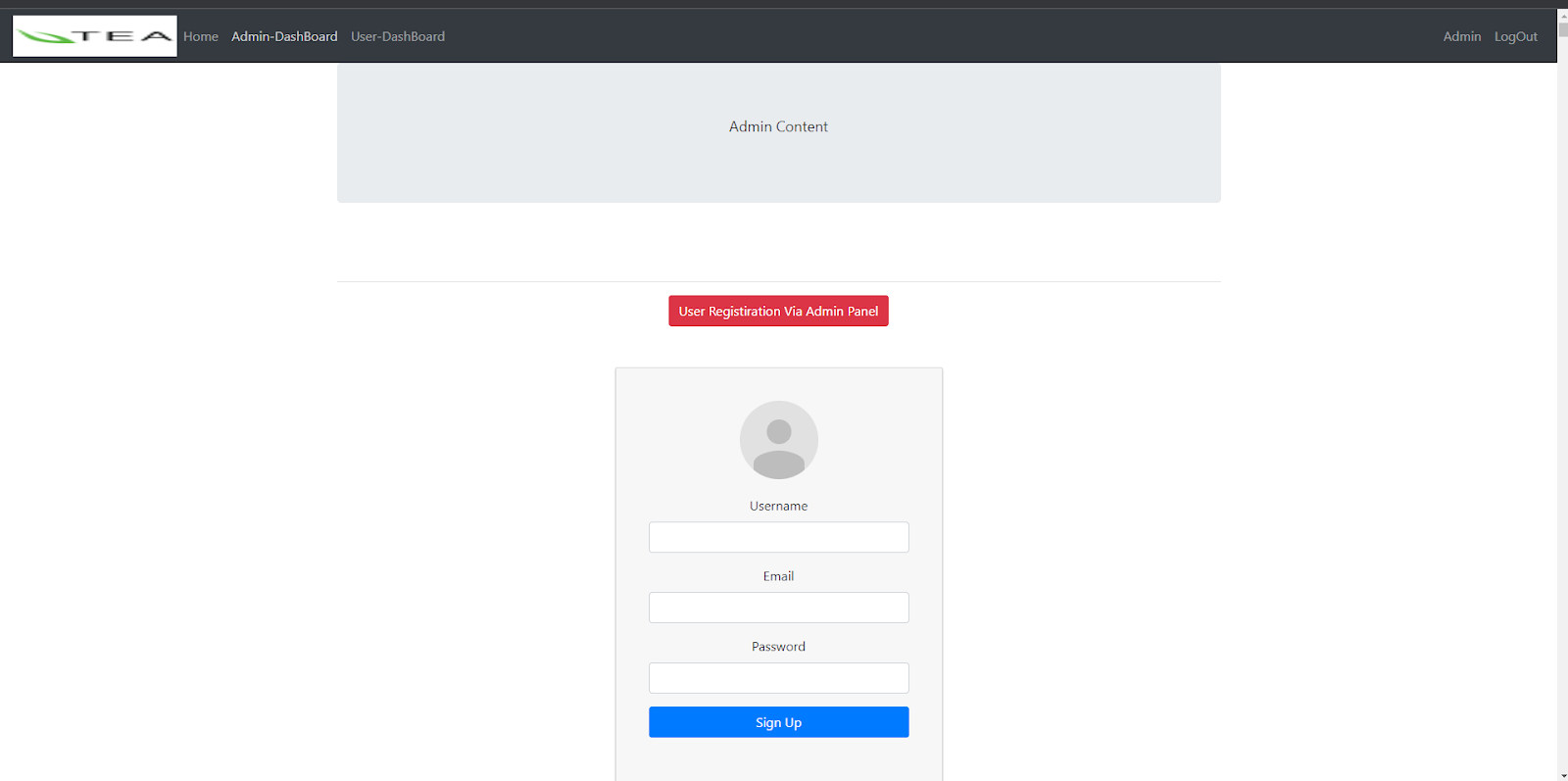
**Tea Networks Weather Data Full-Stack Web Application**

Developer: Ashraf Yawar

Major: Software Engineer

* This web application has developed considerin the MVC design pattern using Angular (for front-end)+ Spring-Boot(for back-end) + MySQL (for the database).

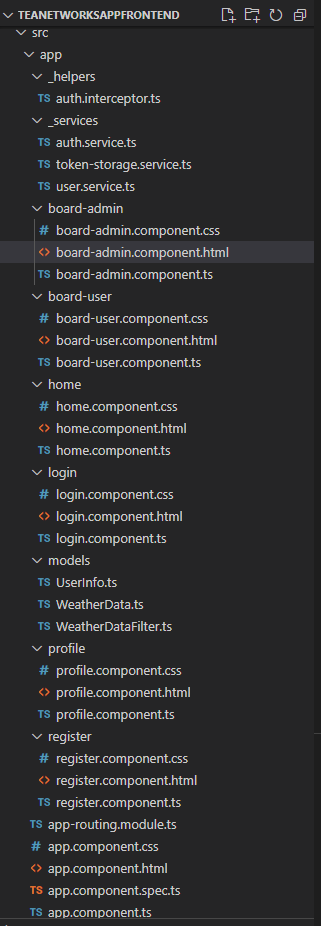
* I used **JWT**(Json Web Tokens) and **SessionStorage** of browsers for authentication and authorization.

* 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 locals servers were used in order to develop and test the app:
  + <http://locahost:4200> (to run front-end server)
  + <http://localhost:8080> (to run back-end server)
  + jdbc:mysql://localhost:3306 (to run database server)

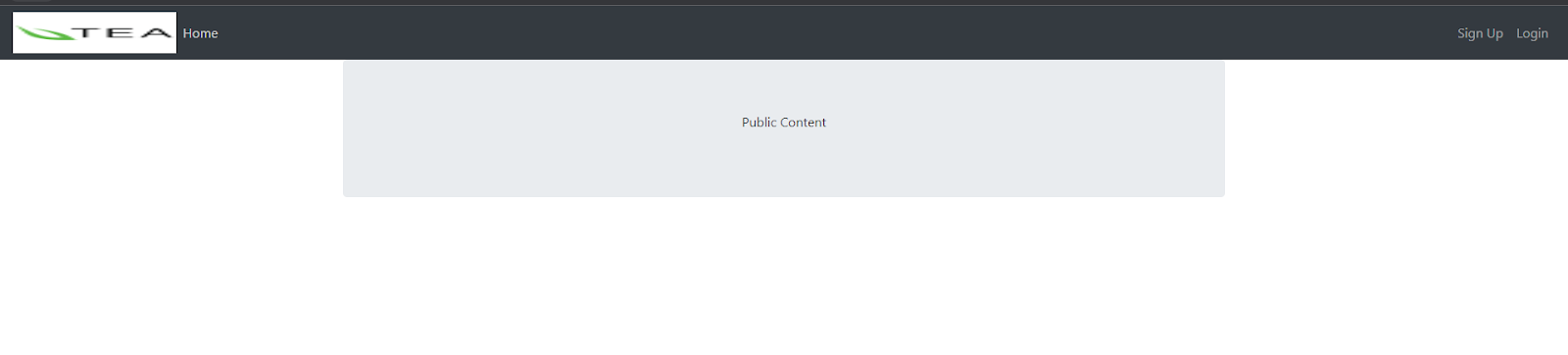
- These 3 server talk to each other and exchange data when requires through http protocol and all the data transmissions are encrypted and all the exposed urls are authenticated.

**FROND-END**

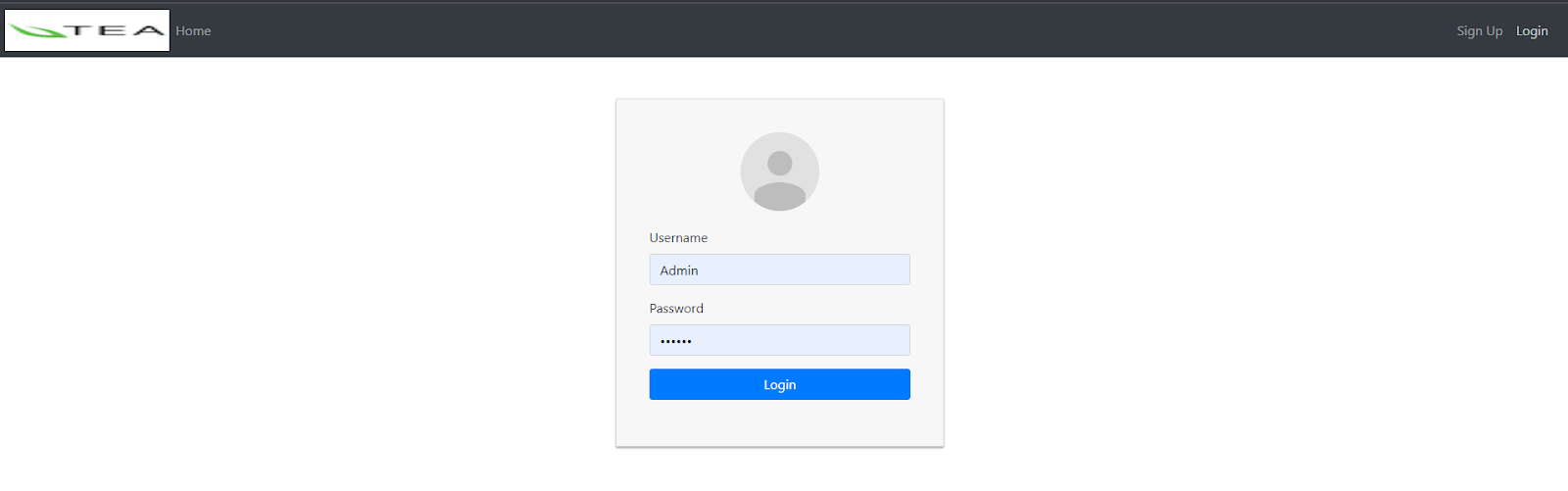


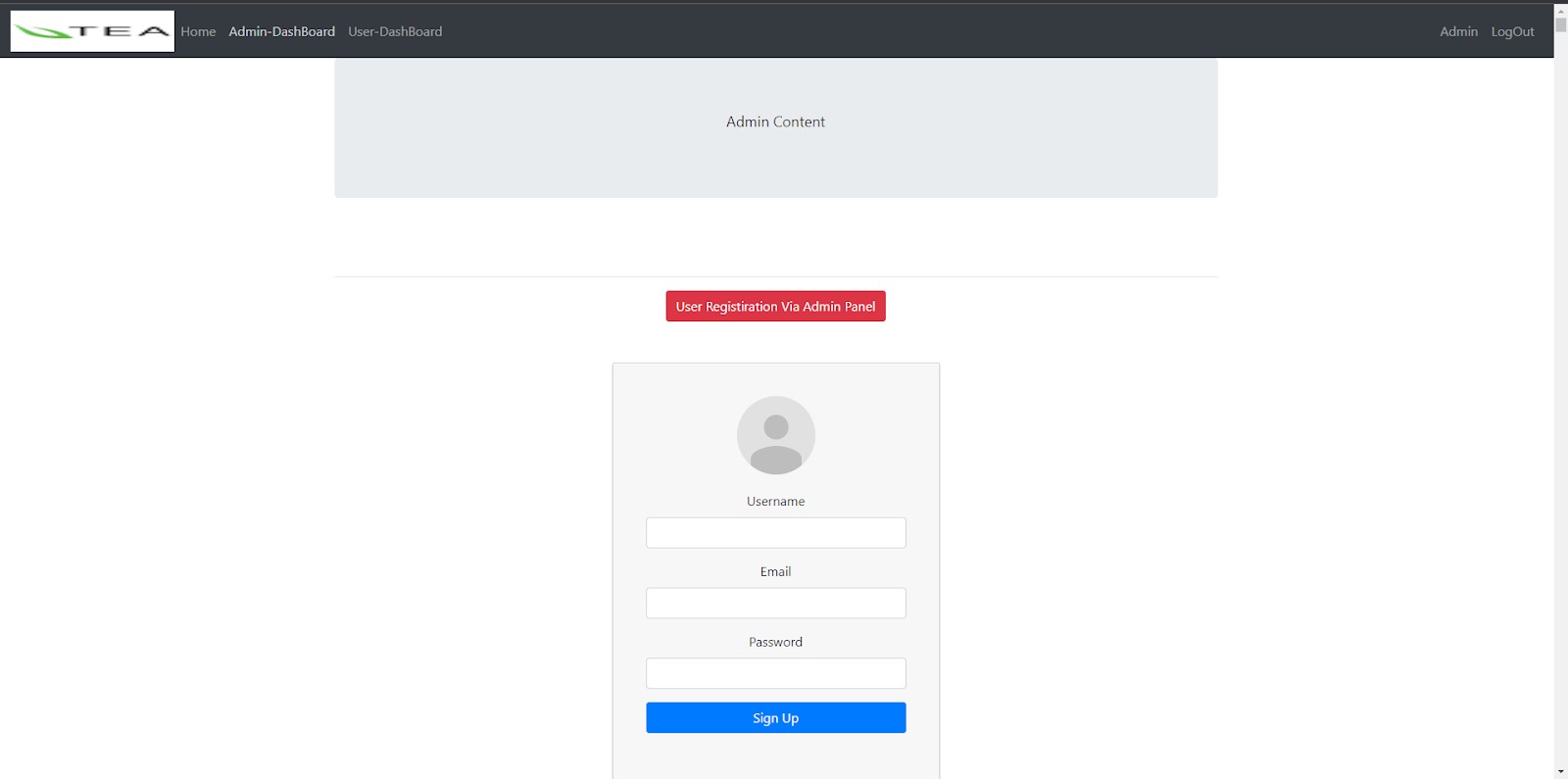
* Above is the file structure for the fron-end part, it’s integrated and configured with the back-end part where it talks to the <http://localhost:8080> and exchanges data with.
* It stores the token in the browsers local db as session storage.
* The App component is a container using Router. It gets user 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 visitor.
* – Profile component get 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).

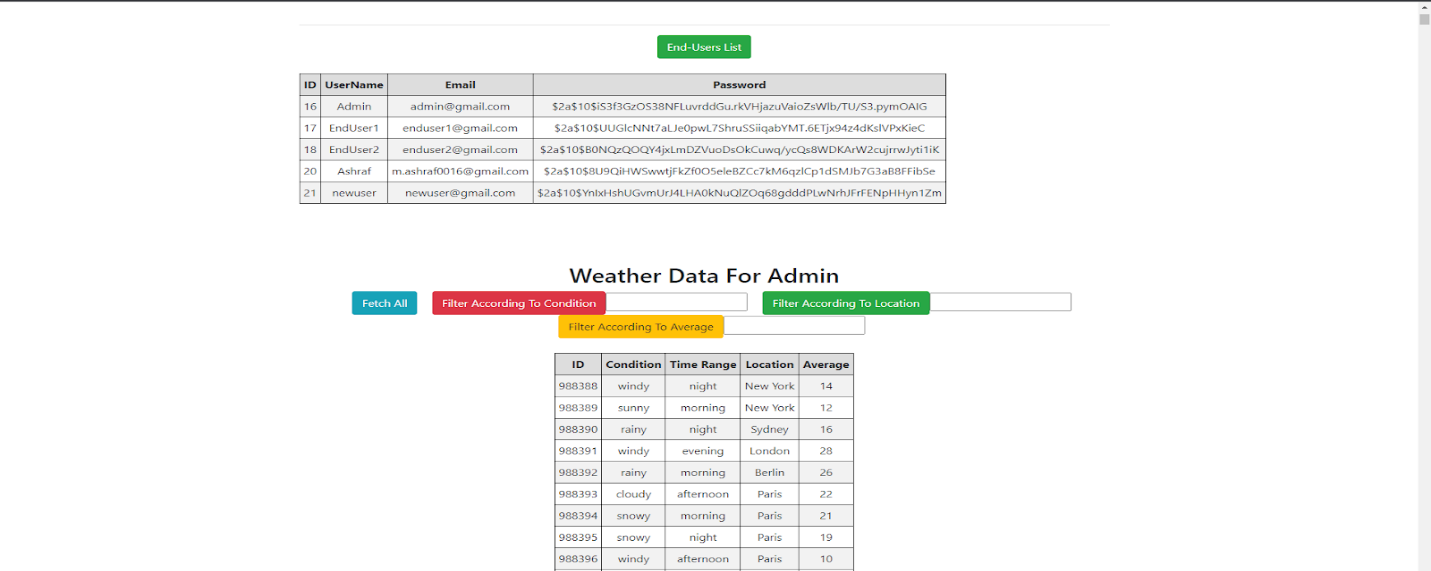
SOME SCREEN SHOTS FROM THE WORKING FRONT-END



LOGING IN AS ADMIN WITH VALID CRIDENTIALS

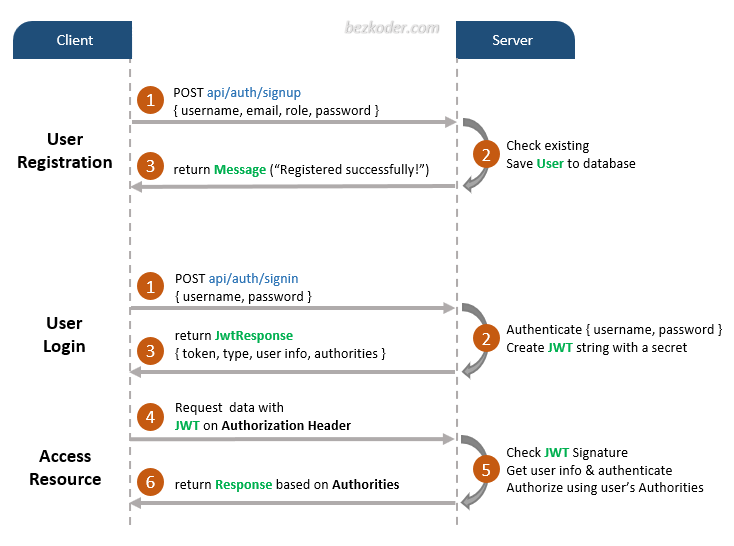


ADMIN PANEL

****

**BACK-END**

User Registration, User Login and Authorization process

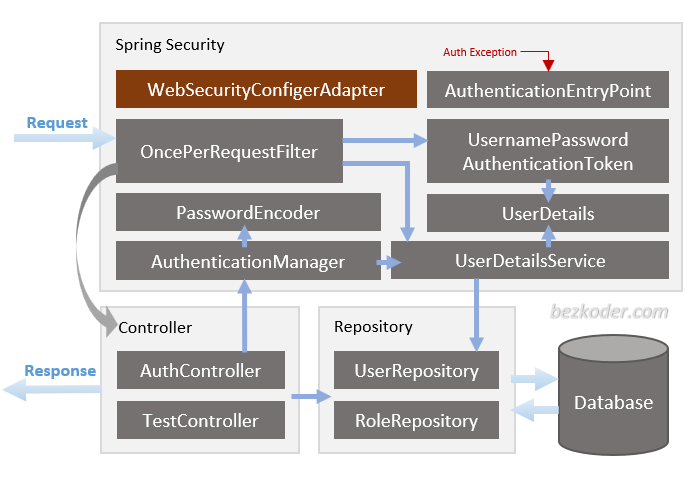


Post: api/auth/login

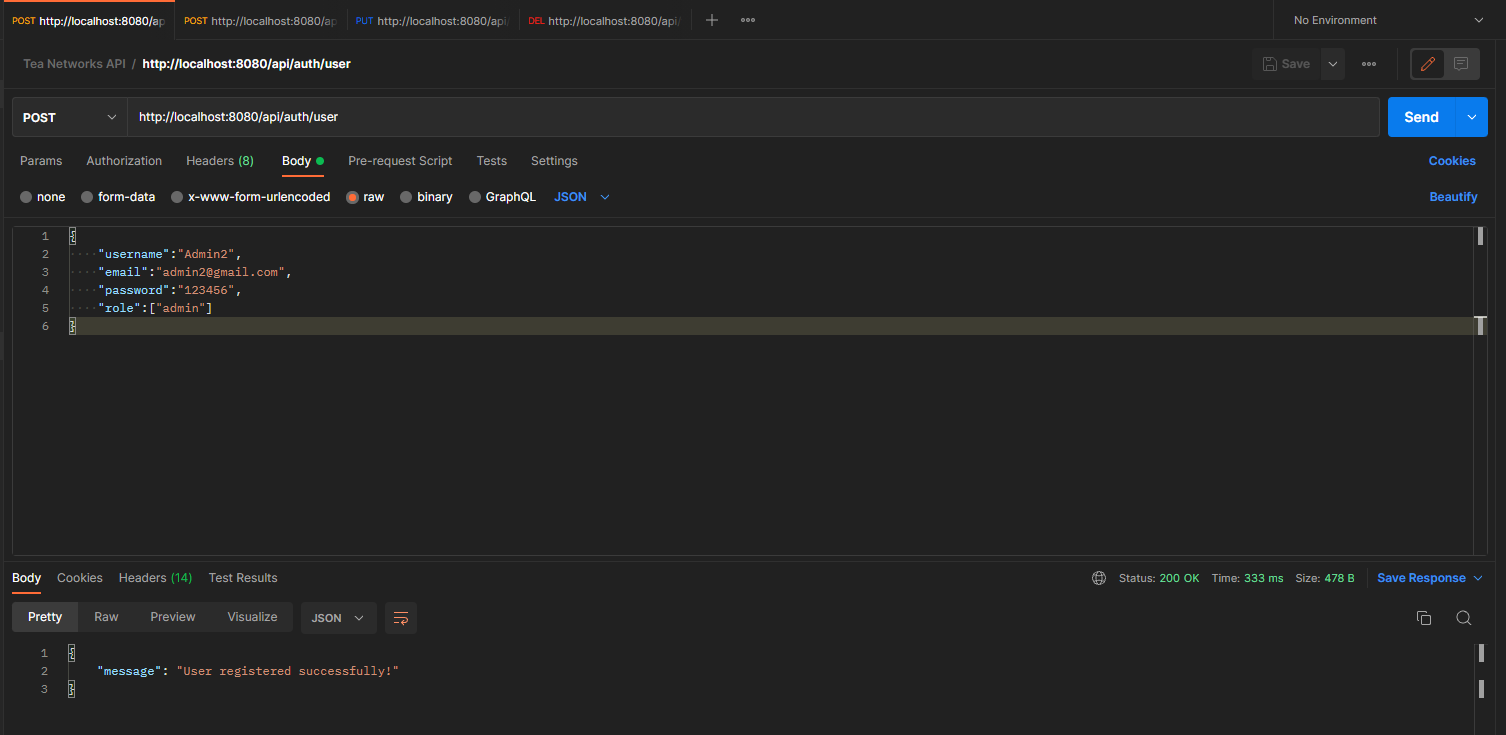
Post: api/auth/user

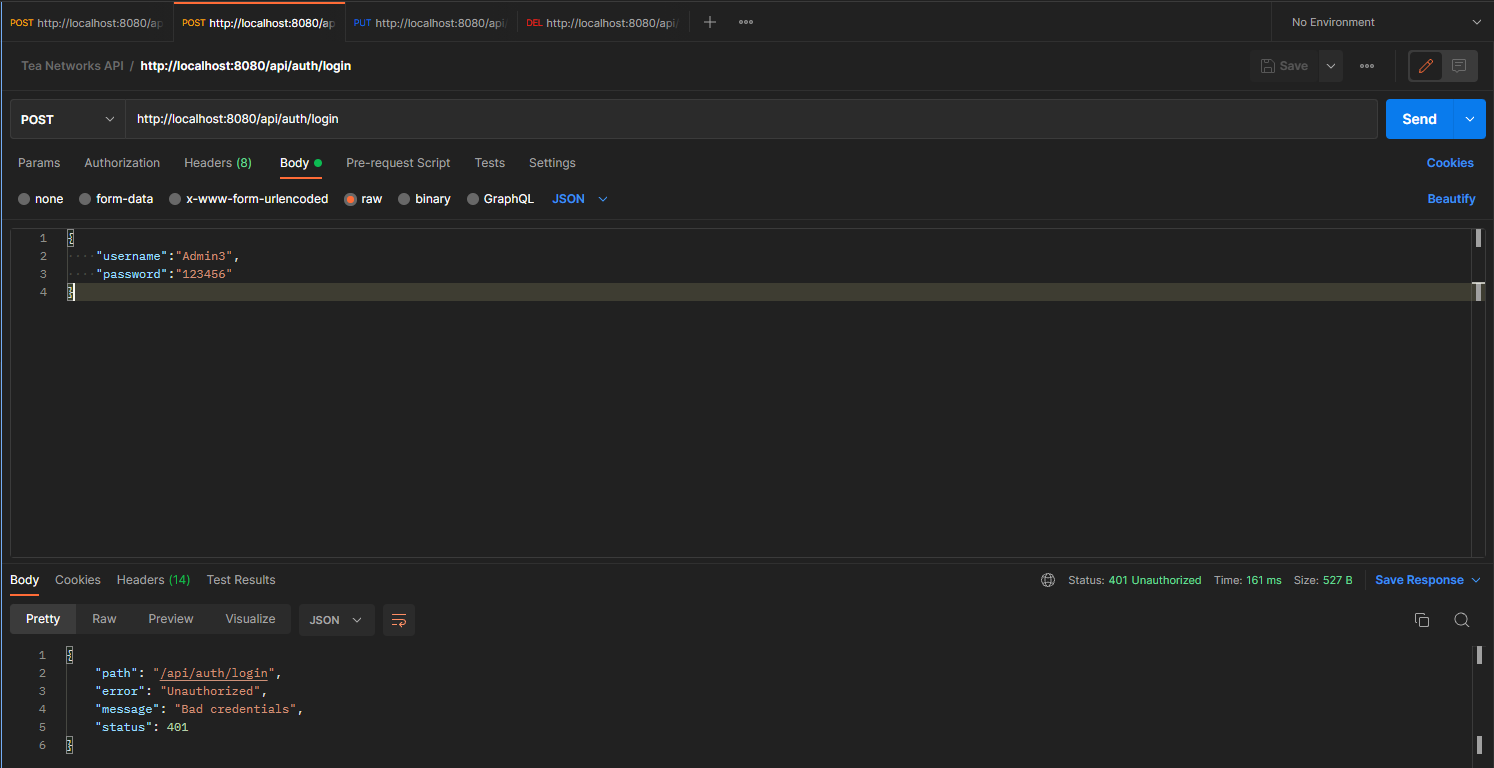
**Configure Spring Datasource, JPA, App properties**

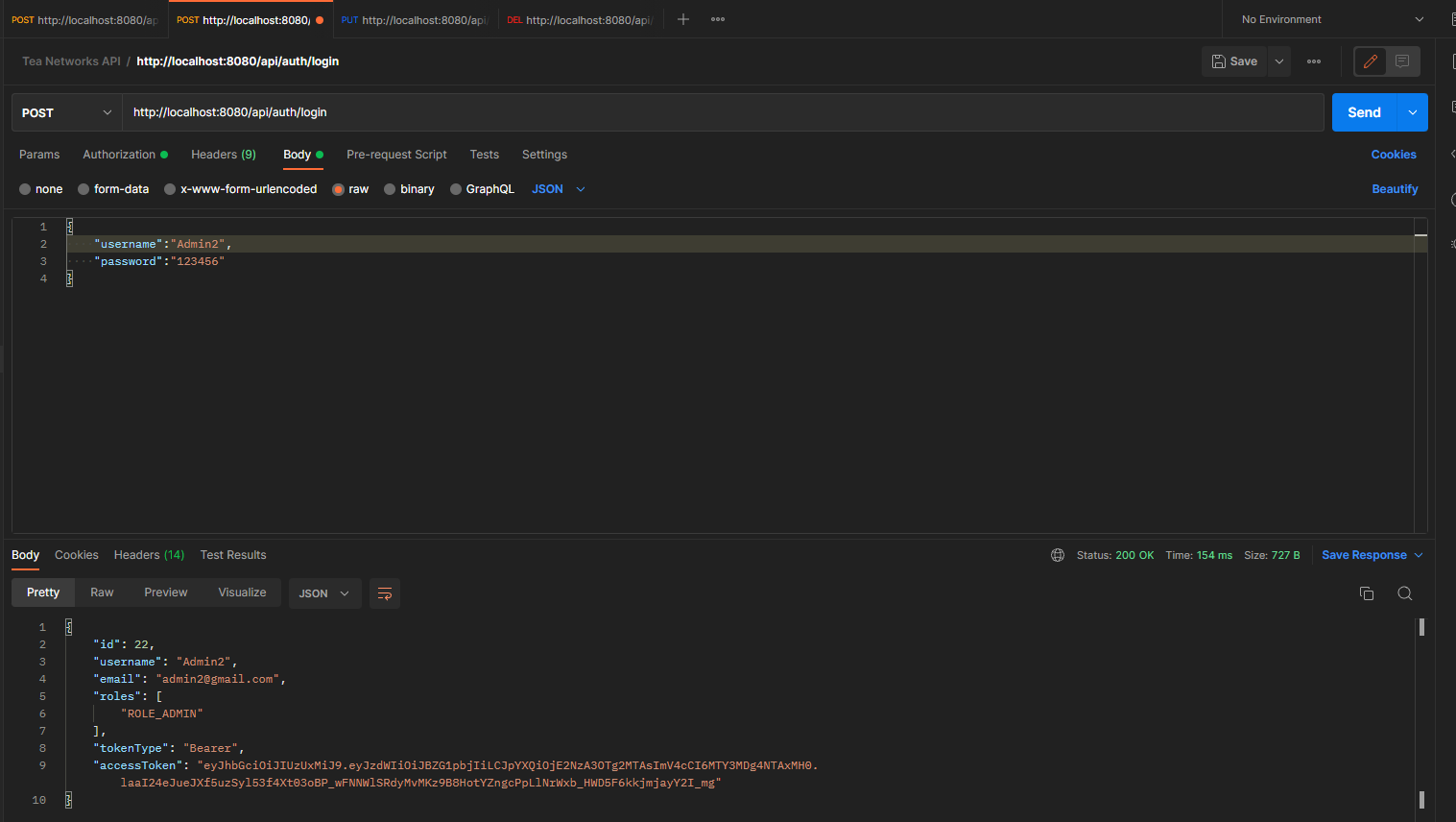
spring.datasource.url= jdbc:mysql://127.0.0.1:3306/tea\_networks?useSSL=false  
spring.datasource.username= root  
spring.datasource.password= root  
  
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect  
spring.jpa.hibernate.ddl-auto= update  
  
# App Properties  
teanetworks.app.jwtSecret= teaNetworksSecretKey  
teanetworks.app.jwtExpirationMs= 86400000

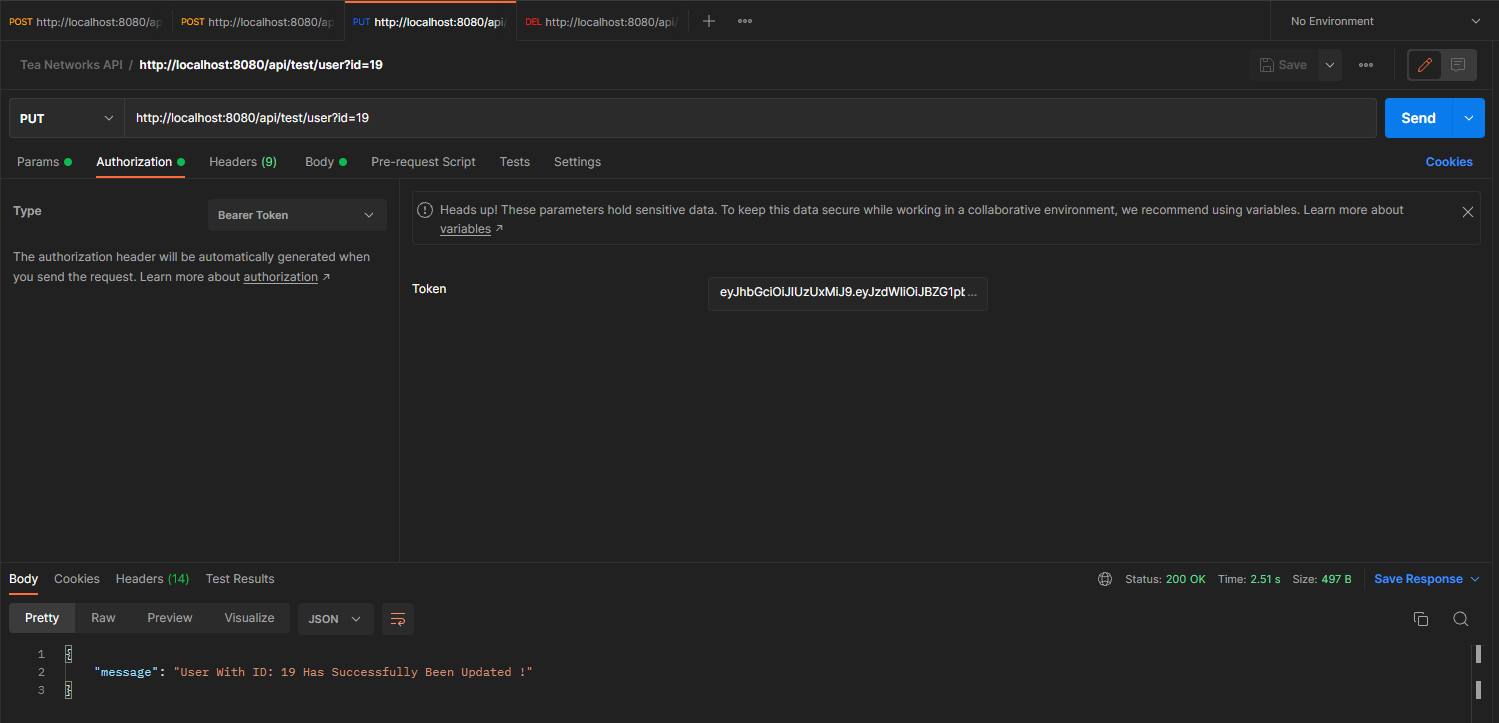
**Spring Boot Server Architecture with Spring Security**

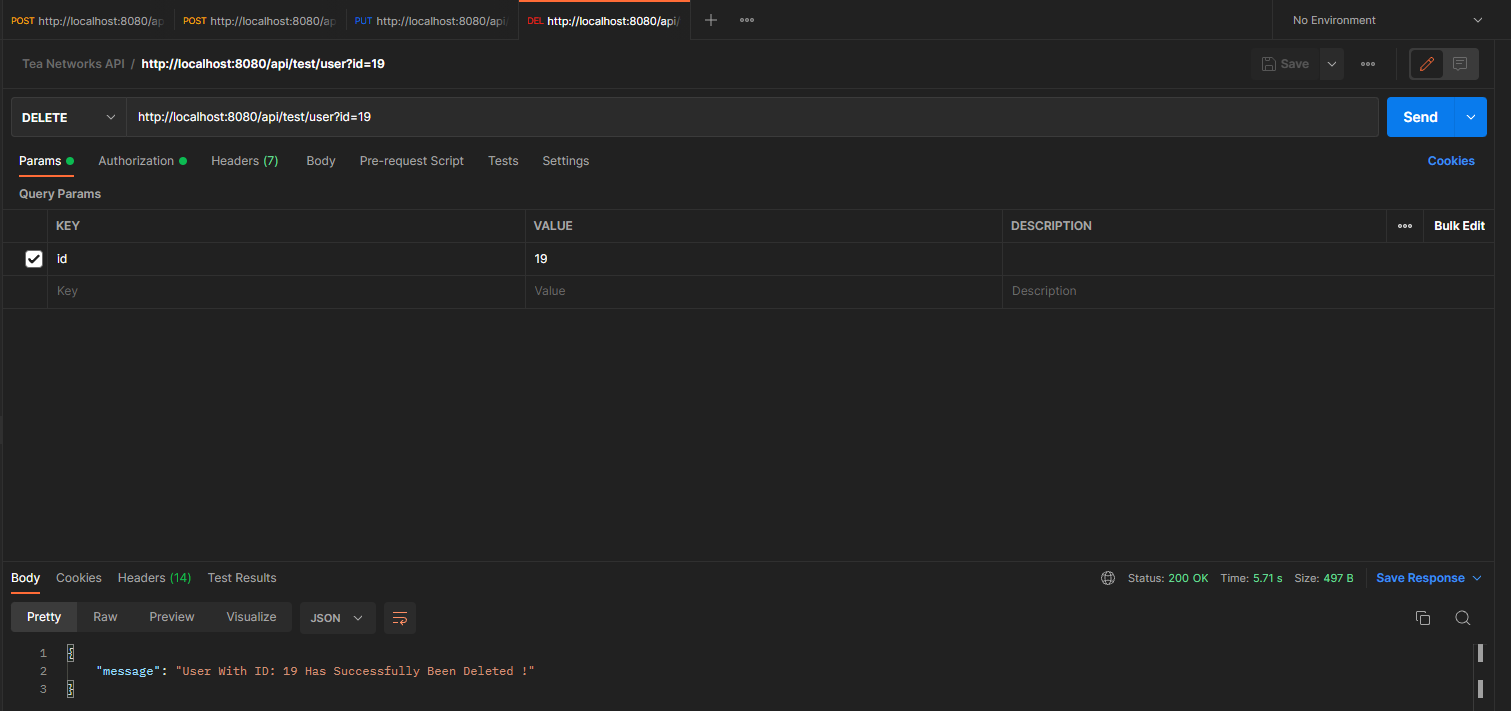
**Postman Back-End endpoint tests**







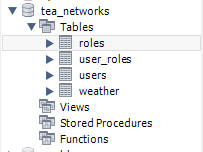


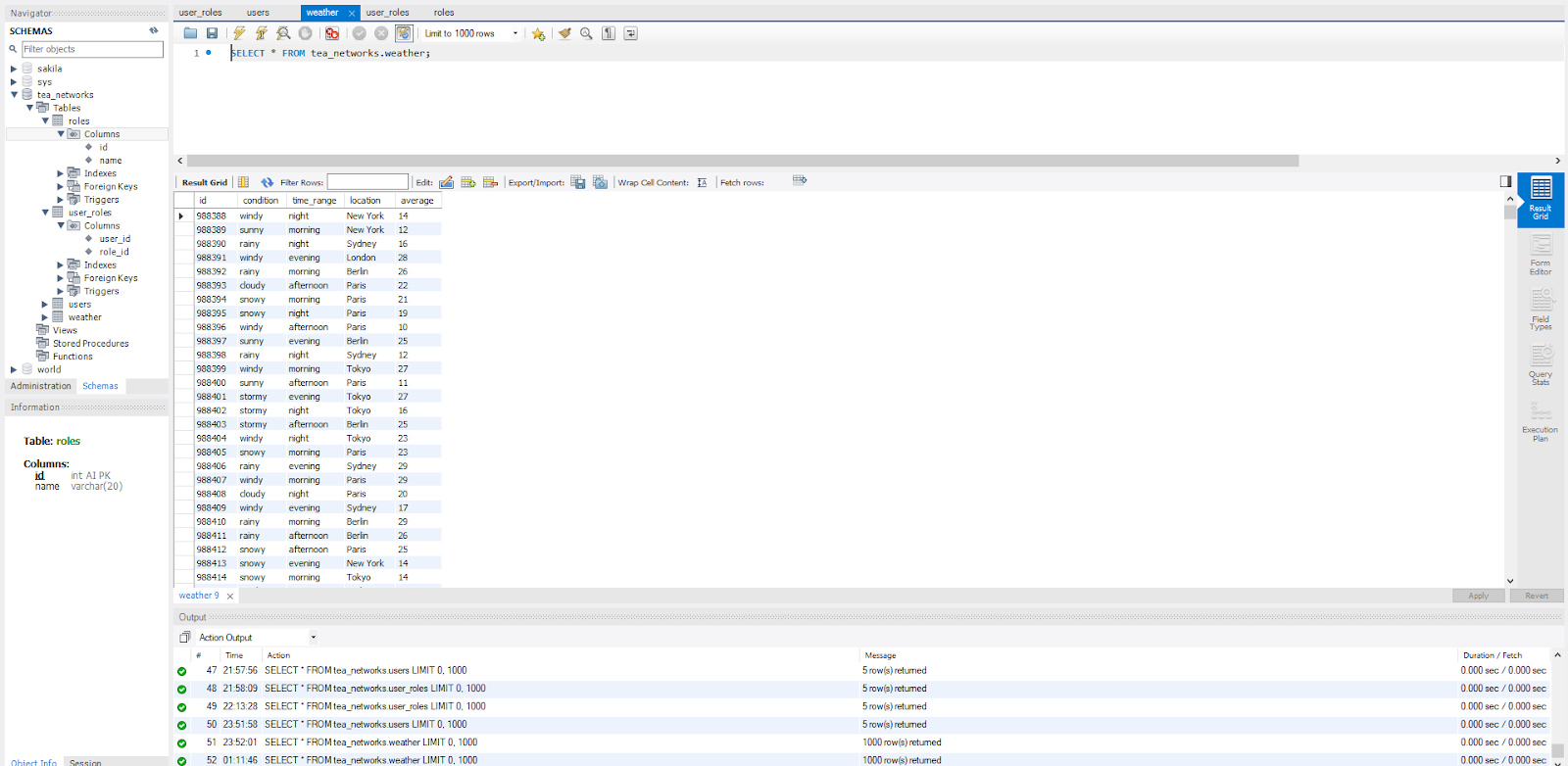


**NOTE: you can import the** Tea Networks API.postman\_collection **file into postman and test the endpoints.**

**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 ☺