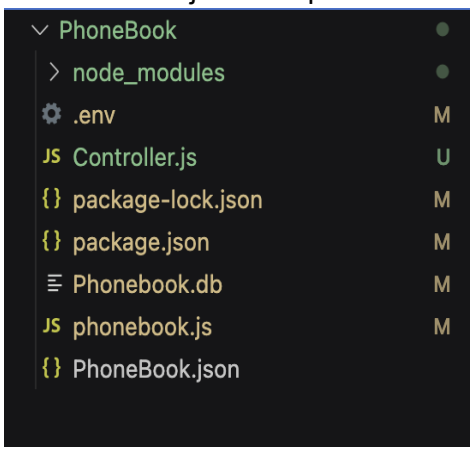
yxk9640
1001879640

File Structure:

The whole project is structured as in fig below. The information regarding all dependencies is available in **package.json**. **phonebook.js** is the main file that has to be run and phonebook.js calls Contoller.js and implements the services and database requests.



Compile:

No compilation required. However a few setup has to be done before getting started.
The user system PORT:5001 should be available. If a process is already using this prot the user can either kill the process by

1. lsof -i :5001 → obtain ProcessID and then implement
2. kill -9 <ProcessID>

Alternatively user can change the port in the Controller.js file line:20 to desired port number.

```
const port = process.env.PORT || 5001;
```

Programs/ Software required to be installed to run the code

1. VS Code or Basic Terminal
2. Postman Desktop which can be downloaded here : https://www.postman.com/downloads/

Installation:

In order to run the phonebook.js file the user system must have a few installations:

1. node: can be installed by visiting the link : https://nodejs.org/en/download/

Further its recommended to install below packages for smoother implementation:

1. npm init
2. npm install express
3. npm install sqlite3
4. npm install nodemon
5. npm install winston

6. npm install log-timestamp

Run:

The project can be implemented either from command line

1. Terminal: The user should be in the PhoneBook folder and then implement following commands.
    a. $ nodemon phonebook.js

Once the program is running the following lines are displayed in terminal

```
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node phonebook.js`
Started
Connect to API
Initiate connection to DB
App running at http://localhost:5001/PhoneBook/list
```

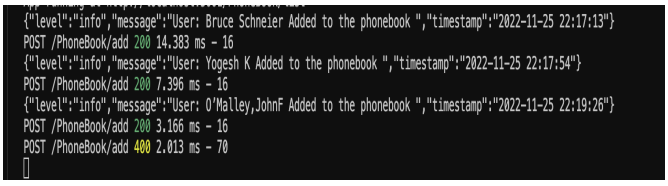
Now, the user can perform GET, POST and PUT requests using Postman Desktop.
Setting up Postman:

1. Configuring Postman Desktop
    a. Once the Postman is started user now has to click on " + " to open a new tab.
    b. Next, user can select request from dropdown menu.
    c. Mention the localhost url (say GET request http://localhost:5001/PhoneBook/list if user successfully implements the above instructions.)

Description of Code:

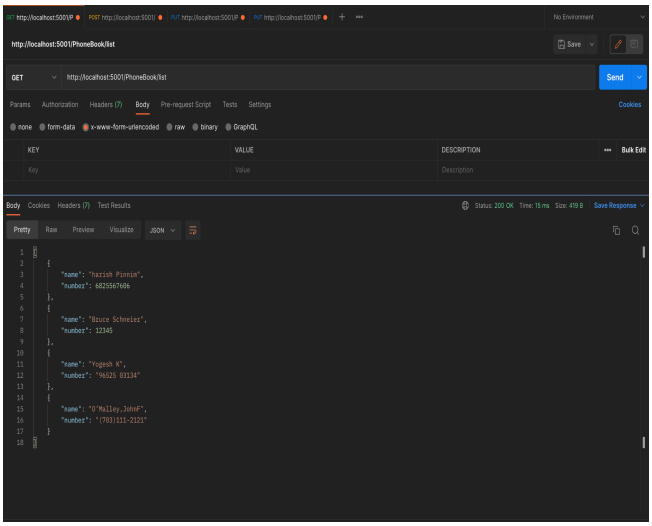
The code require for the successful implementatio is observed in "**phonebook.js**" and "**Controller.js**". Phonebook.db is used to store the name and number obtained from requests. phonebook.js file is just used to call Controller.js and no mian logic is implemented.
The whole logic is in Controller.js

1. First all the libraries that are required are called.(line: 7 to line: 16)
2. The connection is established in one of the mentioned port.(line:20 to line:24)
3. Further requests are implemented GET, POST, PUT in same order(line: 27 to line: 111)
    a. GET(/PhoneBook/list) request is used to display the list of names and numbers of users in phonebook in database which is done by `SELECT- query`. Upon successful retrieval of list status:200 OK is returned along with the list.



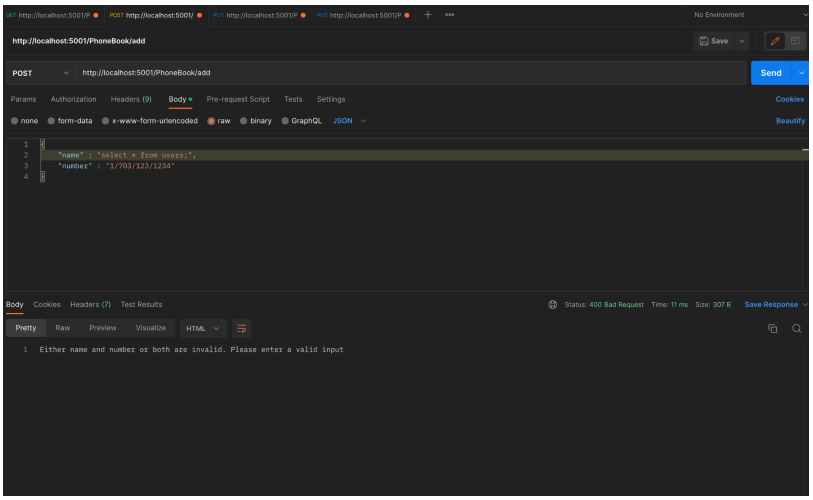
Picture of Logger message for GET(/PhoneBook/list)

yxk9640
1001879640



Picture of POSTMAN implementation for GET(/PhoneBook/list)

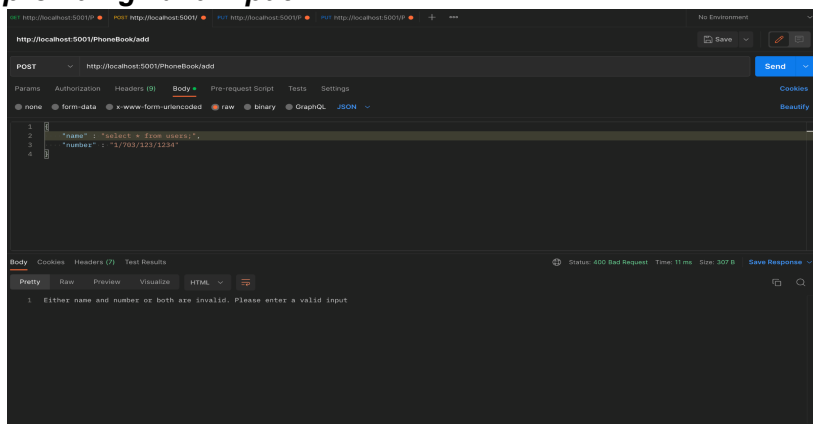
b. POST(/PhoneBook/add) is used to insert the data into the database and before inserting into the db the inputs are checked with a custom regular expression.



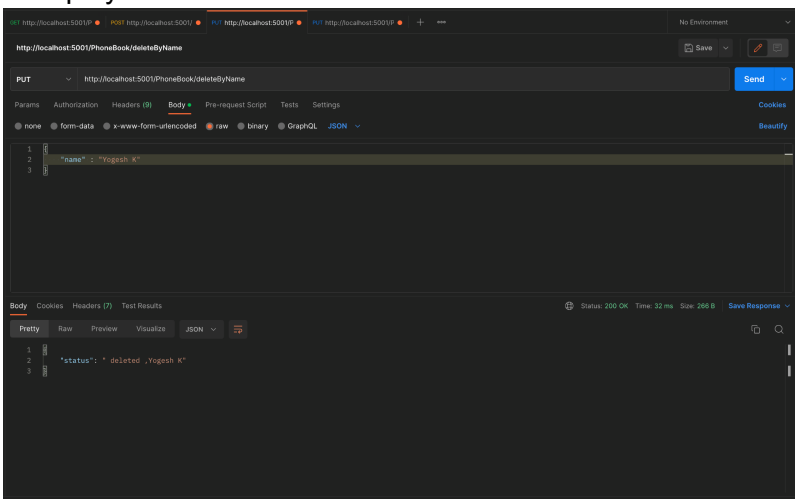
Picture of POSTMAN implementation for GET(/PhoneBook/list) on providing invalid input

yxk9640
1001879640

Picture of POSTMAN implementation for GET(/PhoneBook/list) on providing valid input

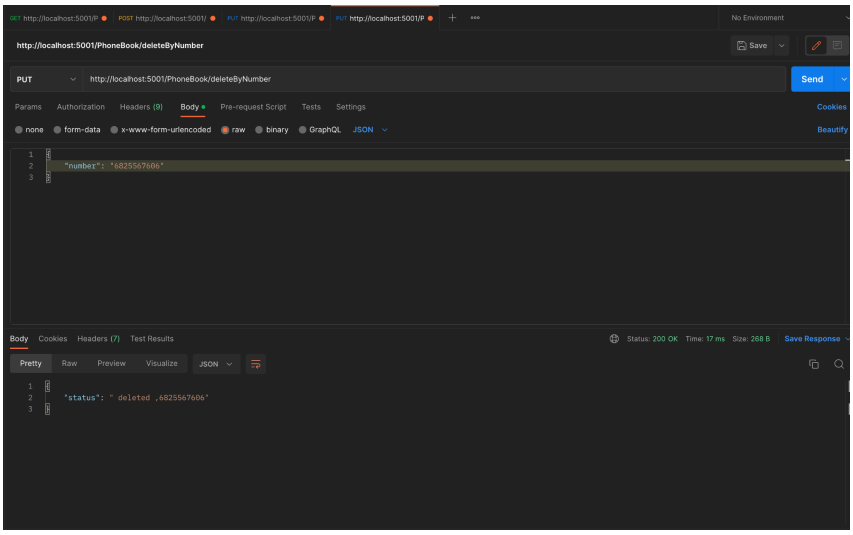


c. PUT(/PhoneBook/deleteByName) is used to delete the data of the user whose name is mentioned and if the user is present in the database the user name and number are deleted and if not status:404 Not Found and corresponding message is displayed.



d. PUT(/PhoneBook/deleteByNumber) is used to delete the data of the user whose number is mentioned and if the user is present in the database the user name and number are deleted and if not **status:404 Not Found**

yxk9640
1001879640



PUT(/PhoneBook/deleteByNumber) in Postman

Design of regular expression:

1. Name:

```
const Namepattern = new
RegExp(/^([a-z]|[A-Z]|[\s]|,|[?!.*']|-)*$/);
```

Checks if SQL injection can happen and if scripts are passed as inputs. Further specified inputs are validated and inserted.

2. Number:

```
const patternNumber = new
RegExp(/(^[+]*[-\s(\s\.0-9]{0,2}[-\s\.0-9]{1,3}[\s)\s]{0,3}[-\s\.0-9]*$)|([0-9]
{5,})/);
```

Checks if SQL injection can happen and if scripts are passed as inputs. Further specified inputs are validated and inserted.

Assumptions:

For PUT request if a user tries to attack by providing data other than name/number the program does not allow are data provided by user does not tally with data in database.

Pros/ Cons:

Input validation is only done in POST method.