

Experiment no 07

Validating RESTful APIs using Postman.

```
> Users > Lenovo > Desktop > jwt-auth-app > models.py > ...
1  from werkzeug.security import generate_password_hash, check_password
2
3  class User:
4      def __init__(self, id, username, password, role='user'):
5          self.id = id
6          self.username = username
7          self.password_hash = generate_password_hash(password)
8          self.role = role
9
10     def check_password(self, password):
11         return check_password_hash(self.password_hash, password)
12
13     # In-memory user store
14     users = []
15     user_id_counter = 1
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Users > Lenovo > Desktop > jwt-auth-app > app.py > ...
1  from flask import Blueprint, request, jsonify
2  from flask_jwt_extended import create_access_token
3  from models import create_user, get_user_by_username
4
5  auth_bp = Blueprint('auth', __name__)
6
7  @auth_bp.route('/register', methods=['POST'])
8  def register():
9      data = request.get_json()
10     username = data.get('username')
11     password = data.get('password')
12     role = data.get('role', 'user')
13
14     if not username or not password:
15         return jsonify({'message': 'Username and password required'}), 400
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
requirements.txt  models.py  auth.py  main.py  app.py  postman_collection.json  TODO

C: > Users > Lenovo > Desktop > jwt-auth-app > {} postman_collection.json > ...

1  {
2    "info": {
3      "name": "JWT Auth API Validation",
4      "description": "Postman collection for validating the JWT authentication API endpoints",
5      "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
6    },
7    "item": [
8      {
9        "name": "Register User",
10       "request": {
11         "method": "POST",
12         "header": [
13           {
14             "key": "Content-Type",
15             "value": "application/json"
16           }
17         ]
18       }
19     ]
20   }
21 }
```

```
C: > Users > Lenovo > Desktop > ☒ TODO.md > abc # TODO: Implement JWT Auth

1  # TODO: Implement JWT Authentication and User Roles
2
3  - [x] Create project directory `jwt-auth-app`
4  - [x] Create `requirements.txt` with necessary dependencies
5  - [x] Create `models.py` for User model with roles
6  - [x] Create `auth.py` for authentication routes (register, login, refresh token)
7  - [x] Create `app.py` for main application setup
8  - [x] Create `main.py` for protected routes with role checks
9  - [x] Install dependencies using pip
10 - [x] Create Postman collection for API validation
11 - [x] Run the Flask application and test endpoints
12
```

User Dashboard

Logout

Profile Information

USER

Your account details

Email

mehakhauja@gmail.com

User ID

770b5cc0-036a-46ca-a007-d54daf7768d1

Role

User

Available Endpoints

Full-stack authentication API

POST /register Create new user account

POST /login Authenticate and get JWT

GET /user Access user dashboard

GET /admin Admin-only protected route