



Gyalpozhing College of Information Technology

Royal University of Bhutan

Kabjisa, Chamjekha, Thimphu

**AI and Data Science**

PRJ303 3rd Year Project

Spring 2024

AI-Powered Sign Language to Voice Assistant for Mute

Submitted by:

Bidash Gurung 12210001

Asseh Nepal 12210041

Pema Chozom 12200024

Pema yangchen 12200025

Thukten Dema 12200037

Supervisor: Ms.Tawmo

| [1. Background Information](#_78c9jjqc85xr) | 4 |
| --- | --- |
| 2.[. Problem Statement](#_ctk5868cyufw) | 4 |
| [3. Aim](#_jiji9hwwrx8y) | 4 |
| [4. Objectives](#_gggxligh9lsm) | 5 |
| [5. Features](#_5rla69xrarrw) | 5 |
| [6. Scope](#_u7xenfkdj6em) | 6 |
| [6.1. System Scope](#_2hzalq5m0u02) | **6** |
| [6.2. User scope](#_nksl5f7c1hco) | 6 |
| [7. Feasibility](#_6t3kr19jdgvj) | 6 |
| [7.1. Technical Feasibility](#_k4gie38up00q) | 6 |
| [7.2. Operational Feasibility](#_djfuecjlz50n) | 6 |
| [7.3. Legal and Ethical Feasibility](#_puobdmyc6yo3) | 7 |
| [7.4. Schedule Feasibility](#_1c165d5pxhpi) | 7 |
| [8. Requirements](#_irqa93dglsq8) | 7 |
| [8.1. Functional Requirements](#_idkuuydt1ts1) | 7 |
| [8.2. Non- functional Requirements](#_jhbvfzdzfalo) | 8 |
| [8.3. System requirements](#_7ansxff79ls9) | 8 |
| [9. Data Sources](#_9je7pd22bfj3) | 9 |
| [10. Methodology](#_7ermdayy0n5p) | 9 |
| [11. Design](#_7gxsogg99x32) | 12 |

| [11.1. Architecture: 3-tier architecture](#_3ezzp1eoj8m5) | 12 |
| --- | --- |
| [11.2. Workflow based on model training](#_m1a7uvs5cnsg) | 13 |
| [11.3. Workflow based on user interaction](#_ezfc5ksa4dfe) | 14 |
| [11.4. Use Case Diagram](#_1un2g5yn4lz5) | 15 |
| [12. Literature Review](#_m6qzdm73hobf) | 16 |
| [13. Project Milestone](#_f8zw3wo1wwc5) | 17 |
| 14. Prototype | 20 |

# Project Background

In the ever-evolving landscape of artificial intelligence (AI), researchers and developers continually seek innovative ways to enhance user experiences. One such breakthrough is the emergence of AI-powered image-to-voice assistants. These systems combine visual understanding with natural language processing (NLP) to create a more intuitive and versatile interaction paradigm.

The primary purpose of these assistants is to bridge the gap between visual information and spoken language. Traditionally, users interacted with AI systems either through text input or voice commands. However, the real world is inherently multimodal, where we perceive and communicate using both visual cues and spoken language. Image-to-voice assistants aim to replicate this natural interaction by allowing users to engage with AI using both visual input (captured through images) and voice commands.

AI-powered image-to-voice assistants represent a significant leap in human-computer interaction. By integrating visual and auditory modalities, they offer a more holistic and engaging experience. As technology continues to evolve, we can expect further advancements in multimodal AI, making our interactions with machines even more intuitive and delightful.

# Problem Statement

The disabled individuals, particularly those who are mute, often face challenges as they are not easily understood by caregivers or others who do not understand sign language. To address this issue, we propose the development of a mobile AI application capable of interpreting gestures and translating them into text and speech.

# 

# Aim

To develop an AI-powered voice assistant that enables non-verbal individuals to communicate using images and receive textual as well as spoken responses.

# Objectives

* Train deep learning models on a diverse dataset to improve the accuracy and robustness of image recognition.
* Implement a speech synthesis system that can generate natural-sounding speech from text based on the gestures.
* Implement notification sending system in case there is an emergency with the specified emergency gesture.
* Design a user-friendly interface for capturing images and interacting with the voice and text assistant.
* Integrate the developed Deep learning model in mobile applications. And deploy in one of the free deployment platforms.

# Features

1. **Image Recognition:**

* Ability to recognize gestures within images captured by the camera or through video capture.
* Support for a wide range of gestures, especially all basic hand gestures.

1. **Speech Synthesis:**

* Integration with a speech synthesis system to convert text-based responses into natural-sounding speech.

1. **Emergency Gesture Notification:**

* Detection of predefined emergency gestures, such as specific hand movement, indicating an urgent situation.
* Automatic sending of notifications to one of their contacts for immediate help, along with an alarming sound produced from the device so that caregivers can be alerted.

1. **Real-time Processing:**

* Real-time processing of images and gestures to provide immediate feedback and response to user inputs.
* Optimization of algorithms and hardware resources to minimize latency and ensure smooth performance of the system.

1. **Privacy and Security Measures:**

* Implementation of robust data encryption, authentication, and access control mechanisms to protect user privacy and prevent unauthorized access to sensitive information.

# Scope

## 6.1. System Scope

The system will be able to interpret basic hand gestures and provide textual as well as spoken responses based on the gesture.

## 6.2. User scope

This system is for the dumb/mute people (those people who don't have the ability to speak), their caregivers, and those people who are willing to interact with them.

# **Feasibility**

## **7.1.** **Technical Feasibility**

Assessing the availability of robust image recognition and speech synthesis technologies, along with high processing power computers for training models.

## **7.2. Operational Feasibility**

Determining user acceptance, training needs, support requirements, and integration capabilities with existing systems.

## 7.3. Legal and Ethical Feasibility

Ensuring compliance with data privacy regulations, addressing intellectual property rights, and considering ethical implications such as bias and consent.

## 7.4. Schedule Feasibility

Developing a realistic project timeline and assessing resource availability to ensure timely delivery and effective project management.

# Requirements

## 8.1. Functional Requirements

### 8.1.1. Register

User needs to register to access the app, he/she can register with a username, caregiver name, and caregiver phone number.

### 8.1.2. Login

Users can login and logout of the system using their username.

### 8.1.3. Show Gesture

Users can show gestures, especially hand gestures.

### 8.1.4. Output

If the gesture matches with the trained model then it would show the output saying what the gesture is trying to convey in text and voice form.

### 8.1.5. Notification

The caregiver will receive a notification in his or her email/phone if the gesture indicates the emergency.

### **8.1.**6**. Profile**

User can edit and update username, caregiver name and caregiver phone number

## 8.2. Non- functional Requirements

Some of the non-functional requirements of the Ai Powered Sign Language to Voice Assistant for are:

1. **Reliability:** The system should be able to operate consistently and reliably, with a low probability of failure.
2. **Security:** The system should be secure, with appropriate measures in place to prevent unauthorized access or misuse of data.
3. **Scalability:** The system should be in position to scale up with the additional or the improvement of existing features.
4. **Maintainability:** The system should be easy to maintain and repair, with simple and clear documentation.
5. **Usability:** The system should be intuitive and easy to use, with user-friendly interfaces and training materials.
6. **Performance:** The system should perform well in accuracy, speed, and sensitivity, with minimal delays or downtime.
7. **Portability:** The system should be easily portable to different locations or environments, with minimal setup or configuration required.

## 8.3. System requirements

Tools and Technologies:

### 8.3.1. Front-end

* Programming languages: Dart
* Visual design tools: Figma, Sketch, Adobe Illustrator.
* Code editors: Visual Studio Code and Google Colab.

### 8.3.2. Back-end

* Programming languages: Dart.
* Databases: Firebase.
* Frameworks: Flutter
* Libraries: TensorFlow, Keras, PyTorch, Scikit-learn, OpenCV, imgaug, fast.ai
* Version control systems: GitLab.

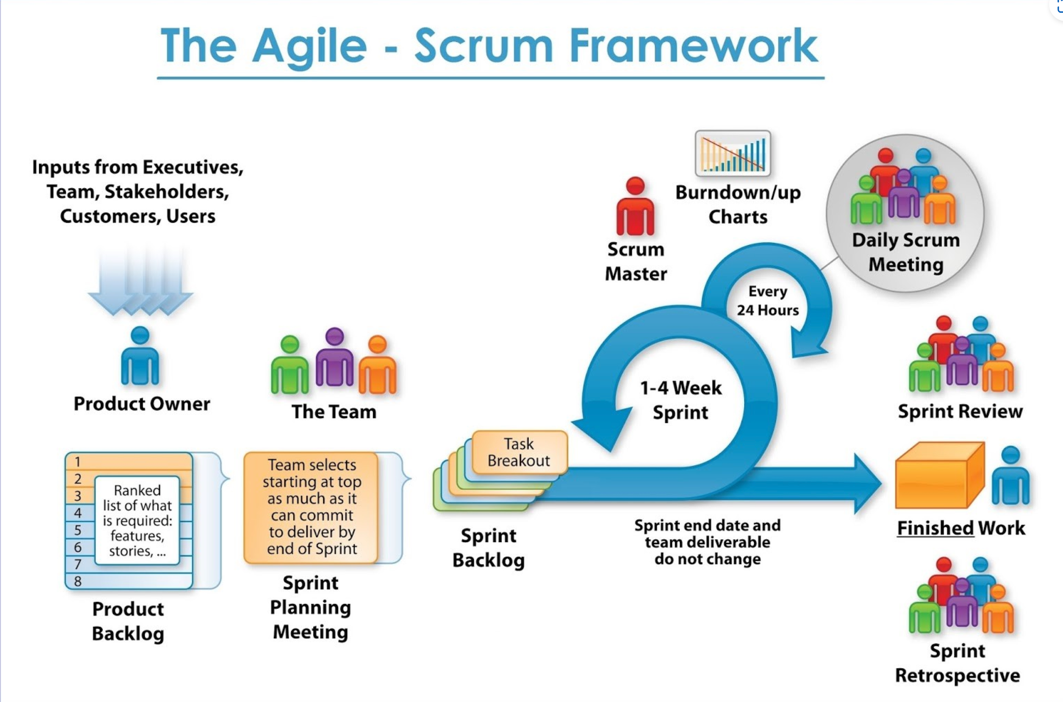
# 9. Data Sources

Since there is no good or expected data available on online platforms, we are going to collect the data manually by clicking photos, with the help of some E-books, YouTube videos, and some teachers who teach the dumb/mute people.

# 10. Methodology

## **10.1. Methodology of the study**

We want to use/follow Agile for the development of both the systems; “AI Powered Sign Language to Voice Assistant for Mute” Mobile Application. Agile project management is not a singular framework; it can be used as an umbrella term to include many different frameworks. For these systems, the SCRUM framework was used as it is most suited for software development. Scrum framework consists of meetings, roles, and tools to help teams working on complex projects collaborate, better structure, and manage the workload.



*Figure 1 Agile methodology*

The stages of agile methodology that were followed are:

1. **Project initiation**

The product owner, scrum master, and development team compose the scrum team during this phase. The team held discussions to determine the project's requirements and came to a thorough knowledge of its goals, by having a meeting with Mr. Adwin, who is like a client of our project.

1. **Design**

The project's design is created based on the requirements gathered, and this includes developing a prototype as well as the architectural, user interface, and database designs.

1. **Planning and Estimate**

The team decided which user stories from the product backlog to work on during the sprint during this phase. The team decided on the sprint target, listed the tasks needed to finish the chosen backlog items, and calculated the amount of time needed for each work.

1. **Implementation**

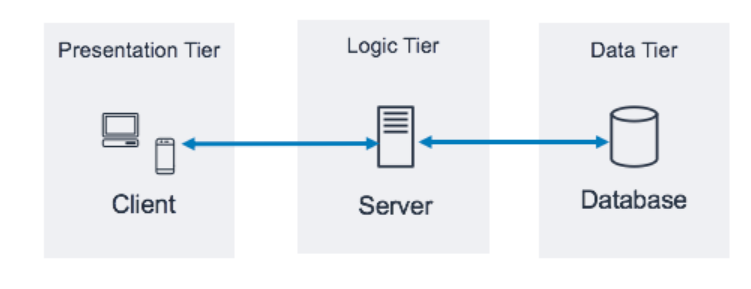
The team will begin working on the tasks listed in the sprint backlog as soon as the sprint planning is finished. Also assessed simultaneously were other tasks. Daily stand-up meetings, often known as daily scrums, are quick gatherings held throughout the sprint to coordinate work, talk about progress, and identify and remove any obstacles. Each team member discussed the tasks they completed the day before, the tasks they planned to complete today, and any challenges they were currently facing.

## 10.2. Overall training of the model

* **Data Collection:** Gather images, text descriptions, and speech recordings.
* **Preprocessing:** Clean and format data for analysis.The image data by resizing, normalizing, and augmenting images if necessary.
* **Feature Extraction:** Extract features from images, text, and speech. Utilizing pre-trained convolutional neural network (CNN) models for image feature extraction. Extract features from intermediate layers of the CNN to capture rich representations of images.
* **Model Training:** Train a model to understand relationships between data. Train a multimodal neural network, integrating preprocessed image, text, and speech features, while optimizing with suitable loss functions and regularization techniques to prevent overfitting.
* **Evaluation:** Assess model performance. Perform dataset splitting into training, validation, and test sets, evaluate model performance using validation metrics like accuracy, precision, recall, and F1-score, tune hyperparameters accordingly, and validate the model's generalization on the test set.
* **Prediction:** Use the model to process images and generate voice descriptions.
* **Iteration:** Continuously improve the model based on feedback and performance.

# 11. Design

## 11.1. Architecture: 3-tier architecture

*****Figure 2: 3-tier architecture*

### 11.1.1. Presentation tier

The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as a desktop application, or a graphical user interface (GUI), for example. Web presentation tiers are usually developed using HTML, CSS and JavaScript. Desktop applications can be written in a variety of languages depending on the platform.

### 11.1.2. Application tier

The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

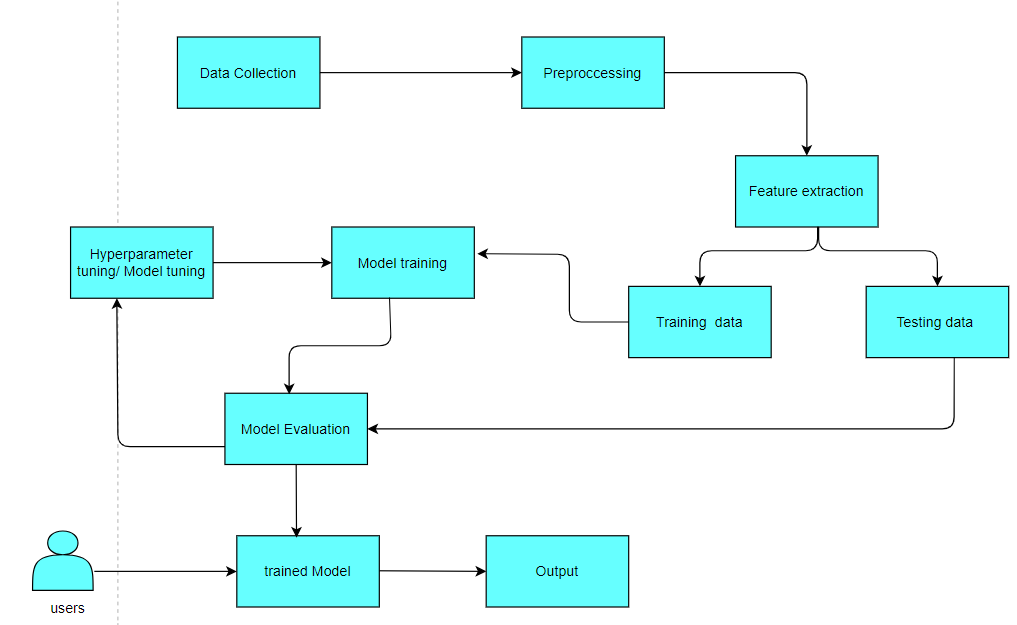
The application tier is typically developed using Python, Java, Perl, PHP or Ruby, and communicates with the data tier using [API](https://www.ibm.com/topics/api) calls.

### 11.1.3. Data tier

The data tier, sometimes called database tier, data access tier, or back-end, is where the information processed by the application is stored and managed. This can be a [relational database management system](https://www.ibm.com/topics/relational-databases) such as [PostgreSQL](https://www.ibm.com/topics/postgresql), MySQL, or in a [NoSQL](https://www.ibm.com/topics/nosql-databases) Database server such as [MongoDB](https://www.ibm.com/topics/mongodb).

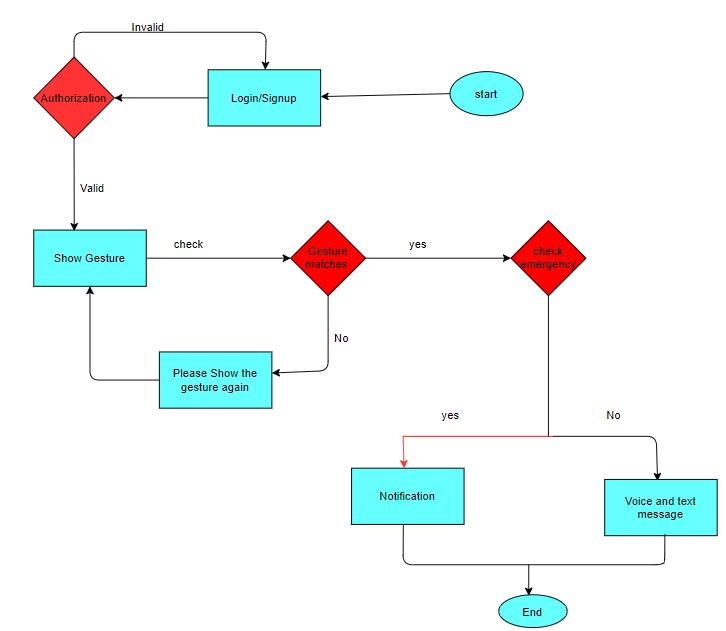
In a three-tier application, all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another.

## 11.2. Workflow based on model training

****

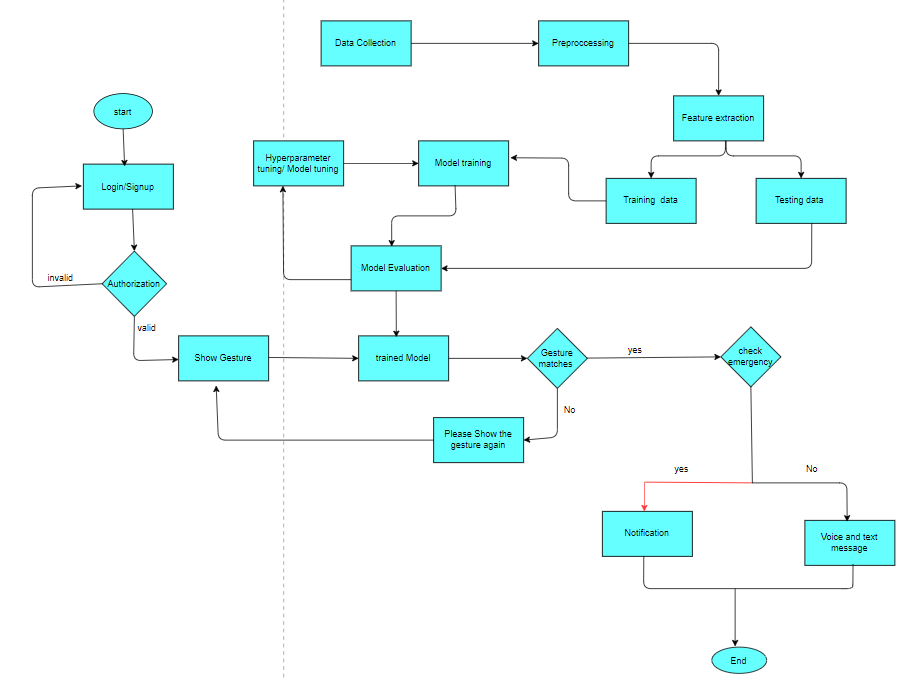
*Figure 3 workflow for model training*

## 11.3. Workflow based on user interaction

****

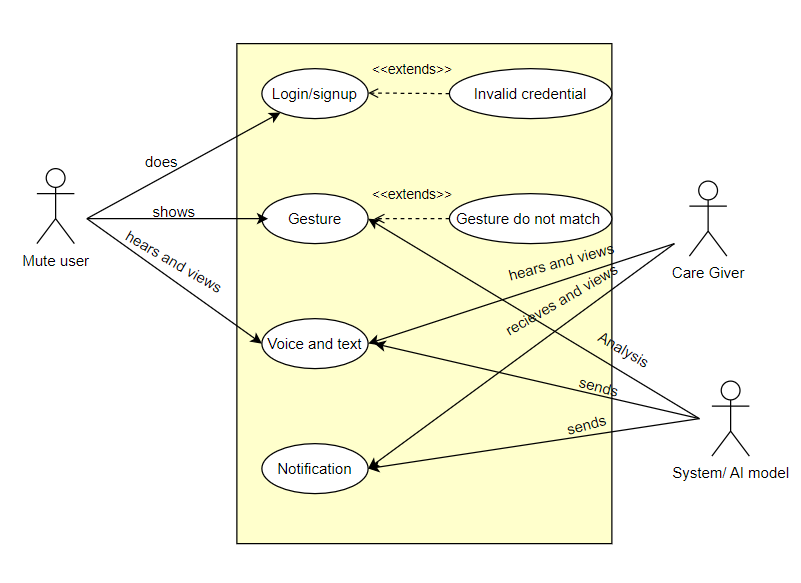
*Figure 4 workflow based on user*

**11.4. Overall Workflow**

****

*Figure 5 Overall workflow*

## 11.4. Use Case Diagram

*****Figure 6 use case diagram*

# 12. Literature Review

## 12.1.Giphy

GIPHY, the online database and search engine that lets you search for animated GIFs, has a new channel now that consists of more than 2,000 educational sign language clips. The channel was created in collaboration with Robert DeMayo, the creator of Sign With Robert, who has been deaf since birth. The Sign with Robert gifs can be browsed by categories (Date & Time & Weather, Expression, Emotions, Lifestyle, etc.) and subcategories (Days of the Week, Months and Season; Common Expressions, Everyday Expressions; Positive Emotions, Neutral Emotions; Fashion & Clothing, Internet & Social Media, etc.).

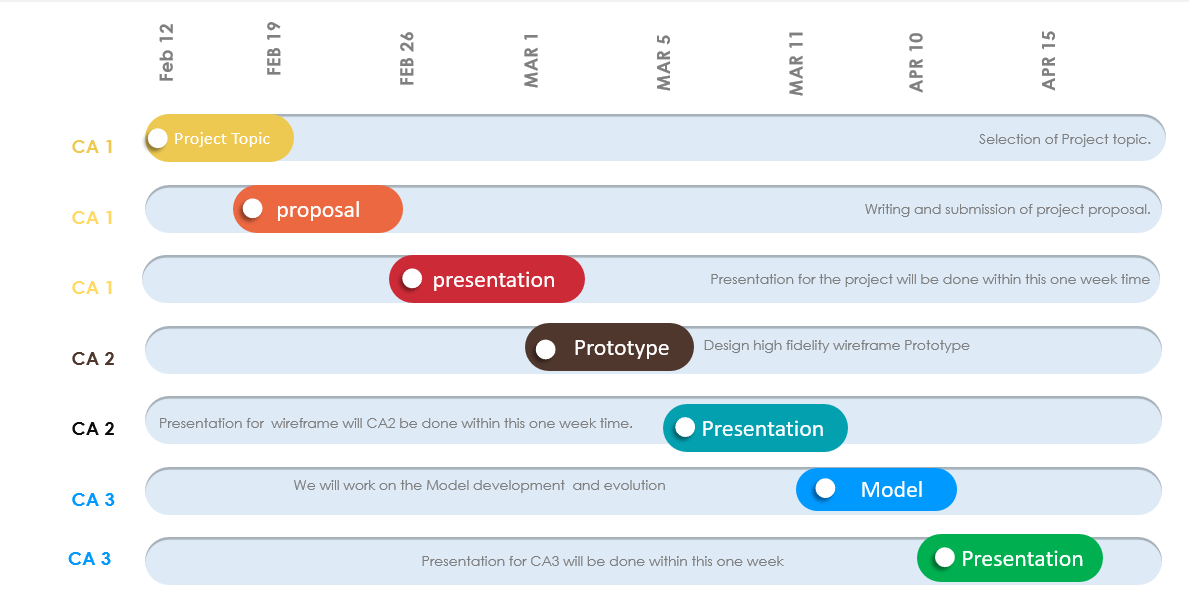
## 12.2. BHSL

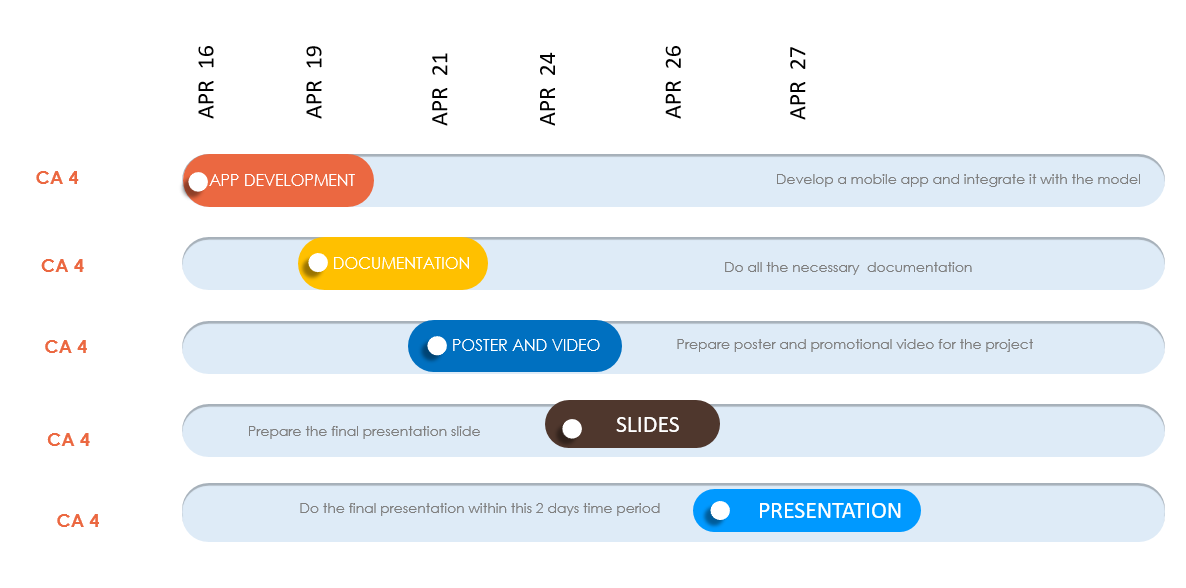
BHSL is another mobile application developed by Bhutanese developers. In this application, users can only type predefined words and receive voice (in Dzongkha language) and video-based gestures. By observing these video-based gestures, users can learn to understand the gestures of non-speaking individuals. Similar to Synopsis, the weakness of this system is that non-speaking individuals may struggle to communicate with others who have not learned sign language or gestures through the app. To address this weakness, our system will immediately generate voice and text messages when it detects the gestures made by non-speaking individuals. This will enable normal individuals or caregivers to understand the basic needs of non-speaking individuals without knowledge of sign language or gestures, with the assistance of our mobile application.

## 12.3. Hand Talk

Hand Talk is also a mobile application similar to Synopsis. However, unlike Synopsis, this application can display video-based gestures and handle some simple sentences such as "What are you doing?", "How are you doing?", "Where are you going?", and so on. As with the previous applications, the weakness of this system is that non-speaking individuals may struggle to communicate with others who have not learned sign language or gestures through the app. To address this weakness, our system will immediately generate voice and text messages when it detects the gestures made by non-speaking individuals. This will enable normal individuals or caregivers to understand the basic needs of non-speaking individuals without knowledge of sign language or gestures, with the assistance of our mobile application.

# 13. Project Milestone





# 14. Prototype

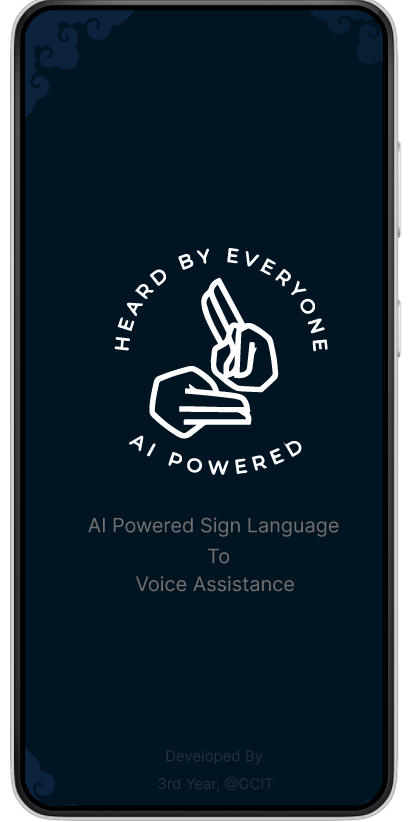
14.1 Rough Reference

Rough reference of the pictures, icons, logo and colors used in the application

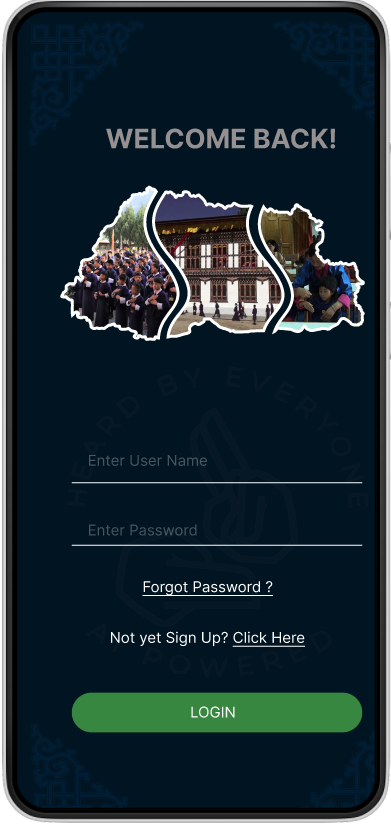


14.2 Authentication

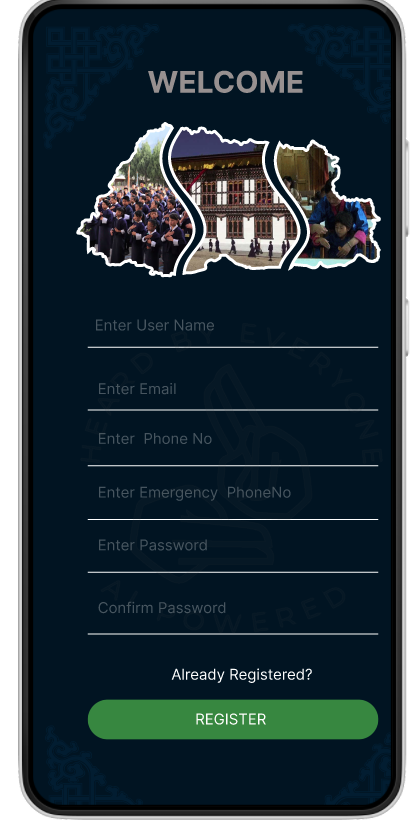
This is the landing page.



Here, users have the opportunity to either log in if they are already registered, using their designated username and password, or utilize the "forgot password" option if they require password reset assistance.

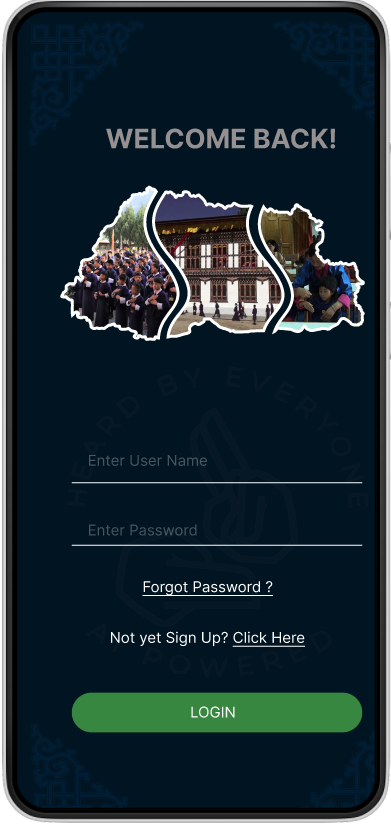


For new users, registration entails providing a username, email address, phone number, emergency contact phone number, and password. If a user is already registered, they will be promptly redirected to the login page.

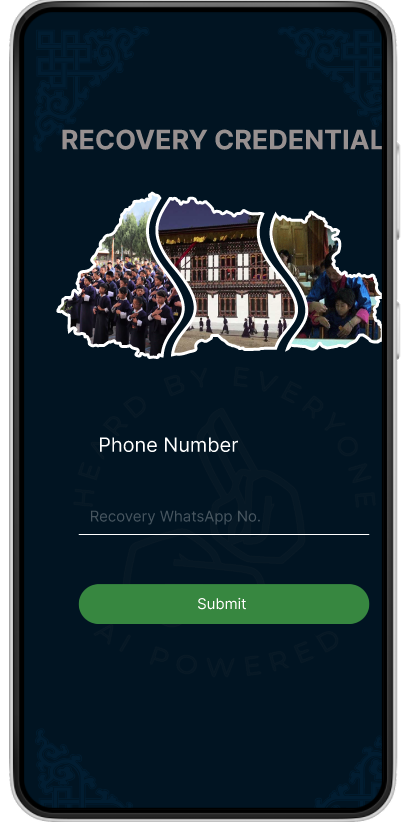


14.3 Reset password

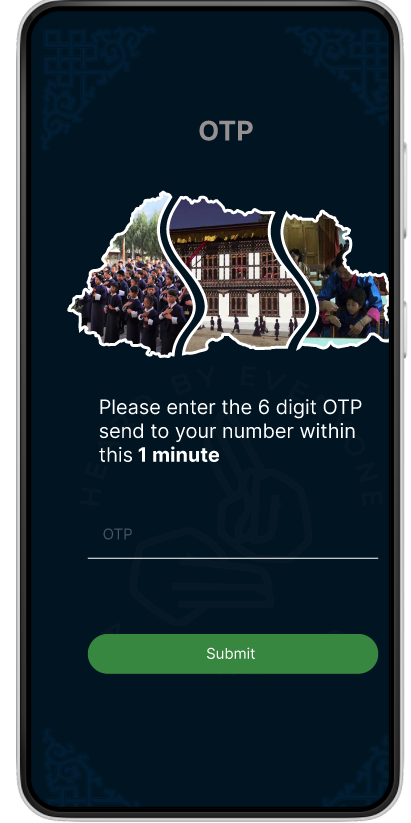
If the user forgets the password then click at the forgot password link.



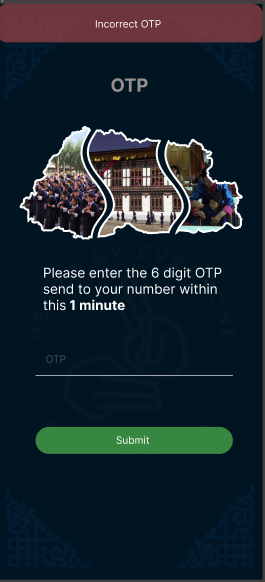
This is the reset password page where it is reset through phone number.



After submitting the phone number, the service will prompt you to request an OTP for password reset. This OTP is typically sent to your registered mobile phone number.

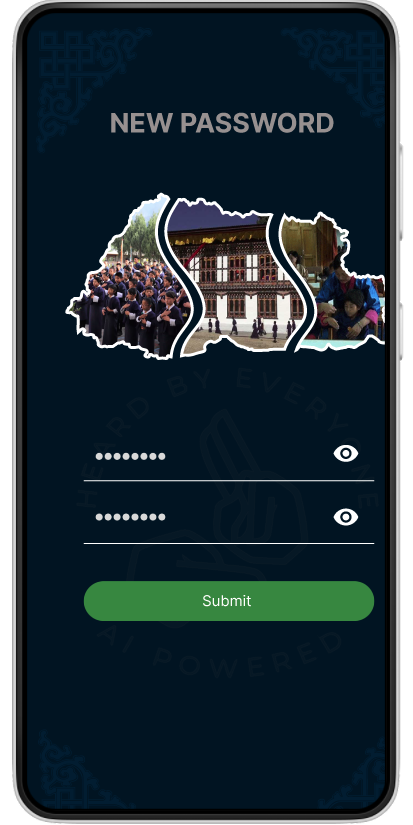


If the OTP send and OTP written is mismatched then an Incorrect message is prompt.

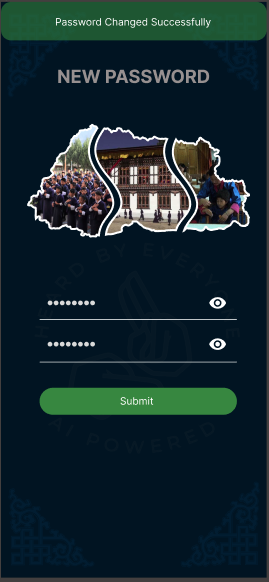


After successfully verifying the OTP, a new password for your account is given. Follow the instructions provided and choose a strong, secure password.

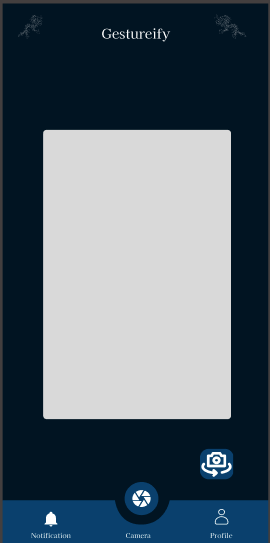
Services also ask users to confirm your new password by entering it again. Make sure to enter the same password in both fields.



Once a user sets a new password and confirms it, the user will receive a confirmation message indicating that the password has been successfully reset or updated.

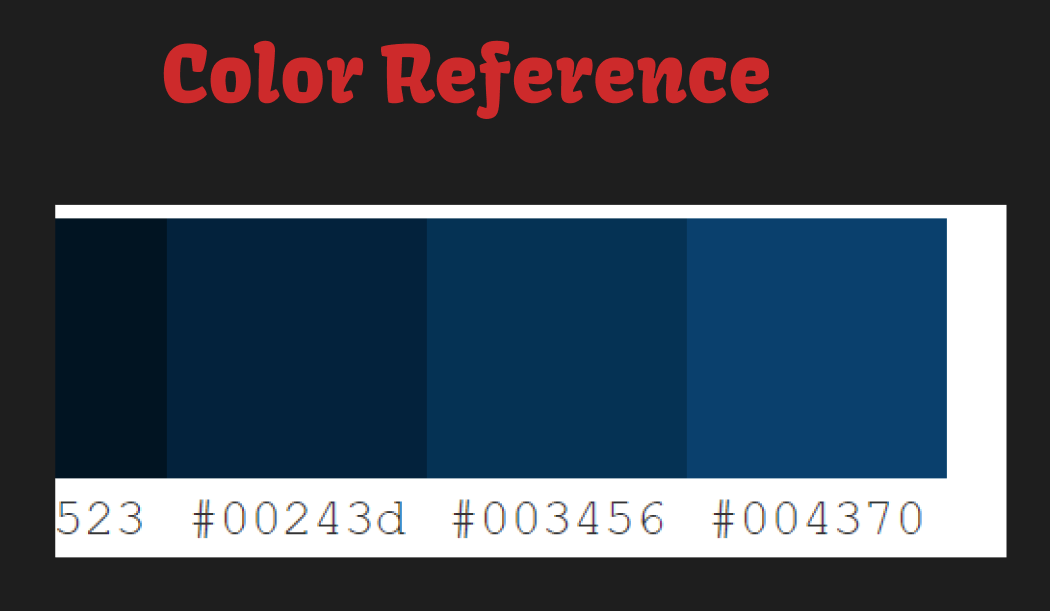


Redirect to the landing page to show gestures so that it is converted into audio or text.



14.4 Color Reference

This is the color reference used in the prototype and will be used in the application development.

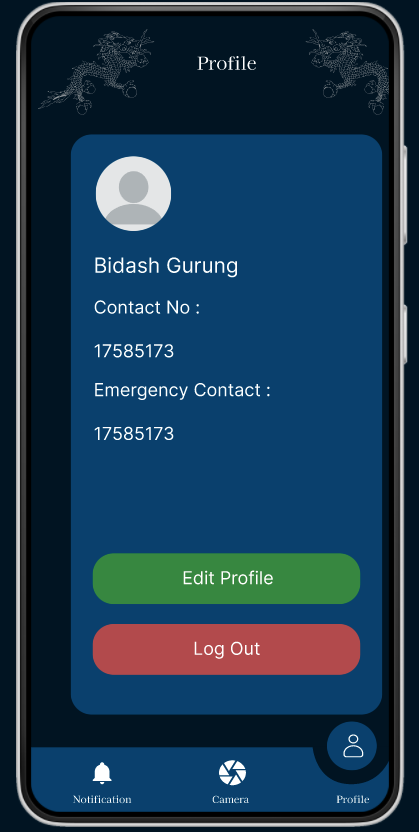


14.5 Functional Pages

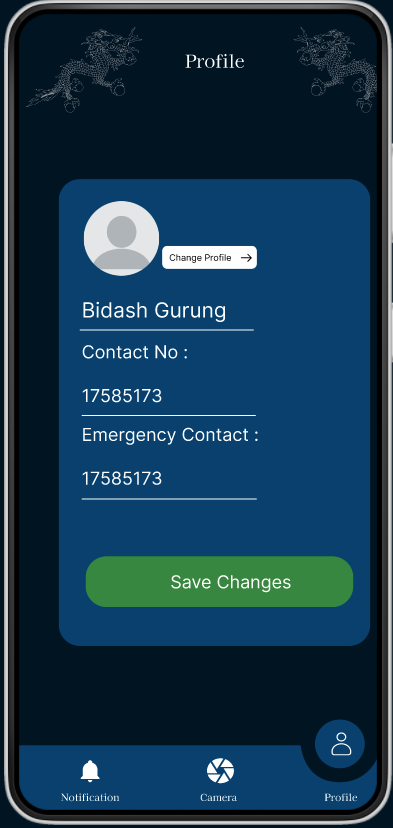
After signing up the landing page of the application to show our desired sign language to audio as well as text.



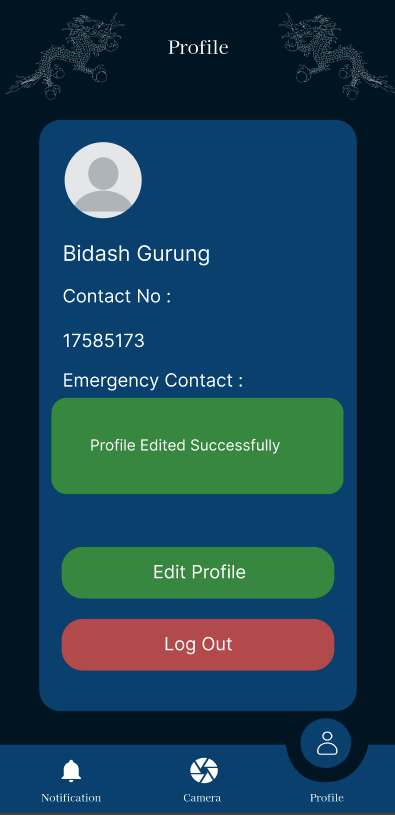
Here is the user profile page. Here, users can view and manage their personal information. The page features functionalities such as editing the profile and logging out.



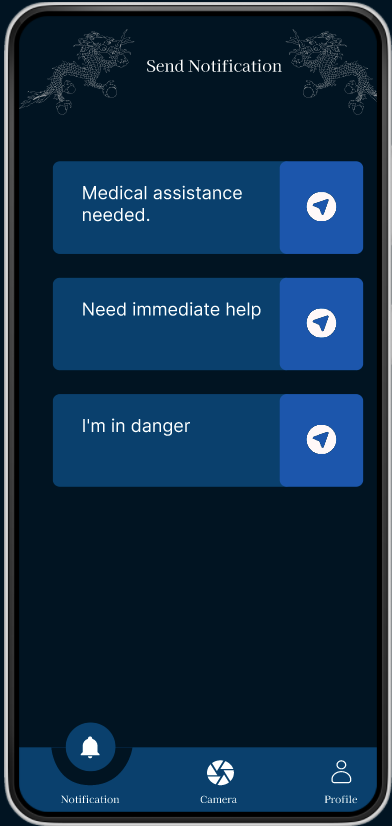
Upon selecting "Edit Profile," users can modify their personal details including their name, contact number, and emergency contact number. They can then save the changes they've made.



The app will show the message for successful update.

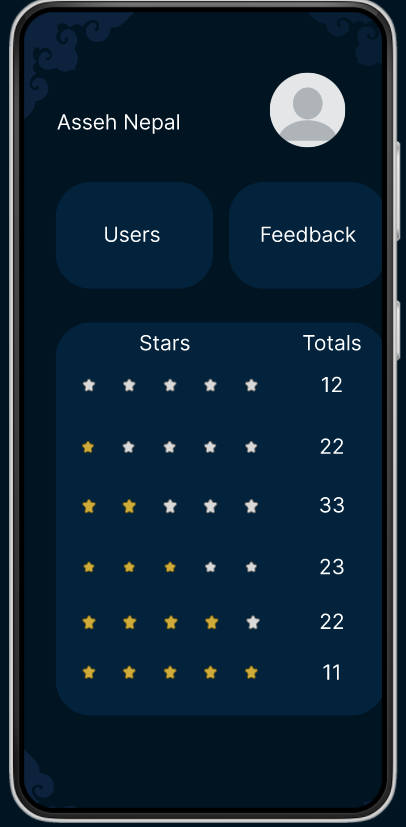


Here is the send Notification page where users can send notifications or messages to their designated emergency contacts during critical situations or when displaying emergency signs.

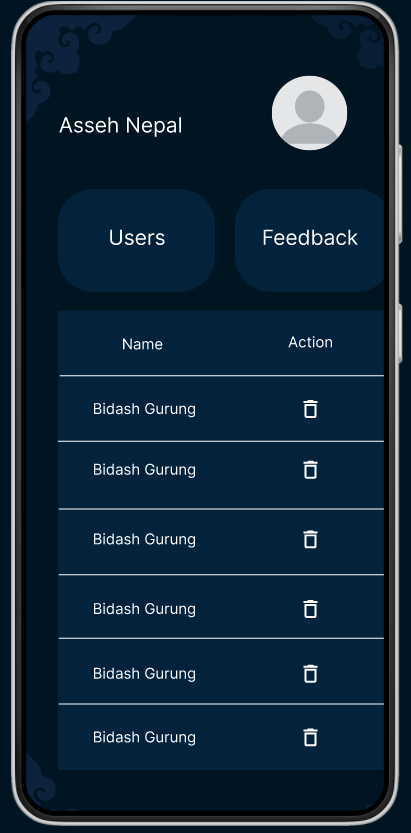


14.6 Admin dashboard

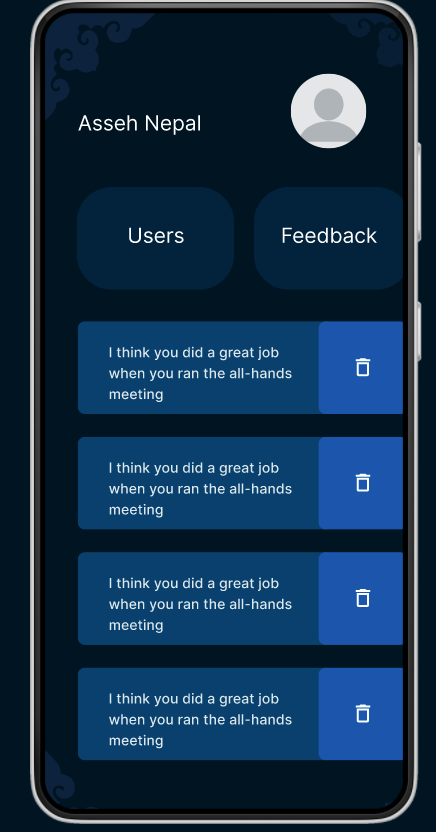
Here is the landing page of the admin dashboard where we can view star rating feedback and there are “Users” and “Feedback” links.



After clicking at “Users”, it will display the users.



If you click at “Feedback” then you can view the feedback.



PrototypeLink: <https://www.figma.com/file/kPcTl3eH7nd3xjXLqxjpJx/Prototyping-group7?type=design&node-id=114-2&mode=design&t=fmaYloOzS4Ah4Com-0>

# 15. Model Trained

1. RestNet50

ResNet50 is a deep convolutional neural network with 50 layers, designed to address the vanishing gradient problem in deep networks. It employs "residual connections" or shortcuts within its layers, enabling easier training and improved performance. This architecture allows the network to learn complex patterns efficiently, making it highly effective for tasks like image classification and object detection.

Preprocessing:

The preprocessing step for the video data is a step to make them as standardized as possible and prepare them for the next stage, which is the analysis of the machine learning algorithm. The preprocess\_videos\_in\_directory function opens each class directory sequentially and uses a created mapping to label the files. For each video file, it reads the video using OpenCV counts how many frames are there; and then processes the number of frames to a certain number of frames (default 100). From this layer, frames are reduced to 224 by 224 pixels and scaled to the range [0, 1]. If one video has fewer frames than the maximum allowed it will add blank frames to make the video more frames to the maximum; in the case where the one has many more frames than needed, then it will remove all the frames that are surplus. This results in a constant frame rate for all videos anchored on the number of frames of a specific video. The frames are then preprocessed before they are converted to a NumPy array, combined with its respective labels. This pipeline cleans up the data and ensures all frames are standardized to the same size and number, with normalization applied as needed.

Architecture Overview of ResNet50:

Conv1: 7x7, 64, stride 2

Max Pooling: 3x3, stride 2

Conv2\_x: 1x1, 64; 3x3, 64; 1x1, 256 (repeated 3 times)

Conv3\_x: 1x1, 128; 3x3, 128; 1x1, 512 (repeated 4 times)

Conv4\_x: 1x1, 256; 3x3, 256; 1x1, 1024 (repeated 6 times)

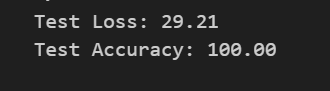
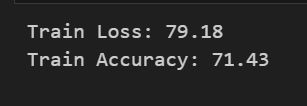
Conv5\_x: 1x1, 512; 3x3, 512; 1x1, 2048 (repeated 3 times)

Average Pooling: 7x7

Fully Connected Layer: 1000 (number of classes in ImageNet)

Training Results:

After completing the training process, the model's performance was evaluated based on training and testing datasets. The key metrics are as follows:



Training Loss: The model achieved a training loss of 79.18, indicating the average error on the training data.

Training Accuracy: The model reached a training accuracy of 71.43%, demonstrating its ability to classify the training data correctly.

Testing Loss: The testing loss was 29.21, reflecting the model's error on unseen data.

Testing Accuracy: The model achieved a testing accuracy of 100.00%, which measures its generalization capability to new, unseen data.

These metrics suggest that while the model was moderately successful in learning from the training data, as indicated by its training accuracy, its performance on the testing data was lower. This discrepancy points to potential overfitting, where the model may have learned the training data well but needed help to generalize to new data.

1. ResNet101

ResNet101 is a deep convolutional neural network with 101 layers, designed to solve the vanishing gradient problem in very deep networks. It uses "residual connections" or shortcuts within its layers to allow easier training and better performance. This architecture helps the network learn complex patterns efficiently, making it highly effective for tasks like image classification and object detection.

Preprocessing:

The preprocessing step is the same as RestNet50

Architecture Overview:

Conv1: 7x7, 64, stride 2

Max Pooling: 3x3, stride 2

Conv2\_x: 1x1, 64; 3x3, 64; 1x1, 256 (repeated 3 times)

Conv3\_x: 1x1, 128; 3x3, 128; 1x1, 512 (repeated 4 times)

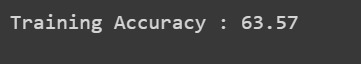
Conv4\_x: 1x1, 256; 3x3, 256; 1x1, 1024 (repeated 23 times)

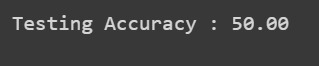
Conv5\_x: 1x1, 512; 3x3, 512; 1x1, 2048 (repeated 3 times)

Average Pooling: 7x7

Fully Connected Layer: 1000 (number of classes in ImageNet)

Training Results:





The model achieved a training accuracy of 63.57%, indicating its proficiency in classifying the training data.

However, its testing accuracy was 50.00%, showing its struggle to generalize to new, unseen data.

Code link-

<https://colab.research.google.com/drive/1aPspMSvDeyxQFLl0u2aOflwqBlmxZyLx>

1. DenseNet121

DenseNet121, short for Dense Convolutional Network with 121 layers, is a type of deep learning architecture designed for image classification tasks. In DenseNet121, each layer is connected to every other layer in a feed-forward fashion. This means that the feature maps of all preceding layers are used as inputs into each subsequent layer.

This dense connectivity ensures maximum information flow between layers and helps mitigate the vanishing gradient problem.

Preprocessing:  
The preprocessing step is the same as ResNet101

Architecture Overview:

Our DenseNet121 consists of multiple dense blocks, each containing several convolutional layers with dense connections, interspersed with transition layers. Here’s a high-level view of the DenseNet121 architecture:

Input Layer: Initial convolution with 64 filters of size 7×7, followed by a 3×3 max pooling.

Dense Block 1: Multiple layers connected densely.

Transition Layer 1: 1×1 convolution followed by 2×2 average pooling.

Dense Block 2: More layers with dense connections.

Transition Layer 2: 1×1 convolution followed by 2×2 average pooling.

Dense Block 3: Further dense layers.

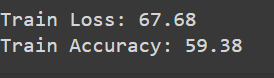
Transition Layer 3: 1×1 convolution followed by 2×2 average pooling.

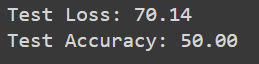
Dense Block 4: Final set of densely connected layers.

Output Layer: Global average pooling followed by a fully connected (dense) layer with softmax activation for classification.

Training Results:

After completing the training process, the model's performance was evaluated based on training and testing datasets. The key metrics are as follows:





Training Loss: The model achieved a training loss of 67.68, indicating the average error on the training data.

Training Accuracy: The model reached a training accuracy of 59.38%, demonstrating its ability to classify the training data correctly.

Testing Loss: The testing loss was 70.14, reflecting the model's error on unseen data.

Testing Accuracy: The model achieved a testing accuracy of 50.00%, which measures its generalization capability to new, unseen data.

These metrics suggest that while the model was moderately successful in learning from the training data, as indicated by its training accuracy, its performance on the testing data was lower. This discrepancy points to potential overfitting, where the model may have learned the training data well but struggled to generalize to new data.

Code link- <https://drive.google.com/file/d/1_keGMqg_CUhkblkb8WgKRMWV7c9TSQfO/view?usp=sharing>

1. SlowFast

The SlowFast model is a convolutional neural network architecture designed for video understanding tasks, particularly action recognition. It was introduced by Feichtenhofer et al. in their paper "SlowFast Networks for Video Recognition". Unlike traditional convolutional neural networks (CNNs) that process each frame uniformly, the SlowFast model leverages a dual-pathway architecture to handle both spatial and temporal aspects of video data more effectively.

Preprocessing:

First, the code scans through your dataset folders (both for training and testing) and lists out all the different classes (categories) of videos you have. It then counts the number of video files in each class to give you a sense of how your data is distributed.

Next, there's a class called PreprocessVideoDataset designed to handle the actual processing of the video files. This class goes through each video, reads it frame by frame, and converts the colors to a standard format. Each frame is resized to 224x224 pixels, which is a common size for image processing tasks.

Since videos can be of different lengths, the code makes sure each video has the same number of frames (up to 100 frames). If a video is shorter, it duplicates the last frame until it has 100 frames. This way, every video has a consistent size, making it easier to feed into a neural network.

The frames from each video are then stacked together into a 4D tensor (a multidimensional array) so that they can be processed by the model. Before the data is ready for training, several transformations are applied to enhance the dataset. These transformations include random cropping and resizing to add variability, adjusting the color properties, converting the frames to a format the model can understand, normalizing the pixel values, and randomly erasing parts of some frames to make the model more robust.

Architecture Overview:

The SlowFast model for video analysis works like this:

1. Input:

* Video sequences are fed into the SlowFast model, each containing a series of frames showing motion and spatial information over time.

1. Dual Pathways:

* The Slow Pathway: Processes spatial information slowly, focusing on detailed spatial features.
* The Fast Pathway: Handles temporal information quickly, emphasizing rapid motion and temporal dynamics.

1. Feature Extraction:

* Both pathways extract features independently, with the Slow Pathway capturing spatial details and the Fast Pathway capturing temporal dynamics.

1. Fusion:

* Features from both pathways are combined at multiple stages to leverage their strengths, enhancing overall understanding.
* Fusion methods include concatenation or addition to integrate spatial and temporal information effectively.

1. Network Depth and Width:

* The Slow Pathway is deeper and wider to capture spatial details comprehensively.
* The Fast Pathway is shallower and narrower to handle temporal information efficiently.

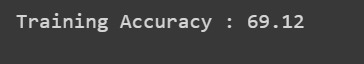
1. Classification:

* Fused features are passed to a classification layer to predict action labels or categories for the input video sequence.

1. Training:

* The SlowFast model is trained on labeled video datasets, learning to extract features and make accurate predictions for action recognition tasks.

Training Results:





The model's training process yielded a training accuracy of 69.12%, indicating its effectiveness in correctly classifying the training data. This suggests that the model has learned to recognize patterns and features within the training set.

However, the testing accuracy, which stands at 63.52%, reflects the model's ability to generalize its knowledge to new, unseen data. While there's a noticeable drop from the training accuracy, the testing accuracy shows that the model can still perform reasonably well on unfamiliar data, though there might be room for improvement to enhance its generalization capabilities.

1. 3D Convolution

3D convolution, also known as spatiotemporal convolution, extends the concept of 2D convolution to three dimensions. While 2D convolution operates on images represented by 2D grids of pixels, 3D convolution operates on volumes or sequences of images, such as video data or medical imaging volumes, where the third dimension represents time or depth.

Preprocessing:

The `PreprocessVideo` class in your code takes several important preprocessing steps to prepare video data for deep learning. First, it scans the specified folder to identify subfolders representing different classes, mapping class names to integer labels, and collecting paths to all video files. For each video, frames are extracted using OpenCV, with each frame converted from BGR to RGB and resized to a standard 224x224 pixels. To handle variable video lengths, the class ensures each video has a consistent number of frames by padding shorter videos with duplicates of the last frame until they reach a set maximum. These frames are then converted to PyTorch tensors and stacked into a 4D tensor, with dimensions permuted to match the expected input format for deep learning models. This process ensures that all videos are uniformly processed and ready for effective model training.

Architecture Overview:

Input: Instead of single images, we use volumes of data, like videos or stacks of medical images.

Convolutional Layers: These layers scan through the data in three dimensions (width, height, and depth). They detect features like edges, textures, or motion across both space and time.

Pooling Layers: Similar to 2D CNNs, pooling layers reduce the size of the data by summarizing regions. This helps in capturing the most important information while reducing computational load.

Activation Functions: Functions like ReLU are applied to introduce non-linearities, enabling the network to learn complex patterns.

Normalization and Regularization: Techniques like batch normalization and dropout help stabilize training and prevent overfitting.

Fully Connected Layers: At the end, we have fully connected layers that map the extracted features to the desired output, like predicting categories or quantities.

Output: The final layer produces the network's output, which could be a classification label, a set of coordinates, or any other relevant output for the task at hand.

Training Result:





The model achieved a training accuracy of 59.26%, indicating its capability to classify the training data correctly.

Its testing accuracy was 62.05%, demonstrating its ability to generalize and perform well on new, unseen data.

1. VGG-19

VGG19 is one of the deep learning convection layer Neural Networks where this network is simple but very effective for many applications in image classification. It was introduced in a paper by Karen Simonyan and Andrew Zisserman from the Visual Geometry Group at the University of Oxford in the very deep convolutional networks for large-scale image recognition.

It was developed by the Visual Geometry Group at the University of Oxford and presented in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition” by Karen Simonyan and Andrew Zisserman. In detail, VGG19 is derived from VGG network of 16, which comprises of 19 layers and hence the name.

Preprocessing:

This preparatory function takes video data of an input format and prepares it in a format consumable for machine learning models through resizing and normalization of frames. To do so, it analyses video files located in subdirectories treated as classes, and assigns each of them a single index number. Specifically for the two video files, the function extracts each frame of a video file, converts it to a size of 224X224 pixels, and changes the pixel intensities to float values ranging from 0 to 1. These preprocessed frames are then elaborated into a list together with a class label to go with it. This makes it easy for Mile groupId to format all the video data into the required format for training and other model evaluations. The function returns two lists containing the movie's frames that the classifier and their corresponding classes have processed.

Architecture Overview:

Input Shape: (64, 64, 3)

1. Base Model: Pre-trained VGG19 (include\_top=False, input\_shape=(64, 64, 3))

* Block 1: 2 convolutional layers (64 filters each) + max-pooling
* Block 2: 2 convolutional layers (128 filters each) + max-pooling
* Block 3: 4 convolutional layers (256 filters each) + max-pooling
* Block 4: 4 convolutional layers (512 filters each) + max-pooling
* Block 5: 4 convolutional layers (512 filters each) + max-pooling

1. Custom Classification Layers:

* Flatten Layer
* Dense Layer: 128 units, ReLU activation
* Output Layer: 2 units, softmax activation (for classification)

1. Compilation:

* Optimizer: Adam
* Loss: Categorical cross-entropy
* Metrics: Accuracy

1. Sequence Padding:

* Pad sequences to uniform length using maximum sequence length in the training data

Training Results: Since the model is really heavy, we got runtime errors since the 1st epoch.

Code link:

<https://colab.research.google.com/drive/1xHCHq_uPVOqtj3PyvixXdbgZexKqDK70#scrollTo=_JZsSJXOmBsq>

1. LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to handle the problem of learning long-term dependencies, which is common in sequence prediction tasks. The LSTM module is composed of several key components that work together to control the flow of information through the network.

Preprocessing:

For the preprocessing it includes only keypoints of human body which is stored in numpy array. We collected the data using openCv which stored BGR color, so we converted the color to RGB which is accepted by the mediapipe, a framework which provides cross-platform, customizable, and high-performance solutions for various machine learning tasks, particularly those involving computer vision. It offers pre-trained models and tools for real-time, multimodal perception, making it an excellent choice for building applications that require real-time image and video analysis.

Keypoints extraction:

We used a system capable of detecting various points on a person’s body, face, and hands from an image or video. These points were known as landmarks. The system was able to detect four types of landmarks: pose landmarks, such as the position of joints and limbs, face landmarks, such as the positions of facial features, and landmarks for the left and right hands.

Sometimes, the system might not have detected landmarks for one or more parts (pose, face, left hand, right hand). In such cases, we needed a way to handle missing data. If landmarks were not detected, we created a placeholder filled with zeros to represent the absence of data.

When landmarks were detected, we extracted specific details for each point. For pose landmarks, we got the 3D coordinates (x, y, z) and an additional visibility score, which told us how confident the system was about the detection of each point. For face and hand landmarks, we only extracted the 3D coordinates (x, y, z).

Then, we collected all the extracted points into a single, long list for each type of landmark (pose, face, left hand, right hand). This made the data easier to manage and use. For pose landmarks, this list included 132 values, covering the positions and visibility scores for 33 points. For face landmarks, the list included 1404 values, covering the positions of 468 points. And for each hand, the list included 63 values, covering the positions of 21 points.



By organizing the data into long lists (or arrays), we can use them as input for machine learning models. These models often require fixed-size input, so having a consistent format for the landmark data is important.

This structured data can be used for various applications, such as recognizing gestures, estimating poses, or analyzing facial expressions.

Architecture Overview:

1. Input Shape: (35, 1662) (sequence length, number of features per frame)
2. LSTM Layers:

* LSTM(64, input\_shape=(35, 1662), return\_sequences=True): Adds an LSTM layer with 64 units, expecting sequences of length 35 with 1662 features per time step. The parameter return\_sequences=True ensures the LSTM layer returns sequences rather than single outputs.
* Dropout(0.01): Dropout layer with a regularization rate of 0.01 to prevent overfitting.
* Additional LSTM Layers:

LSTM(64, return\_sequences=True): Adds another LSTM layer with 64 units and return\_sequences=True to maintain sequence output.

Dropout(0.01): Dropout layer for regularization.

LSTM(64): Adds a final LSTM layer with 64 units. This layer does not need to return sequences since it's followed by dense layers.

Dropout(0.01): Dropout layer for regularization.

1. Dense Layers:

* Dense(256, activation='relu'): Fully connected dense layer with 256 units and ReLU activation function for feature extraction.
* Dropout(0.01): Dropout layer for regularization.
* Dense(128, activation='relu'): Dense layer with 128 units and ReLU activation.
* Dropout(0.01): Dropout layer for regularization.
* Dense(64, activation='relu'): Dense layer with 64 units and ReLU activation.

1. Output Layer:

* Dense(actions.shape[0], activation='softmax'): Dense layer with units equal to the number of classes (actions), using softmax activation for multi-class classification.

1. Compilation:

* Optimizer: Adam optimizer is used for model optimization.
* Loss: Categorical cross-entropy loss function is employed, suitable for multi-class classification tasks.
* Metrics: The accuracy metric is used to evaluate model performance.

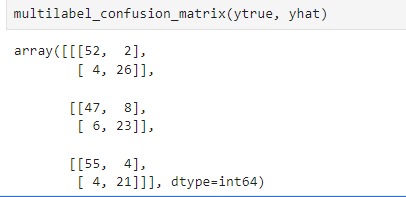
1. Sequence Padding:

* Sequences are padded to a uniform length using the maximum sequence length in the training data to ensure consistent input dimensions.

Training results:

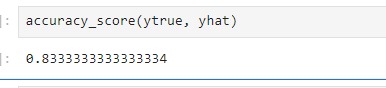
* Confusion matrix

We trained for 3 classes. The provided confusion matrix is a 3D array, indicating that it represents confusion matrices for multiple classes



* Test Accuracy:

The model achieved a testing accuracy of 83.33%, which measures its generalization capability to new, unseen data.



* Train Accuracy:

The model reached a training accuracy of 84.53%, demonstrating its ability to classify the training data correctly. The model has a loss of 45.85%, a Validation Accuracy of 83.74%, and a validation loss of 39.00%.



1. GRU

Preprocessing:

The preprocessing step is the same as LSTM.

Architecture Overview:

1. Input Shape: (35, 1662) (sequence length, number of features per frame)
2. GRU Layers:

* GRU(64, input\_shape=(35, 1662), return\_sequences=True): Adds a GRU layer with 64 units, expecting sequences of length 35 with 1662 features per time step. The parameter return\_sequences=True ensures the GRU layer returns sequences rather than single outputs.
* Dropout(0.01): Dropout layer with a regularization rate of 0.01 to prevent overfitting.
* Additional GRU Layers:

GRU(64, return\_sequences=True): Adds another GRU layer with 64 units and return\_sequences=True to maintain sequence output.

Dropout(0.01): Dropout layer for regularization.

GRU(64): Adds a final GRU layer with 64 units. This layer does not need to return sequences since it's followed by dense layers.

Dropout(0.01): Dropout layer for regularization.

1. Dense Layers:

* Dense(256, activation='relu'): Fully connected dense layer with 256 units and ReLU activation function for feature extraction.
* Dropout(0.01): Dropout layer for regularization.
* Dense(128, activation='relu'): Dense layer with 128 units and ReLU activation.
* Dropout(0.01): Dropout layer for regularization.
* Dense(64, activation='relu'): Dense layer with 64 units and ReLU activation.

1. Output Layer:

* Dense(actions.shape[0], activation='softmax'): Dense layer with units equal to the number of classes (actions), using softmax activation for multi-class classification.

1. Compilation:

* Optimizer: Adam optimizer is used for model optimization.
* Loss: Categorical cross-entropy loss function is employed, suitable for multi-class classification tasks.
* Metrics: Accuracy metric is used to evaluate model performance.

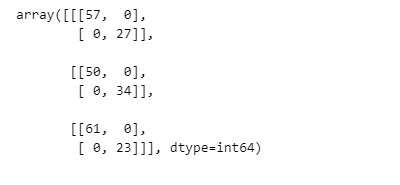
1. Sequence Padding:

* Sequences are padded to a uniform length using the maximum sequence length in the training data to ensure consistent input dimensions.

Training Results:

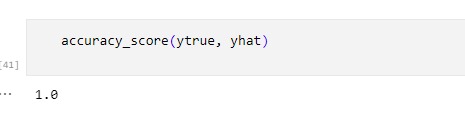
* Confusion matrix

We trained for 3 classes. The provided confusion matrix is a 3D array, indicating that it represents confusion matrices for multiple classes



* Testing Accuracy:

The model achieved a testing accuracy of 100.00%, which measures its generalization capability to new, unseen data



* Training Accuracy:

The model reached a training accuracy of 100.00%, demonstrating its ability to correctly classify the training data. The model has loss of 2.66%, Validation Accuracy of 100.00%, and validation loss of 2.30%.



Accuracy Comparision

| Sl no | Model | Training Accuracy | Testing Accuracy |
| --- | --- | --- | --- |
| 1 | ResNet50 | 71.43 | 100 |
| 2 | ResNet 101 | 63.57 | 50 |
| 3 | DenseNet121 | 59.38 | 50 |
| 4 | SlowFast | 69.12 | 63.56 |
| 5 | 3D Convolution | 59.26 | 62.05 |
| 6 | VGG -19 | - | - |
| 7 | LSTM | 84.53 | 83.33 |
| 8 | GRU | 100 | 100 |

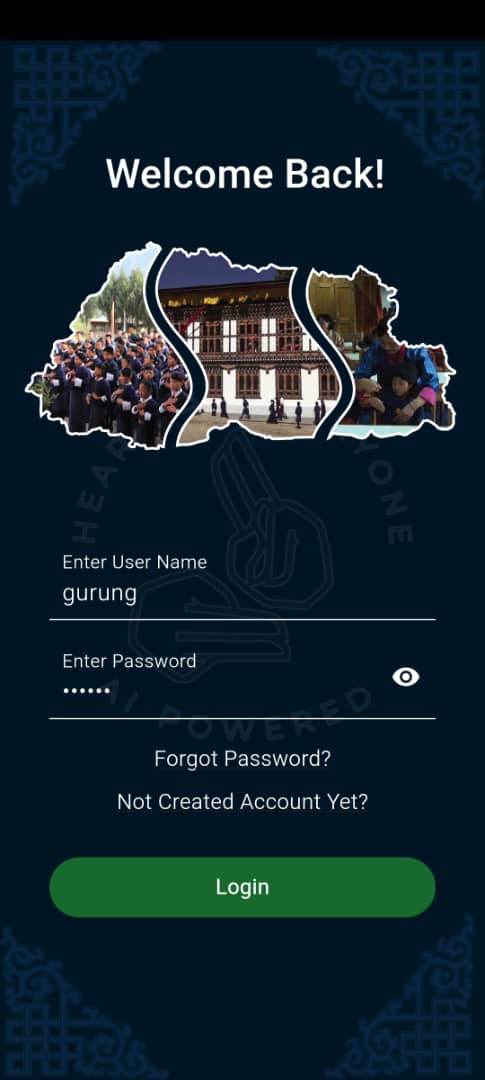
# 16. User Interface

## 16.1 User

SignIn

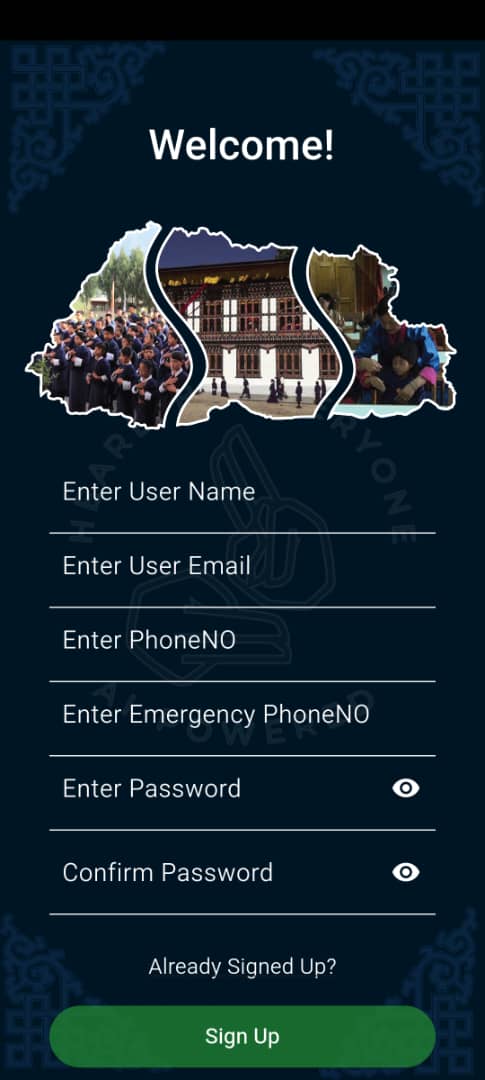
* Users can log in if they have previously registered using their assigned username and password.

If they are new users, they can register.



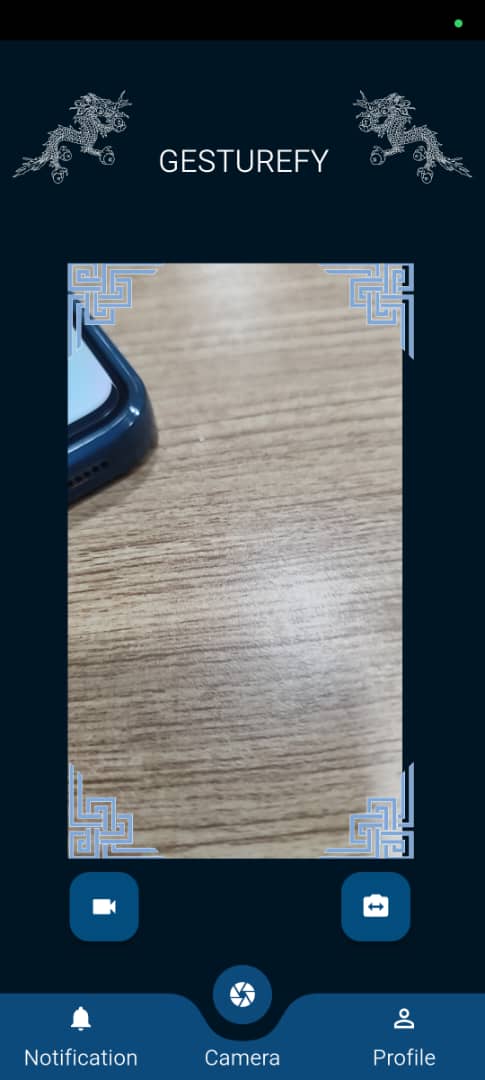
SignUp

* When registering, new users must enter their password, emergency contact number, phone number, email address, and username. A user will be quickly forwarded to the login page if they have already registered.



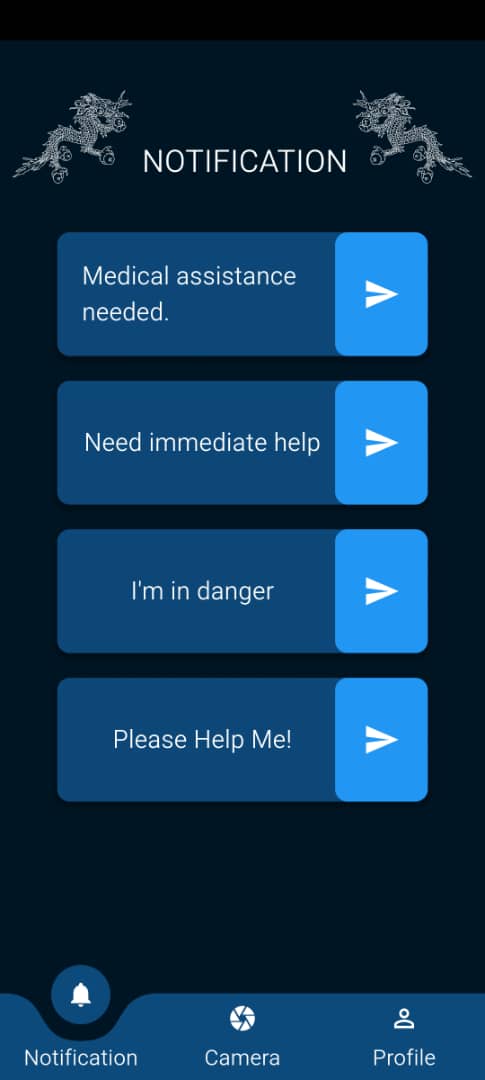
Gesturefy

Redirect to the landing page to show gestures so that it is converted into audio or text.



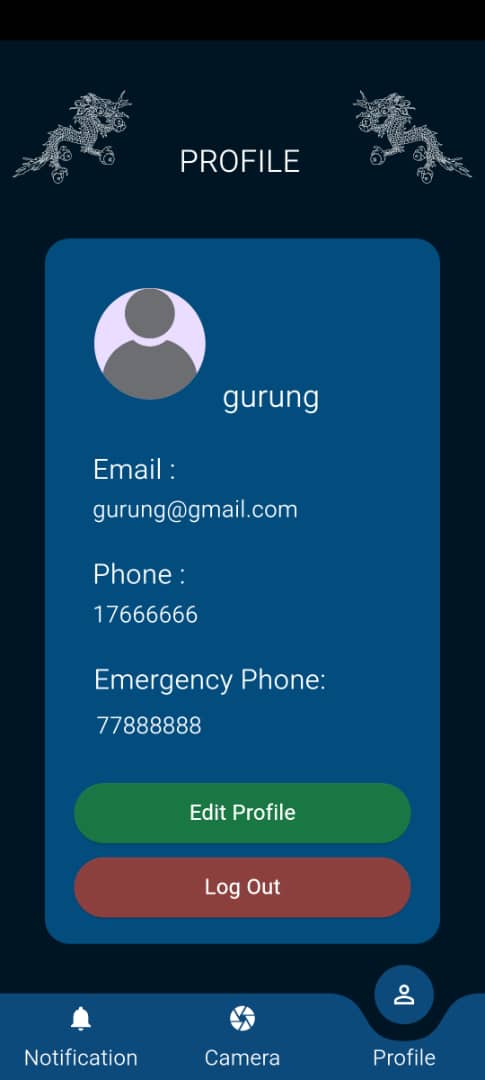
Notification

* This is the send notification page, where users can communicate with their emergency contacts by sending messages or alerts in case of an emergency or when emergency indications are displayed.



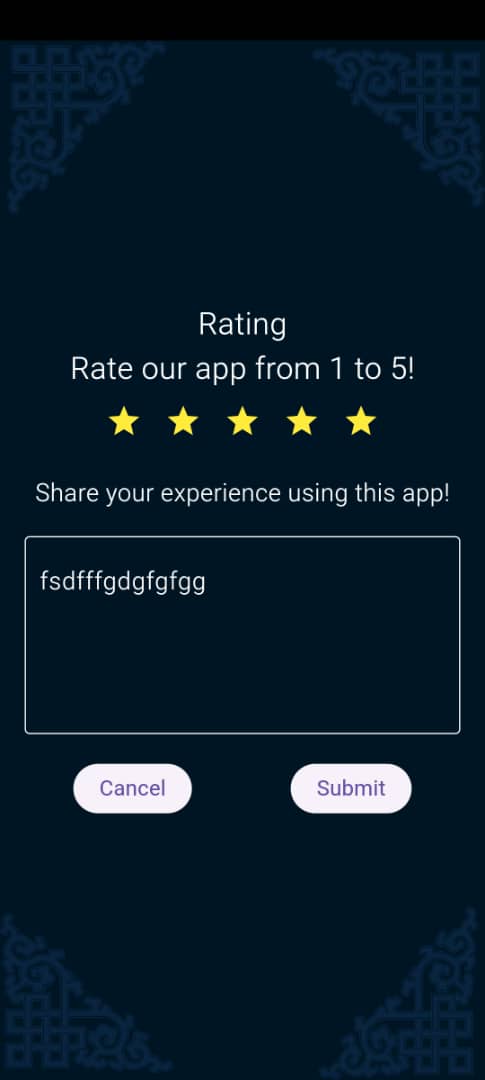
Profile

* The user profile page is shown here. Users are able to access and control their personal data here. The page has options like logging out and modifying the profile.



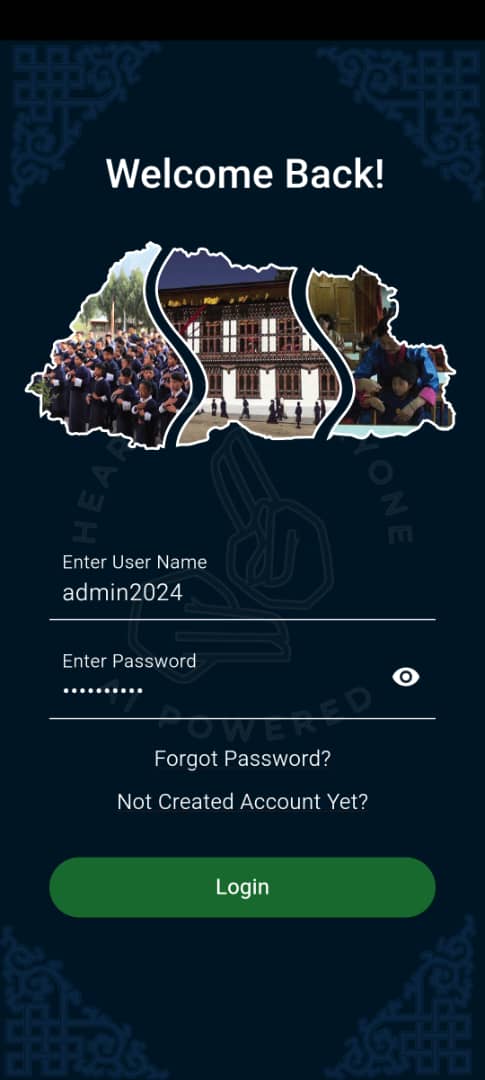
Star Rating and Feedback

* Users can rate the application as well as give feedback



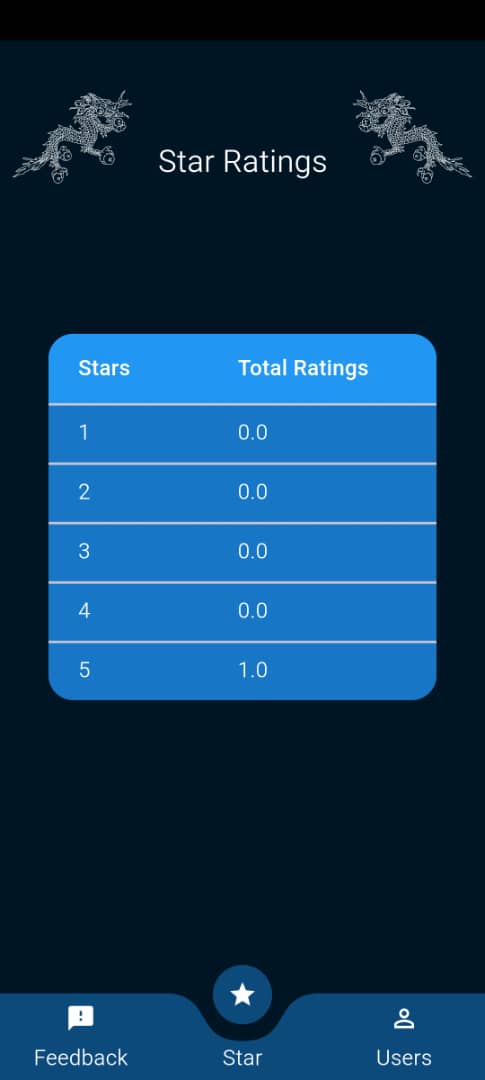
## 16.2 Admin

Login - admin can login using the credentials



Star rating

* This is the admin dashboard's landing page, where we can see user and feedback ratings in the form of stars.



Users

It will show all the users after you click on "Users."

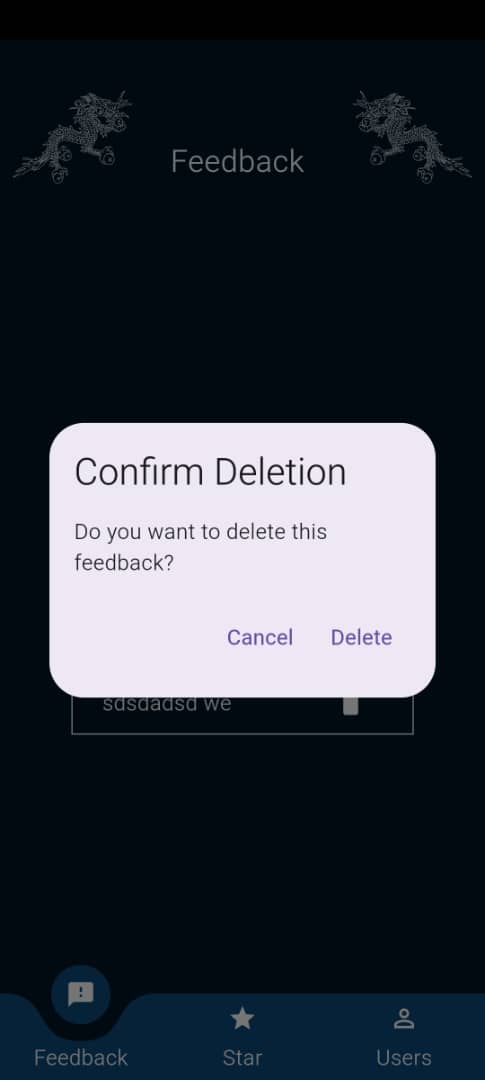


Feedback

It will show the feedback given by users



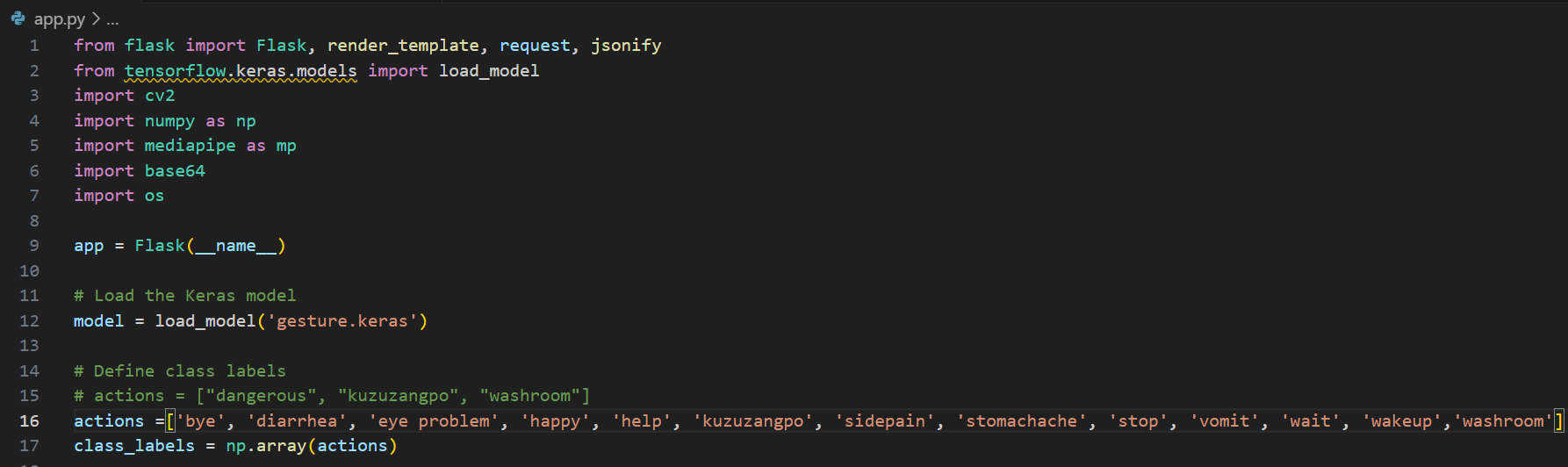
And the alert message is shown, if we click the delete button.



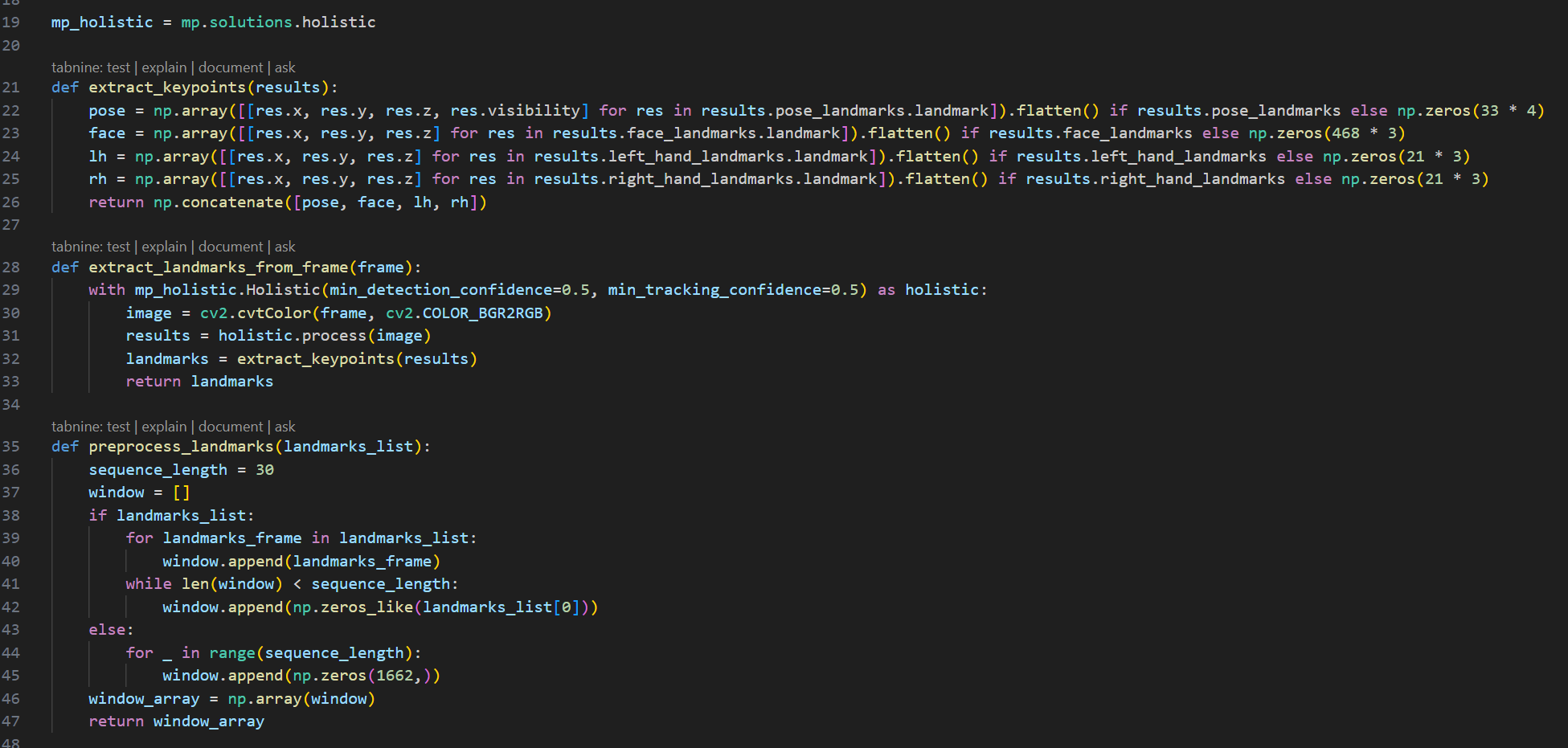
# 17. Model Integration

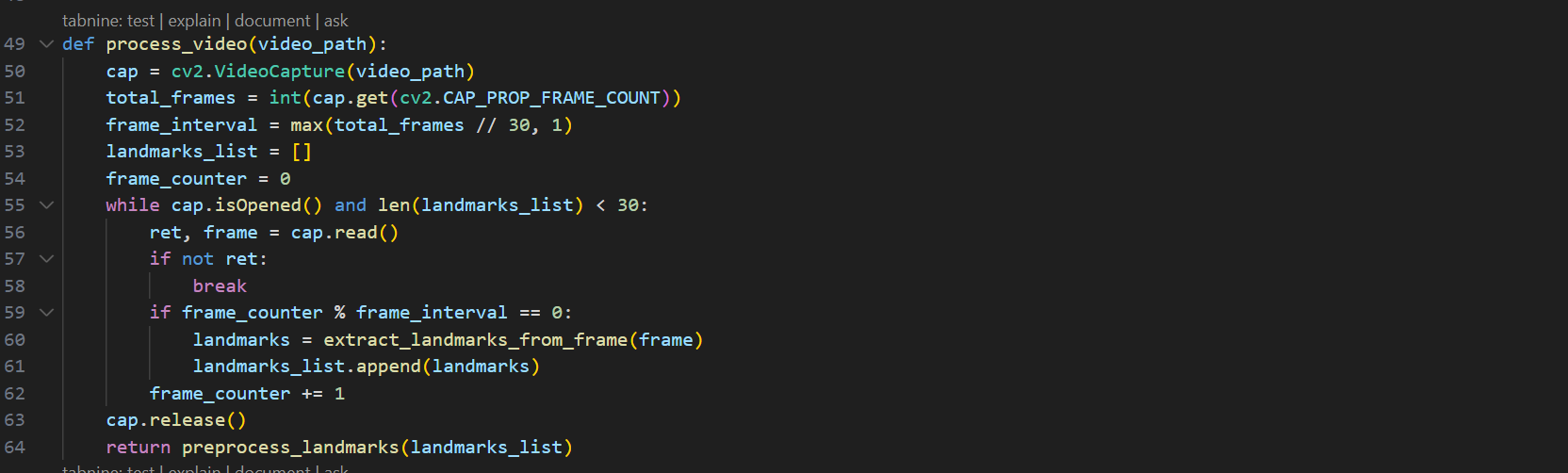
We have integrated our model with the flask app. After running the flask app we get the URL which we use as API for our mobile application.

The below screenshot shows the imports of the necessary library, loading the model, and then defining the necessary classes.



The below screenshot shows the preprocessing steps for the video file that we send form our mobile application

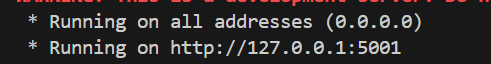




The below screenshot shows the main flask application on which running the app will give the URL that we use as API in our Flutter mobile application



The below screenshot shows the URL link generated from the flask application.



The below screenshot shows the calling of API URL that is been generated from the flask app. In flask app we are been given with the 127.0.0.1:5001 but in the Flutter application we use 10.9.90.33:5001. It is because we need the IP address of our local network. So from this URl, we pass the video recorded from our mobile app and make predictions.



# 18. References

Contributor, U. U. (2022, July 8). Deaf students in Bhutan create country’s first sign language. *Forbes*. <https://www.forbes.com/sites/unicefusa/2022/07/08/deaf-students-in-bhutan-create-countrys-first-sign-language/?sh=70cd614049e1>

*Deaf students construct first sign language for Bhutan*. (n.d.). UNICEF Bhutan. <https://www.unicef.org/bhutan/stories/deaf-students-construct-first-sign-language-bhutan>

Hand Talk. (2023, July 1). *Hand Talk: your website accessible in ASL*. Hand Talk - Learn ASL Today. <https://www.handtalk.me/en/>

Venkat. (2017, March 30). *“Sign With Robert” channel on GIPHY teaches ASL through GIFs - Assistive Technology blog*. Assistive Technology Blog. <https://assistivetechnologyblog.com/2017/03/sign-with-robert-channel-giphy-teaches-asl-gifs.html#:~:text=The%20Sign%20with%20Robert%20gifs,Social%20Media%2C%20etc>.

THANK YOU!