**1. Overview**

This Django-based web app (“ASL Sign Translator”) allows users to:

* Register with email verification
* Login/Logout securely
* Manage profiles (name, contact, picture)
* Change passwords
* Make real-time ASL predictions via webcam or uploaded landmarks
* View personal histories (predictions, audit logs, sentences, generated videos)
* Generate English sentences from ASL predictions (using Google Gemini)
* Generate ASL videos from English text
* Administer users and view/delete any records (predictions, logs, sentences, videos)

All key actions are recorded in anAuditLog model for full traceability. A background task automatically cleans up old video files.

**2. Prerequisites**

* Python ≥3.8 and Django 4.x
* MySQL (or another DB); configure in “settings.py” under “DATABASES”
* Environment variables for sensitive keys (“SECRET\_KEY”, DB creds, “EMAIL\_HOST\_PASSWORD”, “GEMINI\_API\_KEY”)—use a “.env” and “django-environ” or “python-dotenv”
* Email backend (SMTP): host, port, TLS, user, password (use App Password if Gmail+2FA)
* Google Gemini API key in “GEMINI\_API\_KEY”
* Installed Apps & Middleware:
* **Python:**

INSTALLED\_APPS += ['accounts','asl','corsheaders']

MIDDLEWARE = ['corsheaders.middleware.CorsMiddleware', ...]

CORS\_ALLOW\_ALL\_ORIGINS = True

* **Static & Media:**

python

STATIC\_URL = '/static/'; STATICFILES\_DIRS=[BASE\_DIR/'static']

MEDIA\_URL = '/media/'; MEDIA\_ROOT = BASE\_DIR/'media'

* **Redirects:**

python

LOGIN\_URL = '/accounts/login/'

LOGIN\_REDIRECT\_URL = '/predict/'

LOGOUT\_REDIRECT\_URL = '/accounts/login/'

**3. Installation & Setup**

**1. Install Python & Django**

* Download and install **Python 3.8+** from https://python.org/downloads/ (choose the installer for your OS).
* Verify installations: python –version or pip --version
* Install Django globally: pip install django

2. **Clone & venv**

* bash
* git clone <repo-url>
* cd <project\_root>
* python -m venv venv
* source venv/bin/activate # macOS/Linux
* venv\Scripts\activate # Windows
* pip install -r requirements.txt

**3. Migrate & Superuser**

* bash
* python manage.py migrate
* python manage.py createsuperuser

**4. Run Server**

* bash
* python manage.py runserver
* Then open http://localhost:8000/ in your web browser.

**5. Verify Email**

* Register a user at /accounts/register/.
* Register a user and check your SMTP logs/inbox for the verification email.

**4. User Registration & Email Verification**

1. Visit “/accounts/register/” and fill in:

- Username

- First & Last Name

- Email

- Contact Number (“+977XXXXXXXXXX”)

- Password & Confirm

**2. You’ll receive an email with a link:**

* https://<your-domain>/accounts/verify/<uuid-token>/

**3. Clicking it sets `is\_verified=True`.**

**4. Attempting login before verification redirects to “/accounts/email-not-verified/”, where you can “resend” or “logout”.**

**5. Login & Logout**

* “Login”: “/accounts/login/” (uses “CustomLoginForm”).
* “Logout”: “/accounts/logout/” → redirects to login.
* Both events fire Django signals (“user\_logged\_in”, “user\_logged\_out”) and are logged.

**6. Profile Management**

Accessible at “/profile/” (login required):

* Update Profile: Name, Contact, Profile Picture
* Change Password: Current + New twice

Success and errors show via Django messages. All updates are audited.

**7. Password Reset Flow**

URL patterns:

* “/password-reset/” → request link
* “/password-reset/done/” → email sent confirmation
* “/reset/<uidb64>/<token>/” → set new password
* “/reset/done/” → complete

Each step logs activity via overridden class-based views.

**8. Prediction & ASL Media Features**

**8.1 Prediction Dashboard**

* “/predict/” renders a live dashboard.
* AJAX POST to “/predict\_landmarks/” with JSON “{ "sequence": [frame1, …, frame10] }”.
* Returns top-3 labels + confidences.
* Records each in `ASLPrediction` (landmarks JSON + labels + confidences + timestamp).
* Logs the prediction event.

**8.2 Personal User Features**

* Prediction History: “/prediction-history/”
* Audit Log History: “/user-history/”
* Generate Sentence: POST to “/generate\_sentence/” with “{ "predictions": ["A","B",…] }”.
* Uses Google Gemini (`gemini-1.5-flash`), saves to `ASLSentenceGeneration`.
* Sentence History: “/sentence-history/”
* English→ASL Page : “/english-to-asl/”
* Generate ASL Video: POST form-data “text="HELLO"”.
* Reads “<BASE\_DIR>/asl\_images/”, creates MP4 at 2 FPS, overlays user & timestamp.
* Saves to “media/asl\_videos/”, logs in “ASLVideoHistory”.
* Video History: “/asl-video-history/”

**8.3 Admin History & Management**

**(superuser only)**

* All Predictions: “/admin\_prediction-history/”
* All Audit Logs: “/admin-user-history/”
* All Sentences: “/admin\_sentence-history/”
* All Videos: “/admin-video-history/”
* Deletion Endpoints (`/delete-…/ID/`) for predictions, logs, sentences, videos via JSON, also audited.

**9. Audit Logging**

**Utility in “asl/utils.py”:**

**Python:**

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

)

* Called in views, signals, and class-based overrides.
* Captures user (nullable), action, description, IP, User-Agent, timestamp.

**10. Background Cleanup Task**

Defined in “accounts/apps.py” under `ready()`, spawning a daemon thread that runs:

**Python:**

while True:

threshold = now() - timedelta(minutes=1)

expired = ASLVideoHistory.objects.filter(created\_at\_\_lt=threshold, is\_deleted=False)

for vid in expired:

os.remove(vid.absolute\_file\_path)

vid.is\_deleted=True; vid.deleted\_at=now(); vid.save()

time.sleep(300)

Removes video files older than 1 minute, every 5 minutes.

**11. Root URL & WSGI**

* “mysite/urls.py” includes:
* “accounts.urls” under “/accounts/”
* “asl.urls” for prediction & history
* Auth and password-reset patterns
* Root redirect → login
* “static()” for “MEDIA\_URL” in dev
* “mysite/wsgi.py” exposes “application” for Gunicorn/uWSGI.

**12. Deployment Tips**

* Collect static: “python manage.py collectstatic”
* Run: “gunicorn mysite.wsgi:application --bind 0.0.0.0:8000”
* Use “systemd” + “Nginx” to serve static and proxy to Gunicorn.
* **In production:**
* “DEBUG=False”, populate “ALLOWED\_HOSTS”
* Secure “SECRET\_KEY”, creds via env vars
* Enable HTTPS (Let’s Encrypt)

**13. Troubleshooting**

* Email fails : verify SMTP settings and App Passwords.
* Media errors: ensure “media/” subdirs exist & are writable.
* CORS issues : confirm “corsheaders.middleware.CorsMiddleware” is first.
* Permissions: double-check “is\_superuser” checks on admin views.

That completes the installation, setup, and feature walkthrough.