Flask API with JWT Authentication and File Upload

**Ibad Ur Rahman Mohammed**

**Rahul Putta**

**Sasidhar Jonnalagadda**

This is a Flask-based API that includes JWT authentication, file upload functionality, and MySQL database integration. The application allows user registration, login, and the ability to upload files with JWT authentication.

# Features

- User Registration  
- User Login with JWT Authentication  
- File Upload (authenticated users only)  
- CRUD operations on public items  
- JWT token-based access control for protected routes

# Requirements

To set up and run this project, you need the following:  
- Python 3.7+  
- MySQL Database  
- Flask  
- Required Python packages (see `requirements.txt`)

# Setup

## 1. Clone the Repository

First, clone the repository to your local machine:  
```bash  
git clone https://github.com/rahulputta0209/flask-api.git  
cd flask-api  
```

## 2. Install Dependencies

Create a virtual environment and install the required Python packages:  
```bash  
python3 -m venv venv  
source venv/bin/activate # For Windows use `venv\Scripts\activate`  
pip install -r requirements.txt  
```

## 3. Configure MySQL Database

Ensure that MySQL is installed and running on your system. Then, create a new MySQL database and user:  
1. Log in to MySQL:  
```bash  
mysql -u root -p  
```2. Create the database and user:  
```sql  
CREATE DATABASE flask\_api;  
CREATE USER 'root'@'localhost' IDENTIFIED BY 'Ibad'; # Update with your password  
GRANT ALL PRIVILEGES ON flask\_api. TO 'root'@'localhost';  
FLUSH PRIVILEGES;  
```3. Create the necessary tables in the `flask\_api` database:  
```sql  
CREATE TABLE users (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(255) UNIQUE NOT NULL,  
 password VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE public\_items (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 description TEXT NOT NULL,  
 is\_public BOOLEAN DEFAULT TRUE  
);  
```

## 4. Set Environment Variables

Make sure to set the `SECRET\_KEY` for the application. You can do this by creating a `.env` file in the project root directory with the following content:  
```env  
SECRET\_KEY=your\_secret\_key # Replace with a secure random key  
```Make sure to add any other sensitive information here, such as MySQL credentials.

## 5. Running the Application

To run the Flask application, use the following command:  
```bash  
python app.py  
```This will start the Flask development server at `http://127.0.0.1:5000`.

# API Endpoints

## User Registration

- POST `/register`  
 - Request Body:  
 ```json  
 {  
 "username": "your\_username",  
 "password": "your\_password"  
 }  
 ```

## User Login

- POST `/login`  
 - Request Body:  
 ```json  
 {  
 "username": "your\_username",  
 "password": "your\_password"  
 }  
 ```  
 - Response:  
 ```json  
 {  
 "token": "your\_jwt\_token"  
 }  
 ```

## Protected Route

- GET `/protected`  
 - Requires a valid JWT token in the `Authorization` header:  
 ```bash  
 Authorization: Bearer <your\_jwt\_token>  
 ```

## Upload File

- POST `/upload`  
 - Request Body:  
 - The file should be included as part of the form-data in the request.  
 - Requires authentication (JWT token).

## Public Items

- GET `/public-items`  
 - Retrieves a list of public items.  
- POST `/public-items`  
 - Request Body:  
 ```json  
 {  
 "name": "Item name",  
 "description": "Item description"  
 }  
 ```  
 - Requires authentication (JWT token).  
- PUT `/public-items/<item\_id>`  
 - Request Body:  
 ```json  
 {  
 "name": "Updated item name",  
 "description": "Updated item description"  
 }  
 ```  
 - Requires authentication (JWT token).  
- DELETE `/public-items/<item\_id>`  
 - Requires authentication (JWT token).