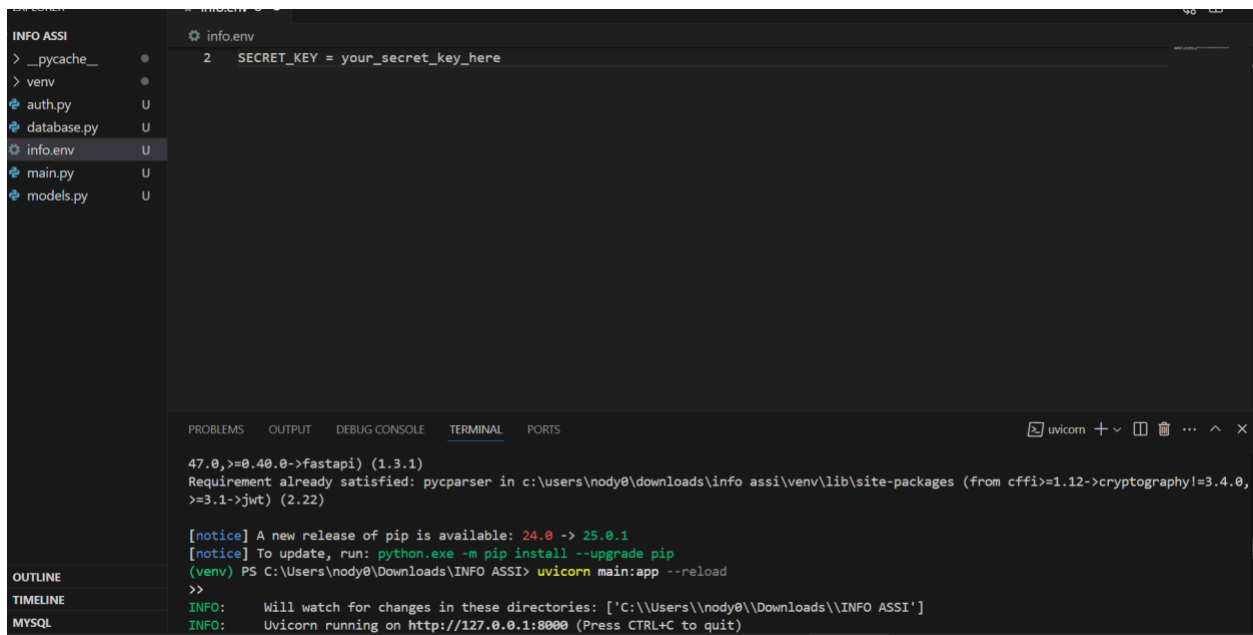


Nada mohamed abd elsatar

2205173

**Note:** I was unable to upload the database or the entire project folder to GitHub directly.



```
INFO ASSI
> _pycache_
> venv
+ auth.py U
+ database.py U
+ info.env U
+ main.py U
+ models.py U

2 SECRET_KEY = your_secret_key_here

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
47.0.0>=0.40.0->fastapi) (1.3.1)
Requirement already satisfied: pycparser in c:\users\nody0\downloads\info assi\venv\lib\site-packages (from cffi>=1.12->cryptography!=3.4.0,
>=3.1->jwt) (2.22)

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Users\nody0\Downloads\INFO ASSI> uvicorn main:app --reload
>>
INFO: Will watch for changes in these directories: ['C:\\Users\\nody0\\Downloads\\INFO ASSI']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

## 1. Introduction

This report outlines the steps taken to complete the Information Security Management task, which involved designing a database, developing a RESTful API, and implementing authentication and CRUD operations. The project was developed using PHP, MySQL (phpMyAdmin), and JWT authentication.

## 2. Database Design

A database named security\_mgmt was created using phpMyAdmin.  
Two tables were designed:

#### Users Table

id (Primary Key, Auto-increment)

name (String, Required)

username (String, Unique, Required)

password (Hashed, Required)

#### Products Table

pid (Primary Key, Auto-increment)

pname (String, Required)

description (Text)

price (Decimal, Required)

stock (Integer, Required)

created\_at (Timestamp, Default Current Time)

### 3. API Development

A RESTful API was developed with the following functionalities:

#### Authentication

Implemented JWT-based authentication.

Created a SignUp endpoint to register users.

Implemented a Login endpoint that generates a JWT token valid for 10 minutes.

Secured protected routes by requiring a valid token.

### User Operations

POST /signup → Registers a new user.

POST /login → Authenticates user and returns JWT token.

PUT /users/{id} → Updates user details (Only authorized users with valid tokens).

### Product Operations (Require JWT Token)

POST /products → Adds a new product.

GET /products → Retrieves all products.

GET /products/{pid} → Retrieves a single product by ID.

PUT /products/{pid} → Updates product details.

DELETE /products/{pid} → Deletes a product.

## 4. Security Measures

Password Hashing: User passwords are securely hashed before storing.

JWT Middleware: Implemented middleware to validate JWT tokens before accessing protected routes.

Environment Variables: Used environment variables for storing sensitive data such as database credentials and JWT secret keys.