**Inventory Management System API**

This project is a **RESTful API** built using **Django Rest Framework (DRF)** for managing inventory items. The API supports authentication via **JWT** and optimizes performance using **Redis caching** for frequently accessed items. The database used is **PostgreSQL**.

**Table of Contents**

* [Features](#features)
* [Setup Instructions](#setup-instructions)
* [API Endpoints](#api-endpoints)
* [Usage Examples](#usage-examples)
* [Logging](#logging)
* [Running Tests](#running-tests)

**Features**

* **JWT Authentication** for secure API access.
* **CRUD Operations** for managing inventory items.
* **Redis Caching** to improve response times for frequently accessed data.
* **PostgreSQL** database for reliable and scalable data storage.
* **Comprehensive Unit Tests** to ensure code reliability.

**Setup Instructions**

**Prerequisites**

* **Python 3.x** installed.
* **PostgreSQL** database.
* **Redis** installed and running.

**Step 1: Clone the Repository**

git clone https://github.com/Sahil/inventory-management-api.git

cd inventory-management-api

**Step 2: Set Up a Virtual Environment**

python -m venv

env source env/bin/activate # For Linux/Mac

env\Scripts\activate # For Windows

**Step 3: Install Dependencies**

pip install -r requirements.txt

**Step 4: Configure the Database**

Update the DATABASES setting in inventory\_management/settings.py with your PostgreSQL credentials:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'your\_db\_name',

'USER': 'your\_db\_user',

'PASSWORD': 'your\_db\_password',

'HOST': 'localhost',

'PORT': '5432',

}

}

**Step 5: Configure Redis**

CACHES = {

'default': {

'BACKEND': 'django\_redis.cache.RedisCache',

'LOCATION': 'redis://127.0.0.1:6379/1',

'OPTIONS': {

'CLIENT\_CLASS': 'django\_redis.client.DefaultClient',

}

}

}

**Step 6: Apply Migrations**

python manage.py makemigrations

python manage.py migrate

**Step 7: Create a Superuser**

python manage.py createsuperuser

**Step 8: Run the Development Server**

python manage.py runserver

Access the application at [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/).

**API Endpoints**

**Authentication**

* **User Registration**: POST /api/users/register/
* **JWT Login**: POST /api/token/
* **Refresh JWT Token**: POST /api/token/refresh/

**Inventory Management**

* **Create Item**: POST /api/additems/
* **Get All Items**: GET/api/items/
* **Get Item by ID**: GET /api/items/{id}/
* **Update Item by ID**: PUT /api/items/{id}/update/
* **Delete Item by ID**: DELETE /api/items/{id}/delete/

**JWT Authentication**

Add the Authorization: Bearer <access\_token> header to requests to access protected endpoints.

**Usage Examples**

* 1. **User Registration**

**POST /api/users/register/**

{

"username": "newuser",

"email": "newuser@example.com",

"password": "password123"

}

**Response**:

{

"id": 1,

"username": "newuser",

"email": "newuser@example.com"

}

* 1. **Obtain JWT Token**

**POST /api/token/**

{

"username": "newuser",

"password": "password123"

}

**Response**:

{

"access": "<access\_token>",

"refresh": "<refresh\_token>"

}

* 1. **Create an Inventory Item**

**POST /api/additems/**

**Authorization: Bearer <access\_token>**

{

"name": "Laptop",

"description": "A high-performance laptop."

}

**Response**:

{

"id": 1,

"name": "Laptop",

"description": "A high-performance laptop",

"created\_at": "2024-09-28T12:34:56.789Z"

}

* 1. **Get an all Inventory Item**

**GET /api/items/**

**Authorization: Bearer <access\_token>**

**Response:**

{

"id": 1,

"name": "Laptop",

"description": "A high-performance laptop",

"created\_at": "2024-09-28T12:34:56.789Z"

}

* 1. **Get an Inventory Item**

**GET /api/items/1/**

**Authorization: Bearer <access\_token>**

**Response:**

{

"id": 1,

"name": "Laptop",

"description": "A high-performance laptop",

"created\_at": "2024-09-28T12:34:56.789Z"

}

* 1. **Update an Inventory Item**

**PUT /api/items/1/update/**

**Authorization: Bearer <access\_token>**

{

"name": "Gaming Laptop",

"description": "A laptop for gaming."

}

**Response**:

{

"id": 1,

"name": "Gaming Laptop",

"description": "A laptop for gaming",

"created\_at": "2024-09-28T12:34:56.789Z"

}

* 1. **Delete an Inventory Item**

**DELETE /api/items/1/delete/**

**Authorization: Bearer <access\_token>**

{

"detail": "Item deleted."

}

**Logging**

The API includes a logging system that captures significant events, such as:

* **API Usage**: Logs the requests made to the API and the responses returned.
* **Errors**: Captures any errors that occur during API calls.
* **Events**: Important operations, such as item creation, retrieval, updating, and deletion, are logged for monitoring purposes.

**Log File Locations:**

* **logs/inventory\_api.log**: Contains general logs for API usage and operations.
* **logs/error.log**: Contains logs specifically for errors encountered during API operation.

**Viewing Logs:**

You can view the log files in the logs/ directory to monitor API usage and errors.

**Running Tests**

You can run the test suite using Django’s test command:

**python manage.py test**

This will run the unit tests for the inventory system, ensuring that all CRUD operations, JWT authentication, and Redis caching work as expected.