CY321

Secure Software Development Lifecycle

Week 04 Deliverables

Secure Password Manager

Team Members :

Ammaid Saleem 2022344

Humayun Nasir 2022189

Muhammad Mursaleen 2022401

Asad Ali 2022903

Faculty : Cyber Security

Submitted to : Dr Zubair Ahmad

Secure Password Manager - Week 04 Documentation

1. Introduction

This documentation outlines the deliverables completed during Week 04 of the Secure Password Manager project. The main objective of this week was to begin secure implementation by applying secure coding practices and building a secure authentication system with initial backend development.

2. Project Folder Structure

The Week4 folder is structured as follows to separate responsibilities and maintain clean modular code:

```
Week4/
├── backend/
│   ├── app.py
│   ├── auth.py
│   ├── .env
│   └── requirements.txt
├── database/
│   ├── db_config.py
│   └── init_db.sql
├── README.md
└── secure_coding_checklist.md
```

3. Secure Authentication System

Secure user registration and login functionality was implemented using Flask. The following practices were followed:

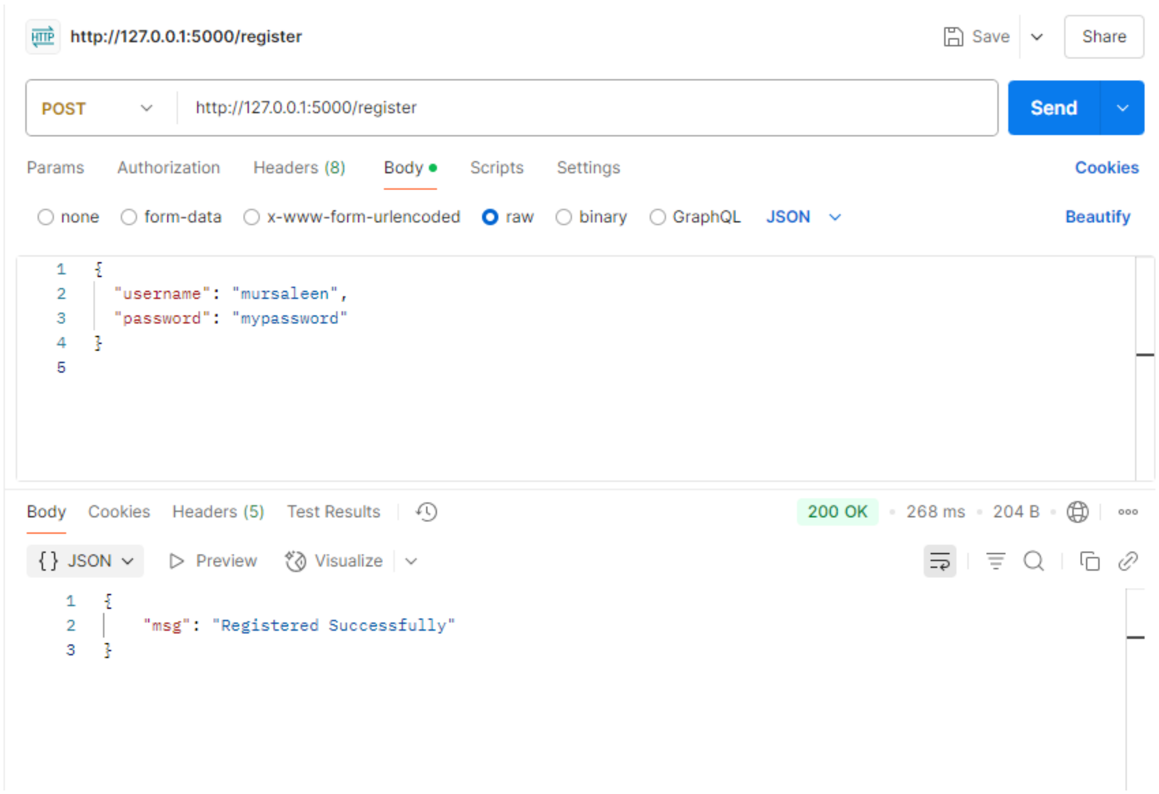
- Passwords are hashed using bcrypt before being stored.
- JWT tokens are used for authentication and session handling.
- A secret key is stored in a .env file (not pushed to GitHub).

3.1 Register API

Registers a user with a hashed password.

Endpoint: POST /register

Request body (JSON): { 'username': '...', 'password': '...' }



POST http://127.0.0.1:5000/register

```
{
  "username": "mursaleen",
  "password": "mypassword"
}
```

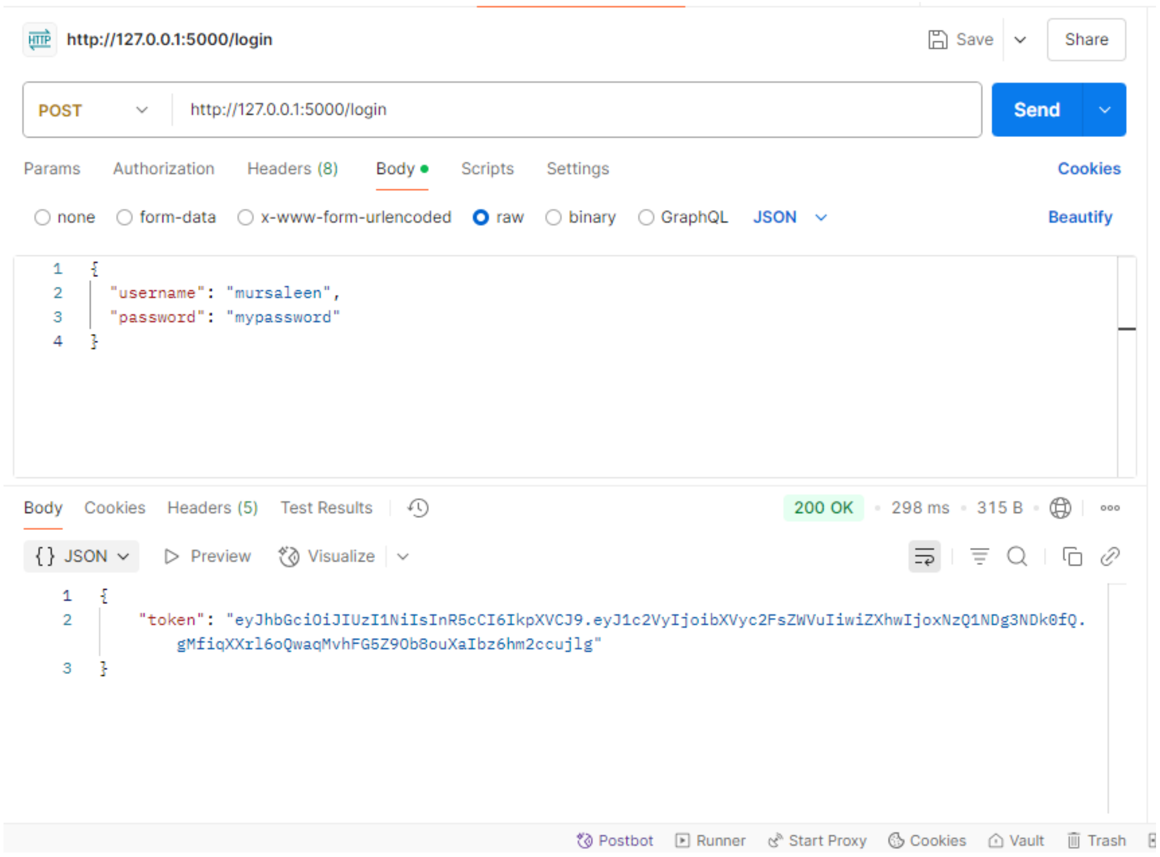
```
{
    "msg": "Registered Successfully"
}
```

3.2 Login API

Authenticates the user and returns a JWT token.

Endpoint: POST /login

Request body (JSON): { 'username': '...', 'password': '...' }



http://127.0.0.1:5000/login

POST http://127.0.0.1:5000/login

```
{
  "username": "mursaleen",
  "password": "mypassword"
}
```

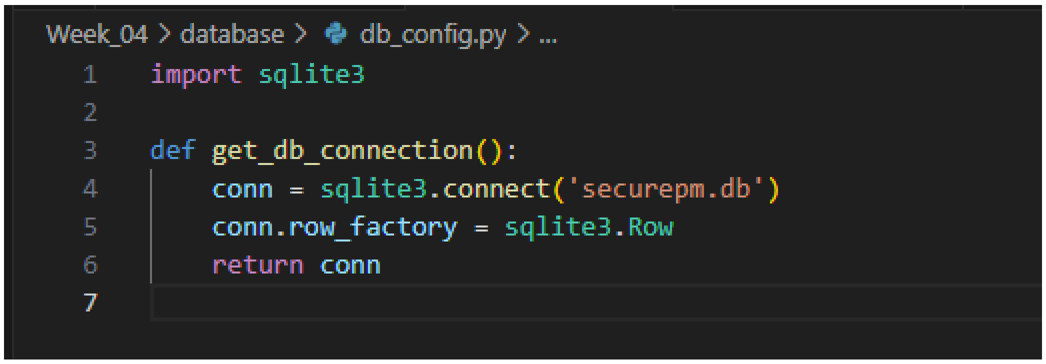
200 OK • 298 ms • 315 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoibXVyc2FsZWVuIiwiZXhwIjoxNzQ1NDg3NDk0fQ.
        gMfiqXXrl6oQwaqMvhFG5Z9Ob8ouXaIbz6hm2ccujlg"
}
```

4. Database Setup and Security

A SQLite database was used to store user credentials. The database schema was created using SQL script, and database connectivity was configured in Python using sqlite3.

Security Measures:
- Used parameterized queries to prevent SQL injection.
- Passwords are stored only after hashing.
- Configuration separated into db_config.py for clarity.



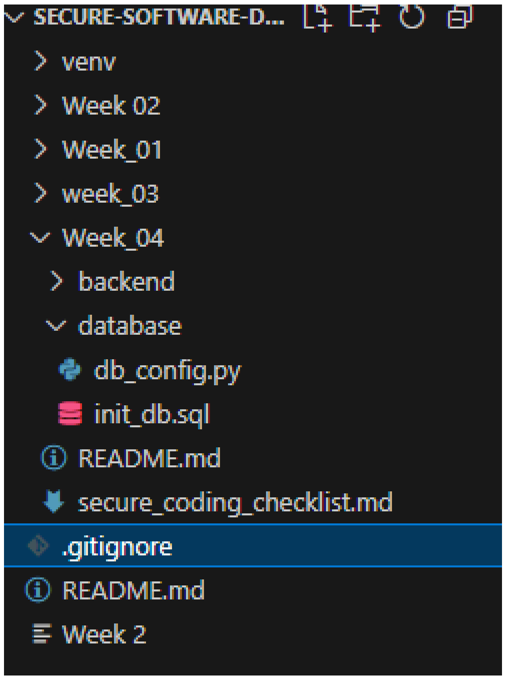
Week_04 > database > db_config.py > ...

```python
import sqlite3

def get_db_connection():
    conn = sqlite3.connect('securepm.db')
    conn.row_factory = sqlite3.Row
    return conn
```

5. Secret Management (.env)

Environment variables were used to securely store sensitive data like secret keys. The .env file is excluded from GitHub by adding it to .gitignore.

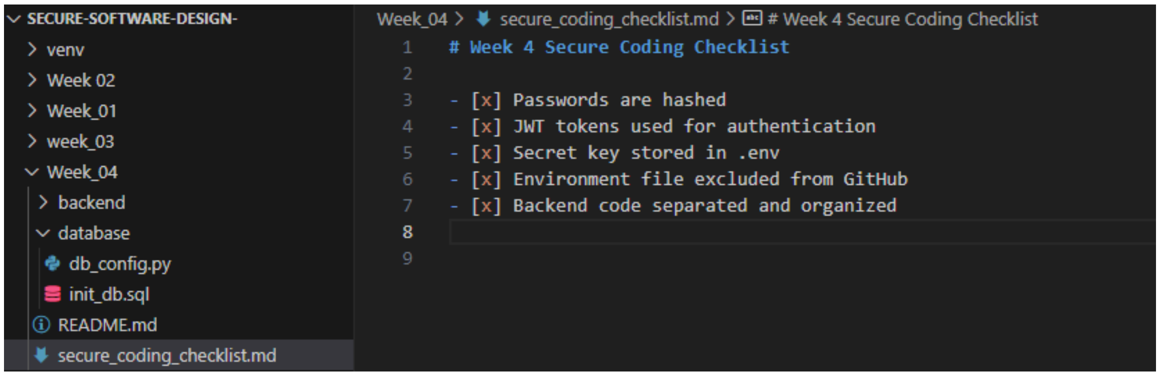


```
SECURE-SOFTWARE-D...
  venv
  Week 02
  Week_01
  week_03
  Week_04
    backend
    database
      db_config.py
      init_db.sql
    README.md
    secure_coding_checklist.md
  .gitignore
  README.md
  Week 2
```

6. Secure Coding Practices Followed

A checklist of secure coding practices was created and followed during implementation. These include:

- Hashed passwords (bcrypt)
- Token-based authentication (JWT)
- Parameterized database queries
- Separation of configuration using .env
- Code modularization and reuse



```
SECURE-SOFTWARE-DESIGN-
  venv
  Week 02
  Week_01
  week_03
  Week_04
    backend
    database
      db_config.py
      init_db.sql
    README.md
    secure_coding_checklist.md
```

Week_04 > secure_coding_checklist.md > # Week 4 Secure Coding Checklist

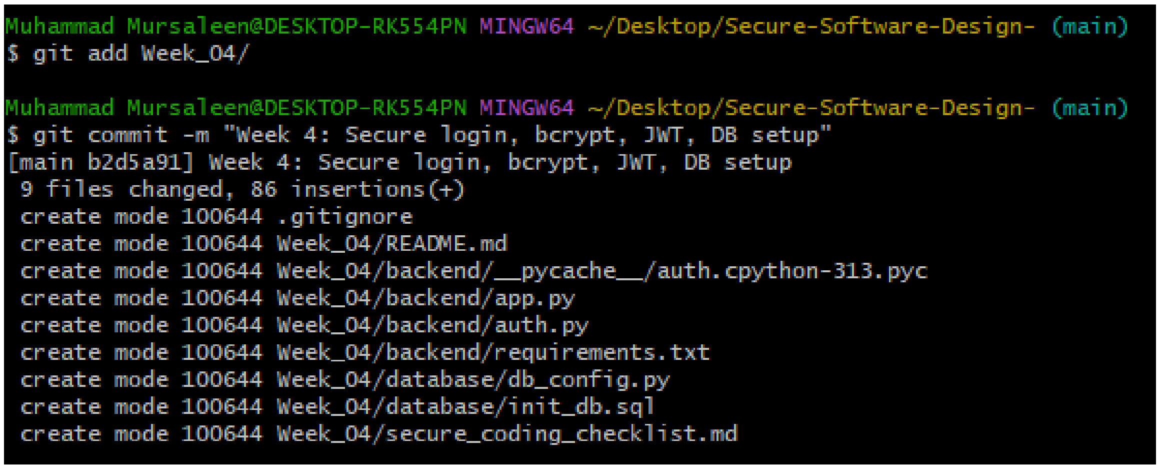
```
# Week 4 Secure Coding Checklist

- [x] Passwords are hashed
- [x] JWT tokens used for authentication
- [x] Secret key stored in .env
- [x] Environment file excluded from GitHub
- [x] Backend code separated and organized
```

7. GitHub Submission & Version Control

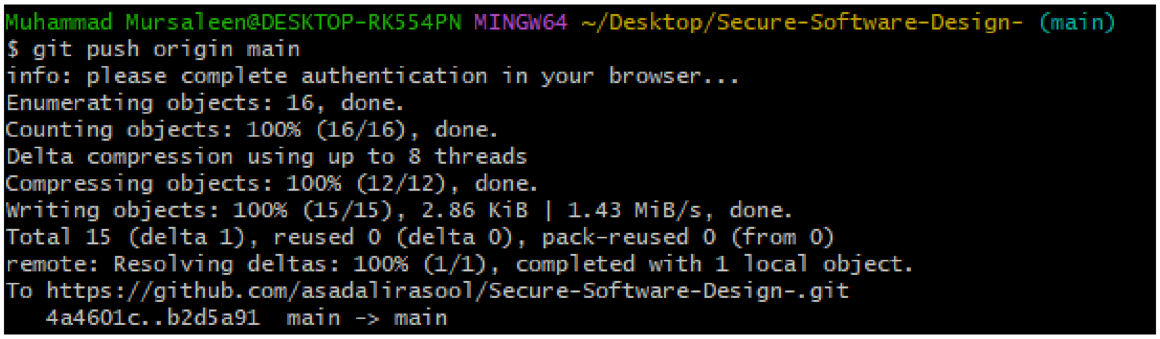
Progress was pushed to GitHub under the `Week4/` folder. The following were ensured:

- Code is modular and easy to understand
- Sensitive files are excluded using .gitignore
- A README.md file describes the work done in Week 4



```
Muhammad Mursaleen@DESKTOP-RK554PN MINGW64 ~/Desktop/Secure-Software-Design- (main)
$ git add Week_04/

Muhammad Mursaleen@DESKTOP-RK554PN MINGW64 ~/Desktop/Secure-Software-Design- (main)
$ git commit -m "Week 4: Secure login, bcrypt, JWT, DB setup"
[main b2d5a91] Week 4: Secure login, bcrypt, JWT, DB setup
 9 files changed, 86 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Week_04/README.md
 create mode 100644 Week_04/backend/__pycache__/auth.cpython-313.pyc
 create mode 100644 Week_04/backend/app.py
 create mode 100644 Week_04/backend/auth.py
 create mode 100644 Week_04/backend/requirements.txt
 create mode 100644 Week_04/database/db_config.py
 create mode 100644 Week_04/database/init_db.sql
 create mode 100644 Week_04/secure_coding_checklist.md
```



```
Muhammad Mursaleen@DESKTOP-RK554PN MINGW64 ~/Desktop/Secure-Software-Design- (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (15/15), 2.86 KiB | 1.43 MiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/asadalirasool/Secure-Software-Design-.git
   4a4601c..b2d5a91  main -> main
```

8. Summary and Next Steps

In Week 04, we started our secure implementation phase. Authentication and database handling were developed using secure practices. The next step will be to add input validation, encryption for password vault data, and build out more features with secure integration.