1. Project Overview:

The project has been developed entirely in Python, using the FastAPI framework, which is a contemporary tool designed specifically for creating REST APIs in Python. FastAPI boasts several crucial attributes, including high performance, minimal bugs, reliability, ease of use, and learnability. To organize the project, it has been divided into separate modules, such as schemas, services, and models. All endpoints specified in the document are included in the project. The schema module includes a phonebook schema, as well as additional fields like record_date_created and record_date_updated, solely for tracking purposes.

The project's database is SQLite. Initially, the project connects to the database to establish a session that is used in all methods to interact with the database, allowing the creation, reading, and deletion of records through the created endpoints. The get_users method retrieves a list of users from the database, while the create_phonebookuser method creates a phonebook record, validating the name and phone number using the validate(name) and validate_number(phone_number) methods. These methods use multiple regexes to handle both acceptable and unacceptable inputs outlined in the document. Additionally, the status codes 200, 404, and 400 are considered based on user inputs in the method. The delete_user_by_phone and delete_user_by_number methods delete records based on name and phone number, respectively, while also handling validations and returning appropriate error messages and status codes.

SQLAlchemy is used for object-relational mapping to the database, and all utility functions for CRUD operations are in the services layer. The project also implements logging functionality using Python's logging module, and all log traces are recorded in the apptraces.log file. A testing module is also present in the project for performing unit testing on various inputs, creating a dummy database for testing instead of using the production database.

2. Instructions to setup Environment:

1. Download folder and extract the zip file Input_Validation_1001911486. This folder contains the readme and project files.

2. Go to ‘code’ folder using ‘cd code’ command.

3. Open command line prompt/ terminal inside ‘code’ folder. Users must have docker installed. This is prerequisite.

4. Type/ copy below command inside quotes as it is:

   “docker build -t secured-project .” (Note: **There is period/ dot at the end of command**)

3. Project Execution:

1. Type/copy below command inside quotes to run docker image built in last section:
   "docker run -d --name cont -p 80:80 secured-project"

2. The server would start on the given URL:
   http://127.0.0.1:80

3. Endpoints of the application:

   http://127.0.0.1:80/PhoneBook/list -- GET Request to list all entries.

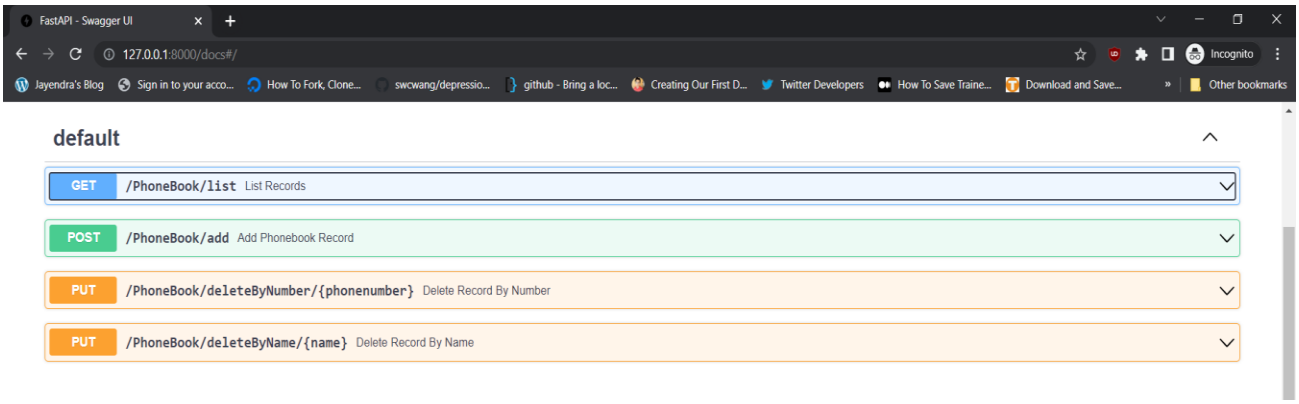
   http://127.0.0.1:80/PhoneBook/add -- POST Request to add new person.

   http://127.0.1.80/PhoneBook/deleteByName/{name} – PUT Request to delete person by name.

   http://127.0.0.1:80/PhoneBook/deleteByNumber/{phonenumber} – PUT Request to delete person by number.

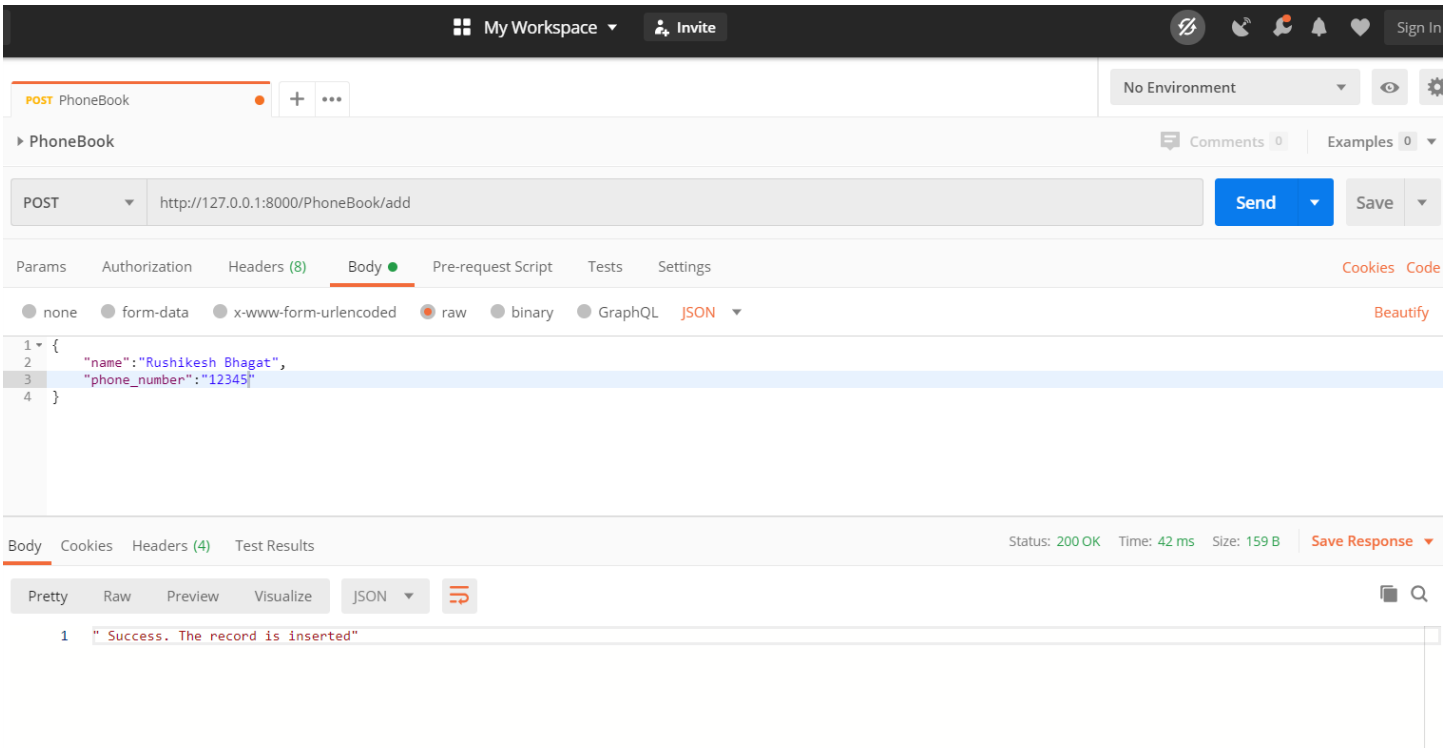
4. Fastapi provides UI to test endpoints. Enter below url to use it:

   http://127.0.0.1:8000/docs



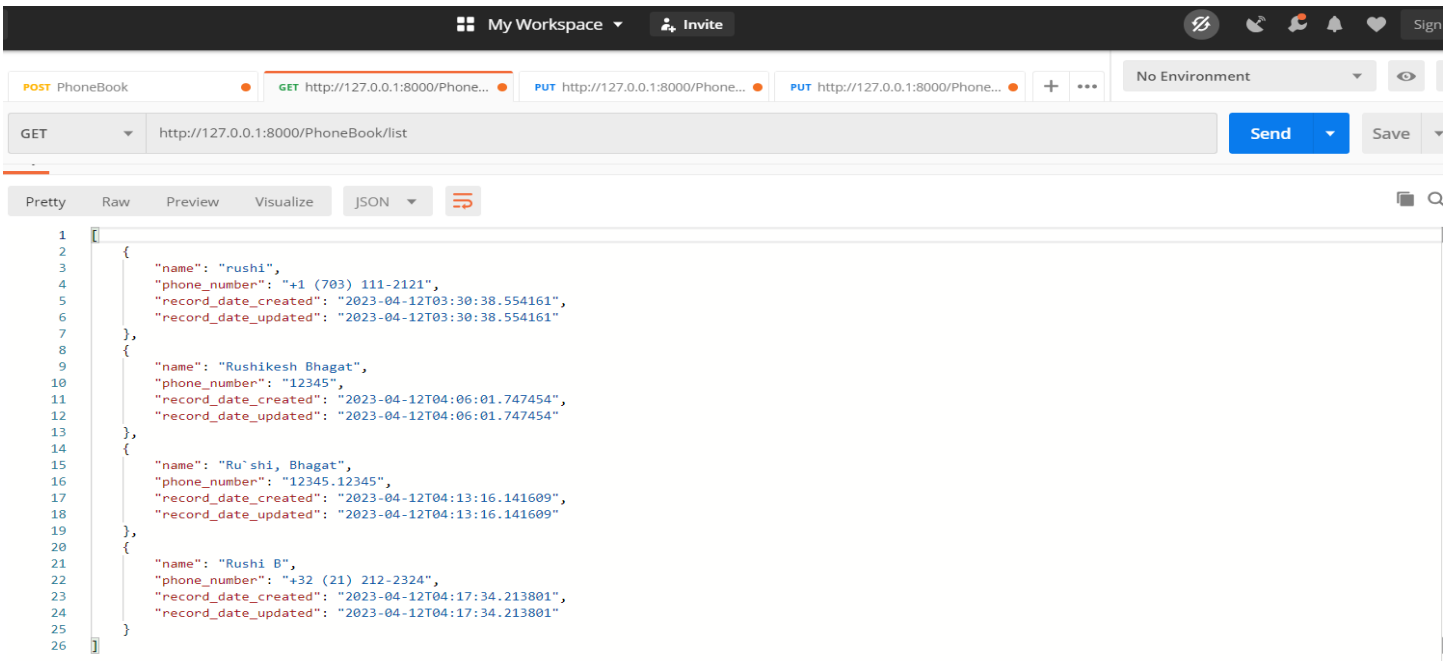
5. To test app on Postman, use below end point urls:

   a) Create a new HTTP POST request for the below url:
   http://127.0.0.1:80/PhoneBook/add
   Click on Body> raw, select JSON form drop down and click on send.



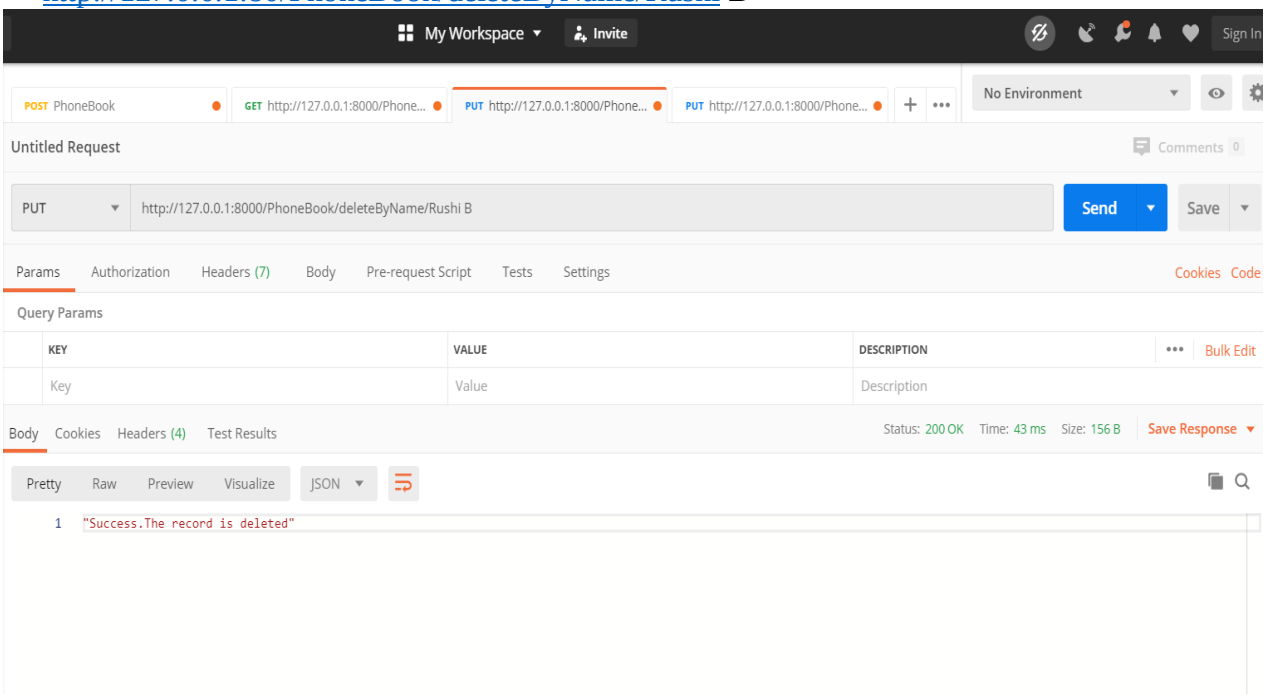
b) Create a new HTTP GET request for the below url:
http://127.0.0.1:80/PhoneBook/list

Click on Body and click on Send.



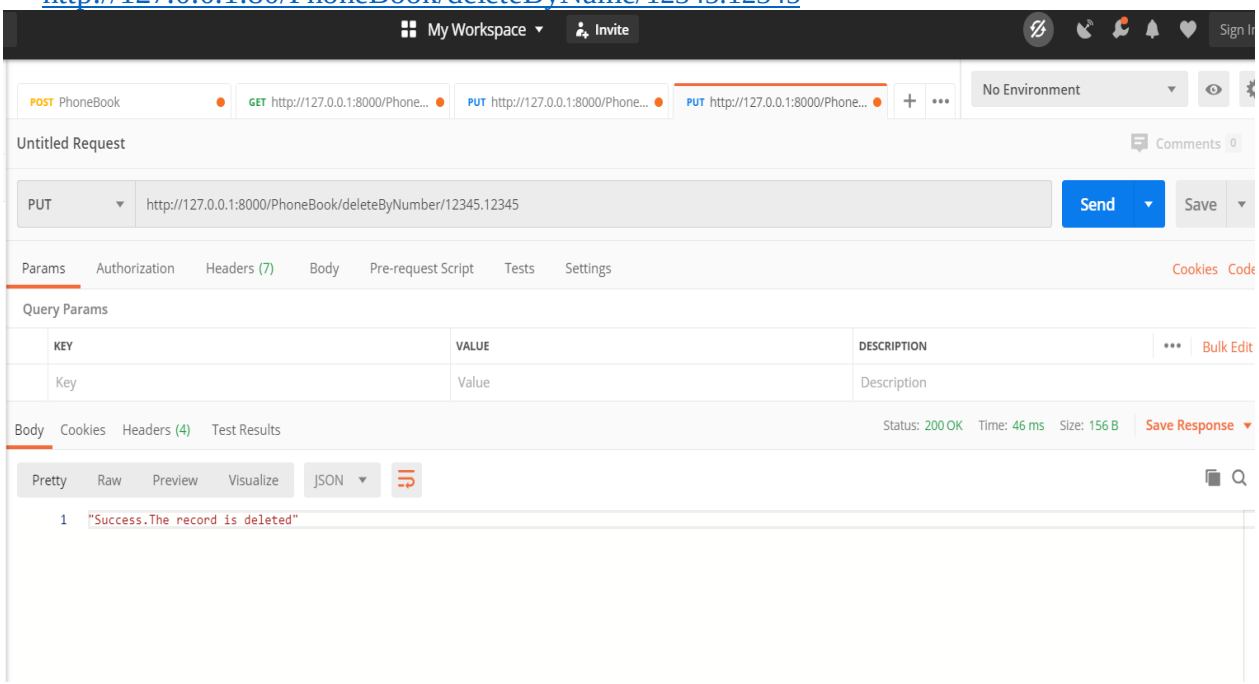
c) Create a new HTTP PUT request and pass the name to be deleted from the database as described in the below url:

http://127.0.0.1:80/PhoneBook/deleteByName/Rushi B



d) Create a new HTTP PUT request and pass the phone number to be deleted from the database as described in the below url:

http://127.0.0.1:80/PhoneBook/deleteByName/12345.12345



6. Unit test cases are written under **Input_Validation/tests** folder.
   To run them:
   Inside cmd/ terminal run the below command in cmd/ terminal.
   "docker exec -it cont pytest -v" (Note: Here 'cont' is container name)

7. To view audit log, run below command:
   "docker exec -it cont cat auditlog.log"

4. Assumptions:

Must have knowledge of Docker and installed Docker on Operating System. The endpoint for creating a user validates acceptable phone number and name using multiple regexes. Data is only inserted into the database if both validations succeed. If they fail, the code returns a 404-status code with an invalid input response. Additionally, the methods check duplicates of both user and phone number to avoid redundancy. While deleting users, similar logic is used for names or phone numbers i.e., if either does not exist it would give an error response. While the regex used in the project is assumed to be able to handle all major validations required for phone numbers internationally, there may be limitations. For running unit test cases, a dummy database is used instead of a live production database.

5. Pros / Cons:

Pros:

1. The FastAPI framework used for developing the Rest API is specifically used for data validation. Therefore, it delivers high performance, and it provides good UI for debugging.
2. The given project protects us from SQL injection attacks as we are validating user input against potentially vulnerable values.
3. Since FastAPI is solely used for testing and validation, it significantly reduces development time and development costs.

Cons:

1. A relational database is very effective against complex queries. NoSQL platforms like MongoDB perform better than RDBMS.
2. There is scope for improvement in phone number validation as patterns were matched according to this assignment. Better regular expression could be implemented than existing one.