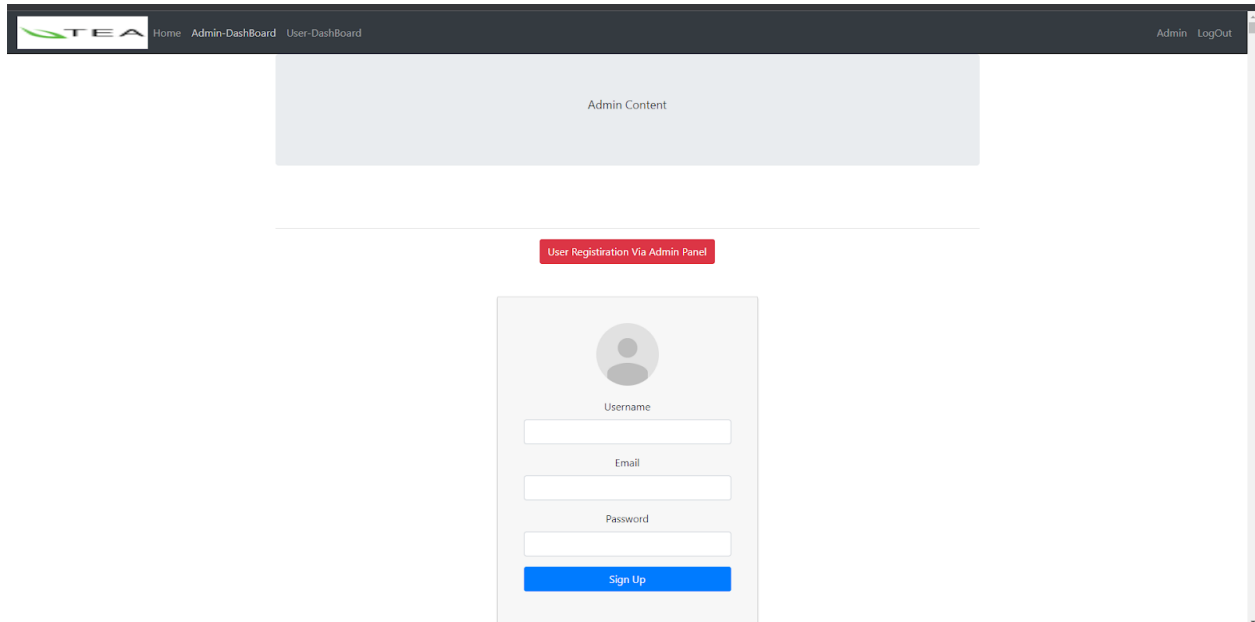


# Tea Networks Weather Data Full-Stack Web Application

Developer: Ashraf Yawar

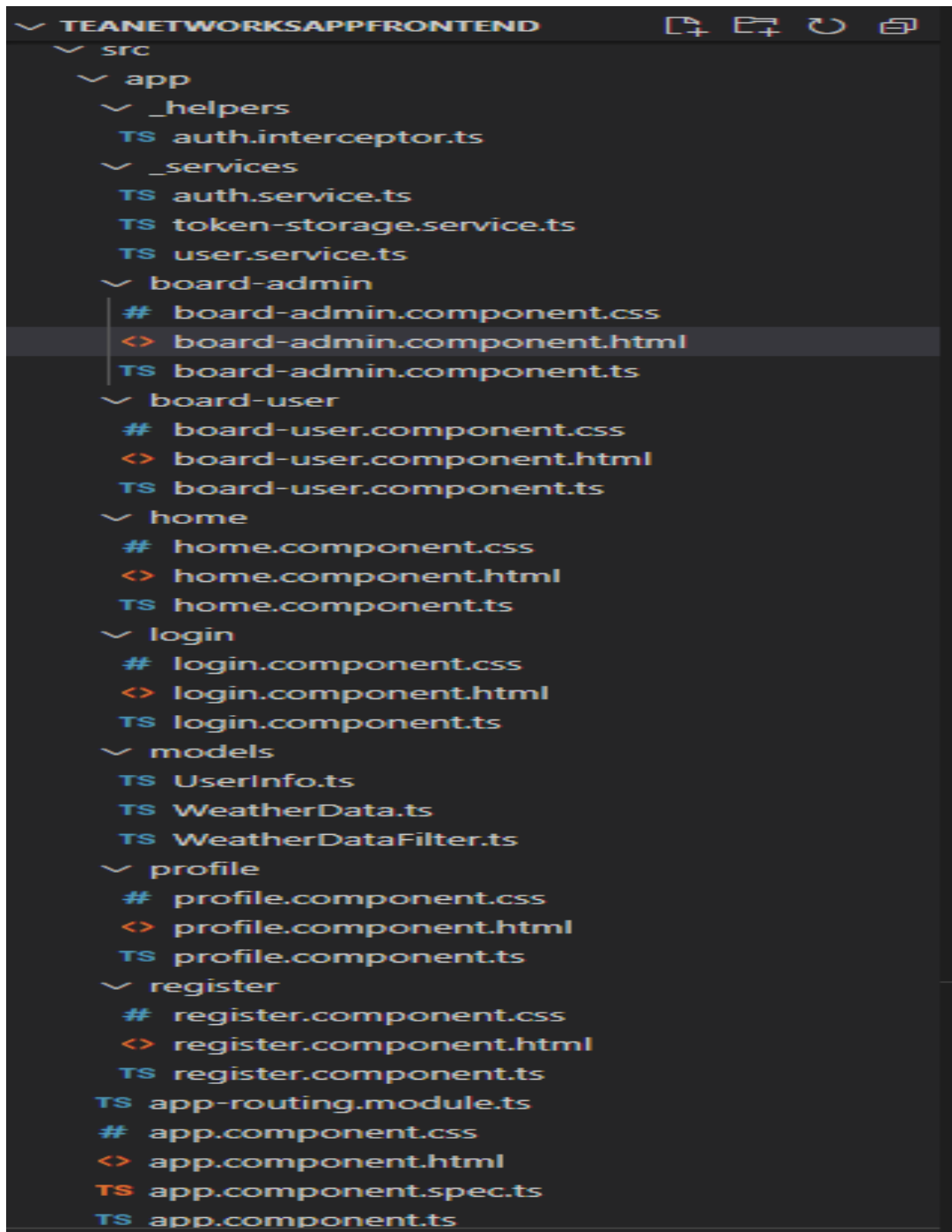
Major: Software Engineer



- This web application has developed considering the MVC design pattern using Angular (for front-end) + Spring-Boot (for back-end) + MySQL (for the database).
- I used **JWT** (Json Web Tokens) and **SessionStorage** of browsers for authentication and authorization.
- There are 2 types of users:
  - **Admins**: can **create**, **delete**, **update**, **list** users, **list** weather data, filter weather data.
  - **EndUsers**: can **view** their own personal info.
- 3 local servers were used in order to develop and test the app:
  - <http://localhost:4200> (to run front-end server)
  - <http://localhost:8080> (to run back-end server)
  - jdbc:mysql://localhost:3306 (to run database server)

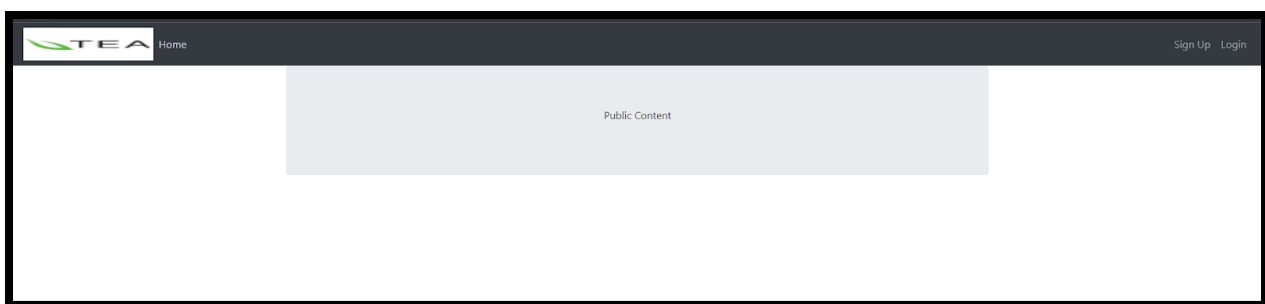
- These 3 server talk to each other and exchange data when requires through http protocol and all the data transmissions are encrypted and all the exposed urls are authenticated.

## FROND-END




- Above is the file structure for the front-end part, it's integrated and configured with the back-end part where it talks to the `http://localhost:8080` and exchanges data with.
- It stores the token in the browser's local db as session storage.
- The App component is a container using Router. It gets user information from Browser Session Storage via `storage.service`. Then the navbar now can display based on the user login state & roles.
- 
- – Login & Register components have form for submission data (with support of Form Validation). They use `storage.service` for checking state and `auth.service` for sending signin/signup requests.
- 
- – `auth.service` uses Angular HttpClient (`$http` service) to make authentication requests.
- – every HTTP request by `$http` service will be inspected and transformed before being sent by `auth-interceptor`.
- 
- – Home component is public for all visitors.
- 
- – Profile component gets user data from Session Storage.
- 
- – `BoardUser`, `BoardAdmin` components will be displayed depending on roles from Session Storage. In these components, we use `user.service` to get protected resources from API (with JWT in HttpOnly Cookie).

## SOME SCREEN SHOTS FROM THE WORKING FRONT-END



## LOGGING IN AS ADMIN WITH VALID CREDENTIALS




Username

Admin

Password


\*\*\*\*\*

Login

 Home Admin-DashBoard User-DashBoard Admin Logout

Admin Content

User Registration Via Admin Panel



Username

Email

Password

Sign Up

## ADMIN PANEL

End-Users List

ID	UserName	Email	Password
16	Admin	admin@gmail.com	\$2a\$10\$1S3F3Gr0S38NFUvddGurkVhJazuVai0zZWlB/TU/S3.pymOAIG
17	EndUser1	enduser1@gmail.com	\$2a\$10\$UUGkNN7aLJe0pwl7Shnu5SliqabYMT.6ETp04z4dKsIVPkKcG
18	EndUser2	enduser2@gmail.com	\$2a\$10\$80NQzQOQY4jxLmDZVuoDsOkCuwq/ycQs8WDKAw2cujrwyjt1IK
20	Ashraf	m.ashraf0016@gmail.com	\$2a\$10\$8U9G8iW5wvtJfK2f0O5eleBZCc7kM6q4Cp1dSMJb7G3a8BFFibSe
21	newuser	newuser@gmail.com	\$2a\$10\$YmbtshUGvmUrJ4LHA0kNuGZCq68gdddPwNhJfRfENpHHyn1Zm

Weather Data For Admin

Fetch All

Filter According To Condition

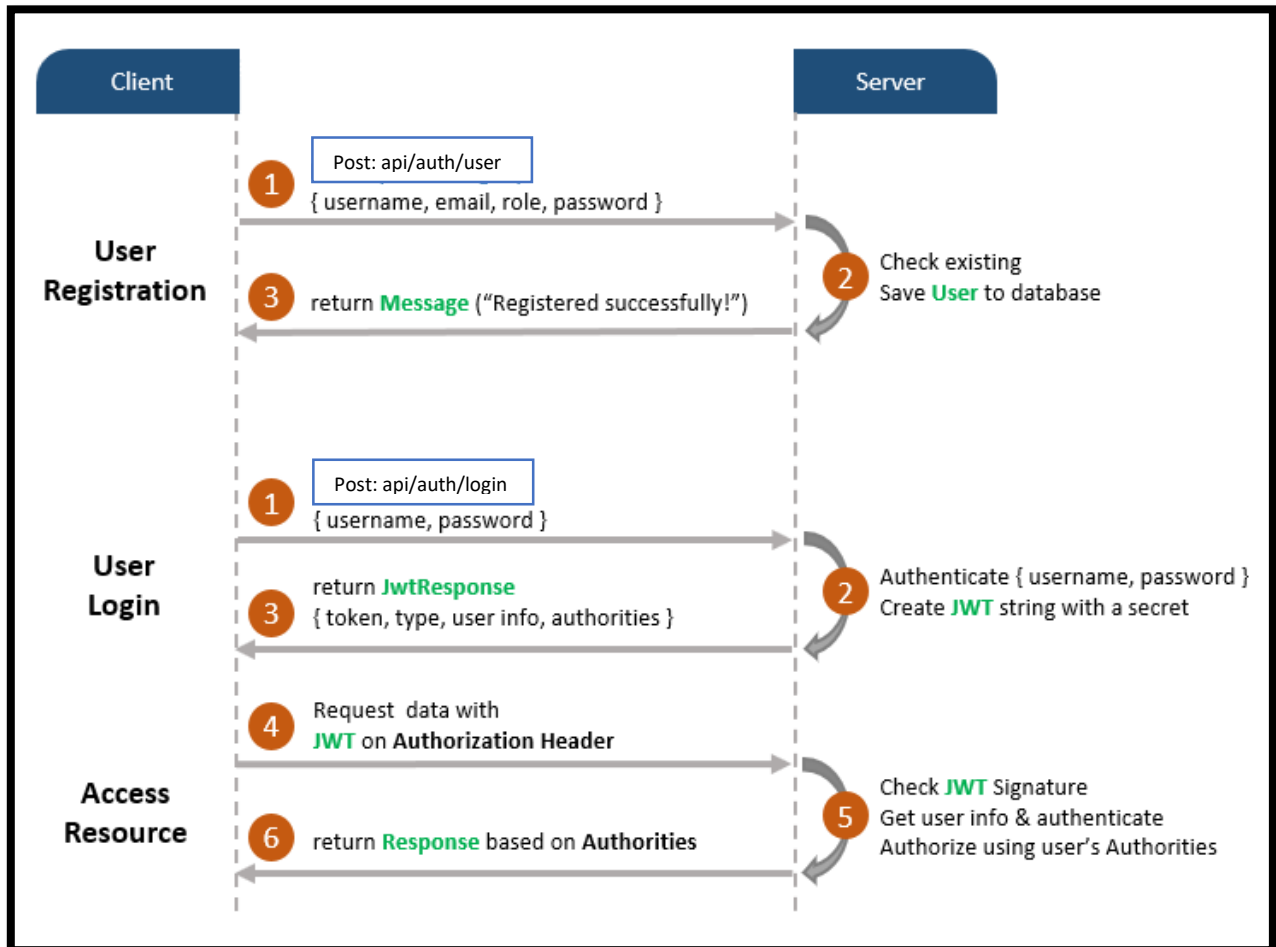
Filter According To Location

Filter According To Average

ID	Condition	Time Range	Location	Average
988388	windy	night	New York	14
988389	sunny	morning	New York	12
988390	rainy	night	Sydney	16
988391	windy	evening	London	28
988392	rainy	morning	Berlin	26
988393	cloudy	afternoon	Paris	22
988394	snowy	morning	Paris	21
988395	snowy	night	Paris	19
988396	windy	afternoon	Paris	10

# BACK-END

User Registration, User Login and Authorization process



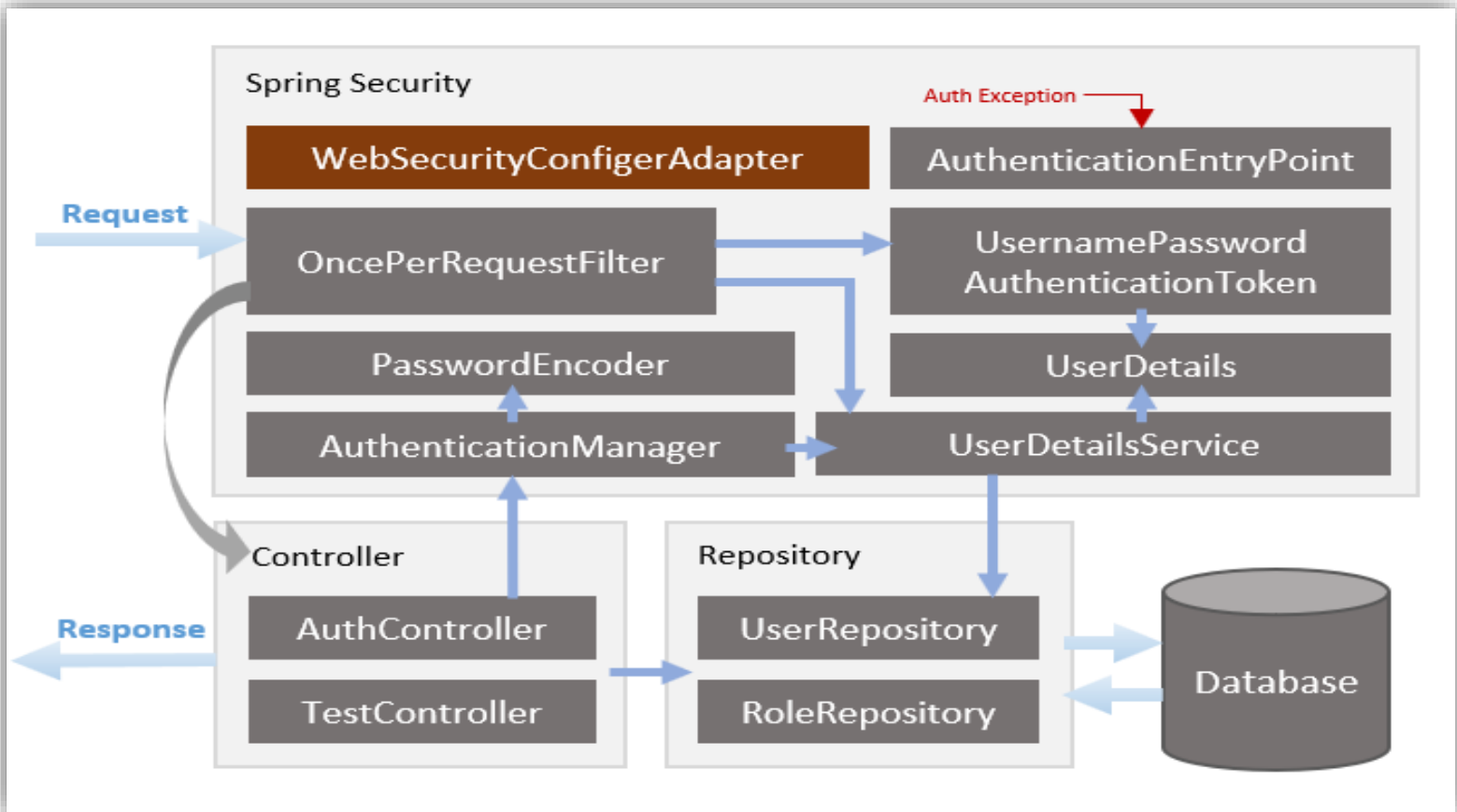
## Configure Spring Datasource, JPA, App properties

```
spring.datasource.url= jdbc:mysql://127.0.0.1:3306/tea_networks?useSSL=false
spring.datasource.username= root
spring.datasource.password= root

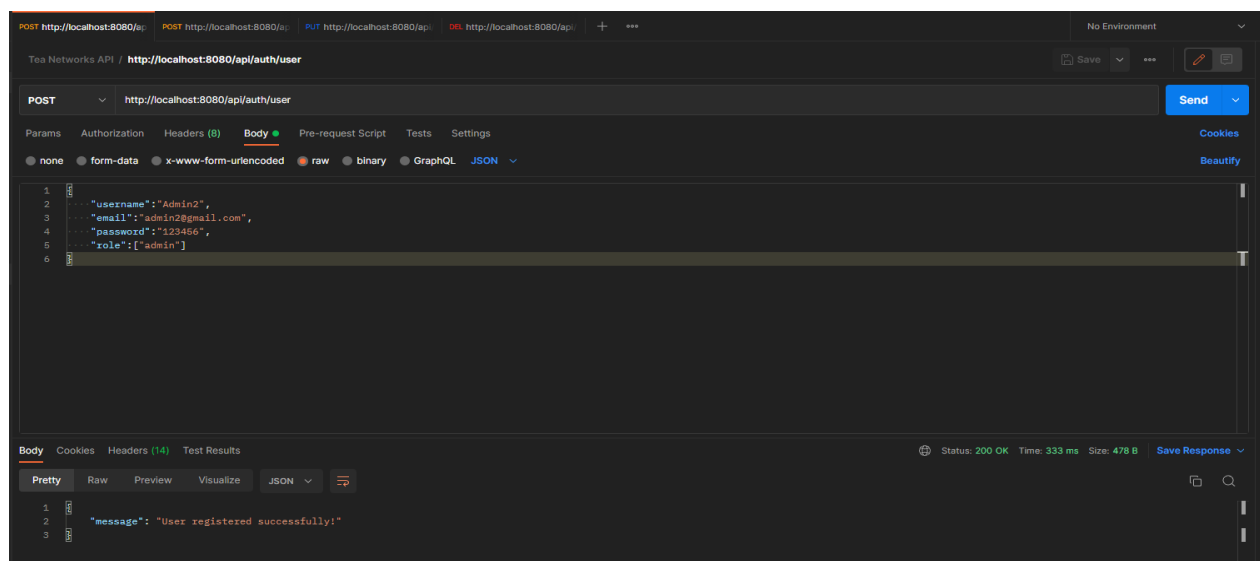
spring.jpa.properties.hibernate.dialect=
org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto= update

# App Properties
teanetworks.app.jwtSecret= teaNetworksSecretKey
teanetworks.app.jwtExpirationMs= 86400000
```

## Spring Boot Server Architecture with Spring Security



## Postman Back-End endpoint tests



POST http://localhost:8080/api/auth/login

Tea Networks API / http://localhost:8080/api/auth/login

POST http://localhost:8080/api/auth/login

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "Admin3",
3   "password": "123456"
4 }
```

Body Cookies Headers (14) Test Results

Status: 401 Unauthorized Time: 161 ms Size: 527 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "path": "/api/auth/login",
3   "error": "Unauthorized",
4   "message": "Bad credentials",
5   "status": 401
6 }
```

POST http://localhost:8080/api/auth/login

Tea Networks API / http://localhost:8080/api/auth/login

POST http://localhost:8080/api/auth/login

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

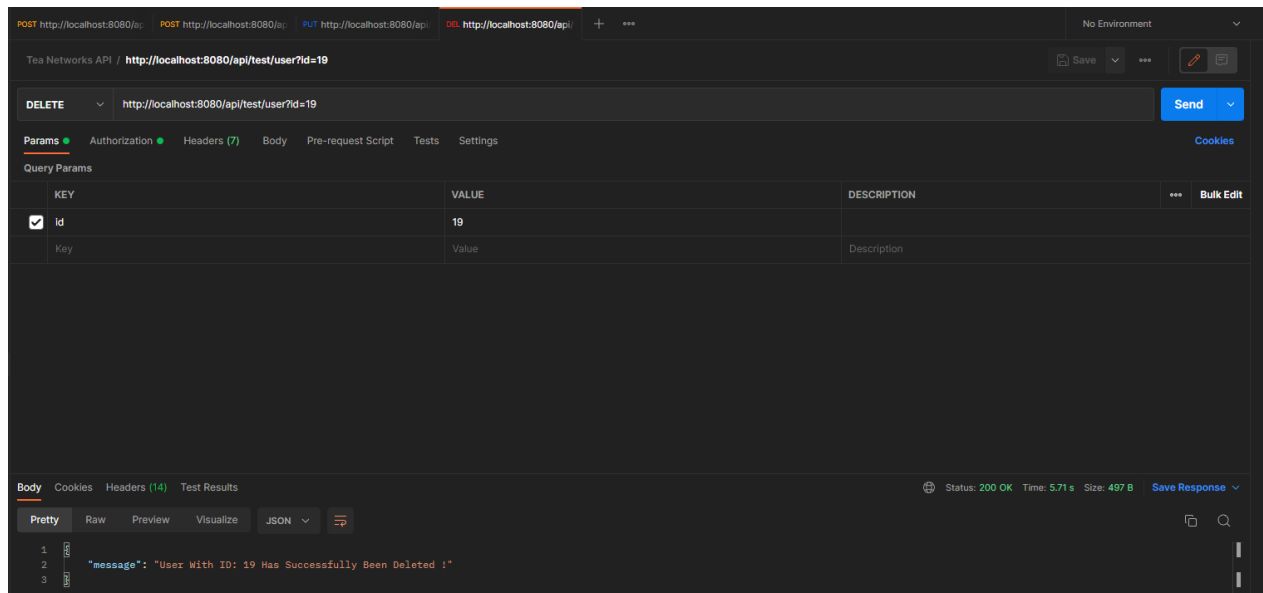
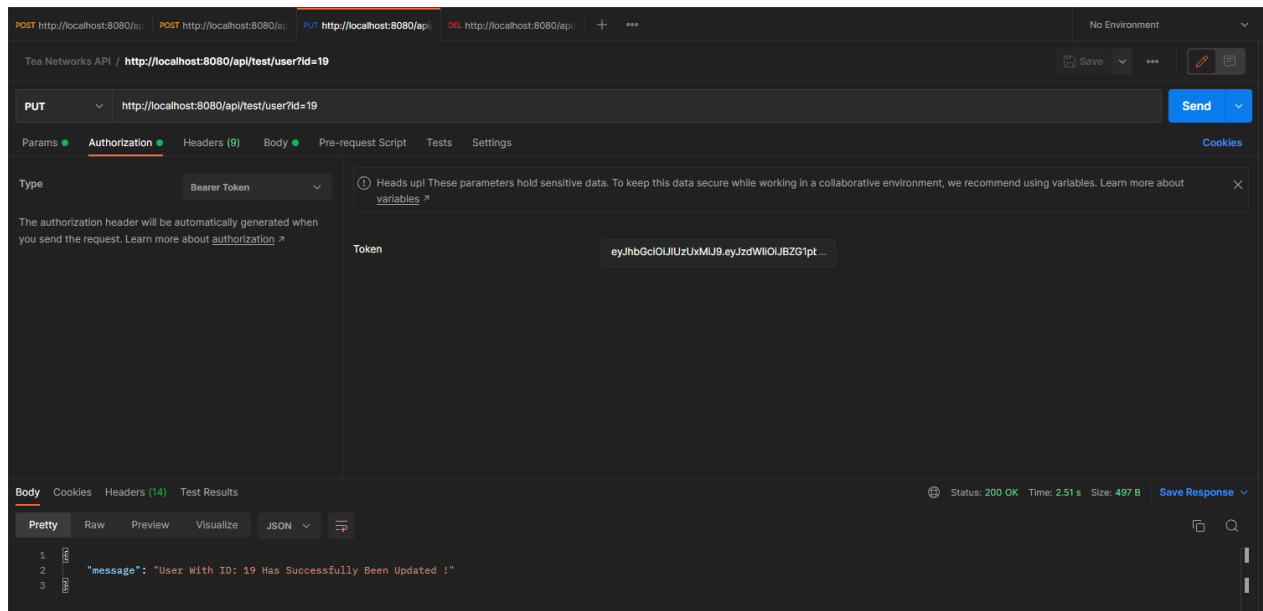
```
1 {
2   "username": "Admin2",
3   "password": "123456"
4 }
```

Body Cookies Headers (14) Test Results

Status: 200 OK Time: 154 ms Size: 727 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 22,
3   "username": "Admin2",
4   "email": "admin2@gmail.com",
5   "roles": [
6     "ROLE_ADMIN"
7   ],
8   "tokenType": "Bearer",
9   "accessToken": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIyZG1pbjIiLCJpYXQiOiJlZ2NzA3OTg2MTAsImV4cCI6MTY3MDg4NTAxMjB9.1aaI24eJueJXf5uzSyl3f4Xt83oBP_wFNWlSRdyMvMKz9BBHotYZngcPpL1NzWxb_HMD5F6kkcjayV2I_mg"
10 }
```

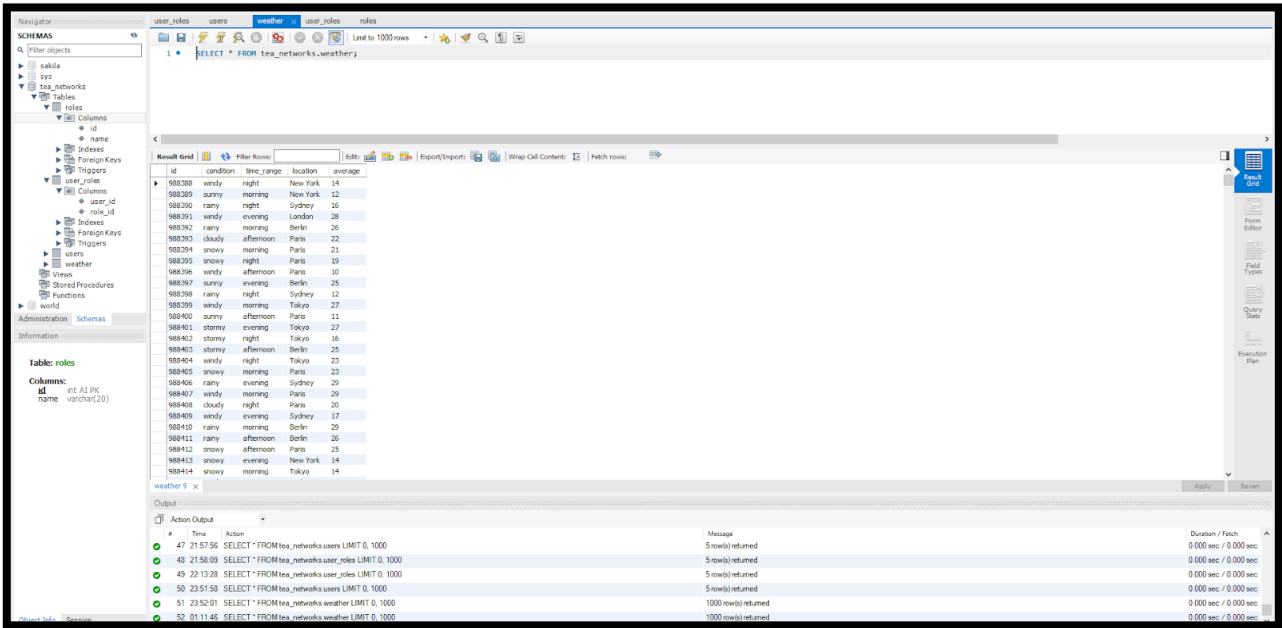
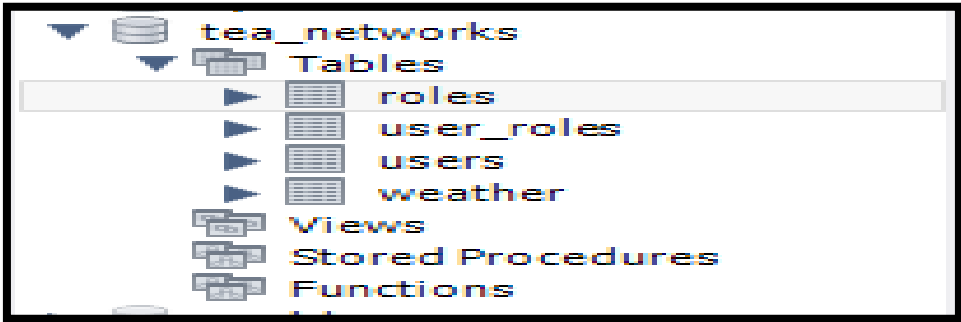


**NOTE:** you can import the `Tea Networks API.postman_collection` file into postman and test the endpoints.



# DATABASE

- The teanetworks local db is simulated as:



I wrote script to generate random weather data and converted it into .csv file, you can find it on the Folder if needed.

- This Application Could be improved in terms of code optimization and UI.
- ChatGPT taking over the tech jobs 😊