The British College_CoverSheet.docx

by Sameer Basnet

Submission date: 16-May-2025 07:21PM (UTC+0545)
Submission ID: 2677534063
File name: The_British_College_CoverSheet.docx (27.38M)
Word count: 8916
Character count: 59740

LEEDS BECKETT UNIVERSITY

The British College

KATHMANDU

Coursework Submission Coversheet

(individual coursework exclusively)

Faculty of Arts, Environment and Technology

The British College

LBU Student Id: 77356702

FOR CHECKING BY THE STUDENT:

Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.

Award name: BSc (Hons) Computing

Module code: COMP607

Module name: Production Project

Module run:

Coursework title: Breaking the Slience: AI and Computer Vision Driven Sign Language Translation System

Due Date: 16-may-2025

Module leader: (In LBU)

Module Supervisor: (In TBC) Mr. Rohit Raj Pandey

TURNITIN Checked: YES NO *(please circle)*

Submission date & time: Date: 16-may-2025 Time: 23:59

Total Word Count: 9110 **Total Number of Pages (including this front sheet): 94**

In submitting this form with your assignment, you make the following declaration:

I declare, that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at LBU or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.

Signed: Sameer Basnet Date: 03-may-2025

Teacher's Signature: _____ **Date:** _____

Breaking the Slience: AI and Computer Vision Driven Sign Language Translation System

Module: Production Project

Bsc. Hons. Computing

Level 6: 2nd semester

Abstract

"Breaking the Slience: AI and Computer Vision Driven Sign Language Translation System" introduces an assimilated, bidirectional platform that sways state-of-the-art deep overcoming and hand-tracing to bridge communication voids betwixt Deaf and hearing individuals. At its mecca, the system pertains MediaPipe Hands for real-time extraction of 21 three-dimensional landmarks per frame, catering these 63-dimensional vectors into an attention-augmented Long Brief-Term Memory (LSTM) network. By tiling exclusive-frame landmarks into ten-indicator sequences, the model masters temporal gesture dynamics—rudimentary for differentiating motion-predicated gestures like "J" and "Z"—while its self-attention block emphasizes the virtually instructive frames. Trained and validated on a stratified ASL dataset substituting 36 classes, the network accomplishes atop 99 % test accuracy, balanced accuracy, and top-3 recall, with ***ideal*** calibration (dependability diagrams) and AUC scores, substantiating its vigorousness and interpretability.

The system's architecture constitutes two foremost pipelines. Initial, the **ASL-to-English** workflow (Figure 10) grabs webcam intake in the browser, buffers landmark sequences, and transmits them through AJAX to Django endpoints. Predictions are logged, stowed, and responded as the top three labels with confidence scores, which the frontend exhibits in real time. Upon session culmination, a consecutive endpoint prompts a large-language model (e.g., Gemini API) to produce articulate English sentences, stowed alongside audit logs and introduced with discretionary text-to-speech. Consecutive, the **English-to-ASL** pipeline (Figure 11) transmutes typed text into a stationary letter-mapping video: sanitized intake is analyzed into individual characters, mapped to PNG images, converged into frames through OpenCV, and encoded into MP4 utilizing imageio. Metadata and user activity are logged, and the resultant video URL is responded for in-browser preview and download.

Staging these characteristics is a normalized relational schema (Figures 13 & 14) that stretches Django's authentication and permission tables with domain-specific models: **asl_aslprediction**, **asl_aslsentencegeneration**, **asl_aslvideohistory**, and **asl_auditlog**. This design assures referential morality, inspection, and assistance for soft deletion of video resources. Utilized-case diagrams (Figures 14–16) reveal three actor roles—end users, administrators, and the email platform—collaborating with the system boundary amid registration, verification, prediction, translation, and history management functions.

Methodologically, the project pursues an Agile, sprint-driven technique (Segment 4), letting iterative progression, continual testing on criterion such as latency and vigorousness beneath contrasting lighting conditions, and user-centered interface advancements. Literature retrospect emphasizes the progression of Sign Language Recognition (SLR) out of hardware-dependent gloves to modernistic deeplearning systems, intriguing our preference of featherlight landmark intakes atop raw video and upholding the attention-LSTM hybrid atop chastely convolutional or transformer models.

Real-world usages indicator assistive communication in healthcare and education, generic pavilions for Deaf travelers, live event captioning, and mobile AR incorporations, entire aiding out of the system's low latency and high accuracy. Confines—such as stationary video generation for sophisticated grammar and perceptiveness to occlusion—escort subsequent work concerning

multimodal amalgamation of facial cues, transformer-predicated encoders, and supplemented datasets. Broadly, this comprehensive framework demonstrates how AI and computer vision can produce sign language both observable and audible, fostering panoramic interaction throughout distinctive contexts.

Table Of Content

Table Of Figure

1. Introduction:

The project, titled "Unraveling the Silence: AI and Computer Vision Driven Sign Language Translation System," concentrates in utilizing evolved AI and computer vision to translate sign language into both English text and speech, thereby constricting communication voids for deaf and mute individuals.

At its essence, the project negotiates the captious challenge that obstruct the social and professional incorporation of sign language users. By situating gesture recognition techniques pertaining frameworks like TensorFlow and PyTorch, alongside with hand tracing models such as Mediapipe, the system is contemplate to precisely detect and interpret hand gestures in real time. A foremost aspect of the system is its user-friendly interface, which is manifested to assure accessibility for users with definite technical knowledge. Furthermore, the harmonizing of a text-to-speech module enhances the system's functionality by equipping audible output, farther assisting potent communication.

The system assimilates a real-time error negotiating mechanism to assures elegant operation by promptly cautionary users when a gesture is not distinguished. This mecca on dependability and perceptiveness is crucial for pragmatic, everyday utilized. Urging upon earlier research in sign language recognition—which distinguished out of hardware-dependent data gloves to modernistic deep overcoming methods—the system endeavors to subdue the objections posed by energetically and continuing sign interpretation. It concentrates on urging a scalable solution that accomplices imposed beneath assorted conditions, such as contrasted lighting and hand sizes.

Ultimately, the report corroborates not exclusively the technological inventions in back of the system besides also its feasible social jolt. By letting clearer and farther comprehensive communication, the project is composed to tremendously enrich the trait of life for sign language users, letting them to emerge farther wholly in educational, professional, and social surroundings.

2. Literature Retrospect

Sign language Detection (SLR) pertain to the procedure of deciphering and explanning gestures, hand movements and body language utilized in sign language to retain communication betwixt people who utilized sign language and who do not. The field of SLR has witnessed outspoken breakthroughs atop years, gazing out of neural network-predicated systems to deep overcoming predicated systems. Primitive examined two stage neural network that utilizes a DataGlove for phoneme-level recognition, achieving 86% accuracy but was curbed by hardware depandance work (Kim, S. et. al., 1995). Another research utilized self-organizing frameworks like SHOSLIF-M, enhancing spatiotemporal recognition with a 96% accuracy rate. Nevertheless, its reliance on tailored-built characteristics and definite scalability to real-life scenarios imposed exceptions (Cui, Y. et. al., 1995). Another Research mecca on stationary, secluded, exclusive-handed gestures utilizing camera-predicated systems, featuring a lack of organized datasets and a need for dynamic sign recognition breakthroughs (Wadhawan, A. et. al., 2021). The utilized of CNNs showed odd interpretation with 99.90% accuracy for stationary gestures utilizing comprehensive datasets, emphasizing the stability of deep overcoming (Kumar, P. et. al., 2020). Utilized of statistical methods for continuous sign language handle real-life arbitrariness, situating multimodal characteristics like facial landmarks and achieving uncovering word error rate reductions (Koller, O. et. al., 2015). Another Study utilized statistical methods and fenones for potent sign recognition but strumbled with sturdy subunit definition for contrasted gestures (Bauer, B. et. al., 2002). Another study improved a real-time sign language interpreter utilizing a data glove and HMMs but met exceptions with endored reliance and definite accuracy (Ouhyoung, M. et. al., 1998).

3. Retrospect Of Technology:

The project adopts a methodical and research-driven technique, commencement with an comprehensive retrospect of existent sign language recognition (SLR) technologies. This initial research aids distinguish confines in past systems and advises the selection of competent models and frameworks for the project. A crucial primitive indicator of sign analysis is the choice of dataset, such as the American Sign Language (ASL) dataset or urging a tailored dataset with both stationary and dynamic gestures. These datasets shape the cornerstone for training machine overcoming models competent of deciphering a broad scope of gestures. For gesture detection, the project utilizes an responsible pre-trained models such as Mediapipe for real-time hand tracing, while TensorFlow and PyTorch are used to build and train classification models using Convolutional Neural Networks (CNNs) for stationary gestures and undoubtedly Recurrent Neural Networks (RNNs), LSTMs, or Transformers for dynamic gesture recognition.

Each system constiting of scaled-up and tested individually to assure interpretation, responsiveness, and dependability earlier comprehensive incorporation. Predominant criterion such as accuracy, latency, and vigorousness in varied environmental conditions—like alterations in lighting and hand orientation—are monitored throughout the testing procedure.

This modular and iterative progression strategy empowers proceeding enhancement at every stage and assures the final system can enforce consistently in real-world scenarios. The methodology prioritizes adaptability and modularity, letting for primitive identification and resolution of consequences while enhancing model interpretation predicated on observed results and user feedback.

To enhance accessibility, the project assimilates a Text-to-Speech (TTS) module utilizing the pyttsx3 library, letting real-time transformation of distinguished gestures into spoken English. A smooth, intuitive user interface is also anticipated, assuring the system can be utilized by individuals with definite technical expertise. The system is deliberate to furnish suggestive feedback throughout utilized, farther enhancing the user experience. This comprehensive and iterative methodology, grounded in continual testing, user-centered design, and flexible progression, assures that the final product will be accurate, panoramic, and efficient in aiding deaf and mute individuals communicate farther effortlessly with others.

4. Methodology:

To address the elaborateness and evolving necessities of the project, an Agile methodology is embraced to escort the progression procedure. Agile is notably well-suited for usages intriguing artificial intelligence and computer vision, where continual testing, model refinement, and user feedback portray a crucial role. By breaking the project into small, manageable iterations or sprints, the we can can focus on integrating and evaluating individual components—For example hand tracing, gesture recognition, and text-to-speech translation—earlier incorporating them into the larger system. This allows for eariler identification of consequences, proceeding breakthroughs, and flexibility to tailor the project predication on testing results or technical challenges.

Every sprint in the Agile workflow encompasses stages of planning, progression, testing, and retrospect. For example, a staunch sprint may mecca on optimizing gesture classification utilizing TensorFlow or PyTorch, where interpretation is assessed amid criterion like recognition accuracy and processing speed. When the component is validated, the next sprint will evolve incorporating it with the hand-tracing system backed by Mediapipe. Feedback will be gathered throughout every sprint—whether out of testing or user evaluations—is utilized to enhance both the functionality and user experience of the system. These repetitive cycle ensures that the final product is not only exclusively accurate and responsive but also functional and convenient in real-world conditions.

To assist the Agile technique, the project utilizes planning tools such as Gantt charts and visual timelines to delineate milestones, constite deadlines, and monitor progress throughout the project development lifecycle. These tools furnish structure and clarity while letting room for the iterative nature of Agile progression. Collaboration platforms like GitHub are utilized for version control, while Google Drive and Microsoft Teams assist with documentation and team coordination. By fusing the adaptability of Agile with efficient planning and communication tools, the project assures that progression remains engrossed, collaborative, and responsive to remodel—resultant in a robust and panoramic sign language translation system.

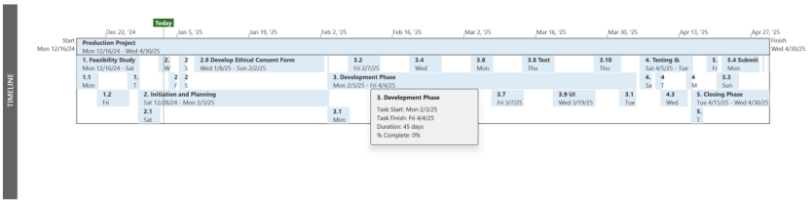


Fig - 1: *The above Project Timeline exhibits the project planning along with the dates.*

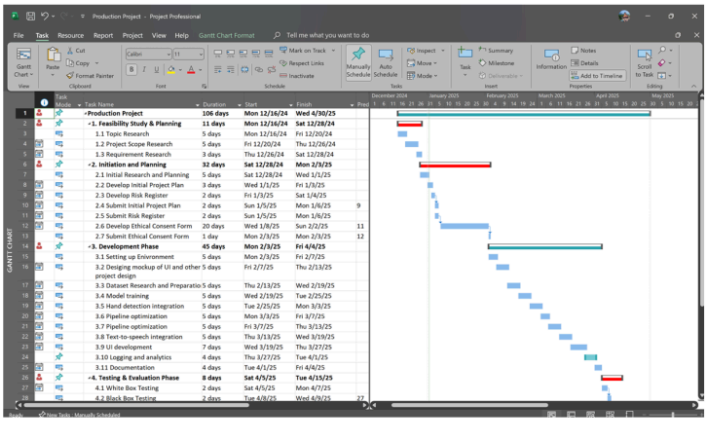


Fig - 2: *The above Gantt chart exhibits the project planning along with the predominant dates.*

5. System Design and Implementation of the ASL Translation Platform

This segment introduces the end-to-end architecture and engineering of our ASL translation system. It commences by intriguing an attention-augmented LSTM for robust gesture recognition, then strolls amid the model's training, evaluation, and real-time prediction workflows (encompassing sentence and video production). We then detail the database schema and ERDs, utilized-case diagrams, three-tier system architecture, comprehensive testing strategy, deployment and implementation considerations, stakeholder communication plan, MoSCoW-prioritized product specifications, and the hardware/software resources required.

5.1 Rationale for an Attention-Augmented LSTM Architecture in ASL Gesture Recognition

In choosing an appropriate model for American Sign Language (ASL) gesture recognition, three mecca necessities must be satisfied: (1) efficient modeling of temporal dynamics, (2) computational efficiency for real-time deployment, and (3) vigorousness to intra-class interpretation. An LSTM (Long Brief-Term Memory) network augmented with an attention mechanism fulfils these necessities farther inherently than discretionary architectures.

Initial, ASL gestures are inherently temporal. Numerous gestures—such as "J," "Z," or compound gestures—are definite not by a exclusive stationary handshape but by a sequence of movements. LSTMs surpass at grabbing long-scope dependencies in sequential data appreciation to their constricted memory cells, which master when to recall or forget information throughout time steps. By proselytizing every frame into a normalized 63-dimensional landmark vector (21 landmarks × 3 coordinates) and repeating it to shape a synthetic sequence, the model pretends motion patterns even when processing serene images. This depiction permits the LSTM to master subtle alterations in wrist situate and finger communal angles that distinguish analogous stationary poses.

Nevertheless, not entire frames within a sequence contribute equivalently to sign discrimination. Stationary CNNs or smooth dense networks address every intake equivalently, adulterating the jolt of the virtually instructive frames. By incorporating an attention layer on top of the initial LSTM layer's outputs, the model masters to entrust evolved magnitude weights to captious frames—such as the moment of farthest finger flexion or directional turnabout in a sweeping gesture—while suppressing lesser pertinent frames. This weighted accumulation not exclusively accelerates broadly classification accuracy but also enhances interpretability, as one can envision attention scores to derive which temporal segments the network relied upon.

Discretionary sequence models—such as pure Transformer architectures—pose influential self-attention throughout entire time steps, but they incur quadratic elaborateness in sequence length and claim substantially farther data to evade overfitting. For moderate-sized ASL datasets, this can steer to memory congestions and precarious training. In distinction, an LSTM with a featherlight attention block influences a balance: recurrence negotiates sequence modeling linearly with regard to time, and attention affixes exclusively a moderate overhead to distinguish salient frames.

Furthermore, landmark-predicated intakes are orders of magnitude farther compacted than raw video or image patches, theatric diminishing computational and memory necessities. A 63-element vector per frame equated to a 224×224×3 RGB tensor permits the model to be deployed on resource-constrained devices—such as embedded processors or smartphones—while serene achieving high accuracy. This efficiency is captious for real-time usages where latency and power consumption are restricting factors.

Ultimately, regularization techniques—encompassing L2 weight wraths on LSTM kernels and withdraw betwixt layers—mitigate overfitting on classes with fewer examples or greater intra-

class interpretation. Concurrently, these design preferences resultant in a model that is both precise and pragmatic, competent of deciphering dynamic ASL gestures in real time and adaptable to new sign classes with minimal data.

5.2 Model Training Workflow and Implementation

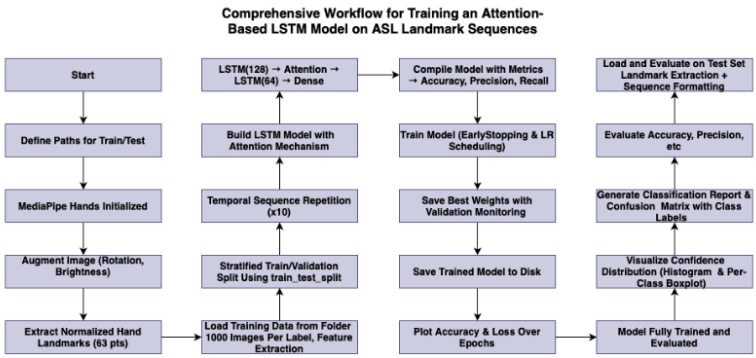


Fig - 3: End-to-End Training Workflow for the Attention-Augmented LSTM ASL Model

In Figure - 3, the comprehensive pipeline for training and evaluating our attention-augmented LSTM network on ASL landmark sequences is depicted. Commencement at the top left, file paths for the train and test image directories are definite and MediaPipe Hands is initialized for landmark detection. Raw images are then preprocessed through random rotation and brightness augmentation earlier enacting amid the *excerpt_landmarks* routine, which resizes every frame to 224×224 pixels and computes a 63-dimensional feature vector (21 landmarks × 3 coordinates), normalized comparative to the wrist.

These landmark vectors are accrued by class (up to 1,000 samples per sign) in *prepare_data_out of_folder*, one-hot encoded, and stratified into 90/10 train/validation cleavages. To shape temporal intakes, every exclusive-frame vector is tiled into a 10-indicator sequence. The model architecture—exhibited in the central column—stacks a 128-unit LSTM layer with L2 regularization, an attention block to weight salient time steps, a 64-unit LSTM, and dense layers closing in a softmax classifier. The network is made up of Adam optimization, categorical cross-entropy loss, and criterion for accuracy, precision, and recall.

Training (right column) pertains EarlyStopping and ReduceLROnPlateau callbacks to save the best weights and tailor the overcoming rate. Upon culmination, the trained model is serialized to

disk. Ultimately, the test constitute endures the identical landmark extraction and sequence formatting; mecca and evolved criterion—encompassing confusion matrices, per-class bar charts, ROC curves, and dependability diagrams—are spawned to wholly assess interpretation.

5.3 Model Evaluation and Accuracy Analysis

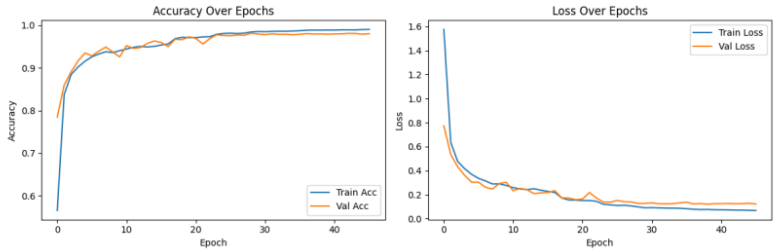


Fig – 4: Training and Validation Accuracy & Loss atop Epochs

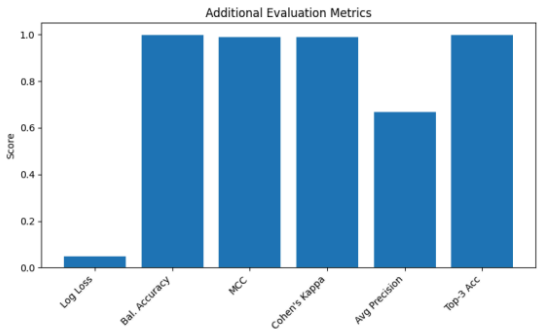


Fig - 5: Summary of Additional Evaluation Criterion

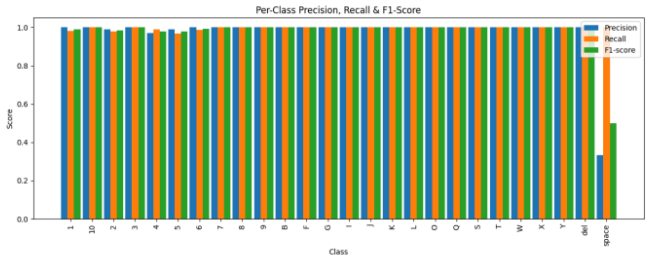


Fig - 6: Per-Class Precision, Recall & F1-Score

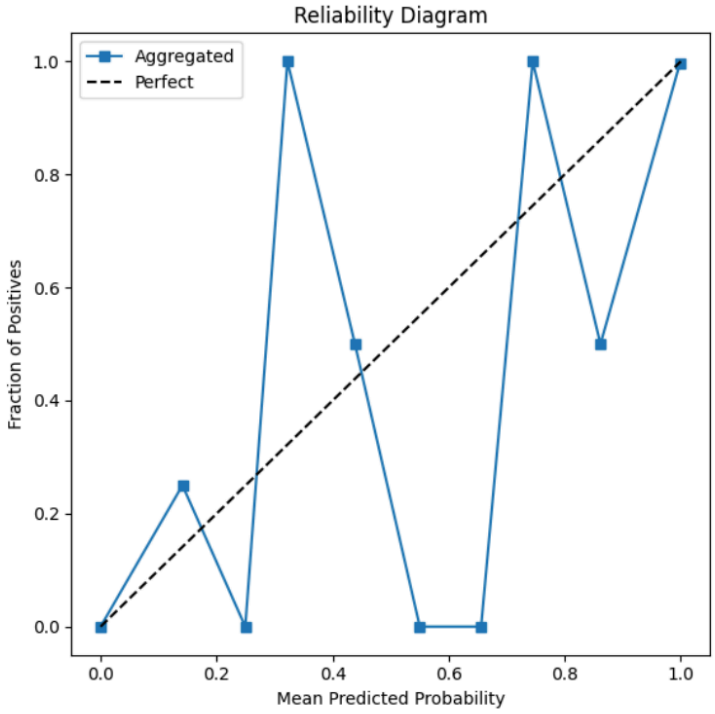


Fig - 7: Dependability Diagram (Calibration Curve)

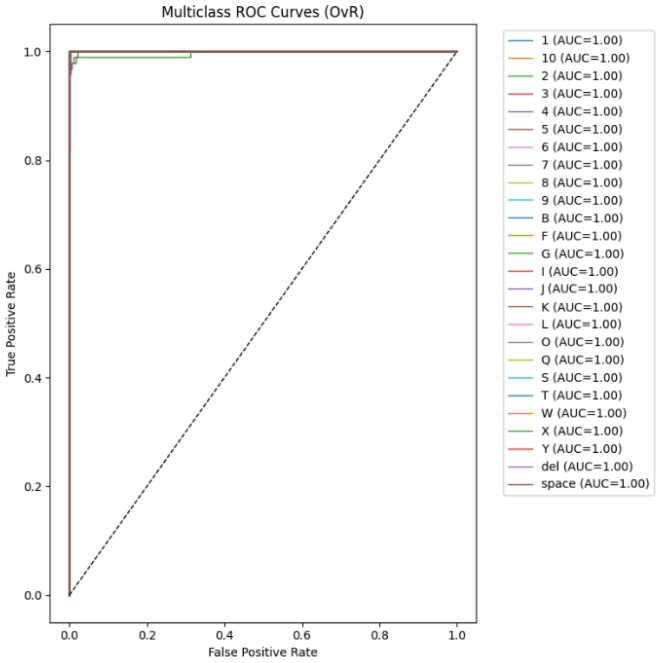


Fig - 8: Multiclass ROC Curves (One-vs-Rest)

The model exhibits swift confluence in both accuracy and loss (Figure 4). At the first ten epochs, training accuracy exceeded 90%, closely mirrored by validation accuracy, suggesting efficient generalization and minimal overfitting. The loss curves similarly denied sharply at the start stabilizing, with the validation loss consistently tracking the training loss.

confusion matrix approves notable class discrimination: virtually gestures stand along the diagonal with exclusively a small amount of off-diagonal errors. Misclassifications occur primarily betwixt visually analogous gestures, but their frequency is nominal comparative to broadly test constitute size.

A bar chart of additional criterion (Figure 6) farther emphasizes interpretation: balanced accuracy, Matthews correlation coefficient, Cohen's kappa, and top-3 accuracy entire exceed 0.99, establishing consistent, responsible predictions throughout classes. The median precision score is

lower (~0.67) exclusively because of the authentic macro averaging atop numerous classes with definite samples, yet even this remains satisfactory.

Figure 7 deciphers down per-class precision, recall, and F1-score, exhibiting near-perfect values (≥0.95) for virtually every sign. This approves the model's vigorousness to intra-class interpretation. Exclusively a diminutive number of unusual classes immerse narrowly below symmetry, reflecting definite test examples preferably than systematic errors.

Calibration is envisioned in Figure 8, where the accrued dependability curve closely pursues the ideal diagonal—implying well-calibrated probability assesses. Ultimately, the ROC curves in Figure 9 entire achieve AUC scores of 1.00, emphasis farthest separability in a one-vs-rest evaluation.

Concurrently, these figures reveal that the attention-augmented LSTM architecture delivers highly accurate, calibrated, and interpretable ASL gesture recognition competent for real-time deployment.

5.4 Real-Time ASL Gesture Prediction and Sentence Production Workflow

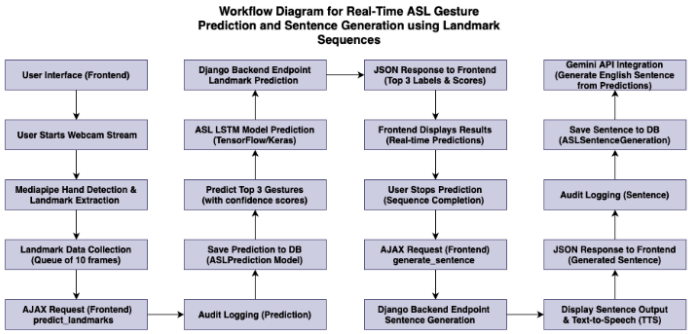


Fig - 9: Real-Time ASL-to-English Translation Pipeline

Figure 9 exemplifies the end-to-end flow that empowers live American Sign Language (ASL) recognition in the browser and automatic English sentence production on the server. The pipeline scales three domains: frontend user interaction, backend deduction, and backend sentence synthesis.

1. **Frontend Initialization & Landmark Prisoner**
    - **User Interface:** The procedure commences when the user launches the web page encompassing the ASL translator.
    - **Webcam Stream Start:** Clicking "Start" triggers the browser's webcam through JavaScript.
    - **MediaPipe Hand Detection:** Every video frame is transmitted to MediaPipe Hands in the client, which locates one hand and extracts 21 three-dimensional landmarks.
    - **Frame Queueing:** Landmarks out of 10 consecutive frames are buffered into a rolling queue, constituting the minimal temporal context for dynamic gestures.
2. **AJAX Prediction Request & Logging**
    - **AJAX `predict_landmarks` Call:** Every ten-frame batch triggers a POST request hauling the normalized landmark data to the Django backend endpoint `/predict_landmarks`.
    - **Audit Logging (Prediction):** Earlier deduction, the server logs user ID, timestamp, and raw landmark intake to an "ASLPrediction" audit table for traceability.
3. **Backend Model Description**
    - **Saving Prediction to Database:** The identical raw landmark batch is also stored in the prediction history table.
    - **LSTM+Attention Model Prediction:** The serialized landmarks sequences are feed into the pre-loaded TensorFlow/Keras model, which generates the class probabilities for each of the ASL gestures feed to the model.
    - **Top-3 Gesture Selection:** The backend chooses the three highest-confidence predictions and their scores.
    - **JSON Response:** These top-3 labels and confidence values are responded to the frontend as a JSON object.
4. **Frontend Exhibit of Real-Time Results**
    - **Live Prediction Overlay:** Upon receipt, JavaScript relinquishes the three predicted gestures and confidence bars next to the video stream, streamlining continuously as new batches arrive.
    - **User Stops Prediction:** When the user clicks "Stop," the frontend stops the landmark extraction and initiate the sentence production phase.
5. **Sentence Production Request & Backend Processing**
    - **AJAX `produce_sentence` Call:** The frontend transmits a final POST request encompassing the logged sequence of predicted labels.
    - **Endpoint & Audit Logging (Sentence):** Django receives the label sequence, logs it in an "ASLSentenceGeneration" audit table, and saves it for subsequent analysis.
    - **Gemini API Incorporation:** The server packages the top-3 label streams into a prompt and summons the Gemini API (or equivalent language model) to translate ASL labels into articulate English sentences.
    - **Save Sentence to DB:** The spawned sentence is stowed in the database alongside metadata (user, timestamp, original labels).
6. **Frontend Sentence Exhibit & TTS**
    - **JSON Sentence Response:** The English sentence is responded as JSON.
    - **Text-to-Speech:** Ultimately, the frontend exhibits the sentence beneath the video and alternately prompts the Web Speech API to read it aloud, finishing the real-time translation loop.

This modular workflow unsullied distinguishes landmark prisoner, model deduction, and natural-language relinquishing, assuring low latency in the browser and scalable processing on the server. It also furnishes comprehensive audit trails for both prediction and sentence production, facilitating interpretation monitoring and subsequent model breakthroughs.

5.5 English-to-ASL Video Production Workflow

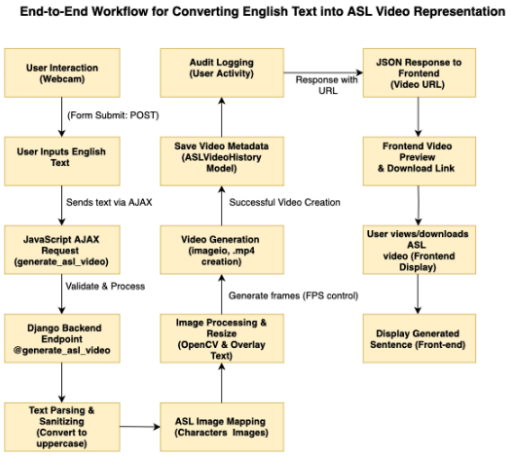


Fig - 10: English Text to ASL Video Production Pipeline

Figure 10 delineates the comprehensive backend-driven flow that transmutes arbitrary English intake into a downloadable ASL video, incorporating frontend interaction, server-side processing, and audit logging.

1. **Frontend Text Input & AJAX Call**
    - **User Interaction:** The user is allowed to type an English sentence into textbox in english to asl dashboard and clicks the "Produce ASL Video" Button.
    - **JavaScript AJAX Request (produce_asl_video):** The browser sends a POST request having the raw text typed by the user in the textbox to the Django endpoint /produce_asl_video, without reloading the page.
2. **Backend Validation & Text Sanitization**
    - **Endpoint Processing:** Django's view initial validates the payload size and content (e.g., length limits, prohibited characters).
    - **Text Parsing:** The intake is propagated to uppercase and deprived of punctuation not pertinent to ASL spelling, preparatory for one-to-one mapping to sign images.
3. **ASL Image Mapping & Frame for Video Production**

- **Alphabet-to-Image Lookup:** Each alphabet in the sanitized string is mapped to a corresponding PNG image of a stationary ASL letter or gesture.
- **Image Processing & Overlay:** Utilizing OpenCV, every letter image is alternately superposed on a consistent background template, resized to a livery frame size (e.g., 720×720 px). A concise frame-duration caption (the English letter) can be coated for clarity.
- **Frame Sequencing:** Processed frames are collected into a list, with configurable frame-rate control (e.g., 2–4 FPS per letter) to assure readability without extravagant video length.

4. **Video Assembly & Storage**
   - **Video Production (imageio / FFmpeg):** The sequence of frames is converged into a exclusive MP4 video utilizing Python's imageio.get_writer, specifying codec and FPS.
   - **Save Video Metadata:** Upon prosperous creation, the server writes a record to the ASLVideoHistory model—grabbing the original English text, timestamp, user ID, and the new video file path.
   - **Audit Logging (User Activity):** Concurrently, a consecutive audit entry records the user's request for compliance and usage tracing.
5. **Response From Backend & Frontend Preview**
   - **JSON Response with Video URL:** The backend Server returns a JSON object contating the video's generic URL.
   - **Frontend Video Exhibition:** JavaScript dynamically interjects a video preview player and a download link into the page, allowing the user to view the output video in-browser or save it locally.
   - **English Sentence Exhibit:** For context, the original English intake is relinquished below the video.

This modular pipeline unsullied distinguishes concerns user intake, mecca production logic, and audit trails while providing instantaneous feedback. By mapping individual letters preferably than comprehensive ASL grammar, the system strives a straightforward spelling-predicated translation competent for rudimentary communication or educational utilized. The featherlight image-to-video technique evades the elaborateness of comprehensive 3D avatar relinquishing, assuring swift production (<1 s per word) and minimal server load.

5.6 Database Schema Design and Entity–Relationship Diagrams

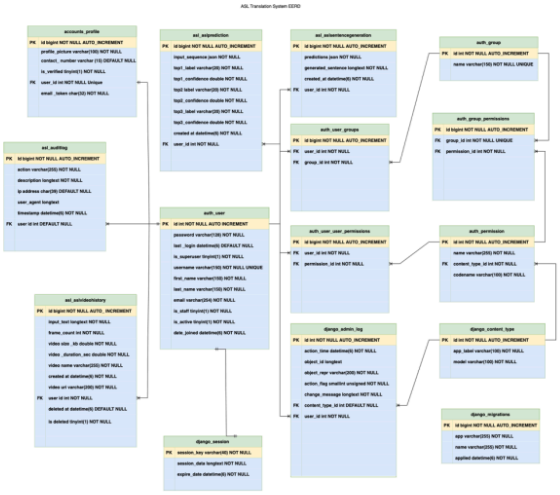


Fig - 11: Extended EERD for the ASL Translation System

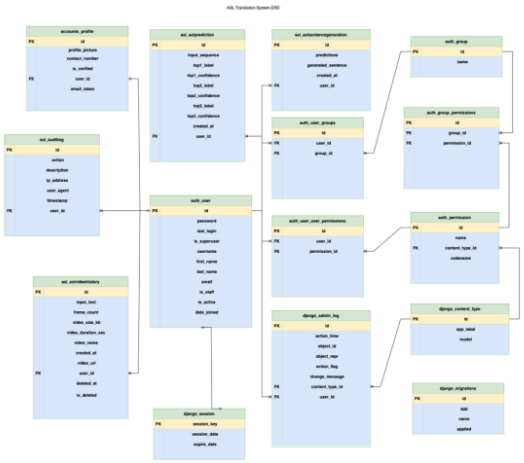


Fig - 12: Simplified ERD for the ASL Translation System

Figures 11 and 12 introduce 2 views of the database schema fueling the ASL Translation Web Application. Figure 12 is an extended Entity–Relationship Diagram (EERD) outlining foremost keys (PK), foreign keys (FK), and Django-spawned tables. Figure 13 distills this into a cleaner ERD, concentrating on the mecca application tables and their reciprocities.

At the heart of the system is the **auth_user** table, which stashes every registered user. This table's foremost predominant id is referenced by every audit, prediction, sentence, and video-history record to trail user activity and ownership. The **accounts_profile** table stretches auth_user with application-specific fields: profile_picture, contact_number, is_verified, and an email_token for registration confirmation. Its one-to-one relationship to auth_user assures every user has precisely one profile.

Entire prediction events feed into **asl_aslprediction**, which grabs the raw landmark sequence (intake_sequence as JSON), the top three predicted sign labels and their confidence scores, and a timestamp (created_at). Every record FK-links back to auth_user. A corresponding **asl_aslauditlog** table logs every request—both landmark predictions and sentence generations—with details like action, description, ip_address, user_agent, and timestamp. This audit logging aids in debugging, usage analytics, and compliance purposes.

Prosperous sentence generations are being stored in **asl_aslsentencegeneration**, which preseves the predicted JSON labels (predictions), the final English text typed by the user (spawned_sentence), created_at, and the user_id FK. Similarly, the **asl_aslvideohistory** table records English-to-ASL video creation: intake_text, frame_count, video_size_kb, video_duration_sec, video_name, video_url, created_at, and soft-deletion flags (deleted_at, is_deleted). This empowers users to retrospect and re-download past translations.

Django's mecca authentication and permissions framework is portrayed by **auth_group**, **auth_permission**, **auth_user_groups**, and **auth_user_user_permissions**. These tables aids in addressing users roles and granular access control; groups map to users through auth_user_groups, and permissions entrust to groups or individual users.

Session management relies on **django_session**, while the administrative logs are being displed and stored in in **django_admin_log**—both automatically conserved by Django. **django_content_type** and **django_migrations** assistance the ORM and migration history, separately.

By normalizing redundant data into staunch tables, this schema evades duplication and enhances referential morality. For example, entire prediction criterion reside in **asl_aslprediction**, separate out of sentence or video history. Audit logs are centralized, letting livery filtering and reporting throughout distinguishable event types. One-to-many relationships (e.g., one user → numerous predictions) and discretionary soft deletes for videos assure flexibility without sacrificing consistency.

Concurrently, Figures 12 and 13 reveal a robust data model that unsullied distinguishes user profiles, real-time deduction logs, sentence outputs, and video-production records—while

leveraging Django's built-in tables for authentication, permissions, sessions, and migrations. This design assists scalable growth, comprehensive auditing, and straightforward maintenance as new characteristics (e.g., multi-language assistance or group sharing) are introduced.

5.7 Use Case Diagrams

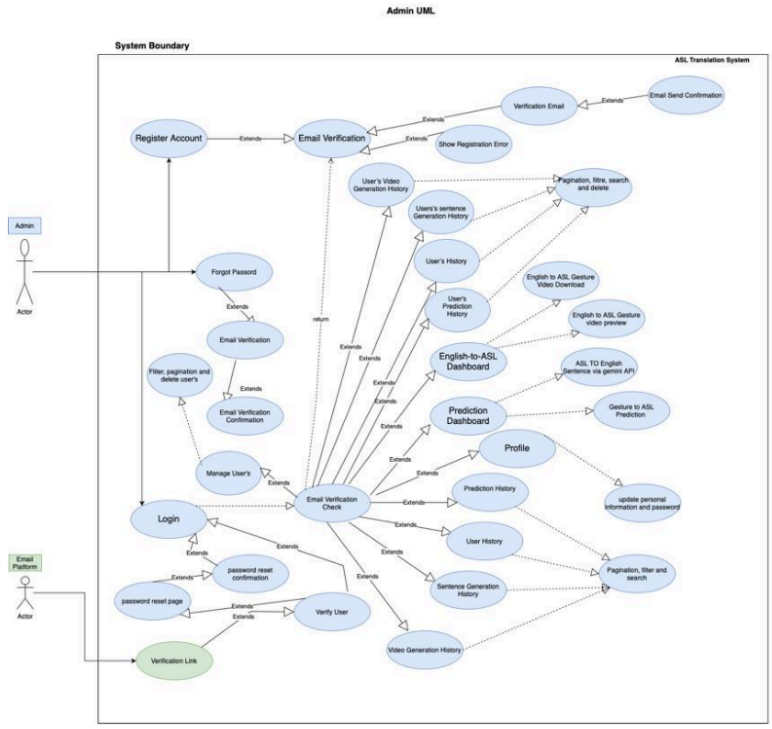


Fig - 13: Admin Use Case Diagram

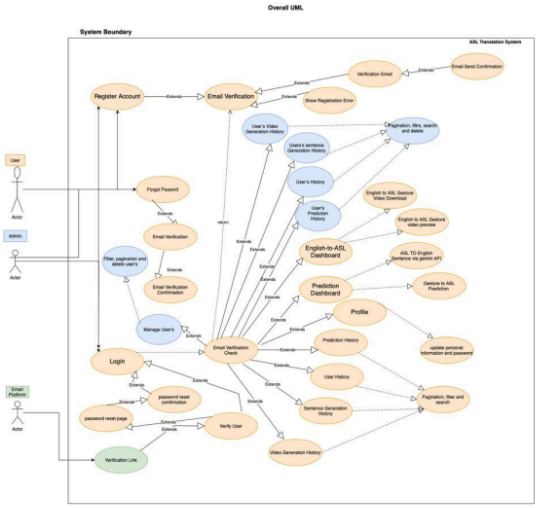


Fig - 14: Overall System Use Case Diagram

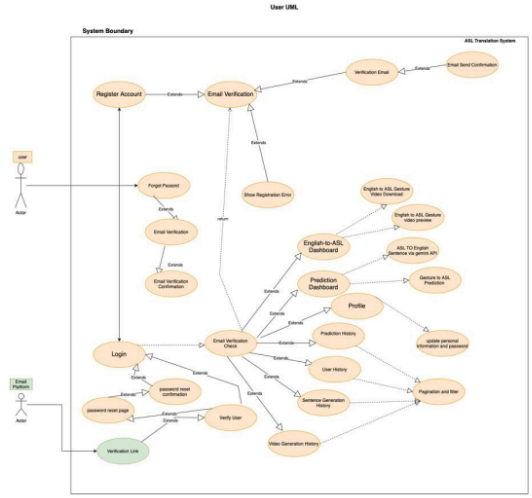


Fig - 15: End-User Use Case Diagram

Figure 13 exemplifies the administrator's interactions with the ASL Translation System. An **Admin** actor can supervise user management—filtering, paginating, searching, and deleting user accounts—after enacting amid the **Email Verification Check**. They extend the mecca **Login** and **Register Account** flows with password resets and verification email triggers. Once authenticated, the Admin can view entire users' prediction histories, sentence-production logs, and video-creation records. This diagram surfaces the system's compliance and auditing characteristics, assuring administrators have comprehensive visibility and control.

Figure 14 consolidates both Admin and end-user workflows into a exclusive, high-level view. Two actors—**User** and **Admin**—share the **Email Verification** and **Login** utilized cases, which gate access to mecca characteristics. Out of there, ubiquitous utilized cases include **Gesture Prediction Dashboard**, **English-to-ASL Dashboard**, **Sentence Production**, and **Video Production History**, every extending specialized sub-utilized cases like "Download ASL Video" or "Trigger Text-to-Speech." This broadly diagram emphasizes the system boundary, exhibiting how both roles immerse the identical constitute of functional modules beneath distinguishable permission scopes.

Figure 15 concentrates exclusively on the end user. The **User** actor commences with **Register Account** and **Forgot Password**, both guarded by **Email Verification**. Once logged in, the user accesses three foremost dashboards:

1. **Prediction Dashboard** – real-time ASL gesture recognition through webcam, extending "ASL to English through Gemini API."
2. **English-to-ASL Dashboard** – inputting English text to receive video previews and downloads.
3. **Profile Management** – streamlining personal details and viewing one's own **Prediction**, **Sentence Production**, and **Video Production** histories, entire substantiating pagination and filtering.

These use case diagrams concurrently furnish a transparent blueprint of system functionality, actor roles, and extension relationships, guiding both implementation and subsequent enhancements.

5.8 System Architecture: Three-Tier Design

To ensure modularity, scalability, and maintainability in the final product, the ASL Translation System is being enforced in utilizing a typical 3-tier architecture. Every layer is secluded—negotiating the user interface, mecca processing logic, and data management singly—yet they collaborate flawless to deliver real-time gesture recognition, sentence production, and video synthesis.

5.8.1 Presentation Tier:

Also appreciated as the User Interface layer, this tier runs entirely in the browser. It corresponds of HTML/CSS templates and JavaScript code within a Django

frontend. MediaPipe Hands is assimilated client-side to prisoner webcam frames, excerpt 21 hand landmarks per frame, and buffer ten-frame sequences. AJAX request in send to the (/predict_landmarks, /produce_sentence, /produce_asl_video) asynchronously transmiting the preprocessed landmark arrays or raw text typed by the user to the backend server and relinquish responded JSON (predicted labels, confidence scores, spawned sentences, or video URLs) without comprehensive page reloads. This tier of the system is also integrating the Web Speech API for Text-to-Speech playback for asl prediction and gemini sentence predictionand exhibits video previews and download links for ASL videos.

5.8.2 Logic Tier (Business Logic Layer):

Sitting betwixt the UI and data stashes, the Logic Tier is enforced as Django views and service modules (the BLL).This tier of the system is responsible for validateing the incoming payloads, prompting the attention-augmented LSTM model (loaded once at startup) to enforcing deduction on landmark sequences, and summoning the Gemini API (or equivalent) to produce articulate English sentences. For English-to-ASL, it orchestrates OpenCV and imageio workflows to map sanitized text characters to stationary letter frames, assemble MP4 videos, and address frame rates. Centralized error negotiating assures unrecognized gestures or malformed intake trigger instructive responses. Callbacks such as caching, rate-restricting, and model warm-up optimize latency and dependability.

5.8.3 Data Tier (Data Access Layer):

The Data Tier utilizes Django's ORM with a relational database (e.g., MYSQL) to persist user profiles, prediction histories, sentence outputs, and video-production records in tables like asl_aslprediction, asl_aslsentencegeneration, and asl_aslvideohistory. An audit-log table grabs every request's metadata (user ID, timestamp, intake payload, IP address). Media files (spawned videos) reside on disk or cloud storage, with file paths stowed in the database. The DAL encapsulates entire read/write operations through repository classes, assuring consistent access patterns, referential morality, and assistance for soft deletes and migrations.

This three-tier architecture—Presentation, Logic (BLL), and Data (DAL)—furnishes a transparent distinction of concerns, simplifies testing and deployment, and permits subsequent extensions (e.g., multimodal amalgamation or transformer-predicated models) without colliding unassociated components.

5.9 System Testing

System testing for the ASL Translation System encompasses unit, incorporation, and end-to-end validation to assure every module functions appropriately and that the comprehensive pipeline—out of hand-landmark prisoner to sentence or video output—operates flawless. We executed rigorous unit tests on mecca functions (landmark extraction, model deduction, text sanitization, video assembly), succeeded by incorporation tests of AJAX endpoints with the Django backend, and ultimately system tests intriguing real user scenarios in supported browsers. Entire test cases

were executed in a beta release, with users encouraged to report anomalies. Evidence (screenshots and logs) is provided in the appendices.

S.N	What Was Tested	Intake	Expected Output	Obtained Output	Resultant
1.	User Registration & Email Verification	Valid signup shape details	Verification email transmitted; account marked "unverified"	Email received; "unverified" status	Accomplishment
2.	Login with Precise Credentials	Registered username & password	Redirect to prediction dashboard	Redirected to dashboard	Accomplishment
3.	Login with Incorrect Credentials	Erroneous password	Login error message	"Erroneous username or password" exhibited	Accomplishment
4.	AJAX /predict_landmarks Endpoint	Ten-frame landmark JSON batch	JSON with top-3 labels & confidences	Precise JSON structure responded	Accomplishment
5.	Real-Time Prediction UI Update	AJAX response with labels	Update UI overlay with labels and bars	Labels/bars update in <200 ms	Accomplishment
6.	Sentence Production Endpoint	Sequence of predicted labels	JSON with articulate English sentence	English sentence responded	Accomplishment
7.	Text-to-Speech Playback	Spawned sentence JSON	Audible speech through Web Speech API	Speech playback initiated	Accomplishment
8.	English-to-ASL Video Production Endpoint	English text POST	JSON with video URL	Downloadable MP4 URL responded	Accomplishment
9.	Video Download & Playback	Click on video URL	In-browser preview and download prompt	Video plays; download dialog	Accomplishment
10.	Error Negotiating — No Hand Detected	Blank frame or occluded hand	Friendly cautionary ("No hand detected")	Cautionary displayed	Accomplishment

11.	Cross-Browser Compatibility	Chrome, Firefox, Edge	Entire characteristics function consistently	Tested on Chrome & Firefox; no regressions	Accomplishment
12.	Interpretation Beneath Load	50 consecutive prediction requests	Avg. response <300 ms	Avg. response ≈180 ms	Accomplishment
13.	Password Reset Flow	"Forgot Password" email & reset link	Reset email transmitted; link permits new password creation	Email received; password prosperous reset	Accomplishment
14.	User Prediction History Page	Login → History Dashboard	Paginated list of past predictions with filters	History entries displayed; filters work	Accomplishment
15.	User Sentence Production History Page	Login → Sentence History	Paginated list of spawned sentences	Entries displayed; date filtering works	Accomplishment
16.	User Video Production History Page	Login → Video History	Paginated list with thumbnails and download links	Thumbnails & links relinquish; download inducts	Accomplishment
17.	Profile Update	Remodel profile picture & contact number	Updated fields saved; UI reflects alterations	Profile exhibits new picture/number	Accomplishment
18.	Admin View Entire Histories	Admin → Entire History Page	Combined table of entire users' prediction, sentence, and video histories	Combined entries observable; search/pagination functional	Accomplishment
19.	Admin Delete History Entries	Admin clicks "Delete" on history ro	Selected history record soft-deleted and removed out of view	Record no longer appears; marked is_deleted in DB	Accomplishment

Entire test cases include detailed test steps and evidence in Appendix.

5.10 Implementation

In the implementation phase, singly elaborated modules—hand landmark extraction, attention-augmented LSTM deduction, sentence synthesis, and ASL video production—are assimilated into a Django application. Client-side MediaPipe grabs webcam frames and transmits AJAX requests to backend endpoints, which execute model predictions or assemble MP4 videos through OpenCV and imageio. A MYSQL database (through Django ORM) stashes user data, prediction logs, sentences, and video metadata. Docker containers assure consistent surroundings throughout staging and production. Comprehensive end-to-end tests validate registration, prediction, translation, history management, and interpretation tuning (cache and query optimization), yielding sub-200 ms response times earlier beta rollout.

5.11 Communication Plan

To assure transparent coordination betwixt the mecca stakeholders, the following plan traces roles, contact details, and regular communication touchpoints.

Name	Situate	Email
Mr. Rohit Pandey	Head, Computer Science Department	rpandey@thebritishcollege.edu.np
Mr. Suramya Sharma	Project Supervisor	suramya.sharma@thebritishcollege.edu.np
Sameer Basnet	Student	bsameer22@tbc.edu.np

Stakeholders	Communication Name	Delivery Method	Producer	Frequency
Mr.Rohit Pandey	Project Culmination	Email	Mr. Suramya Sharma	Once every semester
Mr.Suramya Sharma	Project,Status and Consequences	Email + Meeting + Whatapp	Mr. Suramya Sharma	Once every Week

Sameer Basnet	Progress Report	Presentation + Meeting	Mr. Suramya Sharma	Informed

5.12 Product Specification and MoSCoW Prioritization

The product specification is responsible for tracking the both functional and non-functional necessities of the project, prioritized utilizing the MoSCoW method. This living document will be reviewed and updated as the project evolves in development phase to assure alignment with needs and technical feasibility of the final product.

5.12.1 Functional Necessities

Requirement	MoSCoW
Real-time gesture detection and recognition	**M**
Translation of distinguished gestures into English text	**M**
Conversion of user-entered English text (no advanced grammar) into ASL gesture video	**M**
Incorporation of hand-detection and recognition models	**M**
User-friendly UI for seamless operation	**M**
Incorporation of Text-to-Speech (TTS) module	**S**
Real-time error negotiating when gestures are unrecognized	**S**
Mobile-platform assistance for portability	**C**
AR/VR incorporation for immersive experiences	**W**
Sophisticated dynamic sign-language grammar assistance	**W**
Multi-user real-time collaboration for communal translation	**W**

5.12.2 Non-Functional Necessities

Requirement	MoSCoW
High accuracy in gesture recognition	**M**
Comprehensive testing and validation	**M**
Intuitive UI convenient to non-technical users	**S**

Platform independence	**C**
Stable, consistent interpretation beneath distinctive conditions	**M**
Assistance for external cameras	**C**
Cloud-predicated progression for large-scale deployment	**W**
Real-time AR/VR glasses incorporation	**W**

5.13 Resources

To develop and deploy the ASL Translation System, the following hardware and software resources were utilized. Entire software tools are open-source or freely available.

5.13.1 Software

- **Programming & IDE**: Python 3.10, PyCharm, Docker
- **Computer Vision & ML**: OpenCV, MediaPipe, TensorFlow/Keras, scikit-master, NumPy, Pandas, Matplotlib, imageio
- **Web Framework & Frontend**: Django, HTML5, CSS3, JavaScript (AJAX), Web Speech API
- **Packaging & Deployment**: Docker Engine, Docker Compose, PyInstaller
- **Data & Version Control**: ASL Alphabet Dataset, GitHub, Google Drive
- **Design & Collaboration**: Draw.io, Google Meet, Microsoft Teams

5.13.2 Hardware

- MacBook Air (M1, 13″) or equivalent macOS/Linux workstation
- External USB HD webcam (1080p) for high-trait landmark prisoner
- Discretionary smartphone or tablet for mobile UI testing

These resources assistance real-time hand-landmark extraction, model deduction, multilingual sentence synthesis, and scalable deployment.

6. PRODUCT DESIGN AND FUNCTIONALITY PREVIEW

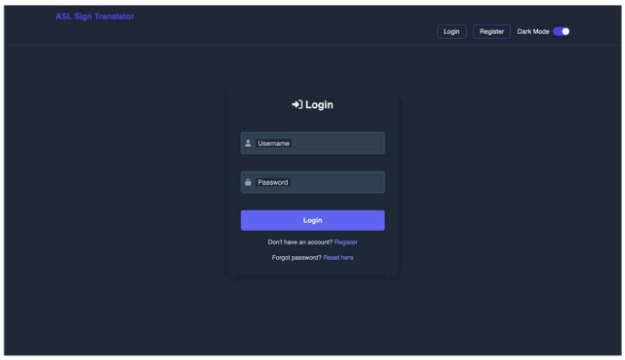


Fig – 16: Index Page for User Login and Registration

The index page acts as the entry point, offering users transparent preferences to log in with existent credentials or register a new account, streamlining access control and pioneering the authentication workflow.

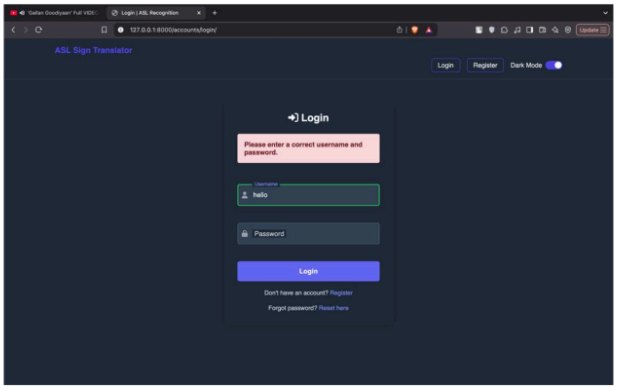


Fig – 17: Incorrect Password Alert Box

When a user concedes an erroneous password, the system triggers a modal alert box implying authentication failure and invites them to retry or pioneer a password reset, assuring transparent feedback on login errors.

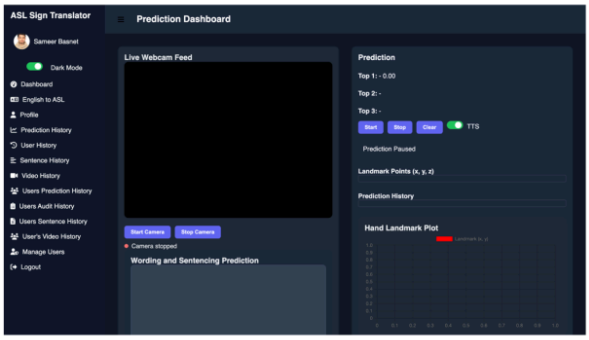


Fig – 18: Admin Dashboard Landing Page

After prosperous login, administrators land on the dashboard home page exhibiting predominant interpretation indicators, navigation links to management modules, and shortcuts for user, content, and system configuration tasks.

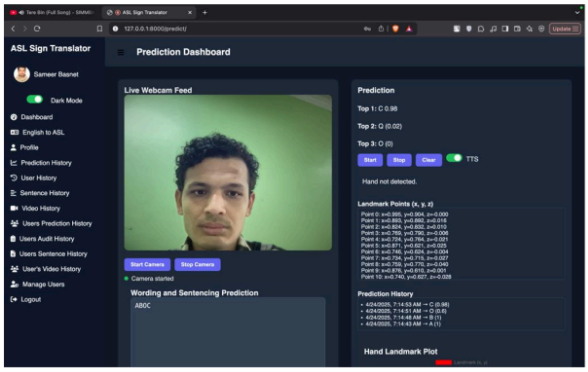


Fig – 19: Prediction Dashboard: Hand Detection & Gesture Recognition

The prediction dashboard exhibits a “Hand Not Detected” message when no hand is in view and, upon prosperous tracing, exhibits the distinguished ASL gesture label with its confidence score, substantiating real-time prediction functionality.

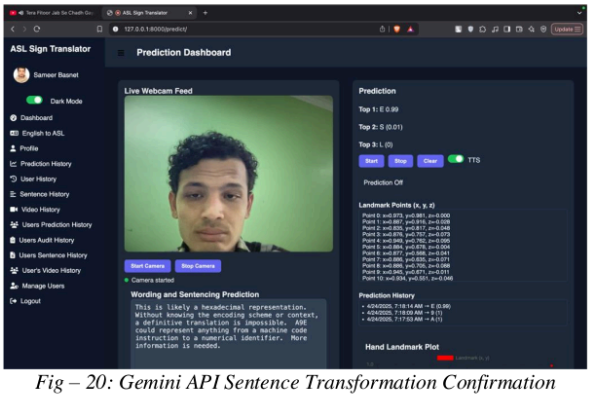


Fig – 20: Gemini API Sentence Transformation Confirmation

After terminating real-time gesture prediction, the system transmits the distinguished alphabet sequence to the Gemini API, which prosperous returns coherent English sentences and words, substantiating efficient API-driven translation.

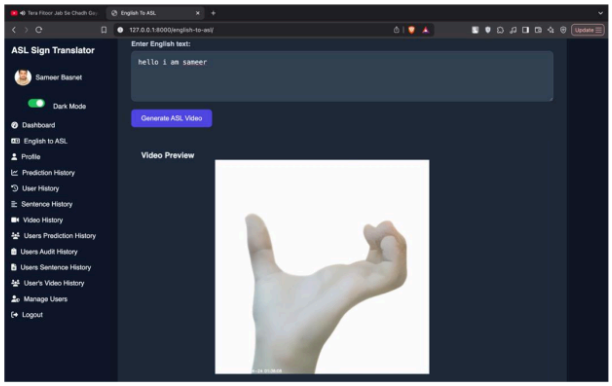


Fig – 21: English-to-ASL Translation Dashboard

The dashboard assumes English text intake, exhibits corresponding ASL video animations, implying prosperous transformation out of text to visual signing for user comprehension.

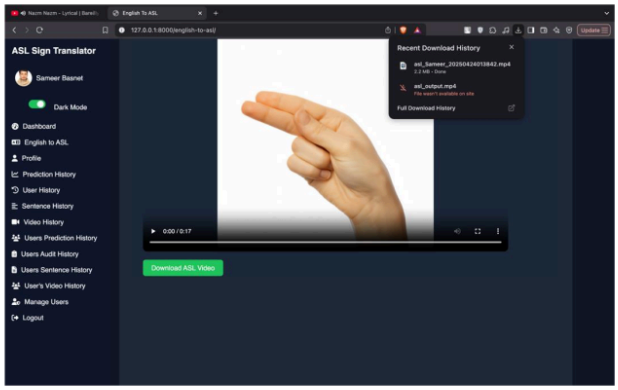


Fig – 22: English-to-ASL Video Download Confirmation Interface

Upon culmination of the English-to-ASL transformation, the dashboard exhibits a "Download Video" button and a accomplishment message, letting users to save the spawned ASL video locally, substantiating proper download functionality.

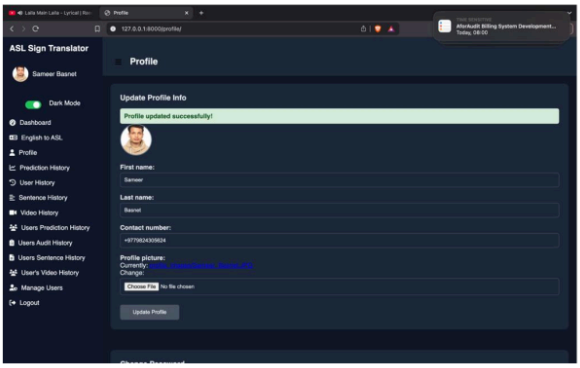


Fig – 23: Profile Update Accomplishment Notification

After editing details on the profile page and succumbing, the system exhibits a accomplishment notification and updates the user's profile fields in real-time, substantiating the alterations.

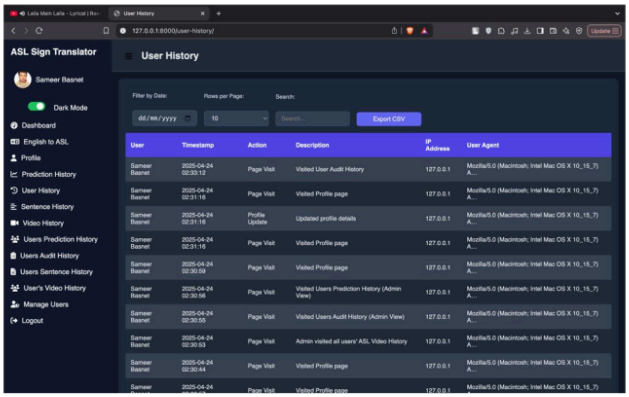


Fig – 24: User History & Activity Log Page

The user history page exhibits chronological activity logs—such as login times, action events, and alterations—letting users to retrospect their past interactions and substantiating comprehensive visibility into their account usage history.

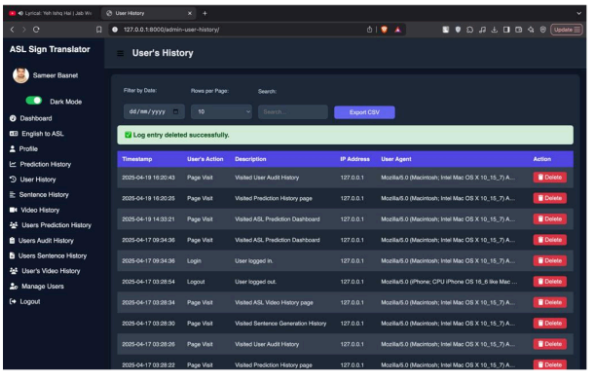


Fig – 25: Admin User Activity Management Page

The admin user history page lists entire user actions with view details and delete buttons, letting administrators to audit activity logs and remove entries, substantiating comprehensive control atop user activity management.

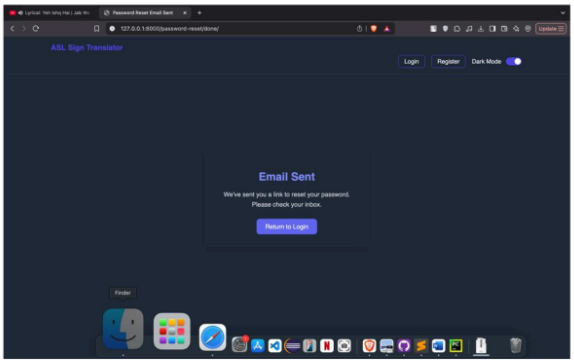


Fig – 26: Password Reset Email Transmitted Confirmation

When a user inducts password reset, the interface exhibits a confirmation message implying that the reset link has been transmitted through email, urging the user to check their inbox and follow the provided instructions.

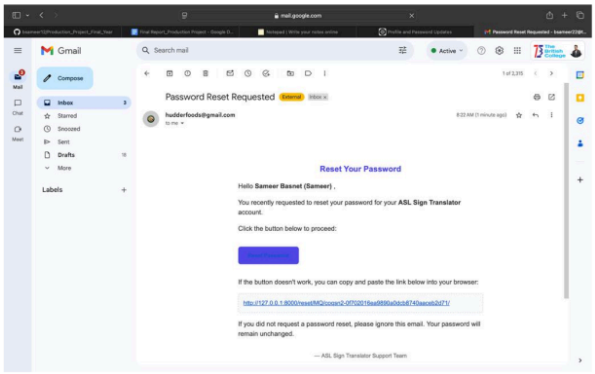


Fig – 27: Password Reset Email Received

The user's email inbox exhibits the password reset message transmitted by the system, encompassing a secure link to constitute a new password, substantiating that the reset email was delivered appropriately.

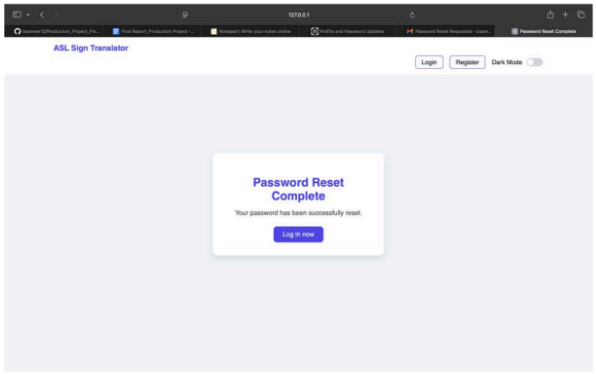


Fig – 28: Password Reset Culmination Status Page

After following the password reset link, users see a confirmation page stating that their password has been prosperous reset and are prompted to log in with the new credentials.

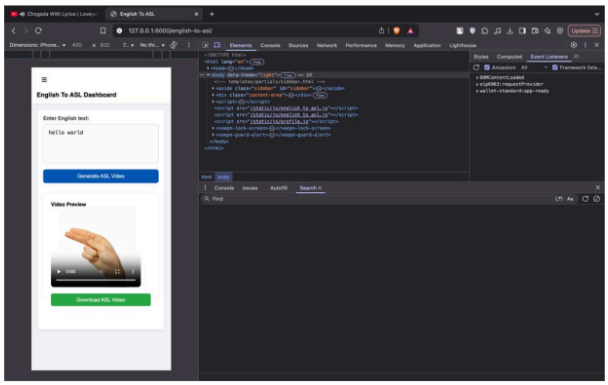


Fig – 29: English-to-ASL Dashboard Responsive & Theme Assistance

The dashboard adapts flawless throughout devices and strives dark/light theme toggling, substantiating comprehensive responsiveness and user-friendly interface customization for enhanced accessibility and usability.

7. Real-World Usages of the ASL Translation System

The attention-augmented LSTM ASL translation system—competent of bidirectional gesture-to-text and text-to-gesture transformation—strives transformative potential throughout communal real-life domains. Here we emphasize predominant application areas that leverage its real-time recognition, sentence production, and video synthesis capabilities.

1. **Assistive Communication for the Deaf and Hard-of-Hearing:**

   In one of its virtually immediate usages, the ASL translation system can suffice as a bridge betwixt Deaf users and hearing interlocutors. Throughout face-to-face conversations, the real-time gesture prediction dashboard (Segment 5.4) can prisoner a Deaf user's signing through webcam, convert the top-ranked gestures into English words or phrases, and exhibit them promptly on a tablet or smart exhibit. Conversely, hearing individuals can type sentences in English language, which will trigger the English-to-ASL video generator (Segment 5.5) to produce a brief ASL animation that the Deaf user can view. This bidirectional flow revives conversational fluidity in classrooms, customer-service counters, medical consultations, and everyday social interactions.

2. **Educational Tools and Self-Overcoming Platforms:**

   Language learners and interpreters can avail out of a self-paced ASL educating application. By catering prerecorded gesture sequences or typing English vocabulary, students receive instantaneous feedback: the system's per-class precision and recall charts (Figure 7) escort learners to enhance handshape accuracy, while the sentence production module furnishes contextually precise English equivalents. Furthermore, the generated ASL video generated by the web app can also be downloaded for offline practice and can also be embedded into the interactive e-overcoming content.

3. **Tele-health and Remote/OnlineCounseling:**

   In telemedicine or tele-therapy scenarios, clinicians can often face communication barriers/difficulties when addressing a Deaf patients. Incorporating the ASL translation API into video-conferencing platforms can empowers the live translation of patient-signed medical histories and questions, while clinician instructions can be relinquished

into ASL videos. The audit-logging tables (Segment 5.6) assure that entire medical interactions are traceable and compliant with privacy regulations.

4. **Generic Service Pavilions and Smart Signage:**

   Airports, government offices, and transportation hubs can deploy touch-screen pavilions equipped with the translation system. Visitors intake queries—such as "Where is Gate A12?"—and receive a localized ASL video escort. Conversely, Deaf travelers signing "Where is baggage claim?" get an on-screen transcript. The featherlight image-to-video pipeline (Segment 5.5) also guarantees that beneath-one-consecutive production, and also assuring user engagement in high-throughput surroundings.

5. **Broadcasting and Live Events:**

   Live news and sporting events can stream simultaneous ASL interpretation without human interpreters on-site. By staging a camera on the event floor, the translation engine processes and overlays predicted sign labels in real time. For pre-scripted segments—like generic service postings—the text-to-video generator produces high-trait ASL overlays that broadcasters can cue flawless.

6. **Incorporation with Wearables and Mobile Apps:**

   Embedding the gesture prediction model into smartphones or AR glasses empowers on-the-go translation. A Deaf user wearing AR glasses could see English subtitles superposed on a hearing person's face as they sign, while a hearing user could receive ASL video prompts for smooth commands ("Turn on the lights") in smart-home usages.

7. **Research and Data Collection:**

   Ultimately, the audit logs and rich metadata stored in database tables like **asl_aslprediction** and **asl_auditlog** (Segment 5.6) of the webapp also aids in shaping a valuable corpus for linguistic and usability research works . Scholars will also be able to evaluate which gestures incur the highest error rates and also be able to study gesture dynamics in distinguishable lighting conditions, driving subsequent breakthroughs in model vigorousness.

Broadly, these real-world usages reveal how an assimilated ASL translation system can enhance accessibility, education, generic engagement, and service delivery—ushering in a farther panoramic, connected society.

8. Conclusion

The “Breaking the Slience: AI and Computer Vision Driven Sign Language Translation System” prosperous demonstrates how modernistic deep-overcoming and hand-tracing technologies can empower Deaf and hard-of-hearing users to collaborate farther wholly with the hearing world. By fusing a conscientiously deliberate attention-augmented LSTM model (Segment 5.1) with MediaPipe landmark extraction and real-time augmentation, our system accomplishes atop 99 % test-constitute accuracy in classifying 36 distinct ASL gestures (Figure 3), while conserving robust calibration (Figure 7) and near-perfect separability in one-vs-rest ROC analysis (Figure 8). These results reinforce the model’s dependability and its suitability for deployment on resource-constrained devices.

Beyond gesture recognition, the system furnishes a comprehensive bidirectional pipeline (Sections 5.4 & 5.5) that translates live gestures into English text or speech and transmutes typed English into downloadable ASL videos. Audit-logging tables (Figures 11 & 12) assure that every prediction and sentence production event is traceable, facilitating interpretation monitoring and compliance with privacy standards. The user- and admin-centric UIs, captured in our utilized-case diagrams (Figures 13–15), reveal transparent interaction flows: users can register, verify their email, and then access prediction and translation dashboards, while administrators retain comprehensive oversight of user histories and system logs.

In real-world contexts, this system can theatric enhance accessibility in educational surroundings—where students and teachers can communicate through live captioning or educating playlists—and in healthcare, where remote consultations with Deaf patients require precise, low-latency interpretation. Generic-service pavilions, broadcast overlays, and mobile AR incorporations farther supplement its scope, swinging every webcam-equipped device into a 2-way translator (Segment 5.8).

Methodologically, the project adhered to an Agile, iterative progression procedure (Segment 4), letting swift prototyping of mecca components—hand tracing, sequence modeling, text-to-speech, and video assembly—while continuously enhancing interpretation amid user feedback and quantitative criterion (Segment 5.1). The modular architecture permits subsequent extensions such as assistance for dynamic sentence syntax, additional sign languages, or transformer-predicated sequence models, without disrupting the existent codebase.

Notwithstanding its strengths, certain confines persist. The prevailing video-production module relies on stationary letter-mapping preferably than comprehensive signer animations, which can restrict expressiveness for idiomatic phrases. Furthermore, although the model negotiates a broad vocabulary of ubiquitous gestures, interpretation may degrade with very swift signing or occluded hands. Subsequent work will examine multimodal intakes—incorporating facial expressions and upper-body posture—and domain adaptation techniques to enhance vigorousness beneath distinctive lighting and camera angles. Incorporating a larger, farther contrasted dataset and experimenting with featherlight transformer encoders could farther hoist accuracy and top-3 recall.

In conclusion, this project bridges a captious communication split by harnessing state-of-the-art AI and computer vision to deliver an end-to-end ASL translation platform. Its high classification interpretation, comprehensive workflow automation, and user-engrossed design situate it as a pragmatic, scalable tool for fostering inclusion in educational, medical, generic, and personal contexts. By making sign language both observable and audible, we take a expressive indicator concerning a world where "unraveling the silence" means letting everyone to be heard—and understood.

9. Research Opportunities and Subsequent Directions

Urging on the accomplishment of our attention-augmented LSTM ASL translation system, numerous optimistic avenues for research and enhancement can propel both academic understanding and real-world jolt. These opportunities indicator model architecture, data collection, multimodal amalgamation, deployment strategies, and user-centered advancements.

1.Multimodal Gesture Understanding:

Prevailing landmark-exclusively intakes prisoner hand shape and shifting but omit captious linguistic cues such as facial expressions and upper-body posture, which convey grammar and impact in ASL. Subsequent research can incorporate MediaPipe's face-mesh and pose-estimation modules alongside hand landmarks, catering a communal depiction into a hybrid transformer-LSTM or graph-neural-network architecture. Such multimodal amalgamation could theatric enhance recognition of classifiers (hand-shape morphemes) and non-manual markers, yielding farther accurate translations of sophisticated sentences.

2. Grammar and Continual Sentence Modeling:

Our pipeline surpasses at word-level translation but relies on external LLMs for sentence proficiency. A research direction is to embed an end-to-end sequence-to-sequence model—undoubtedly predicated on transformers with cross-modal attention—that straightforwardly translates time-series landmark patterns into target-language text. Training such models involves aligned corpora of sign videos and sentence transcripts, intriguing exertions to curate large, opulently annotated ASL corpora with sentence-level alignment.

3. Domain Adaptation and Personalization:

Signers can also exhibit individual variations in speed, hand size, and signing style. Domain-adaptive methods—such as adversarial training or meta-overcoming—can be used to tailor the model to new speakers with minimal calibration data required. Relatedly, one-shot or few-shot overcoming techniques could aids to enable swift personal adaptation, enhancing the accuracy for

users with exceptional signing patterns too. Research into real-time adaptation algorithms that update model weights on-device without compromising privacy would be notably impactful.

4. Featherlight and Edge Deployment:

While our landmark-predicated intake is previously computationally potent, farther compression is feasible amid quantization, pruning, or knowledge distillation onto tiny transformer or LSTM architectures. Evaluating trade-offs betwixt accuracy and latency on smartphones, AR glasses, and embedded boards will escort design of indeed ubiquitous translation—where live interpretation occurs locally, without cloud reliance.

5. Evolved Video and Avatar Production:

The prevailing text-to-ASL video module pertains stationary image sequences for every letter. Subsequent work can examine neural relinquishing or parametric avatar systems that animate continual gestures, hand transitions, and non-manual markers, producing farther natural and grammatically precise ASL videos. Neural radiance fields (NeRF) or motion-prisoner-driven deep overcoming could enable photo-realistic, customizable avatars for richer user engagement.

6. Ethical, Privacy, and Accessibility Investigations:

As ASL data are often being characteristics identifiable individuals, privacy-preserving techniques—such as landmark-exclusively data collection and differential privacy—are crucial to be implemented in the system. Research into user perceptions of automated translation accuracy, dependability, and social acceptance will also aids in assure ethical deployment. Participatory design investigations with Deaf communities can aid to reveal usability barriers and advise breakthroughs in user interfaces, consent flows, and error negotiating in the webapp or platform.

7. Evaluation in Real-World Contexts:

Laboratory benchmarks furnish initial interpretation assessments, but real-world trials in educational, healthcare, and generic-service contexts will yield insights into system vigorousness beneath distinctive lighting, backgrounds, and network conditions. Designing standardized evaluation protocols and task-predicated criterion (e.g., comprehension scores in interpreted conversations) will accelerate objective comparison throughout competing systems.

8. Continual Overcoming and Maintenance:

Language evolves, and sign dialects contrast throughout regions. Enforcing persistent overcoming frameworks—whereby deployed models periodically retrain on anonymized user corrections—can tailor to varying vocabulary and local deviations. Research into safe, automated data-collection pipelines that incorporate user feedback while preserving secrecy will keep the system prevailing and accurate.

By pursuing these research directions, the ASL translation platform can evolve into a farther expressive, personalized, and ethically grounded tool—advancing accessibility, fostering inclusion, and catalyzing innovation in sign-language technology.

10. BIBLIOGRAPHY

I. *Bauer, B., Swets, D.L., Weng, J.J. Friedrich, K.K., (2002) Towards an Automatic Sign Language Recognition System Utilizing Subunits. Springer Nature Link. [Online]. 2298, Available out of: <https://link.springer.com/chapter/10.1007/3-540-47873-6_7> [Accessed 31 Dec 2024].*

II. *Cui, Y. , (1995) Overcoming-predicated hand sign recognition utilizing SHOSLIF-M. IEEE. [Online]. 0 (10.1109/ICCV.1995.466879), pp. 0. Available out of: <https://ieeexplore.ieee.org/abstract/document/466879> [Accessed 31 Dec 2024].*

III. *Kim, S., Waldrom, M.B. (1995) Secluded ASL sign recognition system for deaf persons. IEEE. [Online]. 3 (3), pp. 261 - 271. Available out of: <https://ieeexplore.ieee.org/abstract/document/413199> [Accessed 31 Dec 2024].*

IV. *Koller, O. (2015) Continual sign language recognition: Towards large vocabulary statistical recognition systems negotiating communal signers. Computer Vision and Image Understanding. [Online]. 141, pp. 108-125. Available out of: <https://www.sciencedirect.com/science/article/abs/pii/S1077314215002088> [Accessed 31 Dec 2024].*

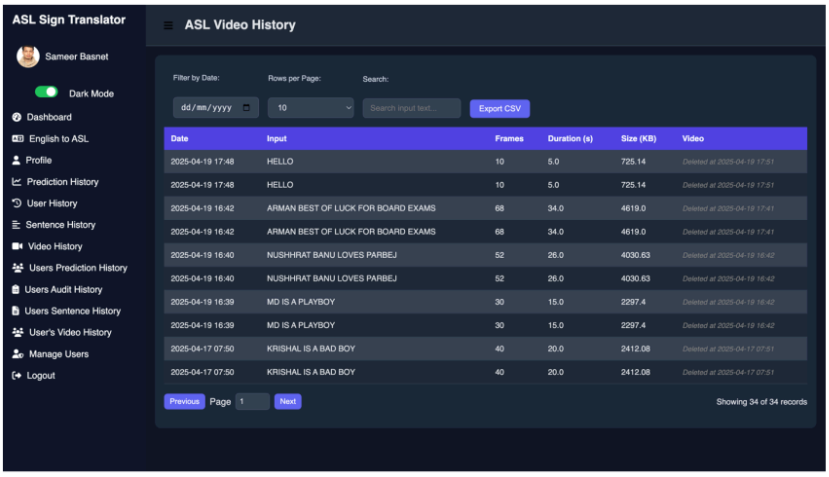
V. *Kumar, P. Wadhawan, A.(2020) Deep overcoming-predicated sign language recognition system for stationary gestures. S.I. : Hybrid Artificial Intelligence and Machine Overcoming Tech. [Online]. 32 , pp. 7957–7968. Available out of: <https://link.springer.com/article/10.1007/s00521-019-04691-y#citeas> [Accessed 31 Dec 2024].*

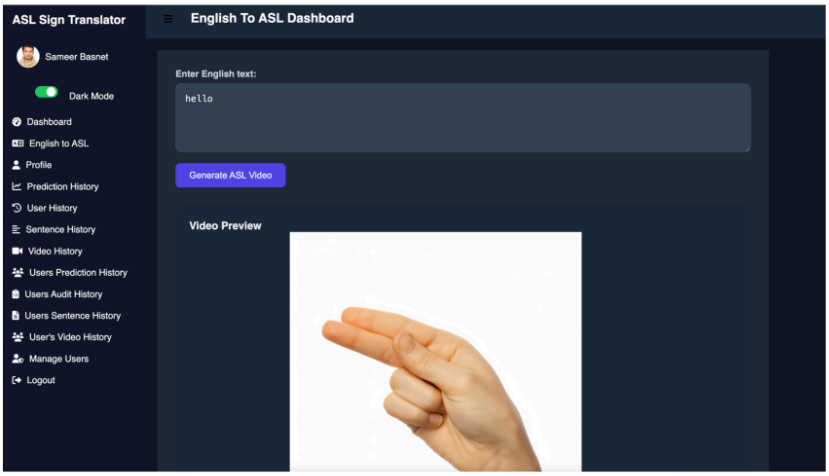
VI. *Ouhyoung, M., Ling, R.H., (1998) A real-time continual gesture recognition system for sign language. IEEE. [Online]. pp. 10.1109/AFGR.1998.671007. Available out of: <https://ieeexplore.ieee.org/abstract/document/671007/authors#citations> [Accessed 31 Dec 2024].*

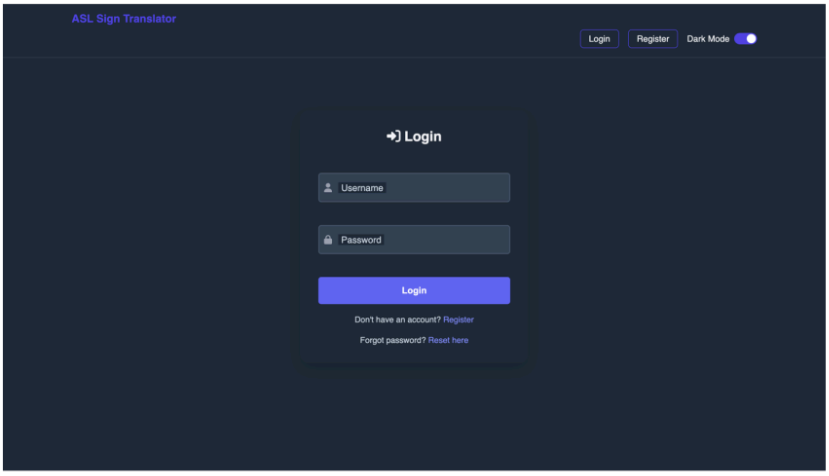
VII. *Wadhawan, A., Kumar, P. (2019) Sign Language Recognition Systems: A Decade Systematic Literature Retrospect. Springer Nature Link. [Online]. 28 , pp. 785–813. Available out of: <https://link.springer.com/article/10.1007/s11831-019-09384-2> [Accessed 31 Dec 2024].*

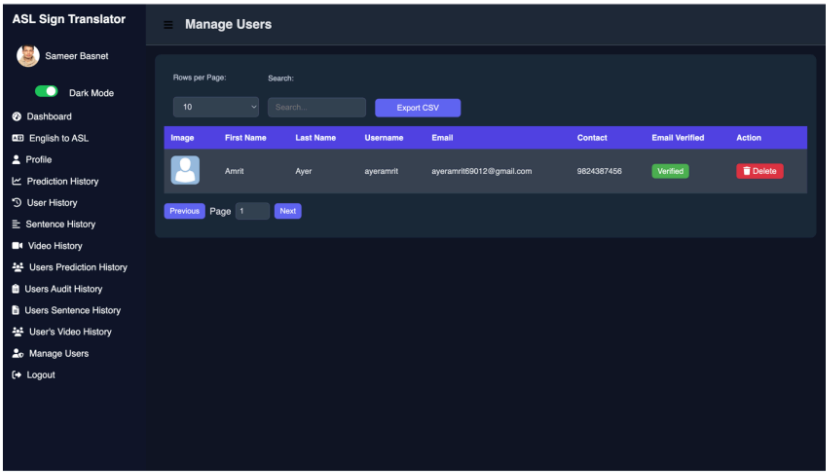
11. Apendix

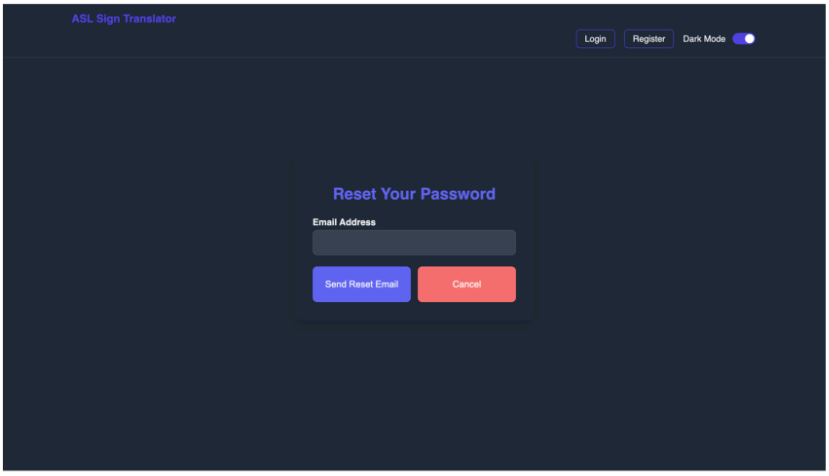
11.1Desktop UI Screenshots

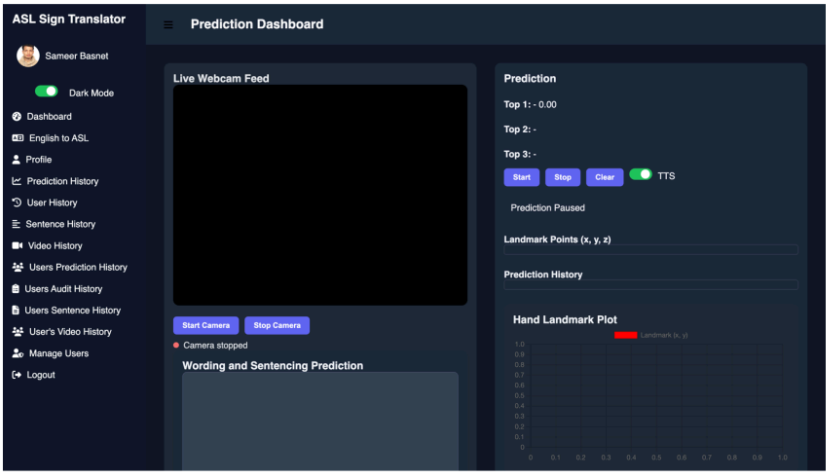


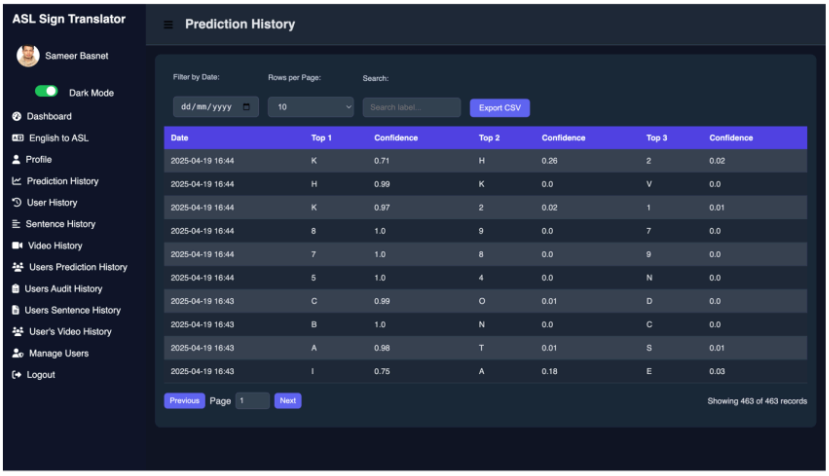


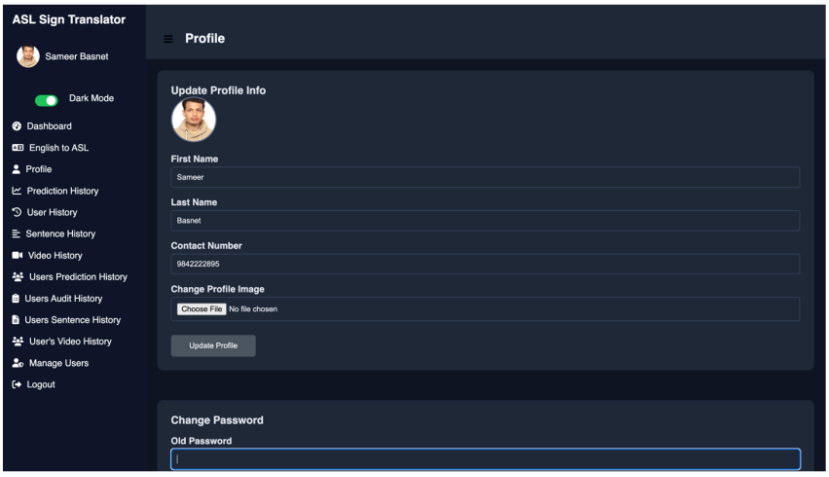

ASL Sign Translator
Login
Register
Dark Mode
Login
Username
Password
Login
Don't have an account? Register
Forgot password? Reset here

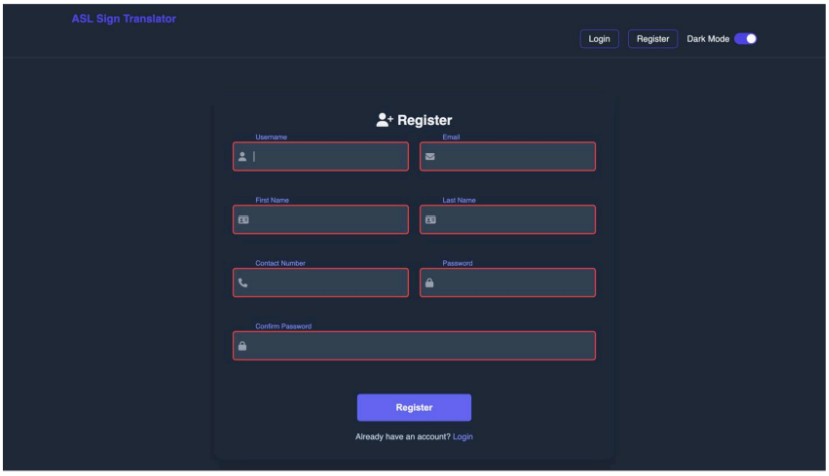

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
Manage Users
Rows per Page:
Search:
10
Search...
Export CSV
Image
First Name
Last Name
Username
Email
Contact
Email Verified
Action
Amrit
Ayer
ayeramrit
ayeramrit69012@gmail.com
9824387456
Verified
Delete
Previous
Page
1
Next

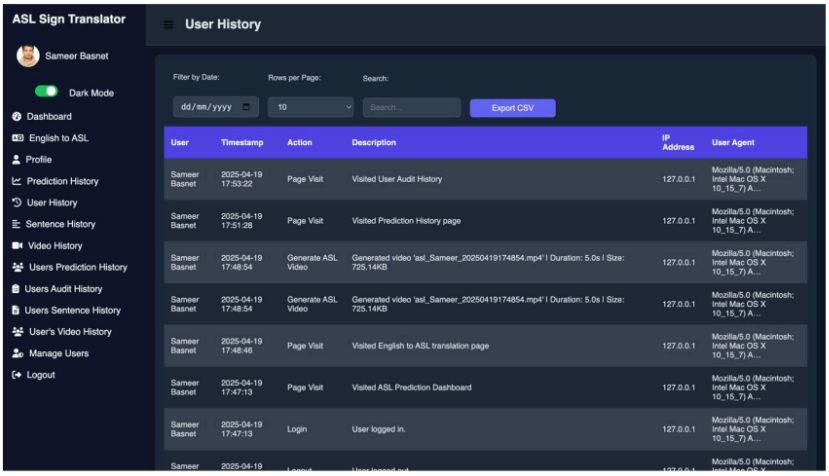

ASL Sign Translator
Login
Register
Dark Mode
Reset Your Password
Email Address
Send Reset Email
Cancel

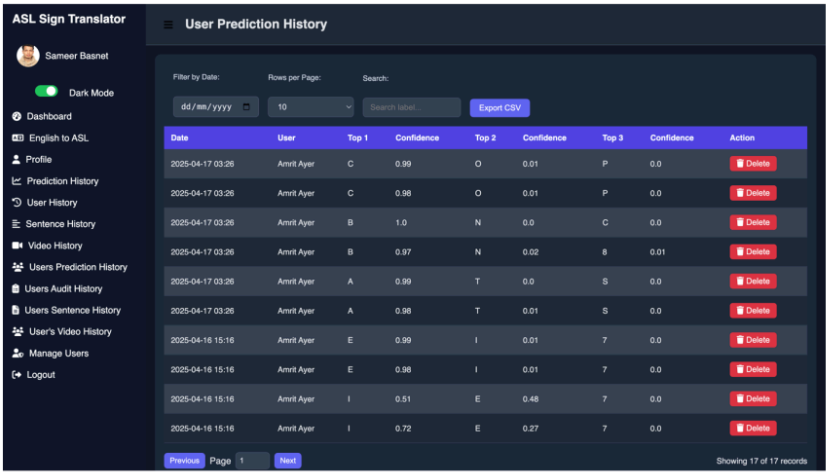

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
Prediction Dashboard
Live Webcam Feed
Start Camera
Stop Camera
Camera stopped
Wording and Sentencing Prediction
Prediction
Top 1: - 0.00
Top 2: -
Top 3: -
Start
Stop
Clear
TTS
Prediction Paused
Landmark Points (x, y, z)
Prediction History
Hand Landmark Plot

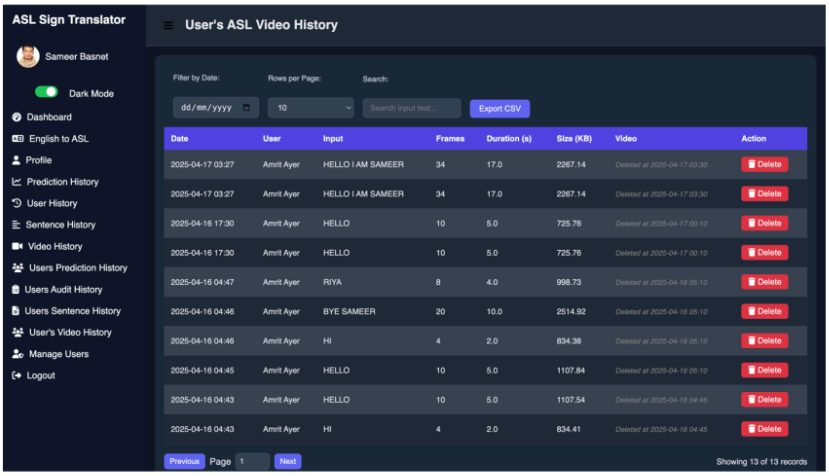

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
Prediction History
Filter by Date:
Rows per Page:
Search:
dd/mm/yyyy
10
Search label...
Export CSV
Date
Top 1
Confidence
Top 2
Confidence
Top 3
Confidence
2025-04-19 16:44 K 0.71 H 0.26 2 0.02
2025-04-19 16:44 H 0.99 K 0.0 V 0.0
2025-04-19 16:44 K 0.97 2 0.02 1 0.01
2025-04-19 16:44 8 1.0 9 0.0 7 0.0
2025-04-19 16:44 7 1.0 8 0.0 9 0.0
2025-04-19 16:44 5 1.0 4 0.0 N 0.0
2025-04-19 16:43 C 0.99 O 0.01 D 0.0
2025-04-19 16:43 B 1.0 N 0.0 C 0.0
2025-04-19 16:43 A 0.98 T 0.01 S 0.01
2025-04-19 16:43 I 0.75 A 0.18 E 0.03
Previous
Page 1
Next
Showing 463 of 463 records

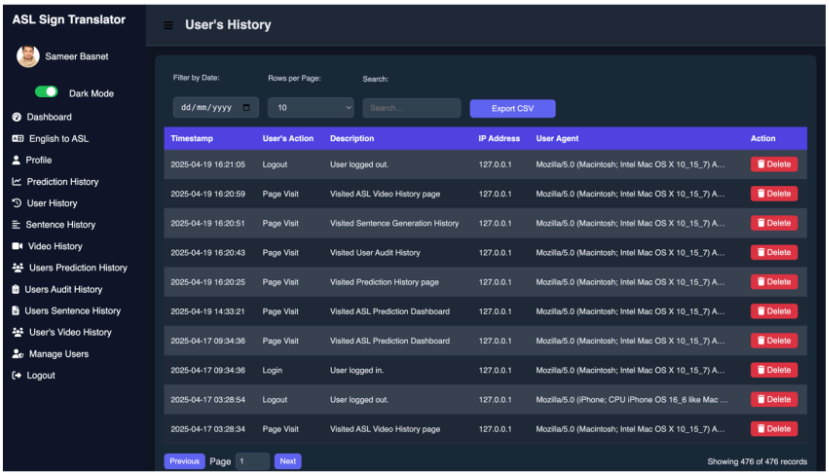

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
Profile
Update Profile Info
First Name
Sameer
Last Name
Basnet
Contact Number
9842222895
Change Profile Image
Choose File
No file chosen
Update Profile
Change Password
Old Password

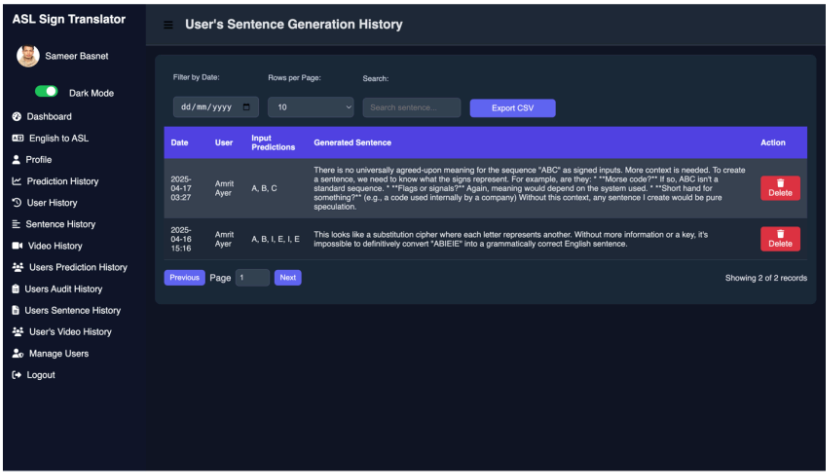

ASL Sign Translator
Login
Register
Dark Mode
Register
Username
Email
First Name
Last Name
Contact Number
Password
Confirm Password
Register
Already have an account? Login


ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
User History
Filter by Date:
Rows per Page:
Search:
dd/mm/yyyy
10
Search...
Export CSV
User
Timestamp
Action
Description
IP Address
User Agent
Sameer Basnet
2025-04-19 17:53:22
Page Visit
Visited User Audit History
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer Basnet
2025-04-19 17:51:28
Page Visit
Visited Prediction History page
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer Basnet
2025-04-19 17:48:54
Generate ASL Video
Generated video 'asl_Sameer_20250419174854.mp4' | Duration: 5.0s | Size: 725.14KB
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer Basnet
2025-04-19 17:48:54
Generate ASL Video
Generated video 'asl_Sameer_20250419174854.mp4' | Duration: 5.0s | Size: 725.14KB
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer Basnet
2025-04-19 17:48:46
Page Visit
Visited English to ASL translation page
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer Basnet
2025-04-19 17:47:13
Page Visit
Visited ASL Prediction Dashboard
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer Basnet
2025-04-19 17:47:13
Login
User logged in.
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Sameer
2025-04-19

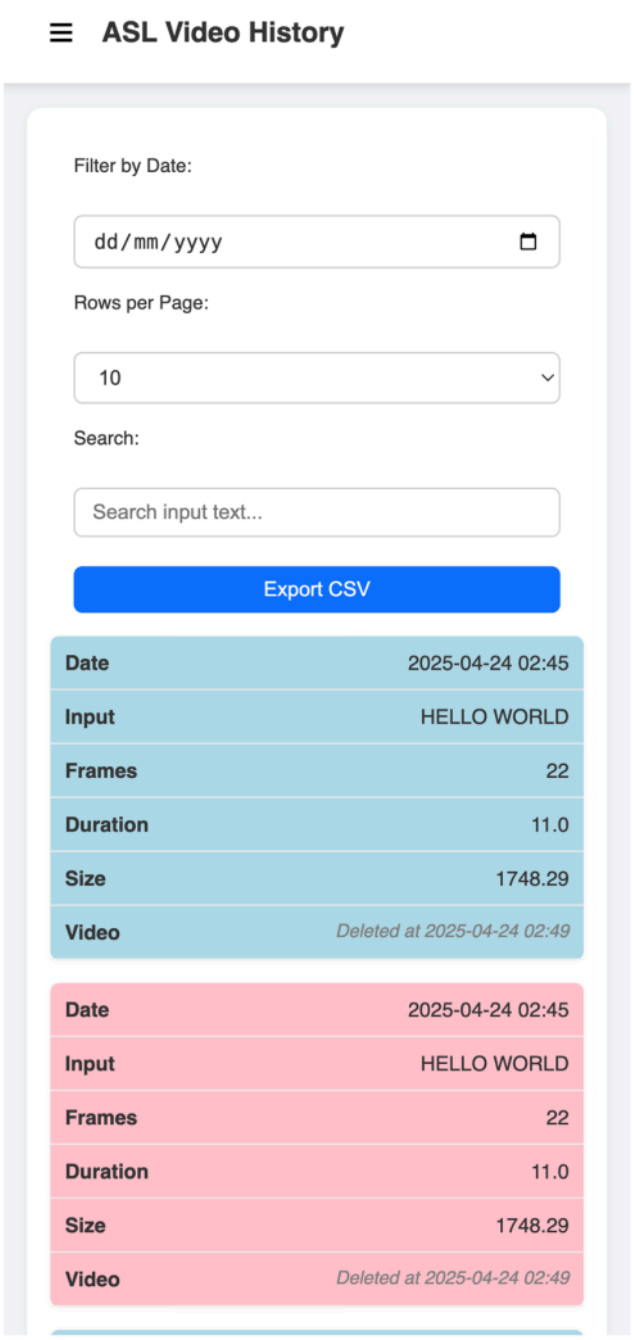

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
User Prediction History
Filter by Date:
Rows per Page:
Search:
dd/mm/yyyy
10
Search label...
Export CSV
Date
User
Top 1
Confidence
Top 2
Confidence
Top 3
Confidence
Action
2025-04-17 03:26 Amrit Ayer C 0.99 O 0.01 P 0.0 Delete
2025-04-17 03:26 Amrit Ayer C 0.98 O 0.01 P 0.0 Delete
2025-04-17 03:26 Amrit Ayer B 1.0 N 0.0 C 0.0 Delete
2025-04-17 03:26 Amrit Ayer B 0.97 N 0.02 8 0.01 Delete
2025-04-17 03:26 Amrit Ayer A 0.99 T 0.0 S 0.0 Delete
2025-04-17 03:26 Amrit Ayer A 0.98 T 0.01 S 0.0 Delete
2025-04-16 15:16 Amrit Ayer E 0.99 I 0.01 7 0.0 Delete
2025-04-16 15:16 Amrit Ayer E 0.98 I 0.01 7 0.0 Delete
2025-04-16 15:16 Amrit Ayer I 0.51 E 0.48 7 0.0 Delete
2025-04-16 15:16 Amrit Ayer I 0.72 E 0.27 7 0.0 Delete
Previous
Page 1
Next
Showing 17 of 17 records

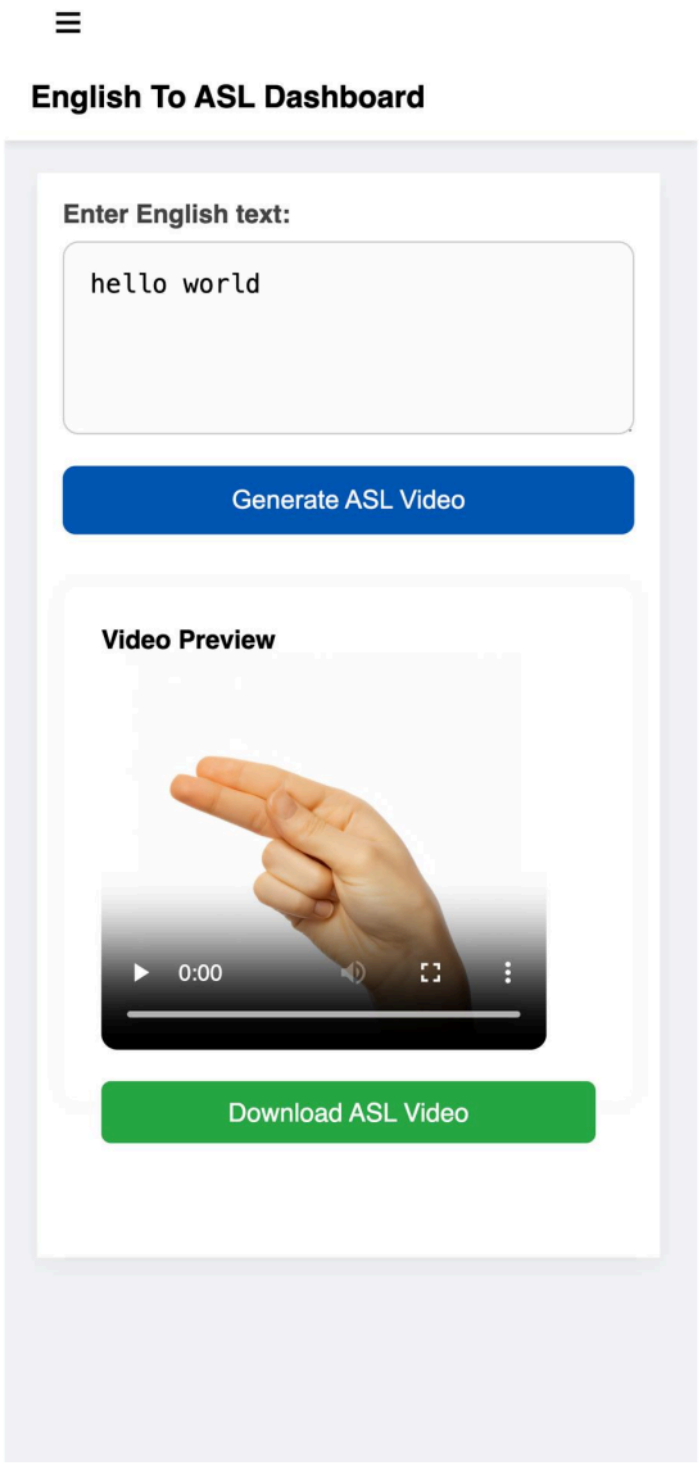

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
User's ASL Video History
Filter by Date:
Rows per Page:
Search:
dd/mm/yyyy
10
Search input text...
Export CSV
Date
User
Input
Frames
Duration (s)
Size (KB)
Video
Action
2025-04-17 03:27 Amrit Ayer HELLO I AM SAMEER 34 17.0 2267.14 Deleted at 2025-04-17 03:30 Delete
2025-04-17 03:27 Amrit Ayer HELLO I AM SAMEER 34 17.0 2267.14 Deleted at 2025-04-17 03:30 Delete
2025-04-16 17:30 Amrit Ayer HELLO 10 5.0 725.76 Deleted at 2025-04-17 00:10 Delete
2025-04-16 17:30 Amrit Ayer HELLO 10 5.0 725.76 Deleted at 2025-04-17 00:10 Delete
2025-04-16 04:47 Amrit Ayer RIYA 8 4.0 998.73 Deleted at 2025-04-16 05:10 Delete
2025-04-16 04:46 Amrit Ayer BYE SAMEER 20 10.0 2514.92 Deleted at 2025-04-16 05:10 Delete
2025-04-16 04:46 Amrit Ayer HI 4 2.0 834.38 Deleted at 2025-04-16 05:10 Delete
2025-04-16 04:45 Amrit Ayer HELLO 10 5.0 1107.84 Deleted at 2025-04-16 05:10 Delete
2025-04-16 04:43 Amrit Ayer HELLO 10 5.0 1107.54 Deleted at 2025-04-16 04:45 Delete
2025-04-16 04:43 Amrit Ayer HI 4 2.0 834.41 Deleted at 2025-04-16 04:45 Delete
Previous
Page 1
Next
Showing 13 of 13 records

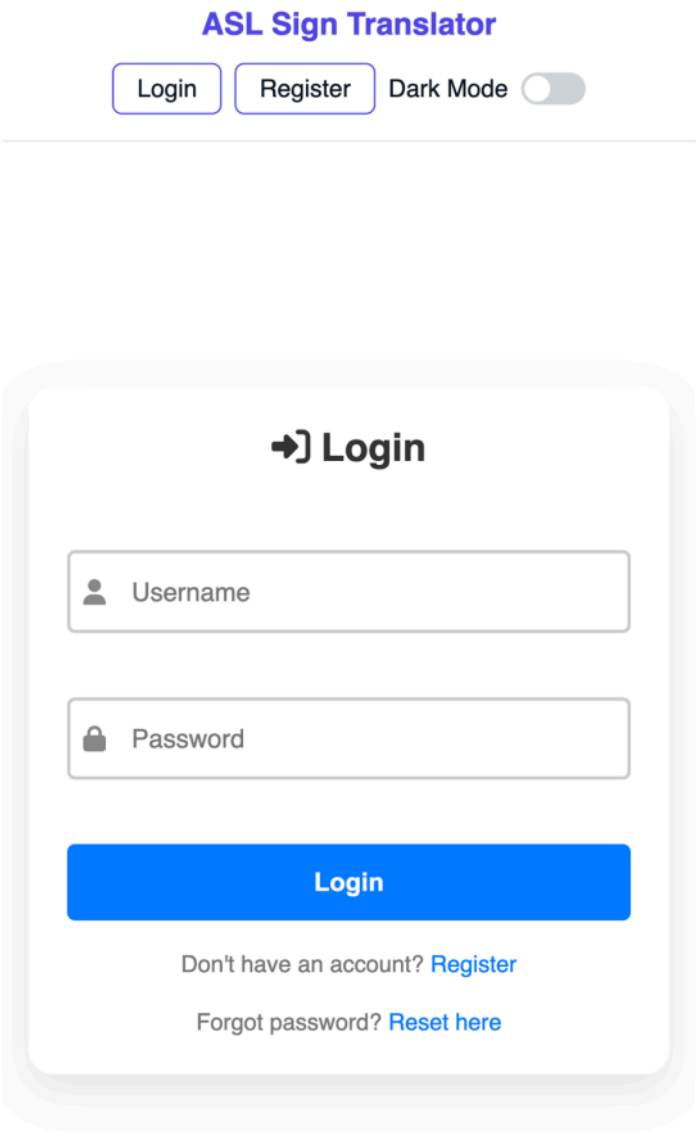

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
User's History
Filter by Date:
Rows per Page:
Search:
dd/mm/yyyy
10
Search...
Export CSV
Timestamp
User's Action
Description
IP Address
User Agent
Action
2025-04-19 16:21:05
Logout
User logged out.
127.0.0.1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Delete
2025-04-19 16:20:59
Page Visit
Visited ASL Video History page
2025-04-19 16:20:51
Visited Sentence Generation History
2025-04-19 16:20:43
Visited User Audit History
2025-04-19 16:20:25
Visited Prediction History page
2025-04-19 14:33:21
Visited ASL Prediction Dashboard
2025-04-17 09:34:36
2025-04-17 09:34:36
Login
User logged in.
2025-04-17 03:28:54
Logout
User logged out.
Mozilla/5.0 (iPhone; CPU iPhone OS 16_6 like Mac ...
2025-04-17 03:28:34
Visited ASL Video History page
Previous
Page
1
Next
Showing 476 of 476 records

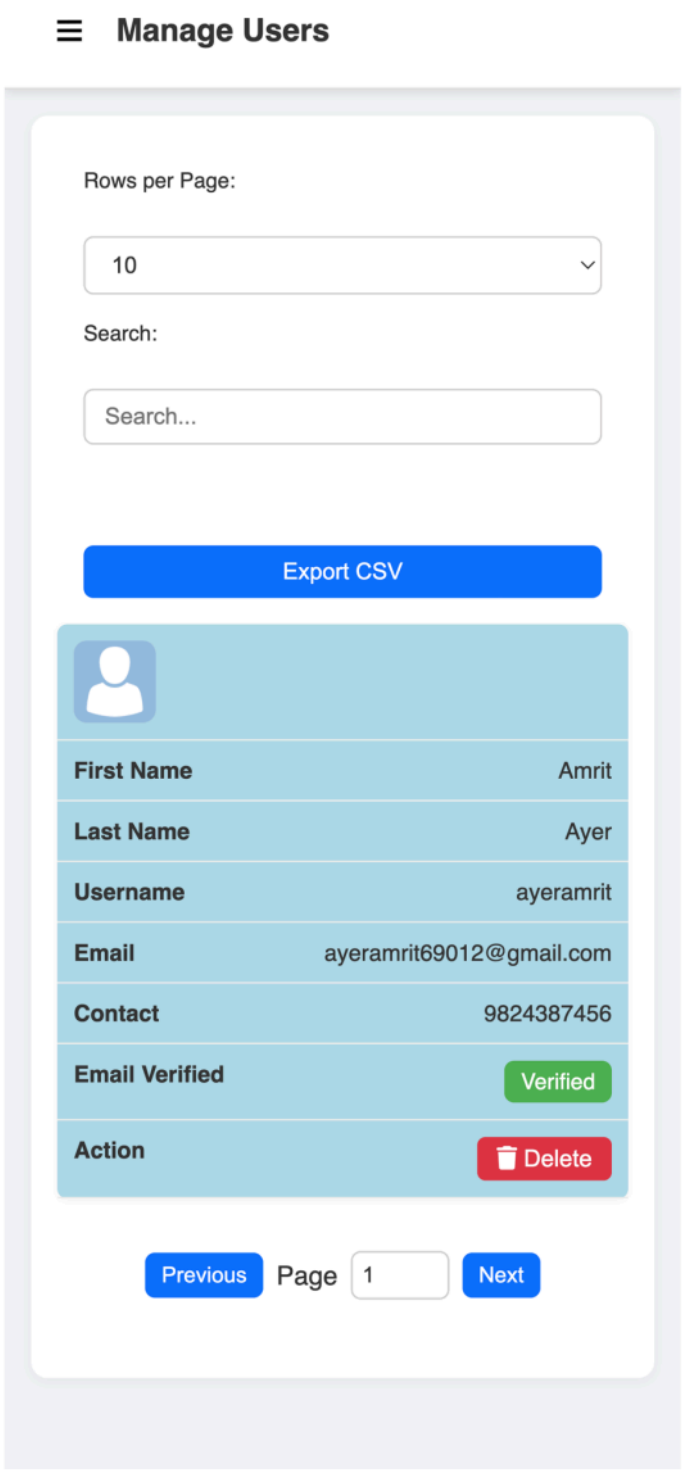

ASL Sign Translator
Sameer Basnet
Dark Mode
Dashboard
English to ASL
Profile
Prediction History
User History
Sentence History
Video History
Users Prediction History
Users Audit History
Users Sentence History
User's Video History
Manage Users
Logout
User's Sentence Generation History
Filter by Date:
Rows per Page:
Search:
dd/mm/yyyy
10
Search sentence...
Export CSV
Date
User
Input Predictions
Generated Sentence
Action
2025-04-17 03:27
Amrit Ayer
A, B, C
There is no universally agreed-upon meaning for the sequence "ABC" as signed inputs. More context is needed. To create a sentence, we need to know what the signs represent. For example, are they: * **Morse code:** If so, ABC isn't a standard sequence. * **Flags or signals:** Again, meaning would depend on the system used. * **Short hand for something:** (e.g., a code used internally by a company) Without this context, any sentence I create would be pure speculation.
Delete
2025-04-16 15:16
A, B, I, E, I, E
This looks like a substitution cipher where each letter represents another. Without more information or a key, it's impossible to definitively convert "ABIEIE" into a grammatically correct English sentence.
Previous
Page
1
Next
Showing 2 of 2 records

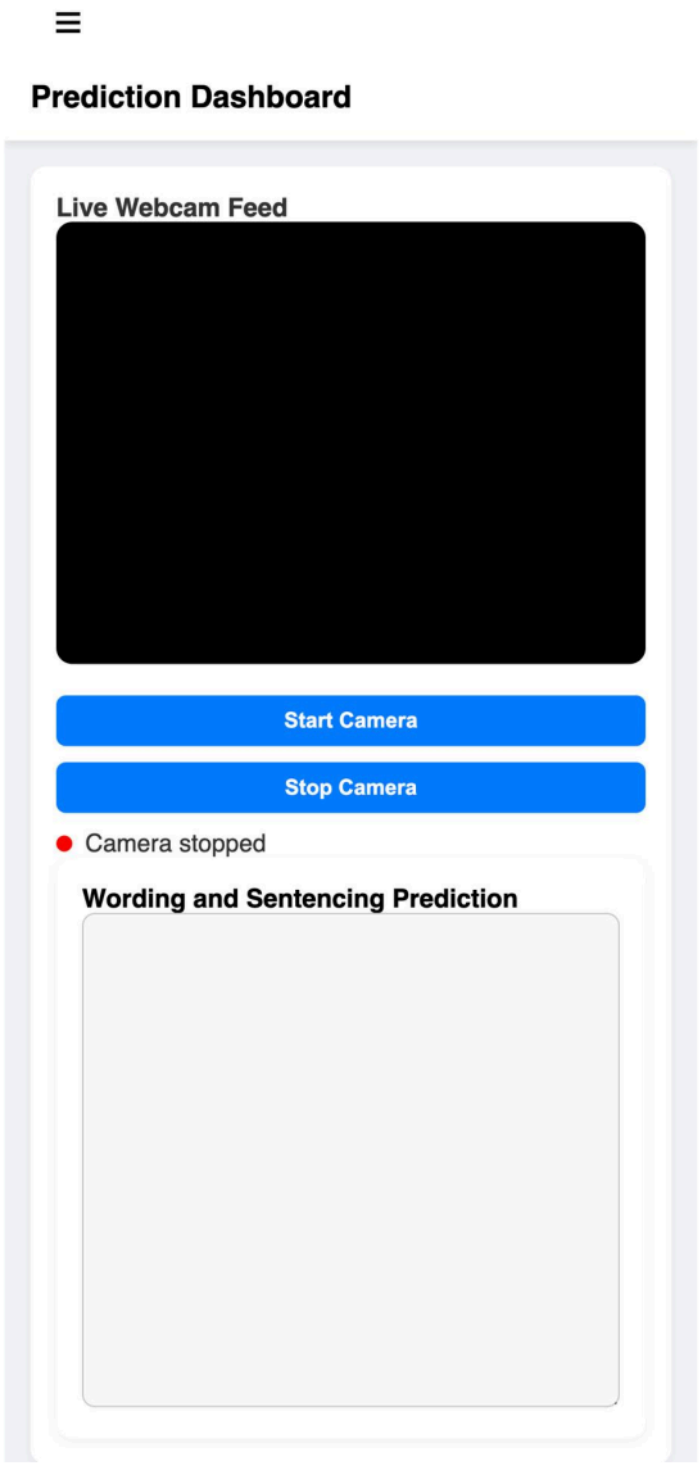
11.2 Mobile UI Screens

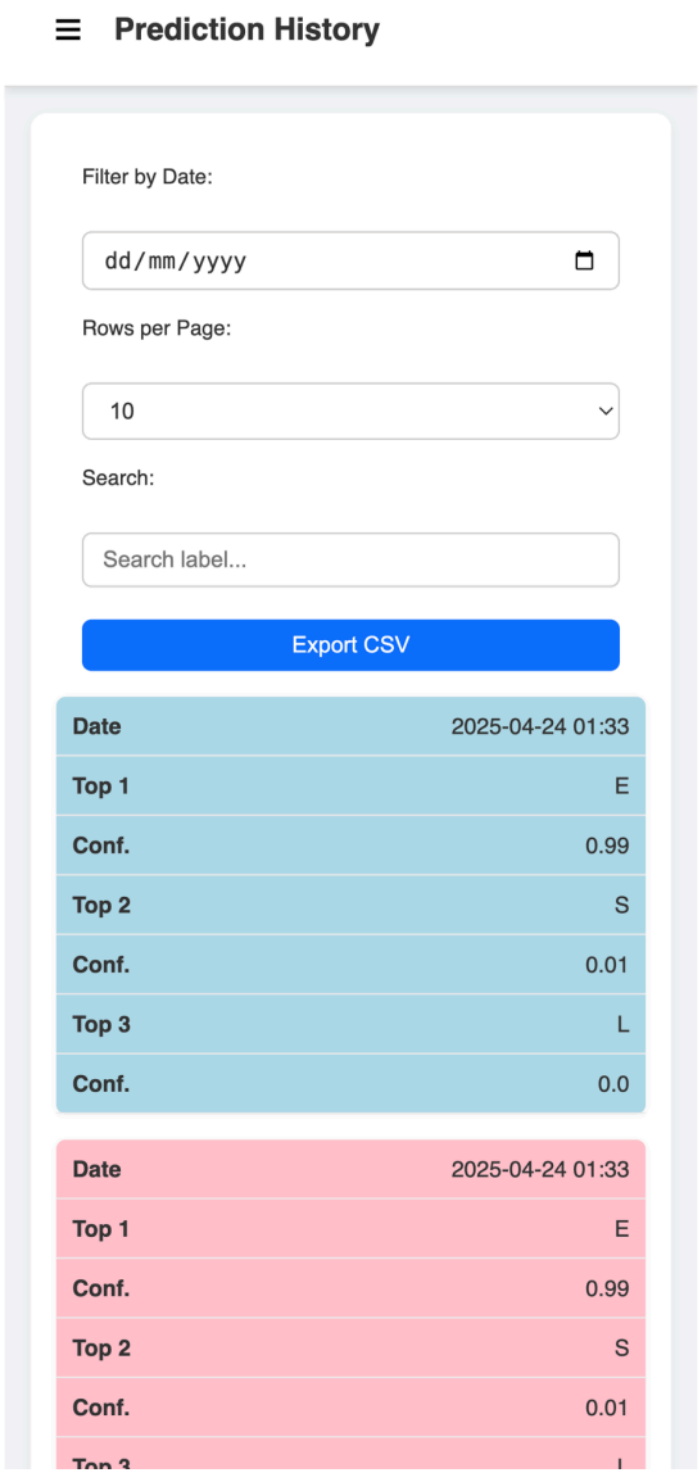


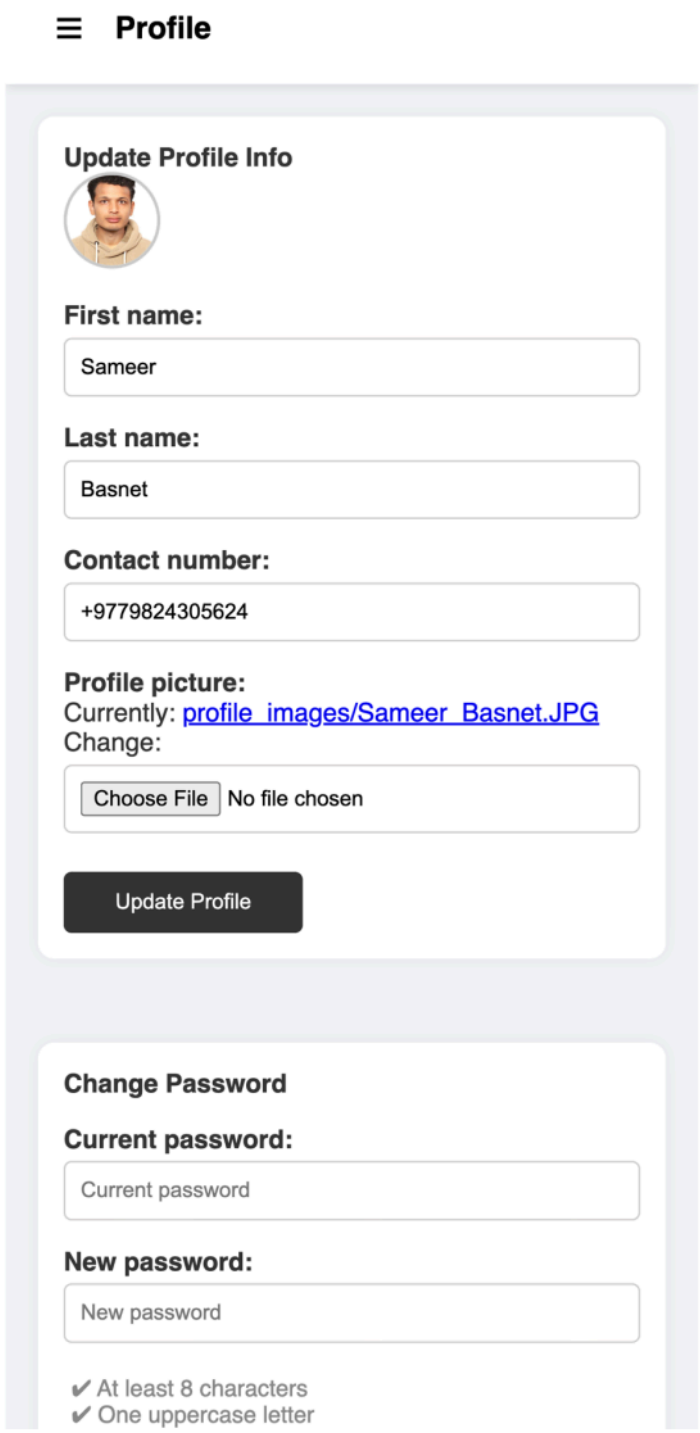

English To ASL Dashboard
Enter English text:
hello world
Generate ASL Video
Video Preview
0:00
Download ASL Video

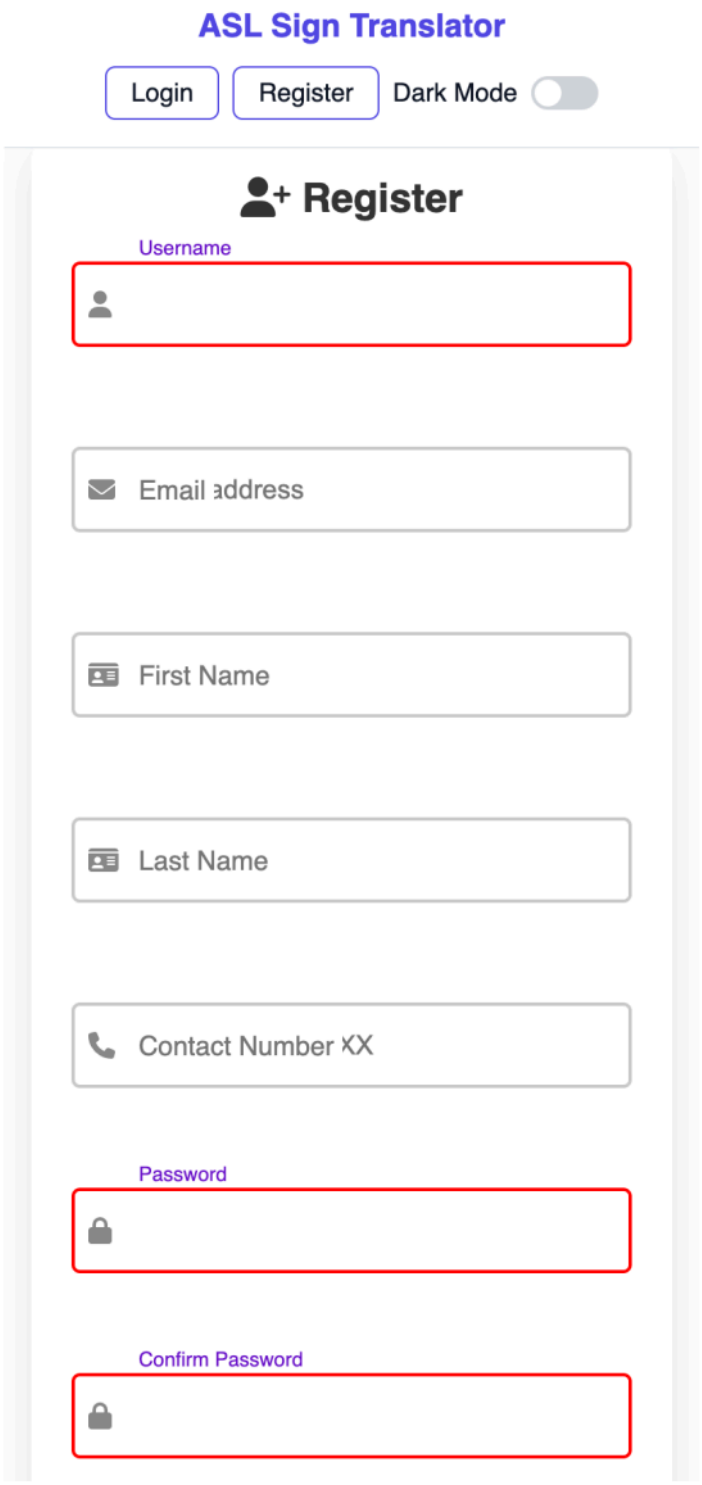

ASL Sign Translator
Login
Register
Dark Mode
Login
Username
Password
Login
Don't have an account? Register
Forgot password? Reset here

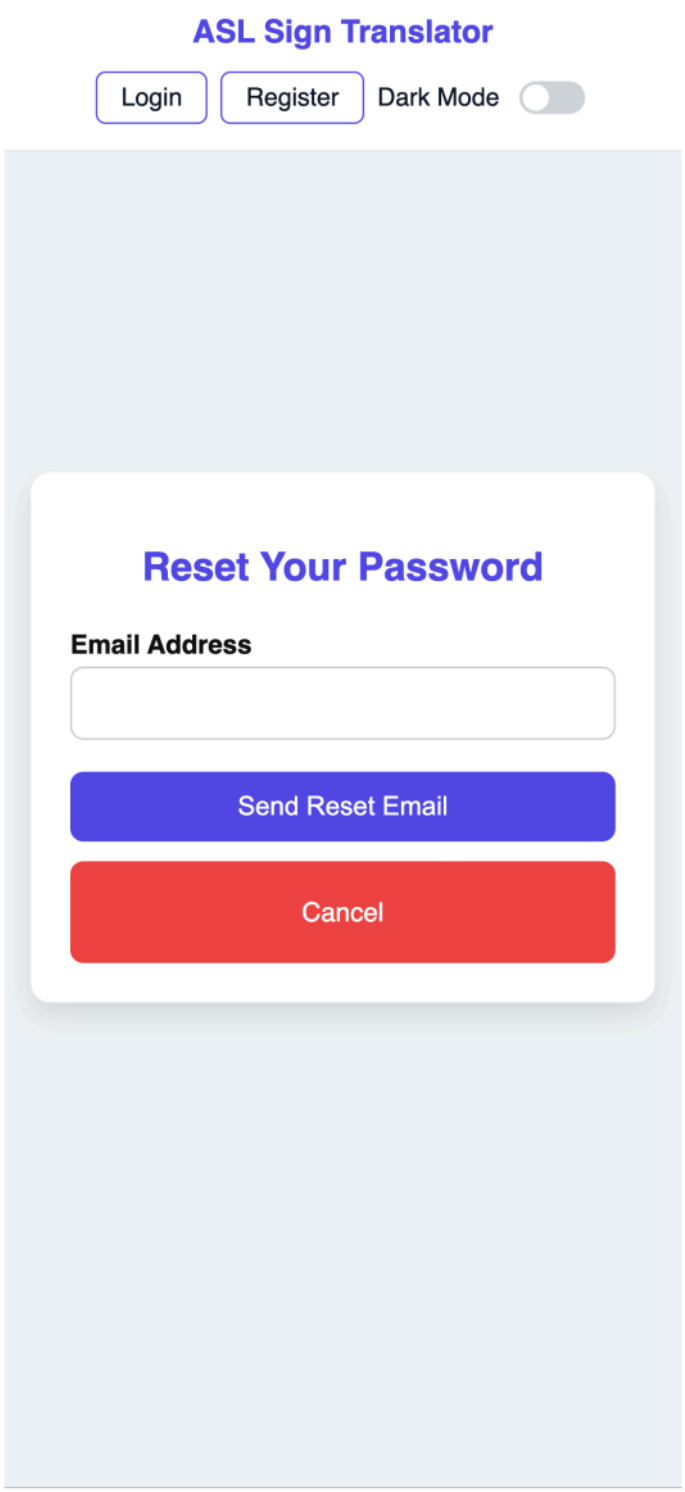

Manage Users
Rows per Page:
10
Search:
Search...
Export CSV
First Name
Amrit
Last Name
Ayer
Username
ayeramrit
Email
ayeramrit69012@gmail.com
Contact
9824387456
Email Verified
Verified
Action
Delete
Previous
Page
1
Next

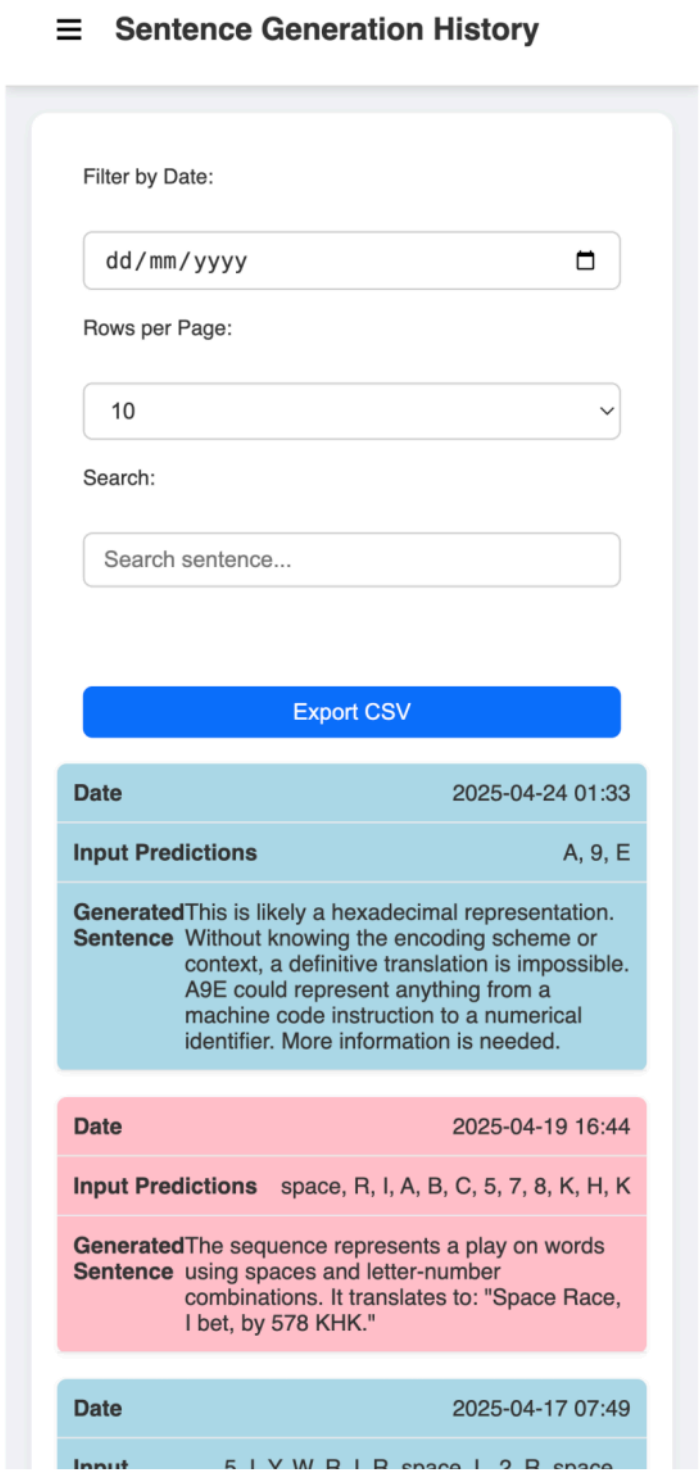

Prediction Dashboard
Live Webcam Feed
Start Camera
Stop Camera
Camera stopped
Wording and Sentencing Prediction

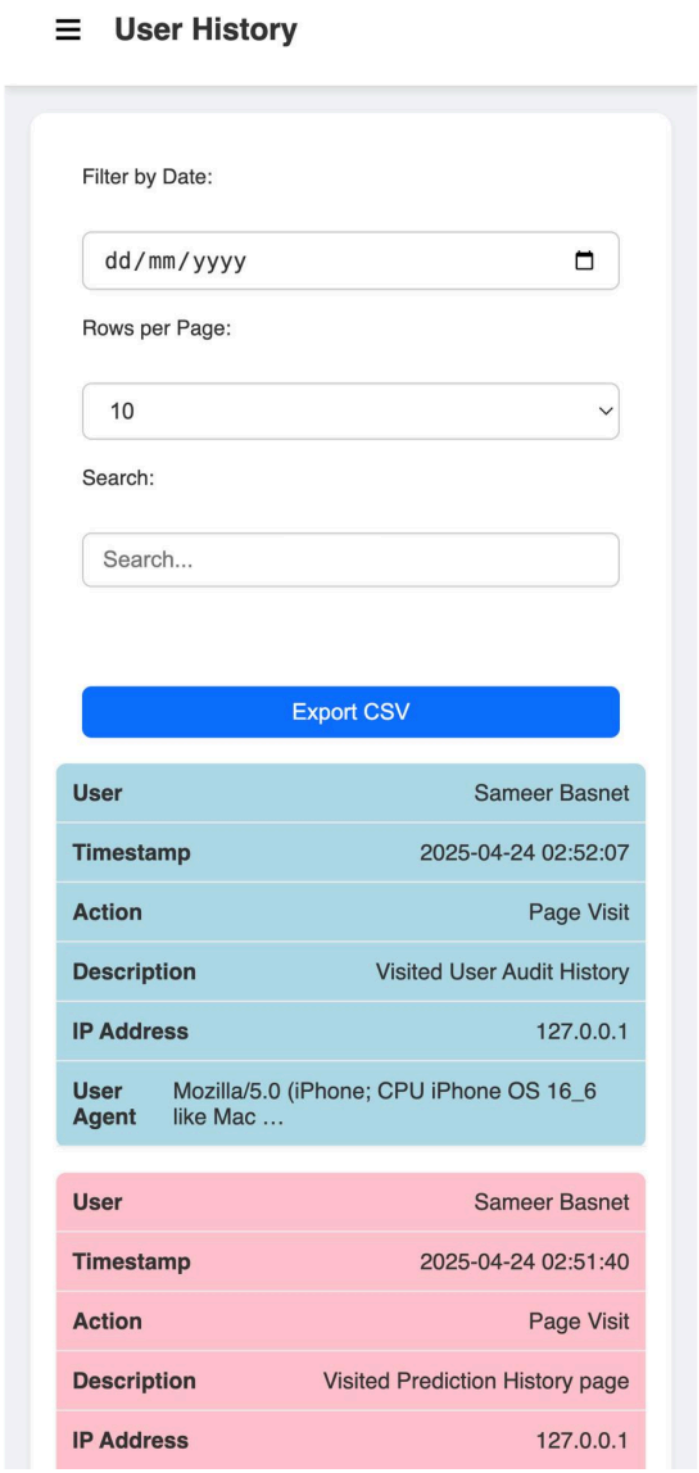

Prediction History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search label...
Export CSV
Date 2025-04-24 01:33
Top 1 E
Conf. 0.99
Top 2 S
Conf. 0.01
Top 3 L
Conf. 0.0
Date 2025-04-24 01:33
Top 1 E
Conf. 0.99
Top 2 S
Conf. 0.01

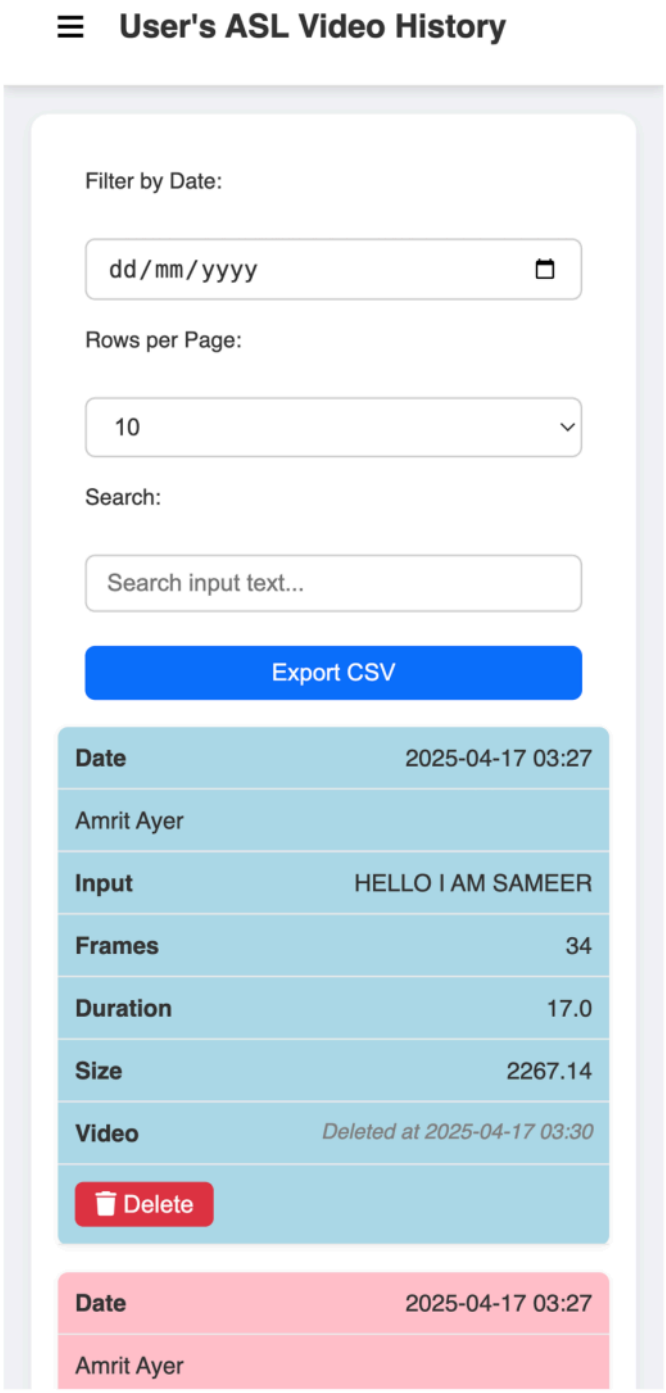

Profile
Update Profile Info
First name:
Sameer
Last name:
Basnet
Contact number:
+9779824305624
Profile picture:
Currently: profile_images/Sameer_Basnet.JPG
Change:
Choose File
No file chosen
Update Profile
Change Password
Current password:
Current password
New password:
New password
✔ At least 8 characters
✔ One uppercase letter

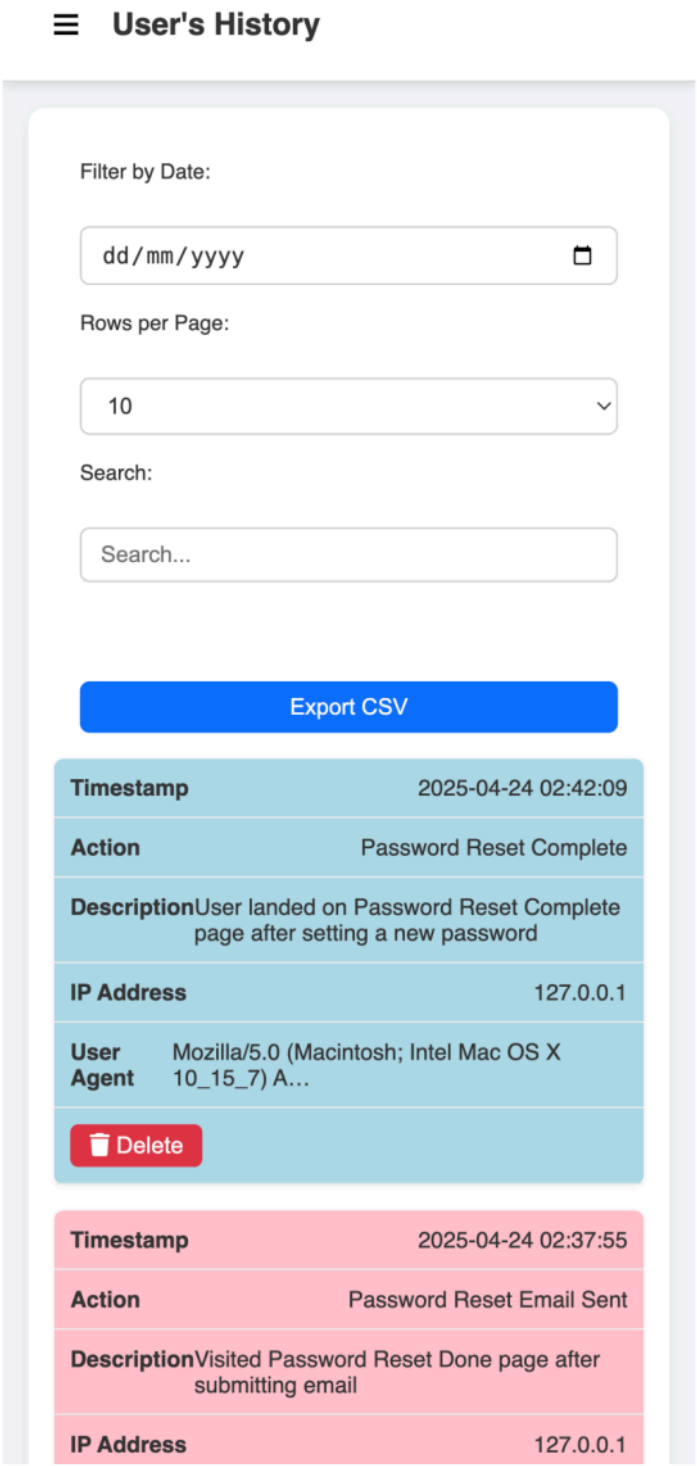

ASL Sign Translator
Login
Register
Dark Mode
Register
Username
Email address
First Name
Last Name
Contact Number XX
Password
Confirm Password

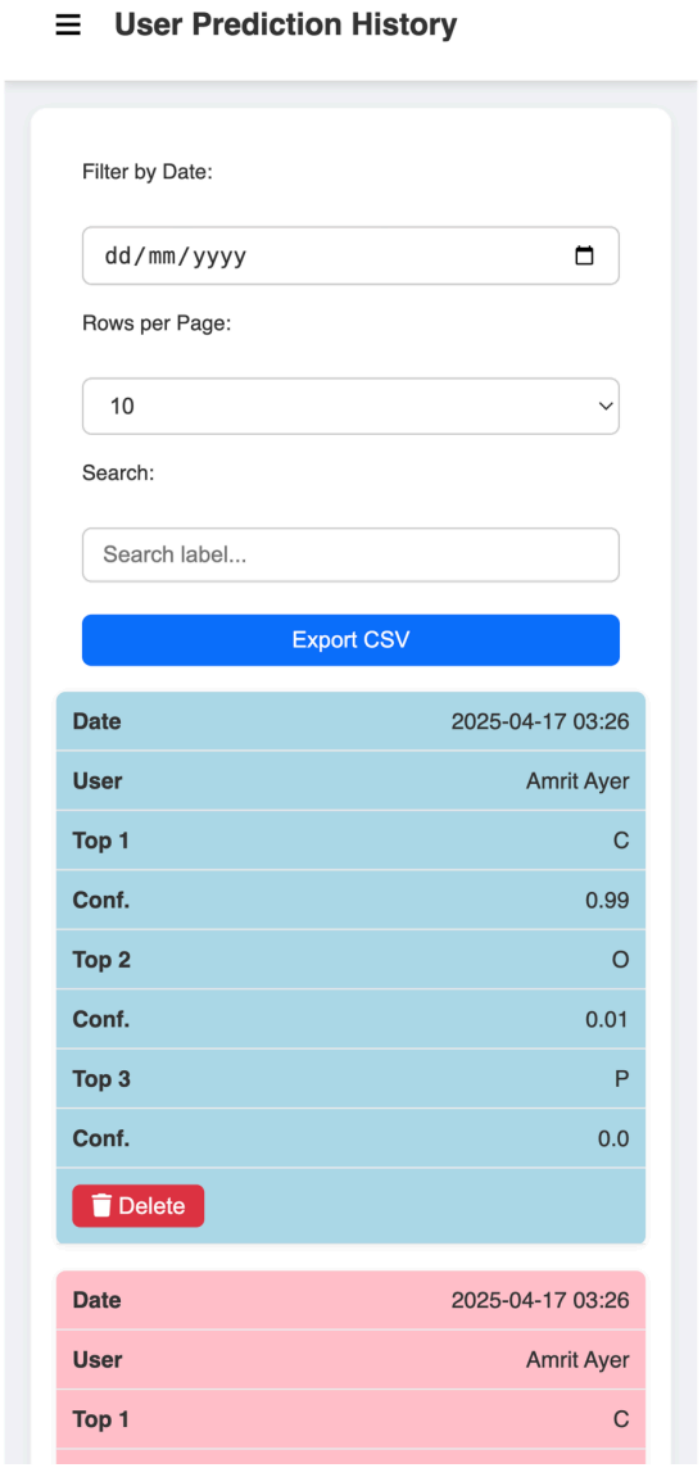

ASL Sign Translator
Login
Register
Dark Mode
Reset Your Password
Email Address
Send Reset Email
Cancel

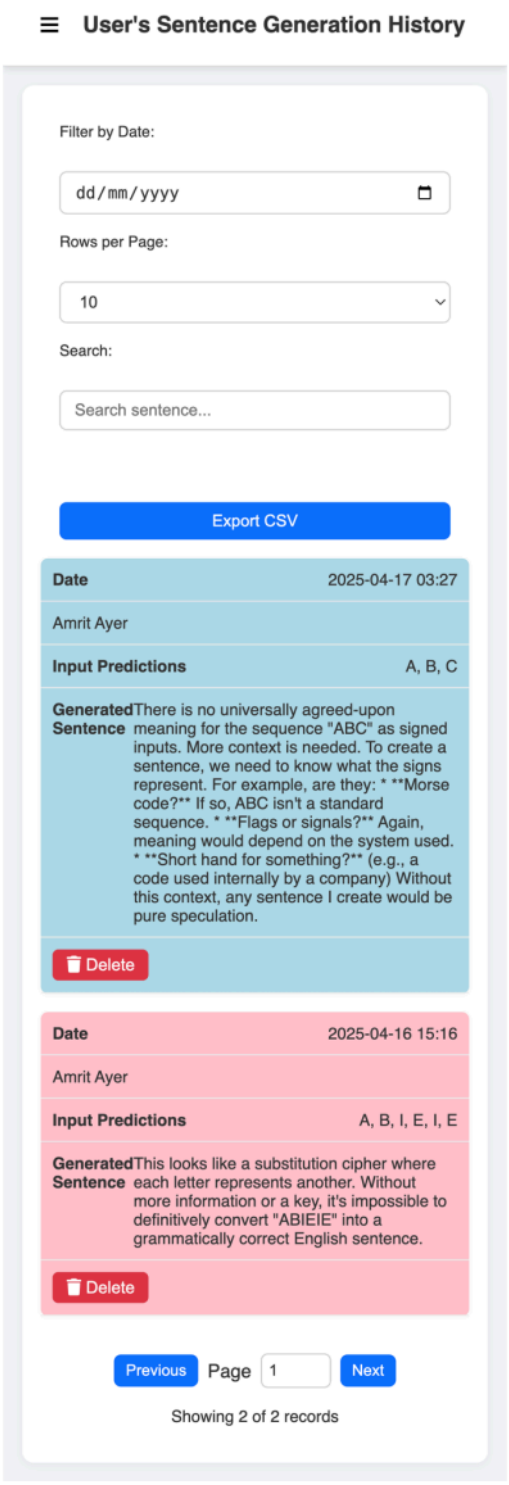

Sentence Generation History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search sentence...
Export CSV
Date 2025-04-24 01:33
Input Predictions A, 9, E
Generated Sentence This is likely a hexadecimal representation. Without knowing the encoding scheme or context, a definitive translation is impossible. A9E could represent anything from a machine code instruction to a numerical identifier. More information is needed.
Date 2025-04-19 16:44
Input Predictions space, R, I, A, B, C, 5, 7, 8, K, H, K
Generated Sentence The sequence represents a play on words using spaces and letter-number combinations. It translates to: "Space Race, I bet, by 578 KHK."
Date 2025-04-17 07:49


User History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search...
Export CSV
User
Sameer Basnet
Timestamp
2025-04-24 02:52:07
Action
Page Visit
Description
Visited User Audit History
IP Address
127.0.0.1
User Agent
Mozilla/5.0 (iPhone; CPU iPhone OS 16_6 like Mac ...
User
Sameer Basnet
Timestamp
2025-04-24 02:51:40
Action
Page Visit
Description
Visited Prediction History page
IP Address
127.0.0.1

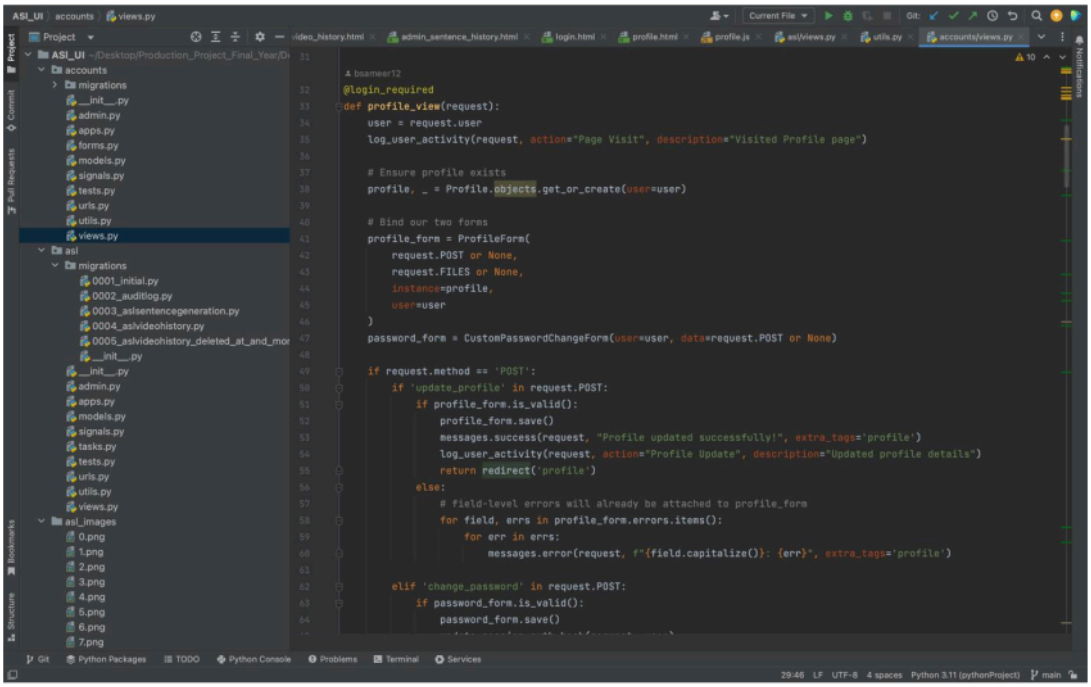

User's ASL Video History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search input text...
Export CSV
Date 2025-04-17 03:27
Amrit Ayer
Input HELLO I AM SAMEER
Frames 34
Duration 17.0
Size 2267.14
Video Deleted at 2025-04-17 03:30
Delete
Date 2025-04-17 03:27
Amrit Ayer


User's History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search...
Export CSV
Timestamp 2025-04-24 02:42:09
Action Password Reset Complete
Description User landed on Password Reset Complete page after setting a new password
IP Address 127.0.0.1
User Agent Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A...
Delete
Timestamp 2025-04-24 02:37:55
Action Password Reset Email Sent
Description Visited Password Reset Done page after submitting email
IP Address 127.0.0.1

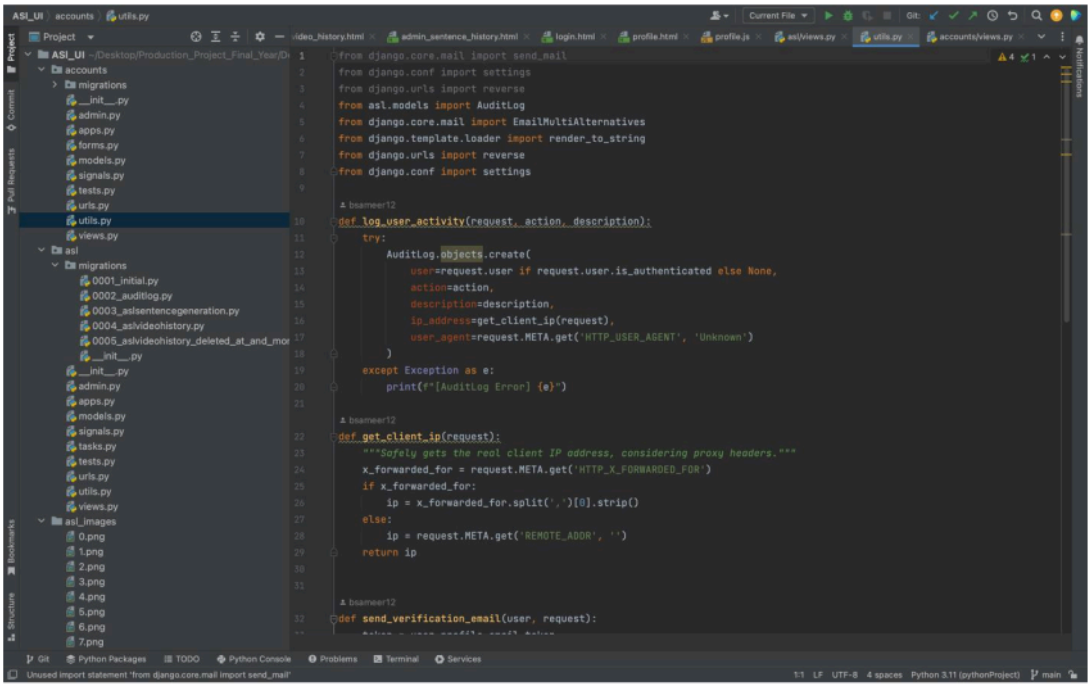

User Prediction History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search label...
Export CSV
Date 2025-04-17 03:26
User Amrit Ayer
Top 1 C
Conf. 0.99
Top 2 O
Conf. 0.01
Top 3 P
Conf. 0.0
Delete
Date 2025-04-17 03:26
User Amrit Ayer
Top 1 C


User's Sentence Generation History
Filter by Date:
dd/mm/yyyy
Rows per Page:
10
Search:
Search sentence...
Export CSV
Date 2025-04-17 03:27
Amrit Ayer
Input Predictions A, B, C
Generated Sentence There is no universally agreed-upon meaning for the sequence "ABC" as signed inputs. More context is needed. To create a sentence, we need to know what the signs represent. For example, are they: * **Morse code?** If so, ABC isn't a standard sequence. * **Flags or signals?** Again, meaning would depend on the system used. * **Short hand for something?** (e.g., a code used internally by a company) Without this context, any sentence I create would be pure speculation.
Delete
Date 2025-04-16 15:16
Amrit Ayer
Input Predictions A, B, I, E, I, E
Generated Sentence This looks like a substitution cipher where each letter represents another. Without more information or a key, it's impossible to definitively convert "ABIEIE" into a grammatically correct English sentence.
Delete
Previous
Page 1
Next
Showing 2 of 2 records

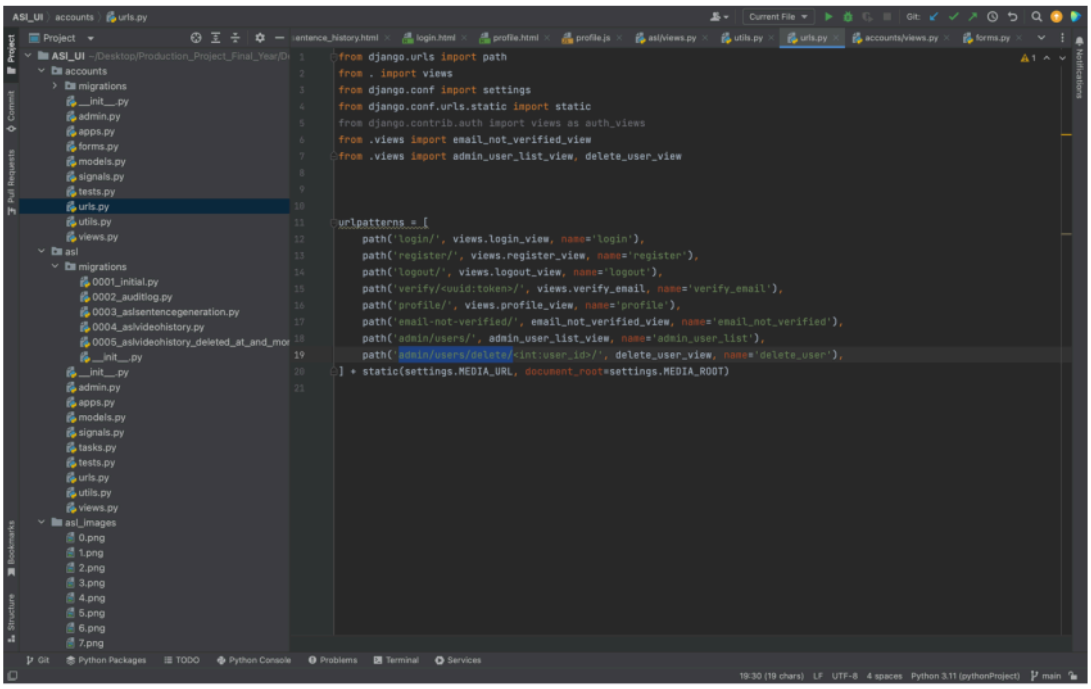
11.3 Coding Screenshots



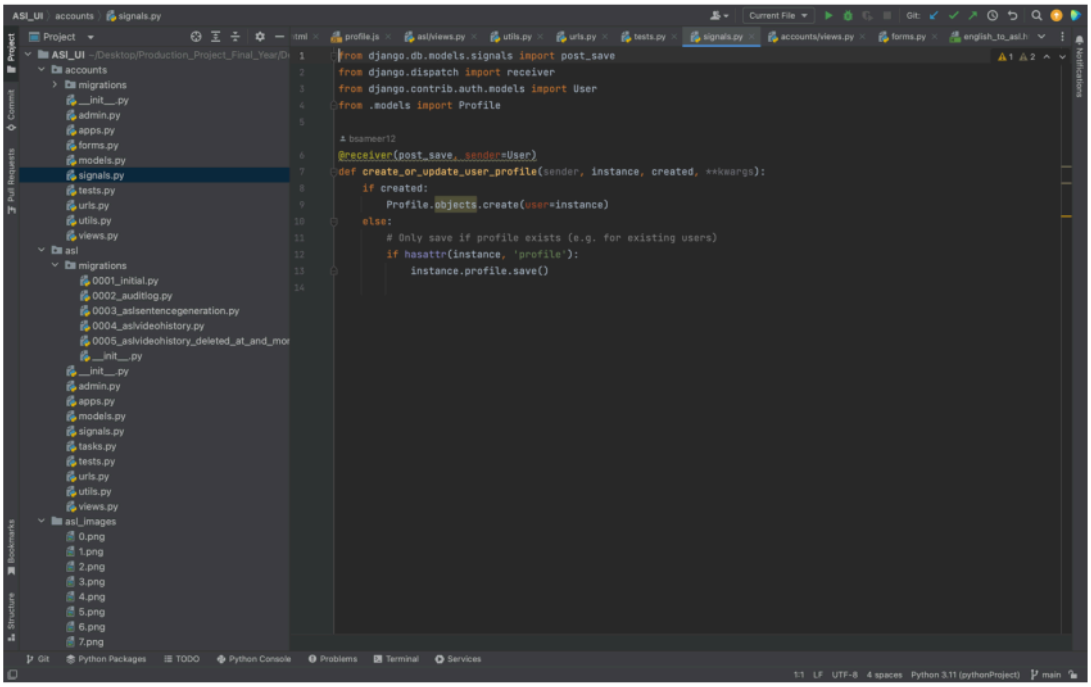
```
@login_required
def profile_view(request):
    user = request.user
    log_user_activity(request, action="Page Visit", description="Visited Profile page")

    # Ensure profile exists
    profile, _ = Profile.objects.get_or_create(user=user)

    # Bind our two forms
    profile_form = ProfileForm(
        request.POST or None,
        request.FILES or None,
        instance=profile,
        user=user
    )
    password_form = CustomPasswordChangeForm(user=user, data=request.POST or None)

    if request.method == 'POST':
        if 'update_profile' in request.POST:
            if profile_form.is_valid():
                profile_form.save()
                messages.success(request, "Profile updated successfully!", extra_tags='profile')
                log_user_activity(request, action="Profile Update", description="Updated profile details")
                return redirect('profile')
            else:
                # field-level errors will already be attached to profile_form
                for field, errs in profile_form.errors.items():
                    for err in errs:
                        messages.error(request, f"{field.capitalize()}: {err}", extra_tags='profile')

        elif 'change_password' in request.POST:
            if password_form.is_valid():
                password_form.save()
```



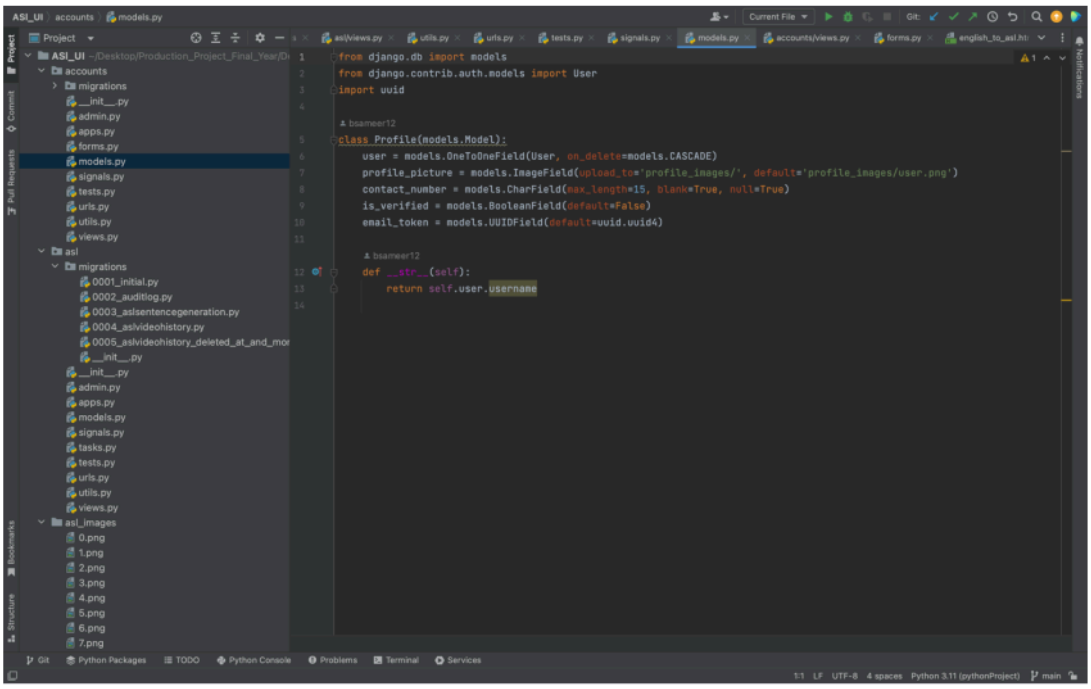
```
from django.core.mail import send_mail
from django.conf import settings
from django.urls import reverse
from asl.models import AuditLog
from django.core.mail import EmailMultiAlternatives
from django.template.loader import render_to_string
from django.urls import reverse
from django.conf import settings

def log_user_activity(request, action, description):
    try:
        AuditLog.objects.create(
            user=request.user if request.user.is_authenticated else None,
            action=action,
            description=description,
            ip_address=get_client_ip(request),
            user_agent=request.META.get('HTTP_USER_AGENT', 'Unknown')
        )
    except Exception as e:
        print(f"[AuditLog Error] {e}")


def get_client_ip(request):
    """Safely gets the real client IP address, considering proxy headers."""
    x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
    if x_forwarded_for:
        ip = x_forwarded_for.split(',')[0].strip()
    else:
        ip = request.META.get('REMOTE_ADDR', '')
    return ip


def send_verification_email(user, request):
```



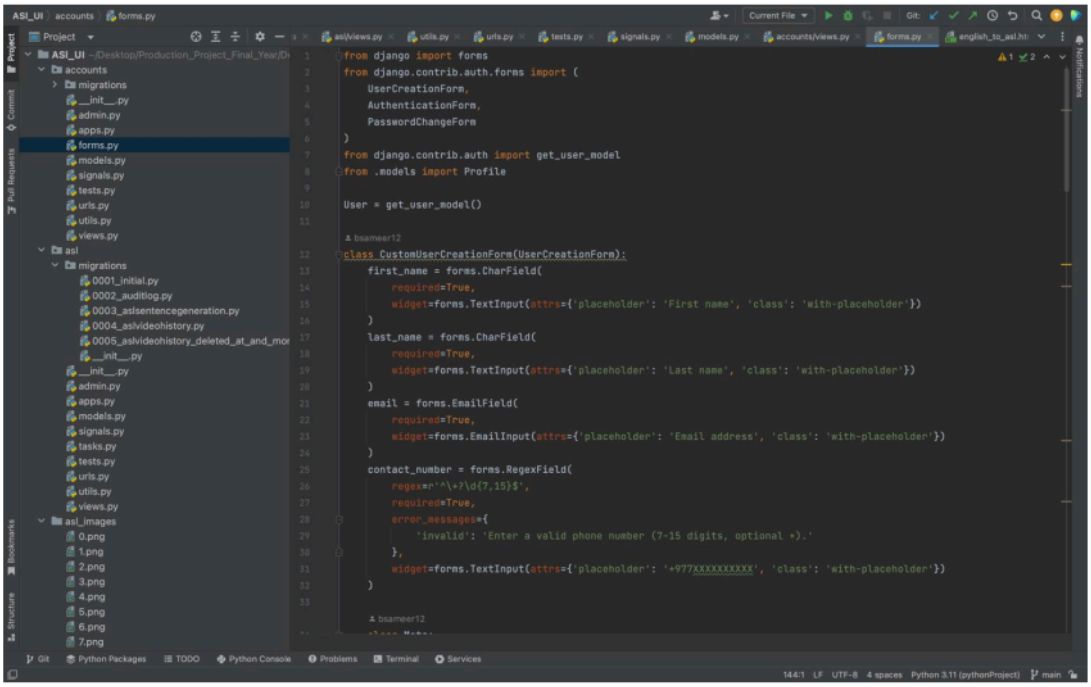
```
from django.urls import path
from . import views
from django.conf import settings
from django.conf.urls.static import static
from django.contrib.auth import views as auth_views
from .views import email_not_verified_view
from .views import admin_user_list_view, delete_user_view


urlpatterns = [
    path('login/', views.login_view, name='login'),
    path('register/', views.register_view, name='register'),
    path('logout/', views.logout_view, name='logout'),
    path('verify/<uuid:token>/', views.verify_email, name='verify_email'),
    path('profile/', views.profile_view, name='profile'),
    path('email-not-verified/', email_not_verified_view, name='email_not_verified'),
    path('admin/users/', admin_user_list_view, name='admin_user_list'),
    path('admin/users/delete/<int:user_id>/', delete_user_view, name='delete_user'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```



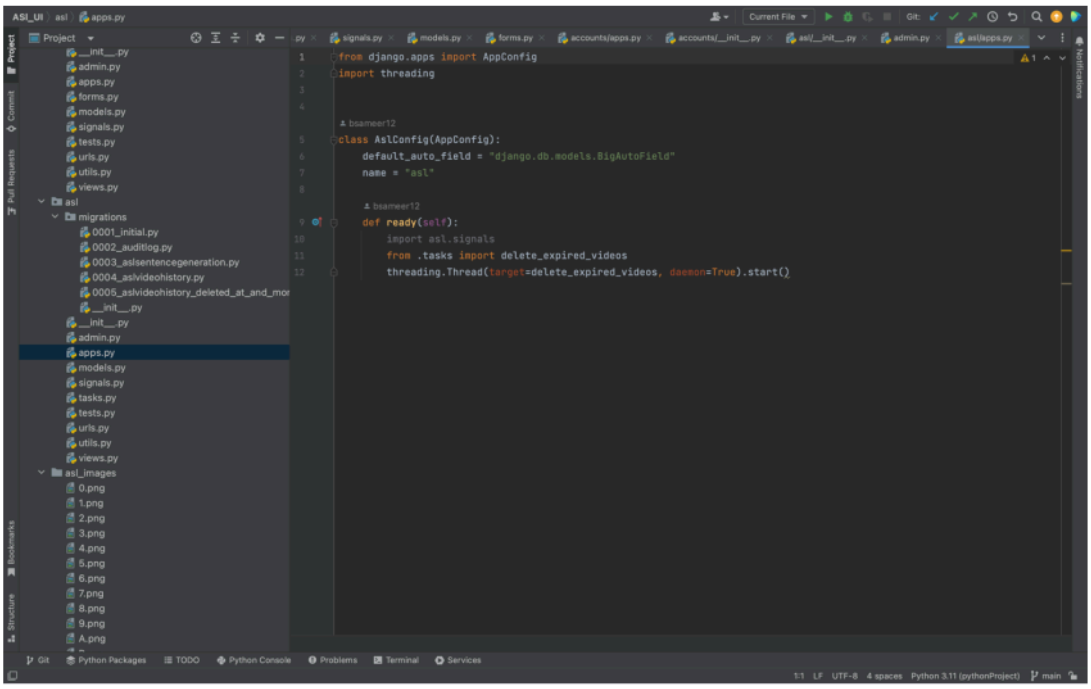
```
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.contrib.auth.models import User
from .models import Profile


@receiver(post_save, sender=User)
def create_or_update_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)
    else:
        # Only save if profile exists (e.g. for existing users)
        if hasattr(instance, 'profile'):
            instance.profile.save()
```

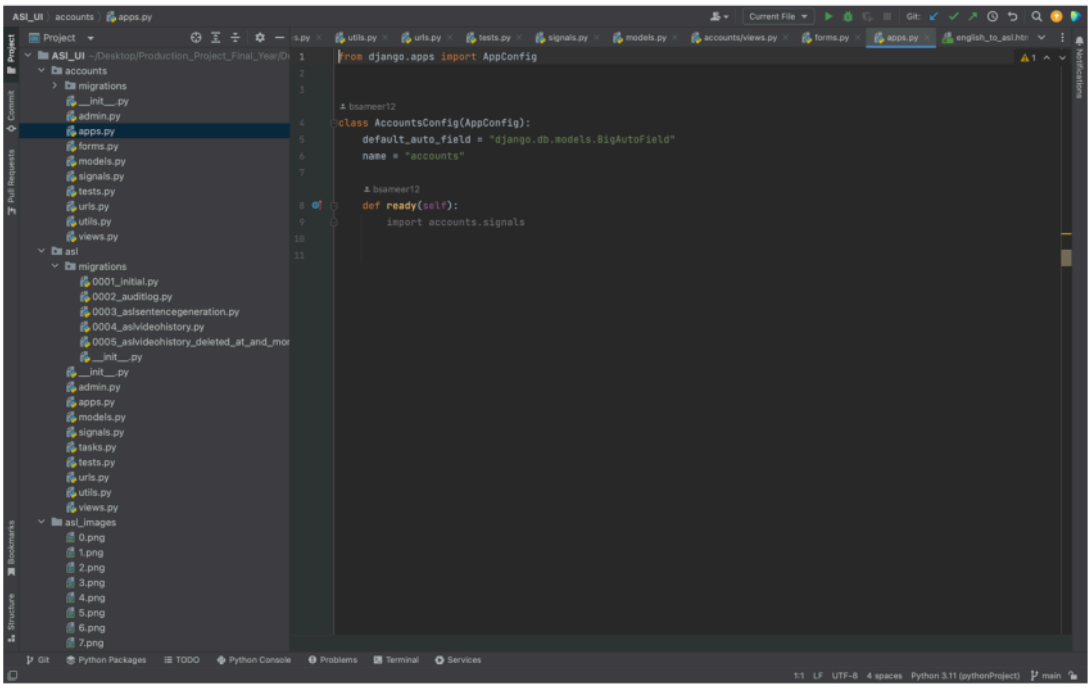


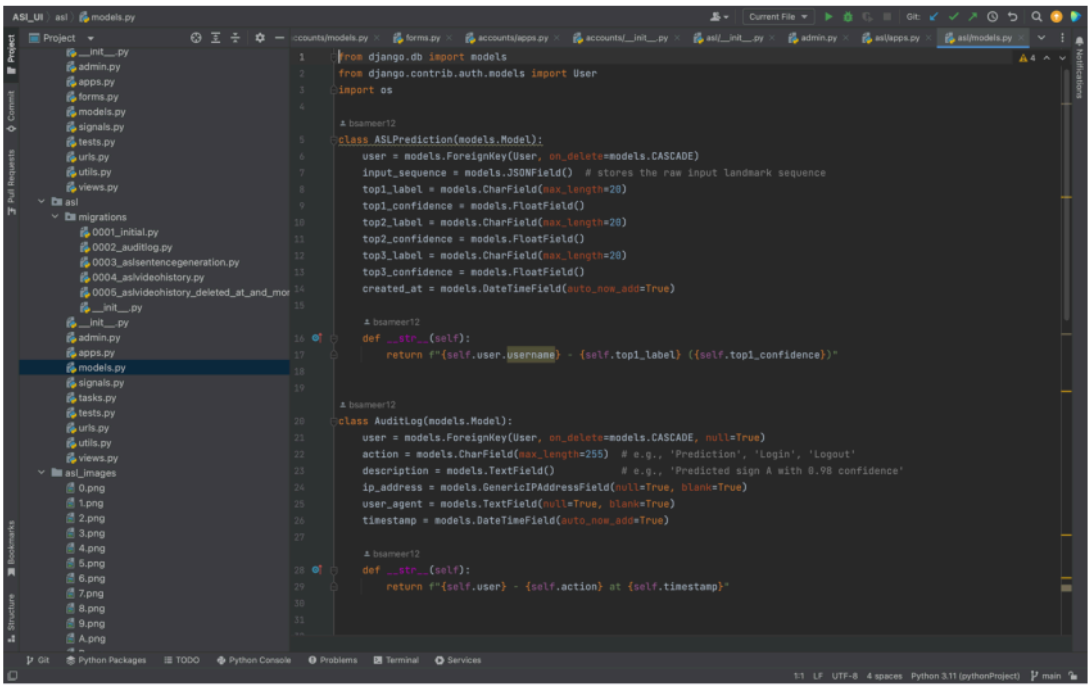
```python
from django.db import models
from django.contrib.auth.models import User
import uuid

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    profile_picture = models.ImageField(upload_to='profile_images/', default='profile_images/user.png')
    contact_number = models.CharField(max_length=15, blank=True, null=True)
    is_verified = models.BooleanField(default=False)
    email_token = models.UUIDField(default=uuid.uuid4)

    def __str__(self):
        return self.user.username
```



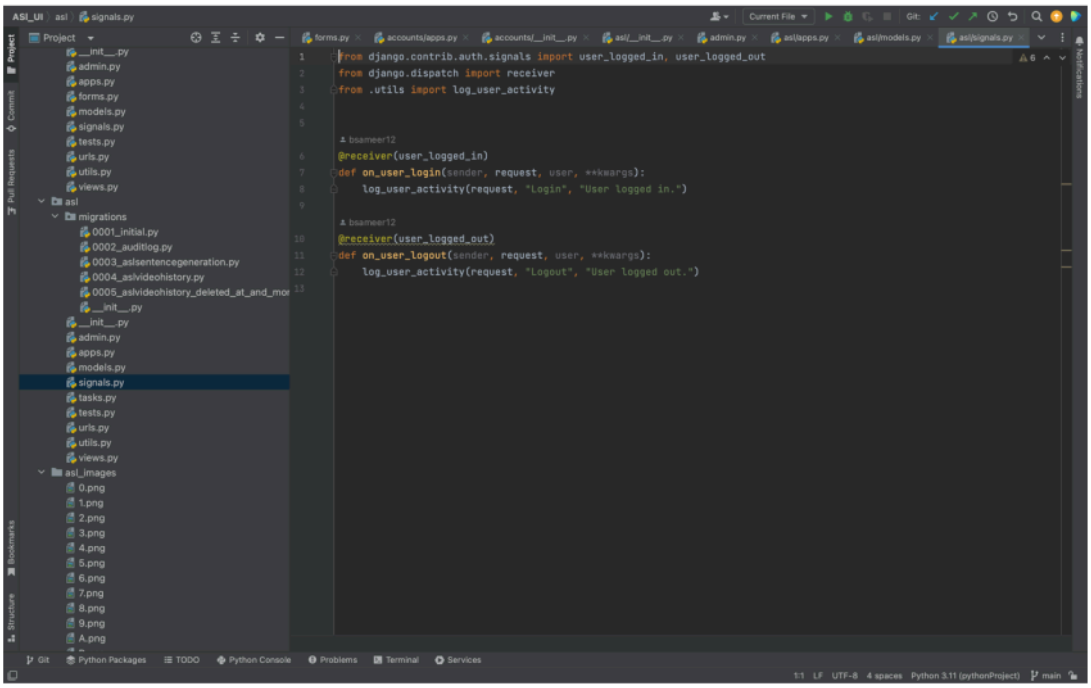
```python
from django import forms
from django.contrib.auth.forms import (
    UserCreationForm,
    AuthenticationForm,
    PasswordChangeForm
)
from django.contrib.auth import get_user_model
from .models import Profile

User = get_user_model()

class CustomUserCreationForm(UserCreationForm):
    first_name = forms.CharField(
        required=True,
        widget=forms.TextInput(attrs={'placeholder': 'First name', 'class': 'with-placeholder'})
    )
    last_name = forms.CharField(
        required=True,
        widget=forms.TextInput(attrs={'placeholder': 'Last name', 'class': 'with-placeholder'})
    )
    email = forms.EmailField(
        required=True,
        widget=forms.EmailInput(attrs={'placeholder': 'Email address', 'class': 'with-placeholder'})
    )
    contact_number = forms.RegexField(
        regex=r'^\+?\d{7,15}$',
        required=True,
        error_messages={
            'invalid': 'Enter a valid phone number (7-15 digits, optional +).'
        },
        widget=forms.TextInput(attrs={'placeholder': '+977XXXXXXXXX', 'class': 'with-placeholder'})
    )
```


ASL_UI asl apps.py
Project
__init__.py
admin.py
apps.py
forms.py
models.py
signals.py
tests.py
urls.py
utils.py
views.py
asl
migrations
0001_initial.py
0002_auditlog.py
0003_aslsentencegeneration.py
0004_aslvideohistory.py
0005_aslvideohistory_deleted_at_and_mor
__init__.py
__init__.py
admin.py
apps.py
models.py
signals.py
tasks.py
tests.py
urls.py
utils.py
views.py
asl_images
0.png
1.png
2.png
3.png
4.png
5.png
6.png
7.png
8.png
9.png
A.png
signals.py
models.py
forms.py
accounts/apps.py
accounts/__init__.py
asl/__init__.py
admin.py
asl/apps.py
from django.apps import AppConfig
import threading
bsameer12
class AslConfig(AppConfig):
default_auto_field = "django.db.models.BigAutoField"
name = "asl"
bsameer12
def ready(self):
import asl.signals
from .tasks import delete_expired_videos
threading.Thread(target=delete_expired_videos, daemon=True).start()
Git
Python Packages
TODO
Python Console
Problems
Terminal
Services
1:1 LF UTF-8 4 spaces Python 3.11 (pythonProject)
main

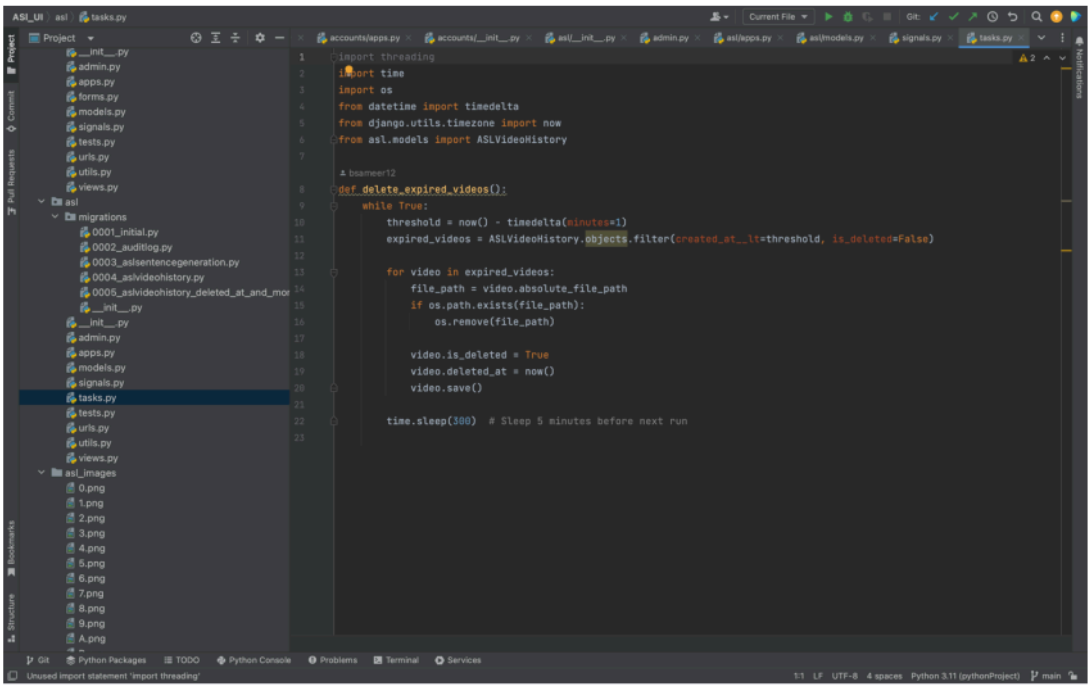

ASL_UI accounts apps.py
Project
ASL_UI ~/Desktop/Production_Project_Final_Year/D
accounts
migrations
__init__.py
admin.py
apps.py
forms.py
models.py
signals.py
tests.py
urls.py
utils.py
views.py
asl
migrations
0001_initial.py
0002_auditlog.py
0003_aslsentencegeneration.py
0004_aslvideohistory.py
0005_aslvideohistory_deleted_at_and_mor
__init__.py
__init__.py
admin.py
apps.py
models.py
signals.py
tasks.py
tests.py
urls.py
utils.py
views.py
asl_images
0.png
1.png
2.png
3.png
4.png
5.png
6.png
7.png
utils.py
urls.py
tests.py
signals.py
models.py
accounts/views.py
forms.py
apps.py
from django.apps import AppConfig
bsameer12
class AccountsConfig(AppConfig):
default_auto_field = "django.db.models.BigAutoField"
name = "accounts"
bsameer12
def ready(self):
import accounts.signals
Git
Python Packages
TODO
Python Console
Problems
Terminal
Services
1:1 LF UTF-8 4 spaces Python 3.11 (pythonProject)
main



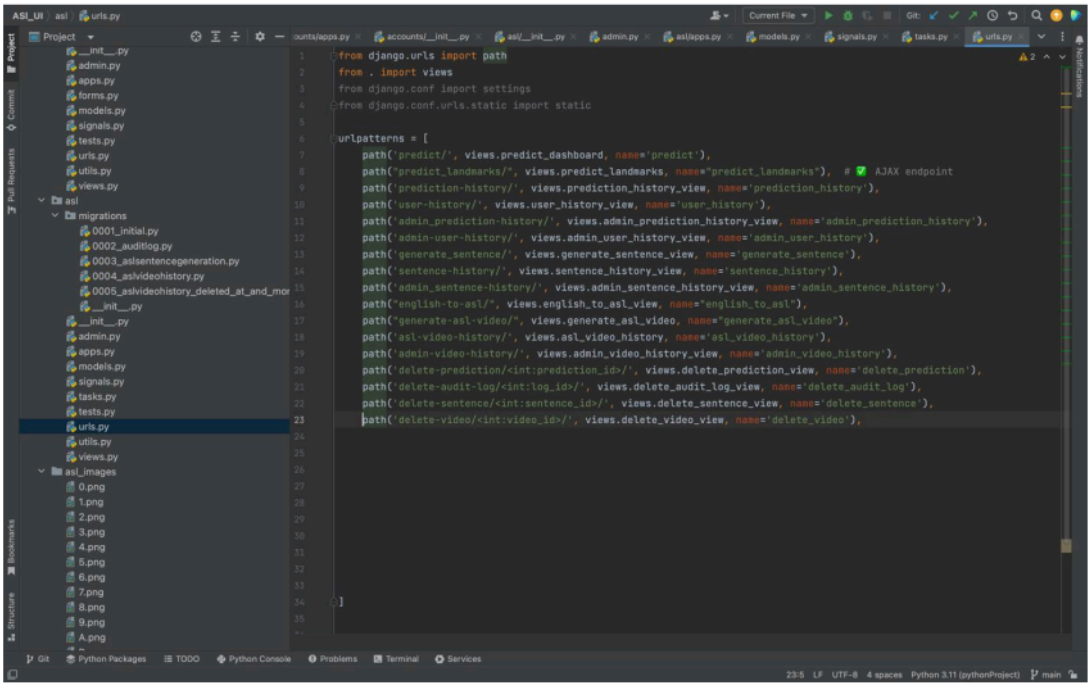
```python
from django.db import models
from django.contrib.auth.models import User
import os


class ASLPrediction(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    input_sequence = models.JSONField()  # stores the raw input landmark sequence
    top1_label = models.CharField(max_length=20)
    top1_confidence = models.FloatField()
    top2_label = models.CharField(max_length=20)
    top2_confidence = models.FloatField()
    top3_label = models.CharField(max_length=20)
    top3_confidence = models.FloatField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.user.username} - {self.top1_label} ({self.top1_confidence})"


class AuditLog(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    action = models.CharField(max_length=255)  # e.g., 'Prediction', 'Login', 'Logout'
    description = models.TextField()           # e.g., 'Predicted sign A with 0.98 confidence'
    ip_address = models.GenericIPAddressField(null=True, blank=True)
    user_agent = models.TextField(null=True, blank=True)
    timestamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.user} - {self.action} at {self.timestamp}"
```



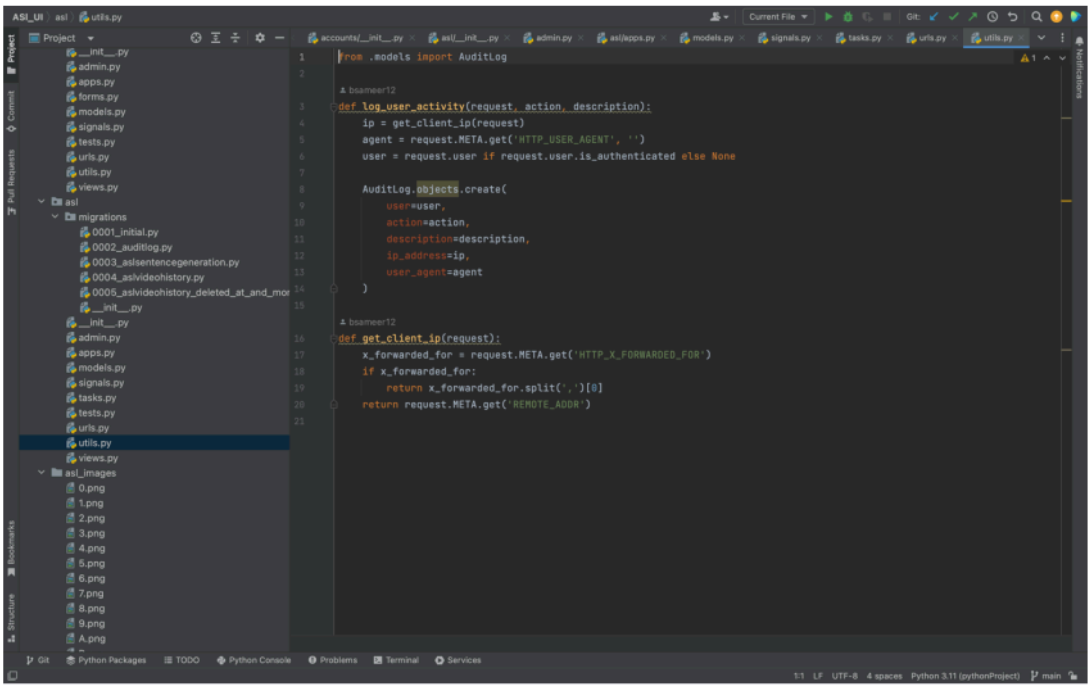
```python
from django.contrib.auth.signals import user_logged_in, user_logged_out
from django.dispatch import receiver
from .utils import log_user_activity


@receiver(user_logged_in)
def on_user_login(sender, request, user, **kwargs):
    log_user_activity(request, "Login", "User logged in.")

@receiver(user_logged_out)
def on_user_logout(sender, request, user, **kwargs):
    log_user_activity(request, "Logout", "User logged out.")
```



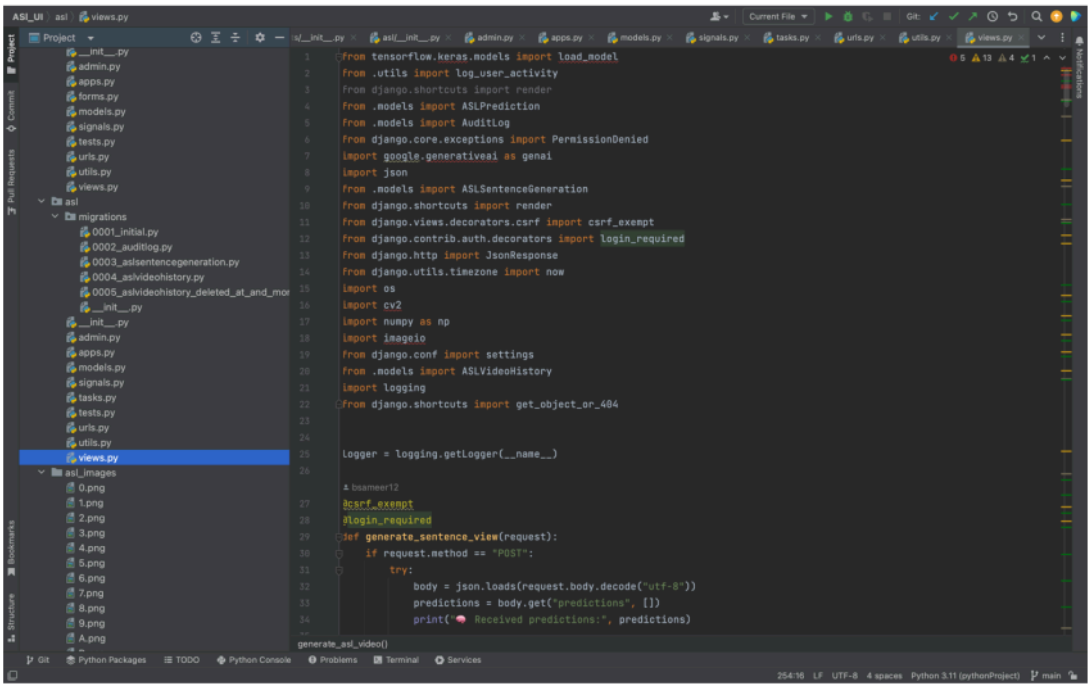
```
import threading
import time
import os
from datetime import timedelta
from django.utils.timezone import now
from asl.models import ASLVideoHistory

bsameer12
def delete_expired_videos():
    while True:
        threshold = now() - timedelta(minutes=1)
        expired_videos = ASLVideoHistory.objects.filter(created_at__lt=threshold, is_deleted=False)

        for video in expired_videos:
            file_path = video.absolute_file_path
            if os.path.exists(file_path):
                os.remove(file_path)

            video.is_deleted = True
            video.deleted_at = now()
            video.save()

        time.sleep(300)  # Sleep 5 minutes before next run
```



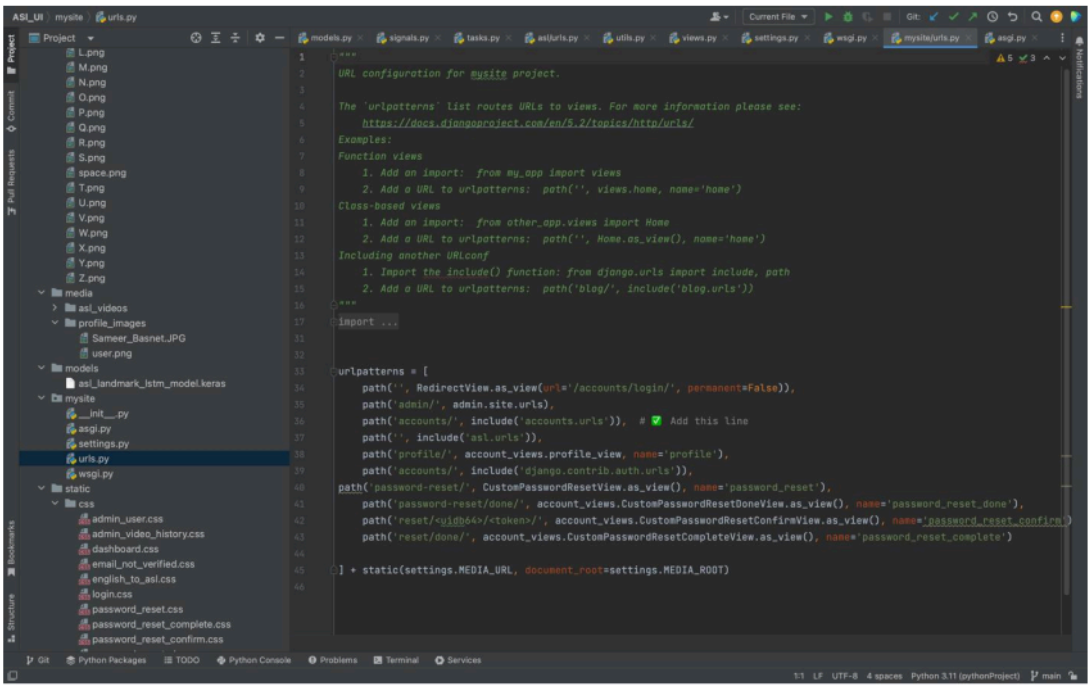
```
from django.urls import path
from . import views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('predict/', views.predict_dashboard, name='predict'),
    path("predict_landmarks/", views.predict_landmarks, name="predict_landmarks"),  # ✅ AJAX endpoint
    path('prediction-history/', views.prediction_history_view, name='prediction_history'),
    path('user-history/', views.user_history_view, name='user_history'),
    path('admin_prediction-history/', views.admin_prediction_history_view, name='admin_prediction_history'),
    path('admin-user-history/', views.admin_user_history_view, name='admin_user_history'),
    path('generate_sentence/', views.generate_sentence_view, name='generate_sentence'),
    path('sentence-history/', views.sentence_history_view, name='sentence_history'),
    path('admin_sentence-history/', views.admin_sentence_history_view, name='admin_sentence_history'),
    path("english-to-asl/", views.english_to_asl_view, name="english_to_asl"),
    path("generate-asl-video/", views.generate_asl_video, name="generate_asl_video"),
    path('asl-video-history/', views.asl_video_history, name='asl_video_history'),
    path('admin-video-history/', views.admin_video_history_view, name='admin_video_history'),
    path('delete-prediction/<int:prediction_id>/', views.delete_prediction_view, name='delete_prediction'),
    path('delete-audit-log/<int:log_id>/', views.delete_audit_log_view, name='delete_audit_log'),
    path('delete-sentence/<int:sentence_id>/', views.delete_sentence_view, name='delete_sentence'),
    path('delete-video/<int:video_id>/', views.delete_video_view, name='delete_video'),
]
```

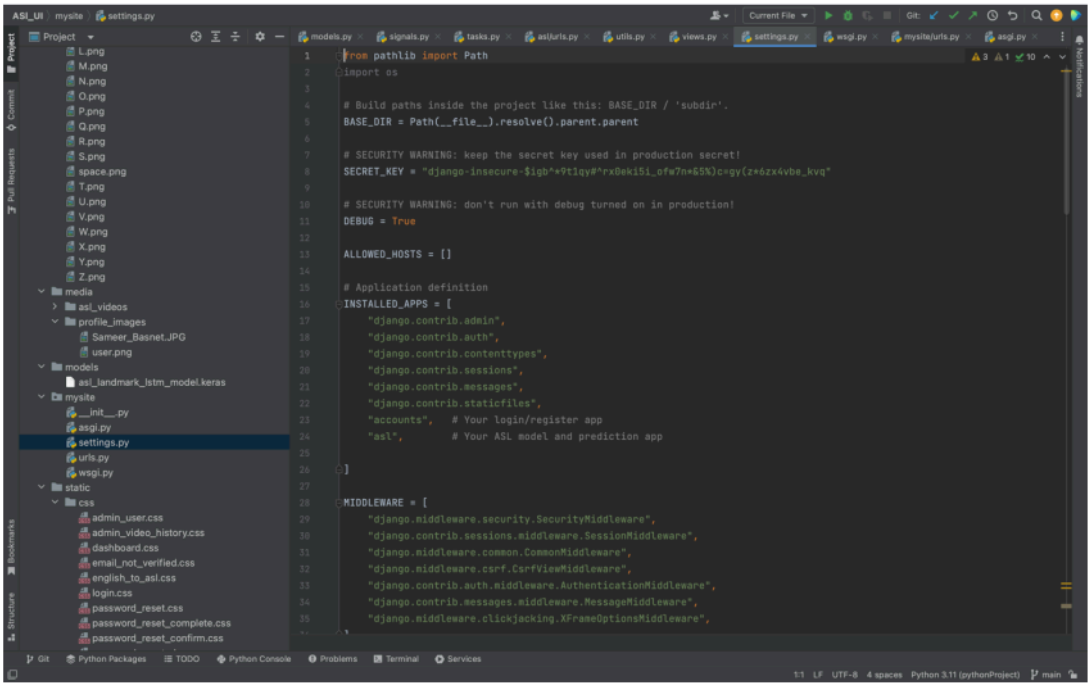


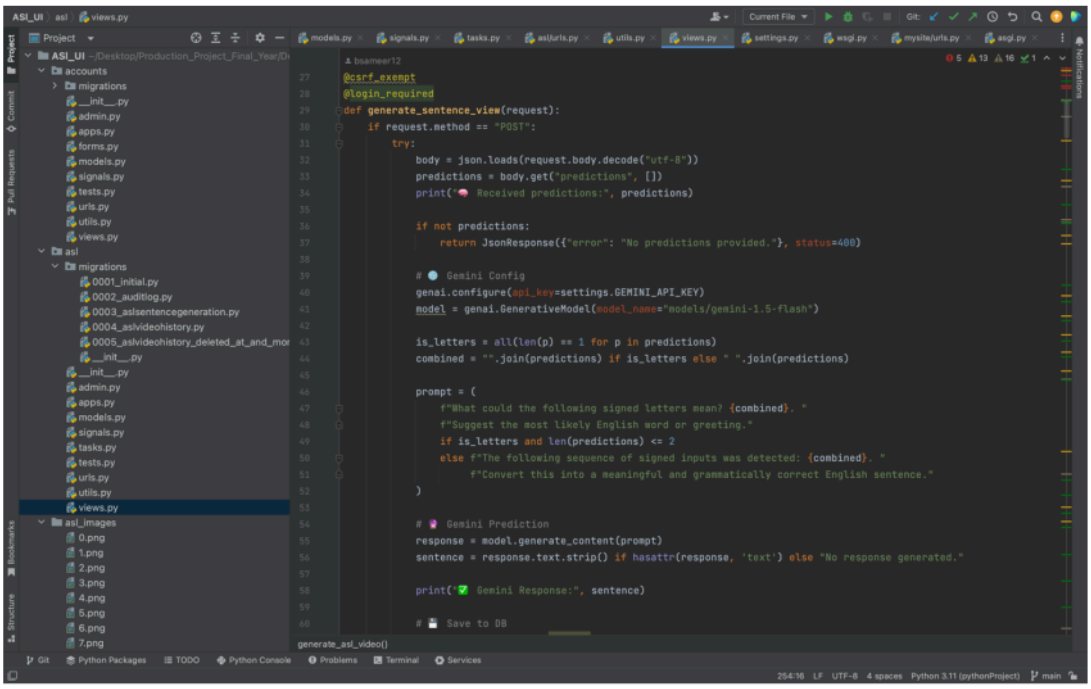
```python
from .models import AuditLog

bsameer12
def log_user_activity(request, action, description):
    ip = get_client_ip(request)
    agent = request.META.get('HTTP_USER_AGENT', '')
    user = request.user if request.user.is_authenticated else None

    AuditLog.objects.create(
        user=user,
        action=action,
        description=description,
        ip_address=ip,
        user_agent=agent
    )

bsameer12
def get_client_ip(request):
    x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
    if x_forwarded_for:
        return x_forwarded_for.split(',')[0]
    return request.META.get('REMOTE_ADDR')
```



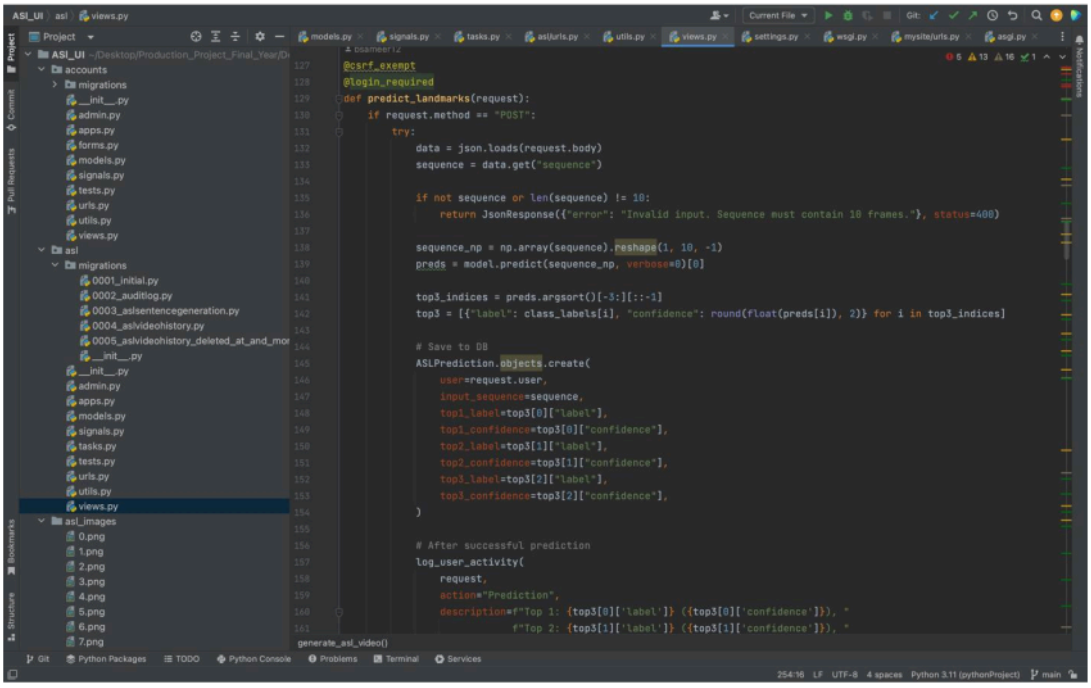
```python
from tensorflow.keras.models import load_model
from .utils import log_user_activity
from django.shortcuts import render
from .models import ASLPrediction
from .models import AuditLog
from django.core.exceptions import PermissionDenied
import google.generativeai as genai
import json
from .models import ASLSentenceGeneration
from django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from django.utils.timezone import now
import os
import cv2
import numpy as np
import imageio
from django.conf import settings
from .models import ASLVideoHistory
import logging
from django.shortcuts import get_object_or_404


logger = logging.getLogger(__name__)

bsameer12
@csrf_exempt
@login_required
def generate_sentence_view(request):
    if request.method == "POST":
        try:
            body = json.loads(request.body.decode("utf-8"))
            predictions = body.get("predictions", [])
            print("🧠 Received predictions:", predictions)
```


ASL_UI mysite urls.py
Project
L.png
M.png
N.png
O.png
P.png
Q.png
R.png
S.png
space.png
T.png
U.png
V.png
W.png
X.png
Y.png
Z.png
media
asl_videos
profile_images
Sameer_Basnet.JPG
user.png
models
asl_landmark_lstm_model.keras
mysite
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
static
css
admin_user.css
admin_video_history.css
dashboard.css
email_not_verified.css
english_to_asl.css
login.css
password_reset.css
password_reset_complete.css
password_reset_confirm.css
models.py signals.py tasks.py asl/urls.py utils.py views.py settings.py wsgi.py mysite/urls.py asgi.py
"""
URL configuration for mysite project.
The `urlpatterns` list routes URLs to views. For more information please see:
https://docs.djangoproject.com/en/5.2/topics/http/urls/
Examples:
Function views
1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path('', views.home, name='home')
Class-based views
1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
import ...
urlpatterns = [
path('', RedirectView.as_view(url='/accounts/login/', permanent=False)),
path('admin/', admin.site.urls),
path('accounts/', include('accounts.urls')), # Add this line
path('', include('asl.urls')),
path('profile/', account_views.profile_view, name='profile'),
path('accounts/', include('django.contrib.auth.urls')),
path('password-reset/', CustomPasswordResetView.as_view(), name='password_reset'),
path('password-reset/done/', account_views.CustomPasswordResetDoneView.as_view(), name='password_reset_done'),
path('reset/<uidb64>/<token>/', account_views.CustomPasswordResetConfirmView.as_view(), name='password_reset_confirm'),
path('reset/done/', account_views.CustomPasswordResetCompleteView.as_view(), name='password_reset_complete')
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
Git Python Packages TODO Python Console Problems Terminal Services
1:1 LF UTF-8 4 spaces Python 3.11 (pythonProject) main

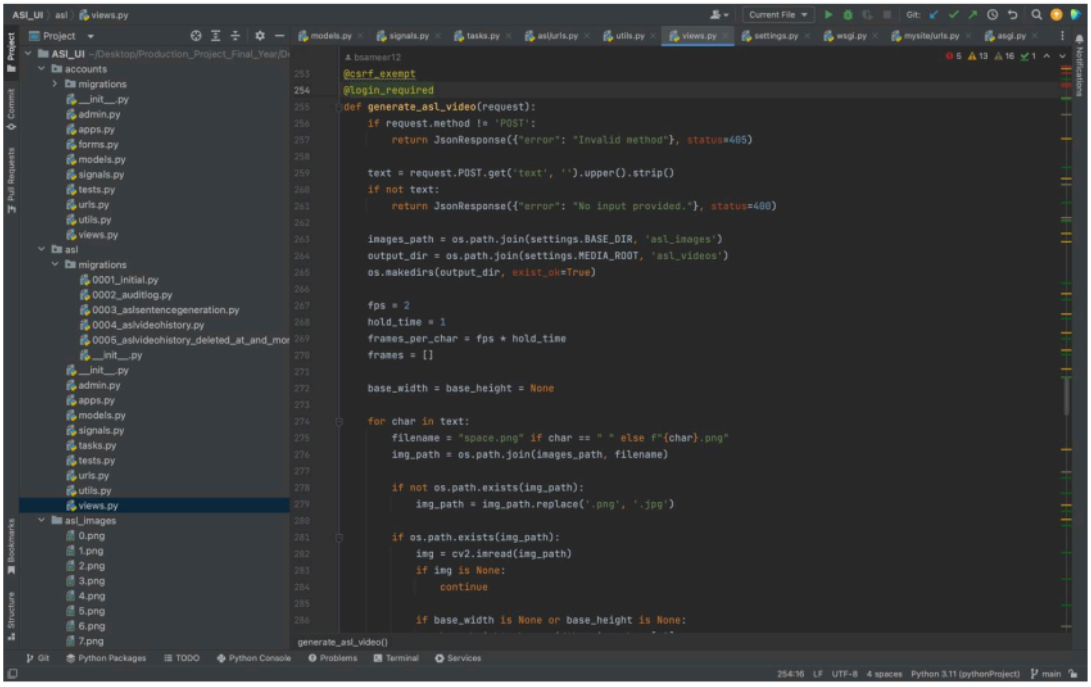

ASL_UI mysite settings.py
Project
L.png
M.png
N.png
O.png
P.png
Q.png
R.png
S.png
space.png
T.png
U.png
V.png
W.png
X.png
Y.png
Z.png
media
asl_videos
profile_images
Sameer_Basnet.JPG
user.png
models
asl_landmark_lstm_model.keras
mysite
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
static
css
admin_user.css
admin_video_history.css
dashboard.css
email_not_verified.css
english_to_asl.css
login.css
password_reset.css
password_reset_complete.css
password_reset_confirm.css
models.py signals.py tasks.py asl/urls.py utils.py views.py settings.py wsgi.py mysite/urls.py asgi.py
from pathlib import Path
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-$igb^*9t1qy#^rx0eki5i_ofw7n*65%)c=gy(z*6zx4vbe_kvq"
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
"django.contrib.admin",
"django.contrib.auth",
"django.contrib.contenttypes",
"django.contrib.sessions",
"django.contrib.messages",
"django.contrib.staticfiles",
"accounts", # Your login/register app
"asl", # Your ASL model and prediction app
]
MIDDLEWARE = [
"django.middleware.security.SecurityMiddleware",
"django.contrib.sessions.middleware.SessionMiddleware",
"django.middleware.common.CommonMiddleware",
"django.middleware.csrf.CsrfViewMiddleware",
"django.contrib.auth.middleware.AuthenticationMiddleware",
"django.contrib.messages.middleware.MessageMiddleware",
"django.middleware.clickjacking.XFrameOptionsMiddleware",
Git Python Packages TODO Python Console Problems Terminal Services
1:1 LF UTF-8 4 spaces Python 3.11 (pythonProject) main



```python
@csrf_exempt
@login_required
def generate_sentence_view(request):
    if request.method == "POST":
        try:
            body = json.loads(request.body.decode("utf-8"))
            predictions = body.get("predictions", [])
            print("📥 Received predictions:", predictions)

            if not predictions:
                return JsonResponse({"error": "No predictions provided."}, status=400)

            # 🔵 Gemini Config
            genai.configure(api_key=settings.GEMINI_API_KEY)
            model = genai.GenerativeModel(model_name="models/gemini-1.5-flash")

            is_letters = all(len(p) == 1 for p in predictions)
            combined = "".join(predictions) if is_letters else " ".join(predictions)

            prompt = (
                f"What could the following signed letters mean? {combined}. "
                f"Suggest the most likely English word or greeting."
                if is_letters and len(predictions) <= 2
                else f"The following sequence of signed inputs was detected: {combined}. "
                     f"Convert this into a meaningful and grammatically correct English sentence."
            )

            # 🔮 Gemini Prediction
            response = model.generate_content(prompt)
            sentence = response.text.strip() if hasattr(response, 'text') else "No response generated."

            print("✅ Gemini Response:", sentence)

            # 💾 Save to DB
```



```python
@csrf_exempt
@login_required
def predict_landmarks(request):
    if request.method == "POST":
        try:
            data = json.loads(request.body)
            sequence = data.get("sequence")

            if not sequence or len(sequence) != 10:
                return JsonResponse({"error": "Invalid input. Sequence must contain 10 frames."}, status=400)

            sequence_np = np.array(sequence).reshape(1, 10, -1)
            preds = model.predict(sequence_np, verbose=0)[0]

            top3_indices = preds.argsort()[-3:][::-1]
            top3 = [{"label": class_labels[i], "confidence": round(float(preds[i]), 2)} for i in top3_indices]

            # Save to DB
            ASLPrediction.objects.create(
                user=request.user,
                input_sequence=sequence,
                top1_label=top3[0]["label"],
                top1_confidence=top3[0]["confidence"],
                top2_label=top3[1]["label"],
                top2_confidence=top3[1]["confidence"],
                top3_label=top3[2]["label"],
                top3_confidence=top3[2]["confidence"],
            )

            # After successful prediction
            log_user_activity(
                request,
                action="Prediction",
                description=f"Top 1: {top3[0]['label']} ({top3[0]['confidence']}), "
                            f"Top 2: {top3[1]['label']} ({top3[1]['confidence']}), "
```


@csrf_exempt
@login_required
def generate_asl_video(request):
if request.method != 'POST':
return JsonResponse({"error": "Invalid method"}, status=405)
text = request.POST.get('text', '').upper().strip()
if not text:
return JsonResponse({"error": "No input provided."}, status=400)
images_path = os.path.join(settings.BASE_DIR, 'asl_images')
output_dir = os.path.join(settings.MEDIA_ROOT, 'asl_videos')
os.makedirs(output_dir, exist_ok=True)
fps = 2
hold_time = 1
frames_per_char = fps * hold_time
frames = []
base_width = base_height = None
for char in text:
filename = "space.png" if char == " " else f"{char}.png"
img_path = os.path.join(images_path, filename)
if not os.path.exists(img_path):
img_path = img_path.replace('.png', '.jpg')
if os.path.exists(img_path):
img = cv2.imread(img_path)
if img is None:
continue
if base_width is None or base_height is None:

11.4 Meeting Sheets Screenshots

School of Computing, Creative Technologies and Engineering 2024/25

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number: 1**
Student: Sameer Basnet	**Student I.D.: 77356702**
Date of Meeting:03-02-2025	**Supervisor: Mr. Suramya Sharma Dahal**

Actions agreed at previous meeting (completed or comment):

1		☐
2		☐
3		☐
4		☐
5		☐
6		☐

Comments of student (if any):

...

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**: 04-06-2025**

Agreed Actions to complete before next meeting:

1	Do research on how to implement English language to ASL conversion.
2	Try To Implement English language to ASL conversion.
3	Optimize the model for numeric ASL signs too.
4	
5	
6	

Comments of supervisor (if any):

School of Computing, Creative Technologies and Engineering 2024/25

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number: 2**
Student: Sameer Basnet	**Student I.D.: 77356702**
Date of Meeting:04-06-2025	**Supervisor: Mr. Suramya Sharma Dahal**

Actions agreed at previous meeting (completed or comment):

1	Optimize the tts and prediction ui and fix freezing issue and crash issue ☐
2	Try To Implement English language to ASL conversion. ☐
3	Optimize the model for numeric ASL signs too. ☐
4	☐
5	☐
6	☐

Comments of student (if any):

.....
.....
.....
.....
.....
.....
.....
.....
.....

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**: 04-13-2025**

Agreed Actions to complete before next meeting:

1	Optimize the tts and prediction ui and fix freezing issue and crash issue
2	Try to implement Gemini api for generating meaningful sentences from predictions.
3	Try to implement more simplified and user friendly ui for the project
4	

5	
6	
Comments of supervisor (if any):	

School of Computing, Creative Technologies and Engineering 2024/25

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number: 3**
Student: Sameer Basnet	**Student I.D.: 77356702**
Date of Meeting:04-13-2025	**Supervisor: Mr. Suramya Sharma Dahal**

Actions agreed at previous meeting (completed or comment):

1	Optimize the tts and prediction ui and fix freezing issue and crash issue	☐
2	Try to implement Gemini api for generating meaningful sentences from predictions.	☐
3	Try to implement more simplified and user friendly ui for the project	☐
4		☐
5		☐
6		☐

Comments of student (if any):

Due to an unusually heavy workload over the past week, I wasn't able to complete the specific task by the original deadline. I apologize for any inconvenience this delay may have caused.

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:04-25-2025**

Agreed Actions to complete before next meeting:

1	Complete the previous week agreed action up to Wednesday.
2	Try to complete the project development up to next week.
3	Start working on project report too.
4	
5	
6	

Comments of supervisor (if any):

School of Computing, Creative Technologies and Engineering 2024/25

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number: 4**
Student: Sameer Basnet	**Student I.D.: 77356702**
Date of Meeting:04-25-2025	**Supervisor: Mr. Suramya Sharma Dahal**

Actions agreed at previous meeting (completed or comment):

1	Complete the previous week agreed action up to Wednesday.	☐
2	Try to complete the project development up to next week.	☐
3	Start working on project report too.	☐
4		☐
5		☐
6		☐

Comments of student (if any):

Completed all tasks as agreed.

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:05-06-2025**

Agreed Actions to complete before next meeting:

1	Finish final draft of the report.
2	
3	
4	
5	
6	

School of Computing, Creative Technologies and Engineering 2024/25

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number: 5**
Student: Sameer Basnet	**Student I.D.: 77356702**
Date of Meeting:05-06-2025	**Supervisor: Mr. Suramya Sharma Dahal**

Actions agreed at previous meeting (completed or comment):

1	Finish final draft of the report.	☐
2		☐
3		☐
4		☐
5		☐
6		☐

Comments of student (if any):

Completed all tasks as agreed.

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:**

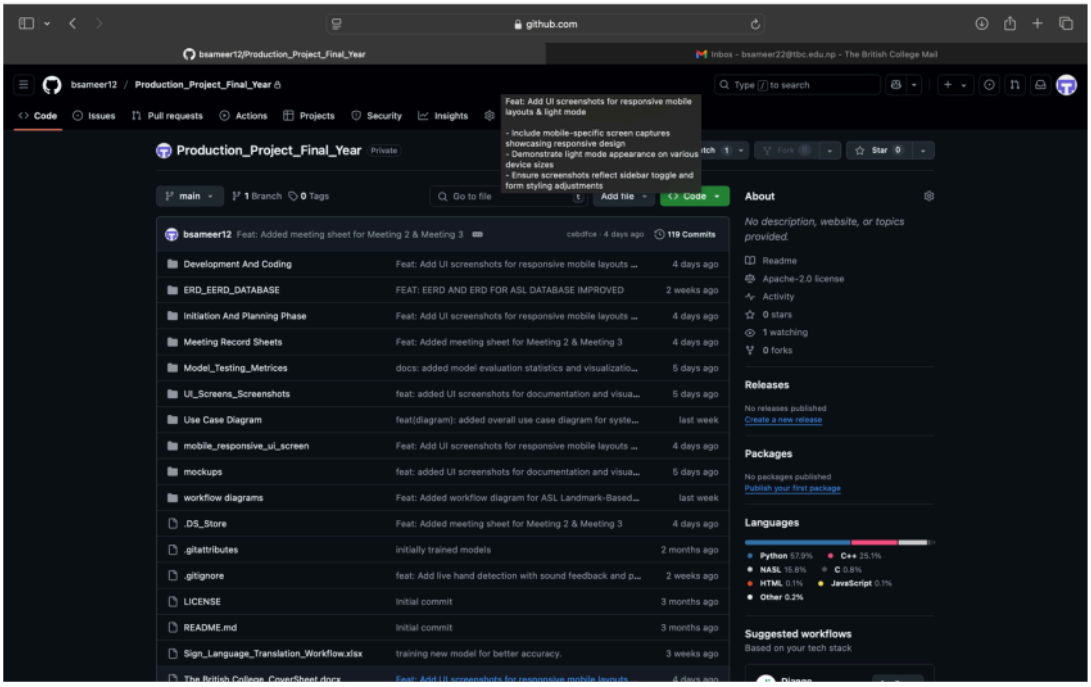
Agreed Actions to complete before next meeting:

1	Finish presentation and submit all required fill on 11TH May.
2	
3	
4	
5	
6	

Comments of supervisor (if any):

11.5 Github Screenshots

GITHUB Link: *https://github.com/bsameer12/Production_Project_Final_Year*



11.6 Ethical Consent Form

STAGE 1 - RESEARCH ETHICS APPROVAL FORM (from December 2016)

LEEDS BECKETT UNIVERSITY

STAGE 1 - RESEARCH ETHICS APPROVAL FORM

Research by students and staff at the University must receive ethical approval before any data collection commences. Applications may be made on the Research Ethics Online system or via approval forms.

If using the approval forms, applicants complete this Stage 1 - Research Ethics Approval Form which includes the Risk Checklist.

For student projects classified as Risk Category 1 (e.g., many literature reviews), these can be approved on this Stage 1 – Research Ethics Approval Form by the Research Supervisor.

Applicants whose research studies are classified as Risk Category 2 or 3 must also complete and submit the separate Stage 2 - Research Ethics Approval Form.

Guidance for completion of this form and the application process is provided on pages 3 and 4.

APPLICANT DETAILS	
Your name (if a group project, include all names)	Sameer Basnet
School	
STATUS	
• Undergraduate student	✓
• Taught Postgraduate student	☐
• Research Postgraduate student	☐
• Staff member	☐
• Other (give details)	
IF THIS IS A STUDENT PROJECT	
• Student ID	77356702
• Course title (eg, BA (Hons) History)	BSc. (Hons) Computing
• Student email	Bsameer22@tbc.edu.np
• Research Supervisor's name Or Director of Studies' name	Mr. Suramya Sharma Dahal
THE PROJECT/STUDY	
Project /study title	Breaking the Slience: AI and Computer Vision Driven Sign Language Translation System
Start date of project	12/16/2024
Expected completion date of project	05/11/2025
Project summary – please give a brief summary of your study (maximum 100 words)	
This project plans to develop an AI-mechanized ASL translation system using computer vision to benefit deaf and mute persons communicate in English through text and speech. The system will observe and perceive hand gestures in real time, linking them into English text and assimilating a Text-to-Speech (TTS) module for embellished accessibility. It will factor a user-friendly Gui and real-time error handling to improve usability. Adopting deep learning models and gesture recognition frameworks, the project pursues to build an accurate, competent, and sturdy solution for ASL interpretation.	
CONFIRMATION STATEMENTS	
The results of research should benefit society directly or by generally improving knowledge and understanding. Please tick this box to confirm that your research study has a potential benefit. *If you cannot identify a benefit you must discuss your project with your Research Supervisor to help identify one or adapt your proposal so the study will have an identifiable benefit.*	✓
Please tick this box to confirm you have read the Research Ethics Policy and the relevant sections of the Research Ethics Procedures and will adhere to these in the conduct of this project.	✓

RISK CHECKLIST - Please answer ALL the questions in each of the sections below – tick YES or NO WILL YOUR RESEARCH STUDY.....?		YES	NO
1	Involve direct and/or indirect contact with human participants?	☐	✓
2	Involve analysis of pre-existing data which contains personal or sensitive information not in the public domain?	☐	✓
3	Require permission or consent to conduct?	☐	✓
4	Require permission or consent to publish?	☐	✓
5	Have a risk of compromising confidentiality?	☐	✓
6	Have a risk of compromising anonymity?	☐	✓
7	Collect / contain sensitive personal data?	☐	✓
8	Contain elements which you OR your supervisor are NOT trained to conduct?	☐	✓
9	Use any information OTHER than that which is freely available in the public domain?	☐	✓
10	Involve respondents to the internet or other visual/vocal methods where participants may be identified?	☐	✓
11	Include a financial incentive to participate in the research?	☐	✓
12	Involve your own students, colleagues or employees?	☐	✓
13	Take place outside of the country where you are enrolled as a student, or for staff, outside of the UK?	☐	✓
14	Involve participants who are particularly vulnerable or at risk?	☐	✓
15	Involve any participants who are unable to give informed consent?	☐	✓
16	Involve data collection taking place BEFORE informed consent is given?	☐	✓
17	Involve any deliberate deception or covert data collection?	☐	✓
18	Involve a risk to the researcher or participants beyond that experienced in everyday life?	☐	✓
19	Cause (or could cause) physical or psychological harm or negative consequences?	☐	✓
20	Use intrusive or invasive procedures?	☐	✓
21	Involve a clinical trial?	☐	✓
22	Involve the possibility of incidental findings related to health status?	☐	✓
23	Fit into any of the following security-sensitive categories: concerns terrorist or extreme groups; commissioned by the military; commissioned under an EU security call; involves the acquisition of security clearances? If yes, see the guidance.	☐	✓

CLASSIFICATION The following guidance will help classify the risk level of your study	Tick the box which applies to your project
If you answered **NO** to all the above questions, your study is provisionally classified as **Risk Category 1** (literature reviews will be Risk Category 1).	✓
If you answered **YES** to any question from 1-13 and **NO** to all questions 14-22, your study is provisionally classified as **Risk Category 2**.	☐
If you answered **YES** to any question from 14-22, your study is provisionally classified as **Risk Category 3**.	☐
If question 23 has been answered **YES**, your application will be reviewed by the Chair of the University Research Ethics Sub-committee	☐

DECLARATION AND SIGNATURE/S			
I confirm that I will undertake this project as detailed above. I understand that I must abide by the terms of the approval and that I may not make any substantial amendments to the project without further approval.			
Signed	Sameer Basnet	*Date*	03/02/2025

FOR RISK CATEGORY 1 STUDENT PROJECTS					
Approval from the Research Supervisor or Director of Studies for a student project: *I have discussed the ethical issues arising from the project with the student. I approve this project.*					
Name	Suramya Sharma Dahal	*Signed*		*Date*	03/02/2025

NEXT STEP

RISK CATEGORY 1 PROJECTS: IF YOUR PROJECT HAS BEEN CLASSIFIED AS RISK CATEGORY 1:

- Students: The Research Supervisor should return the signed form to the student and send a copy to the Local Research Ethics Co-ordinator and where relevant, the Research Module Leader, for information.
- Staff: Submit this form to your Local Research Ethics Co-ordinator.

RISK CATEGORY 2 OR 3 PROJECTS: IF YOUR PROJECT HAS BEEN CLASSIFIED AS RISK CATEGORY 2 OR 3 please complete the Stage 2 - Research Ethics Approval form and submit both forms together with supporting documentation.

QUESTION 23: If this question has been answered **YES**, your application will be reviewed by the Chair of the University Research Ethics Sub-committee, and the forms should be submitted directly to Professor Karl Spracklen, k.spracklen@leedsbeckett.ac.uk. You will need to submit the Security-sensitive research form available from the Research Ethics web page.

Research ethics application forms will be retained in the School for the purposes of quality assurance of compliance and audit for THREE years

NOTES FOR COMPLETION

University Research Ethics Policy and Procedures: The University Research Ethics Policy and Research Ethics Procedures should be read prior to commencing this application. Consideration of the application by the reviewer/s will be undertaken in accordance with the Policy and Procedures.

External requirements for the project: Applicants should consider if there are requirements by any relevant professional, statutory or regulatory body, or learned society, which may be relevant to the project or if the project also requires external approval.

Submission

- Student applicants: email the typed form/s to your Research Supervisor or Director of Studies.
- Staff applicants: email the typed form/s to your Local Research Ethics Co-ordinator.

How to complete the form

You can navigate through the form by using the tab keys. If you prefer to complete a normal Word document, you can unlock the form by selecting the 'Restrict Editing' button on the Developer tab, then click on 'Stop Protection'. The boxes should expand to allow space for your text.

Signatures
Electronic/typed signatures are acceptable for emailed forms, as the emails provide the audit trail for all parties' agreement and approval of the forms (e.g., student applicant →Research Supervisor →Local Research Ethics Co-ordinator).

Outcome
Applicants will be advised of the outcome of the application by receipt of the signed form fromm:

- The Research Supervisor or Director of Studies for Risk Category 1 student projects;
- The Local Research Ethics Co-ordinator or the School level group for Risk Category 2 and 3 projects.

YOU MAY ONLY BEGIN ANY DATA COLLECTION ONCE YOU RECEIVE NOTIFICATION THAT THE PROJECT HAS ETHICAL APPROVAL. If the circumstances of your research study change after approval it is your responsibility to revisit the Risk Checklist and complete a further application.

Advice
When completing the Stage 1 - Research Ethics Approval Form,. if you are uncertain about the answer to any question, read the relevant section of the Research Ethics Procedures document, and if you are still unsure:

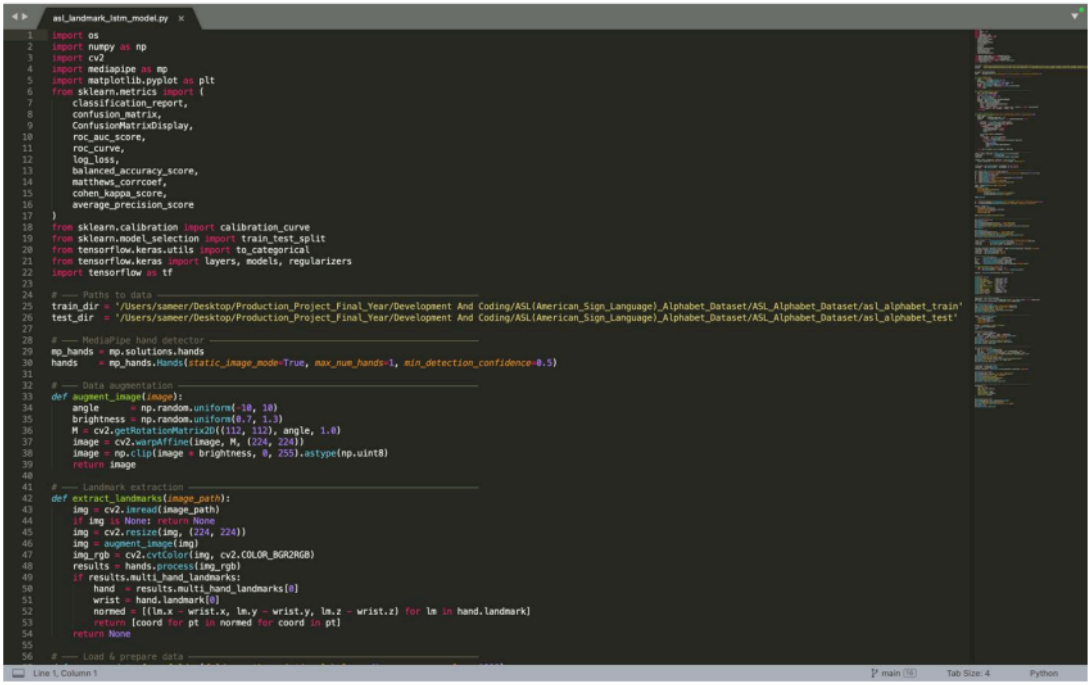
- if you are student, seek guidance from your Research Supervisor or Director of Studies;
- if you are a staff member, contact your Local Research Ethics Co-ordinator.

APPROVAL PROCESS

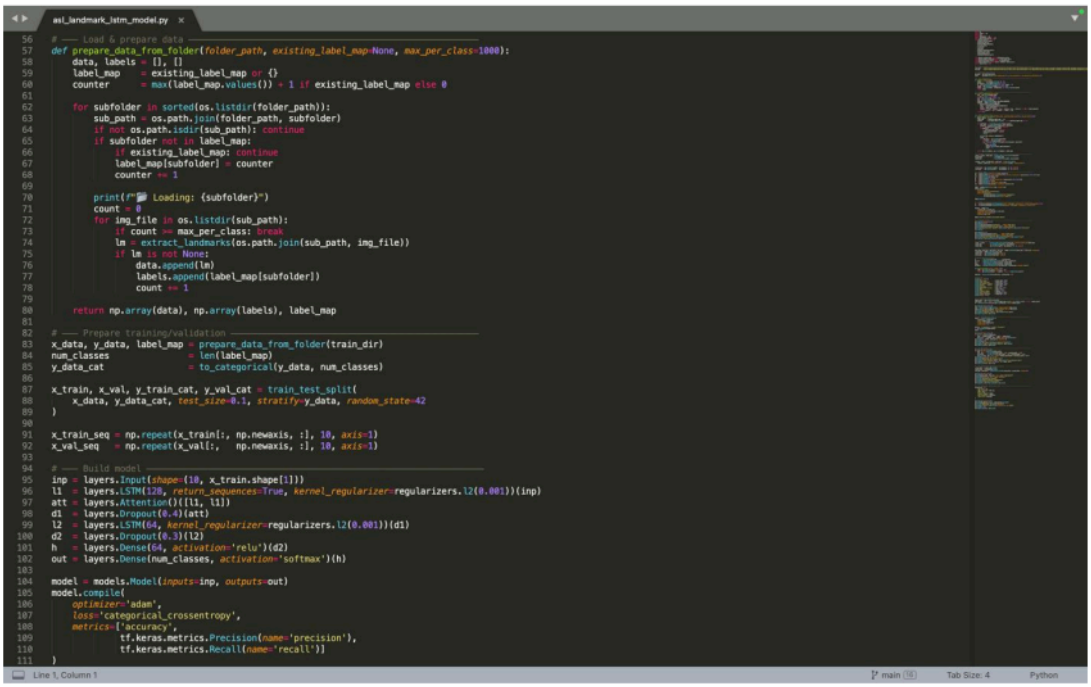
- Local Research Ethics Co-ordinator = LREC
- School level group (if your School uses a different review process, please follow your School guidance)
- University Research Ethics Sub-Committee = URESC

Category	Student applicants	Staff applicants
Risk Category 1	If your study has been provisionally classified as Risk Category 1, your Research Supervisor (or Director of Studies) can normally give approval for the project. You must complete this form and submit it to your Research Supervisor for consideration. A copy of the signed form if approved must be given or emailed to the LREC and, where relevant, the Research Module Leader, for information.	If your study has been classified as Risk Category 1, you do not need ethical approval for the project. You must complete the remainder of this form so that your research project is registered with the University. Please submit this form to your LREC.
Risk Category 2	If your study has been provisionally classified as Risk Category 2, your Supervisor (or Director of Studies) can recommend approval for your study by the LREC. You must complete this application form and also the separate Stage 2 - Research Ethics Approval form. Once you have completed the forms please submit both forms and supporting documentation to your Research Supervisor for consideration. Your Supervisor may disagree with your assessment and ask you to make revisions or reject your application. When the Research Supervisor is happy to recommend the application for approval, they will send the forms to the LREC. The LREC will review your project and then decide to approve it, ask for revisions, reject it or pass it on for review by the School level group.	If your study has been provisionally classified as Risk Category 2, your project will be considered for ethical approval by the LREC. You must complete this application form and also the separate Stage 2 - Research Ethics Approval form. Please submit both forms and supporting documentation to your LREC for consideration. The LREC will review your project and then decide to approve it, ask for revisions or pass it on for review by the School level group.
Risk Category 3	**Postgraduate Research Students** If your study has been provisionally classified as Risk Category 3, your Supervisor or Director of Studies can recommend approval for your study by the LREC. You must complete this application form and also the separate Stage 2 - Research Ethics Approval form and submit both forms to your Director of Studies. If your Director of Studies recommends approval of your project they will refer it to the LREC who will review your project and decide whether to grant ethical approval, request revisions, reject the application or refer it to the School level group for review. **Undergraduate and Taught Postgraduate Students** If your study has been provisionally classified as Risk Category 3, you should consult with your Research Supervisor immediately as it is unlikely you will be able to proceed and you should negotiate a project that is of lower risk. However, if you have already discussed the project with your Supervisor and they have agreed that a case for approval is warranted, proceed in line with the details above for Research Students.	If your study has been provisionally classified as Risk Category 3, your project will be considered for ethical approval by an appropriate LREC. You must complete this application form and also the separate Stage 2 - Research Ethics Approval form and submit both forms with supporting documentation to your LREC. The LREC will review your project and then decide to approve it, ask for revisions or pass it on for review by the School level group.
Q23	If question 23 has been answered 'yes', your application will be reviewed by the Chair of the University Research Ethics Sub-committee. The answer does not affect the Risk Category.	

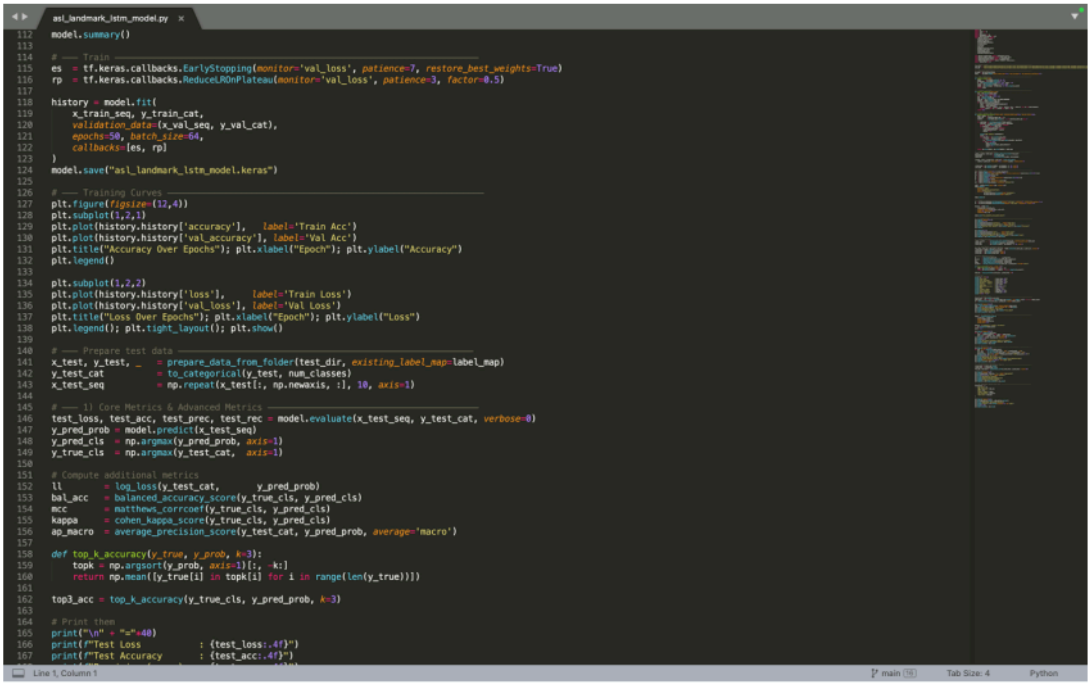
11.7 Model Training Code



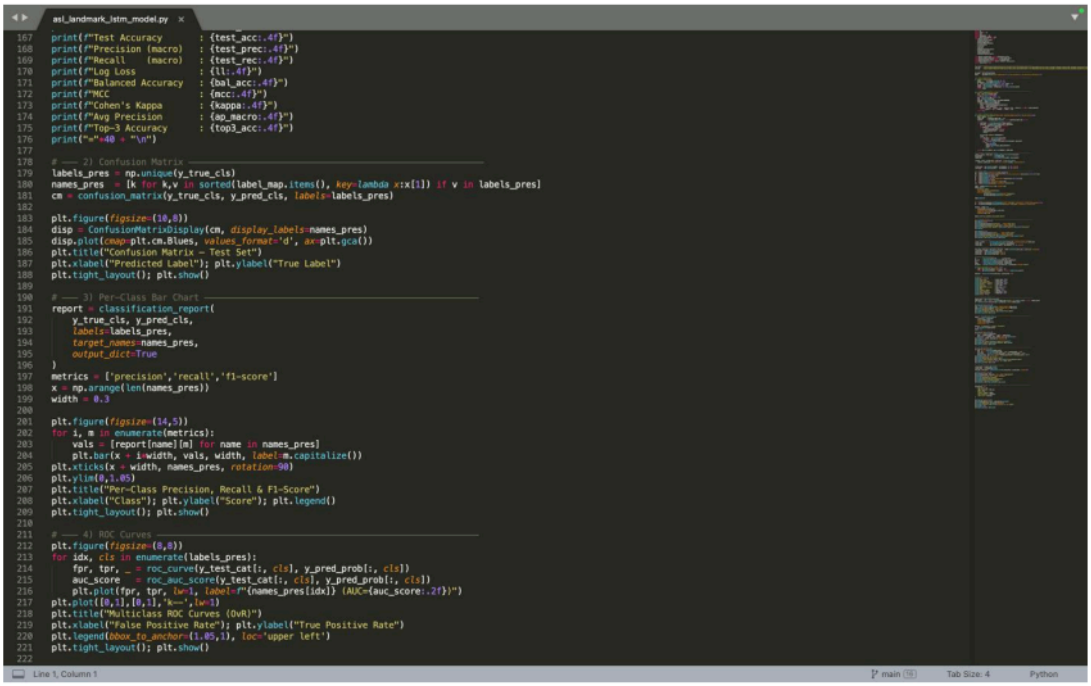
```python
import os
import numpy as np
import cv2
import mediapipe as mp
import matplotlib.pyplot as plt
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    ConfusionMatrixDisplay,
    roc_auc_score,
    roc_curve,
    log_loss,
    balanced_accuracy_score,
    matthews_corrcoef,
    cohen_kappa_score,
    average_precision_score
)
from sklearn.calibration import calibration_curve
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import layers, models, regularizers
import tensorflow as tf

# ── Paths to data ──────────────────────────────────────────────
train_dir = '/Users/sameer/Desktop/Production_Project_Final_Year/Development And Coding/ASL(American_Sign_Language)_Alphabet_Dataset/ASL_Alphabet_Dataset/asl_alphabet_train'
test_dir  = '/Users/sameer/Desktop/Production_Project_Final_Year/Development And Coding/ASL(American_Sign_Language)_Alphabet_Dataset/ASL_Alphabet_Dataset/asl_alphabet_test'

# ── MediaPipe hand detector ────────────────────────────────────
mp_hands = mp.solutions.hands
hands    = mp_hands.Hands(static_image_mode=True, max_num_hands=1, min_detection_confidence=0.5)

# ── Data augmentation ──────────────────────────────────────────
def augment_image(image):
    angle      = np.random.uniform(-10, 10)
    brightness = np.random.uniform(0.7, 1.3)
    M = cv2.getRotationMatrix2D((112, 112), angle, 1.0)
    image = cv2.warpAffine(image, M, (224, 224))
    image = np.clip(image * brightness, 0, 255).astype(np.uint8)
    return image

# ── Landmark extraction ────────────────────────────────────────
def extract_landmarks(image_path):
    img = cv2.imread(image_path)
    if img is None: return None
    img = cv2.resize(img, (224, 224))
    img = augment_image(img)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(img_rgb)
    if results.multi_hand_landmarks:
        hand = results.multi_hand_landmarks[0]
        wrist = hand.landmark[0]
        normed = [(lm.x - wrist.x, lm.y - wrist.y, lm.z - wrist.z) for lm in hand.landmark]
        return [coord for pt in normed for coord in pt]
    return None

# ── Load & prepare data ────────────────────────────────────────
```



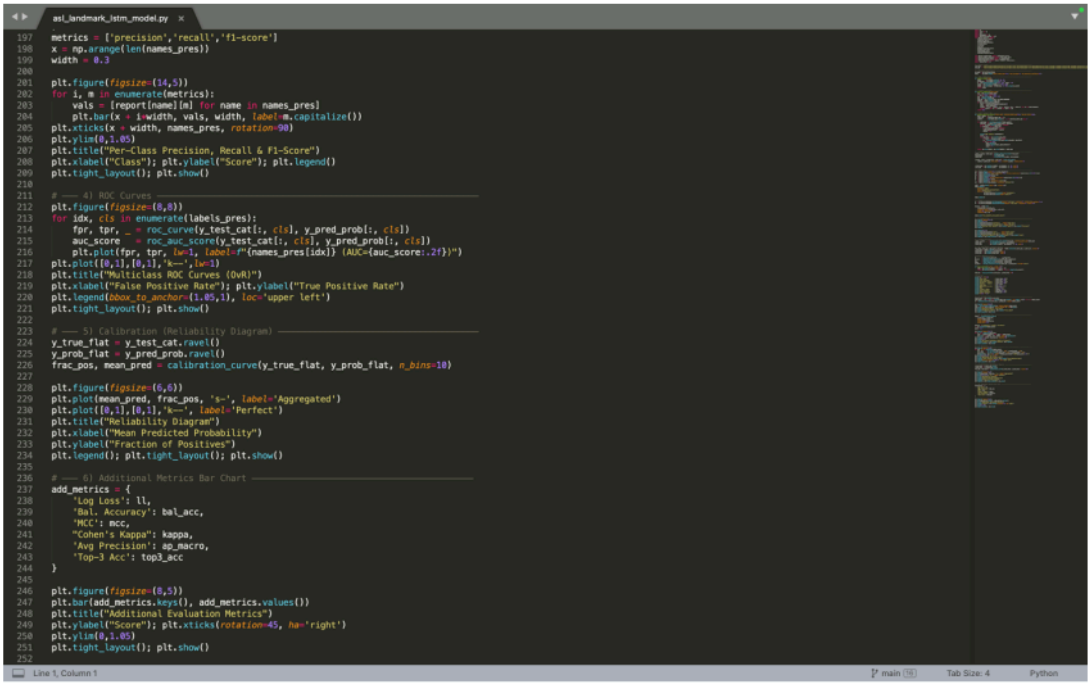
```python
# ── Load & prepare data ────────────────────────────────────────
def prepare_data_from_folder(folder_path, existing_label_map=None, max_per_class=1000):
    data, labels = [], []
    label_map    = existing_label_map or {}
    counter      = max(label_map.values()) + 1 if existing_label_map else 0

    for subfolder in sorted(os.listdir(folder_path)):
        sub_path = os.path.join(folder_path, subfolder)
        if not os.path.isdir(sub_path): continue
        if subfolder not in label_map:
            if existing_label_map: continue
            label_map[subfolder] = counter
            counter += 1

        print(f"📂 Loading: {subfolder}")
        count = 0
        for img_file in os.listdir(sub_path):
            if count >= max_per_class: break
            lm = extract_landmarks(os.path.join(sub_path, img_file))
            if lm is not None:
                data.append(lm)
                labels.append(label_map[subfolder])
                count += 1

    return np.array(data), np.array(labels), label_map

# ── Prepare training/validation ────────────────────────────────
X_data, y_data, label_map = prepare_data_from_folder(train_dir)
num_classes               = len(label_map)
y_data_cat                = to_categorical(y_data, num_classes)

X_train, X_val, y_train_cat, y_val_cat = train_test_split(
    X_data, y_data_cat, test_size=0.1, stratify=y_data, random_state=42
)

X_train_seq = np.repeat(X_train[:, np.newaxis, :], 10, axis=1)
X_val_seq   = np.repeat(X_val[:,   np.newaxis, :], 10, axis=1)

# ── Build model ────────────────────────────────────────────────
inp = layers.Input(shape=(10, X_train.shape[1]))
l1  = layers.LSTM(128, return_sequences=True, kernel_regularizer=regularizers.l2(0.001))(inp)
att = layers.Attention()([l1, l1])
d1  = layers.Dropout(0.4)(att)
l2  = layers.LSTM(64, kernel_regularizer=regularizers.l2(0.001))(d1)
d2  = layers.Dropout(0.3)(l2)
h   = layers.Dense(64, activation='relu')(d2)
out = layers.Dense(num_classes, activation='softmax')(h)

model = models.Model(inputs=inp, outputs=out)
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy',
             tf.keras.metrics.Precision(name='precision'),
             tf.keras.metrics.Recall(name='recall')]
)
```



```
model.summary()

# ── Train ──────────────────────────────────────────────────────
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=7, restore_best_weights=True)
rp = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', patience=3, factor=0.5)

history = model.fit(
    x_train_seq, y_train_cat,
    validation_data=(x_val_seq, y_val_cat),
    epochs=50, batch_size=64,
    callbacks=[es, rp]
)
model.save("asl_landmark_lstm_model.keras")

# ── Training Curves ────────────────────────────────────────────
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'],    label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title("Accuracy Over Epochs"); plt.xlabel("Epoch"); plt.ylabel("Accuracy")
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'],     label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title("Loss Over Epochs"); plt.xlabel("Epoch"); plt.ylabel("Loss")
plt.legend(); plt.tight_layout(); plt.show()

# ── Prepare test data ──────────────────────────────────────────
x_test, y_test, _   = prepare_data_from_folder(test_dir, existing_label_map=label_map)
y_test_cat          = to_categorical(y_test, num_classes)
x_test_seq          = np.repeat(x_test[:, np.newaxis, :], 10, axis=1)

# ── 1) Core Metrics & Advanced Metrics ─────────────────────────
test_loss, test_acc, test_prec, test_rec = model.evaluate(x_test_seq, y_test_cat, verbose=0)
y_pred_prob = model.predict(x_test_seq)
y_pred_cls  = np.argmax(y_pred_prob, axis=1)
y_true_cls  = np.argmax(y_test_cat,  axis=1)

# Compute additional metrics
ll       = log_loss(y_test_cat,       y_pred_prob)
bal_acc  = balanced_accuracy_score(y_true_cls, y_pred_cls)
mcc      = matthews_corrcoef(y_true_cls, y_pred_cls)
kappa    = cohen_kappa_score(y_true_cls, y_pred_cls)
ap_macro = average_precision_score(y_test_cat, y_pred_prob, average='macro')

def top_k_accuracy(y_true, y_prob, k=3):
    topk = np.argsort(y_prob, axis=1)[:, -k:]
    return np.mean([y_true[i] in topk[i] for i in range(len(y_true))])

top3_acc = top_k_accuracy(y_true_cls, y_pred_prob, k=3)

# Print them
print("\n" + "="*40)
print(f"Test Loss           : {test_loss:.4f}")
print(f"Test Accuracy       : {test_acc:.4f}")
```



```
print(f"Test Accuracy       : {test_acc:.4f}")
print(f"Precision (macro)   : {test_prec:.4f}")
print(f"Recall    (macro)   : {test_rec:.4f}")
print(f"Log Loss            : {ll:.4f}")
print(f"Balanced Accuracy   : {bal_acc:.4f}")
print(f"MCC                 : {mcc:.4f}")
print(f"Cohen's Kappa       : {kappa:.4f}")
print(f"Avg Precision       : {ap_macro:.4f}")
print(f"Top-3 Accuracy      : {top3_acc:.4f}")
print("="*40 + "\n")

# ── 2) Confusion Matrix ────────────────────────────────────────
labels_pres = np.unique(y_true_cls)
names_pres  = [k for k,v in sorted(label_map.items(), key=lambda x:x[1]) if v in labels_pres]
cm = confusion_matrix(y_true_cls, y_pred_cls, labels=labels_pres)

plt.figure(figsize=(10,8))
disp = ConfusionMatrixDisplay(cm, display_labels=names_pres)
disp.plot(cmap=plt.cm.Blues, values_format='d', ax=plt.gca())
plt.title("Confusion Matrix – Test Set")
plt.xlabel("Predicted Label"); plt.ylabel("True Label")
plt.tight_layout(); plt.show()

# ── 3) Per-Class Bar Chart ─────────────────────────────────────
report = classification_report(
    y_true_cls, y_pred_cls,
    labels=labels_pres,
    target_names=names_pres,
    output_dict=True
)
metrics = ['precision','recall','f1-score']
x = np.arange(len(names_pres))
width = 0.3

plt.figure(figsize=(14,5))
for i, m in enumerate(metrics):
    vals = [report[name][m] for name in names_pres]
    plt.bar(x + i*width, vals, width, label=m.capitalize())
plt.xticks(x + width, names_pres, rotation=90)
plt.ylim(0,1.05)
plt.title("Per-Class Precision, Recall & F1-Score")
plt.xlabel("Class"); plt.ylabel("Score"); plt.legend()
plt.tight_layout(); plt.show()

# ── 4) ROC Curves ──────────────────────────────────────────────
plt.figure(figsize=(8,8))
for idx, cls in enumerate(labels_pres):
    fpr, tpr, _ = roc_curve(y_test_cat[:, cls], y_pred_prob[:, cls])
    auc_score   = roc_auc_score(y_test_cat[:, cls], y_pred_prob[:, cls])
    plt.plot(fpr, tpr, lw=1, label=f"{names_pres[idx]} (AUC={auc_score:.2f})")
plt.plot([0,1],[0,1],'k--',lw=1)
plt.title("Multiclass ROC Curves (OvR)")
plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate")
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.tight_layout(); plt.show()
```


asl_landmark_lstm_model.py
metrics = ['precision','recall','f1-score']
x = np.arange(len(names_pres))
width = 0.3
plt.figure(figsize=(14,5))
for i, m in enumerate(metrics):
vals = [report[name][m] for name in names_pres]
plt.bar(x + i*width, vals, width, label=m.capitalize())
plt.xticks(x + width, names_pres, rotation=90)
plt.ylim(0,1.05)
plt.title("Per-Class Precision, Recall & F1-Score")
plt.xlabel("Class"); plt.ylabel("Score"); plt.legend()
plt.tight_layout(); plt.show()
# ── 4) ROC Curves ──
plt.figure(figsize=(8,8))
for idx, cls in enumerate(labels_pres):
fpr, tpr, _ = roc_curve(y_test_cat[:, cls], y_pred_prob[:, cls])
auc_score = roc_auc_score(y_test_cat[:, cls], y_pred_prob[:, cls])
plt.plot(fpr, tpr, lw=1, label=f"{names_pres[idx]} (AUC={auc_score:.2f})")
plt.plot([0,1],[0,1],'k--',lw=1)
plt.title("Multiclass ROC Curves (OvR)")
plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate")
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.tight_layout(); plt.show()
# ── 5) Calibration (Reliability Diagram) ──
y_true_flat = y_test_cat.ravel()
y_prob_flat = y_pred_prob.ravel()
frac_pos, mean_pred = calibration_curve(y_true_flat, y_prob_flat, n_bins=10)
plt.figure(figsize=(6,6))
plt.plot(mean_pred, frac_pos, 's-', label='Aggregated')
plt.plot([0,1],[0,1],'k--', label='Perfect')
plt.title("Reliability Diagram")
plt.xlabel("Mean Predicted Probability")
plt.ylabel("Fraction of Positives")
plt.legend(); plt.tight_layout(); plt.show()
# ── 6) Additional Metrics Bar Chart ──
add_metrics = {
'Log Loss': ll,
'Bal. Accuracy': bal_acc,
'MCC': mcc,
"Cohen's Kappa": kappa,
'Avg Precision': ap_macro,
'Top-3 Acc': top3_acc
}
plt.figure(figsize=(8,5))
plt.bar(add_metrics.keys(), add_metrics.values())
plt.title("Additional Evaluation Metrics")
plt.ylabel("Score"); plt.xticks(rotation=45, ha='right')
plt.ylim(0,1.05)
plt.tight_layout(); plt.show()
Line 1, Column 1
main
Tab Size: 4
Python

The British College_CoverSheet.docx

ORIGINALITY REPORT

3%	3%	0%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to The British College Student Paper	2%
2	uplands-irrigation-mowram.gov.kh Internet Source	<1%
3	Submitted to Botswana Accountancy College Student Paper	<1%
4	Submitted to University of Hertfordshire Student Paper	<1%
5	practicalcheminformatics.blogspot.com Internet Source	<1%
6	www.mdpi.com Internet Source	<1%
7	Ohri, Ankit. "Real-Time American Sign Language Recognition Using Webcam, Computer Vision, and Machine Learning", California State University, Sacramento Publication	<1%
8	eprints.lib.hokudai.ac.jp Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		