Ahnaf Ahmad

1001835014

Assignment 11 Input Validation

This program is a RESTFUL API that validates name and phone number inputs using regular expressions. The validated inputs are then stored in a SQLITE database.The program is built using python and its flask library. When the program starts, it is by default running on <https://localhost:8080/PhoneBook> .

Program features 3 types of requests.

GET Request

API endpoint: <http://localhost:8080/PhoneBook/list>

This will return a list which contains all the entries that are in the database. If the database is empty, then it will return an empty list.

POST Request

API endpoint: <http://localhost:8080/PhoneBook/add>

This endpoint takes in a json object containing name and phonenumber. Once the input is passed to the API, it checks both inputs using the regular expressions for name pattern and phone number pattern respectively. If the inputs are valid, then it will add the entry to the database using a SQL prepared statement. Otherwise it will return 400 Invalid inputs error.

PUT Request

API endpoint: <http://localhost:8080/PhoneBook/deleteByName>

This endpoint takes in a json object containing name only. Once the input is passed to the API, it checks the name input using regular expression. If the input is valid, then it checks if the entry with that name actually exists using a SELECT statement. If the entry with the that phoneNumber exists, it will continue on with the DELETE statement. If the entry does not exist, it returns a 404 error. If the input validation fails, it returns a 400 error.

API endpoint: <http://localhost:8080/PhoneBook/deleteByNumber>

This endpoint takes in a json object containing phoneNumber only. Once the input is passed to the API, it checks the phoneNumber input using regular expression. If the input is valid, then it checks if the entry with that name actually exists using a SELECT statement. If an entry with that phoneNumber exists, it will continue on with the DELETE statement. If the entry does not exist, it returns a 404 error. If the input validation fails, it returns a 400 error.

Audit Log:

Everytime the API gets a request, it logs the the request information in the format of “YYYY-MM-DD HH-MM-SS: IP address – “GET/POST/PUT API-Endpoint – Statuscode Statusmessage”. If there is a Json object passed to the request, then it will also request the content of the JSON in the log. The log file is saved as request\_log.txt in the same directory as the program.

Regular Expressions

Name: (([A-Z][a-z](’[A-Z][a-z])?(-[A-Z][a-z])?), ([A-Z][a-z]) [A-Z][.]\Z)|(([A-Z][a-z](’[A-Z][a-z])?(-[A-Z][a-z])?), (([A-Z][a-z]) )([A-Z][a-z])\Z)|(([A-Z][a-z])(,? ([A-Z][a-z](’[A-Z][a-z])?(-[A-Z][a-z])?))?\Z)

This regular expression was designed based on the inputs provided for the unit-test. It looks for the patterns of the following format:

* LastName, FirstName MiddleInitial.
* Lastname, FirstName MiddleInitial
* Lastname, Firstname
* Firstname Lastname

Phone Number: (+[1-9][0-9]{0,2}([ ]?))?(([0-9]{5}\Z)|(((1?[(][0-9]{3}[)])|((1-)?([0-9]{3}[-])))?([0-9]{3}-[0-9]{4}))|((1 )?[0-9]{3} [0-9]{3} [0-9]{4})|((1.)?[0-9]{3}.[0-9]{3}.[0-9]{4})|((45 )?[0-9]{2} [0-9]{2} [0-9]{2} [0-9]{2})|((45 )?[0-9]{4} [0-9]{4})|([0-9]{5}.[0-9]{5})|([0-9]{3} ([0-9] )?[0-9]{3} [0-9]{3} [0-9]{4})|(([0-9]{2}) [0-9]{3}-[0-9]{4}))

This regular expression was designed by considering the following formats:

* Optional country code at the beginning of the number.
* Different US phone number formats such as (nnn), nnn-, 1-nnn followed by the rest of the numbers.
* International numbers such as Danish numbers which have the sequence of two or four groups of two digits separated by spaces or periods.
* Other international numbers, which can have 3 to 4 groups of digits separated by spaces or periods.

Assumptions

The following points are the assumptions I made while I coded the program.

* For delete operations of PUT requests, if the input passes the validation but the there is no entry in the database that matches the input, the API should return a 404 not found error.
* The POST and PUT request will always have the inputs passed into them in a JSON object. Therefore, my program needs to parse the json to get the inputs when it handles those requests.
* No alphabetical characters allowed for phone number inputs (A-Z and a-z)
* If for some reason the program can’t access the phonebook.db, the program will automatically create a new one.

Pros

* The checking of if there is an entry matching the input before executing the DELETE statement can prevent any database side problems or errors.
* The configuration of having the program always run on localhost allows easy access, even if the program is running from a docker container.
* The SQL statements are in prepared format, which avoids any SQL injection vulnerabilities.

Cons

* Since both regular expressions were inspired by the unit tests and the description given in the assignment instruction, if it encounters an input that has a completely different format, it may accept it when it is supposed to reject it, or it may reject it when it is supposed to accept it.

How to run the program

Pre-requisites: Must have phonebook.py, Dockerfile, requirements.txt, and preferably phonebook.db files in the same directory.

1. Build the docker file by running the command in terminal “docker build -t my-flask-app .”

Text

Description automatically generated

1. After the docker file has been built, run the command “docker run -p 8080:8080 my-flask-app”

Text

Description automatically generated

Once this message appears in the terminal, the API is ready to be accessed at <https://localhost:8080/PhoneBook/>

API Endpoints:

GET request: <http://localhost:8080/PhoneBook/list> - returns all the entries in the phonebook in an array

A screenshot of a computer

Description automatically generated with medium confidence

POST request: <http://localhost:8080/PhoneBook/add> - must pass a json with a “name” and “phoneNumber” attribute. It adds the entry to the database upon checking the input.

Graphical user interface

Description automatically generated

Example of how to pass in a json object on Postman.

PUT Requests:

1. <http://localhost:8080/PhoneBook/deleteByName> - deletes the entry matching the name from database upon input validation. Must pass in a json object with the field “name”.

Graphical user interface

Description automatically generated

Example of how to pass in a json object to the endpoint via Postman

1. <http://localhost:8080/PhoneBook/deleteByNumber> - deletes the entry matching the phone number upon input validation. Must pass in a json object with the filed “phoneNumber”

Graphical user interface

Description automatically generated

Example of how to pass in a json object to the endpoint via Postman

How to test on Postman

1. On postman, click File -> Import then import the “phonebook-api-testing.postman\_collection.json” file.

Graphical user interface, text

Description automatically generated

After importing the json.

1. Make sure the API is running through docker. Now it is ready to be tested. Each unit tests are already categorized based on if they are acceptable or unacceptable, and based on their request type. Additionally, each folder contains unit tests. There are 9 acceptable phone numbers and 9 unacceptable phone numbers, 6 acceptable names and 6 unacceptable names, and finally my own set of 4 acceptable name and phone numbers and set of 4 unacceptable name and phone numbers. To test each, click on the folder, then click Run on top right.

Text

Description automatically generated



Graphical user interface, text

Description automatically generated

Then click Run phonebook-api-testing button as shown above. After clicking the button, the screen should look like the following:

Graphical user interface, text, application

Description automatically generated

In this example, I tested the acceptable inputs, therefore the response I got from the api was all ‘200 OK’ as shown in the image above. In other case, if the entry is unacceptable, the api will return ‘400 BAD REQUEST’ on the tester. In particular case for PUT requests where the input is valid but the entry does not exist in the database, the API will return ‘404 NOT FOUND’.