**INPUT VALIDATION**

1. **Working of the Code:**

The whole project is built using Python and utilized the framework called FastAPI. FastAPI is a modern framework for building Rest API’s in Python. It has many key features such as high performance, fewer bugs, robust , easy to use and learn. The project is divided into different modules such as schemas, services and models. The project includes all the endpoints specified by the document. The schema module contains schema for the phonebook, and it also has fields such as record\_date\_created and record\_date\_updated just for tracking purposes.

The database used in the project is **SQLite**. The first step in the project involves connecting to the database and establishing connection with it. Once the connection is made a session is created and used in all the methods i.e., in creating endpoints to interact with the database i.e., to perform create, read, and delete operations. The **get\_users** method is used to get the list of users from the database. The **create\_phonebookuser** method is used to create a record for the phonebook i.e., the name and the phone number is validated by the **validate(name)** and **validate\_number(phone\_number)** . Those two methods internally use multiple regexes to handle the acceptable and unacceptable inputs provided in the document. Furthermore, the status codes such as 200, 404 and 400 are taken into consideration based on the inputs provided by the user in the method. The **delete\_user\_by\_phone** and **delete\_user\_by\_number** is to delete the records from the database based on name and phone number. In addition to that all the validations are handled in all the methods and appropriate error message along the status code is returned . I have used **SQLAlchemy** to achieve object relational mapping to the database. All the utility functions to perform the CRUD operations are present in services layer. In the project , I have also implemented the logging functionality using logging module from Python and all the log traces can be found under **apptraces.log file**.

The project also contains the testing module as well to perform unit testing for various inputs which creates a dummy database for testing instead of using the production database.

1. **Compilation/build instructions:**
2. Download the folder and extract the zip file Assignment\_11\_1002026832. This folder contains the report and project folder (Input\_Validation).
3. To Open a terminal :

In windows:

 Click inside the search box from the taskbar and type *“terminal”* or *“windows terminal.”* Then, click or tap the Windows Terminal search result. You can also click or tap the Open option on the right. To see the “Run as *administrator”* option, first press on the arrow pointing downwards.

In mac :

* 1. If you have the **Spotlight Search button** in your menu bar, click it. Otherwise, you can use the keyboard shortcut **Command + Space**.
  2. Type in “Terminal.”
  3. You should see the Terminal application under **Top Hit** at the top of your results. **Double-click it** and Terminal will open.

1. This step involves installing Python and version is 3.9.7 .

On Windows :

1. Open any web browser and go to <https://python.org/>:
2. Click on the **Download > Latest Python 3.9.7** link.
3. A 64-bit machine should download the file labeled as Windows x86-64 executable installer and place it on your Desktop.
4. Double-click on the file which was downloaded recently.
5. **Add Python 3.9.7 to PATH** option
6. The installer checks for administrative privileges or for confirmations.
7. Press **Yes**.
8. click on the **Disable path length limit** option, before closing the installer.
9. You can verify if Python is installed on the machine by opening the terminal

and entering the below command.

python --version

On Mac:

* + 1. Download the installer package from [Python's official website](https://www.python.org/downloads/) for MacOS for the version mentioned .
    2. Steps are similar to Windows as mentioned above .

1. Install pip using the below curl command in the terminal (Windows):

curl [https://bootstrap.pypa.io/get-pip.py -o get-pip.py](https://bootstrap.pypa.io/get-pip.py%20-o%20get-pip.py)

Next run –

python get-pip.py

Pip is installed on Mac machines generally by default.

1. Navigate to the project directory using cd.
2. Then once in the project directory install all the required packages using the below command.

pip install -r requirements.txt

1. **Installation, setup, and execution instructions:**

1. Navigate to the project directory using cd command

For ex : /Users/rav\_1797/Documents/Input\_Validation

1. Once all the requirements are installed.
2. Use the below command to start the server in the terminal:

**uvicorn main:app --reload**

1. The server would start and below is the URL.

<http://127.0.0.1:8000>

1. The various endpoints of the application are:

<http://127.0.0.1:8000/PhoneBook/add> -- It’s a POST Request

<http://127.0.0.1:8000/PhoneBook/list> -- It’s a GET Request

[http://127.0.1.8000/PhoneBook/deleteByName/{name}](http://127.0.1.8000/PhoneBook/deleteByName/%7bname%7d) – It’s a PUT Request

[http://127.0.0.1:8000/PhoneBook/deleteByNumber/{phonenumber}](http://127.0.0.1:8000/PhoneBook/deleteByNumber/%7bphonenumber%7d) – It’s a PUT Request

1. The Fastapi provides a Swagger like UI to test the API endpoints and we can access it using the below url:

<http://127.0.0.1:8000/docs>

**Graphical user interface, text, application, email

Description automatically generated**

1. To test the on Postman you can use the below end point urls:

a) Create a new HTTP request and then select POST request for the below url:

<http://127.0.0.1:8000/PhoneBook/add>

Click on Body and raw select as JSON form drop down as shown below and pass the data.

Graphical user interface, text, application

Description automatically generated

b) Create a new HTTP request and then select POST request for the below url:

<http://127.0.0.1:8000/PhoneBook/list>

Click on Body and select as None and press as Send.

Text

Description automatically generated

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

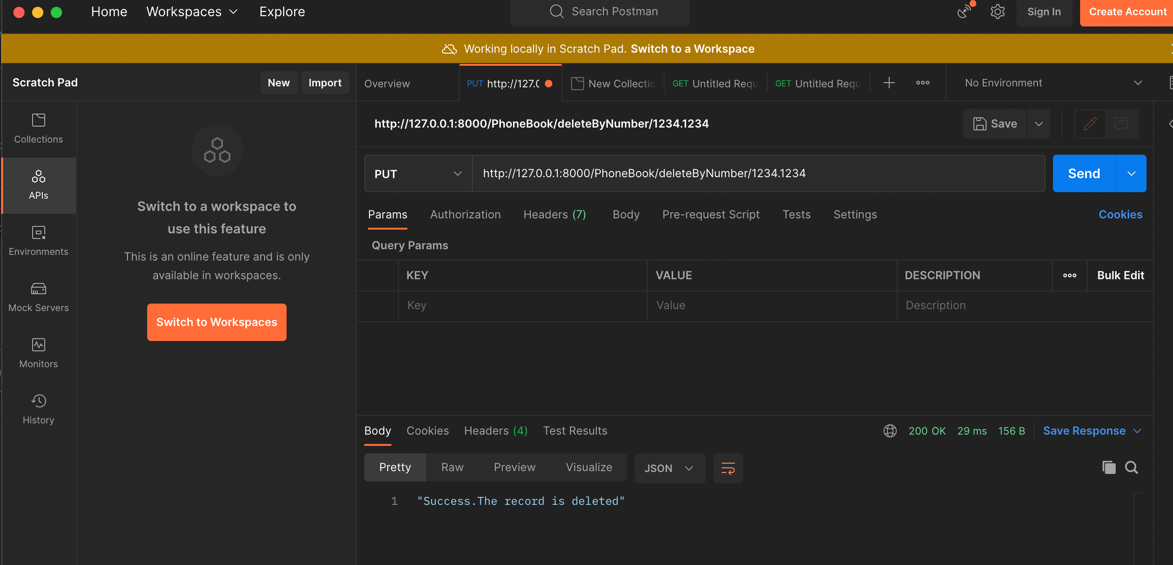
<http://127.0.1.8000/PhoneBook/deleteByName/Kohli>

Graphical user interface, application

Description automatically generated

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

<http://127.0.1.8000/PhoneBook/deleteByName/1234.1234>



1. All the unit test cases are defined in under **Input\_Validation/tests** folder.

To run the test cases, close the existing terminal.

Open a new terminal and navigate to project folder using cd .

Run the below command in terminal.

**pytest -v**

1. **The project can also be executed using visual studio code:**

a) To download vs code <https://code.visualstudio.com/download> [choose specific to OS]

b) Steps to install for specific OS can be found here <https://www.toolsqa.com/blogs/install-visual-studio-code/>

c) Once the project is extracted and unzipped click on File and open Folder and select the Input\_Validation folder and click on New Terminal .

d) Next Follow steps mentioned from Step 3 of this section.

1. **Assumptions:**

The endpoint for creating a user is using multiple regexes to validate the acceptable phone number as mentioned in the document and the same procedure is followed to validate the acceptable phone number. The data is inserted to the database only if both the validations are successful otherwise the code return 404 status code and response as invalid inputs . Furthermore, this method has checkpoints for if user or the phone number already exists in the database. Similarly they are validations handled while deleting the user by name and by phone number. We also assume that the regex used in the project can handle all major validations required for all telephone number across the globe. To run the unit test cases we are creating a dummy database rather than running it on a production database.

1. **Pros and Cons:**

**Pros:**

* 1. The pros of the approach are the framework used for developing the Rest API is quite robust , has high performance , easy for debugging.
  2. It allows to use object relational mapping .
  3. Using object relational mapping one can write optimized sql queries
  4. It protects us from SQL injection attacks.
  5. It reduces development time and development costs.

**Cons:**

1. Use of object relational mapping fails against complex queries.
2. Use of object relational mapping tends to be slow.
3. The regex used in the project may not be efficient to validate all telephone numbers across the globe.