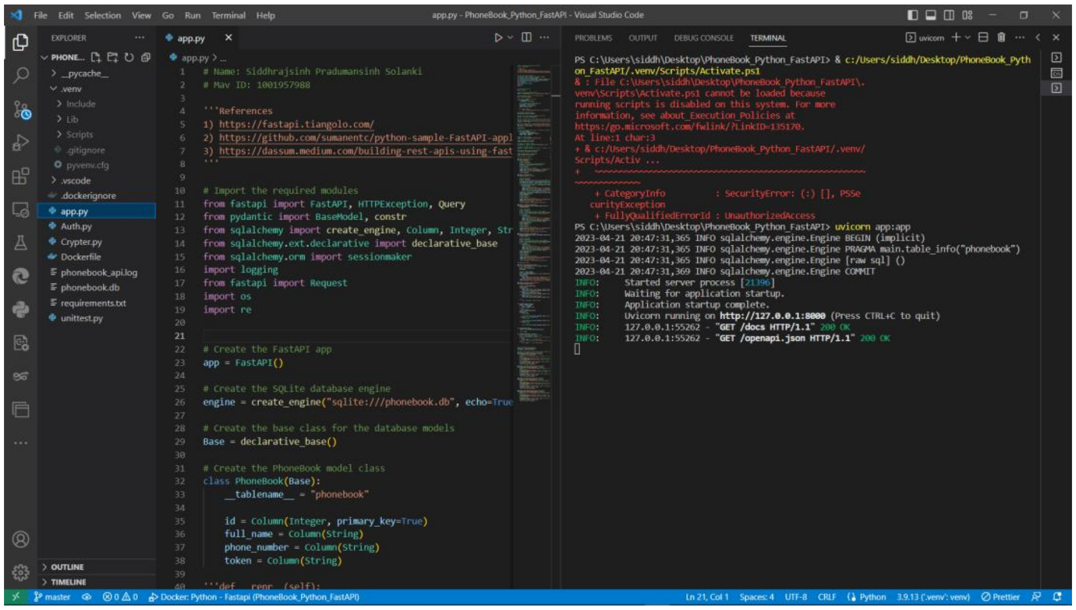
PHONEBOOK PROJECT REPORT

- **Description of project:**
  - This is a Python code for a RESTful API built with FastAPI that interacts with a SQLite database. The API allows users to add and retrieve and delete phonebook entries, which consist of a person's full name and phone number.
  - It uses Pydantic for data validation and SQLAlchemy as the ORM.
  - The application uses FastAPI, a Python web framework, for creating the REST API.
  - The SQLite database is used for storing the phone book data.
  - The application requires users to authenticate using a token before they can create, read, update, or delete contacts.
  - The phone number and full name fields have specific regex patterns to ensure that the data entered meets certain criteria.
  - The application logs events using the Python logging module and saves them to a file called phonebook_api.log.
  - It meets all project requirements about validation,authentication,log audit,Security Tests, Using database to store the input data, Parameterized queries and using Docker containers.

- **Stack Used:**
  - Python:Multipurpose programming language with rich library collection
  - fastapi: FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints.
  - uvicorn: Uvicorn is a lightning-fast ASGI server implementation, using uvloop and httptools.
  - sqlalchemy: SQLAlchemy is a popular SQL toolkit and ORM for Python. It provides a set of high-level API to work with relational databases.
  - pydantic: Pydantic is a data validation and settings management library, which uses Python type annotations to validate and parse data.
  - Postman: It is a collaboration platform for API development that allows users to design, test, and document APIs.

- **Instructions for building and running software and unit tests:**
  - **Running the code:**
    - Open Visual Studio.
    - Click on "File" in the top left corner and select "Open Folder".
    - Navigate to the folder where your Python code is located and select it.
    - If your code requires any dependencies, make sure they are installed in your Python environment.
    - Open the "Terminal" tab in Visual Studio by clicking on "View" and then "Terminal".
    - Install libraries by using command in terminal: **pip install -r requirements.txt**
    - In the terminal, navigate to the folder containing your Python code.

- In our case keep the app.py tab open.
- To run the app, type the command in terminal: **uvicorn app:app –reload**
- Following is output when you run code successfully.

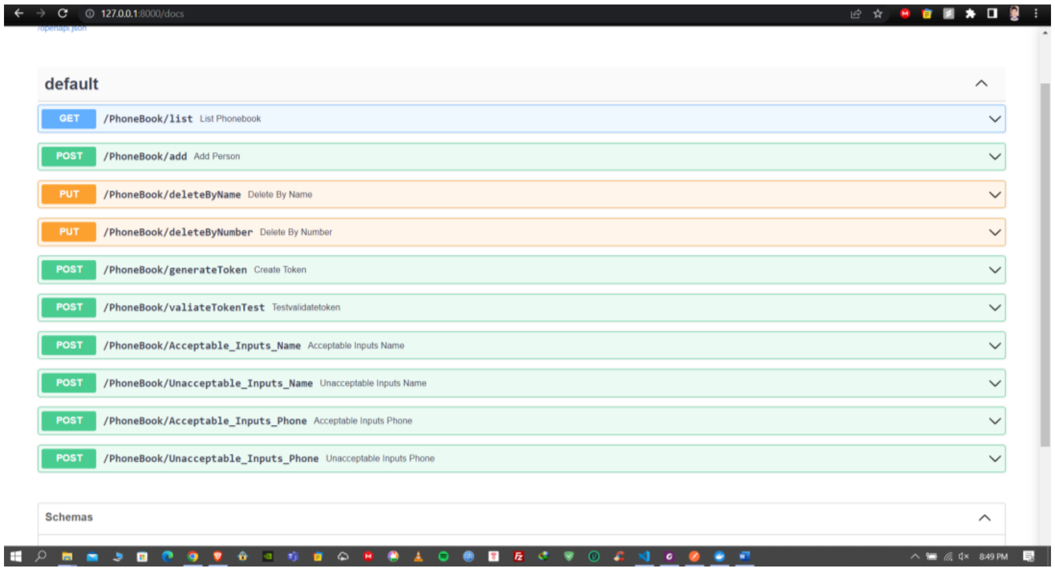


```python
# Name: Siddhrajsinh Pradumansinh Solanki
# Mav ID: 1001957988

'''References
1) https://fastapi.tiangolo.com/
2) https://github.com/sumanentc/python-sample-FastAPI-appl
3) https://dassum.medium.com/building-rest-apis-using-fast
'''

# Import the required modules
from fastapi import FastAPI, HTTPException, Query
from pydantic import BaseModel, constr
from sqlalchemy import create_engine, Column, Integer, Str
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import logging
from fastapi import Request
import os
import re


# Create the FastAPI app
app = FastAPI()

# Create the SQLite database engine
engine = create_engine("sqlite:///phonebook.db", echo=True

# Create the base class for the database models
Base = declarative_base()

# Create the PhoneBook model class
class PhoneBook(Base):
    __tablename__ = "phonebook"

    id = Column(Integer, primary_key=True)
    full_name = Column(String)
    phone_number = Column(String)
    token = Column(String)

'''def __repr__(self):
```

```
PS C:\Users\siddh\Desktop\PhoneBook_Python_FastAPI> & c:/Users/siddh/Desktop/PhoneBook_Pyth
on_FastAPI/.venv/Scripts/Activate.ps1
& : File C:\Users\siddh\Desktop\PhoneBook_Python_FastAPI\.
venv\Scripts\Activate.ps1 cannot be loaded because
running scripts is disabled on this system. For more
information, see about_Execution_Policies at
https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:3
+ & c:/Users/siddh/Desktop/PhoneBook_Python_FastAPI/.venv/
Scripts/Activ ...
+   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSe
   curityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\siddh\Desktop\PhoneBook_Python_FastAPI> uvicorn app:app
2023-04-21 20:47:31,365 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-21 20:47:31,365 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("phonebook")
2023-04-21 20:47:31,365 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-21 20:47:31,369 INFO sqlalchemy.engine.Engine COMMIT
INFO:     Started server process [21396]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:55262 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:55262 - "GET /openapi.json HTTP/1.1" 200 OK
```

- **Creating Docker image:**
  - Docker files are created and setup.
  - Build it using command: docker build -t phonebook .
  - Run image: docker run -p 8000:8000 phonebook
  - Once build and run finishes, browser window will open.
  - Navigate to: http://127.0.0.1:8000/docs

- **Testing the RESTful API:**
  - Once code runs ,we can test the API by using POSTMAN or by opening the browser and navigating to: http://127.0.0.1:8000/docs

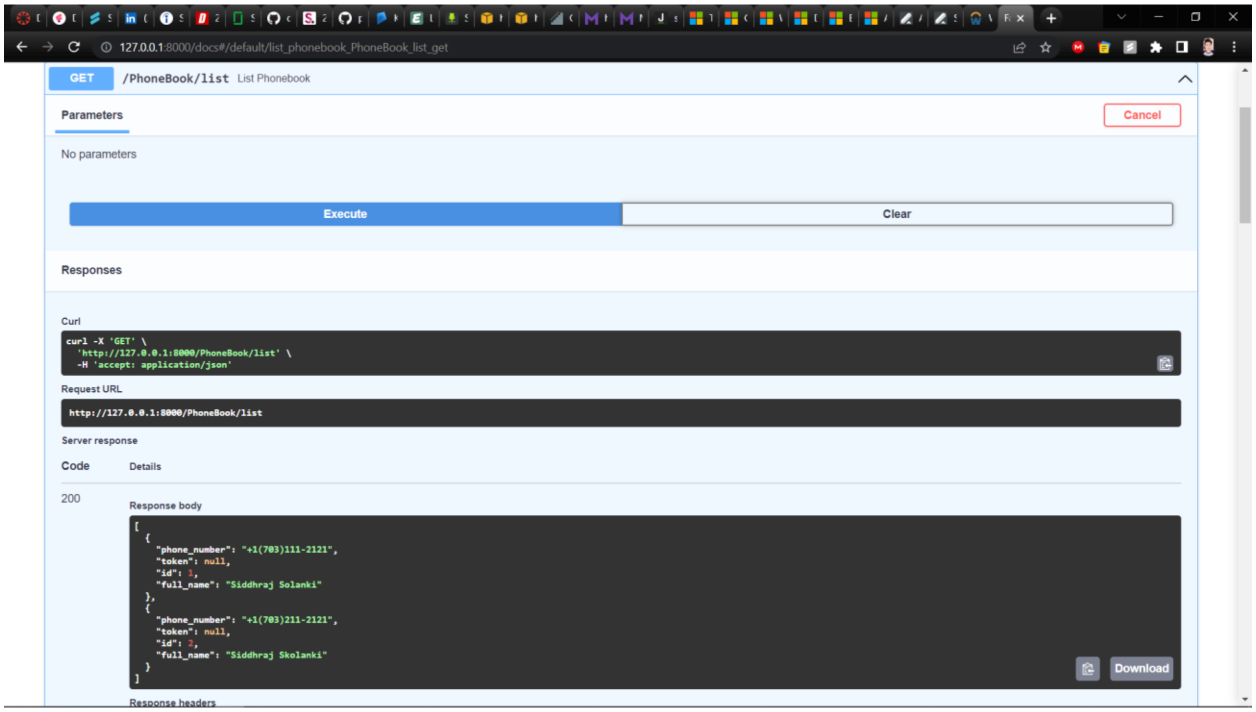

127.0.0.1:8000/docs

default

GET	/PhoneBook/list	List Phonebook
POST	/PhoneBook/add	Add Person
PUT	/PhoneBook/deleteByName	Delete By Name
PUT	/PhoneBook/deleteByNumber	Delete By Number
POST	/PhoneBook/generateToken	Create Token
POST	/PhoneBook/valiateTokenTest	Testvalidatetoken
POST	/PhoneBook/Acceptable_Inputs_Name	Acceptable Inputs Name
POST	/PhoneBook/Unacceptable_Inputs_Name	Unacceptable Inputs Name
POST	/PhoneBook/Acceptable_Inputs_Phone	Acceptable Inputs Phone
POST	/PhoneBook/Unacceptable_Inputs_Phone	Unacceptable Inputs Phone

Schemas

- Then navigate to: http://127.0.0.1:8000/docs#/default/add_person_PhoneBook_add_post For adding new person in database.If validation is successful person will be added to database, otherwise error will be thrown.
- Then navigate to: http://127.0.0.1:8000/docs#/default/list_phonebook_PhoneBook_list_get For fetching all records from database.


127.0.0.1:8000/docs#/default/list_phonebook_PhoneBook_list_get

GET /PhoneBook/list List Phonebook

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/PhoneBook/list' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/PhoneBook/list
```

Server response

Code Details

200

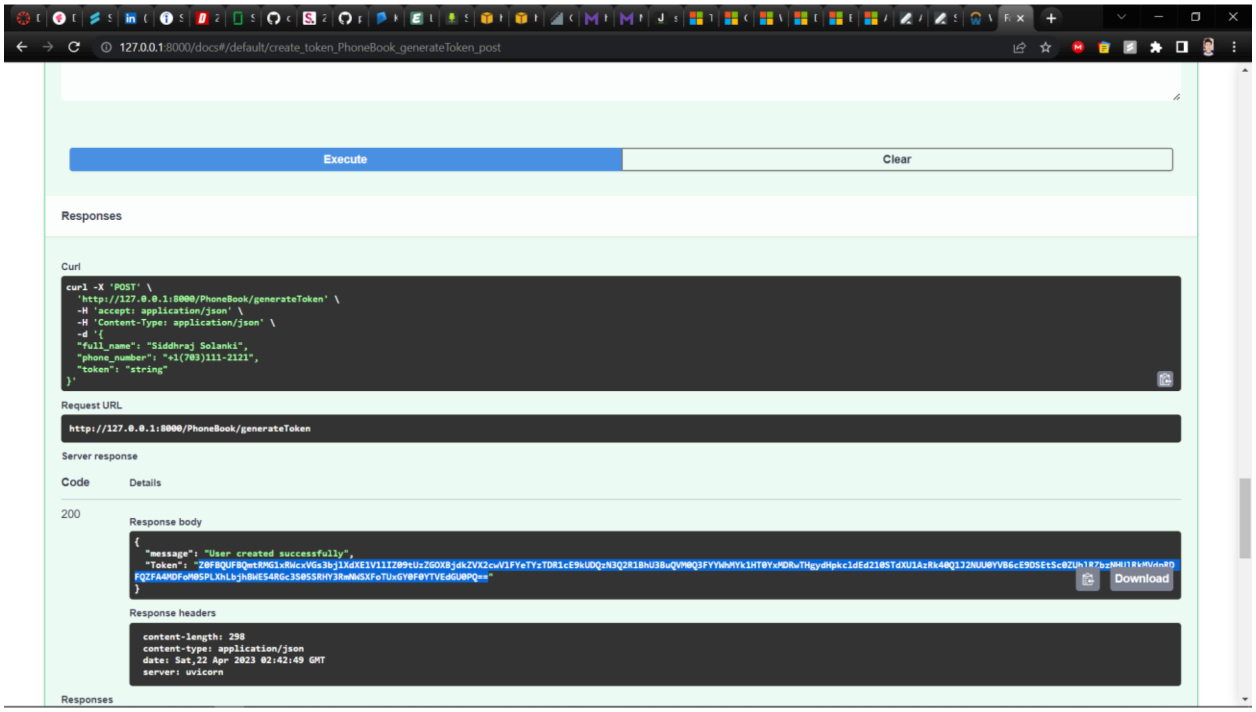
Response body

```
[
  {
    "phone_number": "+1(703)111-2121",
    "token": null,
    "id": 1,
    "full_name": "Siddhraj Solanki"
  },
  {
    "phone_number": "+1(703)211-2121",
    "token": null,
    "id": 2,
    "full_name": "Siddhraj Skolanki"
  }
]
```

Response headers

- For deleting the record ,we first need to generate token by going to : http://127.0.0.1:8000/docs#/default/create_token_PhoneBook_generateToken_post

- The token is used for both authentication and authorization. When a user logs in, the server generates a token, which is subsequently transmitted to the client. This token is then given to the server with each subsequent request, which verifies it to confirm that the request is coming from a trustworthy source.
- By entering correct full name and phone number already stored in database, a user can generate token , otherwise error is thrown.
- The token generation algorith is almost designed from scratch using some libraries for encoding and decoding .
- Token expire in 10 minutes, so user will need to generate token again to perform delete operations.


127.0.0.1:8000/docs#/default/create_token_PhoneBook_generateToken_post
Execute
Clear
Responses
Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/PhoneBook/generateToken' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "full_name": "Siddhraj Solanki",
  "phone_number": "+1(703)111-2121",
  "token": "string"
}'
```

Request URL

```
http://127.0.0.1:8000/PhoneBook/generateToken
```

Server response

Code Details

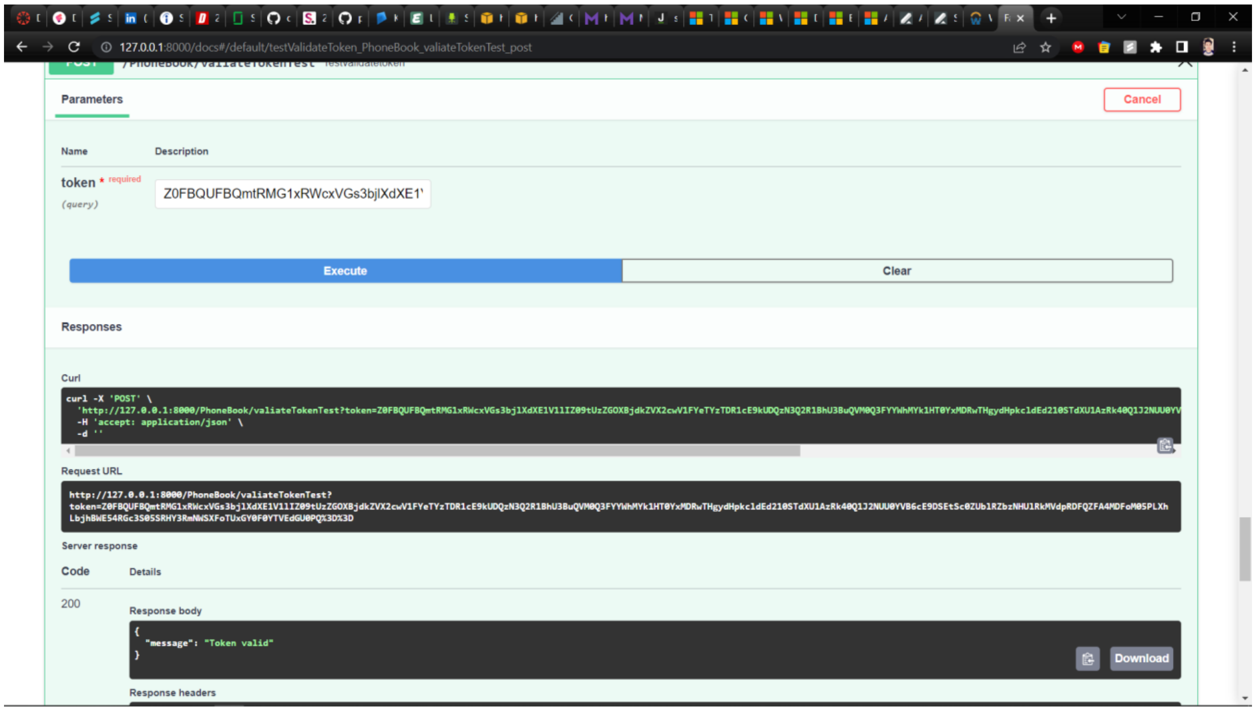
200 Response body

```
{
  "message": "User created successfully",
  "Token": "Z0FBQUFBQmtRMG1xRWcxVGs3bj1XdXE1V11IZ09tUzZGOXBjdkZVX2cwV1FYeTYzTDR1cE9kUDQzN3Q2R1BhU3BuQVM0Q3FYYWhMYk1HT0YxMDRwTHgydHpkcldEd210STdXU1AzRk40Q1J2NUU0YVB6cE9DSEtSc0ZUh1R7bzNHU1RkMVdpRD
FQZFA4MDFoM05PLXhLbjhBWE54RGc3S05SRHY3RmNWSXFoTUxGY0F0YTVEdGU0PQ=="
}
```

Response headers

```
content-length: 298
content-type: application/json
date: Sat,22 Apr 2023 02:42:49 GMT
server: uvicorn
```

- Copy the received token as it needs to be attached manually with each request using POSTMAN.
- You can check if token is valid or not going to : http://127.0.0.1:8000/docs#/default/testValidateToken_PhoneBook_valiateTokenTest_post



127.0.0.1:8000/docs#/default/testValidateToken_PhoneBook_valiateTokenTest_post

/PhoneBook/valiateTokenTest testvalidatetoken

token * required (query): Z0FBQUFBQmtRMG1xRWcxVGs3bjlXdXE1

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/PhoneBook/valiateTokenTest?token=Z0FBQUFBQmtRMG1xRWcxVGs3bjlXdXE1V1lIZ09tUzZGOXBjdkZVX2cwV1FYeTYzTDR1cE9kUDQzN3Q2R1BhU3BuQVM0Q3FYYWhMYk1HT0YxMDRwTHgydHpkcldEd210STdXU1AzRk40Q1J2NUU0YV
  -H 'accept: application/json' \
  -d ''
```

Request URL

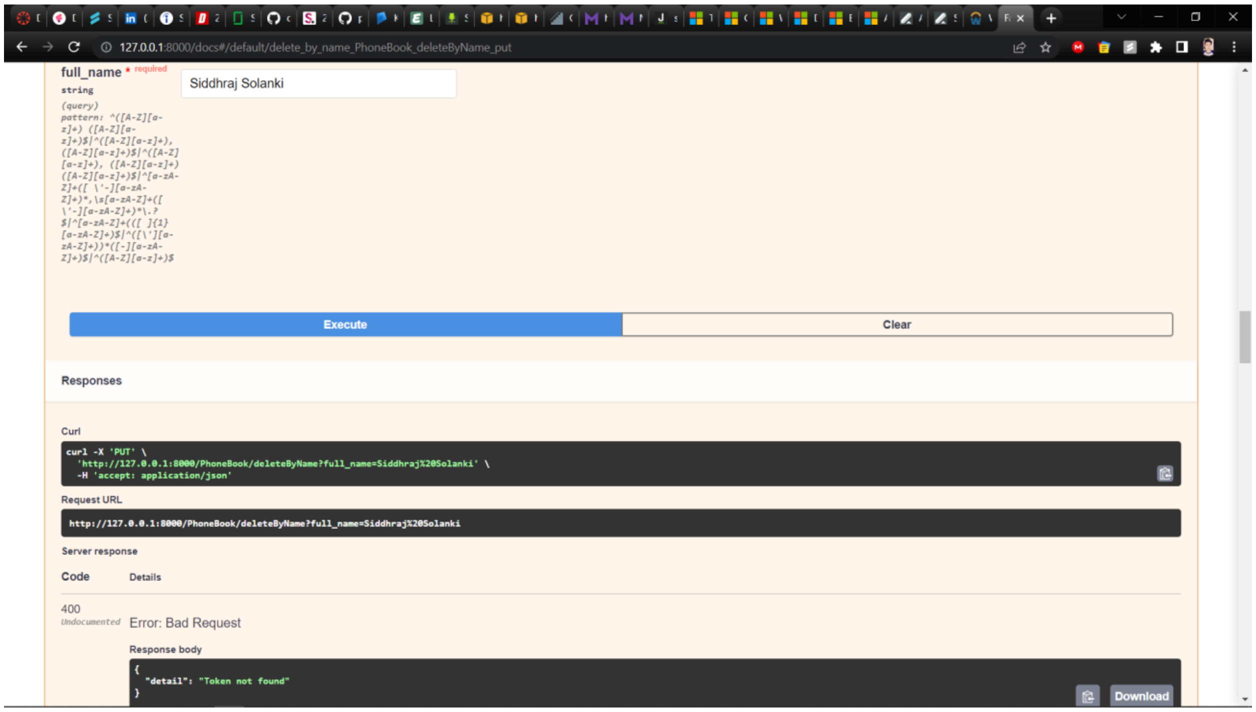
```
http://127.0.0.1:8000/PhoneBook/valiateTokenTest?
token=Z0FBQUFBQmtRMG1xRWcxVGs3bjlXdXE1V1lIZ09tUzZGOXBjdkZVX2cwV1FYeTYzTDR1cE9kUDQzN3Q2R1BhU3BuQVM0Q3FYYWhMYk1HT0YxMDRwTHgydHpkcldEd210STdXU1AzRk40Q1J2NUU0YVB6cE9DSEtSc0ZUblRZbzNHU1RkMVdpRDFQZFA4MDFoM05PLXh
LbjhBWE54RGc3S05SRHY3RmNWSXFoTUxGY0F0YTVEdGU0PQ%3D%3D
```

Server response

Code	Details
200	Response body

```
{
  "message": "Token valid"
}
```

- Then we can open POSTMAN and manually attach token in header to successfully delete the record.



127.0.0.1:8000/docs#/default/delete_by_name_PhoneBook_deleteByName_put

full_name * required: Siddhraj Solanki

string
(query)
pattern: ^([A-Z][a-z]+) ([A-Z][a-z]+)$|^([A-Z][a-z]+), ([A-Z][a-z]+)$|^([A-Z][a-z]+), ([A-Z][a-z]+) ([A-Z][a-z]+)$|^[a-zA-Z]+([\'-][a-zA-Z]+)*,\s[a-zA-Z]+([\'-][a-zA-Z]+)*\.?$|^[a-zA-Z]+(([ ]{1}[a-zA-Z]+)$|^([\'][a-zA-Z]+))*([-][a-zA-Z]+)$|^([A-Z][a-z]+)$

Curl

```
curl -X 'PUT' \
  'http://127.0.0.1:8000/PhoneBook/deleteByName?full_name=Siddhraj%20Solanki' \
  -H 'accept: application/json'
```

Request URL

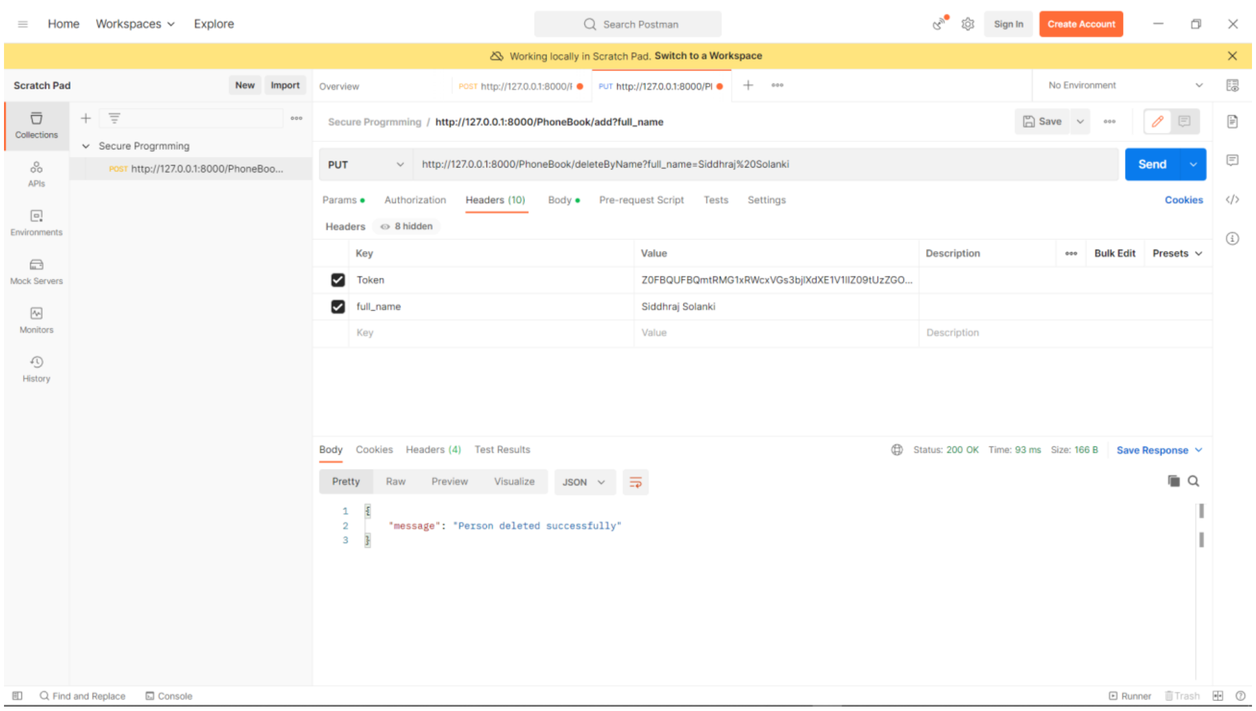
```
http://127.0.0.1:8000/PhoneBook/deleteByName?full_name=Siddhraj%20Solanki
```

Server response

Code	Details
400 Undocumented	Error: Bad Request Response body

```
{
  "detail": "Token not found"
}
```

- Open postman and navigate to url and fill in the data to delete record.



Secure Progrmming / http://127.0.0.1:8000/PhoneBook/add?full_name

PUT http://127.0.0.1:8000/PhoneBook/deleteByName?full_name=Siddhraj%20Solanki

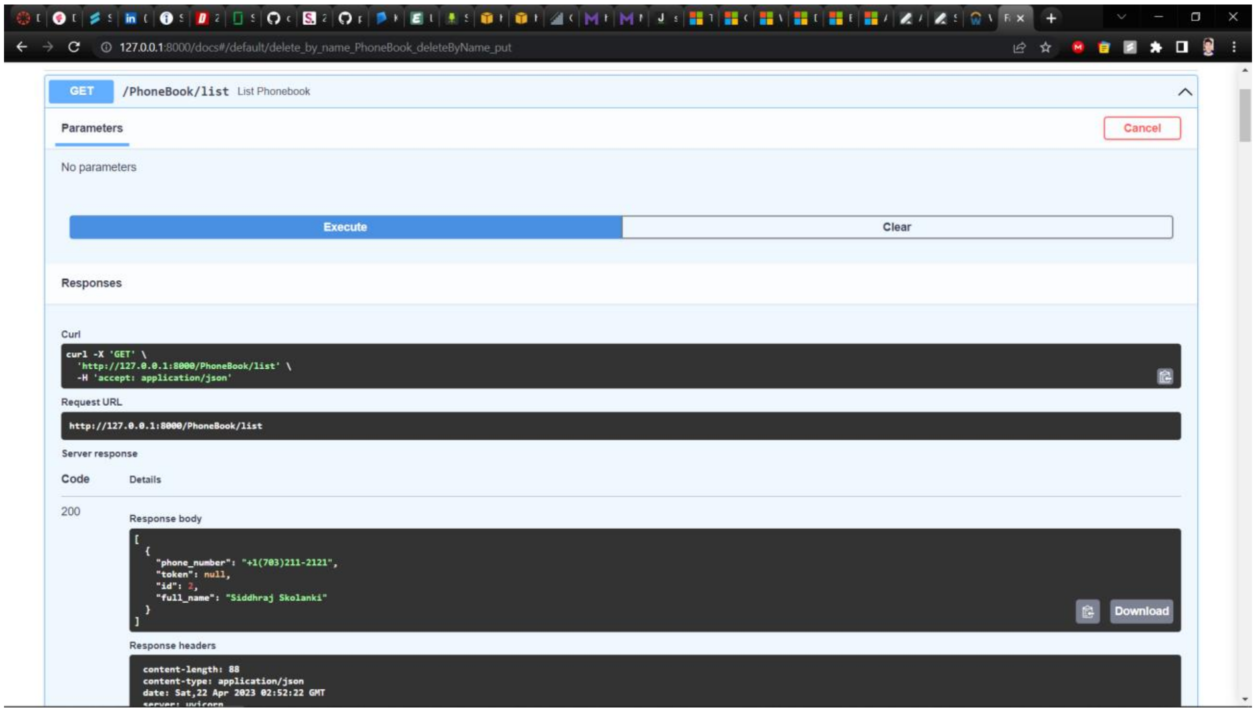
Headers (10)

	Key	Value	Description
✓	Token	Z0FBQUFBQmtRMG1xRWcxVGs3bjlXdXE1V1lIZ09tUzZGO...	
✓	full_name	Siddhraj Solanki	

Status: 200 OK Time: 93 ms Size: 166 B

```
{
    "message": "Person deleted successfully"
}
```

- Now the person will be deleted successfully when you enter full name and attach token in header.
- If now we fetch all records, we can see one entry is deleted.



127.0.0.1:8000/docs#/default/delete_by_name_PhoneBook_deleteByName_put

GET /PhoneBook/list List Phonebook

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/PhoneBook/list' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/PhoneBook/list
```

Server response

Code 200

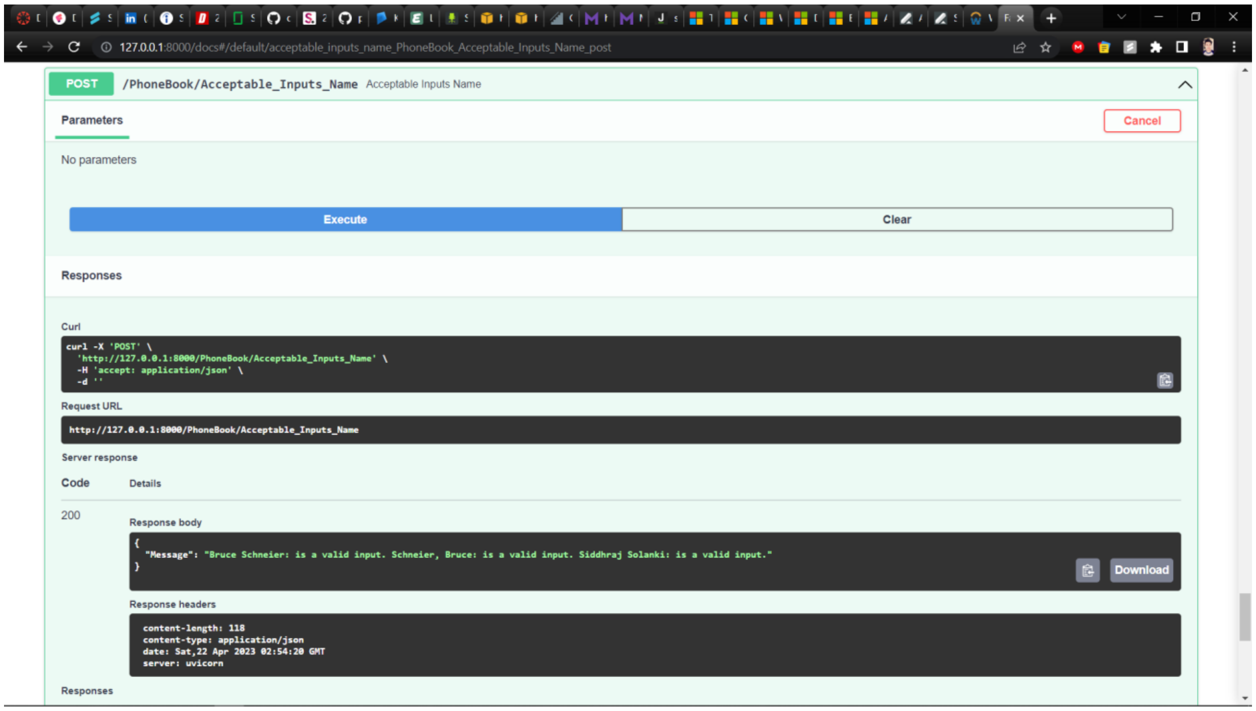
Response body

```
[
  {
    "phone_number": "+1(703)211-2121",
    "token": null,
    "id": 2,
    "full_name": "Siddhraj Skolanki"
  }
]
```

Response headers

```
content-length: 88
content-type: application/json
date: Sat,22 Apr 2023 02:52:22 GMT
```

- We can go throght the unit test by going to Acceptable_Inputs_Name, Unacceptable_Inputs_Name, Acceptable_Inputs_Phone or Unacceptable_Inputs_Phone.


POST /PhoneBook/Acceptable_Inputs_Name Acceptable Inputs Name

Parameters

No parameters

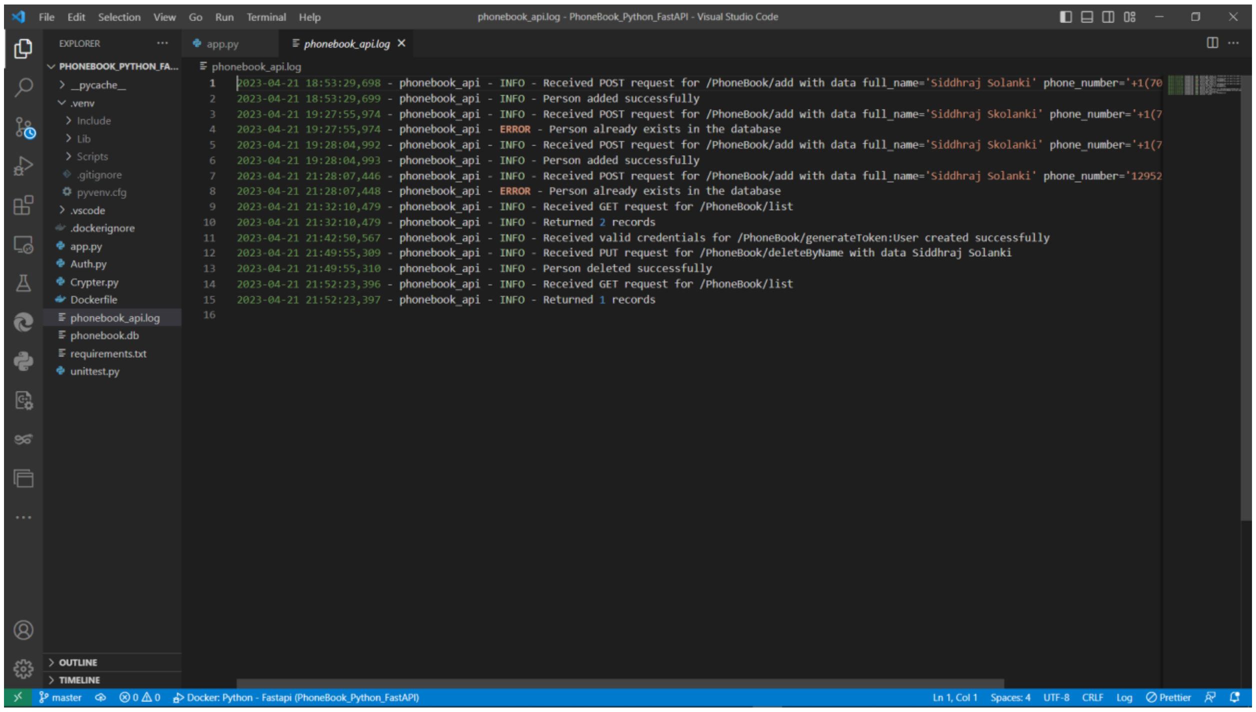
Responses

Curl
```
curl -X 'POST' \
  'http://127.0.0.1:8000/PhoneBook/Acceptable_Inputs_Name' \
  -H 'accept: application/json' \
  -d ''
```
Request URL
```
http://127.0.0.1:8000/PhoneBook/Acceptable_Inputs_Name
```
Server response

Code 200

Response body
```
{
  "Message": "Bruce Schneier: is a valid input. Schneier, Bruce: is a valid input. Siddhraj Solanki: is a valid input."
}
```
Response headers
```
 content-length: 118
 content-type: application/json
 date: Sat,22 Apr 2023 02:54:20 GMT
 server: uvicorn
```

- Logs are recorded in log file . In this code, logging is used to record events or actions that occur during the execution of the application.


phonebook_api.log - PhoneBook_Python_FastAPI - Visual Studio Code
```
2023-04-21 18:53:29,698 - phonebook_api - INFO - Received POST request for /PhoneBook/add with data full_name='Siddhraj Solanki' phone_number='+1(70
2023-04-21 18:53:29,699 - phonebook_api - INFO - Person added successfully
2023-04-21 19:27:55,974 - phonebook_api - INFO - Received POST request for /PhoneBook/add with data full_name='Siddhraj Skolanki' phone_number='+1(7
2023-04-21 19:27:55,974 - phonebook_api - ERROR - Person already exists in the database
2023-04-21 19:28:04,992 - phonebook_api - INFO - Received POST request for /PhoneBook/add with data full_name='Siddhraj Skolanki' phone_number='+1(7
2023-04-21 19:28:04,993 - phonebook_api - INFO - Person added successfully
2023-04-21 21:28:07,446 - phonebook_api - INFO - Received POST request for /PhoneBook/add with data full_name='Siddhraj Solanki' phone_number='12952
2023-04-21 21:28:07,448 - phonebook_api - ERROR - Person already exists in the database
2023-04-21 21:32:10,479 - phonebook_api - INFO - Received GET request for /PhoneBook/list
2023-04-21 21:32:10,479 - phonebook_api - INFO - Returned 2 records
2023-04-21 21:42:50,567 - phonebook_api - INFO - Received valid credentials for /PhoneBook/generateToken:User created successfully
2023-04-21 21:49:55,309 - phonebook_api - INFO - Received PUT request for /PhoneBook/deleteByName with data Siddhraj Solanki
2023-04-21 21:49:55,310 - phonebook_api - INFO - Person deleted successfully
2023-04-21 21:52:23,396 - phonebook_api - INFO - Received GET request for /PhoneBook/list
2023-04-21 21:52:23,397 - phonebook_api - INFO - Returned 1 records
```

- ## Assumptions :
  - Acceptable inputs are assumed to be correct .

    - The SQLite database is used for storing the phone book data.
    - The application requires users to authenticate using a token before they can delete records.
    - The phone number and full name fields have specific regex patterns to ensure that the data entered meets certain criteria.
    - User enters correct information while deleting records by name or number.

- **Pros and Cons of Approach:**
    - Pros:
        - FastAPI is a lightweight web framework for creating Python APIs.
        - SQLite is a small, simple relational database that may be readily integrated into a Python script.
        - Token authentication is a safe method that provides for granular access control.
        - This approach has the potential to provide a scalable and maintainable solution for developing a RESTful API.
        - FastAPI has strong type-checking and validation, making it easy to catch errors early in the development process.
    - Cons:
        - SQLite is not appropriate for high-traffic or large-scale applications since it can become slow when dealing with massive datasets.
        - Token-based authentication can complicate the authentication process and may necessitate additional development work.
        - Because FastAPI is a newer framework, it may not have as much community support or documentation as other web frameworks.
        - Asynchronous programming can be more difficult to understand and debug for some developers who are not familiar with it.
        - FastAPI's automatic API documentation generation can be limiting in terms of customization, and may not meet the needs of more complex projects.

- **References:**

    - https://fastapi.tiangolo.com/
    - https://github.com/sumanentc/python-sample-FastAPI-application
    - https://dassum.medium.com/building-rest-apis-using-fastapi-sqlalchemy-uvicorn-8a163ccf3aa1
    - *https://blog.logrocket.com/using-fastapi-inside-docker-containers/*
    - *https://code.visualstudio.com/docs/python/environments*