User Guide – RESTful Blog Application API

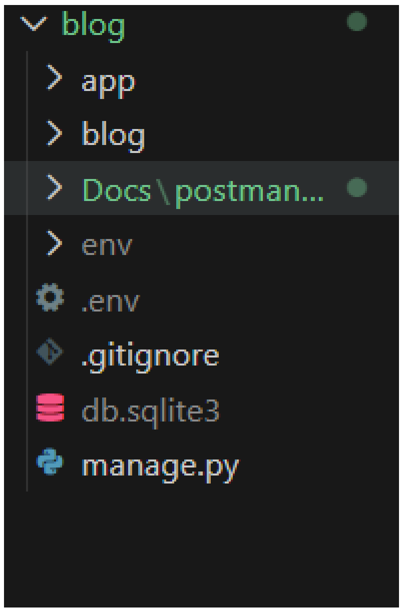
1. Introduction

The API allows users to register, authenticate, and manage blog posts and comments securely using JWT authentication.

2. API Framework , Language Selection ,database and Project structure

- I have selected python Django Rest framework
- Django REST Framework (DRF) was chosen over Flask for developing this Blog Application API because it provides a more structured, scalable, and production-ready solution for building RESTful APIs.
- Firstly, Django REST Framework offers **built-in features** such as **serializers, authentication, permissions, and browsable APIs**. These features significantly reduce development time compared to Flask, where most of these components need to be implemented manually or integrated using third-party libraries.
- Database: postgresql

2.1 Project Structure.



```
v blog
  > app
  > blog
  > Docs\postman...
  > env
    .env
    .gitignore
    db.sqlite3
    manage.py
```

- All the API code is in App folder and documents in docs folder
- First activate the Env
- Running the server:
  (env) C:\Users\MAHANTHESH H R\Desktop\Intership-project-Blog\blog>py manage.py runserver

2. Base URL

http://127.0.0.1:8000/

3. Authentication

The API uses JSON Web Token (JWT) for authentication.

3.1 User Registration

Endpoint: POST /register

Request Body:

```
{
  "username": "example",
  "email": "example@email.com",
  "password": "password123"
}
```

3.2 User Login

Endpoint: POST /api/token/

Request Body:

```
{
  "username": "example",
  "password": "password123"
}
```

Response:

```
{
  "access": "jwt_access_token",
  "refresh": "jwt_refresh_token"
}
```

Use the access token in headers:

Authorization: Bearer <access_token>

For Access token refresh : POST /api/token/refresh/

4. Blog Post APIs

4.1 Create Post

POST /posts

Requires authentication.

4.2 Get All Posts

GET /posts

4.3 Get Single Post

GET /posts/{id}

4.4 Update Post

PUT /posts/{id}

Only the author can update.

4.5 Delete Post

DELETE /posts/{id}

Only the author can delete.

5. Comment APIs

5.1 Create Comment

POST /comments

5.2 Get Comments for a Post

GET /comments?post_id={post_id}

5.3 Update Comment

PUT /comments/{id}

5.4 Delete Comment

DELETE /comments/{id}

6. Error Handling

The API uses standard HTTP status codes such as 200, 201, 400, 401, 403, and 404.

Final Notes

1. For user login or get access token the end point like this : POST /api/token/
   Add “/” at the end for login only

2. For creating and updating a comment the Api end point body requires comment id , post id