**📁 attendance\_app/**

**admin.py**

**This file registers your database models (Attendee, AttendanceLog) with Django’s admin interface. By doing so, Django automatically generates a user-friendly dashboard for these models. Admin users can:**

* **Add or edit attendees manually.**
* **View and manage attendance logs.**
* **Use built-in filters and search features.**

**This removes the need to build custom management tools for your data.**

**apps.py**

**This is a standard Django configuration file that registers the app with the project. It tells Django that this is a valid module to load. You rarely change this file unless you want to execute something when the app starts.**

**attendance\_logger.py**

**Handles all the logic for writing attendance into Excel files using the openpyxl library.  
Main responsibilities:**

1. **Checks if today’s attendance sheet exists. If not, it creates one.**
2. **Adds headers (UID, Name, Subject, Time) to the sheet.**
3. **Checks if the student’s UID is already marked present — prevents duplicates.**
4. **If not already marked, appends a new row with attendance data.**

**Why it matters:**

* **Keeps offline backups in .xlsx format.**
* **Attendance sheets can be opened in Excel, Google Sheets, etc.**

**face\_recognition\_script.py**

**Uses the face\_recognition and cv2 (OpenCV) libraries for verifying the identity of a person based on their face.**

**Steps:**

1. **Loads all registered face images and encodes them into numeric vectors.**
2. **Captures live webcam feed.**
3. **Identifies all faces in the webcam frame.**
4. **Compares each detected face against the known encodings.**
5. **Returns the name of the matched person or 'Unknown'.**

**This file ensures biometric confirmation, adding a layer of security to attendance.**

**forms.py**

**Creates a registration form using Django’s ModelForm. Instead of writing HTML manually, this file generates the form automatically based on the Attendee model.**

**The form handles:**

* **Input validation (e.g., no blank names or invalid photo uploads).**
* **Saves the data to the database.**
* **Provides error messages to the user if the input is invalid.**

**Makes it easy to register students without extra backend code.**

**middleware.py**

**This is a custom middleware that logs every request made to the server. Middleware in Django runs for every request and response.  
Here, it’s used for:**

* **Logging request URLs.**
* **Tracking what pages are visited.**
* **Debugging or performance monitoring.**

**models.py**

**Defines the database structure.  
It has two models:**

1. **Attendee:**
   * **uid: A unique RFID code.**
   * **name: The student or employee name.**
   * **photo: A saved image used in face recognition.**
2. **AttendanceLog:**
   * **Stores a reference to the attendee.**
   * **Records the date, subject, and timestamp of the attendance.**

**These models define how Django interacts with the database.**

**rfid\_reader.py**

**This is the core logic file that handles the real-time attendance process.**

**Workflow:**

1. **Listens to the RFID reader via serial port.**
2. **Reads a UID (unique ID from card).**
3. **Fetches the corresponding student info from the database.**
4. **Opens the webcam and checks if the face matches the saved photo.**
5. **If matched and not already present:**
   * **Records attendance to Excel.**
   * **Plays a success sound and uses speech to say “Attendance marked.”**
6. **If already marked:**
   * **Plays a duplicate sound and says “Already present.”**

**This file brings together all the systems: RFID, webcam, database, Excel, and audio.**

**tests.py**

**Placeholder for unit testing the app. You would use this file to write code that checks whether:**

* **Models are saved correctly.**
* **Views behave as expected.**
* **Forms and logic don’t break on bad input.**

**It ensures code quality and helps detect bugs early.**

**urls.py**

**Maps specific URLs to views in your app. For example:**

* **/register/ → The registration page.**
* **/dashboard/ → The list of attendance logs.**
* **/search/ → A page to filter logs by UID, date, or subject.**

**This file is crucial for telling Django what to do when a user opens a specific link.**

**utils.py**

**Contains helper functions — mainly save\_uploaded\_file () that handles saving the user’s uploaded photo to disk.**

**Keeping such functions in a separate file helps keep views cleaner and more organized.**

**views.py**

**Defines how each webpage works:**

* **register\_attendee(): Displays the form, validates input, and saves it.**
* **dashboard(): Pulls all attendance logs and shows them on a page.**
* **search\_logs(): Lets users search attendance by UID, date, or subject.**

**This is the brain of your web interface — it handles logic behind the pages you see in your browser.**

**📁 rfid\_web/**

**\_\_init\_\_.py**

**Marks the folder as a Python package so Django can load it. Required for the app to work.**

**asgi.py**

**Prepares the app for ASGI (Asynchronous Server Gateway Interface), which is used for real-time apps like WebSockets. Not used here directly, but helpful for advanced deployments.**

**settings.py**

**This is the main config file for the whole project.**

**It defines:**

* **Which apps are used.**
* **What database is connected.**
* **Where media and static files are stored.**
* **Security settings.**
* **Installed middleware.**
* **Templates and URL routing.**

**Think of it as the control center for your Django project.**

**urls.py**

**Defines top-level URL routes and passes them to the app-specific urls.py. Also sets up the admin panel path (/admin/).**

**wsgi.py**

**Prepares the app to run on a WSGI (Web Server Gateway Interface) server. Most used in deployments like Apache, Nginx with Gunicorn, etc.**

**📁 templates/**

**admin/index.html**

**Overrides the default admin dashboard. You can customize branding or show important links here.**

**dashboard.html**

**Shows the attendance records in table format. Also includes filters to search by UID, date, or subject. Data is loaded from the database via the dashboard() view.**

**index.html**

**Landing page of the website. It’s usually the first screen the user sees. Could contain navigation to register, view logs, or dashboard.**

**register.html**

**The form for new users to input UID, name, and upload a photo. It posts data to the server and displays any validation errors if the form is filled incorrectly.**

**success.html**

**Confirmation page shown when a user registers successfully or uploads data correctly.**

**📁 static/**

**css/style.css**

**Defines visual design rules: colors, fonts, layout spacing, etc. Helps keep the site looking clean and uniform.**

**sounds/success.wav**

**Audio file played when a student is marked present successfully.**

**sounds/duplicate.wav**

**Played when the system detects a duplicate (already marked) entry.**

**📄 manage.py**

**A command-line tool for interacting with Django.**

**You use it to:**

* **Start the development server (runserver)**
* **Apply database changes (migrate)**
* **Create admin accounts (createsuperuser)**
* **Run tests**