

REPORT

NAME: VAIBHAVI P. ARJUNWADKAR

STUDENT ID: 1001826818

Prerequisites and links for installation:

1. NodeJS:

steps to install NodeJS on your PC:

visit below link to download and install the latest recommended version of node (windows/mac) on your PC.

LINK: <https://nodejs.org/en/download/>

Set below NodeJS variable path in environment variables if not by default gets added after installation:

C:\Program Files\nodejs\

2. postman:

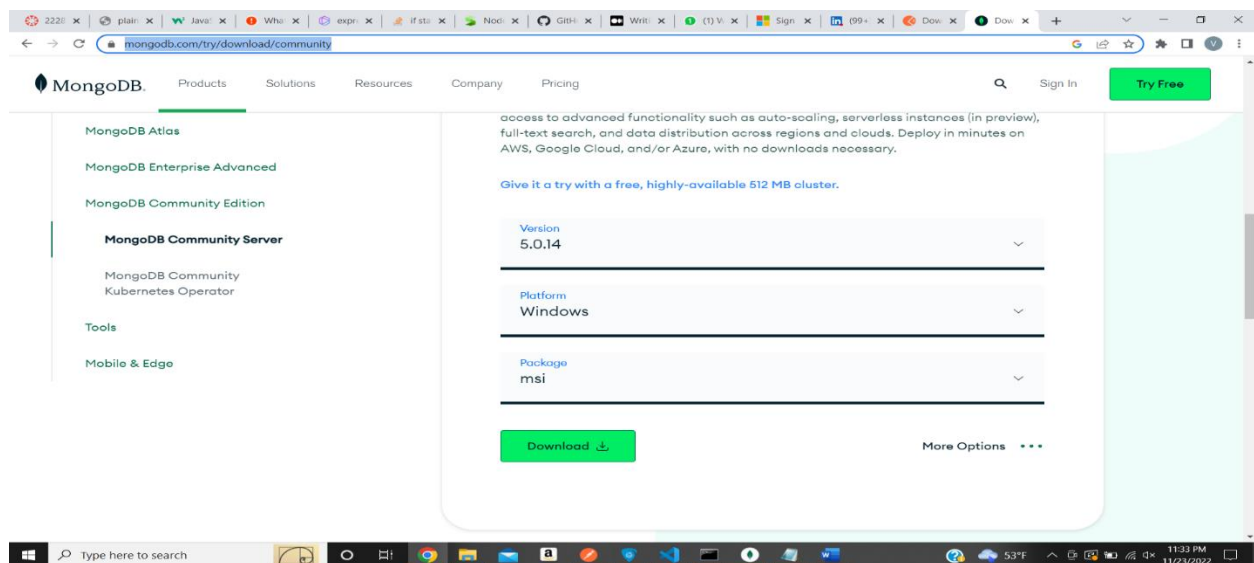
visit below link to download and install the postman application on your PC.

LINK: <https://www.postman.com/downloads/>

3. MongoDB:

visit below link to download and install the MongoDB on your PC.

LINK: <https://www.mongodb.com/try/download/community>



Set below MongoDB variable path in environment variables if not by default gets added:

C:\Program Files\MongoDB\Server\5.0\bin

Running the code on the local machine:

1. Download the zip folder named 'project_SP_1001826818.zip'.
2. unzip the folder.
3. the folder contains below files:

- 1 folder: projectPhonebook
 - app.js
 - package-lock.json
 - package.json
 - phonebook.model.js
- 1 report named 'SP_projectReport_1001826818.pdf'

4. Open cmd window 1 (No need to go to source code file)

Run command: mongod

It should give “msg” like waiting for connections and “port” should be 27017 as shown in the last line of below screenshot:

```
Command Prompt - mongod
iredTiger]]
{"t":{"sdate":"2022-11-24T00:00:18.787-06:00"},"s":"I", "c":"STORAGE", "id":22315, "ctx":"initandlisten","msg":"Opening WiredTiger","attr":{"config":{"create,cache_size=156800,session_max=3000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),builtin_extension_config=(zstd=(compression_level=6)),file_manager=(close_idle_time=600,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=(recovery_progress,checkpoint_progress,compact_progress),}}}}
{"t":{"sdate":"2022-11-24T00:00:18.828-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269618:827650"]}[27876:140715926510032], txn-recover:
[WT_VERB_RECOVERY_PROGRESS] Recovering log 4 through 5}}
{"t":{"sdate":"2022-11-24T00:00:18.863-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269618:863636"]}[27876:140715926510032], txn-recover:
[WT_VERB_RECOVERY_PROGRESS] Recovering log 5 through 5}}
{"t":{"sdate":"2022-11-24T00:00:18.902-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269618:902636"]}[27876:140715926510032], txn-recover:
[WT_VERB_RECOVERY_PROGRESS] Recovering log 5 through 5}}
{"t":{"sdate":"2022-11-24T00:00:18.963-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269618:962635"]}[27876:140715926510032], txn-recover:
[WT_VERB_RECOVERY_PROGRESS] Recovering log 4 through 5}}
{"t":{"sdate":"2022-11-24T00:00:19.005-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269619:5636"]}[27876:140715926510032], txn-recover: [
WT_VERB_RECOVERY_PROGRESS] Recovering log 5 through 5}}
{"t":{"sdate":"2022-11-24T00:00:19.038-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269619:38168"]}[27876:140715926510032], txn-recover:
[WT_VERB_RECOVERY_PROGRESS] Set global recovery timestamp: (0, 0)}}
{"t":{"sdate":"2022-11-24T00:00:19.038-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269619:38168"]}[27876:140715926510032], txn-recover:
[WT_VERB_RECOVERY_PROGRESS] Set global oldest timestamp: (0, 0)}}
{"t":{"sdate":"2022-11-24T00:00:19.040-06:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":["1669269619:39068"]}[27876:140715926510032], WT_SESSION.ch
eckpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 1, snapshot max: 1, snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 3029}}
{"t":{"sdate":"2022-11-24T00:00:19.060-06:00"},"s":"I", "c":"STORAGE", "id":4795906, "ctx":"initandlisten","msg":"WiredTiger opened","attr":{"durationMillis":273}}
{"t":{"sdate":"2022-11-24T00:00:19.060-06:00"},"s":"I", "c":"RECOVERY", "id":23987, "ctx":"initandlisten","msg":"WiredTiger recoveryTimestamp","attr":{"recoveryTimestamp":{"timestamp":{"t":0,"i":0}}}}
{"t":{"sdate":"2022-11-24T00:00:19.066-06:00"},"s":"I", "c":"STORAGE", "id":22262, "ctx":"initandlisten","msg":"Timestamp monitor starting"}
{"t":{"sdate":"2022-11-24T00:00:19.071-06:00"},"s":"I", "c":"CONTROL", "id":22120, "ctx":"initandlisten","msg":"Access control is not enabled for the database. Read and write access to data and configuration
is unrestricted","tags":["startupWarnings"]}
{"t":{"sdate":"2022-11-24T00:00:19.071-06:00"},"s":"I", "c":"CONTROL", "id":22140, "ctx":"initandlisten","msg":"This server is bound to localhost. Remote systems will be unable to connect to this server. Sta
rt the server with --bind_ip address to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip
127.0.0.1 to disable this warning","tags":["startupWarnings"]}
{"t":{"sdate":"2022-11-24T00:00:19.074-06:00"},"s":"I", "c":"NETWORK", "id":4915702, "ctx":"initandlisten","msg":"Updated wire specification","attr":{"oldSpec":{"incomingExternalClient":{"minWireVersion":0,"ma
xWireVersion":13},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient":true},"newSpec":{"incomingExternalClient":{"minWireVersi
on":0,"maxWireVersion":13},"incomingInternalClient":{"minWireVersion":13,"maxWireVersion":13},"outgoing":{"minWireVersion":13,"maxWireVersion":13},"isInternalClient":true}}}
{"t":{"sdate":"2022-11-24T00:00:19.074-06:00"},"s":"I", "c":"STORAGE", "id":5071100, "ctx":"initandlisten","msg":"Clearing temp directory"}
{"t":{"sdate":"2022-11-24T00:00:19.075-06:00"},"s":"I", "c":"CONTROL", "id":20536, "ctx":"initandlisten","msg":"Flow Control is enabled on this deployment"}
{"t":{"sdate":"2022-11-24T00:00:19.219-06:00"},"s":"I", "c":"FTDC", "id":20625, "ctx":"initandlisten","msg":"Initializing full-time diagnostic data capture","attr":{"dataDirectory":"C:/data/db/diagnostic.
data"}}
{"t":{"sdate":"2022-11-24T00:00:19.221-06:00"},"s":"I", "c":"REPL", "id":6015317, "ctx":"initandlisten","msg":"Setting new configuration state","attr":{"newState":"ConfigReplicationDisabled","oldState":"Con
figPreStart"}}
{"t":{"sdate":"2022-11-24T00:00:19.224-06:00"},"s":"I", "c":"NETWORK", "id":23015, "ctx":"listener","msg":"Listening on","attr":{"address":"127.0.0.1"}}
{"t":{"sdate":"2022-11-24T00:00:19.224-06:00"},"s":"I", "c":"NETWORK", "id":23016, "ctx":"listener","msg":"Waiting for connections","attr":{"port":27017,"ssl":"off"}}
```

Possible error and its solution: If mongod gets terminated with the exit code error then you need to add one empty folder in your C directory named 'data' and inside that data folder make another empty folder named 'db'

5. Open cmd window 2 (No need to go to source code file)

Run command: mongo

```
Command Prompt - mongo
C:\Users\Vaibhavi>mongo
MongoDB shell version v5.0.13
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1684e59c-c09f-42b2-a1ef-46f00df2897b") }
MongoDB server version: 5.0.13

*****
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
*****

The server generated these startup warnings when booting:
  2022-11-15T00:43:27.633-06:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
>
```

6. Open cmd window 3

- go inside the folder projectPhonebook by path.

Ex: C:\ 'project_SP_1001826818\projectPhonebook>

- run command on cmd : npm i (It will install all the node_modules require to run the code)

- possible error: Cannot find module 'express'

Solution: run 'npm install express'

- now run the final command to run our source code in app.js file. use command:

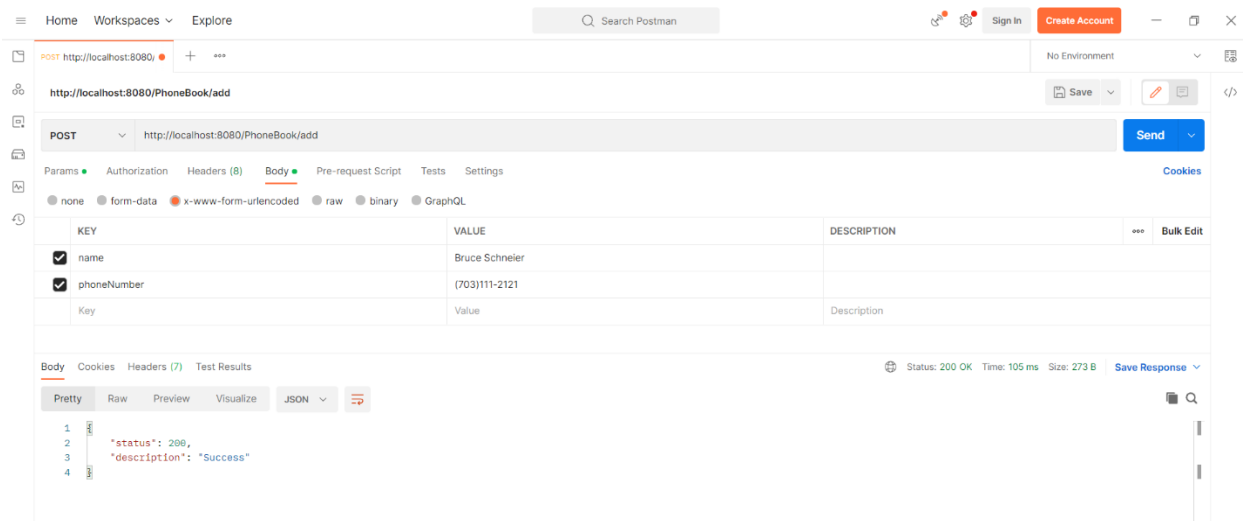
node app.js

this will give output on cmd like below: after getting this output we are good to run and test the api on postman.

```
C:\UTA\UTA_Courses\1st-sem\SP-ThomasJones\project\projectPhonebook>node app.js
App is running on the Port 8080
The logFile has got created! You can now open this file anytime and check for the logs as any api has got executed or any error has occurred...
-
```

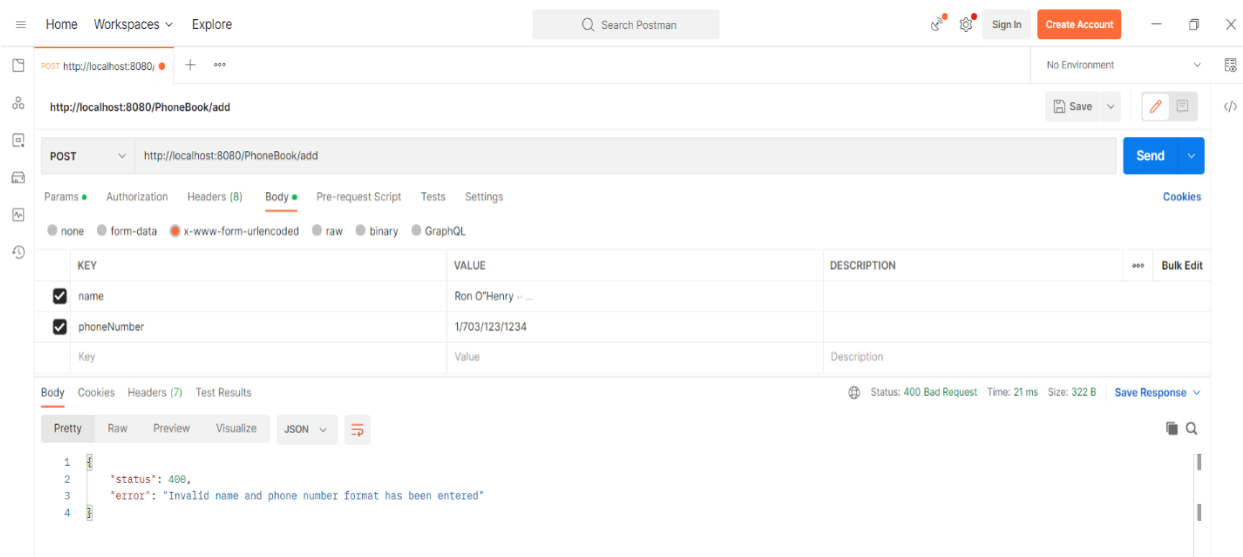
Testing the code on the postman application:

1. Open Postman application.
2. Test POST **/PhoneBook/add** api. It will add a new person to the database.

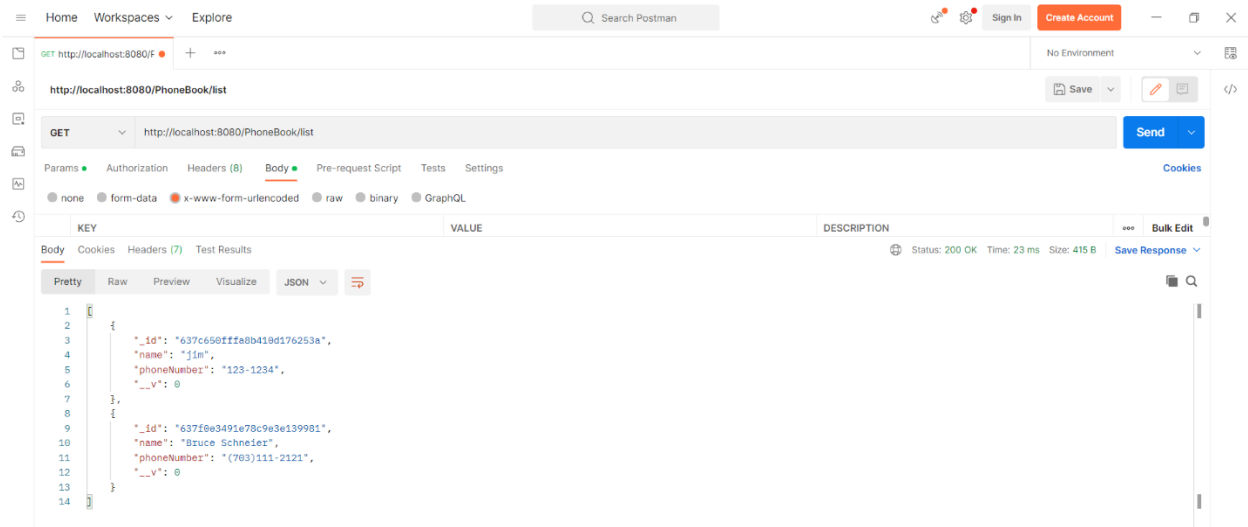


You can run this api as many times as we wish to add new person into the database.

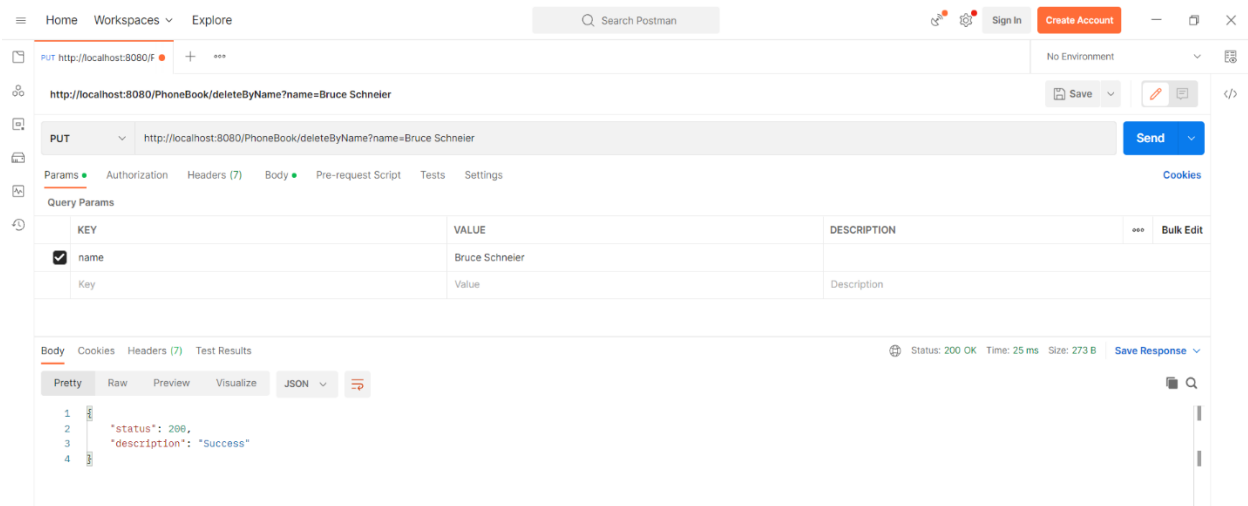
If we try to enter unacceptable name or number or both then it will give “bad request” error with status code 400 like below screenshot:



3. Test GET /PhoneBook/list api. It will produce a list of the members of the database.

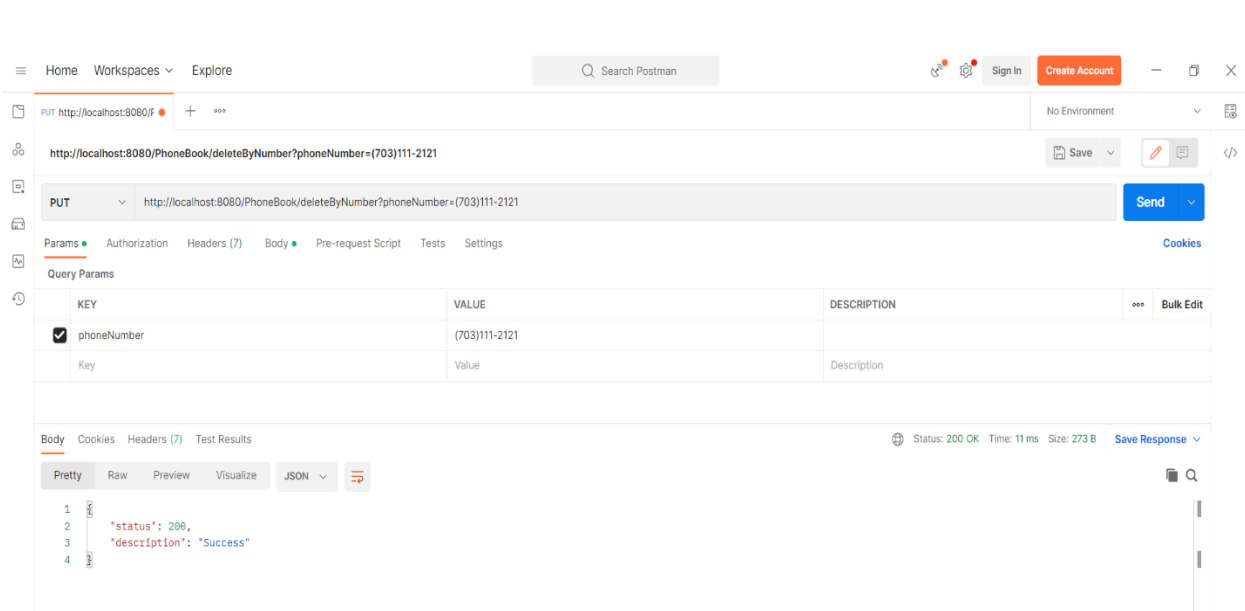


4. Test PUT /PhoneBook/deleteByName api. It will remove someone from the database by name.



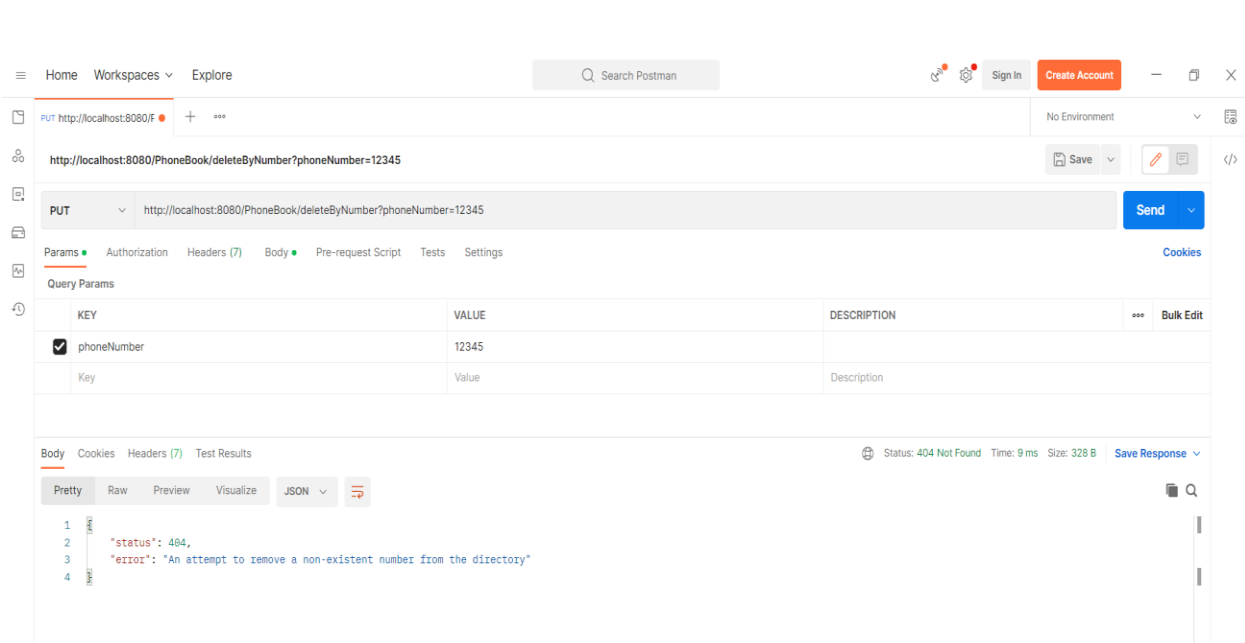
If we try to enter unacceptable name to be deleted from the database, then it will give “bad request” error with status code 400 like second screenshot in point 2.

5. Test PUT /PhoneBook/deleteByNumber api. It will remove someone from the database by number.



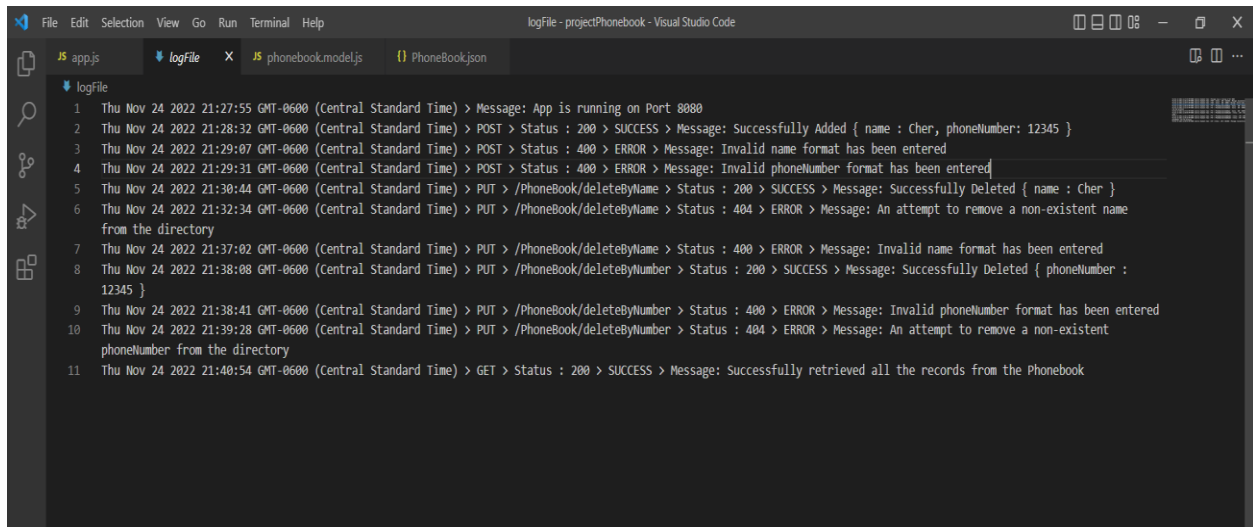
If we try to enter unacceptable number to be deleted from the database, then it will give “bad request” error with status code 400 like second screenshot in point 2.

If we try to delete the record from the phonebook which does not exists in our database, then it will return status code 404 as shown in below screenshot:



Audit log functionality:

When we run our app.js source code file, It will create one text log file named 'logFile' in the 'projectPhonebook' folder. If you open this file, you will find all the logs according to the api call. I have attached the screenshot of logFile after performing some api call as below:



```
logFile - projectPhonebook - Visual Studio Code
1 Thu Nov 24 2022 21:27:55 GMT-0600 (Central Standard Time) > Message: App is running on Port 8080
2 Thu Nov 24 2022 21:28:32 GMT-0600 (Central Standard Time) > POST > Status : 200 > SUCCESS > Message: Successfully Added { name : Cher, phoneNumber: 12345 }
3 Thu Nov 24 2022 21:29:07 GMT-0600 (Central Standard Time) > POST > Status : 400 > ERROR > Message: Invalid name format has been entered
4 Thu Nov 24 2022 21:29:31 GMT-0600 (Central Standard Time) > POST > Status : 400 > ERROR > Message: Invalid phoneNumber format has been entered
5 Thu Nov 24 2022 21:30:44 GMT-0600 (Central Standard Time) > PUT > /PhoneBook/deleteByName > Status : 200 > SUCCESS > Message: Successfully Deleted { name : Cher }
6 Thu Nov 24 2022 21:32:34 GMT-0600 (Central Standard Time) > PUT > /PhoneBook/deleteByName > Status : 404 > ERROR > Message: An attempt to remove a non-existent name from the directory
7 Thu Nov 24 2022 21:37:02 GMT-0600 (Central Standard Time) > PUT > /PhoneBook/deleteByName > Status : 400 > ERROR > Message: Invalid name format has been entered
8 Thu Nov 24 2022 21:38:08 GMT-0600 (Central Standard Time) > PUT > /PhoneBook/deleteByNumber > Status : 200 > SUCCESS > Message: Successfully Deleted { phoneNumber : 12345 }
9 Thu Nov 24 2022 21:38:41 GMT-0600 (Central Standard Time) > PUT > /PhoneBook/deleteByNumber > Status : 400 > ERROR > Message: Invalid phoneNumber format has been entered
10 Thu Nov 24 2022 21:39:28 GMT-0600 (Central Standard Time) > PUT > /PhoneBook/deleteByNumber > Status : 404 > ERROR > Message: An attempt to remove a non-existent phoneNumber from the directory
11 Thu Nov 24 2022 21:40:54 GMT-0600 (Central Standard Time) > GET > Status : 200 > SUCCESS > Message: Successfully retrieved all the records from the Phonebook
```

How the code works:

- The main source code folder 'projectPhonebook' contains following files.

1. app.js

It is the main source code file which contains all the javascript code.

This file will create server locally on the port 8080 after running the node app.js command on cmd. It will also create one log file named logfile.txt to log all the activity. (line 165 – 176).

Line 1-8 contains code to insert all the external files and modules like express, fs, body-parser, mongoose and phonebook.model.

Line 10-12 contains the code for MongoDB database connection.

Line 15-17 contains simple get api (/) code which will give you 'lets start the Secure Programming final project' in response

Line 19-20 contains our main regex pattern to match for name and phoneNumber entered. If the user entered name or phoneNumber match with the pattern in the respective regex then that entered name or phoneNumber will gets rejected stating 'Invalid name and phoneNumber format has been entered' with status 400.

Line 22-35 contains code for GET '/PhoneBook/list' api. It will return all the record from the database after successful call and log activity into logfile.txt file.

Line 78-107 contains code for POST '/PhoneBook/add' api. It will ADD the record into the database after the successful api call and log activity into logfile.txt file. It accepts name and phoneNumber to be added as a body parameters from the user.

Line 110-134 contains code for PUT `"/PhoneBook/ deleteByName'` api. It will DELETE the record from the database by name after the successful api call and log activity into logfile.txt file. It accepts name to be deleted as a query parameter from the user.

Line 137-162 contains code for PUT '/PhoneBook/ deleteByNumber' api. It will DELETE the record from the database by phoneNumber after the successful api call and log activity into logfile.txt file. It accepts phoneNumber to be deleted as a query parameter from the user.

2. phonebook.model.js

This file contains the schema of my MongoDB database Collection 'records'.

The schema is :

```
{
  name: String,
  phoneNumber: String
}
```

We have exported this file at the end so that we can import it into our main source code file `app.js`.

3. package-lock.json

This file contains the exact tree that was generated to allow subsequent installs to have the identical tree.

4. package.json

It contains basic information about the project.

The design of regular expressions:

I have used two regex expressions. One for name matching and other for phoneNumber matching.

1. Name

```
regExptNameCode = /(.*[a-zA-Z][ ]*. *[a-zA-Z].*[a-zA-Z][ ]*. *|.*['].*[\-].+[\-].*|. *['].*[\-].*[\-].*|.*|'[{ }]*. *|.*\d.*|. *|<>.*|.*<\>.*|. *|.*[*];.*|^^[^']*[a-zA-Z].*)/
```

Above pattern will see whether the entered query name matches with the above regex pattern or not. If it matches with the above pattern, then it will return True Boolean value. If the return Boolean value is True, then that user entered query name is unacceptable. After this post/put api will return error message like 'Invalid name format has been entered' with status code 400. If the entered user query parameter value for name doesn't match with above regex pattern then it will return False as a Boolean value and that name will be considered as a accepted name value.

1. Number

```
regExptNumberCode = /(^[d]{3}$|^(001\).*|^[+][0-9]{4,}.*|[a-zA-Z].+|^[+][0][1].*|^[0-9]{10}|.[0-9][a-zA-z].*|.*[V].*)/
```


Above pattern will see whether the entered query phoneNumber matches with the above regex pattern or not. If it matches with the above pattern, then it will return True Boolean value. If the return Boolean value is True, then that user entered query phoneNumber is unacceptable. After this post/put api will return error message like 'Invalid name format has been entered' with status code 400. If the entered user query parameter value for phoneNumber doesn't match with above regex then it will return False as a Boolean value and that phoneNumber will be considered as a accepted phoneNumber value.

Assumptions I have made:

I have assumed and code according to the pattern given in the list of acceptable names and numbers while developing the regular expression. My regular expression will block the entries given in the list of unacceptable names and numbers or block the entries in similar scenarios/patterns. But I don't know the scenarios/patterns other than the patterns provided, that I should block. There might be n number of patterns other than the patterns provided for unacceptable name and phonenummer that I should block, but I have considered the patterns related to one given in the assignment-requirements pdf.

Pros/Cons of the approach used:

Pros:

1. This approach of building regular expression will allow us to block unwanted patterns to be passed in the query parameters for post or put api. For example, entries with the numbers (like L33t Hacker) for the 'name' query parameter will be blocked and api will return 400 status code. Similarly, entries with the alphabets (like Nr 102-123-1234) for the 'phoneNumber' query parameter will be blocked and api will return 400 status code.
2. This approach will block any sql query provided as a name query parameter. For example, if user provides name = "select * from users;" it will block user from accessing the database by returning 400 Bad Request response. This can prevent Hacker from Hacking the database and accessing the crucial data from the database.
3. In short this approach is blocking hacker from doing sql injection attack by blocking sql queries sending as a query parameters in post and put api.

Cons: In this approach I have mentioned *. In the end of the regular expression string related to phonenummer. This will allow attacker to enter number of infinite characters in the number query parameter while calling post/put api resulting into buffer overflow attack.