USER MICROSERVICE - CLIMATETREE

Overview

The climatetree.wiki website offers a variety of functionalities to its users. The users of ClimateTree can post climate related solutions, research on them, like/comment on the solutions/drawdowns posted by other users and much more. The administrators and moderators of this site are keen on providing helpful content on the site to keep spam at bay. They monitor all the content on the site and delete inappropriate content.

With all this functionality being offered, it is necessary that a user registers to ClimateTree.wiki. A user has two options to register to the site:

1) Login via Facebook
2) Login via Google

Once a user logs in, the user has the privilege of exploring the site in a variety of ways as per the role. For example: if a user is just a registered user, then he/she can post solutions, like or comment others solutions. Similarly, if a user gets promoted as a moderator or admin then there are various other set of privileges assigned to him/her.

Intended Audience

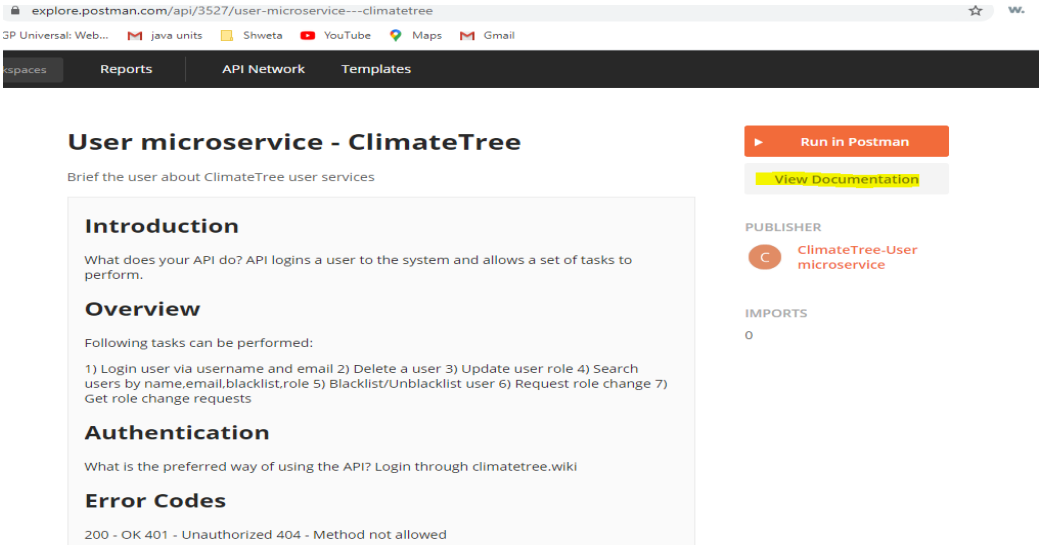
Developers – who wish to learn how user microservice is being implemented and want to scale it further

APIs offered:

Following are the APIs offered by user microservice:

1) Register/Login a user
2) Search users by name, email, blacklisted users, role
3) Delete user
4) Update user role
5) Blacklist/Unblacklist user

Link to documentation: https://explore.postman.com/api/3527/user-microservice---climatetree



Postman Documentation image 1

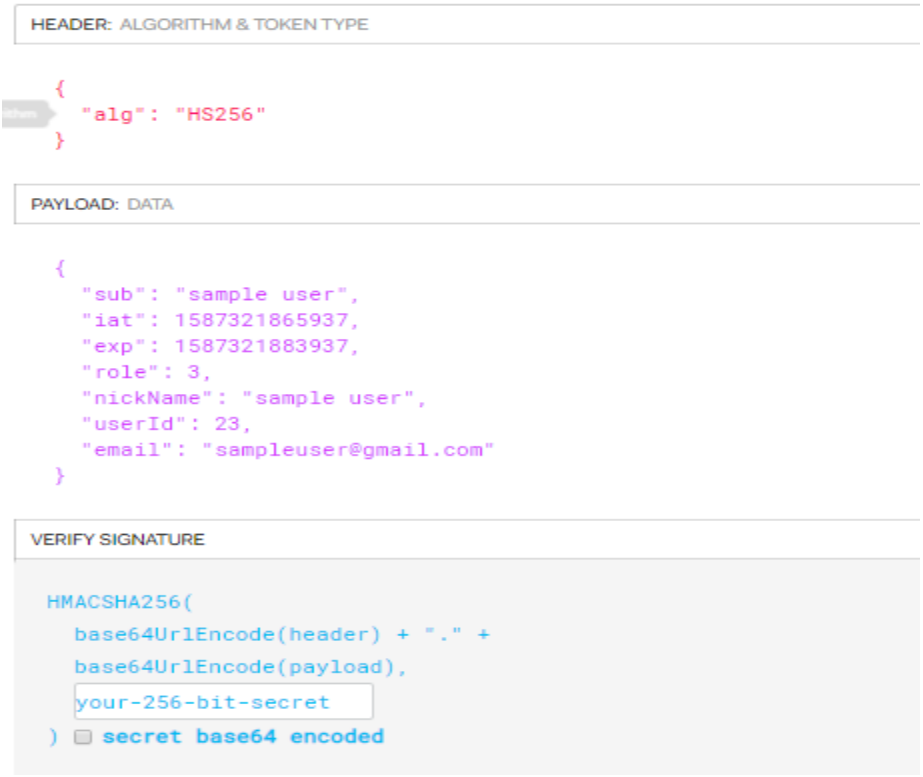
Technologies used:

User microservices used the below tools and technologies:

1) Language - Java SpringBoot Framework
2) Database - PostGreSQL DB
3) Repository - Github
4) Postman for testing

The user microservice follows a token-based approach where we use JWT(JSON web tokens) to secure data transmission within systems.

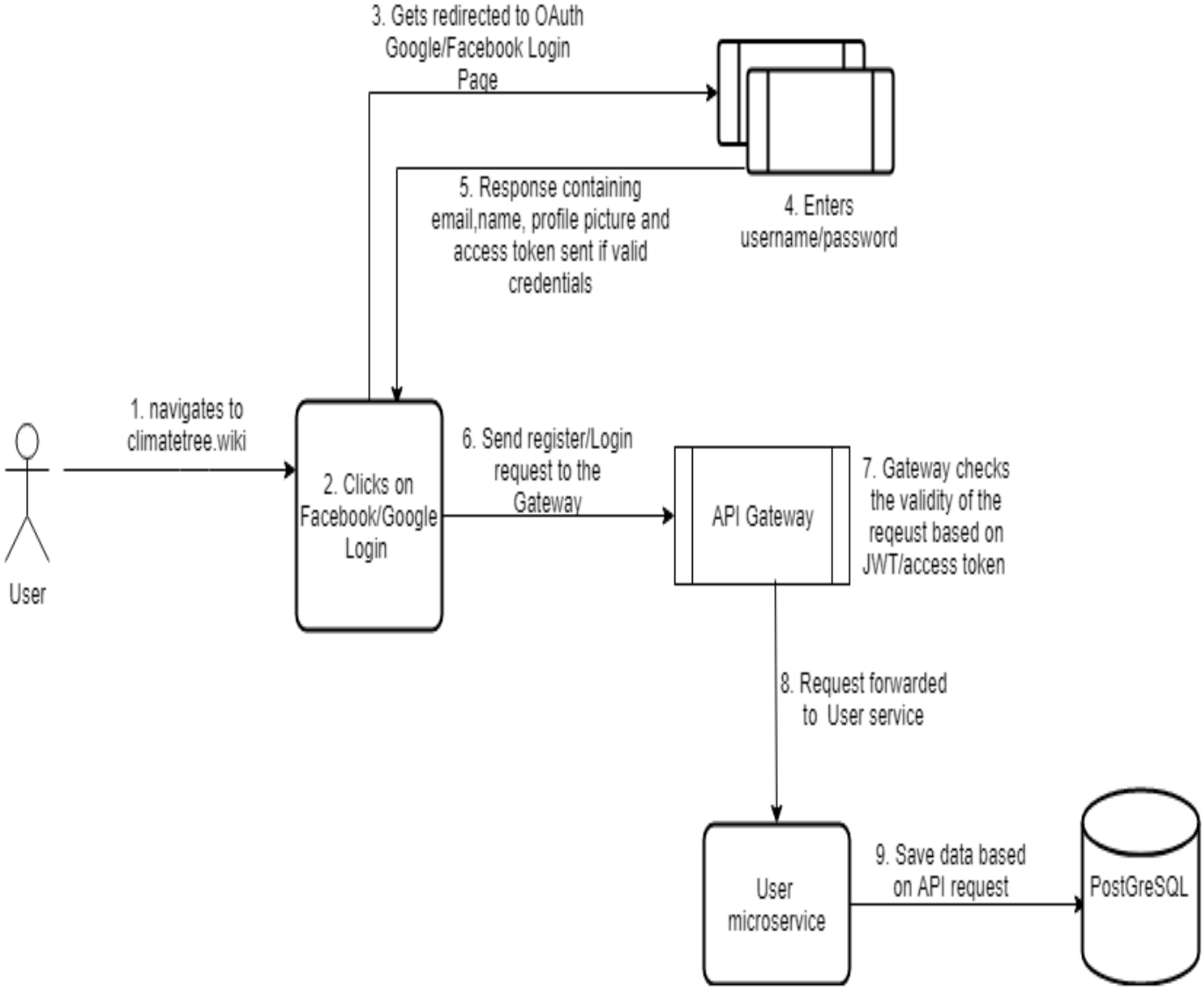
A general decoded JWT of user looks as follows:



Pulling the code from Git:

1) Click the repository link below:
   https://github.com/climatetree/user-microservice
2) Clone the repository
3) Open the user-microservice in your favorite editor
4) You should be good to go with SpringBoot
5) Install pgadmin, postgresql on your system
6) Enter the port number, username, password details of postgre in properties file
7) Run the application
8) It should create 3 tables in postgre – user, role, role_update_request in public schema
9) You can install Postman and test the services as mentioned in the above documentation

User journey:



Roles:

Every user who visits ClimateTree.wiki is assigned one of the below 4 roles:

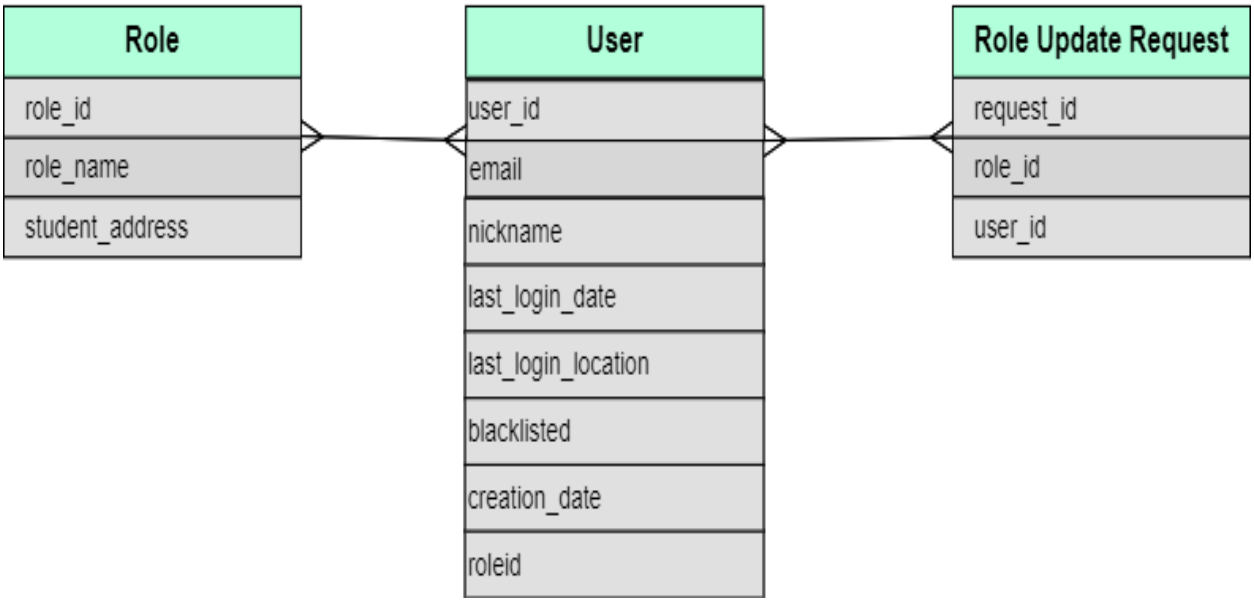
1) A general visitor – who is not logged in and just browsing through the site
2) An advanced user – a logged in user wanting to post stories, like/comment others stories
3) A moderator – a privileged user wanting to also flag inappropriate/spam content or users other than just post and like/comment stories
4) An admin – The owner/ monitor of the entire site and has all the privileges like adding, deleting users, deleting spam content, changing roles of users while still being able to do everything a registered user does

Database design and structure

There are 3 tables used in for storing the functionality of the user:

a) user – stores user details
b) role – stores roles of users
c) role_update_request – stores the role update requests from user so the admin can approve

Below is the ERD:



The database stores these users in reverse order as follows:

1 – Admin
2 – Moderator
3 – Registered users

Folder structure:

The following folder structure is followed:

1) Controllers :
   a) UserController – All the CRUD operations
   b) JWTAuthenticationController - Login/Register
2) DAO layer: POJO classes for all the tables
3) Model – JWTRequest/JWTResponse related classes
4) Service layer – business logic is performed here
5) Config – Has all the request/response related configurations

Working:

1) The client makes the HTTP requests (PUT or GET or POST).
2) The request goes to the controller, and the controller maps that request and handles it. After that, it calls the service logic if required.
3) In the service layer, all the business logic performs. It performs the logic on the data that is mapped to JPA with model classes.
4) A response is returned to the user if no error occurred.

Known issues and risks:

1) The last login date does not function as intended. In future, we expect to make the functionality working
2) Currently the system relies on gateway validation for JWT. This does not 100% guarantee the validation on user side. This validation can be done on user side as well

Future features:

1) User's login location can be stored in the database. Although we have the column in the database for this, this feature was out of scope as of now

Incase you have any queries related to user microservice in future, you can reach out to **shwetamandavgane@gmail.com**