Technical Assignment: JWT Authentication API

Task

Build and deploy a JWT-based authentication API using Docker on AWS EC2.

Requirements

1. API Endpoints (3 total)

- `POST /api/auth/login/` → Takes `{"username": "user", "password": "pass"}` → Returns JWT token
- `POST /api/auth/verify/` → Takes `{"token": "jwt_token"}` → Returns token validation status
- `GET /api/auth/validate/` → Requires JWT in Authorization header → Returns `{"valid": true, "user": "username", "expires": "timestamp"}`

2. Technical Requirements

- Django + Django REST Framework
- JWT authentication (PyJWT or similar)
- **Dockerized application** (mandatory)
- Dockerfile and docker-compose.yml
- Basic user model (can use Django's default User)

3. Sample Responses

JSON

```
// POST /api/auth/login/
{"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...", "expires": "2025-07-08T10:30:00Z"}

// POST /api/auth/verify/
{"valid": true, "message": "Token is valid"}
```

```
// GET /api/auth/validate/
{"valid": true, "user": "admin", "expires":
"2025-07-08T10:30:00Z"}
```

4. Deployment

- Deploy Docker container on AWS EC2
- API should be publicly accessible
- Include sample user credentials in README

5. Deliverables

- GitHub repository with source code
- Dockerfile and docker-compose.yml
- Live API URL with sample credentials
- API testing examples (curl commands or Postman)
- Send your assignment submission to vismay@sharpstakes.ca

Evaluation Criteria

- Correct JWT implementation
- Proper Docker containerization
- Successful EC2 deployment
- Clear documentation

Timeline: 7 days