CSE:5382-001: SECURE PROGRAMMING
ASSIGNMENT: INPUT VALIDATION

SUBMITTED BY: DEVYANI SINGH
UTA ID: 1001959376

Tech Stack:

1. Java 1.8
2. Spring boot 2.7.0
3. Maven framework
4. Swagger UI for API documentation and interacting with the APIs.
5. Junit Test suite
6. SQL Lite database

Functionalities Implemented:

1. Following APIs have been implemented to manage the phone book:
   a. ADD operation.
   b. DELETE by name operation.
   c. DELETE by number operation.
   d. LIST operation
   e. Audit log functionality.
2. Used SQL lite database to store input data. **[BONUS]**
3. Implemented parameterized queries using prepared statements. **[BONUS]**
4. Junit test cases to: **[BONUS]**
   a. Test Acceptable name and number strings.
   b. Test Unacceptable name and number strings.
   c. Test Student provided inputs.

Steps to run the application:

A. Creating a docker image and executing it:

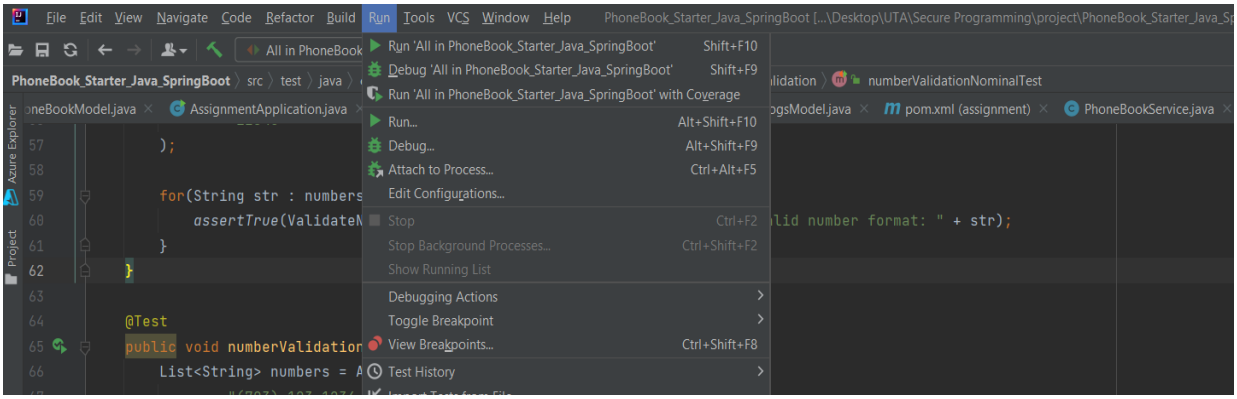
1. Extract the project zip file.
2. Make sure Docker daemon is running. If not, start docker using docker desktop application.
3. Open the command line and navigate to /1001969376_Assignment/PhoneBook_Starter_Java_SpringBoot directory.
4. Run the following commands to create a docker image and execute it:
   > docker build . --tag=assignment:latest
   > docker run -p8080:8080 assignment:latest
5. Open: http://localhost:8080/swagger-ui/index.html#/phone-book-controller and click on 'Try it out' to test the APIs.
6. Under the tab phone-book-controller, all the APIs can be found. Go to the "**Executing API's**" section to execute all the available APIs.

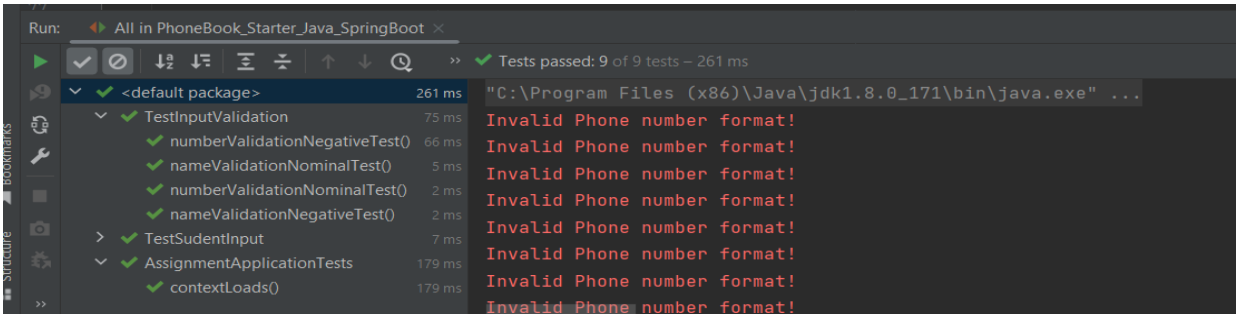
B. Importing project in IntelliJ Idea and running it.

1. Extract the project zip.
2. Open IntelliJ Idea IDE and open the extracted project.
3. Build the Project.
4. Navigate to src->main->java
   Under package: ***com.secureProgramming.assignment***,
   Run -> AssignmentApplication.java
   This will start the spring boot application.
5. Open: http://localhost:8080/swagger-ui/index.html#/phone-book-controller and click on 'Try it out' to test the APIs.
6. Under the tab phone-book-controller, all the APIs can be found. Go to the "**Executing API's**" section to execute all the available APIs.

Execute JUnit test cases:

Navigate to Run -> Run 'All in PhoneBook_Starter Java_SpringBoot' to run all the test cases.



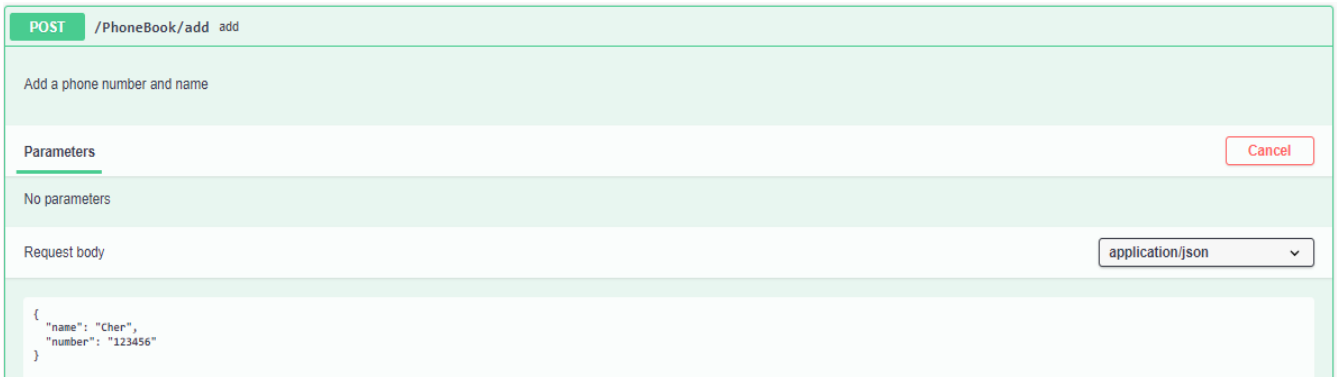


Description of the code:

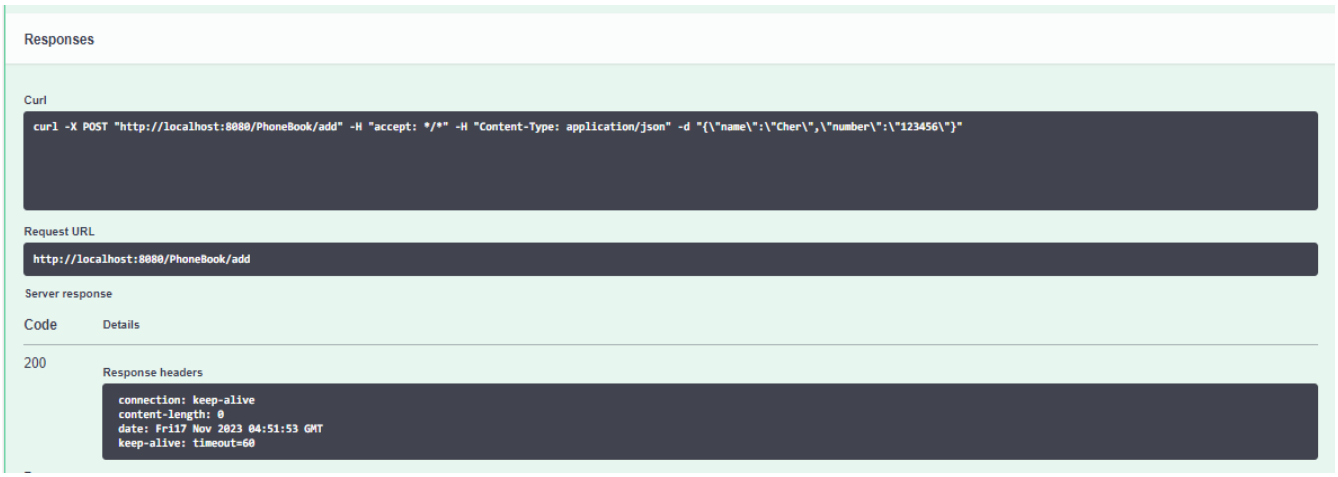
1. Application consists of a controller class -> PhoneBookController.java where the request mapping is done for GET/POST/DELETE APIs.
2. In this controller, we validate the provided input against some regex. The validators are present in inputValidators package, in ValidateName.java and ValidateNumber.java classes.
3. In both the validate classes, if the provided input satisfies the given regex, 200 OK response code is returned. Else, 400 code is returned with error : “Invalid Phone Number format!” or “Invalid Name format”.
4. Description and testing of all the APIs can be found below:

Executing APIs

1. **ADD operation:**
    - This API has been designed to persist the name along with phone numbers in the phone book.
    - **POST API implemented: /PhoneBook/add**
      Parameters required:
        - String: name
        - String: number
        - (id is optional)
    - Provide the desired name and number to add it to the phonebook.
    - Sample:



- Response:



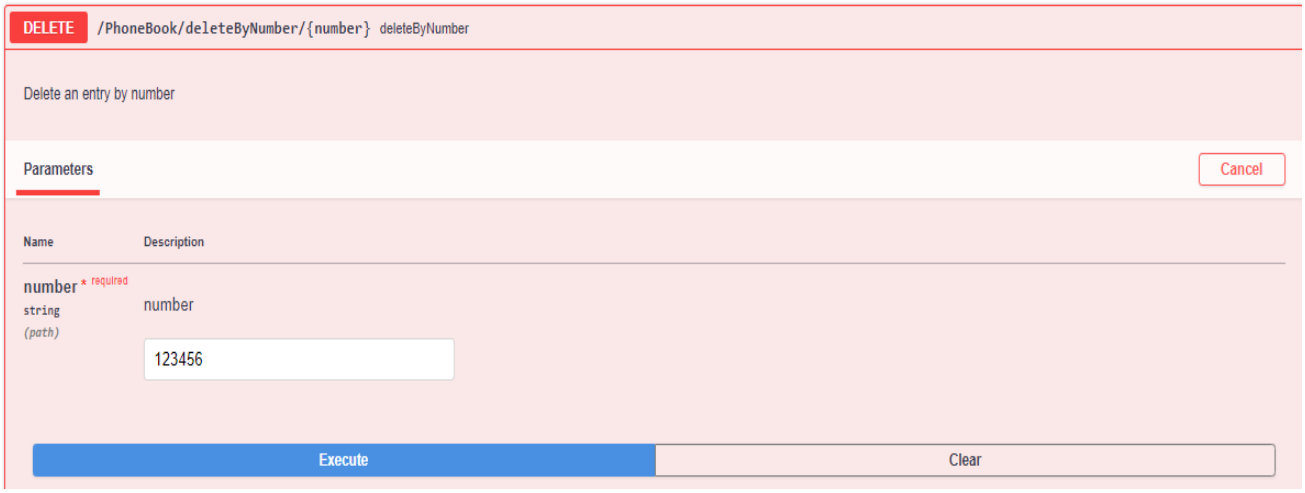
2. **DEL by name operation:**
    - This API has been designed to delete a record by providing a name as input.
    - **DELETE API implemented: /PhoneBook/deleteByName/{name}**
    - **Parameters required:**

        String: name
    - Sample:

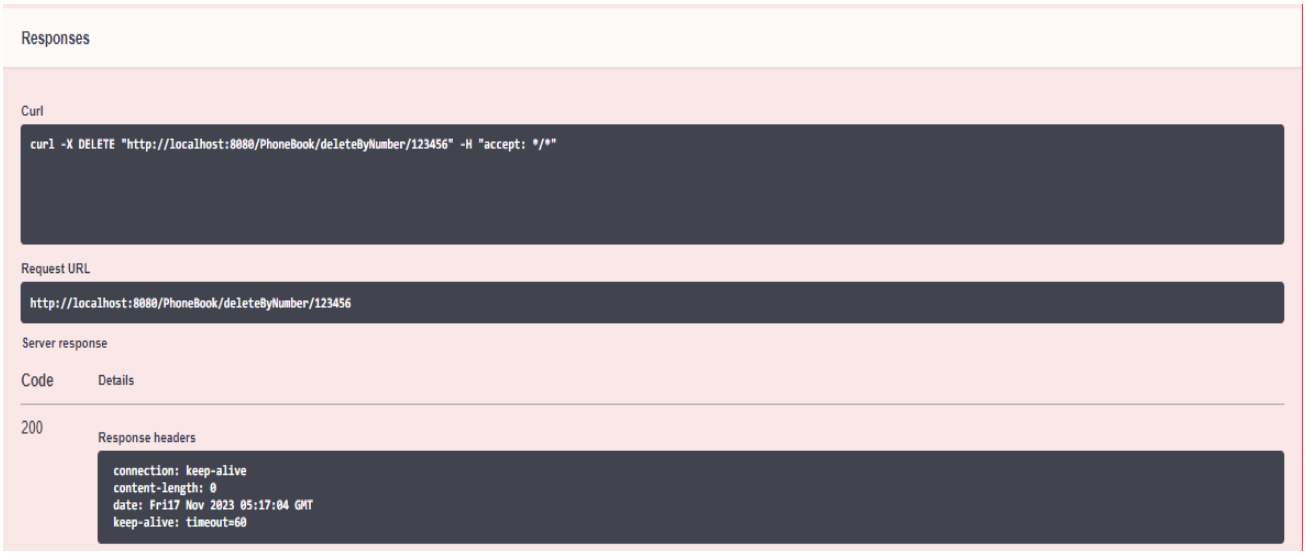
- Response:

3. **DEL by number operation:**
    - This API has been designed to delete a record by providing a number as input.
    - **DELETE API implemented:** /PhoneBook/deleteByNumber/{number}
    - **Parameters required:**

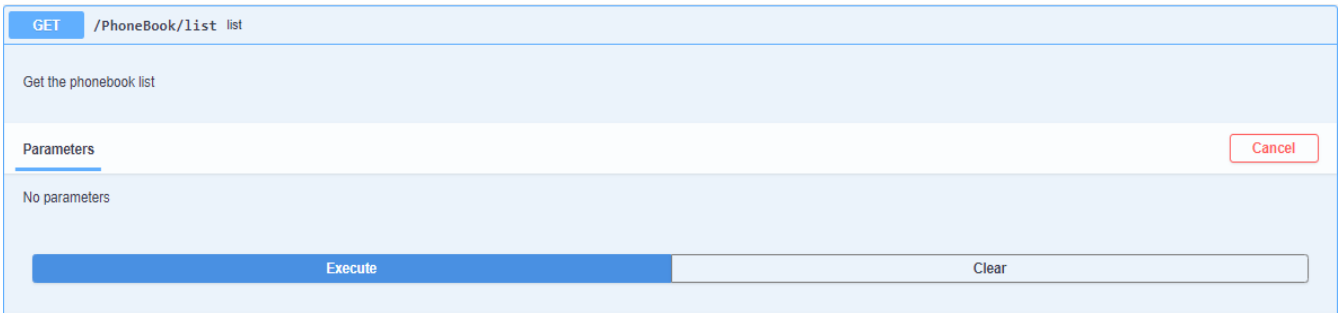
        String: number
    - Sample:



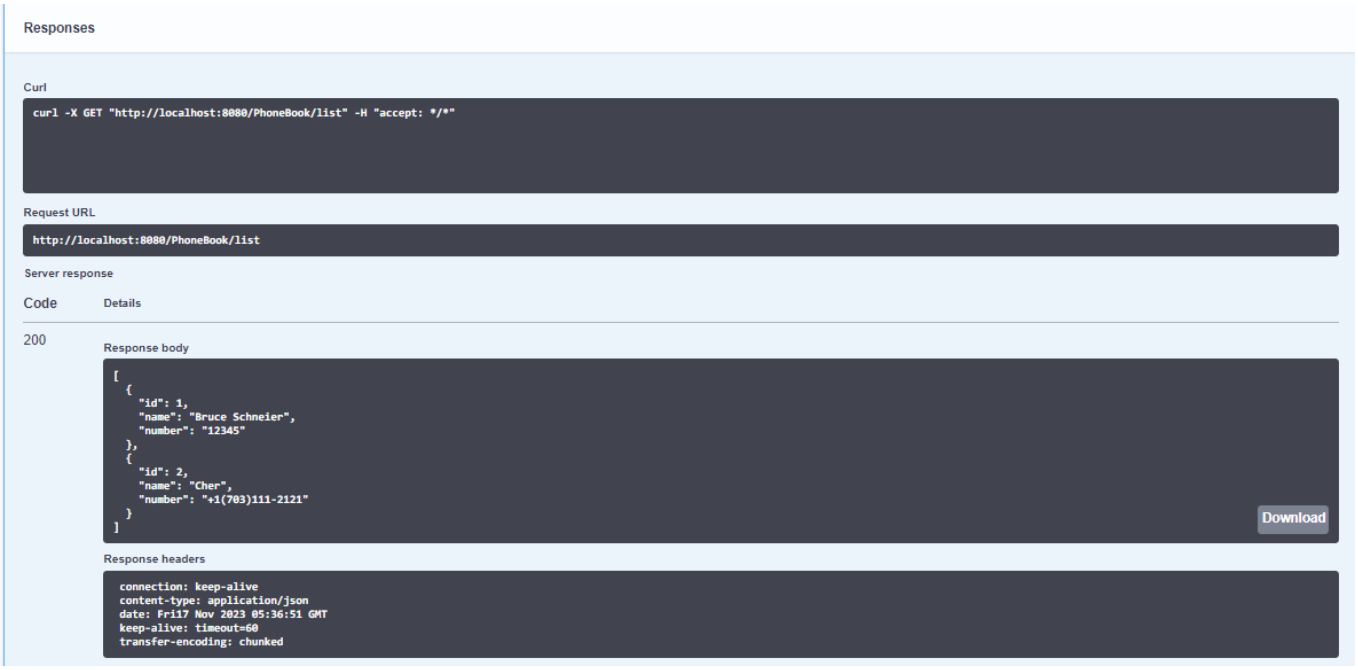
- Response:



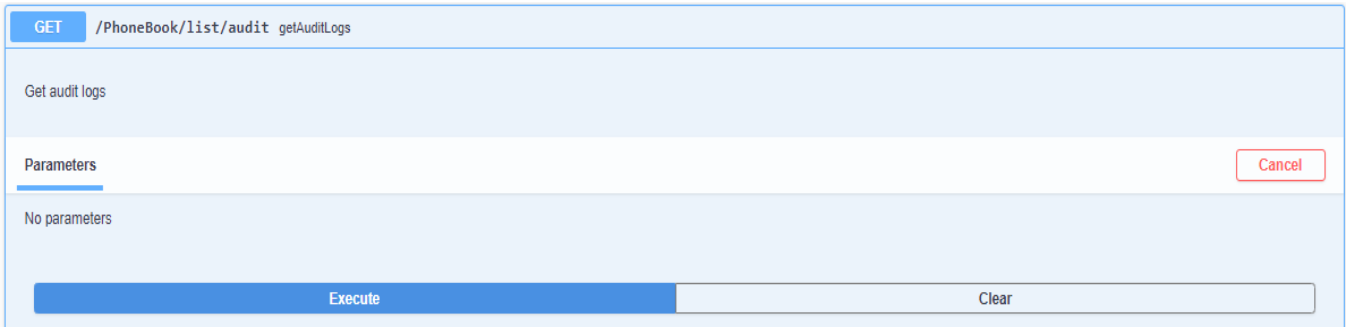
4. **LIST operation:**
    - The API has been implemented to list all the available records in the phone book.
    - **API Implemented:** /PhoneBook/list
    - Parameters required: NONE
    - Sample:



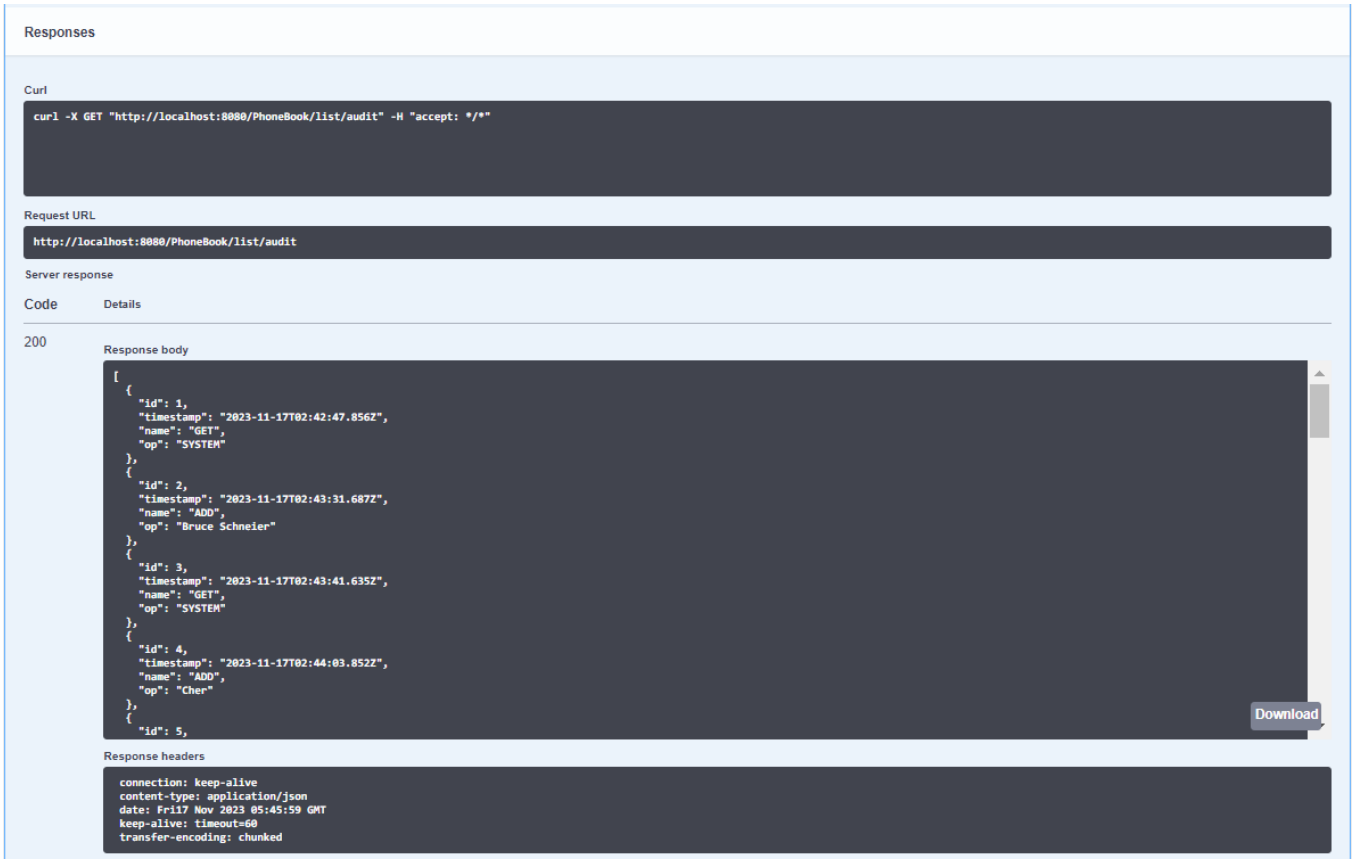
- Response:



5. **Audit log functionality:**
    - The API has been implemented to list all the audit logs.
    - **API implemented:** /PhoneBook/list/audit
    - Parameters required: NONE
    - Sample:



- Response:



Assumptions:

1. Logs are being persisted and retrieved from the database.
2. User is providing phone number with the country code.
3. An individual can have multiple phone numbers.
4. Same number cannot belong to multiple individuals.

Pros:

1. Structure of the project is simple and easy to understand.
2. Swagger UI has been implemented to simplify the testing of the APIs.
3. Prepared statements have been used in sql statements to interact with the database, to prevent the user from performing any SQL injections.

Cons:

1. User can perform only one operation at a time.
2. Has several dependencies.