Farhan Sadiq

1001859500

**PROJECT REPORT**

In this project, I have worked on a program which is a REST API application built using FastAPI, a modern web framework for building APIs with Python. It maintains a phone book of names and phone numbers and provides various API endpoints to interact with the phone book data.

* I have used:
* SQLAlchemy to interact with an SQLite database for data persistence.
* Pydantic models for request and response data validation.
* Regular expressions are employed for input validation of names and phone numbers.
* API key-based authentication is implemented using FastAPI's Security dependency.
* Working mechanism:
* The code initializes an SQLite database using SQLAlchemy.
* Regular expressions are used to validate the format of names and phone numbers provided in the API requests.
* The code configures logging to a file named "phonebook\_audit.log". It logs actions such as adding or deleting a person from the phonebook.
* The application implements API key-based authentication and all the required API endpoints (list, add, deletebyname, deletebynumber)
* To run the program, we first need to ensure that we have python3, docker and all the required dependencies downloaded and installed.
* Install the required packages for python3, if not already installed: **pip3 install -r requirements.txt**
* Build and run using docker:
* Navigate to the directory called phonebook, which contains all the files.
* Execute this command on the terminal to build the Docker image: **docker build -t phonebook-api .**
* Execute this command on the terminal to run the Docker container: **docker run -d -p 8081:8081 --name phonebook-container phonebook-api**
* Open your web browser and search for: [**http://localhost:8081/docs**](http://localhost:8081/docs)
* You should be able to see the Swagger UI interface allowing you to interact with the API endpoints.
* The program sometimes run into an issue as the database does not connect to the server. But when I run the program without the docker, it works fine.
* Navigate to the directory called phonebook, which contains all the files.
* To run the program, execute the command: **uvicorn main:app --host 0.0.0.0 --port 8080 --reload**
* Note: If the port mentioned is already in use, we will encounter an error. In this case, changing the port number fix the issue. I ran into this error several times.
* Open your web browser and search for: [**http://localhost:8080/docs**](http://localhost:8080/docs)
* You should be able to see the Swagger UI interface allowing you to interact with the API endpoints.
* There is a file for the unit tests as well named test\_phonebook.py. I have written 3 tests for my program which passed when I ran it, however it does provide a few warnings which is negligible.
* To run the test file, simply execute this command on the terminal: **pytest test\_phonebook.py**

**Assumptions Made:**

* I have hardcoded the API key for simplicity.
* The database connection is assumed to be SQLite.

**Pros/Cons:**

Pros:

* API key-based authentication makes it secure, preventing unauthorized access.
* Action logging provides us with an audit trail for tracking changes made to the phonebook.

Cons:

* The code does not contain good error handling mechanisms.
* SQLite may not be suitable for large scale applications.

Here are the screenshots of my program after execution:

1. After running the commands on terminal and searching on the web browser:

**A screenshot of a computer

Description automatically generated**

1. Authorization:

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. Add Operation:

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

1. List operation:

**A screenshot of a computer

Description automatically generated**

1. DEL by name operation:

**A screenshot of a computer

Description automatically generated**

1. DEL by number operation:

**A screenshot of a computer

Description automatically generated**