School of Computer Science

Faculty of Science and Engineering

University Of Nottingham, Malaysia

UG FINAL YEAR DISSERTATION REPORT

- A Real-Time Multimodal Communication System Integrating ASL Gesture Recognition and Facial Emotion Analysis for Inclusive Digital Interaction -

Mohammed Ismail Moh'd Abusirdaneh

20398888

Dr. Yasir Hafeez

May, 2025

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE WITH AI (HONS)

THE UNIVERSITY OF NOTTINGHAM

- A Real-Time Multimodal Communication System Integrating ASL Gesture Recognition and Facial Emotion Analysis for Inclusive Digital Interaction -

Submitted in May 2025, in partial fulfillment of the conditions of the award of the degrees **B.Sc Computer Science with AI (Hons).**

- Mohammed Ismail Moh'd Abusirdaneh -

School of Computer Science

Faculty of Science and Engineering

University of Nottingham, Malaysia

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature *Abusirdaneh*

Date 04/05/2025

Abstract

The project handles Ain Portal, which is a multimodal education interface that intends to support Deaf and Hard-of-Hearing (DHH) students with simultaneous real-time American Sign Language (ASL) recognition and feedback regarding emotion awareness. The system combines gesture detection, facial emotion recognition, and sentiment analysis based on a single interactive platform consisting of MediaPipe, TensorFlow Lite, and Django. A lightweight classifier processes static hand keypoints brought from webcam input; while a CNN-based model identifies emotions from facial expressions. Fusion takes place at the interpretation layer where the results of the modalities come together to detect mismatches between signed content and affective state (e.g., those cases when positive sentences are accompanied by sad expressions.)

The solution was deployed as a local-first standalone application with low-end hardware requirements, ensuring both offline compatibility and real-time operations on consumer devices. A custom training pipeline was developed for both gesture and emotion classifiers, with the ASL model achieving 86.1% test accuracy, and the emotion model aligned in five-class output remapped from FER-2013 data. Live-user interaction logs, sentiment validation, and report export, thereby allowing for analytics of the session.

Through modular architecture, user-friendly interface design, and low-latency performance, Ain Portal showcased the potential of accessible, emotionally-aware ASL learning tools that could be the groundwork for future expansion into dynamic sign recognition, cloud-based deployment, and broader educational integration.

Acknowledgment

All praise is due to Allah, the Most Gracious and Most Merciful. By His mercy and will, I received the strength and clarity to be perseverant enough to complete this final year project. On top of this, I thank Allah for his guidance throughout this academic journey as well as His healing after the spinal surgery I underwent during this period.

My heart is filled with gratitude towards my family for their constant love, support, and patience. They have stood by me through all this project and offer attentive encouragement throughout my convalescence. Their presence and prayers were strength and comfort in my time of trial. I also extend my heartfelt appreciation to my supervisor Dr. Yasir Hafeez for the continuous support, insightful feedback, and intellectual guidance. His mentorship was both attentive and compassionate, striking a balance that inspired me to improve continuously. His encouragement and flexibility during my period of recovery were especially appreciated, making it possible to stay on track despite setbacks.

Lastly, I am thankful to the University of Nottingham for creating an environment of academic excellence and providing the resources and facilities that made this work possible.

Table of Contents

Abstract **3**
Acknowledgment **4**
Abbreviations **8**
1. Introduction **9**
1.1 Background 9
1.2 Problem Statement 9
1.2.1 Communication Barriers to the DHH 9
1.2.2 Lack of Emotional Awareness in Gesture Systems 10
1.3 Objectives 11
1.3.1 Core Objective 11
1.3.2 Technical Objectives 11
1.4 Scope of the Project 11
2. Literature Review **12**
2.1 Foundations of Sign Language Recognition 12
2.1.1 Early Systems: Template-Based and Sensor-Based Models 12
2.1.2 CNN-Based Static ASL Detection 12
2.1.3 MediaPipe and Landmark-Based Approaches 13
2.2 Dynamic Sign Language Recognition and Temporal Modeling 14
2.2.1 Deep Learning Approaches for Sequential Gesture Processing 14
2.2.2 Constraints in Deployment, Coarticulation, and Data Generalization 15
2.3 Emotion Recognition and Affective Computing in Assistive Systems 15
2.3.1 Evolution of FER 15
2.3.3 Real-Time Deployment and Fairness 16
2.4 Multimodal Integration and Fusion Models for Communication Systems 16
2.4.1 Rationale for Multimodal Communication 16
2.4.2 Fusion Strategies and Architectures 17
2.5 Sign Language in Educational Technology and Accessibility Tools 17
2.5.1 Importance of Inclusive Educational Platforms 17
2.5.2 ASL and Emotion Integration for Engagement 18
2.5.3 Logging, Analytics, and Modular Architectures 18
3. Methodology **19**
3.1 Methodology Overview 19
3.2 System Architecture 20
3.2.1 Input Acquisition Layer 20
3.2.2 Gesture Detection 20
3.2.3 Emotion Detection 20
3.2.4 Interpretation and Validation Layer 21

3.2.5 User Interface Layer..... 21
3.2.6 Scalability and Modularity.....21
3.3 Deployment Choices.....21
3.3.1 Modularity and Scalability.....21
3.3.2 Standalone Operation and Portability..... 22
3.3.3 Deployment Summary..... 22
3.4 Setup Workflow..... 22
3.5 Ain Portal Folder Structure.....23
3.5.1 The Accounts Folder.....24
3.5.3 adminDashboard Folder.....25
3.5.4 The media Folder..... 25
3.5.5 The meeting Folder.....26
3.5.6 The API Folder..... 27
3.5.6.1 ASL Detection (asl_detector.py).....27
3.5.6.2 Emotion Detection (emotion_detector.py).....28
3.5.6.3 Unified Processing Controller and Supporting Utilities (full_pipeline.py).... 29
3.5.7 The quiz folder.....30
3.5.8 The static folder..... 31
3.5.9 The teacher folder..... 31
3.6 Selection of Framework Tool.....33
3.6.1 MediaPipe for Hand Tracking.....33
3.6.2 TensorFlow Lite and Keras for Gesture and Emotion Models..... 33
3.6.3 OpenCV for Video Processing and GUI.....33
3.6.4 Django Framework for Backend and API Management.....33
3.6.5 Hugging Face Transformers for Sentiment Analysis.....34
3.6.6 Summary of Framework Integration.....34
4. Results and Evaluation.....34
4.1 ASL Classifier Performance..... 34
Table 1 – Classification Metrics Summary (Selected Letters).....37
4.2 Emotion Classifier Performance.....37
4.3 Sentence and Emotion Integration.....39
4.4 Ain Portal User Interaction..... 40
5. Project Management.....40
5.1 Personal Reflections and Technical Learnings..... 41
5.2 Limitations and Out-of-Scope Considerations.....42
5.3 Future Enhancements.....42
6. Conclusion..... 43
7. References.....44

8. Appendix..... 48
Appendix A..... 48
A.1 ASL Fingerspelling (A–Z): Standard Hand Gestures Used by the DHH Community... 48
A.2 FER-2013 Dataset Overview for Facial Emotion Recognition..... 49
A.3 Haar Cascade Feature Visualizations for Face Detection..... 49
Appendix B..... 50
B.1 Full System Architecture Flowchart for ASL, Emotion, and Sentiment Integration.. 50
B.2 ASL Alphabet Classification Metrics (Precision, Recall, F1-Score per Letter)..... 51
B.3: Confusion Matrix for ASL Gesture Classification..... 52
B.4 Confusion Matrix of Emotion Classifier (FER-2013, 5-Class Output)..... 53
Appendix C..... 54
C.1 The Home Page..... 54
C.2 The Login Page..... 54
C.2.1 ReCAPTCHA by Google..... 54
C.3 The Register Page..... 55
C.4 The Student Profile Page..... 55
C.5 The Student Dashboard Page..... 55
C.6 The Teacher Profile Page..... 56
C.7 The Teacher Dashboard Page..... 56
C.8 The Teacher Courses Page..... 56
C.9 The Admin Page..... 57
C.10 The meeting room Page..... 57
C.10.1 The Api View..... 58
C.10.2 The Room View..... 58
C.11 The Quiz..... 59
C.11.1 Teacher’s Quiz Creation..... 59
C.11.2 Teachers’s Question Creation..... 59
C.11.1 Student’s Quiz Solving..... 60
C.11.2 Student’s Past Quizzes..... 60

Abbreviations

Abbreviations	Full Form
3D CNN	3D Convolutional Neural Network
ASL	American Sign Language
AWS	Amazon Web Services
CNN	Convolutional Neural Network
DHH	Deaf and Hard-of-Hearing
DSLR	Dynamic Sign Language Recognition
EHR	Electronic Health Record
FACS	Facial Action Coding System
FER	Facial Emotion Recognition
GAN	Generative Adversarial Network
GNN	Graph Neural Network
HCI	Human-Computer Interaction
LMS	Learning Management System
LSTM	Long Short-Term Memory
MVT	Model-View-Template
NLP	Natural Language Processing
OpenCV	Open Source Computer Vision Library
RNN	Recurrent Neural Network
SLR	Sign Language Recognition
ST-GCN	Spatio-Temporal Graph Convolutional Network
TTS	Text-to-Speech

1. Introduction

1.1 Background

Communication is a basic human activity through which people share everything from thoughts to emotions to intentions. Mere words cannot convey full-spectrum meaning; the other modalities of communication, such as gestures and facial expressions, as well as the tone of emotion, often speak of what were attended to or assumed from the speaker's words themselvesto a huge portion ofcommunicative meaning [1]. This type of communication remains largely undiscovered for many people with hearing or speech impairments. Therefore, important aspects such as education, work, or healthcare exclude them completely [2].

In order to address the communication gap, ASL becomes that visual-spatial medium for the DHH community, as shown in Appendix A.1. With its array of hand gestures, facial expressions, and body posture, ASL integrates these elements into a grammatically complex system to express both abstract ideas as well as feelings and even subtle differences in meaning between discussions[3]. Yet, for all of its expressiveness, the actual digital platforms used for mainstream communication have very little support for ASL. Voice- and text-centric interfaces dominate the design landscape, sidelining users who depend on signed input [4].

These include perception and conveyance of emotion, which the linguistic medium complements. Emotional cues to spoken words include smiling softly while providing reassurance and knitting the brow lightly in concern. These cues are paramount in any interpretation involving some form of sign language, as facial expression is an integral part of both the grammatical and emotional components. Unfortunately, most assistive and communication technologies ignore the emotional dimensions of communication and therefore generate flat and out-of-context interactions. [5] Inclusive design must balance both the linguistic and emotional channels of interaction in digital systems. In recent developments, an emphasis on multimodal approaches would integrate gesture recognition and emotion recognition to provide empathetic, human-like interaction. Such a combination is supportive of both accessibility and a more genuine, emotionally-connected user experience [6].

1.2 Problem Statement

1.2.1 Communication Barriers to the DHH

Despite the proliferation of digital tools and video conferencing platforms, the DHH community continues to experience disproportionate access to real-time, inclusive communication. Many educational institutions, healthcare systems, and workplaces lack built-in support for live sign-language translation, relying on an unfortunate dependence on the availability of human interpreters or post-session transcripts [7]. Although using captioning features integrated within platforms like Zoom, Microsoft Teams, and Google Meet has

improved lots in recent times, none of them have ASL or gesture-based interaction support [8]. This form of technological exclusion restricts access to participate in virtual classrooms, patient consultancies, and collaborative workplace environments for DHH individuals.Without real-time ASL integration, spontaneous interaction is delayed or inaccessible, intensifying the sense of isolation in DHH users digitally [9].

Furthermore, the existing available gesture recognition systems are bulky sensors, gloves, or controlled environments. Such systems lack support for conversational flow [10]. Moreover, the vast majority of assistive systems have quite independently approached gesture recognition with emotion analysis. While a hand gesture might be detected with high precision, the user's emotional state which is the important element for understanding tone or urgency is ignored completely. Resulting in robotic, emotion-detached interaction that goes against what inclusive communication actually aims at achieving. Thus an entire system real time, which interprets the sign language associated with the emotional state detection, would fill in this gap. So, this would not only supply an output of linguistic translation but an affective insight, contributing to transforming the empathetic and context-aware interaction with the DHH population in digital environments.

1.2.2 Lack of Emotional Awareness in Gesture Systems

Human communication extends beyond symbolic language, taking into account facial expressions, the body posture, and emotional tone, nuances on interpersonal sensitivity are conveyed down to intent; as traditional gesture recognition systems reduce communication into a mere mechanical input, tracking only the physical aspects of hand movements while ignoring the affective state of the user [11]. With mechanistic interaction, an emotional depth is not accounted for, and nuances such as sarcasm, urgency, or distress are not captured. For instance, among the DHH students, one signing "I'm fine" while looking sad can be considered emotionally distressed or even disengaged. Therefore the resulting gesture recognition, without emotional awareness of that user's environment, interprets the sentence as neutral and fails to perceive intervention cue. Studies in affective computing emphasize that meaningful interaction requires simultaneous processing of both linguistic and emotional content [12]

The separation of gesture and emotion analysis is a result of legacy architecture along with restrictions in computing resource. Such developments in lightweight neural networks, facial landmark tracking, and real-time sentiment analysis through models such as BERT implicate the feasibility of merging both modalities into a single pipeline [13]. A study provided evidence that combined gesture-emotion interfaces are not only highly accurate, but users also the interfaces with higher trust especially in assistive and clinical applications.

The intent of establishing a unified system to analyze both signed languages with the facial emotion paves the way for emotionally intelligent support technologies providing real-time feedback in classrooms, healthcare, and personal communicative tools that are perceived beyond

linguistically accurate into affective aware bridging not just the communication gap but also the emotional divide for users who rely on non-verbal modalities[14].

1.3 Objectives

1.3.1 Core Objective

The project's core objective is to design and implement an integrated system capable of recognizing ASL gestures against human facial emotions simultaneously in real time with a standard webcam. This ensures the bridging of semantics and affective layers of the non-verbal communication, keeping a record of what one will sign and also how it is being emotionally expressed. The system is intended to support inclusive communication for DHH users by enhancing emotional transparency and fostering natural human-machine interaction in educational, healthcare, and social contexts.

1.3.2 Technical Objectives

In pursuit of the fundamental objective of the system, technical objectives were set up. First of all, real-time ASL gesture recognition was built in using MediaPipe's 21-point hand landmark model, linked to a simple neural classifier for static detection of A-Z fingerspelling. A convolutional neural network (CNN) trained on the FER-2013 data set as shown in Appendix A.2, which was added to classify facial expressions into five emotional states which are 'happy,' 'sad,' 'angry,' 'surprised,' and 'neutral'. The output of the emotion and gesture classifiers is fused within a real-time pipeline to judge the emotion congruence of the signed communication with the user's facial expression. Additionally, human-like communication logic was simulated by introducing features such as dual-hand gestures for space insertion and a timeout-based trigger to finalize sentence formation. All the signed sentences, expressed emotions, and congruency feedback have been stored in structured logs. Real-time user interaction at teaching level for meaningful educational dialogue was incorporated into a live chat-type interface on the web. The scalable back end is to make the entire system applicable even in the educational setting, thus allowing teachers to easily manage their courses, students, lectures, meetings, assessments, and submissions. Finally, all processing was designed to run locally, ensuring user privacy and removing dependence on cloud services making the system suitable for both classroom and clinical deployment.

1.4 Scope of the Project

The system developed is intended to have several functional components and will be designed for communication, continuing education, and session tracking within a unified platform. Real-time recognition of ASL gestures using MediaPipe to a lightweight neural network classifier that can detect and then interpret a static version of the alphabets from A to Z reliably. Gestures were incorporated into natural communication models, such as holding two hands together to group things as spaces and having ends of a sentence based on the absence of detected hands.

Facial emotion recognition (FER) was achieved using a CNN trained with FER-2013 datasets to classify a user's expressions into five types of emotions: happy, sad, angry, surprised, and neutral. These emotional detections complement the gesture stream by verifying if the signed sentences align emotionally with the user's facial expressions, enhancing the system's ability to simulate authentic human interaction.

The project real-time integration of the web portal acts as a live chatroom for students and teachers. The interface can dynamically reflect the signed sentence, an emotion, and sentiment alignment feedback. Each completed interation associated with either of the sentences or emotional mismatch leads to automatically recording it in the session history. At the end of each session, users or administrators can view those records for academic analysis or emotional engagement review.

Apart from communication, the platform deploys important educational management features. Teachers can create and manage courses, upload lectures and resources, schedule events, set quizzes, and upload reports for evaluation. Students, on their side, view and download lectures, take quizzes, submit reports for grading, and attend meetings, all through dedicated interfaces. The grading system allows teachers to review and assess student submissions directly within the portal for educational tracking and feedback loops. All real-time operations, including gesture detection, emotion analysis, chat updates, report generation, and session tracking, are fully localized to operate through the user's machine without reliance on cloud services.

2. Literature Review

2.1 Foundations of Sign Language Recognition

2.1.1 Early Systems: Template-Based and Sensor-Based Models

Early Sign Language Recognition (SLR) systems predominantly relied on hardware-centric and rule-based approaches. These approaches utilized devices such as colored gloves, data gloves, or depth sensors to capture hand movements and positions. Template matching was then exploited to compare these gestures with pre-defined templates for gesture recognition., facilitating gesture recognition. While these methods laid the groundwork for SLR, they were subject to dependence on the environment, restricted scalability, and the need for a special setup, making them impractical for real-world use [15].

2.1.2 CNN-Based Static ASL Detection

The usage of deep learning, particularly through CNN, has changed the process of static hand gestures. CNNs can effectively extract hierarchical features from image data for the right classification of static ASL alphabets. For instance, using hand landmark detection techniques, the system based on CNN achieves a test accuracy rate of 96.42% in recognizing ASL characters [16]. Another approach is using CNN-based ASL fingerspelling translation, which delivers a real-time performance for applications [17]. Such advances point to the promise that CNN holds

in improving the efficiency and accuracy of static gesture recognition systems. A typical static gesture classifier, as shown in Figure 1, will present an input image, seen as a 3D tensor of size 3×W×H, passes with convolutional layers extracting local spatial features, whereas dimensionality reduction and essential pattern-preserving operations are carried out by pooling layers. The feature vectors obtained are flattened into a long column and transmitted to fully connected layers in order to be finally classified within 26 output classes corresponding to the ASL alphabet gestures through softmax activation functions. Various studies have employed this structure and have demonstrated great success in real-time ASL recognition [16],[17].

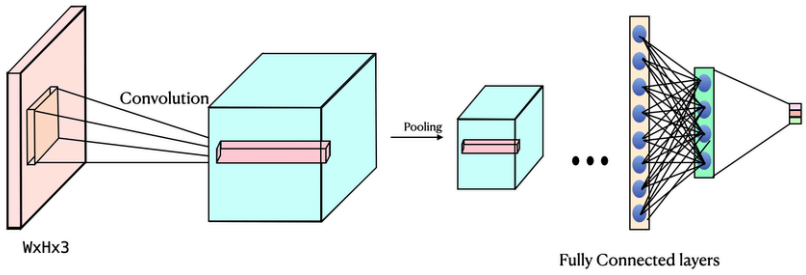
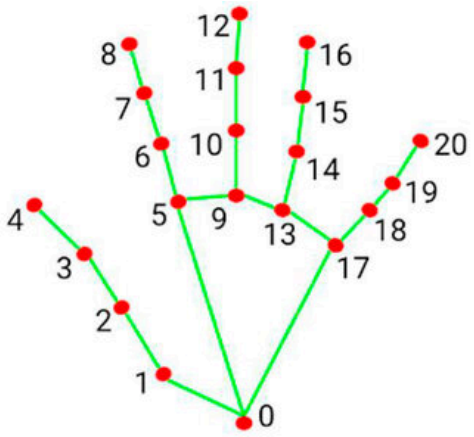


Figure 1 – A typical CNN architecture used for static ASL gesture classification. The model takes an RGB hand gesture image as input and processes it through multiple convolution and pooling layers before classification via fully connected layers into alphabet classes (A–Z).

2.1.3 MediaPipe and Landmark-Based Approaches

To address challenges related to real-time performance and computational efficiency, frameworks like MediaPipe have been introduced. MediaPipe provides real-time hand tracking utilizing 21 keypoints per hand from a single RGB camera, all done by light and CPU-friendly implementations. The anatomical distribution of those keypoints-from the wrist to the tips of each finger-is depicted in Figure 2a and described in the accompanying keypoint mapping table (Figure 2b), hence providing the base input structure for gesture modeling. The integration of MediaPipe and CNN has yielded commendable achievements-for example, using one approach that combined MediaPipe and CNNs a reported accuracy of 99.90% in real-time ASL gesture recognition was achieved [18]. Another implementation made better sign language detection using that integration, giving it an accuracy of 98.83% on ASL datasets [19]. This combination makes it possible to create affordable and scalable SLR systems that can run on off-the-shelf hardware.



(a)

Keypoint ID(s)	Hand Part
0	Wrist
1,2,3,4	Thumb
5,6,7,8	Index Finger
9,10,11,12	Middle Finger
13,14,15,16	Ring Finger
17,18,19,20	Pinky Finger

(b)

Figure 2 – (a) Visual representation of the 21-point MediaPipe hand landmark model used for gesture detection in ASL systems. (b) Corresponding mapping of keypoint indices to specific fingers and wrist segments.

2.2 Dynamic Sign Language Recognition and Temporal Modeling

2.2.1 Deep Learning Approaches for Sequential Gesture Processing

Dynamic sign language recognition (DSLR) extends beyond isolated signs into a series of continuous gestures over time, requiring models that capture spatial and temporal dependencies. Thus, challenges such as gesture coarticulation, signer variability, and background clutter make real-time recognition complex. Recurrent Neural Networks (RNN) methods, especially Long Short-Term Memory (LSTM) networks, have largely formed the backbone of sequential gesture modeling. One study proposed a CNN-LSTM architecture to combine spatial features with temporal dynamics, where it attained an 85% accuracy rate on the RWTH-PHOENIX-Weather dataset [20]. Following up on this, another study adopted a Bi-LSTM approach integrated with attention mechanisms, in which gesture segmentation facilitated the disambiguation of overlapping signs during rapid transitions [21]. More recently, with Graph Neural Network (GNNs)—especially Spatio-Temporal Graph Convolutional Networks (ST-GCNs)—the complexity of gesture recognition has evolved. A multimodal GCN that combined reliable facial and hand keypoints was demonstrated to generalize well across signer styles while handling non-rigid motion [22].

Transformer-based models have emerged more recently as a powerful alternative. Contrary to LSTMs, transformers rely on a self-attention mechanism, which is very efficient in modeling long-term dependencies. The study undertook the construction of a Temporal Transformer Encoder, excelling over any RNN-based models on the CSL-Daily dataset,

especially on complex blended signs such as classifier constructions and fingerspelling. All these attention architectures help improve the interpretability of sequential inputs, and their use is now widening in SLR research [23].

2.2.2 Constraints in Deployment, Coarticulation, and Data Generalization

In spite of the algorithmic improvements, there still remain practical limitations for real-time DSLR. The lightweight continuous recognition system based on MobileNet with temporal pooling achieved under 50 ms of latency on edge devices-important feature for browser-based implementations such as Ain Portal . Coarticulation, the blending of gestures due to rapid execution, is a significant challenge. A segmentation framework was made available to mitigate the issue because it defines gesture boundaries clearly. Furthermore, robust deployment remains limited due to scarcity of high-quality annotated data for dynamic gestures. Actual research on this topic has explored synthetic data generation during the work in a generative adversarial network (GAN) study and a feasibility study on transfer learning from large-scale human action datasets like Kinetics-600 [24]. Both techniques improve signer independence and adaptability to new gesture vocabularies. Current research is also investigating self-supervised pre-training on unlabeled video streams to further reduce the dependency on annotated datasets. This allows the model to learn motion dynamics through raw input, which is essential for scaling gesture recognition to broader domains [25].

2.3 Emotion Recognition and Affective Computing in Assistive Systems

2.3.1 Evolution of FER

FER plays a key role in the field of affective computing, thus accessibility tools supporting hearing-impaired users. The earliest systems relied on the extractive features such as the Facial Action Coding System (FACS) which were fragile under lighting or pose changes. CNNs developed a new dimension for FER with automated feature extraction and generalization [26]. Efficient CNN models working on FER-2013 that allow for real-time CPU-only FER can now find application deployment in web-based educational tools.

2.3.2 Temporal and Multimodal Enhancements

Temporal awareness in FER has been greatly improved through the hybrid deep learning setups. Results have shown that the CNN-LSTM models capture complex expressions like confusion in video sequences much better than with static CNNs that thus improve learning engagement analytics [27]. Fine-tuning, for instance, has also been validated using resourced fine-tuned ResNet models besides varying conditions of lighting and partial occlusions .

To improve further the FER with respect to linguistic sentiment context, state-of-the-art methods have integrated language models such as the BERT into the pipeline for sentence-level sentiment analysis. Indeed, this would enable real-time checks on the alignment of meaning

between signed input and its emotional expression [28]. Take, for instance, the Ain Portal, which integrates this multimodal scheme and provides feedback when emotional signals oppose the sentiment of the signed message, sadness being detected at the same time a positive statement was signed. Such integration structures are illustrated in Figure 3, where it shows how BERT processes raw text input through embedding layers, encoders, and dense output layers to produce sentiment classifications.

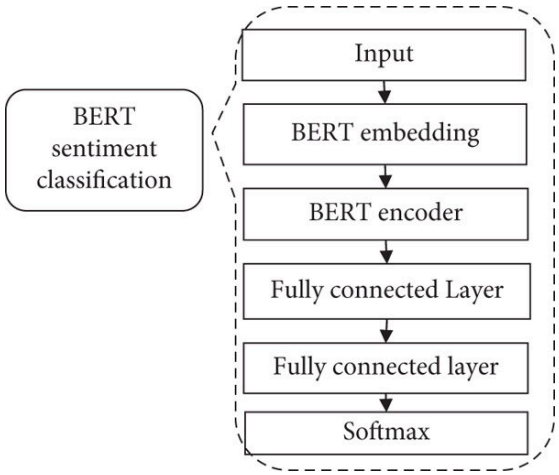


Figure 3 – Architecture of a BERT-based sentiment classification pipeline.

2.3.3 Real-Time Deployment and Fairness

For FER systems to be considered fair and inclusive, they must generalize well over different demographics. One such approach attempted to solve this by utilizing adversarial debiasing methods to improve an FER's accuracy on underrepresented populations [29]. When a self-supervised learning approach was built on top of this, it showed superior performance above conventional CNN models on a variety of ethnic and age groups[30]. In educational and healthcare applications, combining FER with ASL recognition can provide deeper contextual awareness. For example, a system for emotion-aware sign language interpretation has demonstrated the ability to detect sentiment mismatches thereby flagging potential emotional mismatch. Similar logic is implemented in the Ain Portal to issue mismatch warnings during interactions. Transformer-based multimodal architectures, including BERT-CNN fusion models and Vision Transformers with spatiotemporal attention mechanisms, continue to enhance the real-time interpretability and accuracy of such systems [31].

2.4 Multimodal Integration and Fusion Models for Communication Systems

2.4.1 Rationale for Multimodal Communication

Multimodal fusion is used to improve systems of human-computer interaction (HCI), especially those oriented toward education, assistance, or therapy. In ASL systems, gestures

carry the semantic content, while facial expressions contribute affective nuance. Through the fusion of these modalities, machines can approach human levels of understanding: making interpretations of what is being said and how it is being expressed. This fusion enhances user trust, emotional empathy, and interaction quality. [32] .

2.4.2 Fusion Strategies and Architectures

For HCI systems, especially systems for sign language users, merging multiple input modalities such as gestures and facial expressions has shown to significantly increase the quality of communication and user experience. Various studies in environments such as education and assistive technologies have adopted different ways of fusing these inputs to provide an interaction pipeline that is more natural[33].

In general, the fusion strategies can be divided into three categories: early, late, and hybrid integration. Early fusion is basically to fuse the input before the input parameters are independently processed, while late fusion is basically to process the inputs separately and then join them at the decision level. In the case of the Ain Portal, late fusion is implemented because it fits the purpose of being executed in real-time with the least of computing resources. Early fusion is avoided as it raises the computational demand. The fusion of multiple streams has already established a variety of research findings. For instance, research into smart classroom environments has shown that validating the emotional state, along with inputs from ASL, allows teachers to better interpret student engagement. When facial expressions and the signed message do not match there will be an important cue in situations with emotional masking or confusion that is flagged by the system [34].

Recent research has mentioned newer mechanisms, including cross-modal prioritization, where the augmentation system will give different weights to either gesture or emotion depending on which one may be more informative. Some studies have even explored modeling connections between facial and hand movement using network-based representations, enabling better handling of subtleties like sarcasm, hesitation, or uncertainty factors that are often present in real-world communication [35]. Most experimental developments progress towards completely trainable systems featuring end-to-end gesture, emotion, and linguistic content processing, while practical real-world deployments like Ain Portal are still mostly modular and rule based in their integration for the sake of simplicity, interpretability, and portability across educational and clinical settings [36].

2.5 Sign Language in Educational Technology and Accessibility Tools

2.5.1 Importance of Inclusive Educational Platforms

For education technology, the integration of SLR indicates a major advancement in technology services for communities such as the DHH communities of over 70 million individuals worldwide. The users otherwise rely on visual-spatial modalities for communication, and mainstream digital teaching environments still offer limited support for real-time ASL input by relying on captioning or interpreters [37]. Such exclusion hinders active engagement in digital classrooms and assessments.

2.5.2 ASL and Emotion Integration for Engagement

SLR enables a student to question, discuss, or complete an assignment through natural signing. When combined with facial emotion recognition, this capability is significantly enhanced, as it allows systems to interpret not only what is being said through gestures but also how it is being emotionally conveyed. This gives emotional context, which is crucial in a learning environment where signs of confusion, urgency, or distress rarely come forth in conversation [38]. In recent times, classroom-oriented multimodal systems have made use of real-time keypoint-tracking tools and light neural networks to enable ASL-based interaction even on resource-constrained machines. Thus providing a means for DHH learners to truly participate in virtual classrooms. Despite much being achieved, barriers to inclusivity still exist. Most of the sign language recognition systems are trained from a standard ASL dataset and, thus, are not effective for those who communicate with regional dialects, mixed-sign systems, or home sign variations. Possible blue prints to address this challenge incorporate architectures which can be implemented in developing domain-specific ones. In support of such diversity, the Ain Portal will continue developing a modular classification pipeline within which its gesture recognition model may be fine-tuned or completely replaced, thus maintaining continuity and scalability toward broad and diverse demographic and linguistic environments [39].

The Ain Portal follows the same approach by correlating each signed sentence with its concurrently detected facial emotion and carrying out an alignment check. This fusion enhances emotional fidelity and allows the educator to act upon what mismatches arise, such as the positive signing with a sad or neutral expression. In these situations, the system can alert emotional masking or disinterest, which generates a timely intervention and supports the teacher.

2.5.3 Logging, Analytics, and Modular Architectures

Recent tools focus on learning analytics drawing upon gesture logs combined with affective markers. For example, systems that use ASL to capture student responses while FER measures engagement [40]. Logs are visualized with performance graphs to assist the educator in reviewing sessions afterward, Ain Portal does something similar via structured logging and PDF report generation, showing educators trends in emotional and linguistic engagement over time.

The system's modular Django architecture is aligned with best practices for accessible edtech systems by separating out models for quizzes, users, media, and feedback supports

maintainability, scalability, and clarity for cross-role workflows between students and teachers to fully apply on classroom interaction.

3. Methodology

3.1 Methodology Overview

The Ain Portal system is implemented through a modular and layered approach which aims to integrate all aspects of gesture recognition, facial emotion detection, sentiment analysis, and real-time communication into a single platform. The corresponding methodology revolves around the concept of lightweight deep learning models coupled with efficient real-time pipelines designed to operate on standard CPU-based machines without reliance on specialized hardware. The webcam input is recorded and processed simultaneously on two recognition pipelines namely gesture detection and facial emotion recognition which their output merge into a unified, expressive text-based communication interface.

The framework uses MediaPipe's hand tracking pipeline to build the gesture recognition pipeline.Each detected hand with minimal computational overhead would yield 21 keypoints. These landmarks are normalized and incorporated into specially trained TensorFlow Lite model that classifies static ASL letters (A–Z). Concurrently, Open Source Computer Vision Library (OpenCV)’s Haar Cascade classifier, as shown in Appendix A.3, identifies facial regions, which are then analyzed using a CNN trained on the FER-2013 dataset to detect emotional states across five categories: happy, sad, angry, surprised, and neutral.

The central logic module is responsible to handle functions such as gesture stabilization, sentence construction, emotion-sentiment consistency check, and interaction logging. Sentences are built dynamically based on hand gestures, with time-based thresholds managing letter confirmation, space insertion, and sentence completion events. The entire sentence would further undergo real-time sentiment analysis using the Hugging Face’s Transformers library and an extra validation layer by comparing linguistic tone to facial emotion, which is inspired by affective computing studies.

The entire system uses a backend Django framework that helps coordinate real-time API communications with the web interface to modules on the detection side. Live updates on detected gestures, emotions, and feedback are presented to users in a structured chatroom environment, enhancing both transparency and user engagement. An interaction is automatically logged chronologically capturing emotional journeys and signed communications within an external context. This guarantees modularity, scalability, and usability, thus achieving the stated

objectives of the project for expressive and emotionally aware communication channels to DHH communities as well as in educational and clinical contexts.

3.2 System Architecture

The design of the Ain Portal system architecture is based on a real-time, multi-threaded pipeline which has close interactions among the modules, including gesture detection, facial emotion recognition, sentiment analysis, and user interaction management. The design philosophy emphasizes modularity, fault tolerance, low-latency communication, and scalability, so that major functions are easily upgradable or expandable without disturbing the workflow in the whole application. .

At the highest level, the system is composed of five architectural layers: Input Acquisition, Gesture Detection, Emotion Detection, Interpretation and Validation, and UI Layer. These components are tightly integrated yet logically isolated against one another, as illustrated in Appendix B.1.

3.2.1 Input Acquisition Layer

The system begins by continuously acquiring the webcam feed through OpenCV, capturing frames at an average speed of 25-30 frames per second. Each frame is subjected to a basic level of preprocessing, such as horizontal flipping (to provide the user comfort through a mirror view) and color conversion to grayscale where applicable; the next frames are then routed to the gesture and facial recognition pipelines using multi-threading to ensure synchronous but isolated processing streams, minimizing the system's latency and resource-consuming.

3.2.2 Gesture Detection

Gesture detection is performed once a frame arrives and sends the signal to MediaPipe to detect and localize hand landmarks. For each hand detected, the hand detection model of MediaPipe is applied first and then the hand landmark model, which produces 21 (x, y) coordinates. These landmarks are normalized against the wrist position, making them scale and translation invariant. The normalized keypoints are then vectorized into a 42-dimensional feature vector that is input into a lightweight neural network classifier based on TensorFlow Lite, specifically trained to recognize static ASL alphabet gestures. A temporal validation mechanism has been built in, whereby only gestures being detected for at least 1.5 continuous seconds are considered valid and appended to the sentence under construction.

3.2.3 Emotion Detection

The system uses OpenCV's Haar Cascade classifier to form facial regions in each frame. These regions are then preprocessed and reshaped as per the input shape (48 x 48 pixels) of the CNN that was derived during the training of the dataset. The CNN model has multiple convolutional, pooling, and fully connected layers with a softmax output for the five classes of emotion. The predicted results are made stable in real time through a sliding window application in address to frame jitter occurrence and enhance affective consistency.

3.2.4 Interpretation and Validation Layer

This layer represents the fusion of gesture and emotion outputs. It operates with a sentence construction logic with dynamic updates, inserting spaces (underscore _) when two hands are detected together for more than two seconds and marking sentence completion when there are no hands detected for five seconds. When the sentence is complete, the Hugging Face sentiment analysis pipeline goes into operation to determine the sentiment of the text generated. The derived sentiment label (Positive, Negative, or Neutral) is verified against the predominant facial emotion detected; in cases of inconsistency, the system logs it: for example, a sentence with a positive meaning but a facial expression denoting sadness. The system flags these inconsistencies, issues a warning, and logs the event for the educator's review.

3.2.5 User Interface Layer

The interpreted outputs (sentence, emotion, sentiment feedback) are sent to a Django-based REST API that feeds into the live web interface created entirely using HTML, CSS, and JS. The chat interface will show the recently signed sentences alongside their emotional relatedness and would get refreshed every few seconds using the polling mechanism. All interaction will be logged in structured text files and databases. Normally, at the end of a session, reports are generated summarizing exchange into PDF format, with time-series sentences, emotions, and all flagged discrepancies using ReportLab. A storage mechanism such as this would allow teachers and therapists to analyze emotional engagement and communicative efficiency during each session after its completion.

3.2.6 Scalability and Modularity

Each one of the pipelines—gesture, emotion, interpretation—was made as an independent Python class to ensure seamless extension or replacement. For example, it would be possible to replace the MediaPipe Hands tracking system with OpenPose or a 3D skeleton tracker without any effects on the emotion detection logic. The reverse would also be true; the introduction of a possible LSTM-based continuous sign recognition system would affect only the gesture classifying module and not the whole architecture.

3.3 Deployment Choices

The system is locally hosted as a standalone application integrating Django for backend routing, OpenCV for video processing, and TensorFlow Lite for gesture recognition, while the frontend runs through a browser. All components run in a single machine environment without cloud infrastructure, allowing for low latency, data privacy, and offline access-all desirable attributes for an educational or clinical deployment with limited IT resources.

3.3.1 Modularity and Scalability

The architecture is modular design pattern since it contains components like ASL gesture classification, facial emotion detection, sentiment alignment, chat handling, Text-to-Speech (TTS), and session logging, isolated into dedicated scripts or modules. This separation allows testing, debugging, and upgrades in isolation, which could in future be beneficial by eliminating

the component's effect on unrelated components. Additionally, the HaarCascade-based emotion detector can easily be replaced with a deep CNN module. The static gesture recognizer could also be extended to dynamic gesture recognition through LSTM without changing the frontend or backend logic. That modularity also extends to scalable deployment. The ASL and emotion models, built in Keras then converted to TensorFlow Lite, are small enough for the embedded systems and mobile apps. Moreover, the Django backend can be containerized for future deployments in cloud platforms such as Amazon Web Services (AWS) or Azure, enabling multi-user scaling whenever really required.

3.3.2 Standalone Operation and Portability

The system is designed to minimize third-party dependencies. Video input and GUI rendering are performed by OpenCV; inference is carried out with local TFLite/Keras models. For its database back-end, SQLite3 is used to ensure zero configuration deployment. The front-end was designed using plain HTML, CSS, and JavaScript for maximum portability. This solution's usage presents a deployable across all devices ranging from classroom desktops to clinician laptops without the need of specialized hardware or constant internet connections. Logs and reports are exportable (CSV, PDF) and interoperable, supporting effortless integration with external systems such as learning management systems (LMS), electronic health records (EHR), or research dashboards through REST-style endpoints.

3.3.3 Deployment Summary

Local-first deployment architecture achieves a balance between performance, privacy, and portability without being bound to cloud infrastructure and real-time response. Modular codebase with a small model footprint and little dependencies lend themselves to the different institutional environments and provide a strong basis for expanding future cloud migration. The comprehensive breakdown of the development environment with source code organization, model directories, and API endpoints has been detailed in Section 3.6, which demonstrates how practical modularity is achieved by clear directory separation and functional encapsulation.

3.4 Setup Workflow

The Ain Portal assures synchronous gesture and emotion processing integrated by capturing video input through OpenCV and slicing each frame and making it go through two independent processing pipelines: gesture recognition and facial emotion detection. Towards this end, this platform achieves low latency and high availability as well as effectiveness in the response of varied hardware environments. The gesture recognition extracts the landmark points that are normalized and reshaped to form a 42-dimensional vector, which feeds into a small TensorFlow Lite classifier. The classifier is time constrained: it will only validate gestures that lasted for 1.5 seconds or longer. Spaces are introduced when both hands remain together for more than two seconds, while sentence finalization is triggered by a five-second absence of hand detection achieving both accuracy and natural language rhythm in real-time sentence building at a time-aware flow.

Simultaneously, the facial region is detected and resized for emotion inference. This part of the work involves a pre-trained CNN model which classifies the emotions into one of five

versions. Several frames are considered while ensuring consistency. Earlier, the detailed explanation of the CNN architecture and the data preparations was made, but now the paper addresses itself simply to emotional consistency. For every sentence finalized, the system invokes Hugging Face's sentiment analysis model for textual tone detection and then compares it with the most dominating detected emotion at the time of gesture input. Semantic-emotional mismatches are marked well in the interface, adding another level of emotional awareness that is very important in contexts of accessibility.

The integration of non-functional requirements into the system design has been achieved. The application has been designed to give an almost real-time experience by performing full sentence processing and emotional classification in under 2 seconds. Reliability has been built using isolated pipelines with redundant logging to ensure that data is not lost in the event of partial failures. Scalability has been ensured by the adoption of a modular deployment strategy that permits anyone component's upgrade—for example, the emotion classifier, dynamic gesture support—without demanding any rewriting of other components. Portability is ensured as SQLite and lightweight TensorFlow Lite models are used to allow deployment even on low-end educational hardware. A web-based chat interface is upheld for usability, thereby requiring minimal training for both students and faculty.

3.5 Ain Portal Folder Structure

The folder structuring of the project environment influences its maintainability, scalability, and collaborative development. In the Ain Portal project, the directory hierarchy is separated to address concerns across authentication, meetings, quizzes, students, teachers, static files, and APIs. This modularization allows different components to progress independently without interfering with each other’s operation. An overview of the root project structure is presented in Figure 4, while a detailed user interface showcase is provided in Appendix C in full.

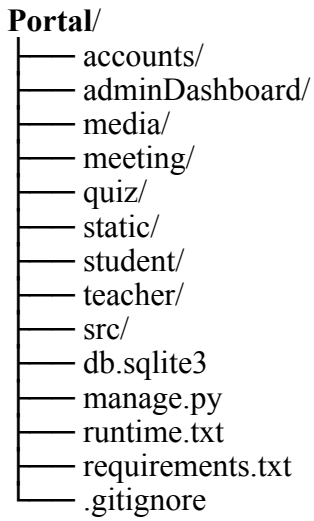


Figure 4 – Top-Level Project Structure

The root structure separates the backend logic (the Django apps) into the folders accounts, teacher, student, and meeting. It also has static for the front-end resources and media

for user-uploaded files. src has the main settings and URL configuration. Support files like manage.py controls the Django server management, while *db.sqlite3* represents the development database.

3.5.1 The *Accounts* Folder

The *accounts* folder as shown in Figure 5 manages user authentication and role management throughout the Ain Portal. It defines how teachers, students, and administrators log onto, register, and interact with the platform based on their privileges. This application customizes the default user model provided by Django to accommodate role-specific attributes and permissions while ensuring a flexible and secure system across different user types.

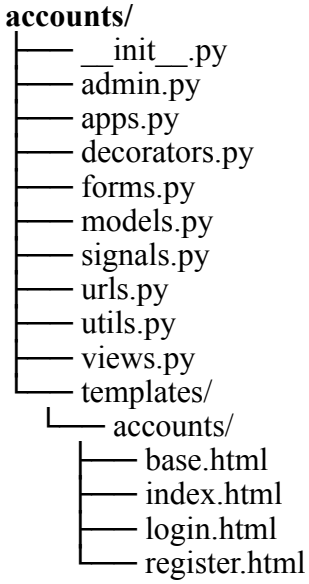


Figure 5 – Directory Tree of Accounts

The *accounts* module defines the structure of the Ain Portal system with respect to authentication and user-role management. Within *models.py,* a customized *User* model extends Django's *AbstractUser* to allow for more flexible role-based behaviors by introducing three Boolean fields-*is_student*, *is_teacher*, and *is_admin*. This system of roles is further augmented by an abstract *Profile* class extended both by the *Student* model and *Teacher* model, administering structured metadata such as name, email, interests, and profile pictures. Picture profiles are automatically resized to ensure uniformity within interface contrivance. The account also contains a lightweight *Contact* model to facilitate feedback submissions by users. Input validations for the various forms used in *Login*, *Registration*, and *Profile* are expertly managed in *forms.py* ; *decorators.py* enforces access control based on roles and protects resources from visibility. The *views.py* file controls the session logic; whereas authenticated users are smoothly routed to their dashboards based on assigned roles. In the front end, the templates in *templates/accounts/* rely on Django's context processor to render the views dynamically. *base.html* serves as the skeletal structure, while *login.html* and *register.html* provide interfaces to authenticate users, with CSRF tokens embedded in both to ensure the secure transaction flow during these two processes.

3.5.3 *adminDashboard* Folder

The *adminDashboard* folder as shown in Figure 6 provides a lightweight but scalable framework for administrative operations in Ain Portal. Although currently minimalist, its architecture prepares the platform for future expansion into system monitoring, user analytics, and administrative reporting.

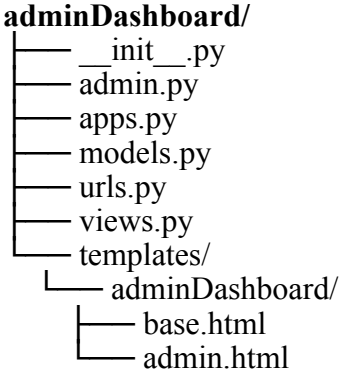


Figure 6 – Directory Tree of adminDashboard

An entry point representing user access to the central point in the form of *admin(request)* view in the *views.py* file of the *adminDashboard* app. This view renders the *admin.html* template when a user requests the route */adminpage/*. This provides the structural basis for expanding administrative features, including potential modules such as system analytics or user moderation tools. URL routing is provided by *urls.py*; thus, it makes endpoint mapping cleaner via Django's dispatcher. The template architecture supports modular separation too: *base.html* in *templates/adminDashboard/* defines the scaffold for layout and reusability, while *admin.html* extends this base to inject dashboard-specific content. *Models.py* and *admin.py* remain unused in this iteration, but their presence will ensure forward compatibility with future administrative models for smooth integration of additional backend logic without disruption of the existing ecosystem.

3.5.4 The *media* Folder

The *media* directory as shown in Figure 7 serves as the pooled storage for all dynamic user-uploaded content in the Ain Portal ecosystem. The built-in system of media management in Django works by mean of *MEDIA_ROOT* and exposed through *MEDIA_URL*. It manages the uploaded files of these sources to this directory without systematic organizing.

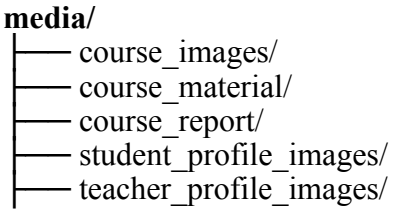


Figure 7– Directory Tree of media folder

The *course_images* folder contains visual thumbnails and banners relating to courses, assisting in the representation of these courses in a manner that is attractive within the UI of the portal. It contains all the lecture files and teaching resources uploaded by instructors for access by students in the *course_material* subfolder. The *course_report* folder holds all student-submitted reports and feedback or graded documents generated by teachers. Finally, the *student_profile_images* and *teacher_profile_images* folders will host uploaded avatars and personal picture data related to the student and teacher models, respectively as part of personalized profiles.

3.5.5 The *meeting* Folder

The *meeting* folder as shown in Figure 8 can be considered as the logic host pertaining to hosting interactive virtual sessions and real-time communication workflows, along with such value proposition features as cross-functional meetings and emotion/gesture interpretation. It coordinates between user actions (students, teachers) and system responses (ASL, emotion, reports, chats).

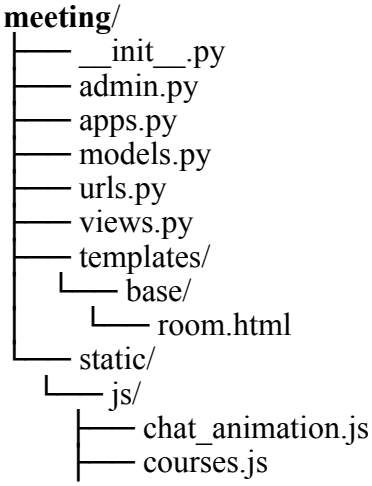


Figure 8 – Directory Tree of meeting folder

The *meeting* app defines the *Meeting* model inside *models.py* preserving information related to session scheduling, room identification, and session metadata. This is applied to the concurrent communication in live meetings and the retrospective review of session activities. Basically, *views.py* directs the workflow for all the major processes such as creating the room, entering the room, fetching live statuses for ongoing sessions and managing chat history that keep backend activities in sync with their representations on the frontend. URL mappings are catered to in *urls.py*, mapping important endpoints for users to access live rooms, updating chat states, and calling the live APIs of the system. Central template *templates/base/room.html* acts as the real-time user interface that streams and displays live signed sentences, emotion detection, and feedback indicators through integrated JavaScript calls. *Static* JavaScript files such as *chat_animation.js* animate the incoming messages and further enhance user experience, while

other collaborative scripts are for page dynamics, validations, and user interactions for the different functional areas (courses.js, reports_page.js, student_page.js). In essence, the *meeting* app acts as the primary convergence point of gesture detection, emotion recognition, and interactive communication into a coherent academic or consulting setting.

3.5.6 The *API* Folder

The *API* subfolder lies within the *meeting* folder as shown in Figure 9 is the cognitive kernel of the Ain Portal system itself. It does live detection and emotion analysis.. It is another module layer, totally separated from all the intensive real-time tasks of a Django MVC flow, thereby ensuring operational responsiveness of the application without burdening the server web.

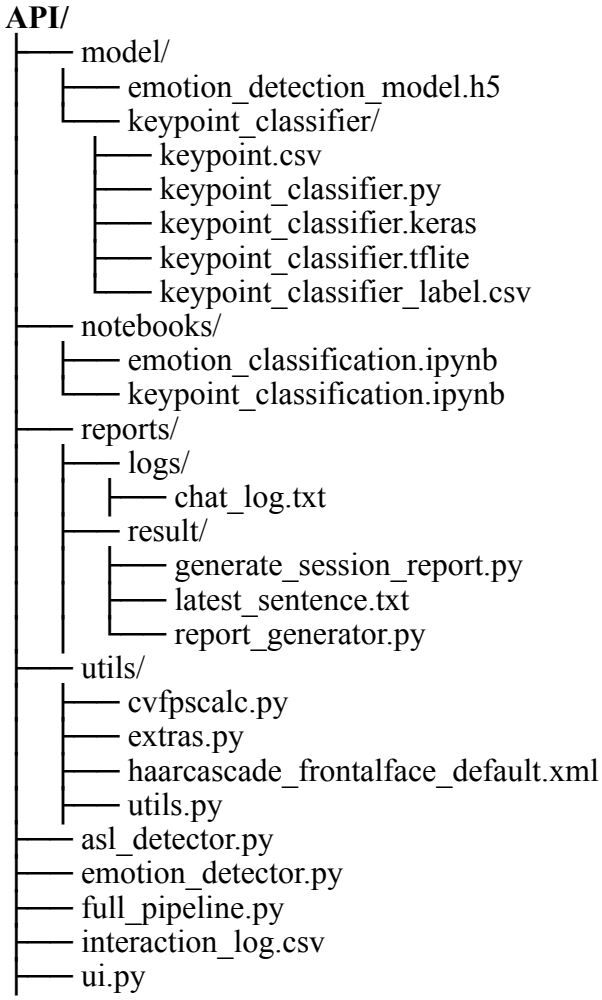


Figure 9 – Detailed Directory Tree of API folder

3.5.6.1 ASL Detection (asl_detector.py)

The *asl_detector.py* module operationalizes real-time ASL recognition. It utilizes MediaPipe's Hands framework to the landmarks from each detected hand capturing both hands

simultaneously. These raw coordinates are normalized relative to the wrist, ensuring rotational and scale invariance across users and conditions. Once normalized, the 42-dimensional vectorr is provided into an efficient neural classifier, designed using TensorFlow, and optimized using TensorFlow Lite, for real-time inference.

This detection logic is rule-based in creating formulated sentences in case of gesture detection. A gesture must be held for a minimum of 1.5 seconds to confirm classification, preventing accidental character repetition. A simultaneous two-hand gesture held for two seconds signals a space ("_"), while a lack of hand detection for five seconds ends the sentence construction phase. The final sentence is then used downstream in the sentiment-emotion alignment pipeline.

The classifier is located in the *model/keypoint_classifier/* directory, which includes a version of the model in *.tflite* format for deployment and one in *.keras* format for retraining. The *keypoint_classifier.py* document processes and invokes the prediction engine used with a TFLite model for live sessions. Label mapping of output indices to characters is done via *keypoint_classifier_label.csv*. A training dataset of normalized landmarks is present as *keypoint.csv* and forms the basis of the classifier's learning process. This system enables accurate spelling of words via finger spelling in ASL. It combines computation perception and structured state management logic in order to articulate grammatically correct sentences.

3.5.6.2 Emotion Detection (*emotion_detector.py*)

The emotion_detector.py module operates in parallel with the ASL gesture detection module to give an emotional context to a user's facial expression recognized in real time. The OpenCV HaarCascade classifier initiates this process by scanning the live feed from the webcam to determine the locations for frontal face regions. Once a face is detected, the area coordinates where the face is detected are cropped from the live video stream, changed in standard dimension into images of size 48x48 pixels, and converted into greyscale to meet the requirements for input into the CNN Model. The pre-trained CNN loaded from emotion_detection_model.h5 was the aforementioned offline trained on FER-2013 modeling but is now integrated for runtime inference into the Ain Portal system. This CNN predicts the probability distribution over the five emotion categories: angry, sad, happy, surprised, and neutral. Real-time inference is done through continuously applying frame-by-frame, with the primary emotion tagged as an overlay to the video stream.

Emotion outputs are evaluated post-sentence construction using Natural Language Proccessing-based (NLP) sentiment analysis. This comparison between the emotional state inferred from facial features and the semantic tone of the signed sentence enables the system to flag any mismatches, such as sarcastic or suppressed emotional expressions. The discrepancies are logged and displayed in the chat interface for instant view by the user or teacher for feedback. The emotion recognition module is vital for the Ain Portal's goals of improving the

HCI experience by not just interpreting language from gesture but also interpreting emotional nuances to detect discomfort, sarcasm, or disinterest.

3.5.6.3 Unified Processing Controller and Supporting Utilities (*full_pipeline.py*)

The integration of ASL gesture modeling with facial emotion recognition modeling functions at a central level through the *full_pipeline.py* script. The pipeline incorporates both the gesture classification and facial emotion detection modules that analyze both in real-time: the webcam feed from OpenCV. Gesture-related outputs are aggregated in real-time to make coherent sentences. Conversely, facial emotion outputs provide continuous affective context to the user's communication. As a core controller, *full_pipeline.py* manages the beginning and the end of the life cycle for each video frame, including all operations: capture, horizontal flip for a natural mirror effect, landmark extraction, keypoint classification, facial area detection, and emotional inference. After the completion of a sentence, there is an NLP-based sentiment analysis of the produced text, which is run in conjunction with a comparison to the detected facial emotion for possible mismatches. The last variable aspects of user interaction is the time stamps, sentences constructed, emotion classification, status of the sentiment agreement, and confidence scores of each model that will be stored into structured CSV and TXT files. They will be used for real-time updates of the dashboard during active sessions and to produce complete session reports post-interaction. This ensures a highly organized flow of gesture, emotion, and sentiment data across the Ain Portal system. Further evaluation metrics, confusion matrices, and performance benchmarking are presented in Chapter 4.

For asynchronous interaction between frontend client and backend server, *API/views.py* specifies lightweight RESTful endpoints. The */live-status/<room_name>/* endpoint fetches the current sentence, detected emotion, and feedback (e.g., mismatch warnings). The */chat-history/<room_name>/* endpoint provides the accumulated messages to display them in a room interface. Message sending is supported through */send-message/<room_name>/* to facilitate bi-directional communication between the user and instructors.

Finally, the utility script *cvfpscalc.py* calculates the frame processing rates and feeds a moving average of FPS values to the interface overlay so that an application's performance remains under real-time thresholds while under load. Logging functions become modularized to avoid blocking frame processing's I/O functions, thus maintaining smooth real-time interaction over the entire session. Thus, the architecture not only enables real-time gesture-emotion fusion, yet is capable of sustaining chat interactivity across sessions and post-session analysis forming therefore the technical and functional backbone of the Ain Portal live communication environment.

3.5.7 The quiz folder

The *quiz* folder as shown in Figure 10 is responsible for the management of evaluation components in the Ain Portal, whereby teachers create quizzes, add questions while students can take assessments and receive evaluations. It forms one of the basis components of the educational perspective of the platform by providing self-assessment, examination, and academic scoring functions.

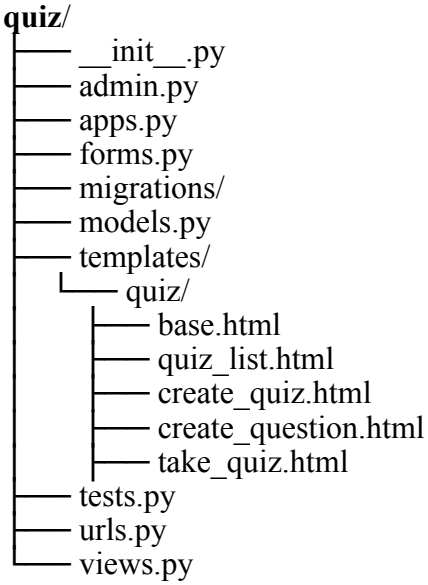


Figure 10 – Directory Tree of Quiz folder

The *Quiz* and *Question* models have been defined in *models.py*. Each quiz associates a course with a teacher, while each question connects to its quiz with the relevant options and a metadata for the correct answer. This model supports scalability and modularization in the design of assessment. The *forms.py* contains standard Django forms for creating both quizzes and questions, thus providing robust validation of data before any such data submission to the database. The *views.py* manages the full lifecycle of quizzes: listing available quizzes, allowing teachers to create or edit quizzes, and handling student submissions with automatic grading based on their selected answers. The *templates* under *templates/quiz* control the front end experience offering dynamic pages for teachers to create and edit quizzes, while students take quizzes and see instant feedback. All the core functionalities mapped with URLs are contained in *urls.py*, which enforces role-based permissions in managing and attempting quizzes between teacher-students. By cross-linking references with the *teacher* and *student* modules, *quizzes* are tied to *course* structures and *user* enrollments such that each *student* would only be afforded access to material pertinent to him or her. Altogether, the *quiz* module enhances the educational strength of the platform by tightly integrating dynamic quiz creation, instant grading, and role-based access control together within a module-based design.

3.5.8 The *static* folder

The *static* directory as shown in Figure 11 contains all front-end assets to deliver a responsive and intuitive user interface. These assets are split into subfolders based on file type and function which keeps the project considerably organized and scalable.

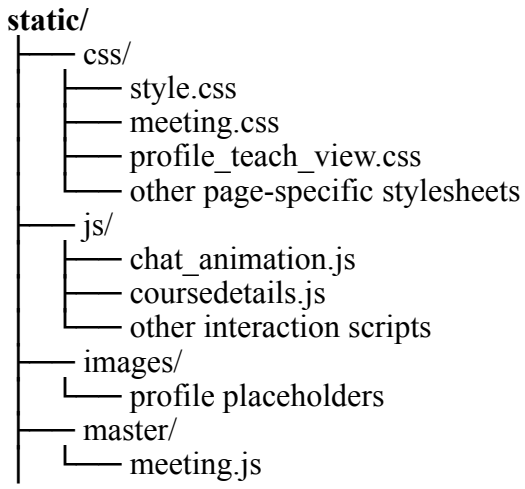
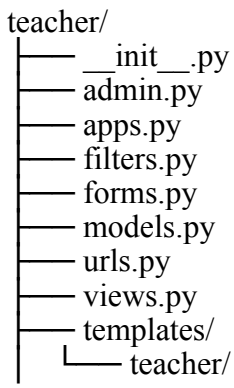


Figure 11 – Directory Tree of static folder

The *css* folder contains all cascading style sheets used by pages within the modules for *students*, *teachers*, *meetings*, and *quizzes*. Each style is page specific which offers localized styling, which is healthier for maintainability. The *js* folder stores JavaScript files that control dynamic page behavior: chat animation effects (*chat_animation.js*), real-time meeting controls (*meeting.js*), and assorted navigation script for courses. The *images* directory contains all visual assets, separated into thematic subfolders to permit clean reference from given templates. This contains button icons, logos, background images for the dashboards, and avatars.

3.5.9 The *teacher* folder

The *teacher* folder as shown in Figure 12 constitutes the teacher's control room into Ain Portal, giving the architecture that enables teachers to control the academic environment. *Teachers* can create courses, upload lectures and materials, assign reports for submission, set quizzes, schedule meetings and monitor student progress.



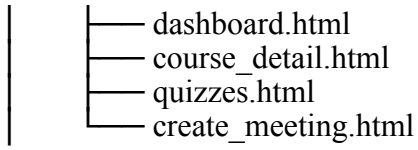


Figure 12 – Directory Tree of teacher folder

The models defined determine the life cycle of academic content which consists of the entities *Course*, *Material*, *Report*, and *Quiz*. Along with this, teachers interact with the portal using easy and intuitive forms for which the validations and input constraints are handled by Django's *forms.py* module. These forms are submitted and processed in *views.py* and integrated in the backend with the templates rendering to the teacher dashboard and course detail pages, quiz creation pages. Furthermore, the teacher's ability to schedule live meetings feeds into the meeting app, ensuring unified communication flow. Importantly, any material generated here immediately becomes visible in the student interface, cementing the teacher-student connection: when a teacher uploads a lecture, students enrolled in that course can access it; when a teacher publishes a quiz, enrolled students are notified and expected to complete it. Thus, the *teacher* folderforms the academic input side of the educational cycle.

3.5.10 The *student* folder

The student folder as shown in Figure 13 facilitates course content and quizzes submission, assignments, as well as education progress monitoring, in addition to usability and clarity towards the student. Usability and clarity for the student were considered while creating this app which renders the personal data to a student based on enrollment and academic status.

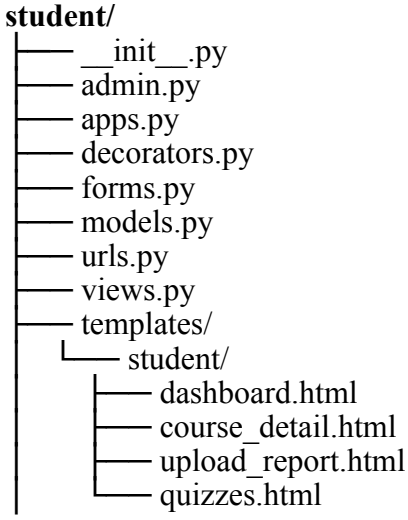


Figure 13 – Directory Tree of Student folder

Students log into their dashboards where they will find courses set up by *teachers* that comprise downloadable lectures, pending report submissions and pending quizzes. Django filters *course materials, quizzes, and report forms* depending on students' profiles and attendance

records in a dynamic way by *views.py*. Report submissions and quizzes are provided through structured forms, which directly update the models synchronous back to the *teacher* dashboard for grading. This design ensures a real-time, bi-directional academic flow between *student* and *teacher* since what a teacher creates is visible to the student, and what the student submits is graded by the teacher. In conjunction with templates for the student application, the decorators and access control mechanisms restrict students to viewing only resources that are relevant to their role and enrollments.

3.6 Selection of Framework Tool

3.6.1 MediaPipe for Hand Tracking

The primary reason for choosing MediaPipe as the basis for extracting real-time hand landmarks is the high computational efficiency associated with this technology as well as extremely accurate tracking, even in systems based on a CPU. In contrast from legacy methods, which required the use of wearing gloves or even using infrared depth sensors, MediaPipe can perform robust 21-point hand landmark detection using RGB frames only. This makes it viable if only using webcams because of the simulataneous multi-hand tracking capability and its robustness in different lighting conditions, enabling replication of gestures smooth enough for creating a natural flow of ASL communication within the Ain Portal system .

3.6.2 TensorFlow Lite and Keras for Gesture and Emotion Models

TensorFlow Lite has been used for gesture recognition aimed at minimizing inference latency and memory usage. The ASL classifier model was trained in Keras and converted into TFLite format to achieve the best runtime speeds on average consumer laptops, avoiding the use of dedicated GPUs. Keras itself was chosen for its intuitive model-building API and its tight integration with TensorFlow, allowing fast development of both the gesture classifier and the emotion detection CNN. The combination of TensorFLow Lite and Keras offers a portable and scalable solution suitable for real-time sign language and emotion processing.

3.6.3 OpenCV for Video Processing and GUI

OpenCV was adopted to manage all interactions with webcams, acquire real-time frames in preprocessing (flipping, resizing, grayscale conversion), detect faces with Haar Cascades, and draw overlays of gesture and emotion labels. Its widespread use in the computer vision community, ease of integration with Python, and mature feature set made it the natural choice. Beyond frame management, it has also been providing for early-stage visuals at a finalized result level before passing it on to a front end based on Django, with near-zero latency between detection and feedback from the user.

3.6.4 Django Framework for Backend and API Management

The Ain Portal implemented Django as the web application backbone, handling session control, storage of chat messages, room management, report generation, and RESTful API endpoints. It had built-in user authentication mechanisms that lay a solid security foundation for managing the teachers and students with different user types. On the other hand, the implementation of the Django ORM made database access with SQLite very easy, whereas the Model-View-Template (MVT) pattern helped a lot in separating the concerns logically, weeding out the aspects of the system that are concerned with maintainability and extensibility for future upgrades.

3.6.5 Hugging Face Transformers for Sentiment Analysis

Hugging Face Transformers was added for quick and reliable sentiment analysis of final ASL signed sentences then it analyzes the sentiment of a sentence as positive, negative, or neutral by leveraging pretrained transformer-based models with simple pipelines. Such analysis was essential for the sentiment-emotion consistency checks within the platform, thus directly contributing to the detection of mismatch analysis towards users

3.6.6 Summary of Framework Integration

The integration between MediaPipe, TensorFlow Lite, Keras, OpenCV, Django, and Hugging Face Transformers met the performance, modularity, and scale ambitions of Ain Portal without requiring expensive hardware or cloud computing. Each framework was not selected in isolation but in relation to the other frameworks, yielding an integrated, lightweight, modular, user-centric, real-time communication system suitable for educational and health environments.

4. Results and Evaluation

This section investigates the all-in-one ASL gesture recognition system, the FER pipeline, and the entire performance of Ain Portal platform. Modules were tested in isolation and as an integrated pipeline to ascertain their real-time responsiveness, classification accuracy, and practical usability for operational deployment. The results come in form of the following: classifier metrics; confusion matrices; interface interaction logs; and screenshots showing system behavior under different scenarios. Such findings endorse the overall validity and reliability of the platform in educational and assistive contexts of communication.

4.1 ASL Classifier Performance

The ASL static alphabet recognition model in Ain Portal is built upon a supervised learning framework trained on a custom dataset comprising 36,000 samples stored in *keypoint.csv*. Each row represents one labeled instance: the first column encodes the ASL letter (as an integer from 0 to 25, corresponding to A–Z), followed by 42 normalized landmark

coordinates (21 keypoints × 2D x,y format). These data points have been recorded by the real-time hand tracking system of MediaPipe, which outputs consistent 21-point hand landmarks per detected hand as shown in Figure 14 . Normalization of these coordinates according to the wrist joint enables them to be translationally and rotationally invariant across all hand sizes, orientations and capture conditions, ensuring that gestures are described equivalently. Demonstrative data entry is displayed in Figure 15 which shows an individual row from the keypoint.csv dataset, used to train the ASL classification model. The first value (0) is the class label corresponding to the letter "A", followed by 42 floating-point values representing the (x, y) coordinates of 21 hand landmarks. These coordinates are normalized relative to the wrist origin to ensure scale, position, and orientation invariance during inference. The structured format supports fast loading into TensorFlow-based models and promotes consistency across training and evaluation pipelines.

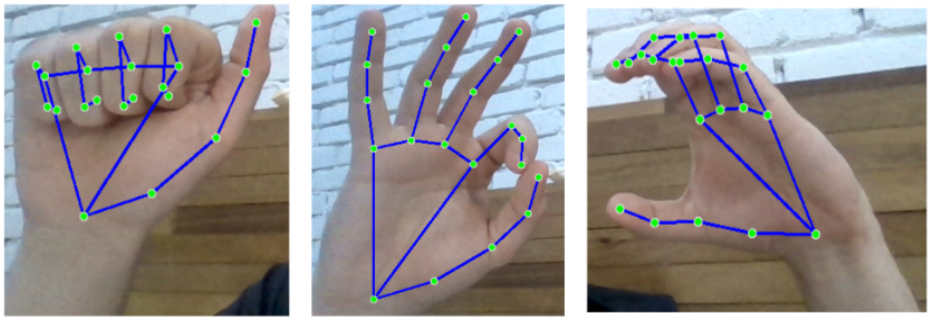


Figure 14– Visualization of 21 detected keypoints hand landmarks using Mediapipe for ASL letters "A", "F", and "C", respectively

```
0      0      0      0.34   -0.04 0.68   -0.26 0.9    -0.28 1      -0.14 0.42   -0.64
0.38   -0.78 0.34   -0.52 0.3    -0.24 0.06   -0.66 0.04   -0.7   0.04   -0.34 0.04
-0.02 -0.2   -0.56 -0.24 -0.58 -0.2   -0.24 -0.16 0.06   -0.44 -0.42 -0.52 -0.4
                                -0.46 -0.16 -0.38 0.06
```

Figure 15 – Normalized Hand Landmark Entry from keypoint.csv

This involved training a lightweight feed-forward neural network with the use of TensorFlow and Keras for classifying gestures. The architecture includes one batch normalization layer followed by three dense layers with 128, 64, and 32 units, all activated by the Mish function. Mish is a better choice than other activations such as ReLU for this architecture, because of superior gradient flow and better generalization compared to those using more substantial datasets. A dropout regularization technique (rate= 0.5) was used after the first two layers to mitigate overfitting. To this model, Adam optimization was used to compile it together with the sparse categorical cross-entropy function as the loss function during training. Model checkpoint callback was included for storage of those optimal weights while the process got interrupted after reaching 20 epochs earlier than weight perturbed conditions. Training ran with a maximum of 1000 epochs, with a batch size of 128 and an unsplit dataset of 75:25 for

training and validation. It shows the model architecture in Figure 16, which displays the regularizing mechanisms and final softmax layer used to classify the ASL characters from A to Z using stacked dense layers in their own classification.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input((42, )),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(128, activation='mish', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(64, activation='mish', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(32, activation='mish', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.Dense(26, activation='softmax')
])
```

Figure 16 – Architecture of ASL Classifier Neural Network

A code demonstration of a TensorFlow model designed for the static hand sign recognition task. The model takes 42 input features and includes batch normalization, three dense layers with Mish activation, dropout regularization, and a softmax output layer covering 26 alphabet classes.

After training, the model was evaluated using *classification_report()* and *confusion_matrix()* from scikit-learn, allowing meaningful per-class metrics to be calculated. Validation accuracies at 86.1% were achieved, showing that strong generalization was achieved across the 26-letter ASL alphabet as shown in Figure 17. Nevertheless, some gestures had a variability in the accuracy between the classes due to their visual and structural similarity. Some examples of per-letter precision, recall, and F1-scores for major alphabet classes are shown in Table 1. With F1-scores greater than 0.90, letters "H" and "Y" performed well due to the distinct differences in finger positions and open-hand postures. On the contrary, classification of characters "M" and "N," which differ only by the number of fingers bent across the thumb, was hard, as these subtle differences were not sometimes well characterized by the static 2D coordinate system and tend to cause confusion. Similarly, "D" and "R," both having the extended index finger, were generally confused with each other due to their visual proximity, but through model-driven improvements which include modifying training samples, this problem was solved, thus allowing the classifier to distinguish very well between them.

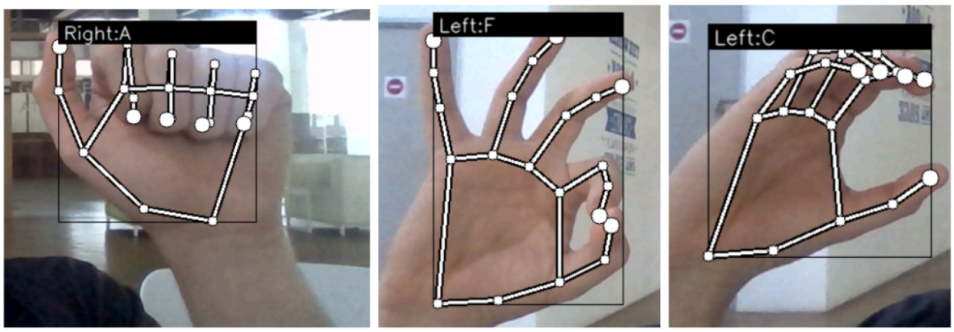


Figure 17 – Trained Model detects ASL letters "A", "F", and "C", respectively

Appendix Table B.2 provides the full classification report, while Appendix B.3 presents the confusion matrix. These visualizations reaffirm that misclassifications are clustered among letters that are visually similar and suggest that model performance may be improved going forward with further targeted data augmentation, for instance, by recording gestures from multiple angles, or additionally taking into account temporal dynamics.

Table 1 – Classification Metrics Summary (Selected Letters)

Letter	Precision	Recall	F1-Score	Support
H	0.97	0.92	0.94	302
M	0.75	0.77	0.76	324
N	0.81	0.87	0.84	373
W	0.95	0.83	0.88	338
Y	0.99	0.82	0.9	531

Partial classification results showing model strengths and weaknesses across selected ASL alphabet letters. High performance is observed in distinct gestures like "H," whereas structural similarities (e.g., between "N" and "M") result in slightly reduced scores.

4.2 Emotion Classifier Performance

The facial emotion recognition aspect in the Ain Portal system uses a CNN trained on the FER-2013 dataset, which is a widely accepted testing ground in affective computing. The FER-2013 dataset consists of 35,887 grayscale facial images of size 48 x 48 pixels, where each image has been labeled under one of the five emotion categories: angry, happy, sad, surprise, and neutral. The improvement mitigates inter-class confusion and enhances the inference accuracy in real-time. The classification pipeline begins from model loading and emotion-sentiment

initialization as illustrated in Figure 18, in which both the CNN model and Hugging Face's sentiment analysis transformer are instantiated:

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from transformers import pipeline

model = load_model("emotion_detection_model.h5")
emotion_labels = ['angry', 'happy', 'neutral', 'sad', 'surprised']
sentiment_analyzer = pipeline("sentiment-analysis")
```

Figure 18 – Import and Initialization of Emotion & Sentiment Models

For evaluation, the model has been validated based on a totally independent test set. The normalized confusion matrix as shown in Appendix B.4, shows enormous diagonal dominance, as a result, most emotions were classified with high accuracy. The emotions with the highest recognition rates were those of happiness (89.98%) and surprise (83.11%), most likely because their predominant facial characteristics — such as smiling or wide opening of the eyes — could be easily identified by the CNN. Confusion arose, however, for more subtle expressions: 40.30% of actual 'Sad' cases were classified as 'Neutral', and 20.65% of 'Angry' faces were also labeled 'Neutral'. These results indicate that with the less salient cues in the face, it is particularly difficult to distinguish emotion in somewhat static images. For live deployment, frames are taken from the webcam and passed through OpenCV's HaarCascade frontal face detector. Once a face is detected, it is cropped from the frame, resized to a 48×48 size, and normalized before starting the emotion classification using the CNN. The whole preprocessing pipeline is shown in Figure 19, which depicts the stages from raw input to model-ready format.

```
frame = cv2.imread("face_sample.jpg")
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        .detectMultiScale(gray)

for (x, y, w, h) in faces:
    face_roi = gray[y:y+h, x:x+w]
    resized = cv2.resize(face_roi, (48, 48)) / 255.0
    reshaped = np.reshape(resized, (1, 48, 48, 1))
    prediction = model.predict(reshaped)
    emotion = emotion_labels[np.argmax(prediction)]
```

Figure 19 – Face Detection and Preprocessing

A key feature of this system is its semantic validation capability. After a sentence has been finalized through ASL recognition, its sentiment is classified through an NLP engine and

checked for consistency with real-time facial emotion. Inconsistency prompts a red flag in the system and logging for educator attention (e.g., A positive sentence spoken with a sad expression). The logic for this comparison is depicted in Figure 20, which shows how mismatches can be detected and logged:

```
sentence = "I am okay"
sentiment = sentiment_analyzer(sentence)[0]['label']
if sentiment == "POSITIVE" and emotion in ["sad", "angry"]:
    print("⚠️ Mismatch Detected")
```

Figure 20– Emotion-Sentiment Mismatch Checker

This integration creates an effect-aware multimodal interface that translates sign language but augments contextual insight into emotional authenticity. It gives educators and therapists a better chance of accurately interpreting user state even when signed communication could be misleading due to contextual constraints.

4.3 Sentence and Emotion Integration

An innovative development of the Ain Portal platform is the inclusion of the semantic emotional coherence layer, implementing the validation of emotional congruence between a user's facial expression and the sentiment of a sentence coded in ASL. The integration of gesture recognition with affective computing aids in imitating conversational dynamics that one might find in the real world, thereby enabling further understanding of the user's intent to communicate.

To test this integration, controlled testing was performed via the *full_pipeline.py* script designed to execute gesture capture, sentence generation, emotion recognition, and sentiment comparison in real-time. During each session, the user would sign emotionally resonating phrases such as *"I am happy", "Not good today"*, or *"Doing great"*. The *ASL_detector.py* module processes these phrases into complete sentences by using gesture hold logic (≥1.5 s/letter) and inactivity thresholds (5 s) to signify the end of a sentence.

Once a sentence was finalized, Hugging Face's pipeline NLP model to draw out the linguistic sentiment which may either be positive or negative, or neutral. Simultaneously, the emotion detection model processes the user's facial expression to predict one of the following five emotional expressions: happy, sad, angry, surprised, or neutral at the moment sentence finalization occurs. It was at this stage that verbal sentiment output and facial emotion output are compared.

4.4 Ain Portal User Interaction

An interactive field test under a user's perspective was conducted to determine the usability and responsive nature of the Ain Portal system with a live interaction in the frontend integrated with Django on the *room.html* interface. This is the interface through which students and teachers directly communicate in the form of real-time dashboards containing ASL-translated sentences, detected emotional states, chat history, and backend-driven feedback. During the tests, there is students signing through webcam interfaces and their teacher responded via a text interface. The latency, reliability, and responsiveness of the full system have been measured from an ASL input to final display feedback loop.

Observed metrics are summarized below:

The system achieved gesture-based sentence construction in an average of 1.5 seconds after the completion of a gesture. Additionally, frame-wise recognition of the emotions was averaged over the inferences performed in 0.8 seconds using the HaarCascade + CNN model. The two ends of chat logs and replies were updated within 3 seconds of input by the user, including RESTful API callbacks. It took less than two seconds to export session summaries, including chat history, emotion records, and sentence logs, as PDF reports along with proper formatting and timestamps. These results prove the system is up to real-time interaction standards with mid-range CPU hardware aligning with accessibility goals in low-resource educational or clinical environments. As illustrated in Appendix C10, the live dashboard summarizes all sentences derived from gestures, real-time facial emotion feedback, and interactions between a teacher and students as presented in one view.

5. Project Management

The Ain Portal project was officially commenced in September 2024 and was developed based on a structured timeline divided into five main phases of development: modeling, interface design, system integration, testing, and documentation. The phases of the Gantt chart are visually represented in Figure 21, thereby providing an outline of the scheduled depiction through the academic year. Although the initial plan targeted completion by March 2025, unforeseen medical circumstances including a spinal disc injury and subsequent microdiscectomy surgery in February 2025, which made a temporary halted any developmental works. On the other hand, with the accomplishments chieved prior to the disruption, combined with the effective modular task planning and focused post-recovery effort, the project was finalized and submitted on time by the official May 2025 deadline. This reflects strong project resilience and adaptability in the face of personal adversity.

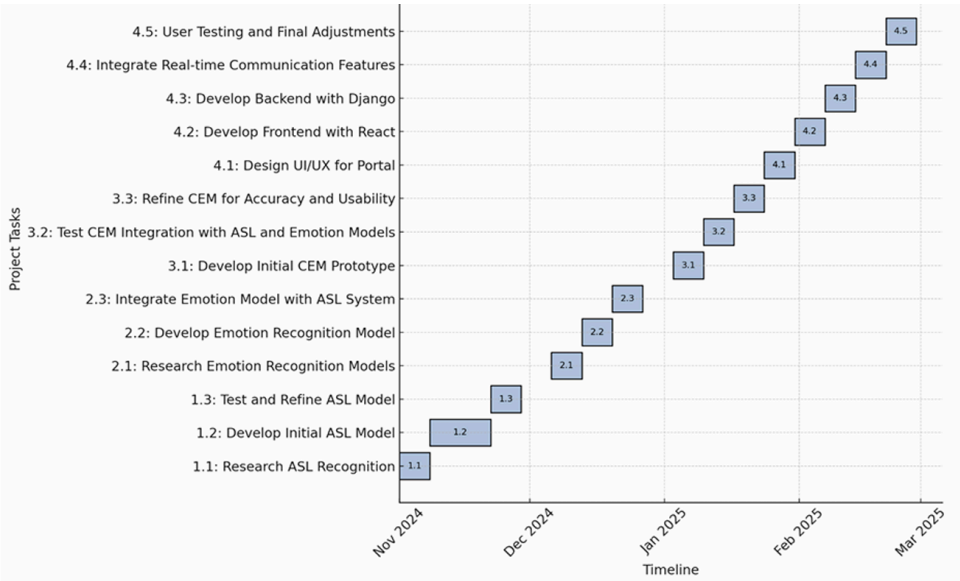


Figure 21 – Project Timeline Gantt Chart showing major milestones from ASL research to final testing (Nov 2024 – Mar 2025).

5.1 Personal Reflections and Technical Learnings

The experience of creating the Ain Portal project was both technically demanding and emotionally close to heart. I suffered a severe spinal disc injury that hospitalized me for treament by operating a microdiscectomy in February 2025. This physical setback brought with it pain and restricted mobility and considerable mental fatigue and emotional strain. Balancing physical rehabilitation with a real-time AI application was an energy-consuming effort. Despite these challenges, the project maintained its schedule due to a modular approach, incremental testing, and completion of core model components early before surgery. Pushing through recovery after surgery while deadlines taught me technical resilience and emotional endurance. It was organized recovery that motivated me and gave a sense of purpose and intent during tough phases both physically and mentally.

On the technical side, I received hands-on exposure to building machine learning models, light-weight deployment using TensorFlow Lite, and synchronized processing of video-streams. The most rewarding experience was ASL learning and practice to rigorously test the classifier. Not only did it improve the model's accuracy, but it also gave me an appreciation for the linguistic diversity and expressive capability of ASL. Parallelly, introducing live facial emotion detection posed tremendous hurdles to my understanding of computer vision and led to an iterative model training learning curve, performance profiling, and UI feedback logic in order to bring about meaningful outcomes. The most valuable take away was always to design with empathy and with inclusivity. Ain Portal is not a technical solution; it's a communications bridge. Understanding how to align user experience and machine prediction made it perfectly obvious that designing for accessibility has both technical know-how and closeness of human interaction.

5.2 Limitations and Out-of-Scope Considerations

While the Ain Portal system demonstrates a functional integration of gesture recognition, emotion classification, and real-time communication, various limitations restrain the operational deployment and scalability of the system. The foremost of these is that the gesture recognition deals only with static alphabet signs using a 21-point hand landmark system, thus punishing itself for disregarding support for dynamic sign sequences or fluid multi-gesture phrases that lay at the core of a natural language-level sign language understanding. Such sentences are still beyond its grasp because of more complex requirements, which would call for sequence modeling using different architectures like RNNs, LSTM units, or 3D Convolutional Neural Networks (3D CNNs). Even so, emotion recognition remains constrained to a five-class output from the FER-2013 database and implemented via a static CNN model. The model lacks personalized online learning capabilities and fails to adapt dynamically over time to an individual user's expressive style or feedback, which could negatively impact contextual reliability in some subtly emotional situations, particularly among various cultural or age-specific groups.

Main operational limitations arise largely from local-only deployment of the system. Although the modular design of the Django backend and the lightweight inference models can work smoothly on mid-level CPUs, there is no support yet for containerization via Docker or deployment to scalable platforms such as AWS or Azure. The application is strictly single-user with no support for multi-user sessions, concurrent access, or cross-device syncing. Additionally, Ain Portal does not connect freely with any of the broader accessibility ecosystems, such as LMS, telemedicine portals, or augmented/virtual reality tools. Within the current scope of implementation are not real-time cloud-based backup, mobile-native apps, or collaborative multi-user sessions. These represent both the limits of the project's engineering and its emphasis on establishing reliability in proof of concept over feature richness. However, given these considerations, design decisions-made, including possibly using TFLite for gesture inference and modular pipelines for future crossways expansion, open the way for significant improvements in the future. Some of the planned future work is dynamic sign phrase recognition using LSTM-based pipelines, integration of adaptive emotion models with online learning capability, and expansion of the reach of the system through cross-platform deployment and LMS compatibility.

5.3 Future Enhancements

With respect to the strengths of modularity and proof of concept from the present Ain Portal system, some suggestions are made for increasing applicability, scalability, and real-world realization. Priority number one would be to extend static ASL gesture recognition to dynamic phrase-level recognition. In this instance, temporal deep learning architectures like LSTM or Transformer-based models would be used to interpret continuous gestures and entire sentences. Future innovations could embed adaptive learning mechanisms to thus renew over time the

emotion classifiers to the styles expressed by the user, increasing precision and relevance in real-time applications. Real-time calibration supported by feedback loops, along with multimodal affective cues like speech tone or physiological signals such as heartbeat, ought to increase richness in understanding emotion.

The addition of cloud-hosted scalability, multi-user session support, and real-time synchronization across devices will enable greater accessibility. Containerization of the application using technologies such as Docker, for deployment over AWS, Azure, or Firebase, offers this scalability, while still being able to maintain modular encapsulation of backend services. Finally, the integration with educational and healthcare ecosystems, like LMS platforms and telehealth services, will translate into real-world value for the system. A real-time teacher dashboard, progress tracker, and automated emotional-intervention alerting will enable ripping this Ain Portal through a functional prototype into a deployable tool for assistive education and therapy.

These forward-looking enhancements address existing limitations and draw upon the system's underlying strengths to present a pathway for a robust user-oriented highly extensible assistive communication framework.

6. Conclusion

The Ain Portal project set out to design a real-time, emotion-aware sign language communication platform that serves educational and assistive domains. It successfully implemented gesture recognition using MediaPipe and TensorFlow Lite, alongside facial emotion detection powered by a CNN model trained on the FER-2013 dataset. By integrating these modalities into a unified pipeline, the system not only interprets user input linguistically but also evaluates emotional alignment, enabling deeper and more empathetic HCI.

Through iterative development, usability testing, and an emphasis on real-time performance, the system achieved responsive interaction with minimal latency, even on standard hardware. The modular architecture ensured maintainability and laid the groundwork for future expansion, such as dynamic gesture support and cloud integration. Educators and therapists can now interpret both what is said and how it is emotionally conveyed, making the tool valuable in contexts where emotional state can guide intervention.

The project also highlighted broader challenges in accessibility tech—such as personalization, dialect diversity, and scalability—many of which are acknowledged as areas for future development. Despite medical setbacks during development, the work was completed on time, demonstrating resilience and commitment. Ultimately, the Ain Portal serves as both a functional prototype and a foundation for broader, emotionally aware communication technologies for the DHH community.

7. References

[1] H. Grote, F. Izagaren, and V. O'Brien, "How to communicate with patients who are D/deaf or have hearing loss," *BMJ*, vol. 373, 2021. [Online]. Available: https://doi.org/10.1136/bmj.n1382

[2] S. Manzoor *et al*., "Voice of Hearing and Speech Impaired People," in *Proc. 2024 Int. Conf. Commun., Comput. Sci. Eng. (IC3SE)*, 2024, pp. 1891–1897. [Online]. Available: https://doi.org/10.1109/IC3SE62002.2024.10593248

[3] E. Hossain *et al*., "Supporting ASL Communication Between Hearing Parents and Deaf Children," in *Proc. 25th Int. ACM SIGACCESS Conf. Comput. Accessibility*, 2023. [Online]. Available: https://doi.org/10.1145/3597638.3614511

[4] C. Lenkaitis, "Technology as a mediating tool: videoconferencing, L2 learning, and learner autonomy," *Comput. Assist. Lang. Learn.*, vol. 33, pp. 483–509, 2019. [Online]. Available: https://doi.org/10.1080/09588221.2019.1572018

[5] G. Caridakis *et al*., "Multimodal emotion recognition from expressive faces, body gestures and speech," in *Artificial Intelligence Applications and Innovations*, vol. 247, pp. 375–388, 2007. [Online]. Available: https://doi.org/10.1007/978-0-387-74161-1_41

[6] S. Güler and F. Akbulut, "Multimodal Emotion Recognition: Emotion Classification Through the Integration of EEG and Facial Expressions," *IEEE Access*, vol. 13, pp. 24587–24603, 2025. [Online]. Available: https://doi.org/10.1109/ACCESS.2025.3538642

[7] R. Hommes, A. Borash, K. Hartwig, and D. DeGracia, "American Sign Language Interpreters Perceptions of Barriers to Healthcare Communication in Deaf and Hard of Hearing Patients," *J. Community Health*, vol. 43, pp. 956–961, 2018. [Online]. Available: https://doi.org/10.1007/s10900-018-0511-3

[8] M. Abou-Abdallah and A. Lamyman, "Exploring communication difficulties with deaf patients," *Clin. Med.*, vol. 21, no. 4, pp. e380–e383, 2021. [Online]. Available: https://doi.org/10.7861/CLINMED.2021-0111

[9] M. Papatsimouli *et al*., "Real Time Sign Language Translation Systems: A review study," in *Proc. 2022 11th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, 2022, pp. 1–4. [Online]. Available: https://doi.org/10.1109/mocast54814.2022.9837666

[10] A. Acharya, N. Patil, U. Pathak, and S. Bhagwat, “Sign Language Translation with fusion of Emotion Detection,” in *Proc. 2024 8th Int. Conf. Comput., Commun., Control Autom. (ICCUBEA)*, 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICCUBEA61740.2024.10774760

[11] C. Carmichael and M. Mizrahi, “Connecting cues: The role of nonverbal cues in perceived responsiveness,” *Curr. Opin. Psychol.*, vol. 53, p. 101663, 2023. [Online]. Available: https://doi.org/10.1016/j.copsyc.2023.101663

[12] L. Malatesta, S. Asteriadis, G. Caridakis, A. Vasalou, and K. Karpouzis, “Associating gesture expressivity with affective representations,” *Eng. Appl. Artif. Intell.*, vol. 51, pp. 124–135, 2016. [Online]. Available: https://doi.org/10.1016/j.engappai.2016.01.010

[13] Y. Chavan, “Emotion and Gesture Recognition,” *Int. J. Sci. Res. Eng. Manage.*, 2024. [Online]. Available: https://doi.org/10.55041/ijsrem31827

[14] L. Chen, M. Li, M. Wu, W. Pedrycz, and K. Hirota, “Coupled Multimodal Emotional Feature Analysis Based on Broad-Deep Fusion Networks in Human–Robot Interaction,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, pp. 9663–9673, 2023. [Online]. Available: https://doi.org/10.1109/TNNLS.2023.3236320

[15] D. Choudhary, R. Singh, and D. Kamthania, “Sign Language Recognition System,” *SSRN Electron. J.*, 2021. [Online]. Available: https://doi.org/10.2139/SSRN.3832151

[16] A. Gangal, A. Kuppahally, and M. Ravindran, “Sign Language Recognition with Convolutional Neural Networks,” Stanford University, 2024. [Online]. Available: https://cs231n.stanford.edu/2024/papers/sign-language-recognition-with-convolutional-neural-networks.pdf

[17] A. Barbhuiya, R. Karsh, and R. Jain, “CNN based feature extraction and classification for sign language,” *Multimedia Tools Appl.*, vol. 80, pp. 3051–3069, 2020. [Online]. Available: https://doi.org/10.1007/s11042-020-09829-y

[18] R. Kumar, A. Bajpai, A. Sinha, and S. K. Singh, “Mediapipe and CNNs for Real-Time ASL Gesture Recognition,” *arXiv preprint*, arXiv:2305.05296, 2023. [Online]. Available: https://arxiv.org/pdf/2305.05296

[19] Y. Xu, T. Xu, S. Han, and K. Kim, “Ensemble Learning of Multiple Deep CNNs Using Accuracy-Based Weighted Voting for ASL Recognition,” *Appl. Sci.*, vol. 12, no. 22, p. 11766, 2022. [Online]. Available: https://doi.org/10.3390/app122211766

[20] W. Yin, Y. Hou, Z. Guo, and K. Liu, “Spatial–Temporal Enhanced Network for Continuous Sign Language Recognition,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, pp. 1684–1695, 2024. [Online]. Available: https://doi.org/10.1109/TCSVT.2023.3296668

[21] S. Abdullahi, K. Chamnongthai, V. Bolón-Canedo, and B. Cancela, “Spatial-temporal feature-based End-to-end Fourier network for 3D sign language recognition,” *Expert Syst. Appl.*, vol. 248, p. 123258, 2024. [Online]. Available: https://doi.org/10.1016/j.eswa.2024.123258

[22] H. Zhou, W. Zhou, Y. Zhou, and H. Li, “Spatial-Temporal Multi-Cue Network for Sign Language Recognition and Translation,” *IEEE Trans. Multimedia*, vol. 24, pp. 768–779, 2021. [Online]. Available: https://doi.org/10.1109/TMM.2021.3059098

[23] M. De Coster, M. Herreweghe, and J. Dambre, “Sign Language Recognition with Transformer Networks,” in *Proc. Conf.*, 2020, pp. 6018–6024. (Note: Specific conference name missing)

[24] Q. Zhang, W. Sun, H. Guo, C. Dong, and H. Zheng, “A Transfer Learning-Enhanced Generative Adversarial Network for Downscaling Sea Surface Height through Heterogeneous Data Fusion,” *Remote Sens.*, vol. 16, p. 763, 2024. [Online]. Available: https://doi.org/10.3390/rs16050763

[25] H. Hu, W. Zhao, W. Zhou, and H. Li, “SignBERT+: Hand-Model-Aware Self-Supervised Pre-Training for Sign Language Understanding,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, pp. 11221–11239, 2023. [Online]. Available: https://doi.org/10.1109/TPAMI.2023.3269220

[26] S. Kim and H. Kim, “Deep Explanation Model for Facial Expression Recognition Through Facial Action Coding Unit,” in *Proc. 2019 IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, 2019, pp. 1–4. [Online]. Available: https://doi.org/10.1109/BIGCOMP.2019.8679370

[27] B. Hung and L. Tien, “Facial Expression Recognition with CNN-LSTM,” in *Proc. Conf.*, 2021, pp. 549–560. [Online]. Available: https://doi.org/10.1007/978-981-15-7527-3_52

[28] S. Lee, D. Han, and H. Ko, “Multimodal Emotion Recognition Fusion Analysis Adapting BERT With Heterogeneous Feature Unification,” *IEEE Access*, vol. 9, pp. 94557–94572, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3092735

[29] T. Xu, J. White, S. Kalkan, and H. Gunes, “Investigating Bias and Fairness in Facial Expression Recognition,” *arXiv preprint*, arXiv:2007.10075, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-65414-6_35

[30] S. Ramachandran and A. Rattani, “A Self-Supervised Learning Pipeline for Demographically Fair Facial Attribute Classification,” in *Proc. 2024 IEEE Int. Joint Conf. Biometrics (IJCB)*, 2024, pp. 1–10. [Online]. Available: https://doi.org/10.1109/IJCB62174.2024.10744477

[31] F. Acheampong, H. Nunoo-Mensah, and W. Chen, “Transformer models for text-based emotion detection: a review of BERT-based approaches,” *Artif. Intell. Rev.*, vol. 54, pp. 5789–5829, 2021. [Online]. Available: https://doi.org/10.1007/s10462-021-09958-2

[32] J. Azofeifa *et al.*, “Systematic Review of Multimodal Human-Computer Interaction,” *Informatics*, vol. 9, p. 13, 2022. [Online]. Available: https://doi.org/10.3390/informatics9010013

[33] L. Pereira, A. Salazar, and L. Vergara, “A Comparative Analysis of Early and Late Fusion for the Multimodal Two-Class Problem,” *IEEE Access*, vol. 11, pp. 84283–84300, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3296098

[34] J. Bird, A. Ekárt, and D. Faria, “British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language,” *Sensors*, vol. 20, 2020. [Online]. Available: https://doi.org/10.3390/s20185151

[35] L. Zhou, D. Wu, J. Chen, and X. Wei, “Cross-Modal Collaborative Communications,” *IEEE Wireless Commun.*, vol. 27, pp. 112–117, 2020. [Online]. Available: https://doi.org/10.1109/MWC.001.1900201

[36] L. Kessous, G. Castellano, and G. Caridakis, “Multimodal emotion recognition in speech-based interaction using facial expression, body gesture and acoustic analysis,” *J. Multimodal User Interfaces*, vol. 3, pp. 33–48, 2010. [Online]. Available: https://doi.org/10.1007/s12193-009-0025-5

[37] A. Deb *et al*., “Enhancing Communication for the Deaf and Hard-of-Hearing: A Custom Deep Learning Model-Based Approach for Real-Time Sign Language Recognition and Translation,” in *Proc. 2024 IEEE 12th Region 10 Humanitarian Technol. Conf. (R10-HTC)*, 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/R10-HTC59322.2024.10778790

[38] N. Goldstein and R. Feldman, “Knowledge of American sign language and the ability of hearing individuals to decode facial expressions of emotion,” *J. Nonverbal Behav.*, vol. 20, pp. 111–122, 1996. [Online]. Available: https://doi.org/10.1007/BF02253072

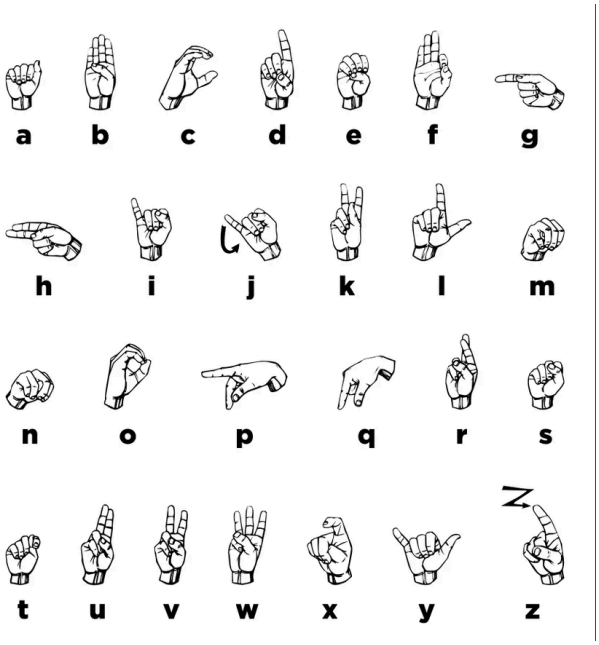
[39] C. Krause, “Brief Report: What You See Is What You Get? Sign Language in the Mathematics Classroom,” *J. Res. Math. Educ.*, vol. 50, no. 1, 2019. [Online]. Available: https://doi.org/10.5951/JRESEMATHEDUC.50.1.0084

[40] T. Nguyen and S. Ranganath, “Facial expressions in American sign language: Tracking and recognition,” *Pattern Recognit.*, vol. 45, pp. 1877–1891, 2012. [Online]. Available: https://doi.org/10.1016/j.patcog.2011.10.026

8. Appendix

Appendix A

A.1 ASL Fingerspelling (A–Z): Standard Hand Gestures Used by the DHH Community



A.2 FER-2013 Dataset Overview for Facial Emotion Recognition

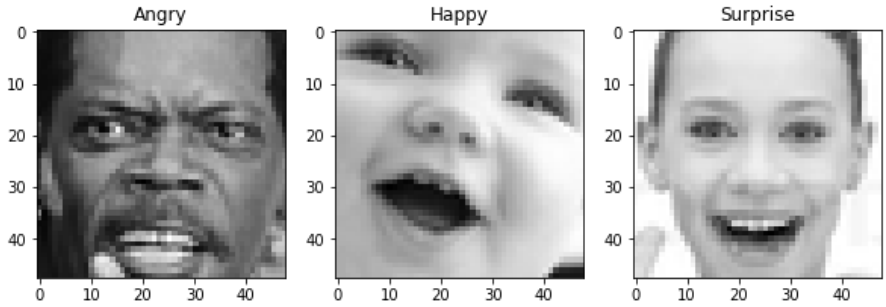


Figure A.2 – Sample grayscale images from the FER-2013 dataset labeled with three of five emotions: Angry, Happy, Sad, Surprise, and Neutral. Images are 48×48 pixels in size.

A.3 Haar Cascade Feature Visualizations for Face Detection

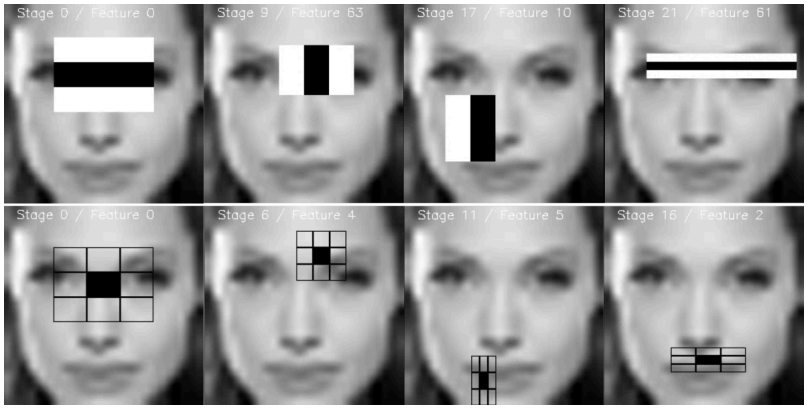


Figure A.3 – Visual examples of Haar-like features used by OpenCV's Haar Cascade classifier to detect key facial regions. Each block represents a feature evaluated during detection stages — such as edge, line, and center-surround patterns — on various areas of the face, such as the eyes, nose, and mouth. These binary templates are convolved over regions in the image to identify contrast patterns that the grayscale frame relies on for detecting facial structures.

Appendix B

B.1 Full System Architecture Flowchart for ASL, Emotion, and Sentiment Integration

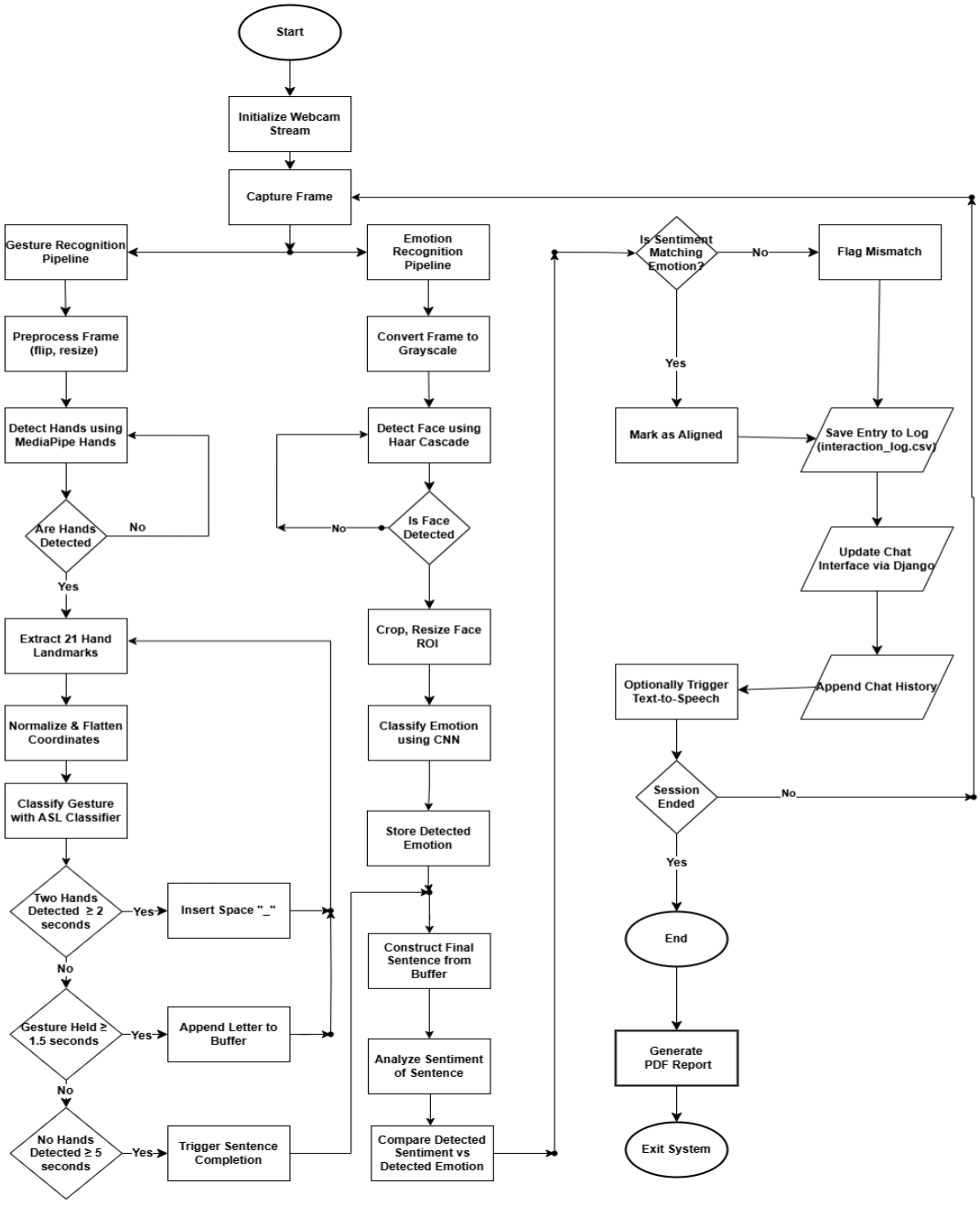


Figure B.1– This flowchart outlines the entire pipeline—from webcam stream initiation to session report generation. It highlights how data flows between gesture and emotion modules and demonstrates conditional logic such as sentence finalization, sentiment mismatch handling, and UI updates.

B.2 ASL Alphabet Classification Metrics (Precision, Recall, F1-Score per Letter)

Letter	Precision	Recall	F1-Score	Support	Letter	Precision	Recall	F1-Score	Support
A	**0.8**	**0.93**	**0.86**	**1051**	**N**	**0.81**	**0.87**	**0.84**	**373**
B	**0.81**	**0.94**	**0.87**	**318**	**O**	**0.81**	**0.93**	**0.87**	**129**
C	**0.87**	**0.83**	**0.85**	**150**	**P**	**0.9**	**0.89**	**0.89**	**158**
D	**0.96**	**0.68**	**0.79**	**128**	**Q**	**0.85**	**0.91**	**0.88**	**163**
E	**0.86**	**0.9**	**0.88**	**553**	**R**	**0.69**	**0.9**	**0.78**	**327**
F	**0.84**	**0.88**	**0.86**	**251**	**S**	**0.91**	**0.86**	**0.88**	**389**
G	**0.88**	**0.95**	**0.91**	**657**	**T**	**0.95**	**0.72**	**0.82**	**302**
H	**0.97**	**0.92**	**0.94**	**302**	**U**	**0.9**	**0.46**	**0.61**	**287**
I	**0.98**	**0.95**	**0.96**	**235**	**V**	**0.9**	**0.87**	**0.88**	**293**
J	**0.91**	**0.95**	**0.93**	**353**	**W**	**0.95**	**0.83**	**0.88**	**338**
K	**0.84**	**0.83**	**0.83**	**489**	**X**	**0.94**	**0.86**	**0.9**	**281**
L	**0.82**	**0.72**	**0.77**	**360**	**Y**	**0.99**	**0.82**	**0.9**	**531**
M	**0.75**	**0.77**	**0.76**	**324**	**Z**	**0.86**	**0.9**	**0.88**	**359**

Table B.2 – Each Letter represents its classification metrics in the format (Letter, Precision, Recall, F1-Score). For example, (A, 0.80, 0.93, 0.86) means the model identified the letter "A" with 80% precision, 93% recall, and an F1-score of 86%. This tabular format offers a quick overview of model performance across all 26 ASL alphabet classes.

B.3: Confusion Matrix for ASL Gesture Classification

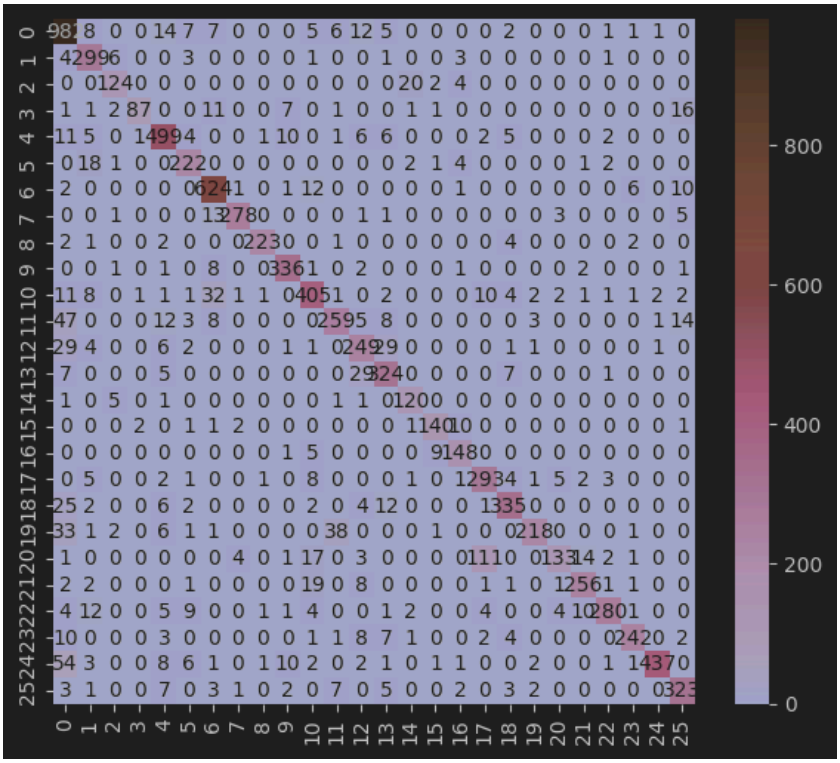


Figure B.3 – *Visual matrix showing classification accuracy and cross-class misidentifications for all 26 alphabet gestures. The full report aids in identifying consistent sources of model error and potential improvements.*

B.4 Confusion Matrix of Emotion Classifier (FER-2013, 5-Class Output)

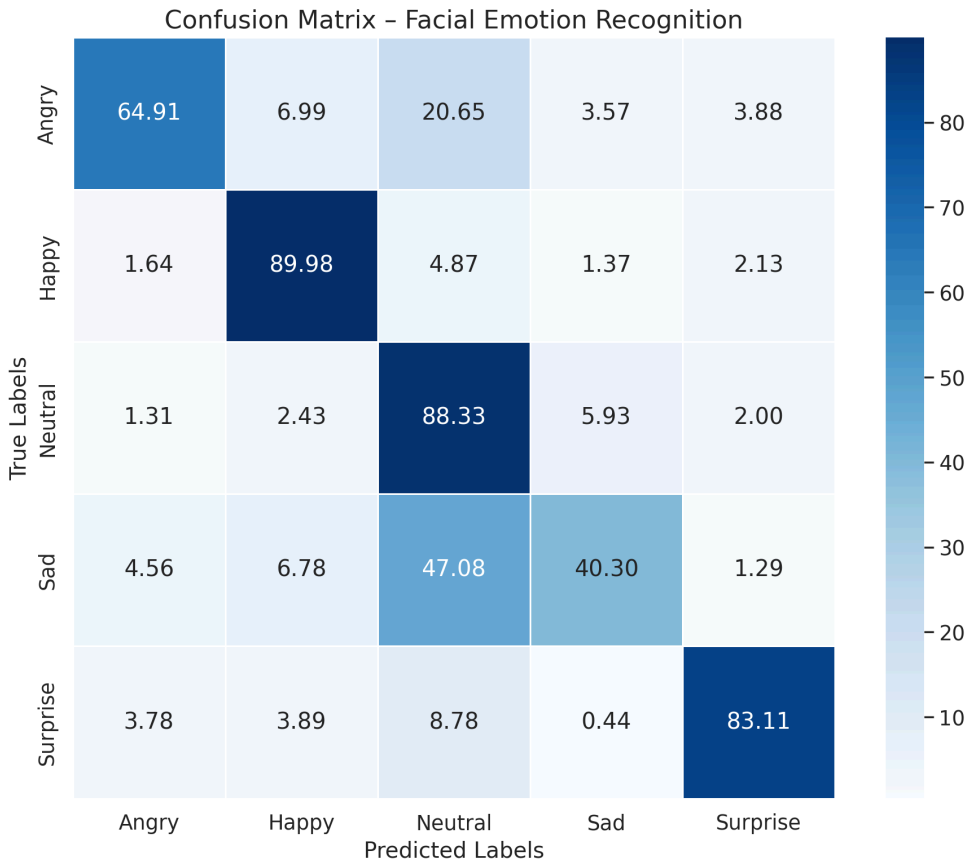
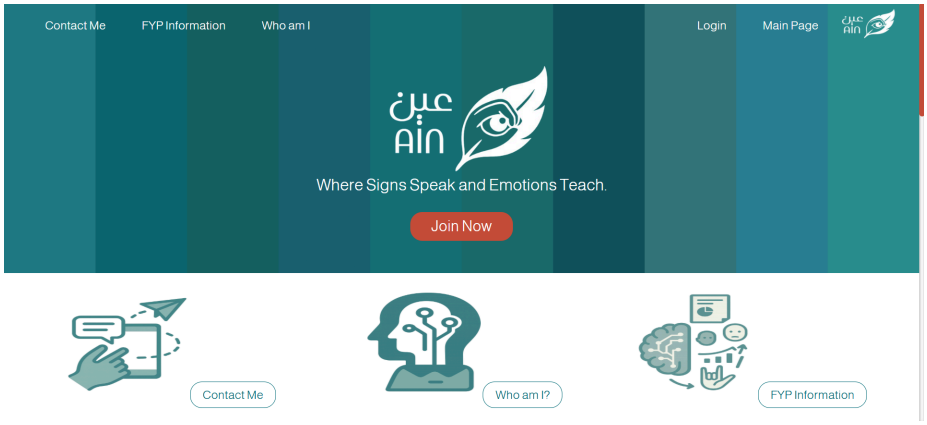


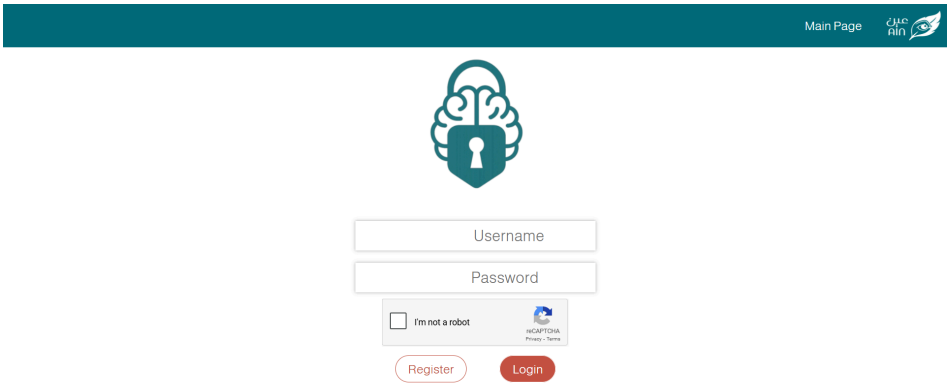
Figure B.4 – This normalized matrix reflects true vs. predicted classifications across five mapped emotion categories. Strong diagonals indicate robust accuracy in happy and surprise classes. Higher misclassification in "sad" and "angry" toward "neutral" highlights subtle facial overlaps and boundary fuzziness in affective states. (True labels on the Y-axis, predicted labels on the X-axis, values as percentage of class occurrences.)

Appendix C

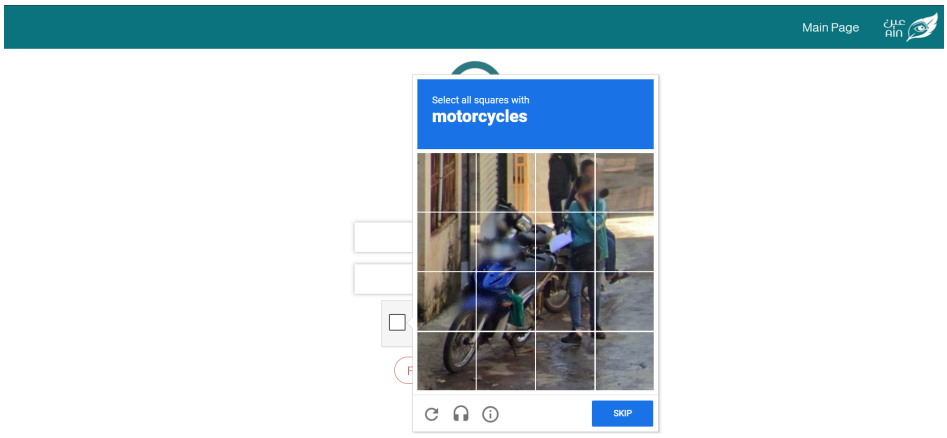
C.1 The *Home* Page



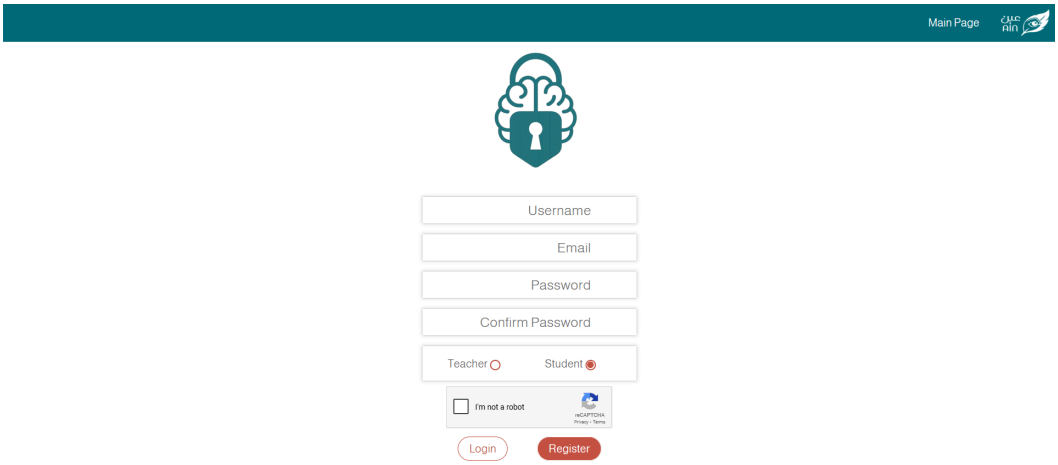
C.2 The *Login* Page



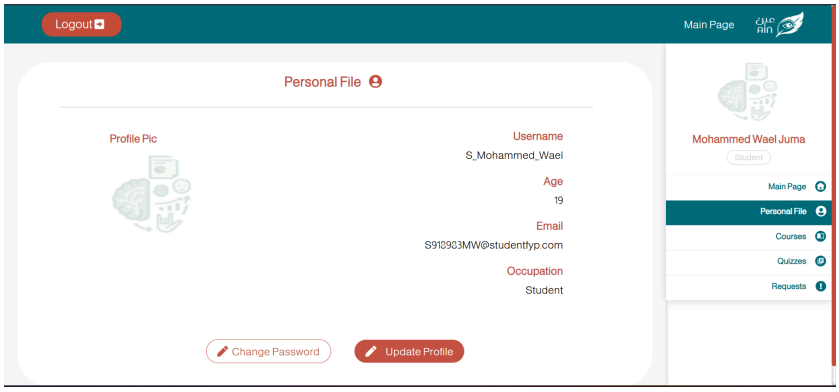
C.2.1 *ReCAPTCHA* by Google



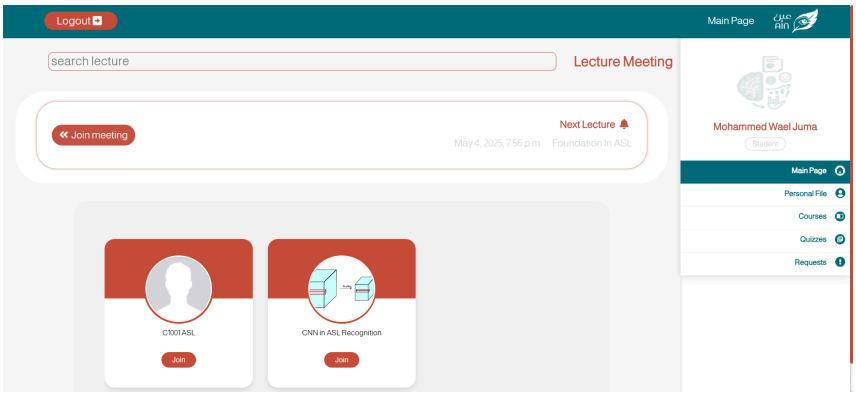
C.3 The *Register* Page



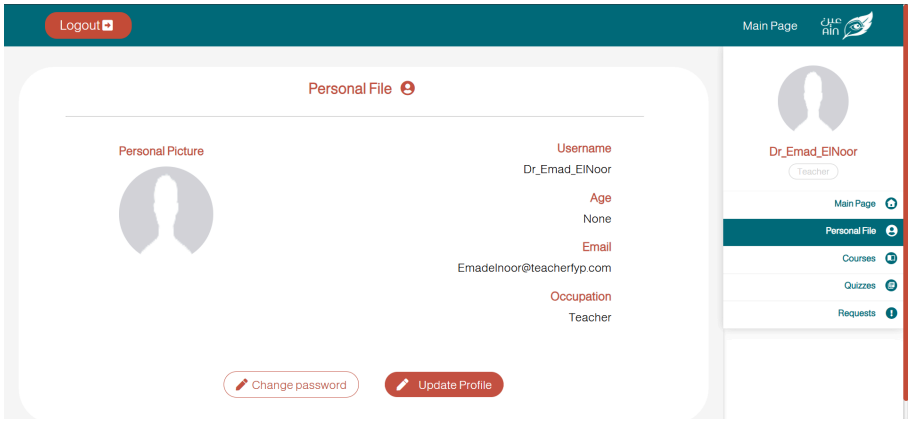
C.4 The *Student Profile* Page



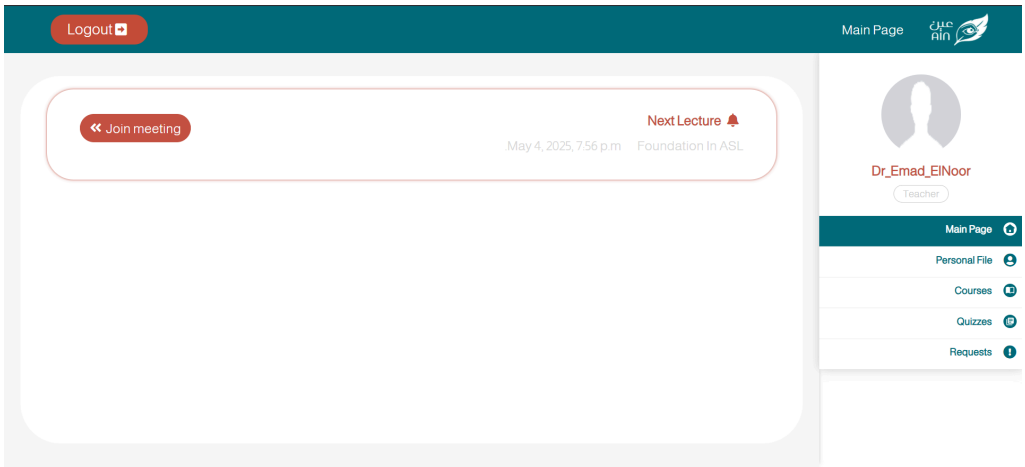
C.5 The *Student Dashboard* Page



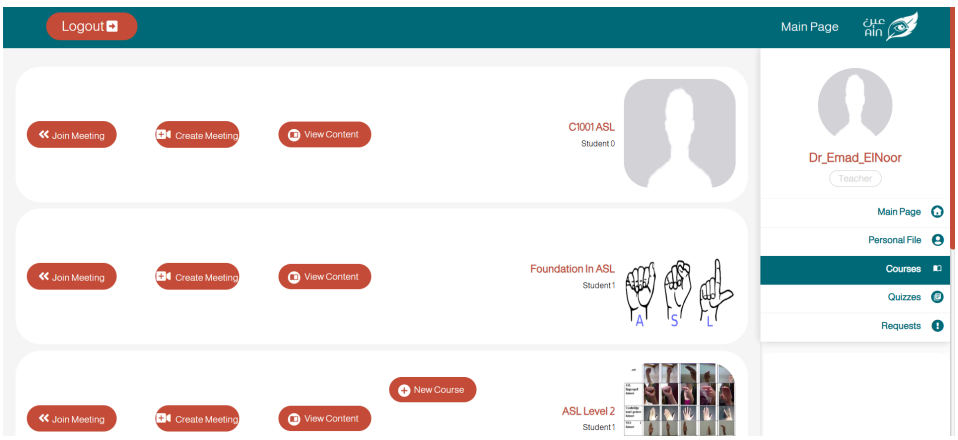
C.6 The *Teacher Profile* Page



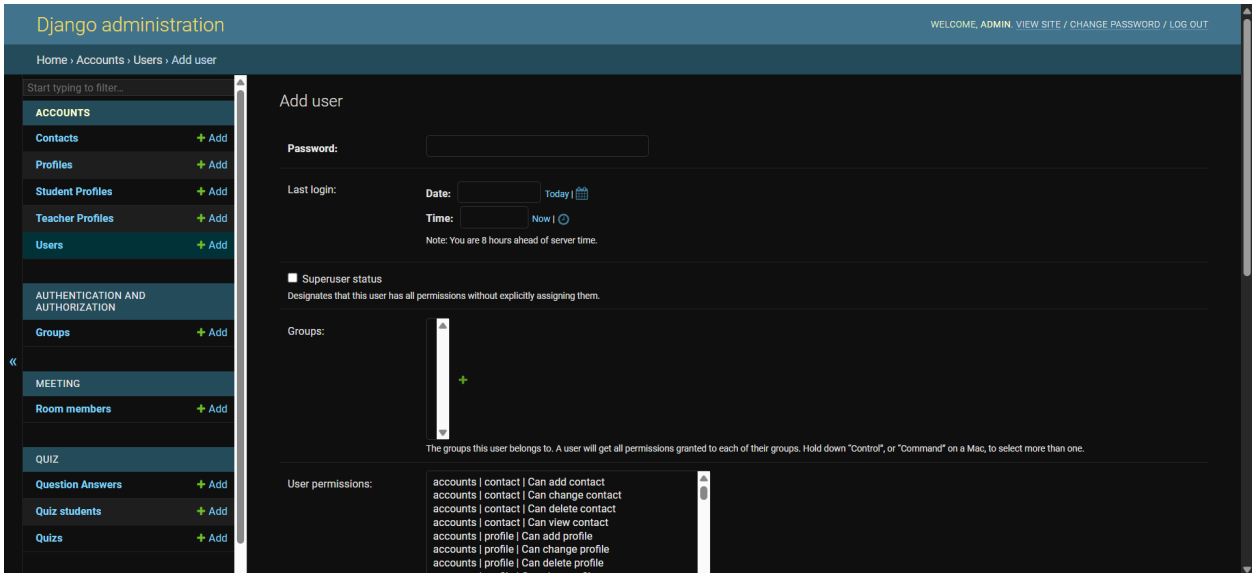
C.7 The *Teacher Dashboard* Page



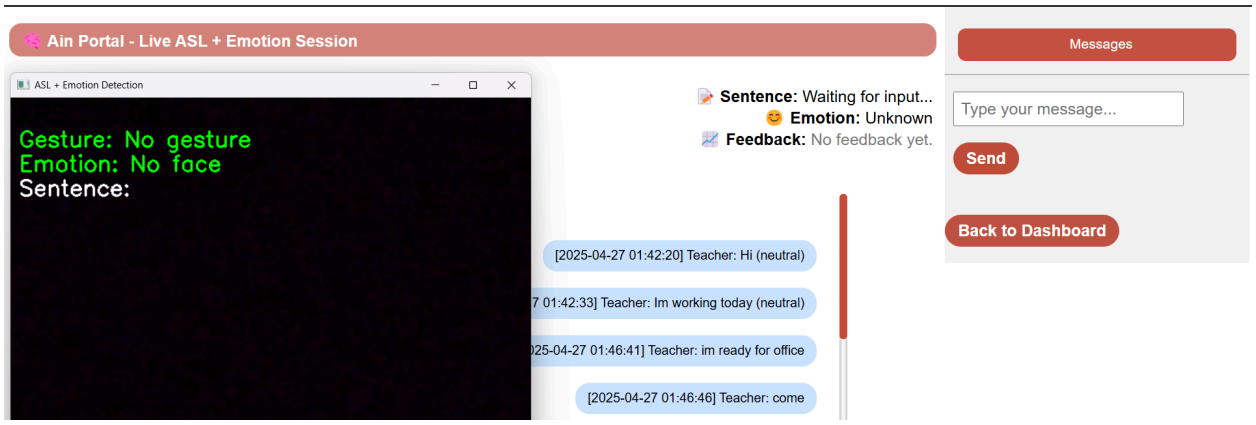
C.8 The *Teacher Courses* Page



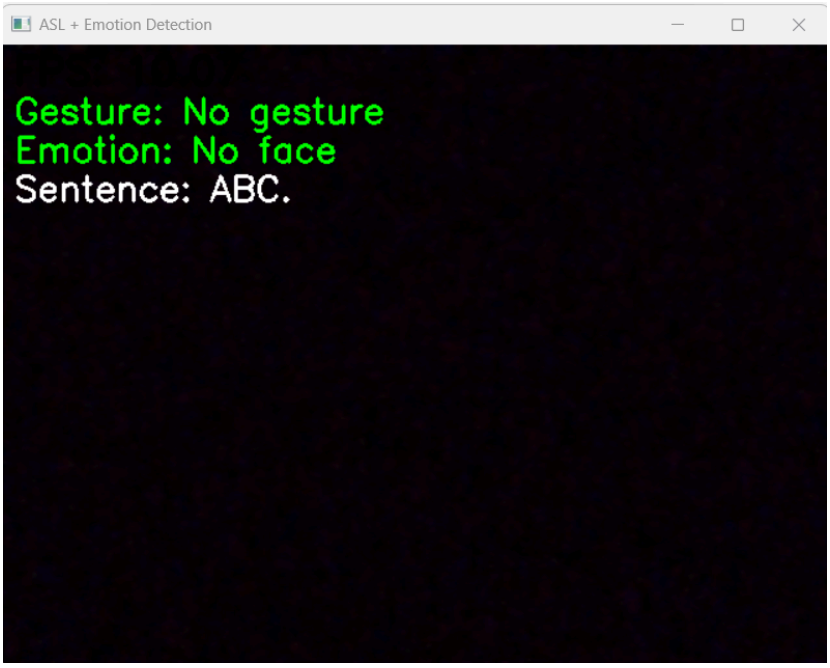
C.9 The *Admin* Page



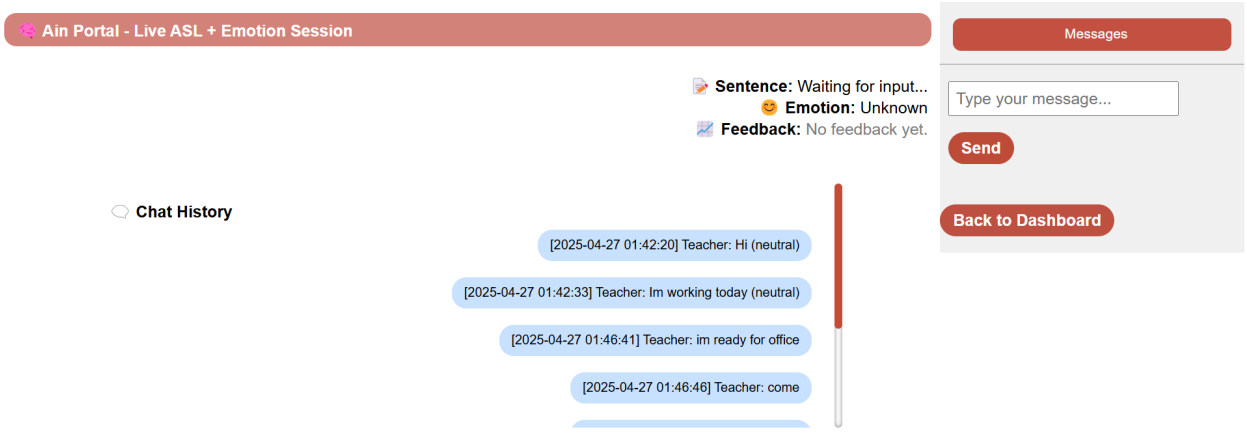
C.10 The *meeting room* Page



C.10.1 The *Api View*



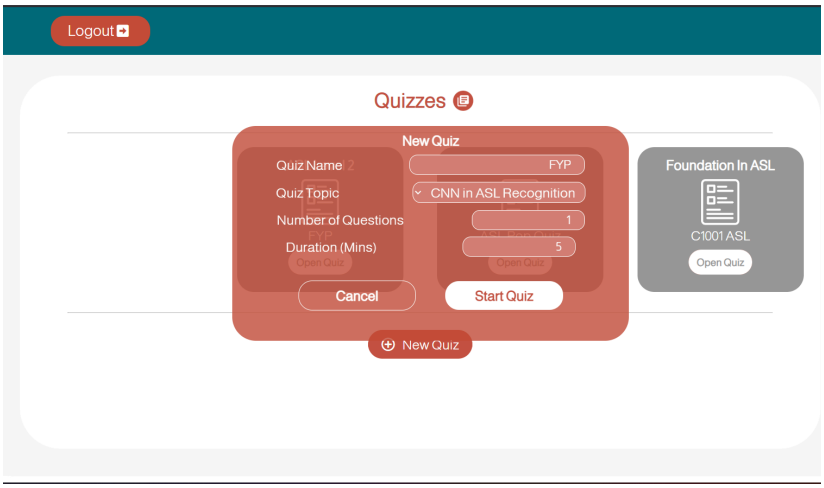
C.10.2 The *Room View*



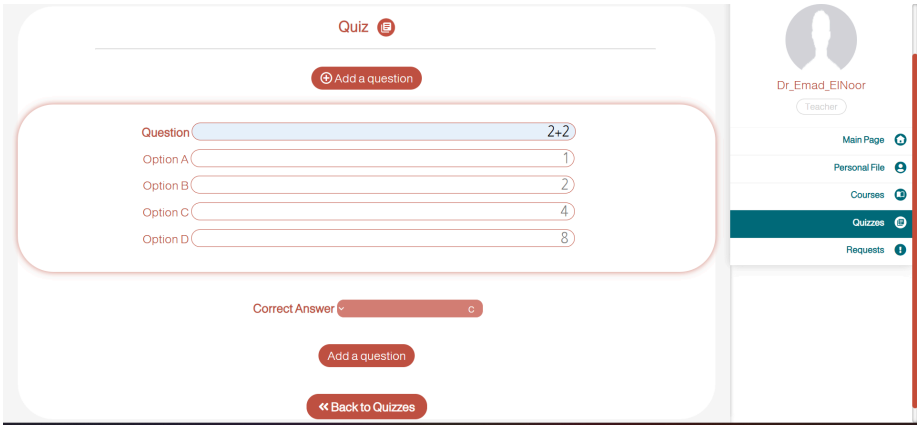
The Chat History has a mix of messages between signed statements like "Hi (neutral)" and written messages such as "come"

C.11 The Quiz

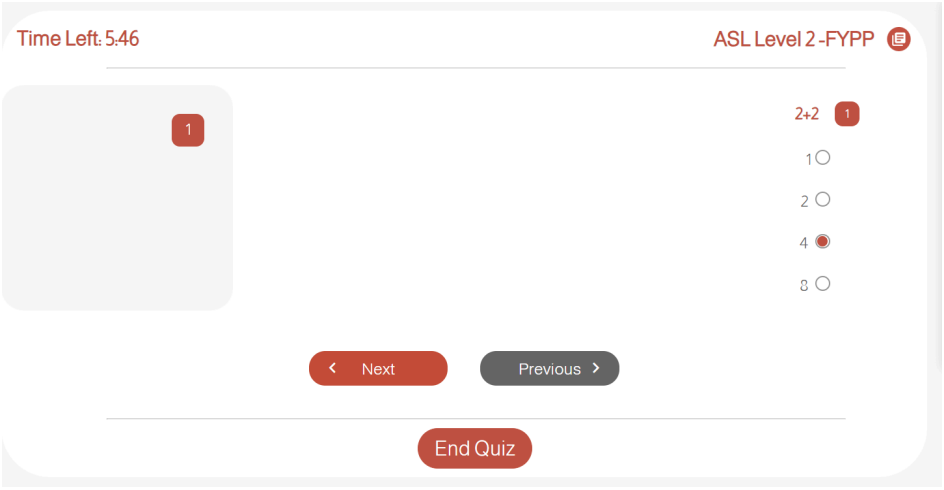
C.11.1 Teacher's Quiz Creation



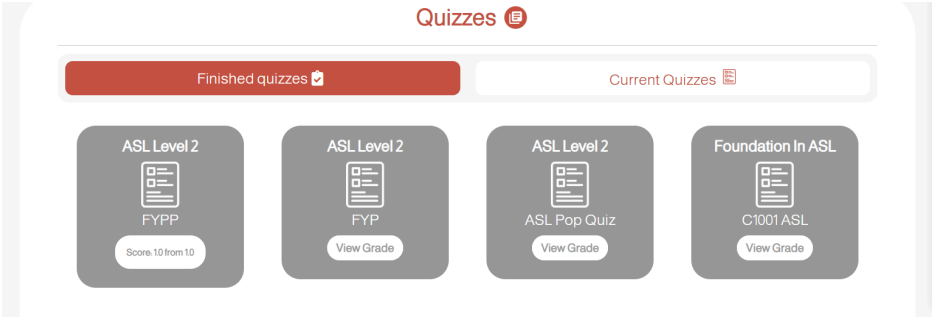
C.11.2 Teachers's Question Creation



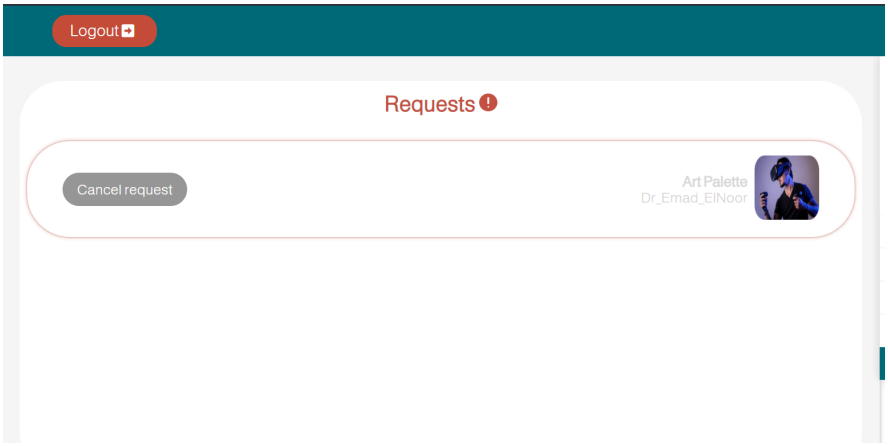
C.11.1 Student's Quiz Solving



C.11.2 Student's Past Quizzes



C.12 Student's Joining Course Request



C.13 Teachers's Approval or Rejection of the Request