DBMS PROJECT REPORT

TEAM :-

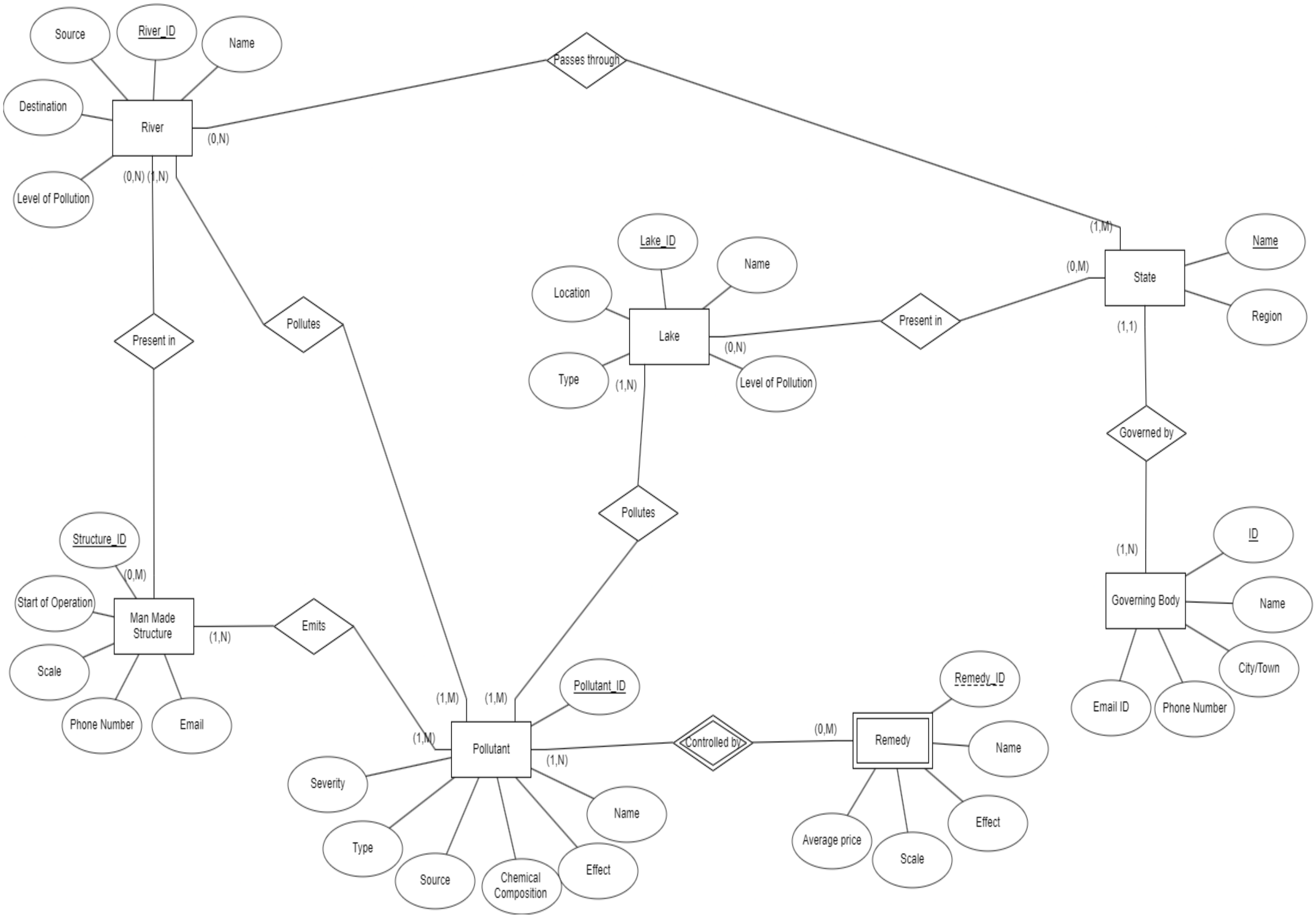
UTKARSH SETH - PES1UG21CS687
VIBHA MARPALLE - PES1UG21CS708

WATER POLLUTION MANAGEMENT DATABASE

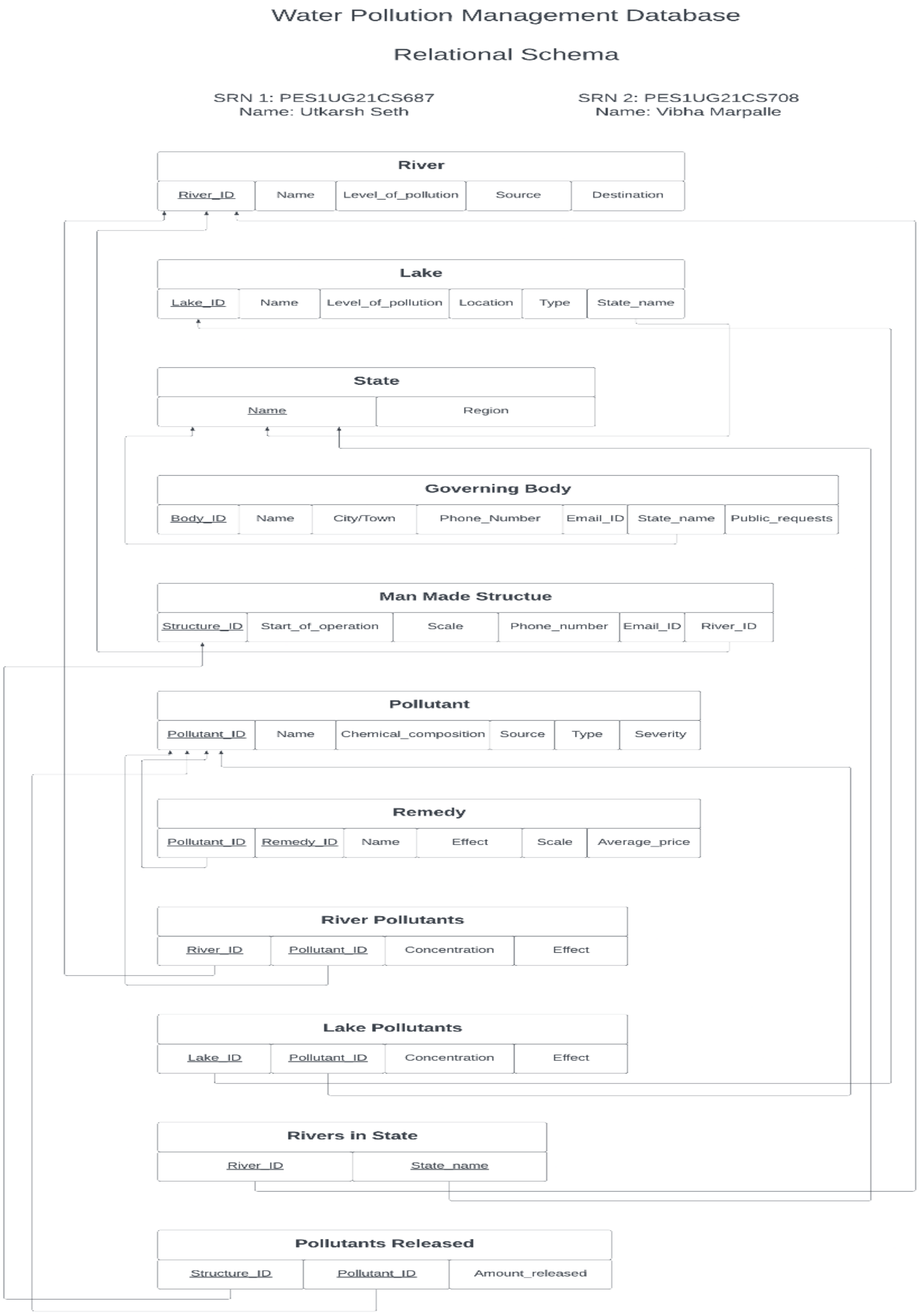
- **Description** : The Water Pollution Management Database Project aims to create a comprehensive and efficient system for monitoring, managing, and mitigating water pollution. This project stores state wise information about water bodies like pollution levels, pollutants and remedies which can be updated by government users and has a feature to raise requests regarding a body by citizen users.

- **List of Software used** : MySQL, Javascript, HTML, CSS, ReactJS, NodeJS, ExpressJS.

- **ER Diagram**


Source
River_ID
Name
Destination
River
(0,N)
Level of Pollution
(0,N) (1,N)
Passes through
(1,M)
(0,M)
State
Name
Region
(1,1)
Lake_ID
Name
Location
Lake
(0,N)
Type
(1,N)
Level of Pollution
Present in
Pollutes
Present in
Governed by
Pollutes
Structure_ID
(0,M)
Start of Operation
Man Made Structure
(1,N)
Scale
Emits
Phone Number
Email
(1,N)
ID
Governing Body
Name
City/Town
Email ID
Phone Number
(1,M)
(1,M)
Pollutant_ID
(1,M)
Pollutant
(1,N)
Controlled by
(0,M)
Remedy_ID
Remedy
Name
Severity
Name
Type
Source
Chemical Composition
Effect
Effect
Average price
Scale

- **<u>Relational Schema</u>**



Water Pollution Management Database

Relational Schema

SRN 1: PES1UG21CS687
Name: Utkarsh Seth

SRN 2: PES1UG21CS708
Name: Vibha Marpalle

River

<u>River_ID</u>	Name	Level_of_pollution	Source	Destination

Lake

<u>Lake_ID</u>	Name	Level_of_pollution	Location	Type	State_name

State

<u>Name</u>	Region

Governing Body

<u>Body_ID</u>	Name	City/Town	Phone_Number	Email_ID	State_name	Public_requests

Man Made Structue

<u>Structure_ID</u>	Start_of_operation	Scale	Phone_number	Email_ID	River_ID

Pollutant

<u>Pollutant_ID</u>	Name	Chemical_composition	Source	Type	Severity

Remedy

<u>Pollutant_ID</u>	<u>Remedy_ID</u>	Name	Effect	Scale	Average_price

River Pollutants

<u>River_ID</u>	<u>Pollutant_ID</u>	Concentration	Effect

Lake Pollutants

<u>Lake_ID</u>	<u>Pollutant_ID</u>	Concentration	Effect

Rivers in State

<u>River_ID</u>	<u>State_name</u>

Pollutants Released

<u>Structure_ID</u>	<u>Pollutant_ID</u>	Amount_released

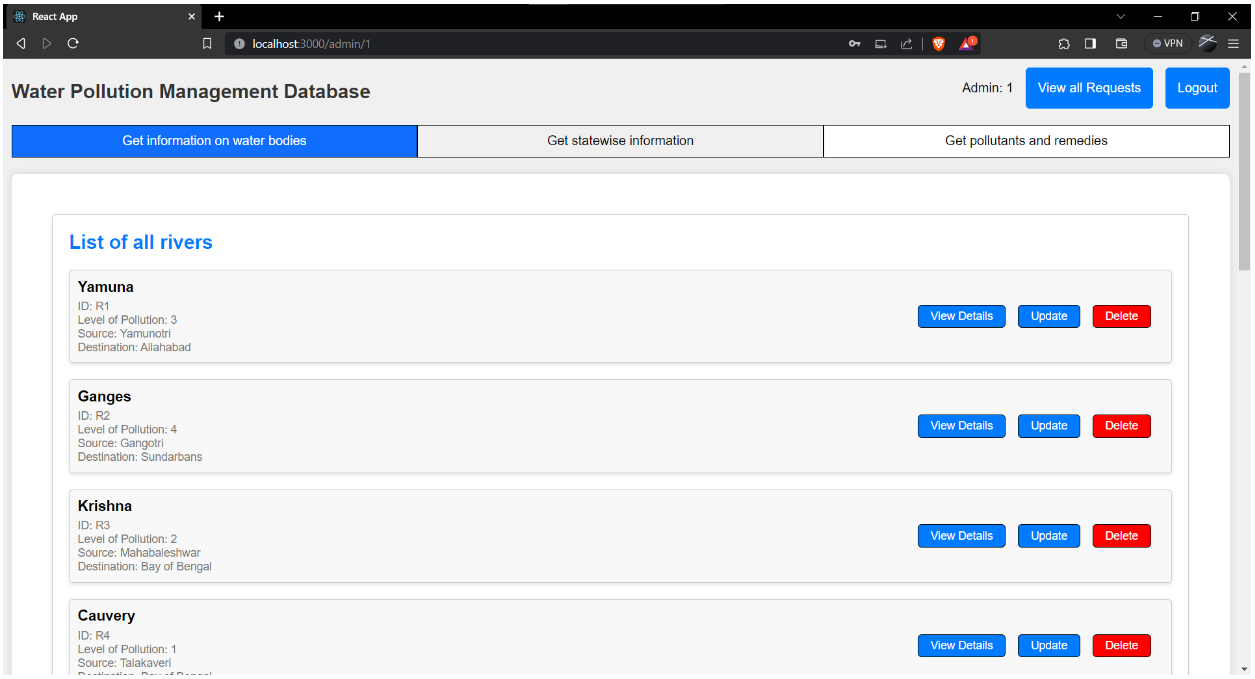
DDL SQL commands

```
CREATE DATABASE miniProject;
USE miniProject;

CREATE TABLE rivers (
          `river_id` VARCHAR(10) PRIMARY KEY,
    `name` VARCHAR(30) NOT NULL,
    `level_of_pollution` ENUM("1", "2", "3", "4", "5"),
    `source` VARCHAR(30),
    `destination` VARCHAR(30)
);

CREATE TABLE states (
          `name` VARCHAR(30) PRIMARY KEY,
    `region` VARCHAR(30) NOT NULL
);

CREATE TABLE lakes (
          `lake_id` VARCHAR(10) PRIMARY KEY,
    `name` VARCHAR(30) NOT NULL,
    `level_of_pollution` ENUM("1", "2", "3", "4", "5"),
    `location` VARCHAR(50),
    `type` ENUM("Natural", "Man-made"),
    `state_name` VARCHAR(30)
);
ALTER TABLE lakes ADD FOREIGN KEY (`state_name`) REFERENCES
states(`name`) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE pollutants (
    `pollutant_id` VARCHAR(10) PRIMARY KEY,
    `name` VARCHAR(30) NOT NULL,
    `chemical_composition` VARCHAR(30),
    `source` VARCHAR(30),
    `type` VARCHAR(30),
    `severity` ENUM("High", "Medium", "Low")
);

CREATE TABLE man_made_structures (
    `structure_id` VARCHAR(10) PRIMARY KEY,
    `start_of_operation` DATE NOT NULL,
    `scale` ENUM("Small", "Medium", "Large") NOT NULL,
    `phone_number` VARCHAR(15) NOT NULL,
    `email` VARCHAR(50) NOT NULL,
    `river_id` VARCHAR(10) NOT NULL,
    CONSTRAINT FOREIGN KEY (`river_id`) REFERENCES rivers(`river_id`) ON
DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE river_pollutants (
    `river_id` VARCHAR(10),
    `pollutant_id` VARCHAR(10),
    `concentration` VARCHAR(20) NOT NULL,
    `effect` VARCHAR(100) NOT NULL,
    CONSTRAINT PRIMARY KEY (`river_id`, `pollutant_id`),
    CONSTRAINT FOREIGN KEY (`river_id`) REFERENCES rivers(`river_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY (`pollutant_id`) REFERENCES
pollutants(`pollutant_id`) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE lake_pollutants (
    `lake_id` VARCHAR(10),
    `pollutant_id` VARCHAR(10),
    `concentration` VARCHAR(20) NOT NULL,
    `effect` VARCHAR(100) NOT NULL,
    CONSTRAINT PRIMARY KEY (`lake_id`, `pollutant_id`),
    CONSTRAINT FOREIGN KEY (`lake_id`) REFERENCES lakes(`lake_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY (`pollutant_id`) REFERENCES
pollutants(`pollutant_id`) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE rivers_in_state (
    `river_id` VARCHAR(10),
    `state_name` VARCHAR(30),
    CONSTRAINT PRIMARY KEY (`river_id`, `state_name`),
    CONSTRAINT FOREIGN KEY (`river_id`) REFERENCES rivers(`river_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY (`state_name`) REFERENCES states(`name`)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE pollutants_released (
    `structure_id` VARCHAR(10),
    `pollutant_id` VARCHAR(10),
    `amount_released` VARCHAR(30) NOT NULL,
    CONSTRAINT PRIMARY KEY (`structure_id`, `pollutant_id`),
    CONSTRAINT FOREIGN KEY (`structure_id`) REFERENCES
man_made_structures(`structure_id`) ON DELETE CASCADE ON UPDATE
CASCADE,
    CONSTRAINT FOREIGN KEY (`pollutant_id`) REFERENCES
pollutants(`pollutant_id`) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE government_bodies (
    `body_id` VARCHAR(10),
    `name` VARCHAR(45) NOT NULL,
    `city/town` VARCHAR(45) NOT NULL,
    `phone` VARCHAR(15) NOT NULL,
    `email` VARCHAR(45) NULL,
    `state_name` VARCHAR(30) NOT NULL,
    PRIMARY KEY (`body_id`)
);
ALTER TABLE government_bodies ADD CONSTRAINT FOREIGN KEY
(`state_name`) REFERENCES states(`name`) ON DELETE CASCADE ON
UPDATE CASCADE;

CREATE TABLE remedies (
    `pollutant_id` VARCHAR(10),
    `remedy_id` VARCHAR(10) NOT NULL,
    `name` VARCHAR(45) NOT NULL,
    `effect` VARCHAR(100) NOT NULL,
    `scale` VARCHAR(45) NULL,
    `avg_price` FLOAT NOT NULL,
    PRIMARY KEY (pollutant_id, remedy_id),
    CONSTRAINT FOREIGN KEY (`pollutant_id`) REFERENCES
pollutants(`pollutant_id`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE users (
   `email` VARCHAR(50) PRIMARY KEY,
   `first_name` VARCHAR(30) NOT NULL,
   `last_name` VARCHAR(30) NOT NULL,
   `password` VARCHAR(50) NOT NULL
);

CREATE TABLE admins (
   `id` INT PRIMARY KEY,
   `first_name` VARCHAR(30) NOT NULL,
   `last_name` VARCHAR(30) NOT NULL,
   `password` VARCHAR(30) NOT NULL
);

CREATE TABLE government_users (
   `government_id` INT PRIMARY KEY,
   `first_name` VARCHAR(30) NOT NULL,
   `last_name` VARCHAR(30) NOT NULL,
   `email` VARCHAR(50) NOT NULL,
   `password` VARCHAR(50) NOT NULL
);

CREATE TABLE requests (
   `request_id` VARCHAR(15) PRIMARY KEY,
   `user_email` VARCHAR(50),
   `lake_id` VARCHAR(10),
   `river_id` VARCHAR(10),
   `city` VARCHAR(45) NOT NULL,
   `state_name` VARCHAR(30),
   `content` VARCHAR(300) NOT NULL,
   CONSTRAINT FOREIGN KEY (`user_email`) REFERENCES users(`email`)
ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT FOREIGN KEY (`lake_id`) REFERENCES lakes(`lake_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT FOREIGN KEY (`state_name`) REFERENCES states(`name`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

- **CRUD Operations Screenshots**

1) Read


React App
localhost:3000/admin/1
View all Requests
Logout
Get information on water bodies
Get statewise information
Get pollutants and remedies
View Details
Update
Delete

Water Pollution Management Database

Admin: 1

List of all rivers

Yamuna
ID: R1
Level of Pollution: 3
Source: Yamunotri
Destination: Allahabad

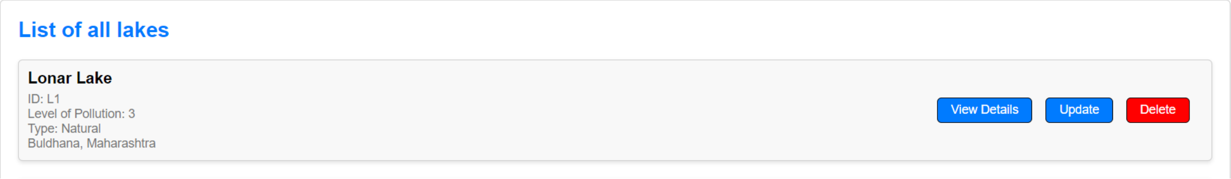
Ganges
ID: R2
Level of Pollution: 4
Source: Gangotri
Destination: Sundarbans

Krishna
ID: R3
Level of Pollution: 2
Source: Mahabaleshwar
Destination: Bay of Bengal

Cauvery
ID: R4
Level of Pollution: 1
Source: Talakaveri
Destination: Bay of Bengal

2) Update

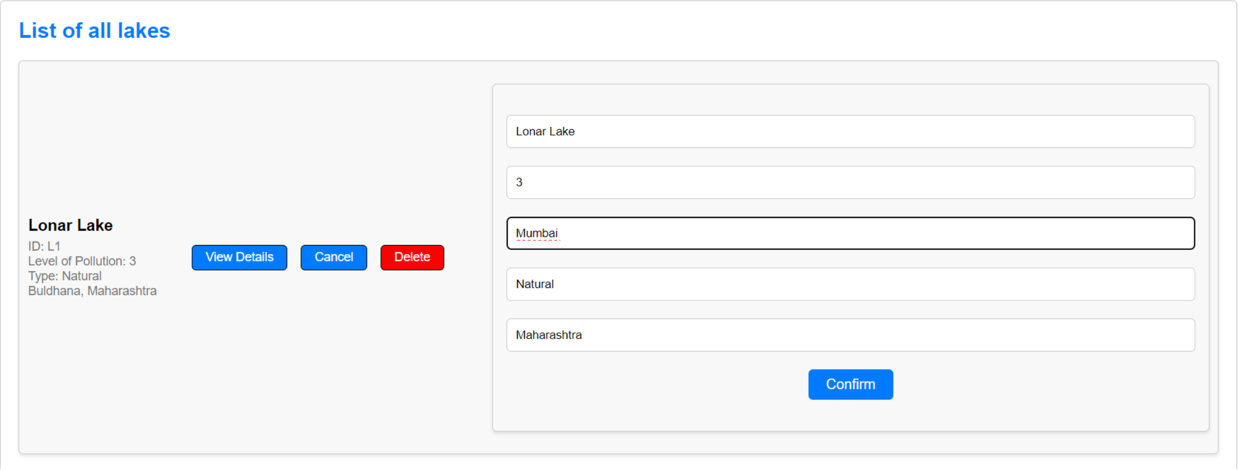
Before:



List of all lakes

Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Buldhana, Maharashtra

Updating the city:



List of all lakes

Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Buldhana, Maharashtra

Lonar Lake

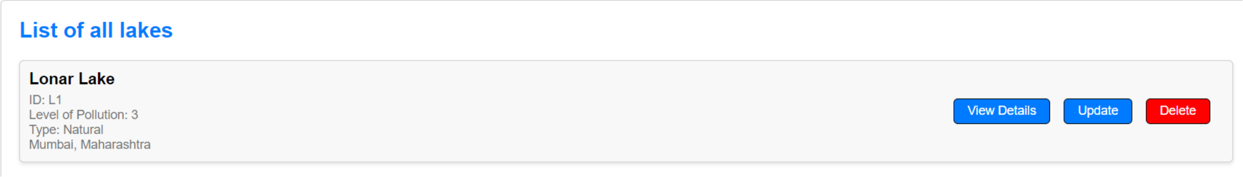
3

Mumbai

Natural

Maharashtra

After:

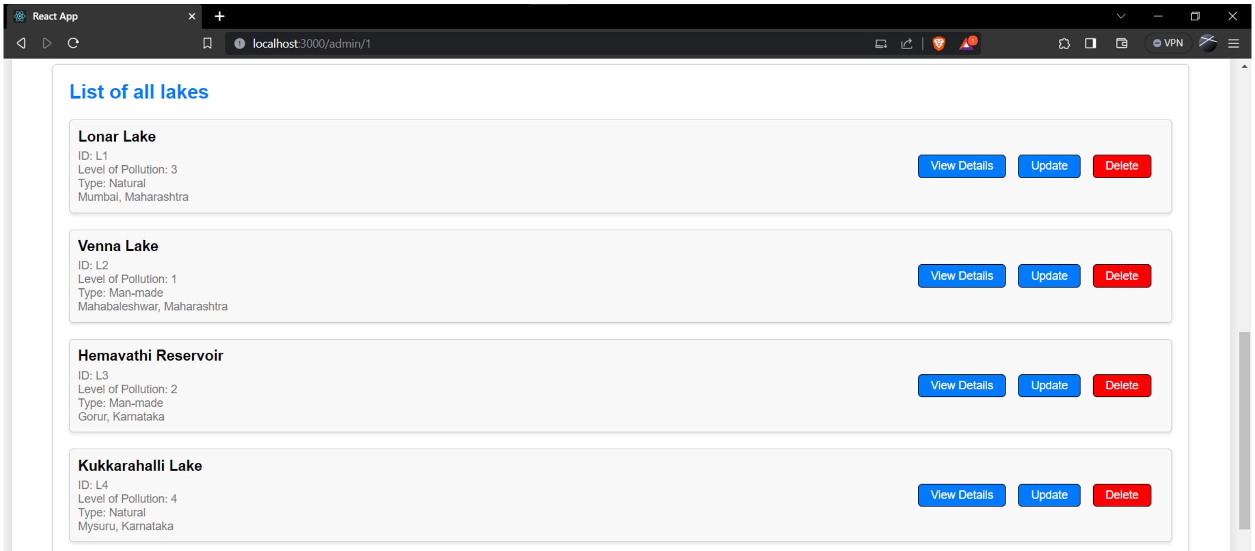


List of all lakes

Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Mumbai, Maharashtra

3) Delete

Before:



List of all lakes

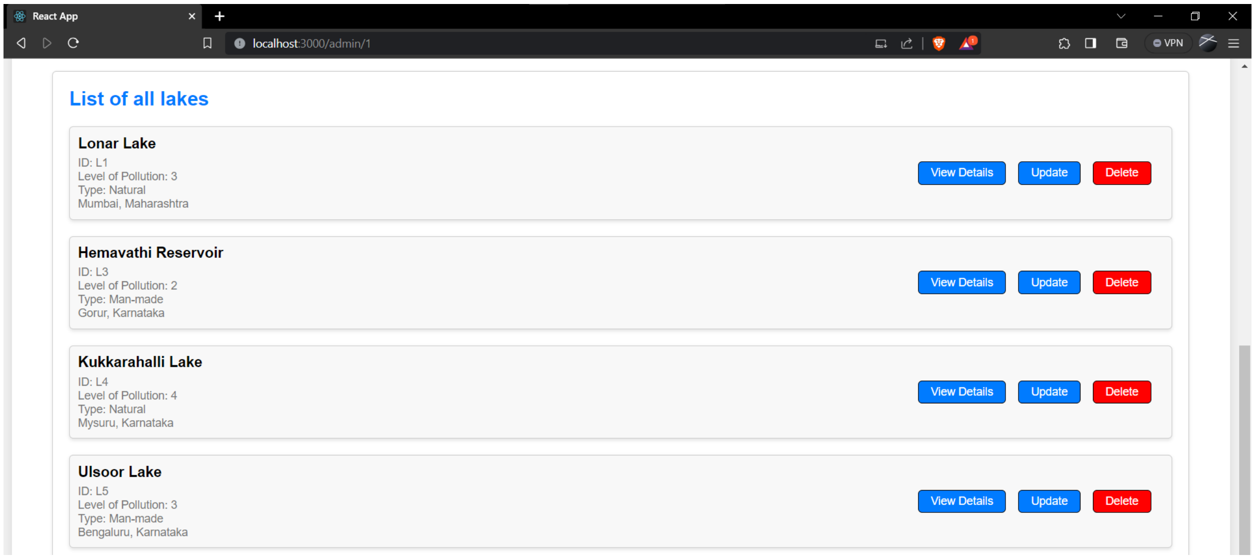
Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Mumbai, Maharashtra

Venna Lake
ID: L2
Level of Pollution: 1
Type: Man-made
Mahabaleshwar, Maharashtra

Hemavathi Reservoir
ID: L3
Level of Pollution: 2
Type: Man-made
Gorur, Karnataka

Kukkarahalli Lake
ID: L4
Level of Pollution: 4
Type: Natural
Mysuru, Karnataka

After deleting Venna Lake:



List of all lakes

Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Mumbai, Maharashtra

Hemavathi Reservoir
ID: L3
Level of Pollution: 2
Type: Man-made
Gorur, Karnataka

Kukkarahalli Lake
ID: L4
Level of Pollution: 4
Type: Natural
Mysuru, Karnataka

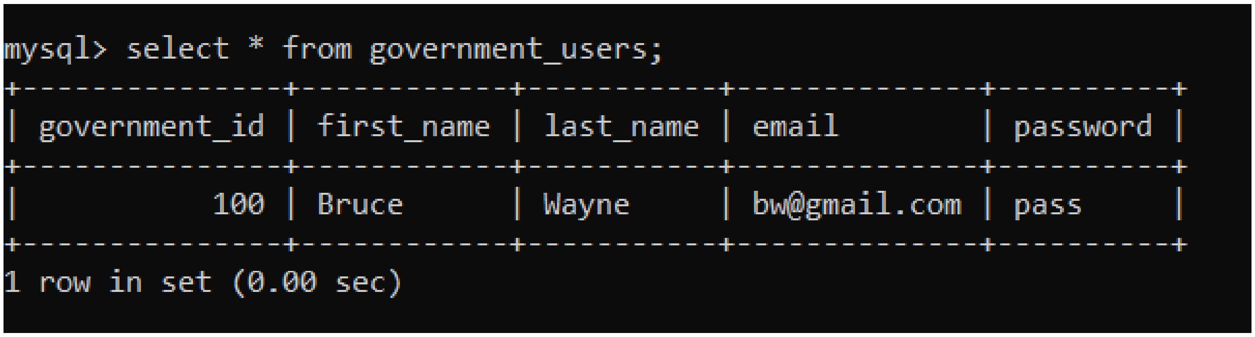
Ulsoor Lake
ID: L5
Level of Pollution: 3
Type: Man-made
Bengaluru, Karnataka

● List of Functionalities with frontend screenshot

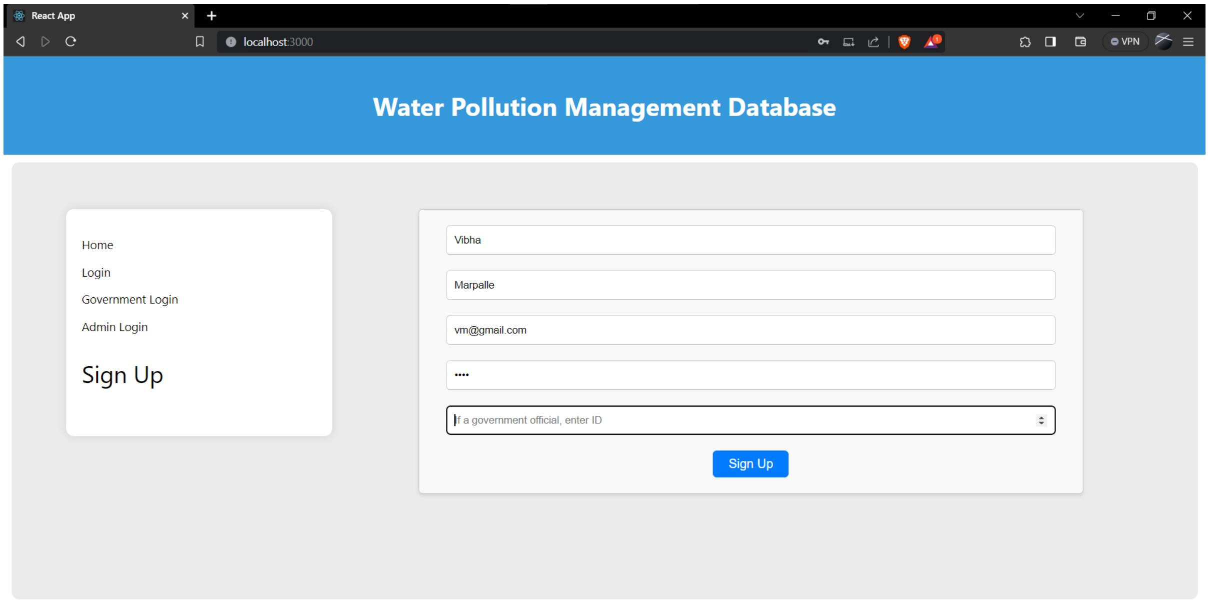
1) Signup/Login for government bodies

a) Signup

Before:



```
mysql> select * from government_users;
+---------------+------------+-----------+--------------+----------+
| government_id | first_name | last_name | email        | password |
+---------------+------------+-----------+--------------+----------+
|           100 | Bruce      | Wayne     | bw@gmail.com | pass     |
+---------------+------------+-----------+--------------+----------+
1 row in set (0.00 sec)
```



Vibha

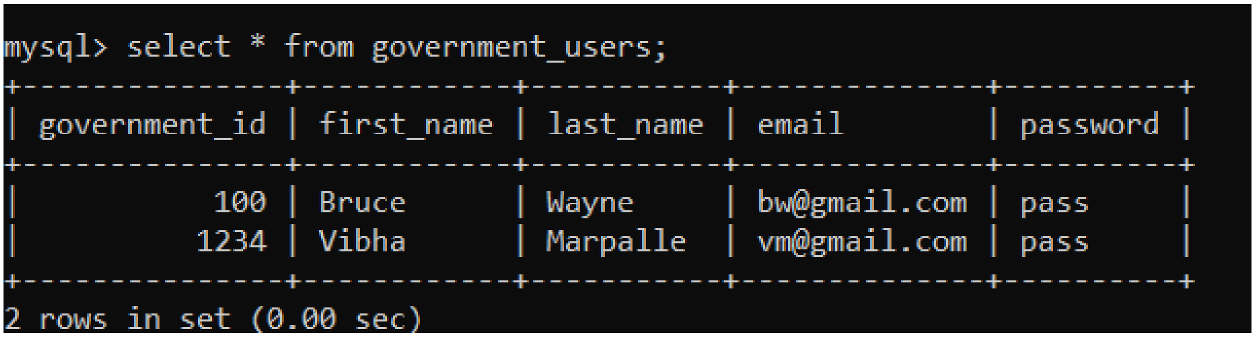
Marpalle

vm@gmail.com

••••

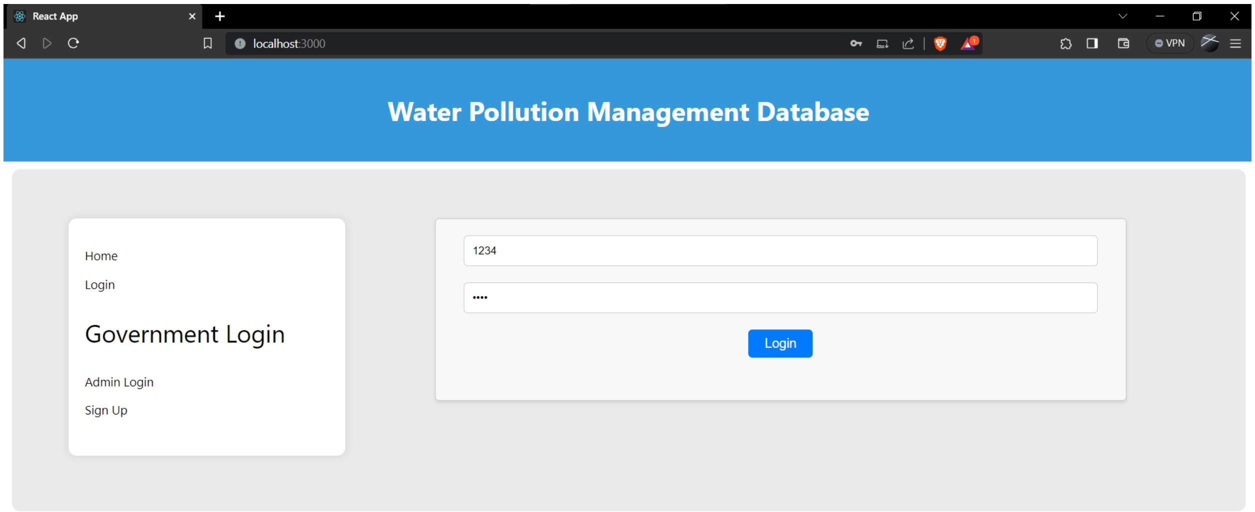
If a government official, enter ID

After:



```
mysql> select * from government_users;
+---------------+------------+-----------+--------------+----------+
| government_id | first_name | last_name | email        | password |
+---------------+------------+-----------+--------------+----------+
|           100 | Bruce      | Wayne     | bw@gmail.com | pass     |
|          1234 | Vibha      | Marpalle  | vm@gmail.com | pass     |
+---------------+------------+-----------+--------------+----------+
2 rows in set (0.00 sec)
```

Login:

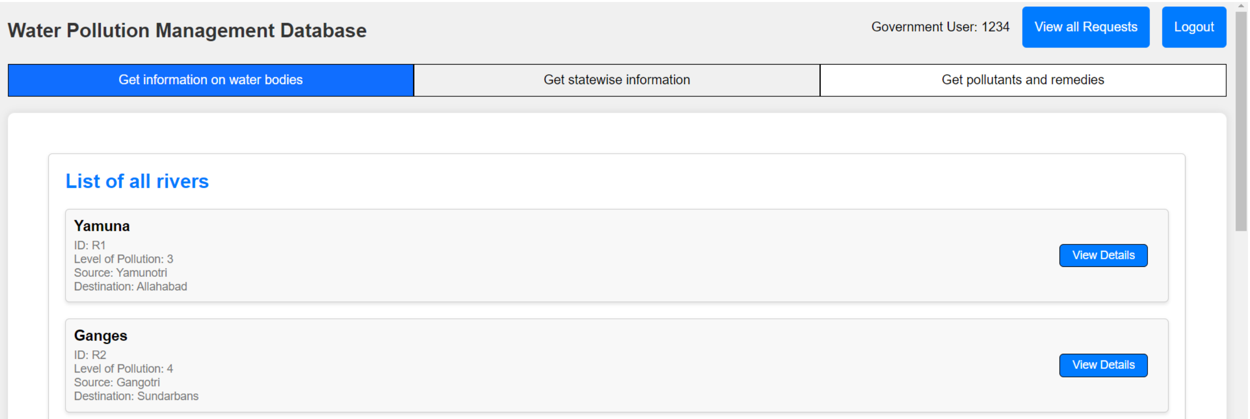


Water Pollution Management Database

Government Login

1234

••••


View all Requests
Logout
Get information on water bodies
Get statewise information
Get pollutants and remedies
View Details
View Details

Water Pollution Management Database

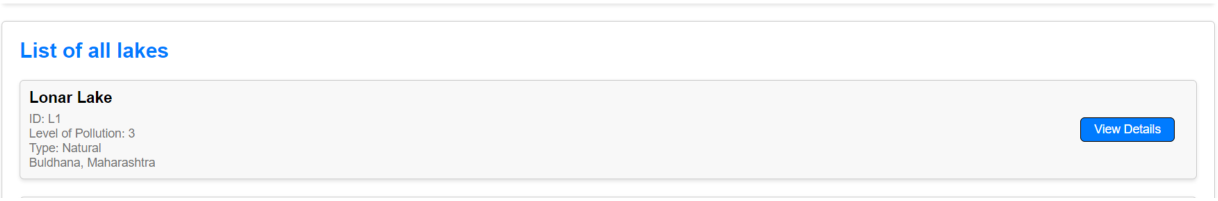
Government User: 1234

List of all rivers

Yamuna
ID: R1
Level of Pollution: 3
Source: Yamunotri
Destination: Allahabad

Ganges
ID: R2
Level of Pollution: 4
Source: Gangotri
Destination: Sundarbans

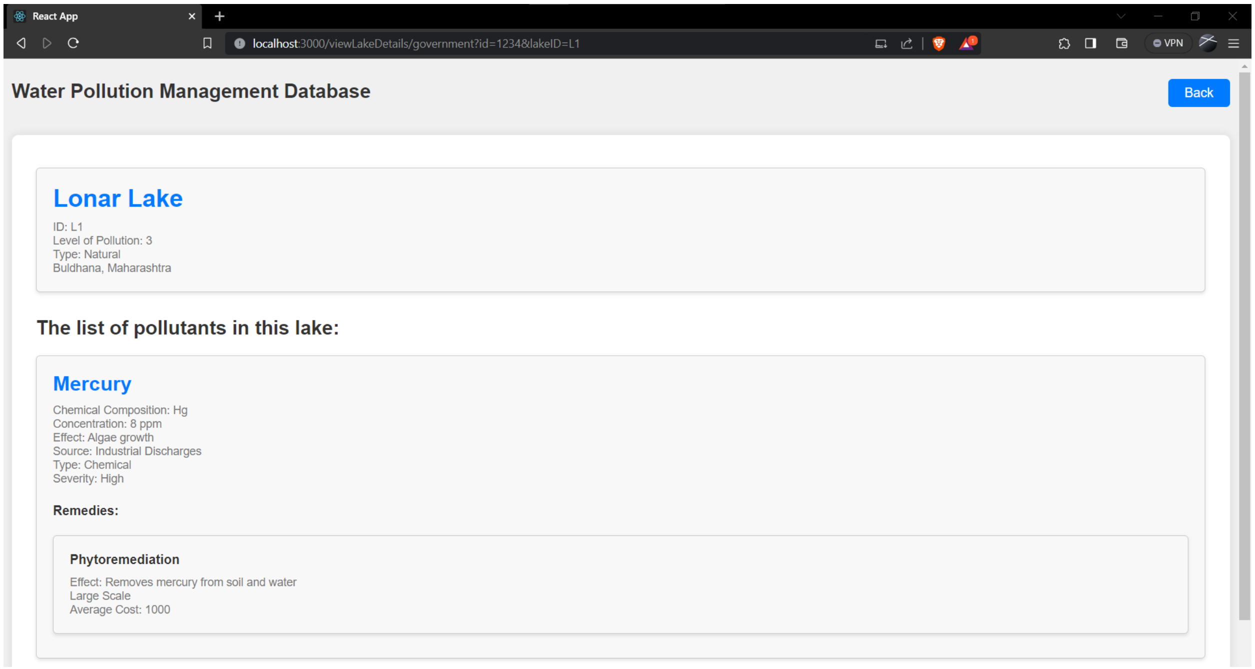
2) Viewing the pollution status for a water body



List of all lakes

Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Buldhana, Maharashtra

After clicking on ‘View Details’:



Water Pollution Management Database

Lonar Lake
ID: L1
Level of Pollution: 3
Type: Natural
Buldhana, Maharashtra

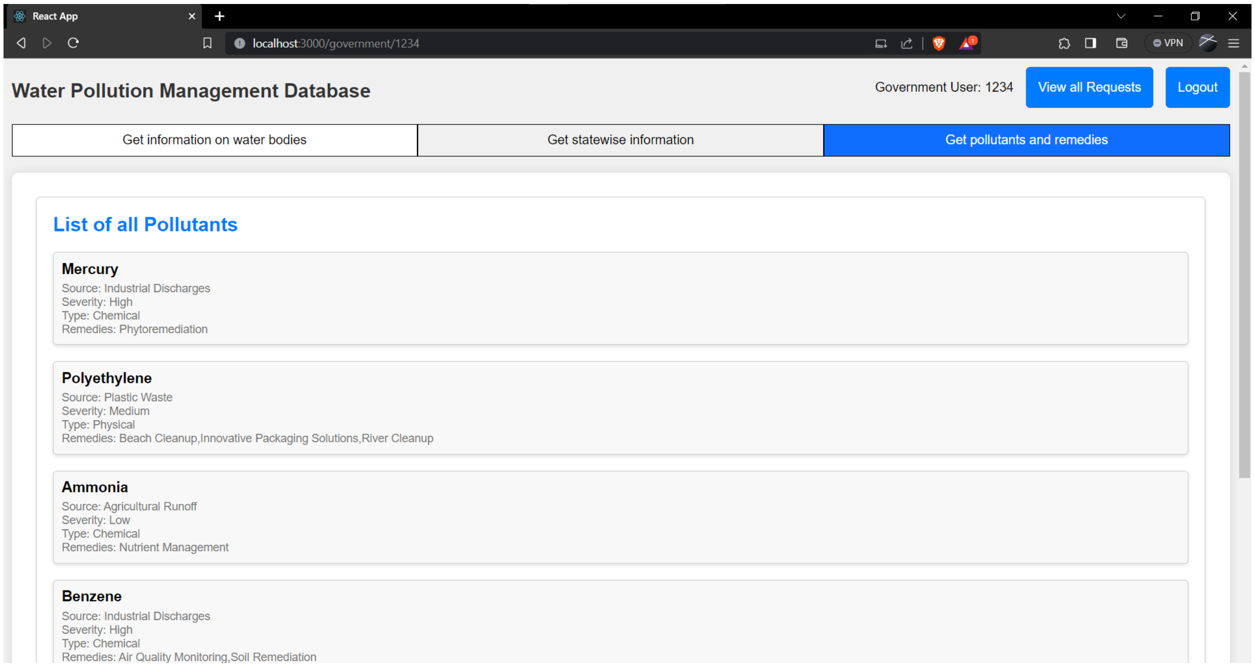
The list of pollutants in this lake:

Mercury
Chemical Composition: Hg
Concentration: 8 ppm
Effect: Algae growth
Source: Industrial Discharges
Type: Chemical
Severity: High

Remedies:

Phytoremediation
Effect: Removes mercury from soil and water
Large Scale
Average Cost: 1000

3) Finding the source of pollution


React App
localhost:3000/government/1234
VPN
Government User: 1234
View all Requests
Logout
Get information on water bodies
Get statewise information
Get pollutants and remedies

Water Pollution Management Database

List of all Pollutants

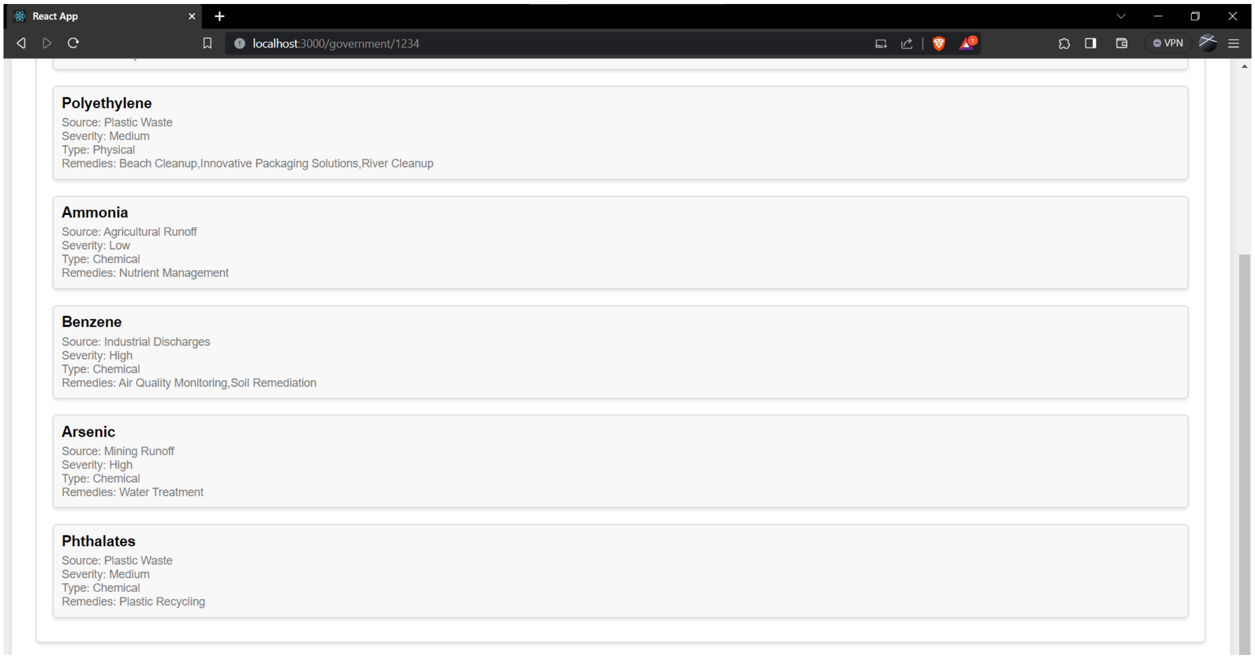
Mercury
Source: Industrial Discharges
Severity: High
Type: Chemical
Remedies: Phytoremediation

Polyethylene
Source: Plastic Waste
Severity: Medium
Type: Physical
Remedies: Beach Cleanup,Innovative Packaging Solutions,River Cleanup

Ammonia
Source: Agricultural Runoff
Severity: Low
Type: Chemical
Remedies: Nutrient Management

Benzene
Source: Industrial Discharges
Severity: High
Type: Chemical
Remedies: Air Quality Monitoring,Soil Remediation

4) Giving remedies for a type of pollutant



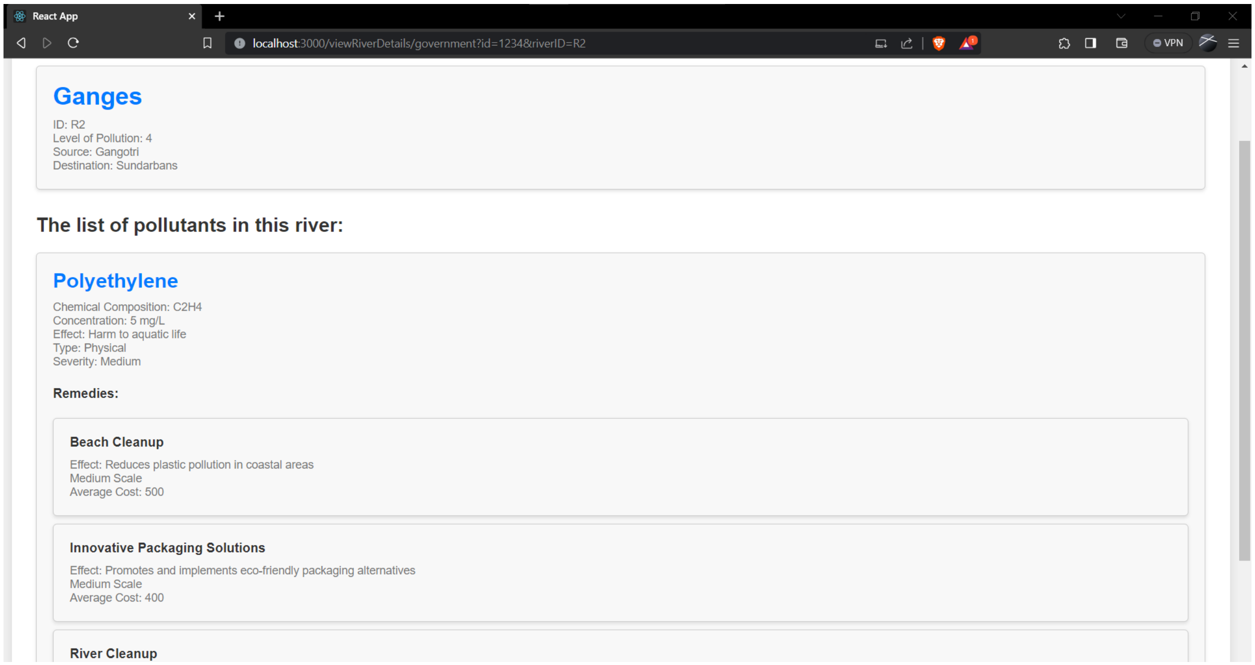
Polyethylene
Source: Plastic Waste
Severity: Medium
Type: Physical
Remedies: Beach Cleanup,Innovative Packaging Solutions,River Cleanup

Ammonia
Source: Agricultural Runoff
Severity: Low
Type: Chemical
Remedies: Nutrient Management

Benzene
Source: Industrial Discharges
Severity: High
Type: Chemical
Remedies: Air Quality Monitoring,Soil Remediation

Arsenic
Source: Mining Runoff
Severity: High
Type: Chemical
Remedies: Water Treatment

Phthalates
Source: Plastic Waste
Severity: Medium
Type: Chemical
Remedies: Plastic Recycling


React App
localhost:3000/viewRiverDetails/government?id=1234&riverID=R2
VPN

Ganges

ID: R2
Level of Pollution: 4
Source: Gangotri
Destination: Sundarbans

The list of pollutants in this river:

Polyethylene

Chemical Composition: C2H4
Concentration: 5 mg/L
Effect: Harm to aquatic life
Type: Physical
Severity: Medium

Remedies:

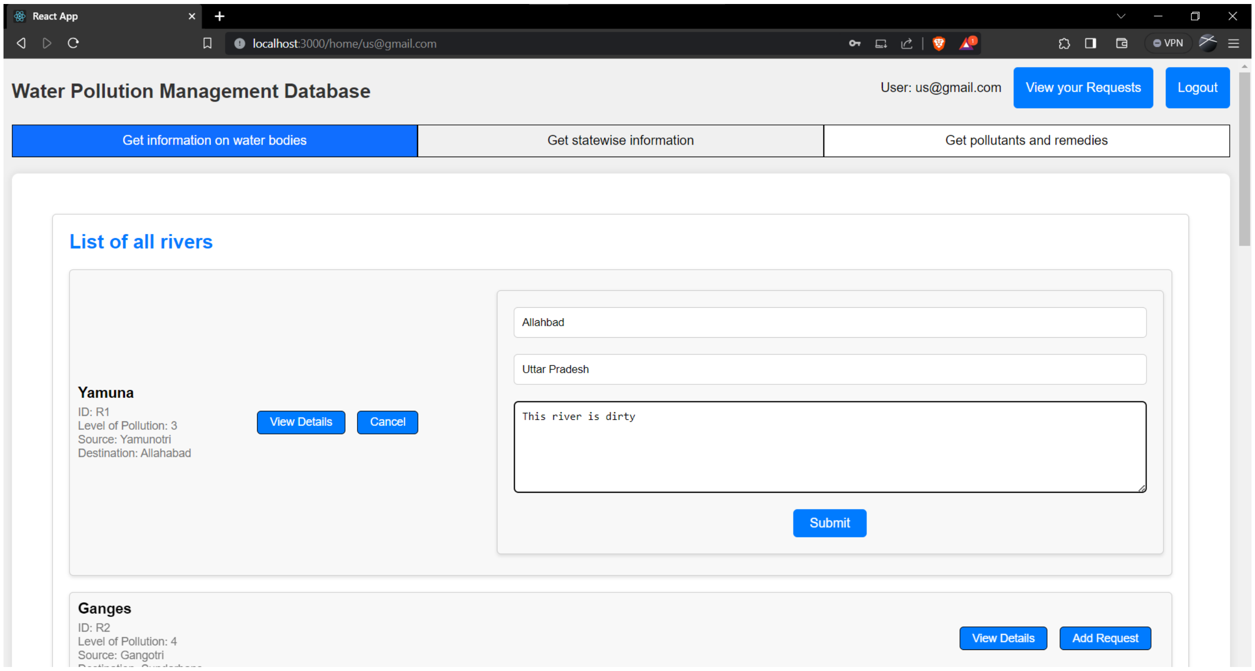
Beach Cleanup
Effect: Reduces plastic pollution in coastal areas
Medium Scale
Average Cost: 500

Innovative Packaging Solutions
Effect: Promotes and implements eco-friendly packaging alternatives
Medium Scale
Average Cost: 400

River Cleanup

5) Raising public request regarding a specific water body

Adding Request:


React App
localhost:3000/home/us@gmail.com
VPN
View your Requests
Logout
Get information on water bodies
Get statewise information
Get pollutants and remedies
View Details
Cancel
Submit
View Details
Add Request

Water Pollution Management Database

User: us@gmail.com

List of all rivers

Yamuna
ID: R1
Level of Pollution: 3
Source: Yamunotri
Destination: Allahbad

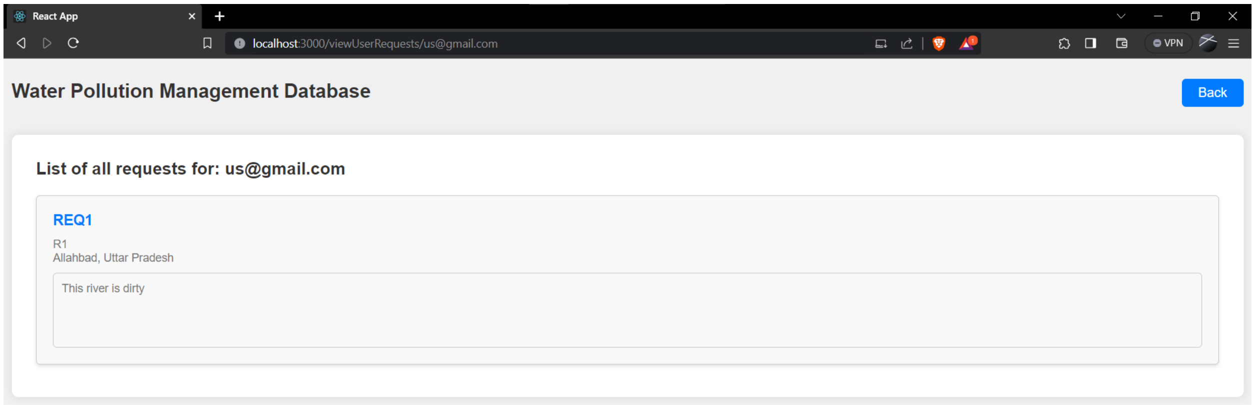
Allahbad

Uttar Pradesh

```
This river is dirty
```

Ganges
ID: R2
Level of Pollution: 4
Source: Gangotri

Viewing your requests as a user:



Water Pollution Management Database

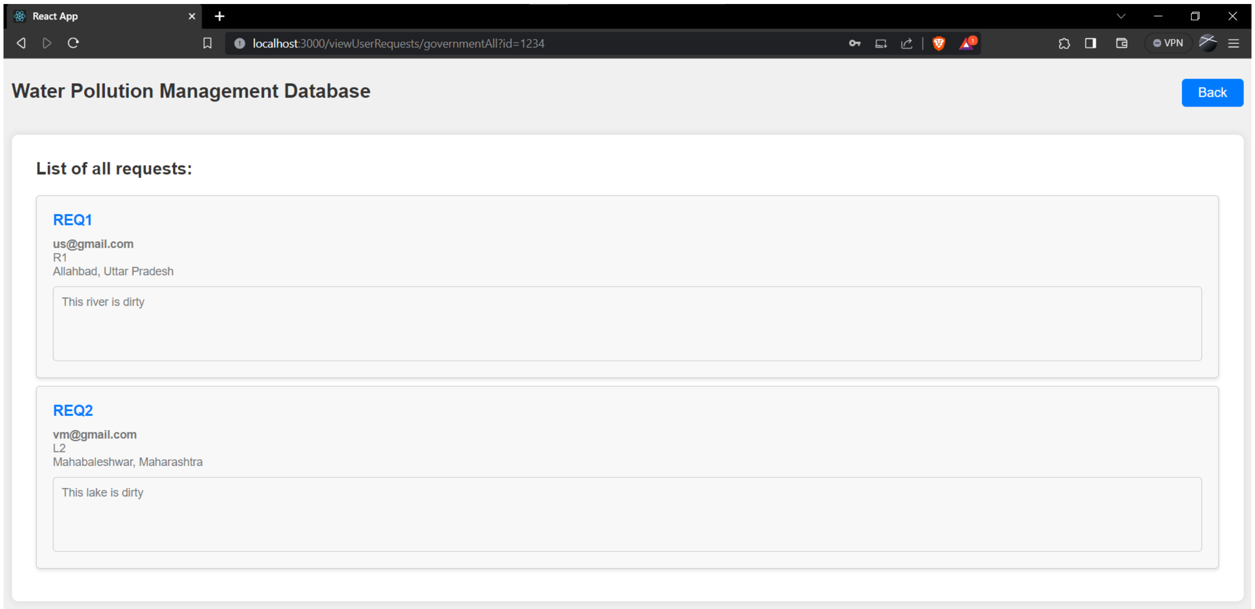
List of all requests for: us@gmail.com

REQ1

R1
Allahbad, Uttar Pradesh

This river is dirty

Viewing all requests as a government user:



Water Pollution Management Database

List of all requests:

REQ1

us@gmail.com
R1
Allahbad, Uttar Pradesh

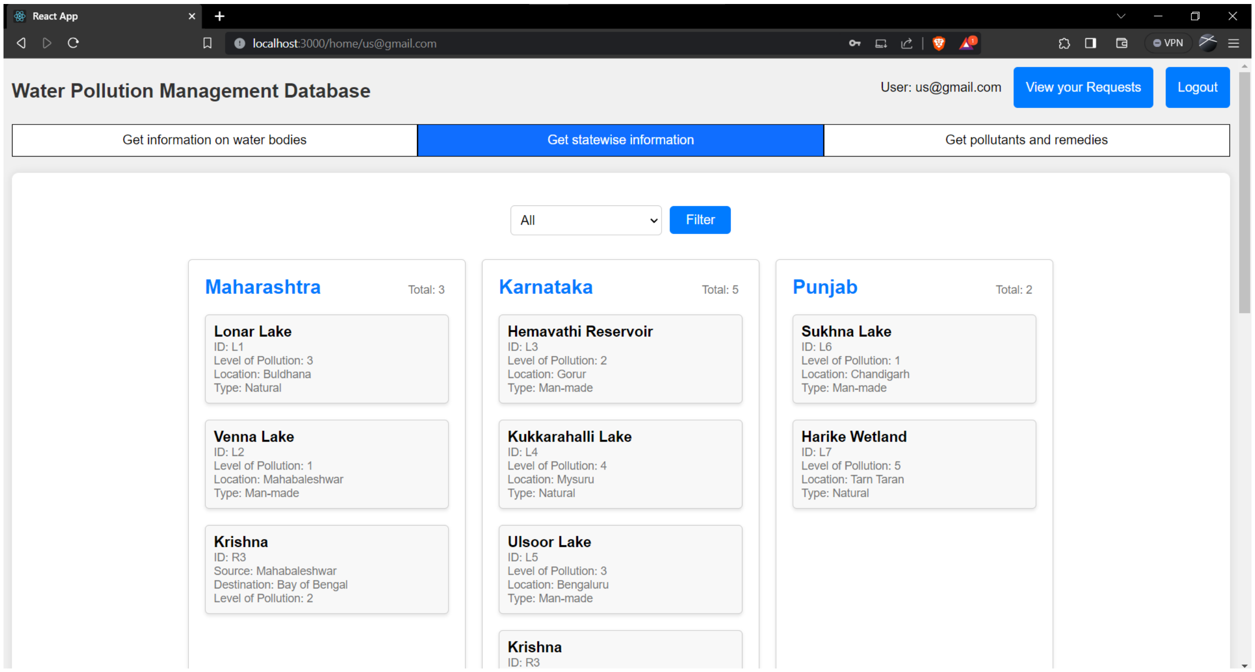
This river is dirty

REQ2

vm@gmail.com
L2
Mahabaleshwar, Maharashtra

This lake is dirty

6) Viewing state wise information


React App
localhost:3000/home/us@gmail.com
VPN
Water Pollution Management Database
User: us@gmail.com
View your Requests
Logout
Get information on water bodies
Get statewise information
Get pollutants and remedies
All
Filter

Maharashtra Total: 3

Lonar Lake
ID: L1
Level of Pollution: 3
Location: Buldhana
Type: Natural

Venna Lake
ID: L2
Level of Pollution: 1
Location: Mahabaleshwar
Type: Man-made

Krishna
ID: R3
Source: Mahabaleshwar
Destination: Bay of Bengal
Level of Pollution: 2

Karnataka Total: 5

Hemavathi Reservoir
ID: L3
Level of Pollution: 2
Location: Gorur
Type: Man-made

Kukkarahalli Lake
ID: L4
Level of Pollution: 4
Location: Mysuru
Type: Natural

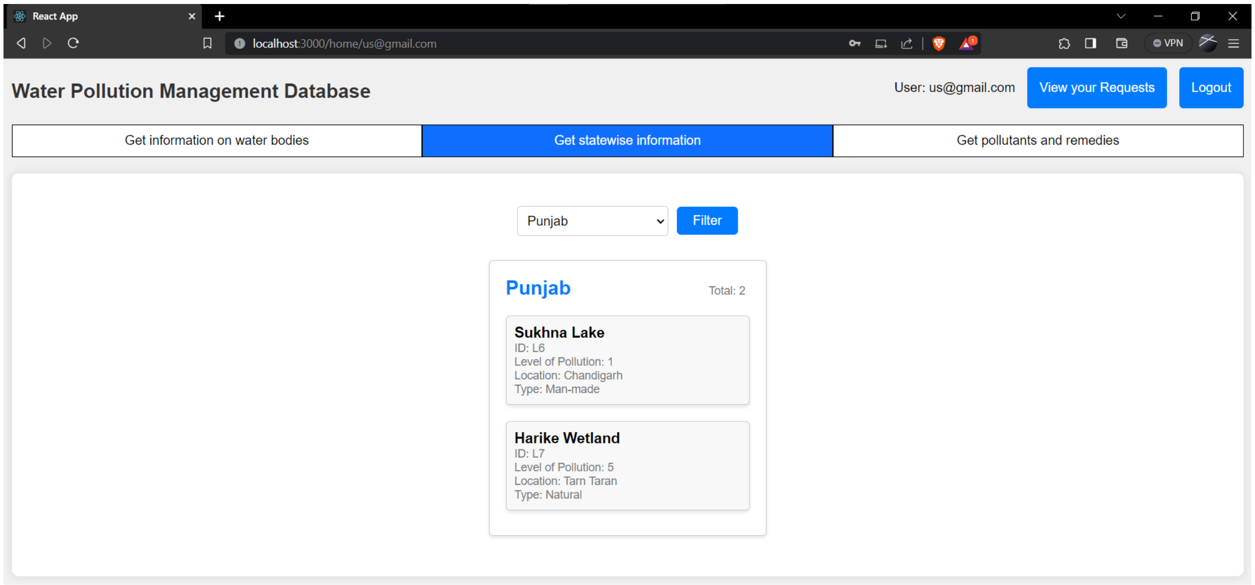
Ulsoor Lake
ID: L5
Level of Pollution: 3
Location: Bengaluru
Type: Man-made

Krishna
ID: R3

Punjab Total: 2

Sukhna Lake
ID: L6
Level of Pollution: 1
Location: Chandigarh
Type: Man-made

Harike Wetland
ID: L7
Level of Pollution: 5
Location: Tarn Taran
Type: Natural


React App
localhost:3000/home/us@gmail.com
VPN
Water Pollution Management Database
User: us@gmail.com
View your Requests
Logout
Get information on water bodies
Get statewise information
Get pollutants and remedies
Punjab
Filter

Punjab Total: 2

Sukhna Lake
ID: L6
Level of Pollution: 1
Location: Chandigarh
Type: Man-made

Harike Wetland
ID: L7
Level of Pollution: 5
Location: Tarn Taran
Type: Natural

● Procedures with Nested and Join queries, Functions with Aggregate queries and Triggers :

1. Procedures, Nested queries and Join queries:

```
USE miniProject;


DELIMITER //
CREATE PROCEDURE GetLakeDetails(IN id VARCHAR(10))
BEGIN
    WITH temp AS (
        SELECT
            L.lake_id AS lake_id,
            L.name AS lake_name,
            L.level_of_pollution,
            L.location,
            L.type,
            L.state_name,
            LP.pollutant_id,
            LP.concentration,
            LP.effect,
            P.name AS pollutant_name,
            P.chemical_composition,
            P.source,
            P.type AS pollutant_type,
            P.severity
        FROM
            (SELECT *
            FROM lakes
            WHERE lake_id = id) AS L
            JOIN lake_pollutants AS LP ON L.lake_id = LP.lake_id
            JOIN pollutants AS P ON LP.pollutant_id = P.pollutant_id
    )
    SELECT
        temp.*,
        R.remedy_id,
        R.name AS remedy_name,
        R.effect AS remedy_effect,
        R.scale AS remedy_scale,
        R.avg_price AS remedy_avg_price
    FROM temp
    JOIN remedies AS R ON temp.pollutant_id = R.pollutant_id;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE GetRiverDetails(IN id VARCHAR(10))
BEGIN
    WITH temp AS (
        SELECT
            RI.river_id AS river_id,
            RI.name AS river_name,
            RI.level_of_pollution,
            RI.source AS source,
            RI.destination AS destination,
            RP.pollutant_id,
            RP.concentration,
            RP.effect,
            P.name AS pollutant_name,
            P.chemical_composition,
            P.source AS pollutant_source,
            P.type AS pollutant_type,
            P.severity
        FROM
            (SELECT *
            FROM rivers
            WHERE river_id = id) AS RI
            JOIN river_pollutants AS RP ON RI.river_id = RP.river_id
            JOIN pollutants AS P ON RP.pollutant_id = P.pollutant_id
    )
    SELECT
        temp.*,
        R.remedy_id,
        R.name AS remedy_name,
        R.effect AS remedy_effect,
        R.scale AS remedy_scale,
        R.avg_price AS remedy_avg_price
    FROM temp
    JOIN remedies AS R ON temp.pollutant_id = R.pollutant_id;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE InsertUserDetails(IN user_email VARCHAR(50))
BEGIN
    SET @create_user_query = CONCAT("CREATE USER ", CONCAT_WS('@',
QUOTE(user_email), 'localhost'), ";");
    PREPARE create_user_stmt FROM @create_user_query;
    EXECUTE create_user_stmt;
    DEALLOCATE PREPARE create_user_stmt;

    SET @grant_query = CONCAT("GRANT 'users'@'localhost' TO ",
CONCAT_WS('@', QUOTE(user_email), 'localhost'), ";");
    PREPARE grant_stmt FROM @grant_query;
    EXECUTE grant_stmt;
    DEALLOCATE PREPARE grant_stmt;
END //
DELIMITER ;


DELIMITER //
CREATE PROCEDURE InsertGovernmentUserDetails(IN user_email
VARCHAR(50))
BEGIN
    SET @create_user_query = CONCAT("CREATE USER ", CONCAT_WS('@',
QUOTE(user_email), 'localhost'), ";");
    PREPARE create_user_stmt FROM @create_user_query;
    EXECUTE create_user_stmt;
    DEALLOCATE PREPARE create_user_stmt;

    SET @grant_query = CONCAT("GRANT 'governmentUsers'@'localhost' TO ",
CONCAT_WS('@', QUOTE(user_email), 'localhost'), ";");
    PREPARE grant_stmt FROM @grant_query;
    EXECUTE grant_stmt;
    DEALLOCATE PREPARE grant_stmt;
END //
DELIMITER ;
DELIMITER //
CREATE PROCEDURE InsertAdminDetails(IN user_email VARCHAR(50))
BEGIN
    SET @create_user_query = CONCAT("CREATE USER ", CONCAT_WS('@', QUOTE(user_email), 'localhost'), ";");
    PREPARE create_user_stmt FROM @create_user_query;
    EXECUTE create_user_stmt;
    DEALLOCATE PREPARE create_user_stmt;

    SET @grant_query = CONCAT("GRANT 'admins'@'localhost' TO ", CONCAT_WS('@', QUOTE(user_email), 'localhost'), ";");
    PREPARE grant_stmt FROM @grant_query;
    EXECUTE grant_stmt;
    DEALLOCATE PREPARE grant_stmt;
END //
DELIMITER ;
```

2. Functions, Aggregate queries :

```
USE miniProject;


DELIMITER //
CREATE FUNCTION getTotalBodiesInState(input_state VARCHAR(30))
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE number_of_rivers INT;
   DECLARE number_of_lakes INT;

   SELECT COUNT(*)
   INTO number_of_rivers
   FROM (
      SELECT state_name, river_id
      FROM rivers_in_state
      WHERE state_name = input_state
   ) AS num_rivers
   GROUP BY state_name;
```

```
    SELECT COUNT(*)
    INTO number_of_lakes
    FROM (
        SELECT state_name, lake_id
        FROM lakes
        WHERE state_name = input_state
    ) AS num_lakes
    GROUP BY state_name;

    IF number_of_lakes IS NULL AND number_of_rivers IS NULL THEN
        RETURN 0;
    ELSEIF number_of_lakes IS NULL THEN
        RETURN number_of_rivers;
    ELSEIF number_of_rivers IS NULL THEN
        RETURN number_of_lakes;
    ELSE
        RETURN number_of_rivers + number_of_lakes;
    END IF;
END //
DELIMITER ;
```

3. Trigger queries :

```
USE miniProject;

DELIMITER //
CREATE TRIGGER InsertLake
BEFORE INSERT ON lakes
FOR EACH ROW
BEGIN
    DECLARE number_of_rows INT;
    SELECT COUNT(*) INTO number_of_rows FROM lakes;
    SET NEW.lake_id = CONCAT("L", number_of_rows + 1);
END //
DELIMITER ;


DELIMITER //
```

```
CREATE TRIGGER InsertRiver
BEFORE INSERT ON rivers
FOR EACH ROW
BEGIN
   DECLARE number_of_rows INT;
   SELECT COUNT(*) INTO number_of_rows FROM rivers;
   SET NEW.river_id = CONCAT("R", number_of_rows + 1);
END //
DELIMITER ;


DELIMITER //
CREATE TRIGGER InsertRequest
BEFORE INSERT ON requests
FOR EACH ROW
BEGIN
   DECLARE number_of_rows INT;
   SELECT COUNT(*) INTO number_of_rows FROM requests;
   SET NEW.request_id = CONCAT("REQ", number_of_rows + 1);
END //
DELIMITER ;
```

- **Code snippets for invoking the Procedures/Functions/Trigger**

1) Procedures:

```
database.query(query, [...values, request.body.governmentID], (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }
        var userQuery
        if(request.body.governmentID == null) {
            userQuery = "CALL InsertUserDetails(?);"
        } else {
            userQuery = "CALL InsertGovernmentUserDetails(?);"
        }
        database.query(userQuery, request.body.email, (error, userResult) => {
            if(error) {
                console.log(error)
                return response.json(error)
            }
            return response.json(result)
        })
    })
```

```
app.get("/viewLakeDetails/:lakeID", (request, response) => {
    const query = "CALL GetLakeDetails(?);"
    database.query(query, request.params.lakeID, (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }

        const data = result[0]
        if(data.length === 0) {
            return response.json([])
        }
```

```
const lakeInfo = {
    lake_id: data[0].lake_id,
    lake_name: data[0].lake_name,
    level_of_pollution: data[0].level_of_pollution,
    location: data[0].location,
    type: data[0].type,
    state_name: data[0].state_name
}

const pollutants = {}
for(var i in data) {
    if(!pollutants[data[i]["pollutant_id"]]) {
        pollutants[data[i]["pollutant_id"]] = {
            pollutant_id: data[i]["pollutant_id"],
            pollutant_name: data[i]["pollutant_name"],
            concentration: data[i]["concentration"],
            effect: data[i]["effect"],
            chemical_composition: data[i]["chemical_composition"],
            source: data[i]["source"],
            pollutant_type: data[i]["pollutant_type"],
            severity: data[i]["severity"],
            remedies: {}
        }
    }
    if(!pollutants[data[i]["pollutant_id"]]["remedies"]["remedy_id"]) {
        pollutants[data[i]["pollutant_id"]]["remedies"][data[i]["remedy_id"]] = {
            remedy_id: data[i]["remedy_id"],
            remedy_name: data[i]["remedy_name"],
            remedy_scale: data[i]["remedy_scale"],
            remedy_effect: data[i]["remedy_effect"],
            avg_price : data[i]["remedy_avg_price"]
        }
    }
}

return response.json([
    lakeInfo,
    pollutants
])
```

```
    })
})


app.get("/viewRiverDetails/:riverID", (request, response) => {
    const query = "CALL GetRiverDetails(?);"
    database.query(query, request.params.riverID, (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }

        const data = result[0]
        if(data.length === 0) {
            return response.json([])
        }

        const lakeInfo = {
            river_id: data[0].river_id,
            river_name: data[0].river_name,
            level_of_pollution: data[0].level_of_pollution,
            source: data[0].source,
            destination: data[0].destination
        }

        const pollutants = {}
        for(var i in data) {
            if(!pollutants[data[i]["pollutant_id"]]) {
                pollutants[data[i]["pollutant_id"]] = {
                    pollutant_id: data[i]["pollutant_id"],
                    pollutant_name: data[i]["pollutant_name"],
                    concentration: data[i]["concentration"],
                    effect: data[i]["effect"],
                    chemical_composition: data[i]["chemical_composition"],
                    pollutant_source: data[i]["pollutant_source"],
                    pollutant_type: data[i]["pollutant_type"],
                    severity: data[i]["severity"],
                    remedies: {}
                }
            }
            if(!pollutants[data[i]["pollutant_id"]]["remedies"]["remedy_id"]) {
                pollutants[data[i]["pollutant_id"]]["remedies"][data[i]["remedy_id"]] = {
                    remedy_id: data[i]["remedy_id"],
                    remedy_name: data[i]["remedy_name"],
                    remedy_scale: data[i]["remedy_scale"],
                    remedy_effect: data[i]["remedy_effect"],
                    avg_price : data[i]["remedy_avg_price"]
                }
            }
        }

        return response.json([
            lakeInfo,
            pollutants
        ])
    })
})
```

2) Functions

```
app.get("/getTotalBodiesInState", (request, response) => {
    const query = "SELECT name, getTotalBodiesInState(name) AS total_count from states;"
    database.query(query, (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }
        const counts = {}
        for(var i in result) {
            counts[result[i]["name"]] = result[i]["total_count"]
        }
        return response.json(counts)
    })
})
```

3) Triggers

```
app.post("/insertLake", (request, response) => {
    const query = "INSERT INTO lakes VALUES (", ?, ?, ?, ?, ?);"
    const values = [
        request.body.name,
        request.body.level,
        request.body.location,
        request.body.type,
        request.body.state
    ]
    database.query(query, [...values], (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }
        return response.json(result)
    })
})


app.post("/insertRiver", (request, response) => {
    const query = "INSERT INTO rivers VALUES (", ?, ?, ?, ?);"
    const values = [
        request.body.name,
        request.body.level,
        request.body.source,
        request.body.destination
    ]
    database.query(query, [...values], (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }
        return response.json(result)
    })
})


app.post("/newRequest", (request, response) => {
    var query = ""
    if(request.body.body_id[0] === "R") {
        query = "INSERT INTO requests (`request_id`, `user_email`, `river_id`, `city`,
`state_name`, `content`) VALUES (", ?, ?, ?, ?, ?);"
    } else {
        query = "INSERT INTO requests (`request_id`, `user_email`, `lake_id`, `city`,
`state_name`, `content`) VALUES (", ?, ?, ?, ?, ?);"
    }
    const values = [
        request.body.user_email,
        request.body.body_id,
        request.body.city,
        request.body.state,
        request.body.content
    ]
    database.query(query, [...values], (error, result) => {
        if(error) {
            console.log(error)
            return response.json(error)
        }
        return response.json(result)
    })
})
```

The triggers are called before inserting into the lakes, rivers and requests tables as shown above.