# Set up and working of the application.

# Django RESTful API Project

This project is a Django-based RESTful API application designed for managing an inventory system. It includes endpoints for user registration, login, and items inventory with token-based authentication.

## Getting Started

These instructions will guide you through setting up and running the project on your local machine for development and testing purposes.

### Prerequisites

Before running the project, ensure you have the following installed:

- Python 3

- pip (Python package manager)

- Virtualenv (optional but recommended for creating an isolated Python environment)

### Installation

1. Download the zip file from the GitHub repository.

2. Extract the zip file and open the folder in Visual Studio Code.

3. Open the integrated terminal in Visual Studio Code. Ensure that you are in the project's root directory (where `manage.py` is located).

### Setting Up the Virtual Environment (Optional)

Create a virtual environment to isolate your project dependencies:

python3 -m venv venv

source venv/bin/activate # On Windows use `venv\Scripts\activate`

**Installing Dependencies**

Install the required packages using pip:

pip install -r requirements.txt

**Running the Application**

Start the development server using the following command:

“python3 manage.py runserver”

A screen shot of a computer

Description automatically generated

**Database**

This project uses SQLite, which is a file-based database. You can add the SQLite extension in Visual Studio Code to view and interact with the database tables related to the inventory and items.

A screenshot of a computer

Description automatically generated

Open the file in separate window of visual studio code to view the database.

A screenshot of a computer

Description automatically generated

**Define your models**: In your Django app (for example, in the **items** app), define model classes in **models.py**.

A screenshot of a computer

Description automatically generated

**Make migrations**: Once you have defined your models, you run the **makemigrations** command to create migration files. These files contain the changes needed to apply to the database schema.

python manage.py makemigrations

**Apply migrations**: After creating migrations, you run the **migrate** command to apply them to the database. This will create the necessary tables and fields.

python manage.py migrate

The **migrate** command is also used to apply any changes you make to your models over time, ensuring your database schema stays in sync with your code.

The initial SQLite database file (**db.sqlite3**) will be automatically created in your project's root directory the first time you run the **migrate** command, if it doesn't already exist.

**API Endpoints**

The API supports the following operations:

* **User Login**: **POST /api/login/** - On successful login, a token will be generated for authentication.
* **Items Retrieval**: **GET /api/items/** - Retrieve a list of items. Requires token authentication.
* **Filtering Items**: Query parameters can be used to filter results based on stock status.
  1. **User Registration (POST):** [**http://localhost:8000/api/v1/auth/register/**](http://localhost:8000/api/v1/auth/register/)
  + A new user registers by providing required details (like username, password, email, etc.).
  + The server stores the user information and may return a confirmation that registration was successful.

A screenshot of a computer

Description automatically generated

* 1. **. User Login (POST):** [**http://localhost:8000/api/v1/auth/login/**](http://localhost:8000/api/v1/auth/login/)
* The user logs in with their credentials.
* Upon successful login, the server responds with an authorization token (often a JWT - JSON Web Token).

A screenshot of a computer

Description automatically generated

* 1. **Items Retrieval (GET):** [**http://localhost:8000/api/v1/items**](http://localhost:8000/api/v1/items)
  + To access protected endpoints, such as retrieving items, the client must include the authorization token in the request header.
  + The token is typically included as a Bearer token in the **Authorization** header.

A screenshot of a computer

Description automatically generated

* 1. **Filtered Inventory Retrieval (GET):** [**http://localhost:8000/api/v1/inventory?stock\_status=in%20stock&min\_stock=0&max\_stock=20**](http://localhost:8000/api/v1/inventory?stock_status=in%20stock&min_stock=0&max_stock=20)
* Similar to the items retrieval, the request must include the **Authorization** header with the Bearer token.
* The endpoint filters the inventory based on the given query parameters (**stock\_status**, **min\_stock**, **max\_stock**).

A screenshot of a computer

Description automatically generated

**Testing**

To run unit tests for each API endpoint, use the following command in the terminal:

“python3 manage.py test items”

This will execute the tests defined in the **tests.py** file within the **items** application.

A screenshot of a computer screen

Description automatically generated