Smart Water Management

# Project Objectives

The primary objective is to create a smart environmental monitoring system that collects data from various IoT sensors, processes it on a Raspberry Pi, and allows users to access and visualize this data through a mobile app.

Monitor and collect data on environmental parameters like temperature, humidity, air quality, and light levels.

Provide real-time data access and historical trend analysis for users through a mobile application.

# Key Objectives:

1. **Real-Time Data Visualization:** Provide users with real-time water consumption data from IoT sensors, allowing them to stay informed about water usage.
2. **Historical Data Storage:** Store historical water consumption data in a MongoDB database for analysis and reference.
3. **User Registration:** Implement a user registration system for personalized access to water consumption data and conservation tips.
4. **Water Conservation Tips:** Offer water conservation tips and information to educate and encourage users to reduce water consumption.
5. **Mobile App Development:** Develop a mobile application to extend access to water consumption data on mobile devices, enhancing user convenience.

# IoT Sensor Setup

**IoT Sensor Configuration:**

The project involves the installation and configuration of IoT sensors, such as flow meters, in public places to measure water consumption. These sensors are connected to a central data-sharing platform that collects and transmits data in real-time.

**IoT Data Flow:**

1. IoT sensors measure water consumption.

## Sensors

We have used the following sensors for our project:

Temperature

Humidity

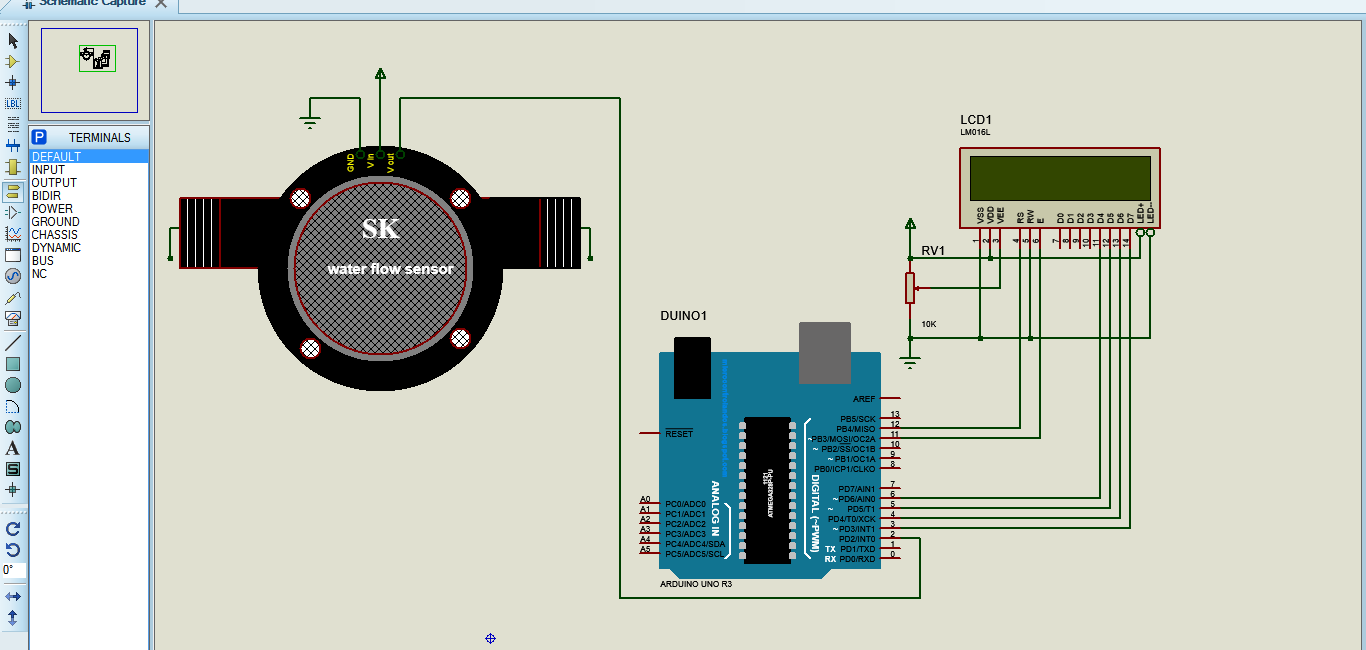
Water Flow

Water Meter

Water Level

Leak Detection

1. The data is transmitted to a central data-sharing platform.
2. The data-sharing platform processes and stores the data in a MongoDB database.
3. The web application and mobile app retrieve data from the database for display.



# Mobile App Development

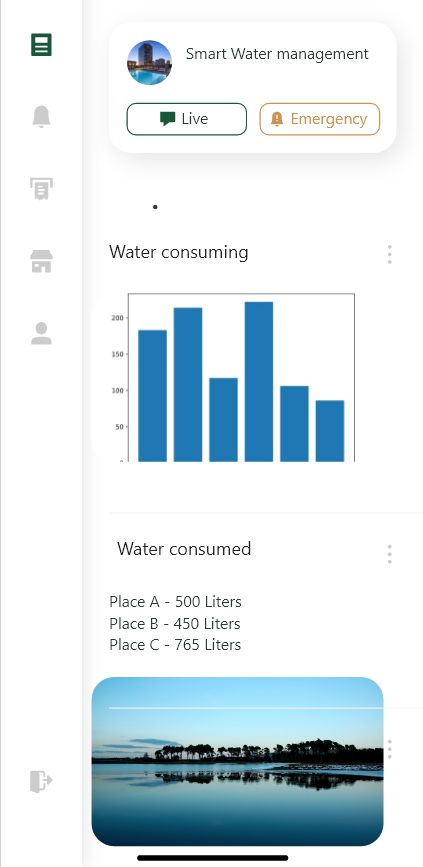
A mobile app for Android platforms that allows users to access the collected sensor data.

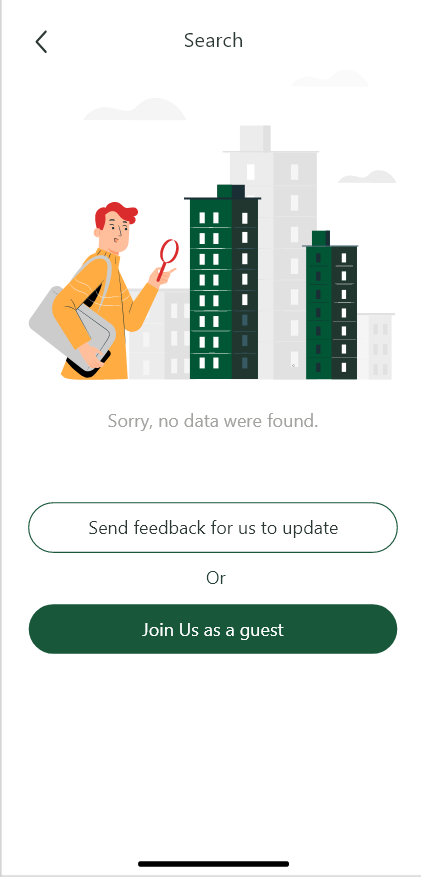
Using programming languages such as Java and appropriate frameworks such as Android Studio

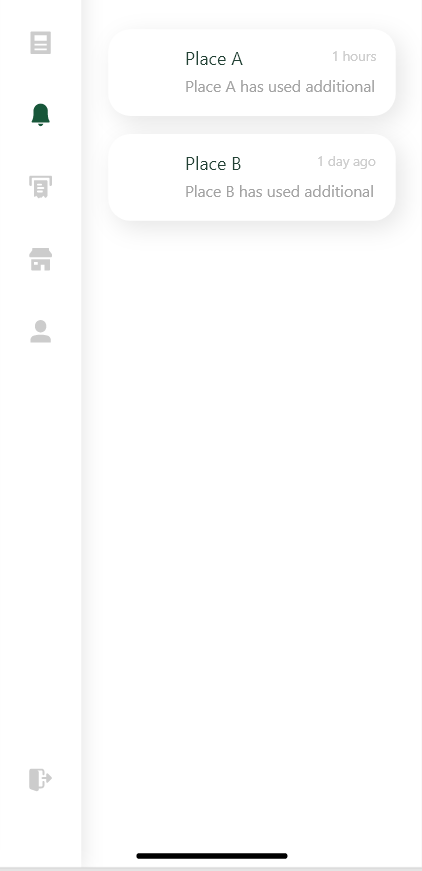
Implementing features for data visualization, user alerts, and data logging.

**Key Features of the Mobile App:**

1. **Real-Time Data Access:** Users can view real-time water consumption data for different locations.
2. **Historical Data Access:** Access historical water consumption data for analysis and reference.
3. **User Registration:** Register and log in to personalize the app experience.
4. **Push Notifications:** Send users water conservation tips and alerts based on their preferences.
5. **Interactive Interface:** Provide a user-friendly and interactive interface for easy navigation and data visualization.







# Code Implementation

**Web Development Technologies:**

* **HTML, CSS, and JavaScript:** The project utilizes these core web development technologies for building the user interface.
* **React.js:** React is used as a JavaScript framework for creating a dynamic and responsive user interface.

**Server-Side:**

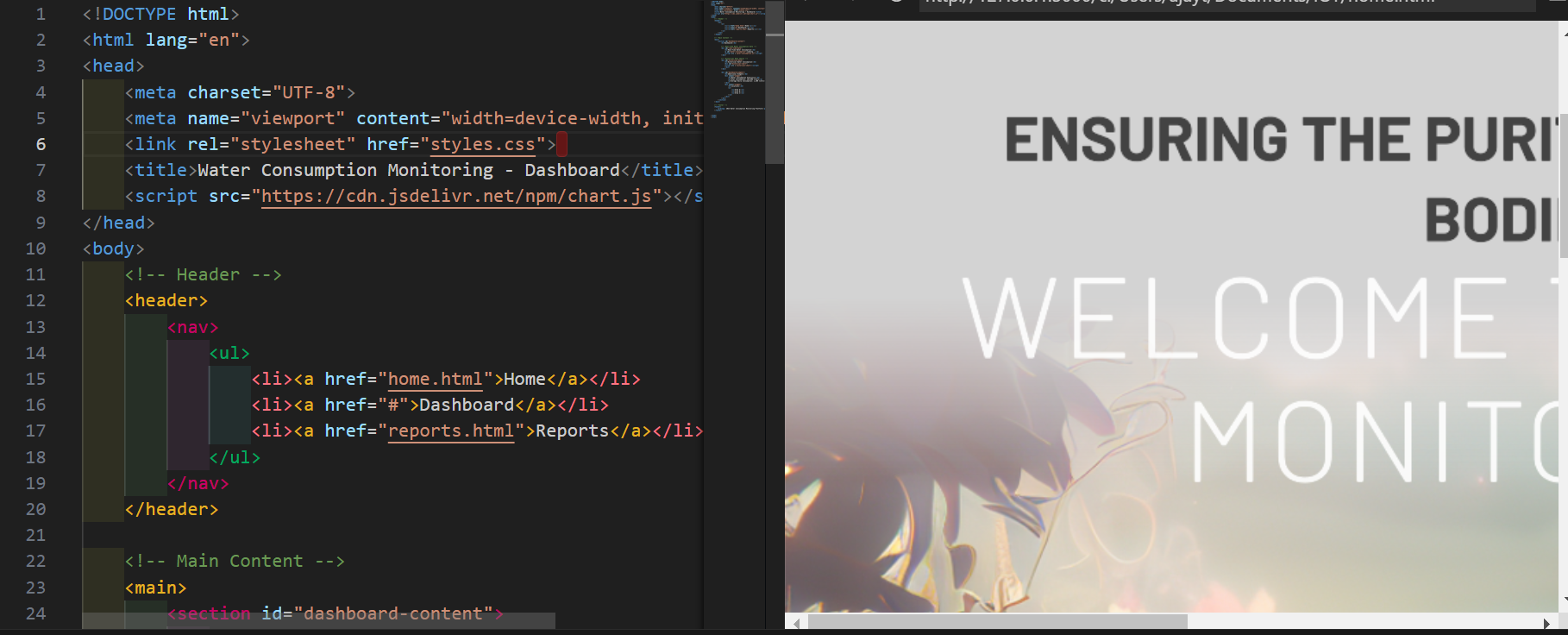
* **Node.js:** The server-side development is powered by Node.js, which handles routing, API endpoints, and data interactions.
* **Express.js:** The Express framework is used to create RESTful APIs for communication between the server and client.
* **Mongoose:** Mongoose, an ODM library, manages interactions with the MongoDB database.

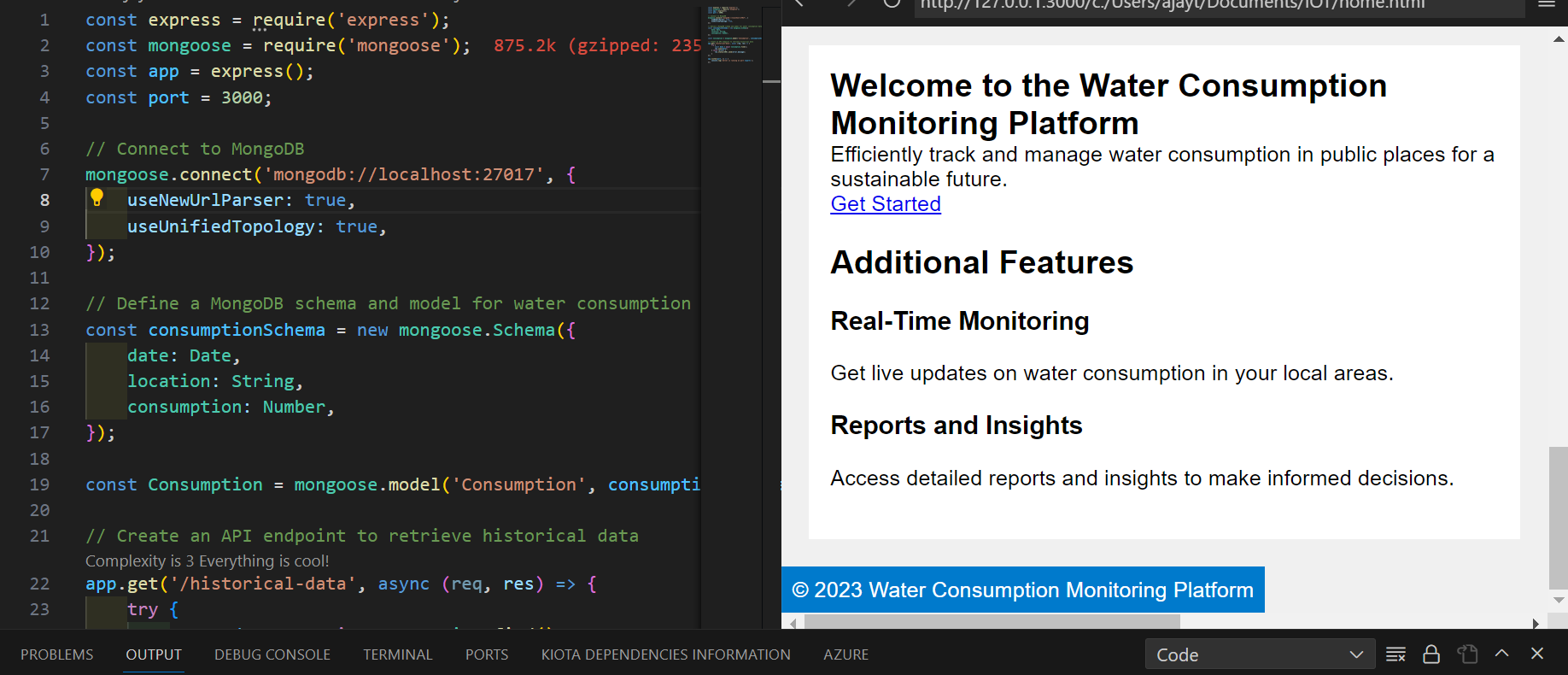
**Database:**

* **MongoDB:** MongoDB is the chosen database system for storing historical water consumption data. A suitable schema is implemented for data storage.

**Real-Time Data Handling:**

* **AJAX or WebSocket:** Real-time water consumption data is fetched from the server using AJAX or WebSocket, ensuring updates in real-time.





The project code implementation encompasses both server-side and client-side development, emphasizing real-time data retrieval, user registration, and data presentation. The use of a MongoDB database supports historical data storage, while the mobile app extends the project's accessibility to mobile users. This multifaceted approach aims to meet the project's objectives of promoting water conservation through informed user interaction.

Replicating the Project– IoT Sensors and Platform

# IoT Sensor Deployment

1. **Select IoT Sensors:** Choose the appropriate sensors for your project, such as GPS sensors for location tracking, temperature sensors for environmental monitoring, and any other sensors relevant to your transit information system.
2. **Connect Sensors:** Connect the sensors to microcontrollers or IoT devices (e.g., Arduino, Raspberry Pi, ESP8266) for data collection. Ensure power and data connectivity.
3. **Programming:** Write code to read data from the sensors, format it, and send it to a central server. You can use Python for coding the microcontrollers or devices.
4. **Data Transmission:** Establish a communication protocol (e.g., MQTT, HTTP) to transmit sensor data to a central server or cloud platform. Python libraries like MQTT or REST APIs can be used for this purpose.

# Developing the Platform

1. **Database Setup:** Create a database to store the incoming sensor data. You can use databases like MySQL, PostgreSQL, or NoSQL databases, and interact with them using Python libraries like SQLAlchemy.
2. **Web Server:** Set up a web server using Python frameworks like Flask, Django, or FastAPI to create APIs for data retrieval and interaction.
3. **Data Processing:** Write Python code to process the incoming sensor data, extract relevant information, and store it in the database.
4. **User Interface:** Create a web-based or mobile application interface to display transit information to users. You can use web development frameworks (e.g., React, Angular, or Vue) for the frontend and connect it to the Python backend.

# Integration Using Python

1. **Data Integration:** The Python backend of your transit information platform should continuously receive data from the IoT sensors and update the database.
2. **APIs:** Expose APIs that can be used by the IoT devices to send data and by the user interface to request transit information.
3. **Real-Time Updates:** Implement real-time functionality to push transit updates to users in real-time using libraries like WebSockets.
4. **Security:** Implement security measures like authentication and encryption to protect data and user privacy.
5. **Testing and Deployment:** Thoroughly test the entire system to ensure it functions as expected. Once tested, deploy it on servers or cloud platforms to make it accessible to users.
6. **Monitoring and Maintenance:** Set up monitoring tools to keep an eye on the system’s health and performance. Regularly update and maintain the system as needed.

The success of this project will depend on the proper integration of these components and the effectiveness of the code implementation. Detailed design, hardware selection, and programming are crucial aspects to achieve the project’s objectives.