Water Management Portal

Objective:

Water Management Portal is an online application to be built in the aspects include information about water supply and waste water management for the city. The site might also provide real-time information about flooding, water supply handling (boil-water alert), water-related work projects.

Users of the System:

1. Admin
2. visitors / city employees

Functional Requirements:

- Provide templates for information entry – e.g. education, dining guide, food handling guide, etc..
- Allow for easy update of information by city employees.
- Allow submission of feedback for improvement.
- Admin will check the feedbacks.
- **Users can submit the feedback only if the water duration is less than 30 min.**

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- Email integration for intimating new personalized offers to customers.
- Multi-factor authentication for the sign-in process

Output/ Post Condition:

- Report of water pressure levels at selected locations
- Report of water flow rates at selected locations
- Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance • Water Management Portal -< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability • Availability • Failover
Logging &	• The system should support logging(app/web/DB) & auditing at

Auditing	all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	React Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Pre-requisites (Do's and Don'ts):

1. The React app should run in port 8081. Do not run the React app in the port: 3000.
2. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
4. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.

2. Manual routing should be restricted by using AuthGaurd by implementing the canActivate interface. For example, if the user enters as http://localhost:3000/signup or http://localhost:3000/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

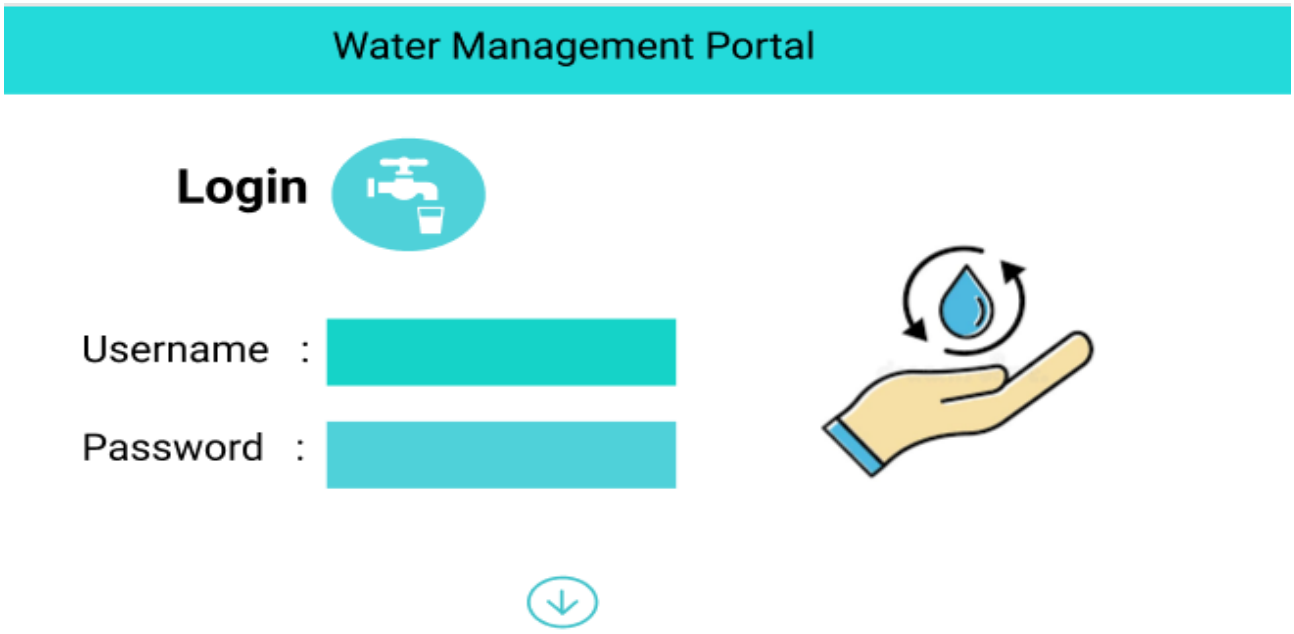
USER			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Add WaterInfo	/addInfo	POST	Water Info Added
Get WaterInfo	/getWaterInfo/{id}	GET	Particular id waterInfo
Add Feedback	/addFeedback	POST	Feedback Added
Get Feedback	/getFeedback	GET	Particular Feedback
ADMIN			
Action	URL	Method	Response
Get All WaterInfo	/admin	GET	Array of WaterInfo
Delete WaterInfo	/admin /{id}	DELETE	WaterInfo deleted
Update WaterInfo	/admin /{id}	PUT	Save the Changes
Get All Feedback	/admin/feedback	GET	Array of Feedback
Get feedback	/admin/feedback/{id}	GET	Feedback detail

Frontend:

Customer:

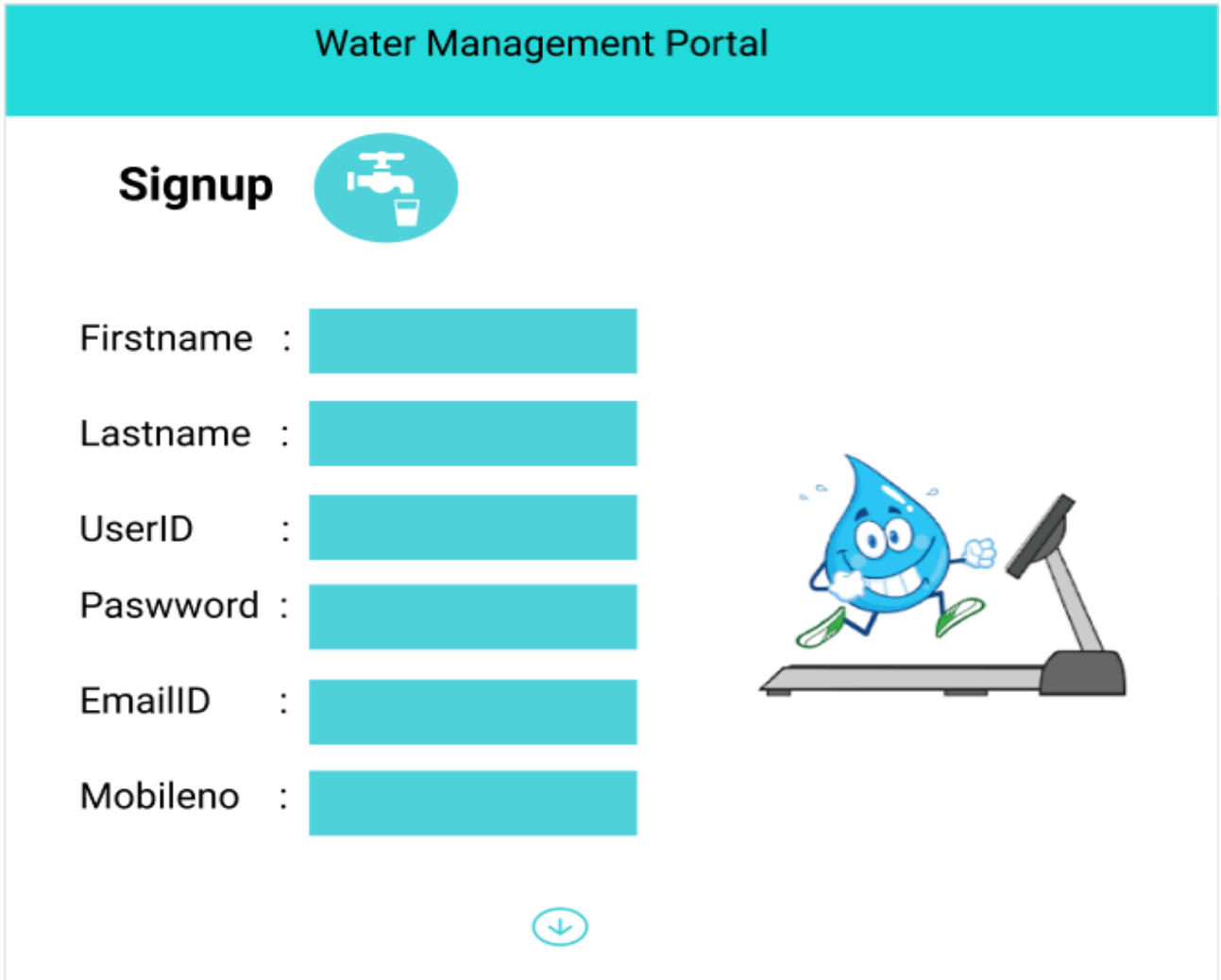
Login:

Output Screenshot:



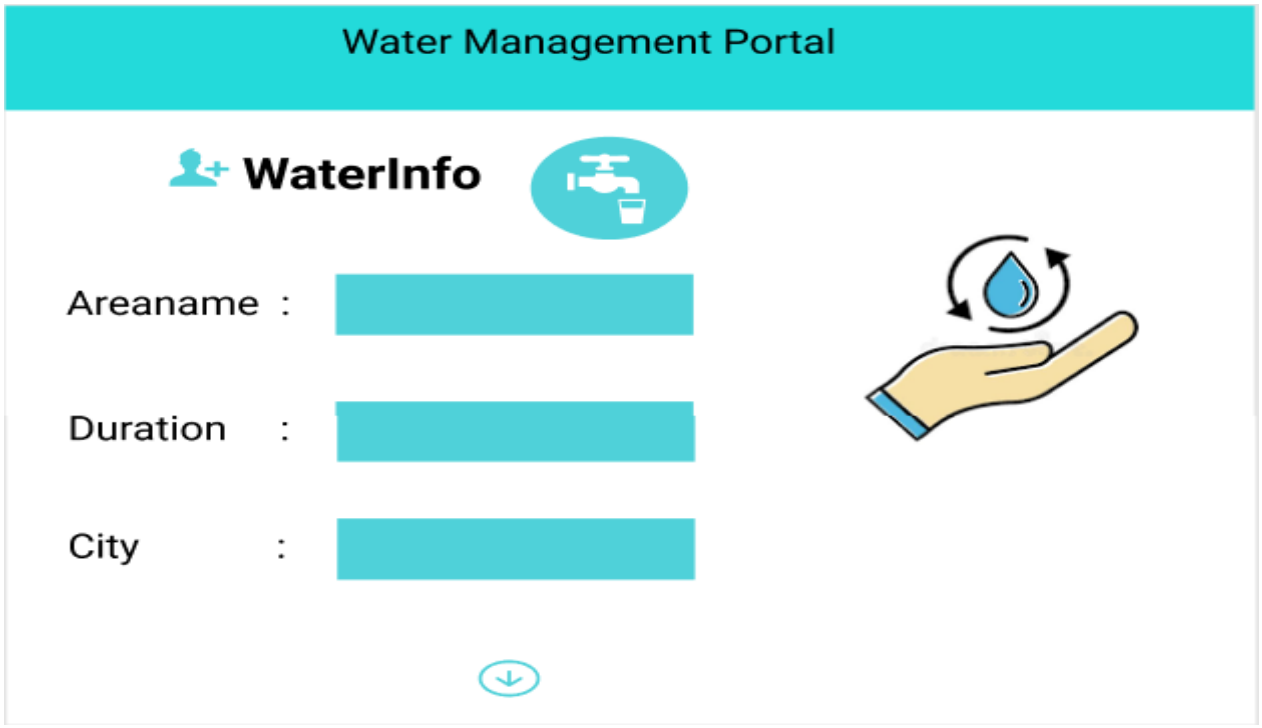
Signup:

Output Screenshot:



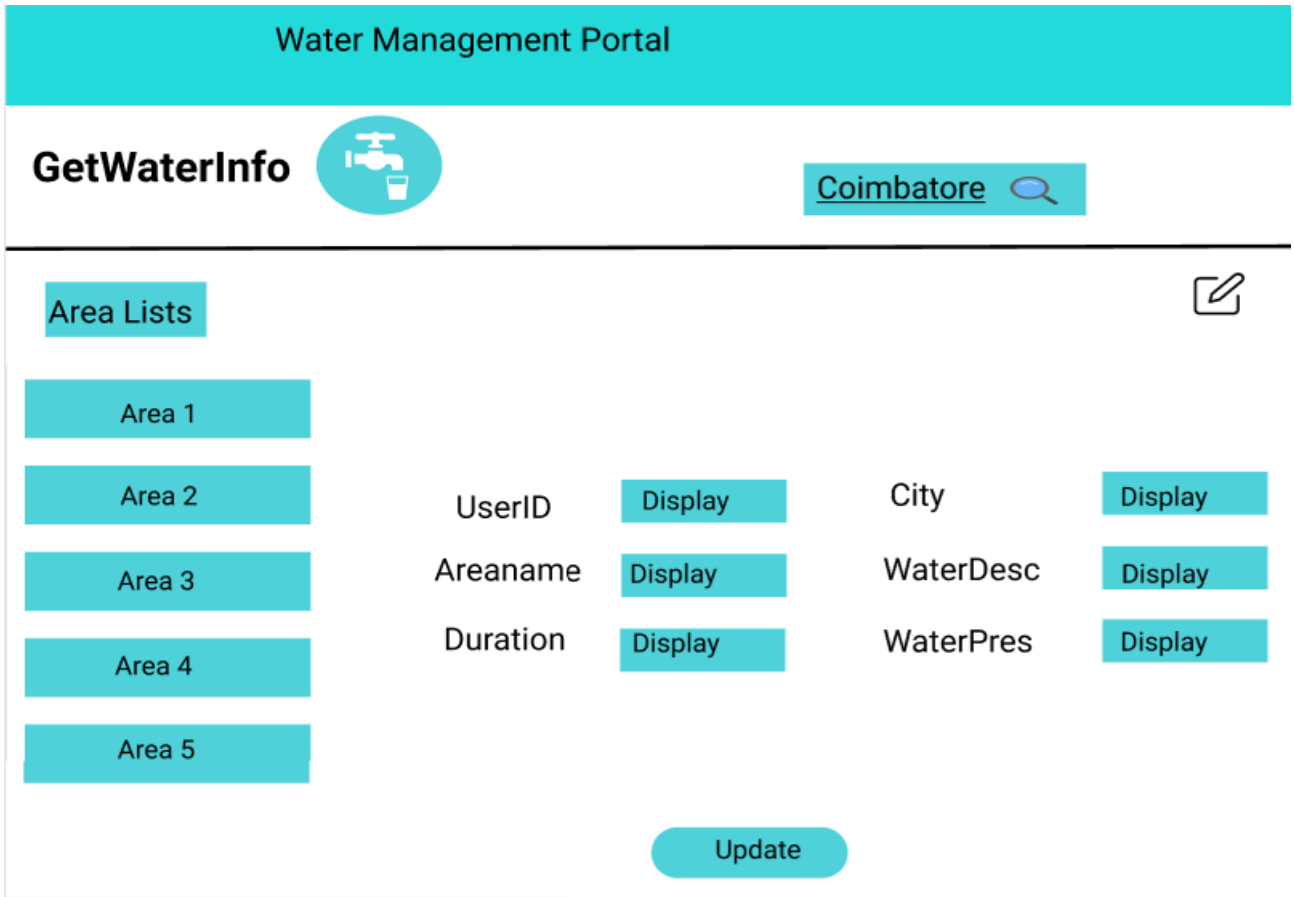
WaterInfo:

Output Screenshot:



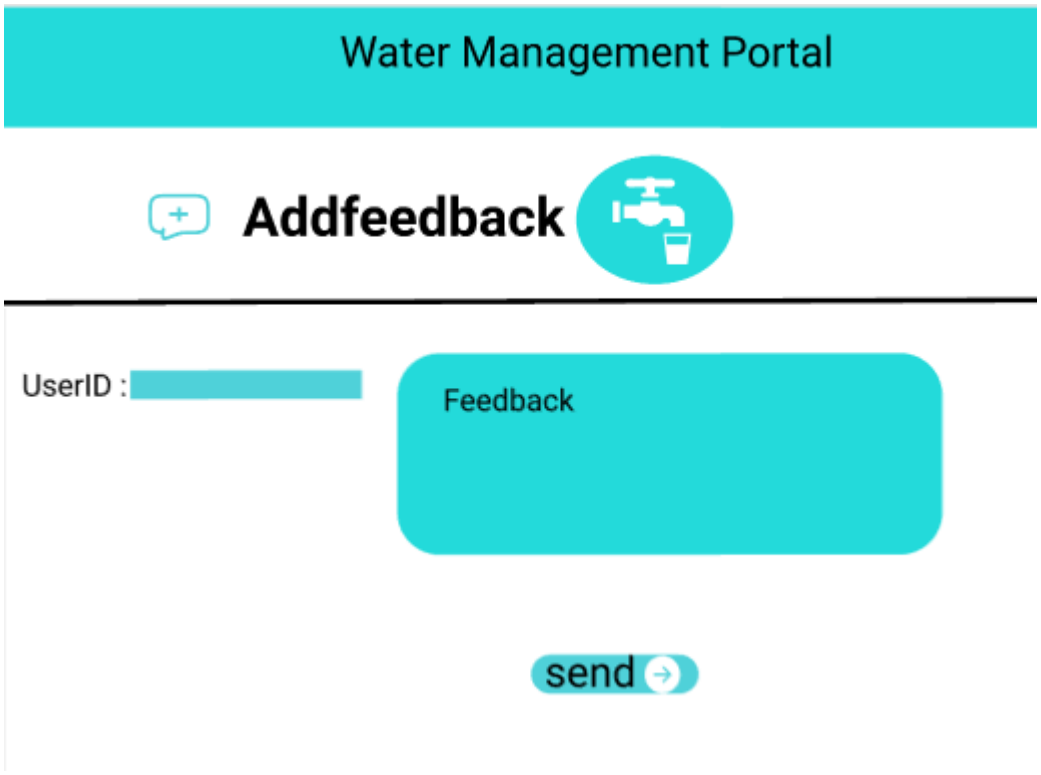
Get Water Info:

Output Screenshot:



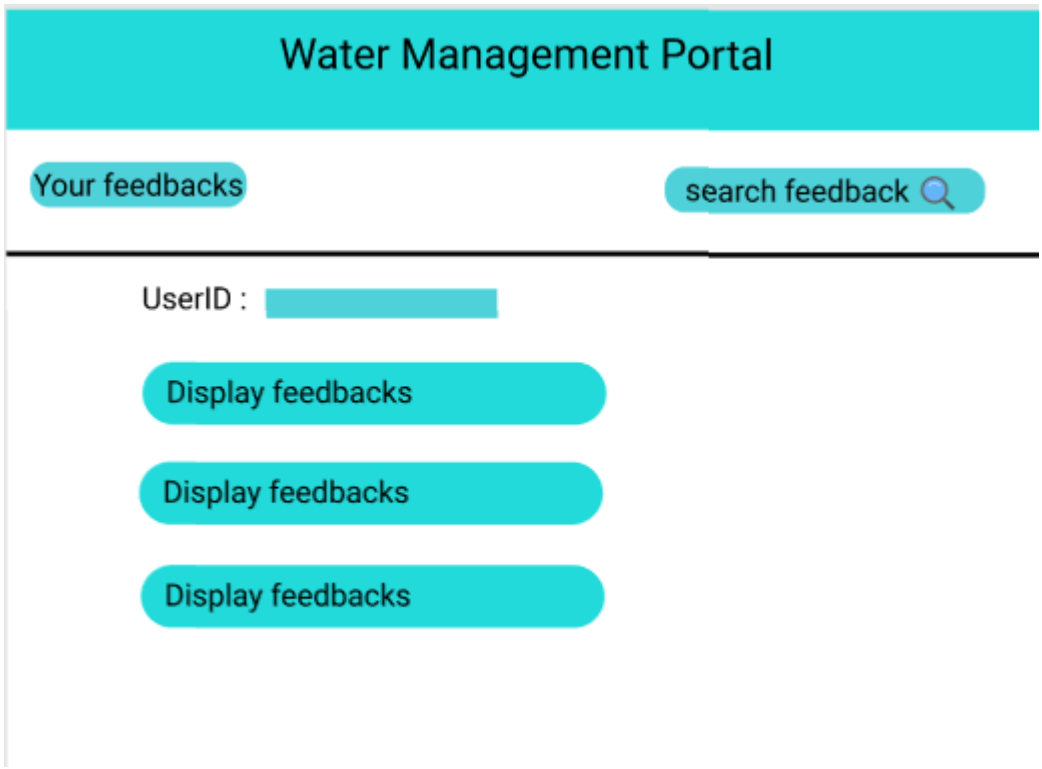
Add Feedback:

Output Screenshot:



Get Feedback:

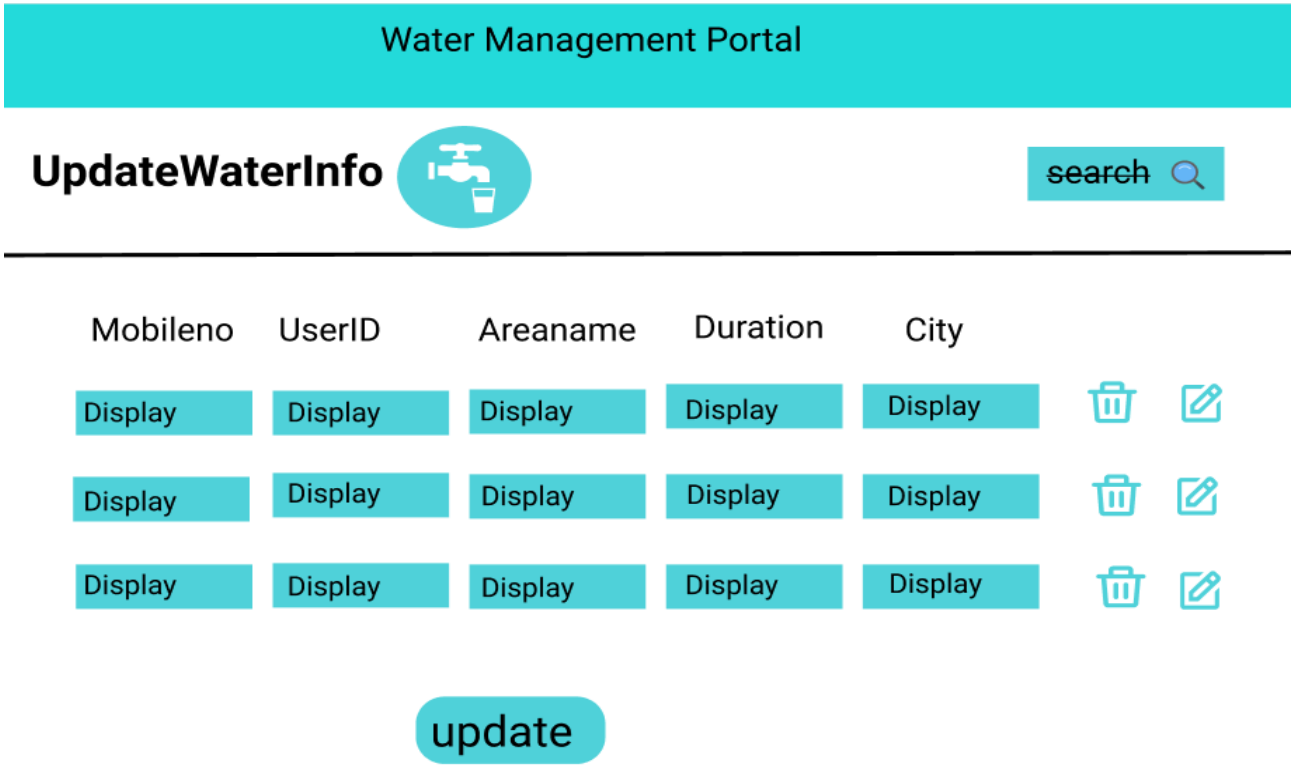
Output Screenshot:



Admin:

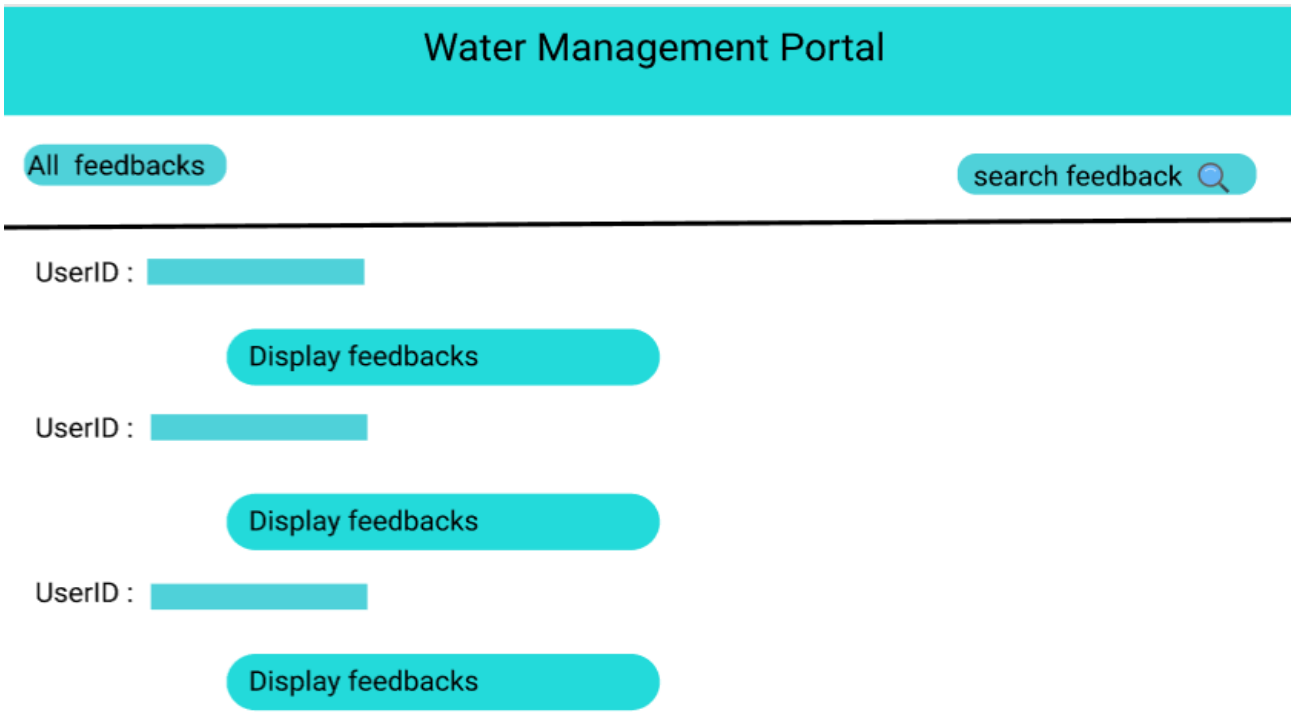
Home:

Output Screenshot:



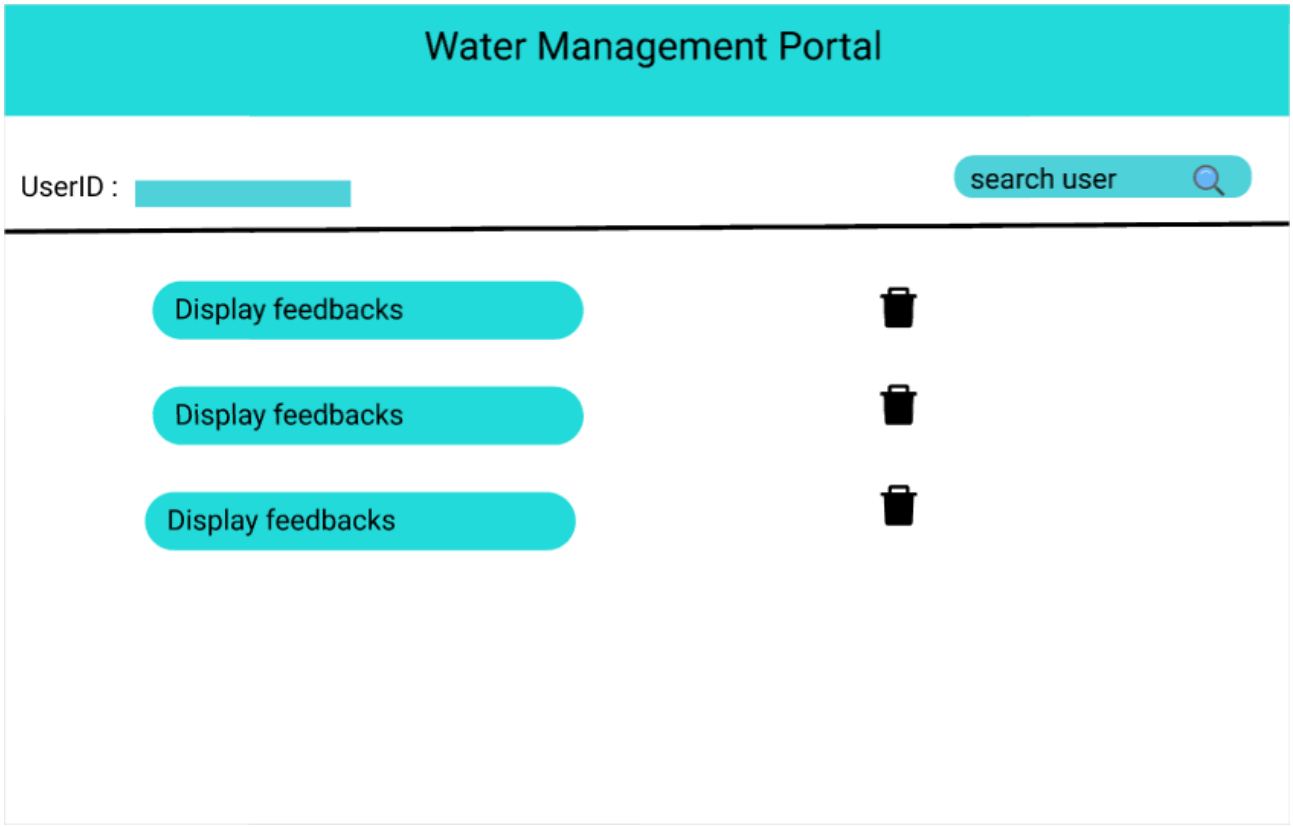
All Feedback:

Output Screenshot:



Feedback By Id:

Output Screenshot:



Backend:

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (city residents or the city employees) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. mobileNumber: String
        iv. active: Boolean
        v. role: String
    b. Methods: -
2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String

ii. password: String

b. Methods: -

3. WaterModel: This class stores the details of the water details.
    a. Attributes:
        i. userId: userModel
        ii. waterPressure: String
        iii. waterDesc: String
        iv. location: String
        v. city: String
        vi. feedback: FeedbackModel
    b. Methods: -
4. FeedbackModel: This class stores the feedback details.
    a. Attributes:
        i. feedbackId: String
        ii. feedbacDesc: String
    b. Methods: -

Controller Layer:

5. SignupController: This class control the user signup
    a. Attributes: -
    b. Methods:
        i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.
6. LoginController: This class controls the user login.
    a. Attributes: -
    b. Methods:
        i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false
7. WaterController: This class controls the add/edit/update/view Waterdetails.
    a. Attributes: -
    b. Methods:
        i. List< WaterModel> getWaterInfo(): This method helps the admin to fetch all Water informatin from the database.
        ii. WaterModel getWaterInfoById (String id): This method helps to retrieve a Water information from the database based on the Water id.

iii. WaterInfoUpdate(WaterModel data): This method helps to edit a Water informarion and save it to the database.

iv. waterInfoSave(WaterModel data): This method helps to add a new Water Information to the database.

v. waterInfoDelete (String id): This method helps to delete a Water Information from the database.

8. FeedbackController: This class helps to Manage the feedback details
    a. Attributes: -
    b. Methods:
        i. feedbackSave(FeedbackModel data): This method helps to add a new feedback to the database
        ii. List<FeedbackModel> getFeedback (): This method helps the admin to fetch all Feedback from the database.
        iii. FeedbackModel feedbackById(String id): This method helps to retrieve a Feedback from the database based on the Feedback id