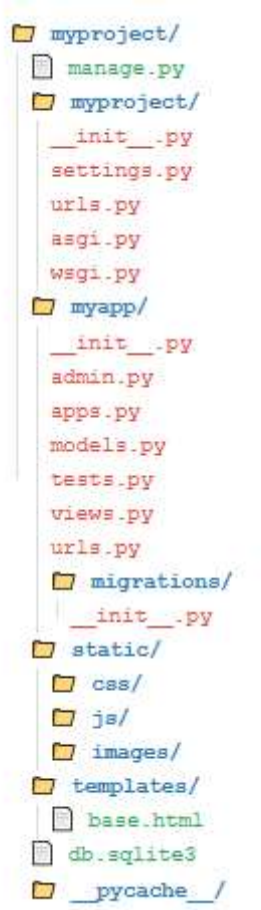
TRAINING DAY2 REPORT

25 JUNE 2025

STRUCTURE OF PROJECT DIRECTORY:



Project Structure

- **manage.py**: A command-line utility that lets you interact with your Django project.

myproject Directory

- **init.py**: An empty file that indicates that this directory should be treated as a Python package.

- **settings.py**: Contains all the configuration settings for your Django project, such as database settings and installed apps.
- **urls.py**: Defines the URL patterns for your project, mapping URLs to views.
- **asgi.py**: An entry-point for ASGI-compatible web servers to serve your project, used for asynchronous applications.
- **wsgi.py**: An entry-point for WSGI-compatible web servers to serve your project, used for synchronous applications.

myapp Directory

- **init.py**: An empty file that indicates that this directory should be treated as a Python package.
- **admin.py**: Registers models with the Django admin site for easy management through a web interface.
- **apps.py**: Contains the configuration for the app, including its name and any specific settings.
- **models.py**: Defines the data models (database schema) for your application using Django's ORM.
- **tests.py**: Contains test cases for your application to ensure that your code behaves as expected.
- **views.py**: Contains the logic for handling requests and returning responses, defining what data to display.
- **urls.py**: Maps URLs to views specific to this app, allowing for modular URL handling.
- **migrations/**: A directory that contains migration files for database schema changes, allowing for version control of the database.

Static Directory

- **css/**: A directory for storing CSS files used for styling your web application.
- **js/**: A directory for storing JavaScript files used for client-side scripting in your web application.
- **images/**: A directory for storing image files used in your web application.

Templates Directory

- **base.html**: A base template file that can be extended by other templates, providing a common structure for your web pages.

Database File

- **db.sqlite3**: The default SQLite database file where your application's data is stored.

pycache Directory

- **pycache/**: A directory that stores compiled Python files (.pyc) for performance optimization.

WHAT IS DJANGO ADMIN?

The **Django Admin** is a **built-in web interface** that allows you to:

- View and manage database records
- Add, edit, delete entries
- Manage users, groups, permissions
- Perform administrative tasks without writing any code

```
(.venv) PS C:\Users\WELCOME\Desktop\django\blogapi> python manage.py createsuperuser
Username (leave blank to use 'welcome'): admin1
Email address: admin123@gmail.com
Password:
Password (again):
Superuser created successfully.
```

FIGURE : How to create a super user

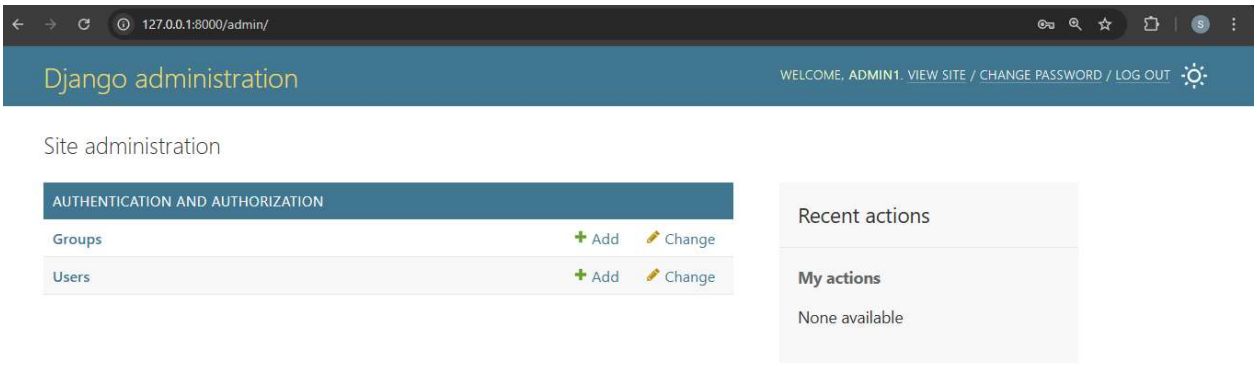


FIGURE : Demonstration of admin page and it's API

Key Components of Django Admin Page

1. Login Page

- Only registered superusers/staff can log in.
- Authenticates users using Django's built-in auth system.

2. Dashboard (Admin Index)

- Displays all registered models from each app.
- Grouped by application name.

3. Model List View

- Clicking on a model shows a **list of all entries** (like a table view).
- Columns can be customized using list_display in admin.py.

4. Model Form View (Add/Edit Entry)

- Clicking "Add" or an entry opens a form to **create or edit** the item.
- Auto-generates forms based on model fields.
- You can customize form layout using fieldsets, readonly_fields, etc.

5. Search Bar

- You can enable search across selected fields using search_fields.

6. Filter Sidebar

- Appears on the right (optional).
- Use list_filter to filter records based on specific fields.

7. Pagination Controls

- Handles large datasets by paginating results.

8. Bulk Actions

- Select multiple entries and apply actions like **Delete**.
- Custom actions can also be added via actions.

9. Navigation Bar

- Located at the top:
    - Shows username, change password, and logout.

10. Permission System

- You can define what a user/staff can:
    - Add / Change / Delete / View specific models.
- Controlled via the User and Group models.

What is db.sqlite3 in Django?

db.sqlite3 is the **default database file** created by Django when you start a new project.

Details about db.sqlite3:

1. **SQLite Database**
    a. A lightweight, file-based **relational database**.
    b. No server installation required — the entire database is stored in this one file.
2. **Created Automatically**

    When you run:

    ```
    python manage.py migrate
    ```

    Django creates db.sqlite3 and sets up tables based on built-in and app models.

3. **Good for Development**
    a. Fast and easy for testing, small projects, and local development.
    b. Not ideal for large, production-scale apps.
4. **Located in Project Root**

    You'll find it here:

    myproject/
    ├── db.sqlite3

5. **Do Not Manually Edit**

    Always interact with it using Django's ORM (models and migrations).

But before using you need to install db.sqlite3 in your system.

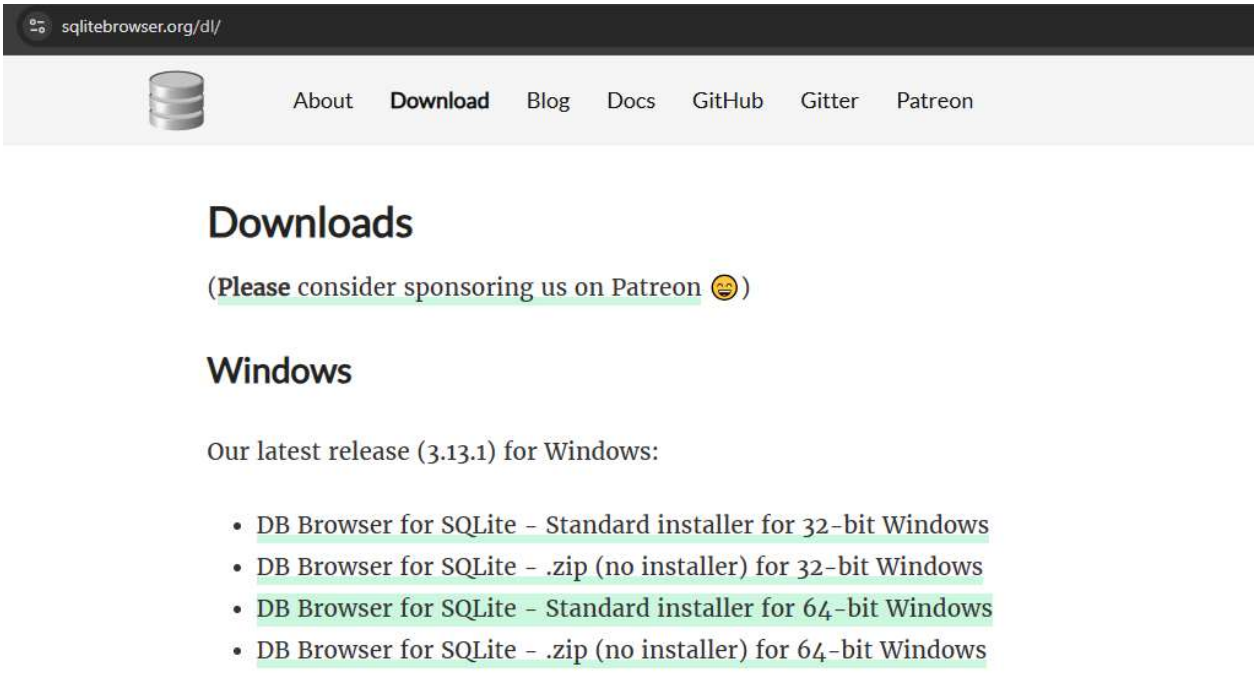


FIGURE : Install db.sqlite3