Anirudh Kaluri
Date: 8 March 2022
LinkedIn: https://linkedin.com/in/anirudhkaluri
Github: https://github.com/anirudhkaluri
Leet Code: https://leetcode.com/anirudh_k97

PROJECT STRUCTURE DOCUMENTATION

There are four important directories:
1) models
2) Dao
3) Services
4) Routes

MODELS/ENTITIES/TABLES

The models directory consists of 4 Entities with following attributes:

1) Users: #user_id, name, phone, email
2) Credentials: *user_id, password
3) PersonalContacts: *user_id, name, phone, email
4) SpamNumbers: *user_id, phone

DATA ACCESS

The Dao directory consists of 4 JS files.
Each file performs different Dao operations on their respective Models/tables.

1) userDao.js – Performs Dao operations on Users table
2) credentialDao.js - Performs Dao operations on Credentials table
3) contactsDao.js – Performs Dao operations on PersonalContacts table
4) spamDao.js – Performs Dao operations on SpamNumbers table

FUNCTIONALITY

Functionality categorized under 3 different heads.
They are implemented in Services directory.

1) Profile Functionality (/profileService)

a) Register new User
b) Login existing user
c) Logout
d) Get details of Searched result along with Email from (2a) and (2b)
   - Returns above search results
   - Will return Email if (i) Searched user registered (ii) Searching user in PersonalContacts of Searched user

2) Search Functionality (/searchService)

a) Search with Name:
   - Returns (name, phone, spam likelihood)
   - Searches on Users and PersonalContacts

b) Search with Phone Number
   - Returns (name, phone, spam likelihood)
   - Searches first on User. If not found searches in PersonalContacts

3) Spam Functionality (/spamService)

a) Add Spam Number

API ENDPOINTS

Functionality	END POINTS	Request Method	Request Data	Response Data
Register New User	/profileService/registerUser	POST	name, phone, password, email (optional)	Text
Login	/profileService/loginUser	POST	phone, password	Text, “userid” cookie set on the client
Logout	/profileService/logout	GET	-	Text, “userid” cookie cleared
Search With Name	/searchService/searchName/:name	GET	Request URL having name in it	JSON. Array of sorted users.
Search with Number	/searchService/ searchPhone/:phone	GET	Request URL having phone number in it	JSON. Array of users.
Get user details with Email	/profileService/getUserDetails	POST	JSON body with Search result	JSON. User details along with Email.
Add Spam Numbers	/spamService/addNumber	POST	JSON body with spam_number	Text.

ROUTES

Routes have been categorized based on functionality. (Routes Directory)

1) searchRoutes.js : for search functionality
2) userRoutes.js : for profile functionality
3) spamRoutes.js : for spam functionality

DATA POPULATION

Data Directory consists of populateData.js File which is executed when the app launches to populated all tables with data.

SPAM LIKELIHOOD FORMULA

When a number (say X) is searched for spam likelihood

1. spamHits=Number of times X marked as spam by Users
2. (spamHits/Spam table size) gives the spamming tendency relative to other spam numbers. It is scored out of 100.
3. (spamHits/Number of Users registered) gives the spamming popularity among registered users . It is scored out of 100.
4. Total scored out of 200.

SCALABILITY/FUTURE REQUIREMENTS

1. Spam Numbers stored along with userid of person who stores it so that users can check if they already marked a number as spam.
2. UserId used as Primary and Foreign Key instead of Phone Numbers despite Phone Numbers being unique so that User can change Phone Number in the future
   a) No need to update PersonalContacts table when Phone Number changes
   b) No need to update SpamNumbers table when Phone Number changes
3. Passwords stored in separate table instead of Users table because it is used only for authentication (Login and Registration)
4. Login and Logout Functionality implemented to make the project wholesome for testing(not explicitly mentioned in the document)

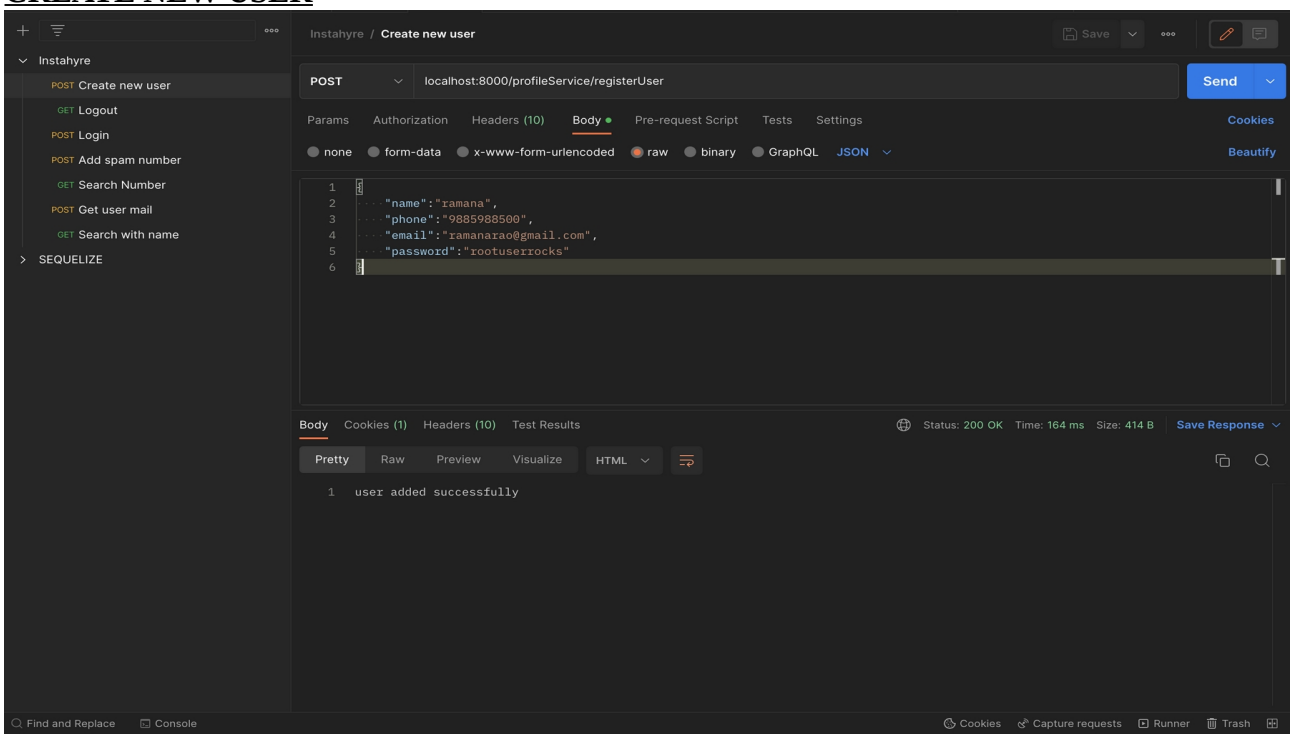
SECURITY FEATURES

1) Services work only if the user is logged in. No unauthorized access.
2) Accessing routes other than API endpoints is restricted.
3) Passwords stored in Hashed format in the database. Any leakage wont give away credentials.

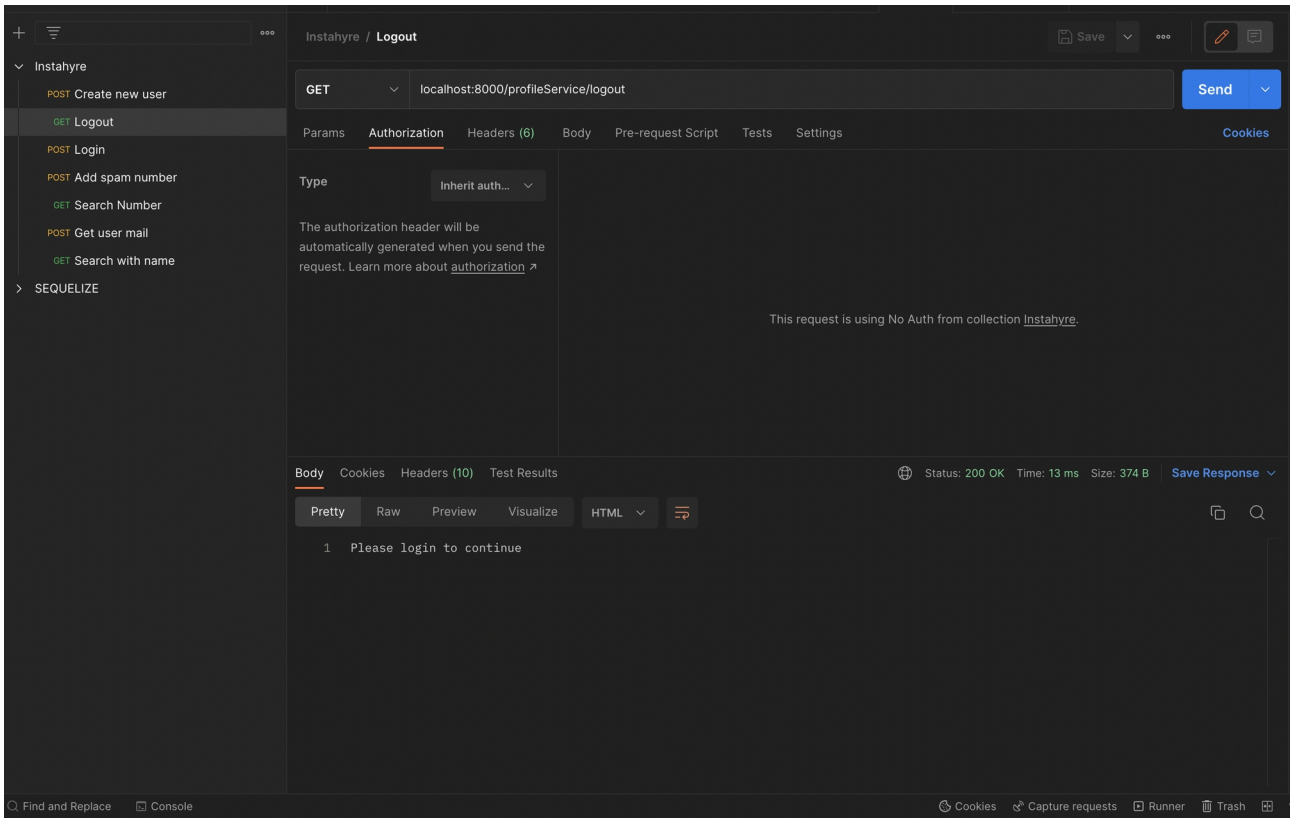
SOME SCREENSHOTS WITH POSTMAN TOOL

These requests have been exported into instahyre.postman_collections.json

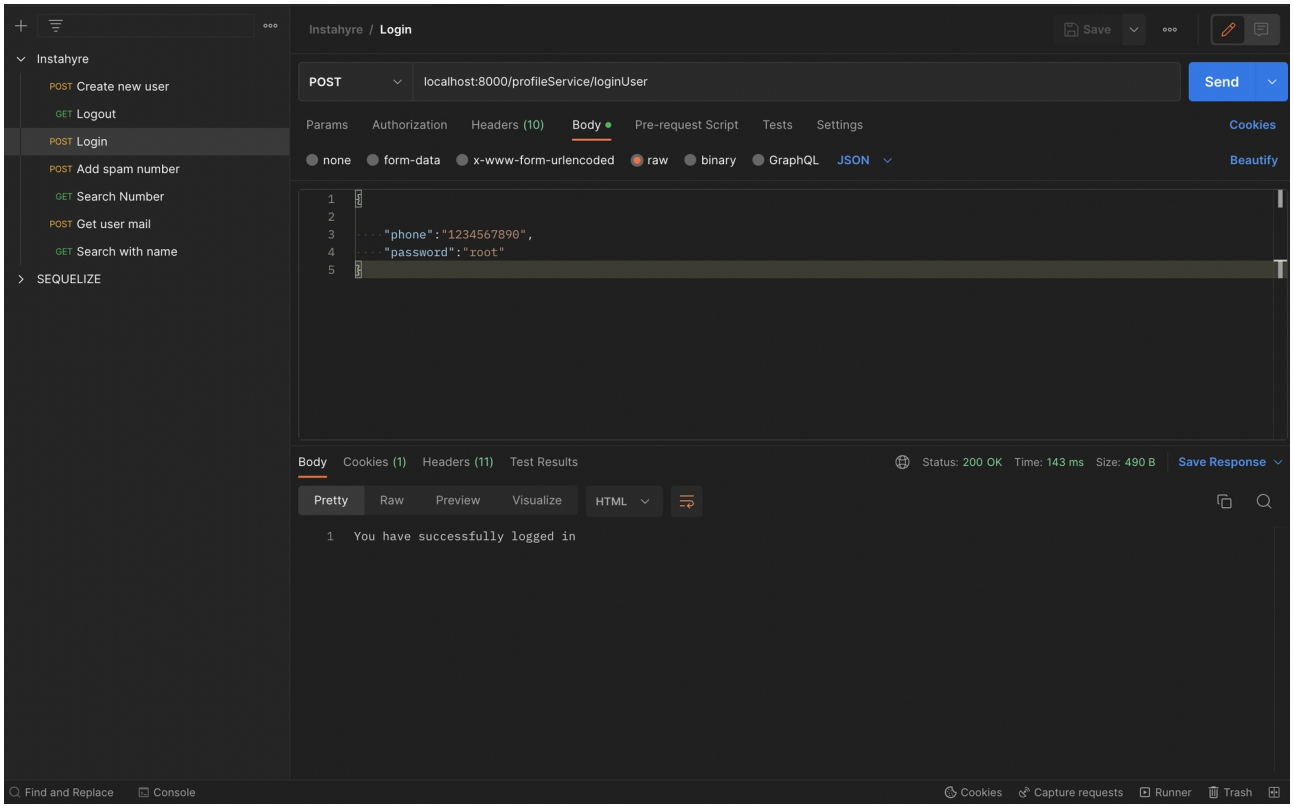
CREATE NEW USER



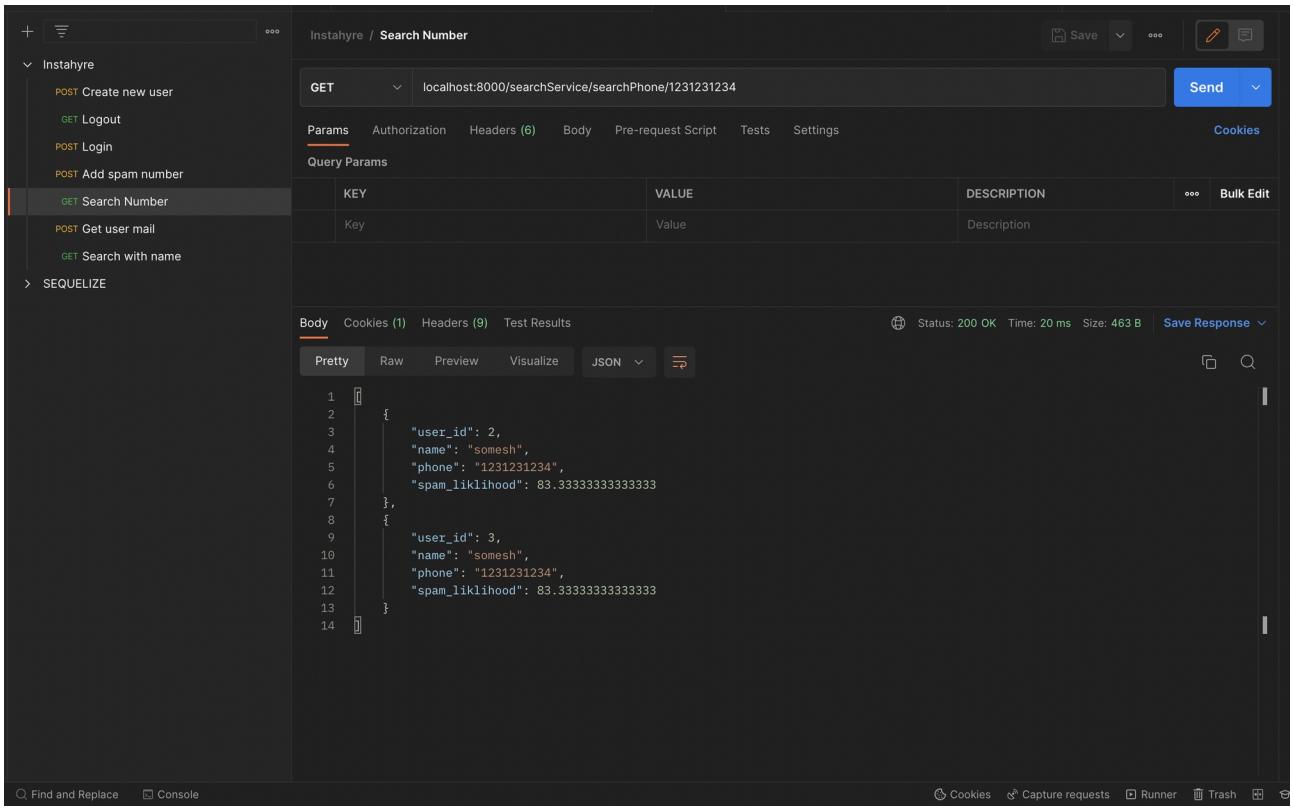
LOGOUT



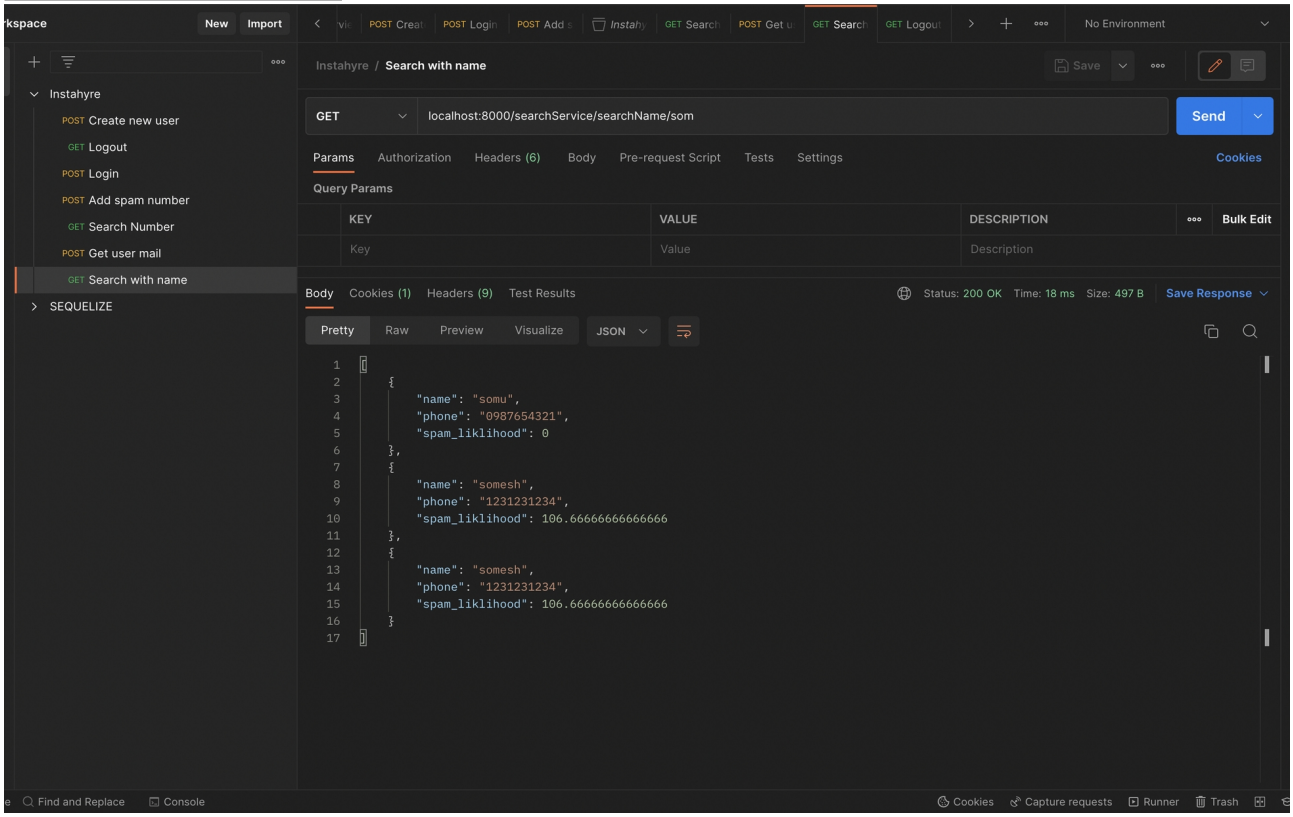
LOGIN



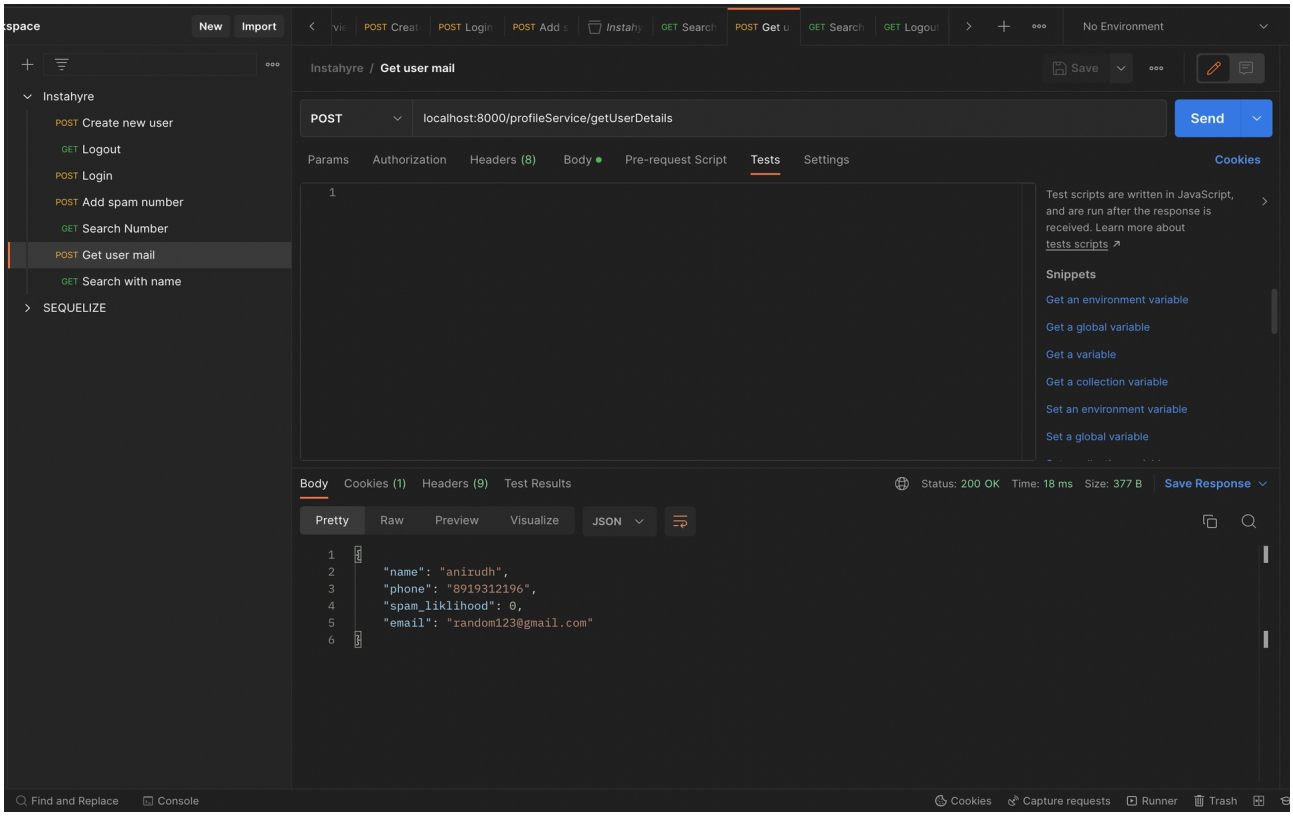
SEARCH WITH NUMBER



SEARCH WITH NAME



GET USER DETAILS ALONG WITH MAILS IF POSSIBLE FROM THE SEARCH RESULT



ADD SPAM NUMBER