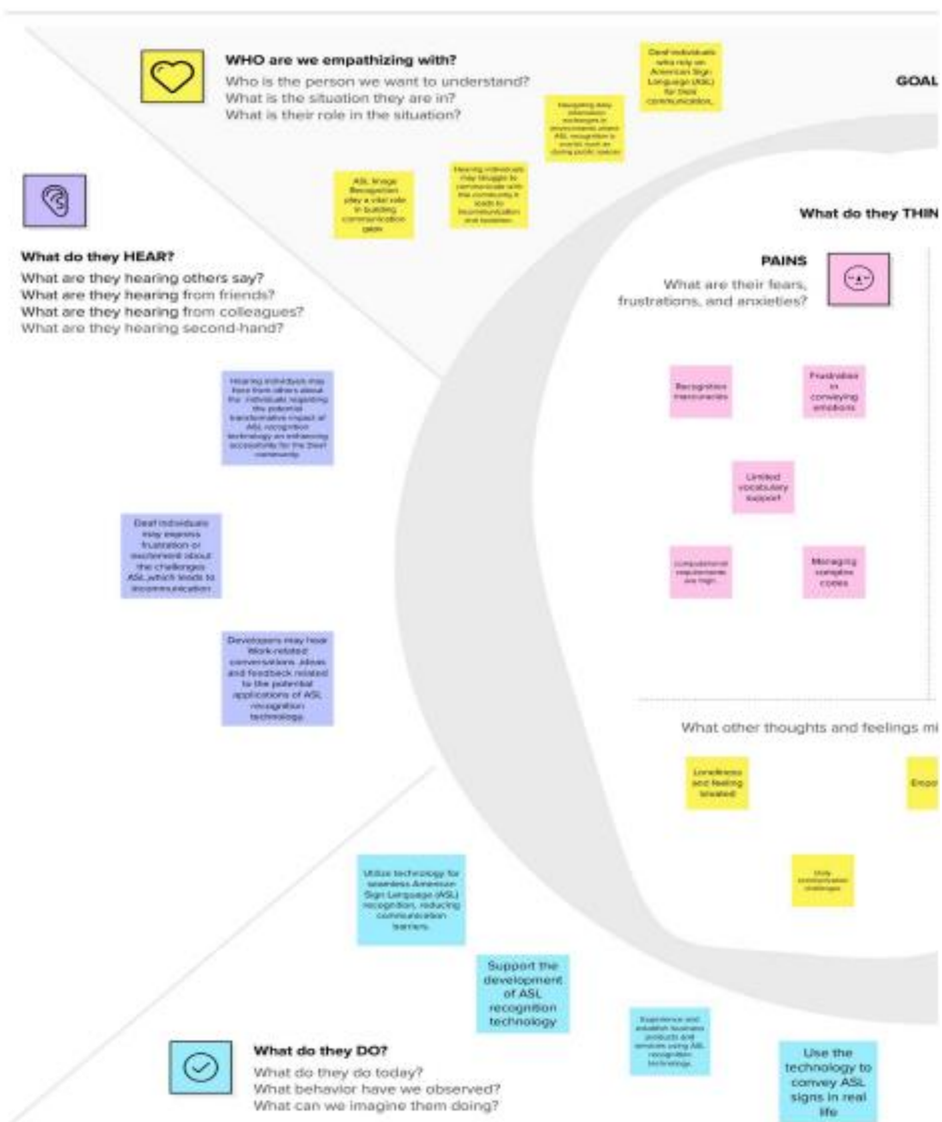
1. How Might We improve real-time translation of ASL gestures into written or spoken language, ensuring fast and accurate communication for users?

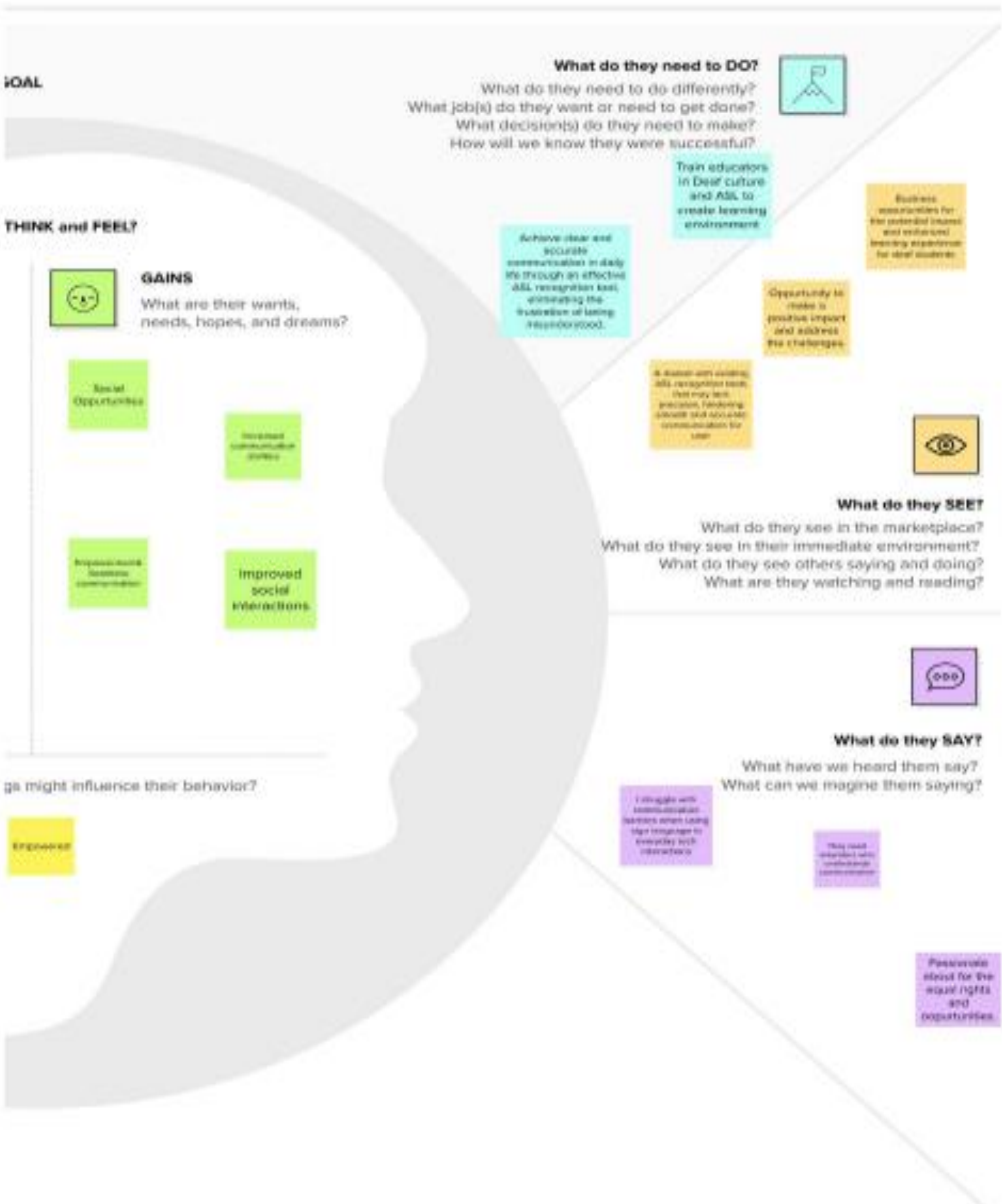
2.How Might We ensure user privacy and data security in ASL recognition systems,

allowing users to have control over their personal information?
3. How Might We enhance ASL recognition by incorporating accurate facial expression recognition and expressive communication?
4.How Might We make communication more accessible for the deaf and hard-of-hearing through an ASL recognition system ?

3.IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

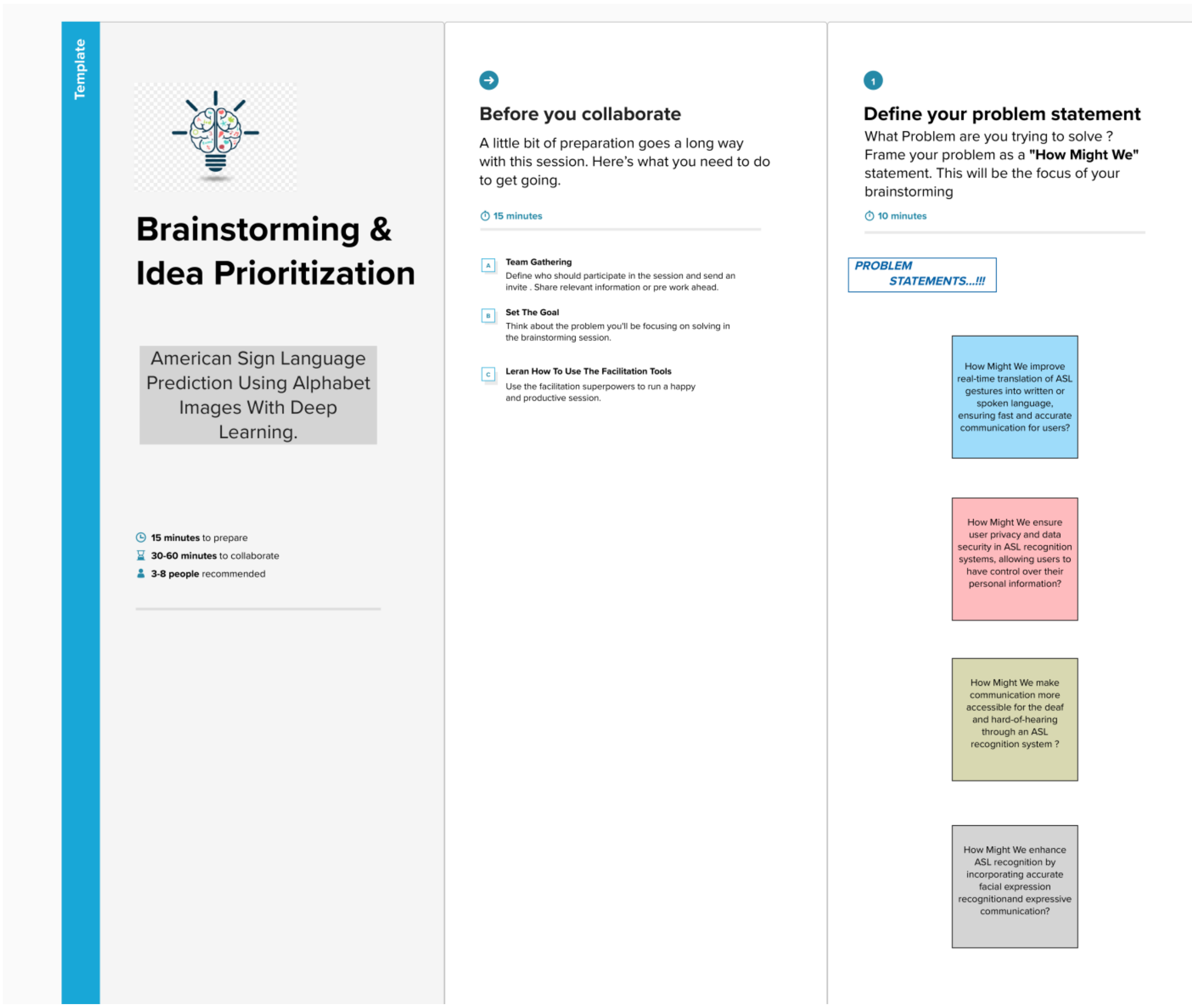




GOAL
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Train educators in Deaf culture and ASL to create learning environment
Achieve clear and accurate communication in daily life through an effective ASL recognition tool, eliminating the frustration of being misunderstood.
Opportunity to make a positive impact and address the challenges
THINK and FEEL?
GAINS
What are their wants, needs, hopes, and dreams?
Social Opportunities
Improved social interactions
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they SAY?
What have we heard them say?
What can we imagine them saying?
Passionate about for the equal rights and opportunities
gs might influence their behavior?
Empowered

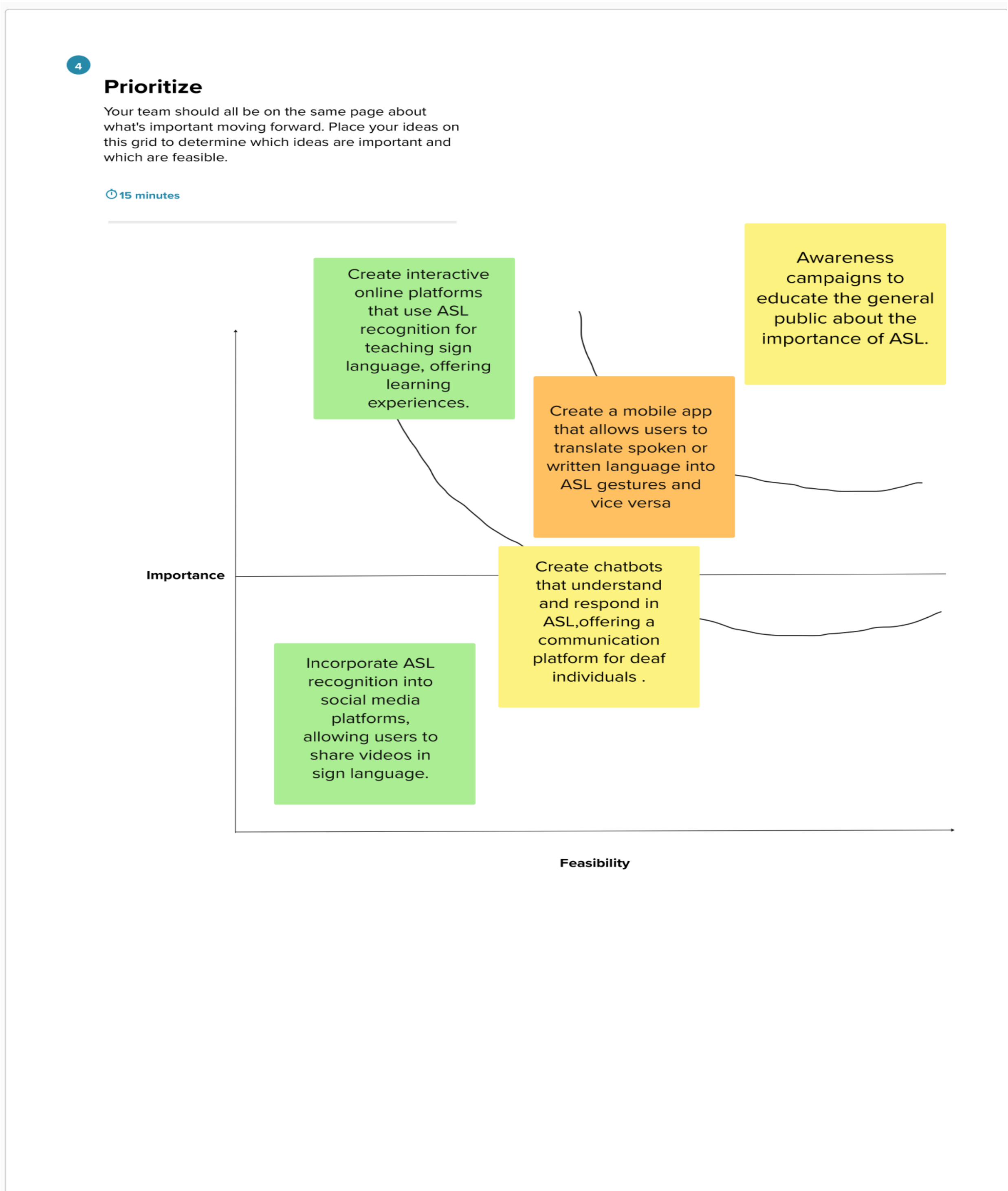
Ideation & Brainstorming

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritization



4. REQUIREMENT ANALYSIS

Functional requirement

1. Image Recognition:

The system is required to precisely identify and categorize ASL hand signs representing the 26 English alphabet letters. Additionally, it should accurately recognize symbols denoting "space," "delete," and "nothing."

2.Real-time Analysis:

The application needs to swiftly process a sequence of images, promptly identifying ASL hand signs and delivering immediate feedback.

3.User Interface:

Develop a user-friendly interface to facilitate seamless interaction with the recognition system, ensuring an easy and intuitive experience for users.

4.Diverse Hand Gestures:

The system must possess the capability to distinguish and categorize various hand gestures, accommodating differences in orientation and size for comprehensive recognition.

Non-Functional requirements

1.Precision:

The system is expected to attain a minimum accuracy threshold of 95% in effectively classifying ASL hand signs, ensuring dependable communication.

2.Real-time Efficiency:

The application must exhibit the capability to process video streams with a maximum latency of 100 milliseconds, ensuring prompt recognition.

3.Robustness:

The system should exhibit resilience, maintaining accuracy despite variations in lighting conditions, backgrounds, and hand orientations.

4.Scalability:

The application's architecture should support an expanding user base without compromising performance,ensuring scalability as user numbers increase.

5.Security Measures:

Implement stringent data privacy and security protocols to safeguard any stored or processed information related to users' interactions with the system.

6.User-Friendliness:

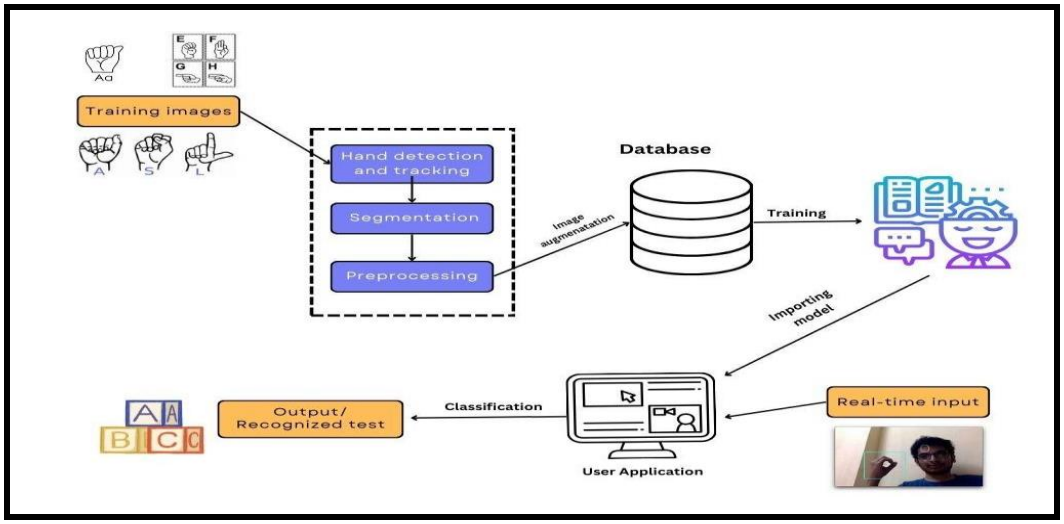
Design the application to be intuitive and accessible, accommodating users with diverse technical expertise or varying levels of familiarity with ASL.

5. PROJECT DESIGN

Data Flow Diagrams & User Stories

The project begins with inputting alphabet images, initiating essential steps in recognition. Initially, hand detection and segmentation accurately isolate hand from backgrounds for precise analysis. Image preprocessing enhances quality, preparing data for machine learning. Image augmentation introduces diversity in the dataset,systematically organized and stored for efficient retrieval during training. The core involves training a deep learning model, fine-tuning it for optimal recognition. A user-friendly web app allows image uploads, utilizing the trained model to interpret hand signs and provide corresponding text, promoting streamlined communication for the hearing-impaired. This comprehensive pipeline advances in recognition and fosters inclusivity by enhancing communication accessibility.

Dataflow Diagram -



User Stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Deaf of hard communities	Project setup and infrastructure	USN-1	Establish the development environment by installing the necessary tools and frameworks to initiate the alphabet image recognition system.	The setup is complete, incorporating all essential tools and frameworks.	High	Sprint-1
Deaf or Hard-of-Hearing Individuals	Devloping environment	USN-2	Collect a varied image dataset featuring diverse ASL alphabet signs to train the deep learning model effectively, ensuring it can accurately recognize and interpret a broad range of American Sign Language gestures for optimal performance.	Assembled a diverse image dataset illustrating different categories of ASL signs.	High	Sprint-1
Normal (Hearing)	Data Collection	USN-3	Prepare the acquired dataset through resizing images,	Prepared the dataset.	High	Sprint-2

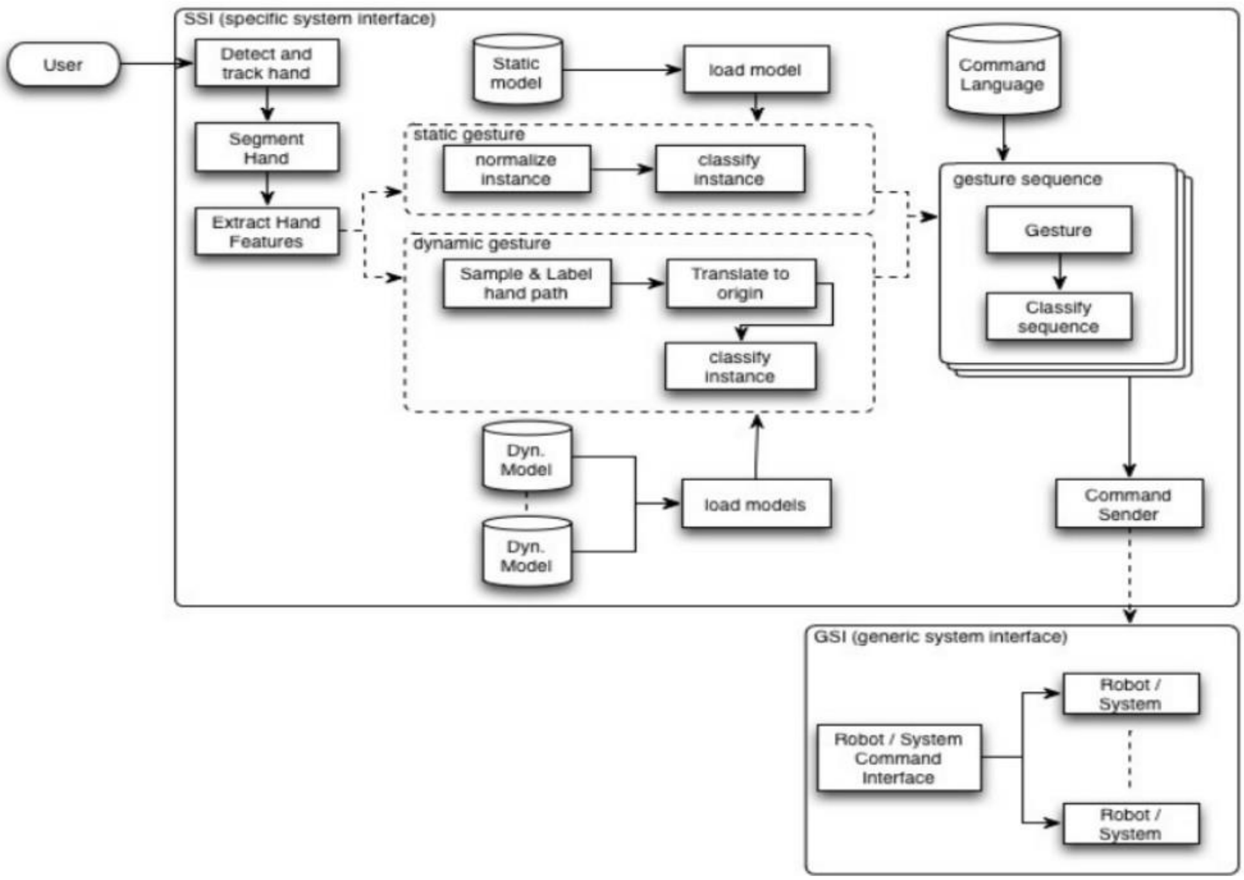
individuals			standardizing pixel values, and partitioning it into training and validationsets.			
Researchers and Academics	Data processing	USN-4	Examine and assess various deep learning architectures to choose the most appropriate model for the alphabet image recognition system.	We have the option to investigate different deep learning models.	High	Sprint-2
	Model devlopment	USN-5	Train the chosen deep learning model with the preprocessed dataset and assess its performance on the validation set.	Validation can be performed.	High	Sprint-3
	Training	USN-6	Incorporate data augmentation techniques, such as rotation and flipping, to enhance the model's resilience and accuracy.	Test can be performed	High	Sprint-3
	Model deployment & Integration	USN-7	Deploy the trained deep learning model as an API or web service for accessible alphabet image recognition. Integrate the model's API into a user-friendly web interface, allowing users to upload images and obtain classification results for garbage recognition.	We can assess scalability.	Medium	Sprint-4
	Testing and quality assurance.	USN-8	Perform comprehensive testing on the model and web interfaceto detect and report any issues or bugs. Refine the model hyperparameters and optimize performance based on userfeedback and testing results.	We have the option to develop a web application.	Medium	Sprint-5

Solution Architecture

Steps involved are:

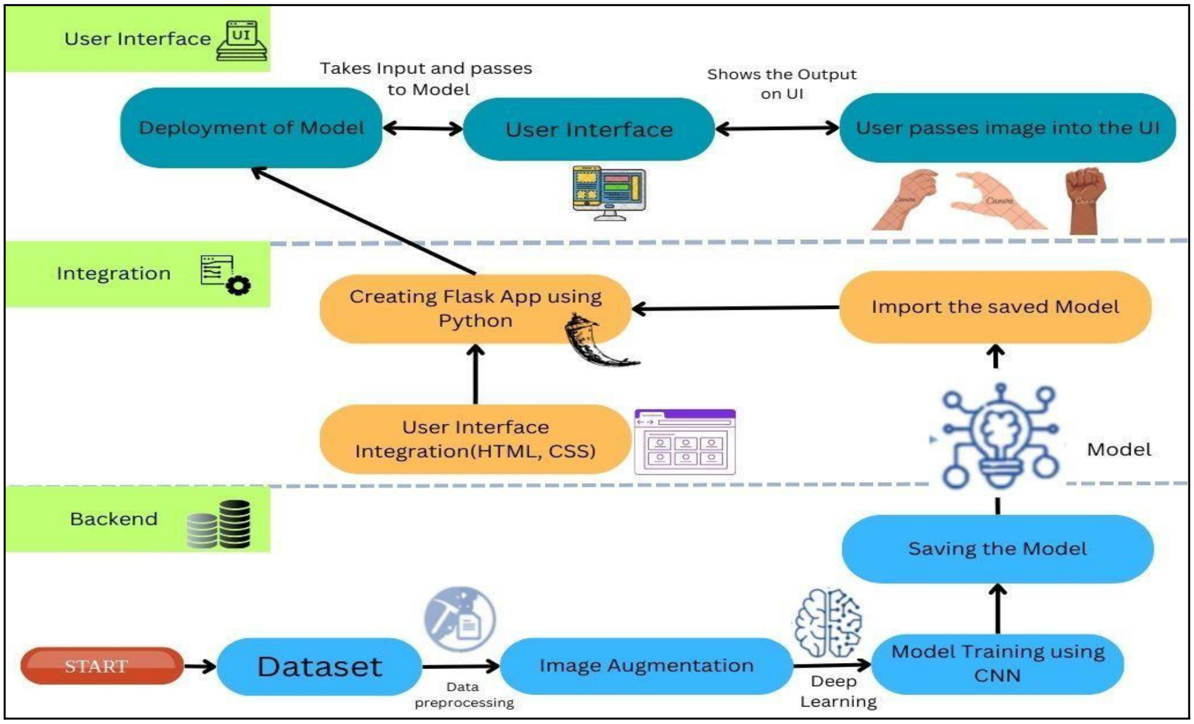
1.Data Collection and Preprocessing
2.Algorithm Selection
3.Model Training
4.Real-time Processing Optimization
5.User Interface Design
6.Testing and Validation
7.Deployment

Solution Architecture Diagram:



6. PROJECT PLANNING & SCHEDULING

Technical Architecture



Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup & Infrastructure	USN-1	Set up the development environment with therequired tools and frameworks to start the alphabet image recognition system	1	High	Sunith, Basera
Sprint-1	Development environment	USN-2	Gather a diverse dataset of images containing different types of ASL images(alphabet images) fortraining thedeep learning model.	2	High	Sunith, Vanshika

Sprint-2	Data collection	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	2	Medium	Vanshika
Sprint-2	Data preprocessing	USN-4	Explore and evaluate different deep learning architectures to select the mostsuitable model for the alphabet image recognition system..	3	High	Sunith, Vanshika
Sprint-3	Model Development	USL-5	Train the selected deep learning model usingthe preprocessed dataset and monitor its performance on the validation set.	4	High	Sunith, Basera
Sprint-3	Training	USL-6	Implement data augmentation techniques (e.g.,rotation, flipping) to improve the model's robustness and accuracy.	6	Medium	Sunith
Sprint-4	Model deployment & Integration	USL-7	Deploy the trained deep learning model as a APIor web service to make it accessible for alphabet image recognition, integrate the model's API into a user-friendly web interface forusers to upload images and receive garbage classification results.	1	Medium	Vanshika, Basera
Sprint-5	Testing & quality assurance	USL-8	Conduct thorough testing of the model and webinterface to identify and report any issues or bugs. fine-tune the model hyperparameters and optimize its performance based on userfeedback and testing results.	1	Medium	Basera
Sprint-6	Re-designing themodel	USL-9	Re-designing the web application and userinterface according to user feedbacks,	2	High	Vanshika, Sunith
Sprint-6	Re-deploying the model	USL-10	Re-deploying the new web interface and testingit with different scenarios	1	High	Sunith, Basera

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (ason Planned End Date)	Sprint Release Date(Actual)
Sprint-1	3	8 Days	2 Nov 2023	9 Nov 2023	3	9 Nov 2023
Sprint-2	5	3 Days	7 Nov 2023	9 Nov 2023	5	9 Nov 2023
Sprint-3	10	3 Days	7 Nov 2023	9 Nov 2023	10	9 Nov 2023
Sprint-4	1	8 days	8 Nov 2023	15 Nov 2023	1	15 Nov 2023
Sprint-5	1	2 days	9 Nov 2023	10 Nov 2023	1	10 Nov 2023
Sprint-6	3	4 days	10 Nov 2023	13 Nov 2023	3	13 Nov 2023

7. CODING & SOLUTIONING

7.1. Features

- Developed a user-friendly web application serving as the interface for interacting with the ASL image recognition system.
- Implemented functionality allowing users to upload both videos and images, enhancing the versatility and dynamism of the user experience for ASL word prediction.
- Incorporated a real-time feature that predicts ASL words from uploaded videos, expanding the application's functionality beyond static image recognition.
- This real-time capability facilitates immediate communication between users, aligning with the project's overarching goal of achieving seamless communication between the deaf and hearing communities.
- Enabled users to upload both videos and images, making the application adaptable to a variety of communication scenarios.
- The incorporation of diverse media uploads not only enriches user engagement but also accommodates different preferences in communication mediums.
- Please refer to the appendix for the added codes related to these features.

8.PERFORMANCE TESTING

Performance Metrics

1. Accuracy:- We have got training and testing accuracy as follows:-
   Training accuracy:- 94.98%
   Testing accuracy:- 95.03%

```
scores = model.evaluate(test_generator)
print("%s: %2f%%" % ("Evaluate Test Accuracy", scores[1]*100))
```

```
136/136 [==============================] - 23s 169ms/step - loss: 0.1744 - accuracy: 0.9503
Evaluate Test Accuracy: 95.034480%
```

2. Confusion matrix:-

```
tf.Tensor(
[[578   1   0   0   2   0   0   0   0   0   0   0   5   1   0   0   0   0
    7   1   1   0   0   1   0   3   0   0   0]
 [  0 585   0   0   1   1   0   0   0   0   1   0   0   0   1   0   0   5
    0   0   5   0   0   0   0   0   0   1   0]
 [  0   1 585   0   0   0   0   3   0   0   0   0   0   0   3   0   1   0
    1   1   0   0   0   0   0   0   2   0   3]
 [  0   6   0 584   0   2   0   0   1   0   0   0   0   0   5   0   0   1
    0   0   0   1   0   0   0   0   0   0   0]
 [ 12   6   0   0 565   1   0   0   5   0   0   0   2   0   1   0   0   0
    7   0   0   1   0   0   0   0   0   0   0]
 [  0   2   0   3   1 590   0   0   0   0   0   0   1   0   1   0   0   0
    1   0   0   0   0   0   1   0   0   0   0]
 [  3   0   0   0   0   0 572   5   2   6   0   0   0   0   0   5   0   0
    2   0   0   2   0   0   1   1   0   0   1]
 [  0   0   0   0   0   0   5 589   0   1   0   0   0   0   1   0   0   0
    0   0   0   0   0   0   0   2   0   2   0]
 [  1   6   0   1   4   0   0   0 568   8   0   0   0   0   0   0   0   4
    1   0   1   1   0   0   2   3   0   0   0]
 [  0   0   0   0   0   0   0   3   6 584   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   4   3   0   0   0]
 [  0  12   0   2   0   0   0   0   2   0 544   0   0   0   0   0   0   9
    0   0   0  27   3   0   0   0   0   1   0]
 [  2   0   0   0   0   0   2   0   5   3   0 572   0   0   0   0   0   0
    1  10   0   0   0   3   1   1   0   0   0]
 [  0   0   0   0   1   0   0   0   0   0   0   0 578  18   2   0   0   0
    1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   1   0   0   0   0  39 551   0   0   0   0
    4   0   3   0   0   0   0   0   1   1   0]
 [  0   0   0   2   0   0   0   0   0   0   0   0   0   0 585   2   0   1
    7   0   0   0   0   0   0   1   1   0   1]
 [  0   0   0   0   0   0   0   0   0   1   0   0   0   0   0 593   2   0
    1   0   1   0   0   0   0   0   1   0   1]
 [  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1 592   0
    0   0   0   0   0   0   0   1   4   0   1]
 [  0   0   0   0   0   0   0   1   4   0   1   0   0   0   0   0   0 560
    6   0  18   6   0   4   0   0   0   0   0]
 [  3   0   0   0   0   0   0   0   0   0   0   0   3   3   0   0   0   0
  578   2   2   0   0   4   0   3   0   1   1]
 [  0   0   0   0   0   0   0   0   0   1   0   0   1   0   0   0   0   0
   12 576   0   0   0   0   4   6   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0  50
    4   0 539   2   0   4   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   2   0  12   0   1   0   0   0   0  10
    7   1   5 548   8   6   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   6   0   1   0   0   0   0   2
    2   0   1  22 563   0   2   1   0   0   0]
 [  2   0   0   0   0   0   0   0   1   0   0   1   2   0   0   0   0   8
   44   3   3   1   0 529   0   5   0   1   0]
 [  0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0
    4   1   0   0   0   0 590   2   0   0   0]
 [  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   1
    9   0   0   0   0   1   3 584   0   0   0]
 [  0   0   0   0   0   0   0   2   0   0   0   0   1   1   1   1   2   0
    1   0   0   0   0   0   1   2 584   2   2]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   1   0   0   0   0   0   0   0 599   0]
```

3. Classification report:-

```
136/136 [==============================] - 24s 173ms/step
              precision    recall  f1-score   support

           A       0.97      0.85      0.91       600
           B       0.94      0.97      0.96       600
           C       0.98      0.99      0.99       600
           D       0.99      0.98      0.98       600
           E       0.95      0.95      0.95       600
           F       1.00      0.97      0.98       600
           G       0.93      0.98      0.96       600
           H       0.97      0.98      0.98       600
           I       0.96      0.87      0.91       600
           J       0.90      0.98      0.94       600
           K       0.90      0.93      0.92       600
           L       1.00      0.98      0.99       600
           M       0.82      0.95      0.88       600
           N       0.96      0.91      0.94       600
           O       0.97      0.97      0.97       600
           P       0.99      0.98      0.99       600
           Q       0.98      0.99      0.98       600
           R       0.92      0.81      0.86       600
           S       0.87      0.94      0.90       600
           T       0.96      0.97      0.97       600
           U       0.86      0.94      0.90       600
           V       0.92      0.89      0.91       600
           W       0.99      0.95      0.97       600
           X       0.95      0.88      0.91       600
           Y       0.98      0.98      0.98       600
           Z       0.94      0.97      0.96       600
         del       0.97      0.97      0.97       600
     nothing       0.98      0.99      0.99       600
       space       0.99      0.99      0.99       600

    accuracy                           0.95     17400
   macro avg       0.95      0.95      0.95     17400
weighted avg       0.95      0.95      0.95     17400
```

8. RESULTS

Output Screenshots –

```
# Testing with an image
image_path = '/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train/H/H108.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img,(32,32))

img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

# Predict the class of the image
predictions = model.predict(np.array([img]))

# Get the class with the highest probability
predicted_class = labels[np.argmax(predictions)]

print(f"The image is predicted to belong to class: {predicted_class}")
```

```
1/1 [==============================] - 0s 72ms/step
The image is predicted to belong to class: H
```

```
# Testing with an image
image_path = '/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train/B/B1008.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img,(32,32))

img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

# Predict the class of the image
predictions = model.predict(np.array([img]))

# Get the class with the highest probability
predicted_class = labels[np.argmax(predictions)]

print(f"The image is predicted to belong to class: {predicted_class}")
```

```
1/1 [==============================] - 0s 72ms/step
The image is predicted to belong to class: B
```

```python
# Testing with an image
image_path = '/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train/del/del274.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img,(32,32))

img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

# Predict the class of the image
predictions = model.predict(np.array([img]))

# Get the class with the highest probability
predicted_class = labels[np.argmax(predictions)]

print(f"The image is predicted to belong to class: {predicted_class}")
```

```
1/1 [==============================] - 0s 20ms/step
The image is predicted to belong to class: del
```

```python
# Testing with an image
image_path = '/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train/L/L100.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img,(32,32))

img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

# Predict the class of the image
predictions = model.predict(np.array([img]))

# Get the class with the highest probability
predicted_class = labels[np.argmax(predictions)]

print(f"The image is predicted to belong to class: {predicted_class}")
```

```
1/1 [==============================] - 0s 36ms/step
The image is predicted to belong to class: L
```

9. ADVANTAGES & DISADVANTAGES

Advantages:

Facilitating Communication: The primary advantage of the ASL Alphabet Image Recognition project is its potential to facilitate seamless communication between the deaf and hearing communities. By accurately recognizing ASL alphabet hand signs, the model can bridge the communication gap and promote inclusivity.

Real-time Application: The project aims to implement the model into a real-time application. This means that users can access the recognition system instantly, enhancing its practical utility in various scenarios where quick and accurate communication is crucial.

Accessibility: The model contributes to making information more accessible to the deaf community. It allows them to interact with technology and communication devices on an equal footing with the hearing population.

Education and Learning Aid: The project can serve as an effective tool for learning and practicing ASL. It can be integrated into educational settings to aid individuals, including those who are not deaf, in acquiring proficiency in ASL.

Empowerment: Enabling individuals to express themselves through ASL without relying on an interpreter empowers the deaf community. This autonomy is crucial for fostering independence and self-expression.

Disadvantages:

Limited Vocabulary: The model may have limitations in recognizing signs beyond the ASL alphabet. It might not be as effective in capturing the nuances of more complex signs or non-alphabetic gestures.

Variability in Gestures: ASL signs can vary based on factors such as speed, handshape variations, and individual differences. The model may struggle to accurately recognize signs in situations where these variables are prominent.

Environmental Factors: The accuracy of the model may be influenced by environmental factors such as lighting conditions and background clutter. Adverse conditions could impact the model's performance, especially in real-world scenarios.

Hardware Requirements: For real-time applications, the project may have specific hardware requirements. High processing power and efficient cameras may be necessary for optimal performance, potentially limiting accessibility for users with older devices.

Ethical Considerations: The use of technology in communication raises ethical concerns, particularly in terms of privacy and data security. It's crucial to address these concerns and implement safeguards to protect user information.

Cultural Sensitivity: ASL is not a universal language, and there can be cultural variations in sign language. The model may need adaptations to cater to different sign language variants, limiting its applicability in diverse cultural contexts.

10. CONCLUSION

In conclusion, the 'ASL Alphabet Image Recognition' project successfully amalgamated empathy-driven design, agile development methodologies, and cutting-edge technology. The incorporation of empathy maps, user stories, and agile sprints ensured the creation of a user-centric solution. Notably, the VGG16 model demonstrated a remarkable 95% accuracy,underscoring its practicality in real-world applications. The web application, featuring seamless video and image uploads, enriches communication by providing real-time predictions of ASL words. This dynamic approach is in alignment with our overarching objective of diminishing communication barriers between deaf and hearing communities. To enhance the project further, future iterations could concentrate on expanding the vocabulary and refining adaptability to diverse signing styles. Overall, the project serves as a testament to the transformative power of technology in promoting inclusivity and accessibility across diverse communities.

11. FUTURE SCOPE

Looking ahead, the 'ASL Alphabet Image Recognition' project holds promising avenues for future development. Expanding the model's vocabulary to encompass a broader range of ASL signs and refining its adaptability to diverse signing styles can enhance its utility. Additionally, incorporating real-time translation features for conversational ASL holds potential for expanding its impact.

Collaboration with the deaf community for continuous feedback and improvement, along with exploring mobile applications for increased accessibility, could be valuable directions.
Integrating cultural adaptations to cater to different sign language variants would make the solution more inclusive on a global scale.

Moreover, exploring opportunities for partnerships with educational institutions could turn the project into a valuable learning tool for ASL acquisition. Ongoing research and development efforts could further optimize the model, making it even more accurate and efficient.

In conclusion, the future scope of the 'ASL Alphabet Image Recognition' project lies in continual refinement, expansion, and collaborative efforts, ensuring its sustained relevance and positive impact in facilitating communication between diverse communities.

12. APPENDIX

GitHub & Project Demo Link -
https://github.com/smartinternz02/SI-GuidedProject-615128-1700663470

Source Codes -

HTML –

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gesture Recognition Hub</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>

<body>
    <header>
        <div class="header-content">
            <h1>Gesture Recognition Hub</h1>
        </div>
    </header>
    <div class="gesture-container">
        <div class="gesture-info">
            <div class="info-section">
                <p>Gestures are a universal language that transcends barriers. Explore the world of visual communication with our recognition tool.</p>
                <p>Get a glimpse of diverse gestures through the image below. It encompasses gestures from A-Z and includes symbols for space, delete, and more.</p>
                <img src="https://img.freepik.com/premium-vector/deaf-mute-language-american-deaf-hand-gesture-alphabet-letters-asl-vector-alphabetical-symbols_176411-1575.jpg" alt="Gesture image">
            </div>
        </div>
        <div class="prediction-container">
            <h3>Curious about gestures? Utilize our AI tool to effortlessly predict gestures</h3>
            <form id="upload-form" action="/predict" method="post" enctype="multipart/form-data">
            <input type="file" name="file" id="file-input" accept="image/*" capture="camera" onchange="previewImage(this)">
            <label for="file-input" id="upload-label">Choose or Capture Image</label>
            <input type="submit" value="Upload" id="upload-button">
        </form>
            <div id="image-preview" class="gesture-preview"></div>
            <button type="button" id="predict-button" onclick="predict(event)">Predict</button>
            <div id="result"> </div>
        </div>
    </div>
    <script src="{{url_for('static',filename='script.js')}}"></script>
    <input type="file" name="file" id="file-input" accept="image/*" capture="camera" onchange="previewImage(this, 300)">
</body>

</html>
```

CSS –

```
        body {
    font-family: 'Verdana', sans-serif;
    background-color: #EFEFEF;
    margin: 0;
    padding: 0;
}

header {
    margin: 15px;
    position: relative;
    text-align: center;
    color: #4A90E2;
    background-color: #FFD700;
    padding: 40px;
}

.header-content {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

.gesture-container {
    display: flex;
    justify-content: space-around;
    max-width: 1300px;
    margin: 15px auto;
}

.gesture-info {
    max-width: 800px;
    margin: 30px auto;
    margin-right: 60px;
    text-align: center;
    padding: 30px;
    background-color: #FFF;
    border-radius: 12px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
}

.prediction-container {
    max-width: 400px;
```

```
    margin: 30px auto;
    padding: 30px;
    text-align: center;
    background-color: #FFF;
    border-radius: 12px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
    position: relative;
    max-height: 800px;
    overflow-y: auto;
}

.image-section {
    background-color: #4285F4;
    color: #FFF;
    padding: 30px;
    border-radius: 12px;
    margin-bottom: 30px;
}

.info-section {
    background-color: #34A853;
    color: #FFF;
    padding: 30px;
    border-radius: 12px;
    margin-bottom: 30px;
}

.info-section h2 {
    margin-bottom: 20px;
}

#file-input {
    display: none;
}

#upload-label {
    background-color: #34A853;
    color: #FFF;
    padding: 20px 40px;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin-bottom: 20px;
}
```

```css
.upload-label {
    background-color: #34A853;
    color: #FFF;
    padding: 20px 40px;
    border-radius: 8px;
    cursor: pointer;
    margin: 10px;
}

#predict-gesture-button {
    background-color: #FF4500;
    color: #FFF;
    padding: 20px 40px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s;
}

#predict-gesture-button:hover,
#predict-button:hover {
    background-color: #D23F00;
}

#upload-label:hover {
    background-color: #228B22;
}

#video-upload,
#image-upload {
    background-color: #34A853;
    color: #FFF;
    padding: 20px 40px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin: 10px;
}

#predict-button {
    position: absolute;
    bottom: 30px;
```

```
    left: 50%;
    transform: translateX(-50%);
    background-color: #FF4500;
    color: #FFF;
    padding: 20px 40px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin-top: 20px;
}

#upload-button:hover,
#predict-button:hover {
    background-color: #D23F00;
}

#result {
    position: absolute;
    bottom: 10px;
    width: 100%;
    font-weight: bold;
    text-align: center;
    color: #333;
}

.gesture-preview {
    margin: 30px auto;
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
    position: relative;
}

.flex-container {
    display: flex;
}

.preview-image {
    max-width: 120px;
    margin: 15px;
}

.upload-form {
```

```
            display: flex;
            flex-direction: column;
            align-items: flex-end;
        }

        #upload-instruction {
            margin-top: 20px;
        }

        .upload-options {
            display: flex;
            flex-direction: column;
            align-items: center;
            margin-top: 20px;
        }

        .upload-label {
            margin-top: 20px;
}
```

Javascript -

```
function previewImage(input) {
    const preview = document.getElementById('image-preview');
    preview.innerHTML = '';

    if (input.files && input.files[0]) {
        const reader = new FileReader();

        reader.onload = function (e) {
            const img = document.createElement('img');
            img.src = e.target.result;
            img.style.maxWidth = '100%';
            preview.appendChild(img);
        };

        reader.readAsDataURL(input.files[0]);
    }
}

function predict(event) {
    event.preventDefault();  // Prevent the default form submission behavior

    const form = document.getElementById('upload-form');
    const resultElement = document.getElementById('result');

    const formData = new FormData(form);
    fetch('/predict', {
        method: 'POST',
```

```
      body: formData
    })
    .then(response => response.json())
    .then(data => {
      resultElement.innerText = 'Prediction: ' + data.prediction;
    })
    .catch(error => console.error('Error:', error));
}
```

Python(Flask application)-

```
from flask import Flask, render_template, request, jsonify
from tensorflow.keras.models import load_model
from PIL import Image
import numpy as np

app = Flask(_name_)

# Load your model
model = load_model('weights.h5',compile=False)  # Update with your actual path

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

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        if 'file' not in request.files:
            return jsonify({'error': 'No file part'})

        file = request.files['file']

        if file.filename == '':
            return jsonify({'error': 'No selected file'})

        labels = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
'del', 'nothing', 'space']

        # Process the image for prediction (you might need to resize, normalize, etc.)
```

```
        img = Image.open(file)
        img = img.resize((32, 32))  # Adjust the size according to your model's input shape
        img_array = np.array(img) / 255.0  # Normalize
        img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension

        # Make prediction
        prediction = model.predict(img_array)

        predicted_class = labels[np.argmax(prediction)]

        text = "Your image represents "+predicted_class
        return jsonify({'prediction': text})


if _name_ == '_main_':
    app.run(debug=False, threaded = False)
```

Python(Deep learning model)-

```
!mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d grassknoted/asl-alphabet
!kaggle datasets download -d grassknoted/asl-alphabet
# Load Data
import os
import cv2
import numpy as np

# Data Visualisation
import matplotlib.pyplot as plt

# Model Training
from tensorflow.keras import utils
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D,
BatchNormalization
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import VGG16
```

```python
# Warning
import warnings
warnings.filterwarnings("ignore")

# Main
import os
import glob
import cv2
import numpy as np
import pandas as pd
import gc
import string
import time
import random
from PIL import Image
from tqdm import tqdm
tqdm.pandas()

# Visualization
import matplotlib
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

# Model
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array,
array_to_img
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
from keras.models import load_model, Model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import classification_report
# Configuration
class CFG:
    # Set the batch size for training
    batch_size = 128
    # Set the height and width of input images
    img_height = 32
    img_width = 32
    epochs = 10
```

```
    num_classes = 29
    # Define the number of color channels in input images
    img_channels = 3
# Define a function to set random seeds for reproducibility
def seed_everything(seed: int):
    random.seed(seed)
    # Set the environment variable for Python hash seed
    os.environ["PYTHONHASHSEED"] = str(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)
# Labels
TRAIN_PATH = "/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train"
labels = []
# Generate a list of uppercase letters in the English alphabet
alphabet = list(string.ascii_uppercase)
labels.extend(alphabet)
# Add special labels for 'delete', 'nothing', and 'space' gestures
labels.extend(["del", "nothing", "space"])
print(labels)
# Create Metadata
list_path = []
list_labels = []
for label in labels:
    # Create a path pattern to match all image files for the current label
    label_path = os.path.join(TRAIN_PATH, label, "*")
    # Use glob to retrieve a list of image file paths that match the pattern
    image_files = glob.glob(label_path)
    sign_label = [label] * len(image_files)
    list_path.extend(image_files)
    list_labels.extend(sign_label)
metadata = pd.DataFrame({
    "image_path": list_path,
    "label": list_labels
})
metadata
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    metadata['image_path'],
    metadata['label'],
    test_size=0.2,
    random_state=2253,
    shuffle=True,
    stratify=metadata['label']
)
```

```
# Create a DataFrame for the training set test set
data_train = pd.DataFrame({
    'image_path': X_train,
    'label': y_train
})

data_test = pd.DataFrame({
    'image_path': X_test,
    'label': y_test
})

# Split the training set into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(
    data_train['image_path'],
    data_train['label'],
    test_size=0.2/0.7,  # Assuming you want 20% for validation out of the training
set
    random_state=2253,
    shuffle=True,
    stratify=data_train['label']
)

# Create a DataFrame for the validation set
data_val = pd.DataFrame({
    'image_path': X_val,
    'label': y_val
})
def data_augmentation():
    datagen = ImageDataGenerator(
        rescale=1/255.,
        # Add other augmentation parameters as needed
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )
    train_generator = datagen.flow_from_dataframe(
        data_train,
        directory='./',
        x_col='image_path',
```

```
        y_col='label',
        class_mode='categorical',
        batch_size=CFG.batch_size,
        target_size=(CFG.img_height, CFG.img_width)
    )
    validation_generator = datagen.flow_from_dataframe(
        data_val,
        directory='./',
        x_col='image_path',
        y_col='label',
        class_mode='categorical',
        batch_size=CFG.batch_size,
        target_size=(CFG.img_height, CFG.img_width)
    )

    test_generator = datagen.flow_from_dataframe(
        data_test,  # Assuming you have a DataFrame for test data
        directory='./',
        x_col='image_path',
        y_col='label',
        class_mode='categorical',
        batch_size=CFG.batch_size,
        target_size=(CFG.img_height, CFG.img_width),
        shuffle=False  # Set to False for test data
    )

    return train_generator, validation_generator, test_generator

# Seed for reproducibility
seed_everything(2253)

# Get the generators
train_generator, validation_generator, test_generator = data_augmentation()
# Define input shape
input_shape = (32, 32, 3)

# Load the VGG16 model without the top (classification) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)

# Add your custom classification layers on top of the base model
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(128, activation='relu')(x)  # You can adjust the number of units as needed
predictions = Dense(29, activation='softmax')(x)  # num_classes is the number of
classes in your dataset
```

```
# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Summarize the model architecture
model.summary()
# Compile the model
model.compile(optimizer=Adam(lr=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])

# Create a ModelCheckpoint callback
checkpoint_callback = ModelCheckpoint(
    filepath='/content/sample_data/best_model_weights.h5',
    monitor='val_accuracy',  # Monitor validation accuracy for saving the best model
    save_best_only=True,
    mode='max',
    verbose=1
)
# Train the model using the fit method
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // CFG.batch_size,  # Number of steps
per epoch
    epochs=CFG.epochs,  # Number of training epochs
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // CFG.batch_size,  # Number of
validation steps
    callbacks=[checkpoint_callback],
    shuffle=True,
    verbose=1
)
scores = model.evaluate(test_generator)
print("%s: %2f%%" % ("Evaluate Test Accuracy", scores[1]*100))
# Confusion Matrix:-
fine_tuned_model = load_model("/content/sample_data/best_model_weights.h5")
predictions = fine_tuned_model.predict(test_generator)
# Get the true labels from the generator
true_labels = test_generator.classes
# Compute the confusion matrix using tf.math.confusion_matrix
confusion_matrix = tf.math.confusion_matrix(
labels = true_labels,
predictions = predictions.argmax(axis=1),
num_classes = 29
)
```

```
#Classification report
predictions = model.predict(test_generator)
predicted_labels = np.argmax(predictions, axis=1)
true_labels = test_generator.classes
report = classification_report(true_labels, predicted_labels, target_names=labels)
print(report)
# Load the saved model
model = tf.keras.models.load_model('/content/sample_data/best_model_weights.h5')
# Testing with an image
image_path = '/content/asl-alphabet/asl_alphabet_train/asl_alphabet_train/Y/Y10.jpg'
img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img,(32,32))
img = tf.keras.applications.mobilenet_v2.preprocess_input(img)
# Predict the class of the image
predictions = model.predict(np.array([img]))
# Get the class with the highest probability
predicted_class = labels[np.argmax(predictions)]
print(f"The image is predicted to belong to class: {predicted_class}")
```