Project Notes:

Main Program Implemented (Water Tracker)

**Microservices Plan (mine):**

Data Management Service - This microservice will be responsible for handling all data-related operations, ensuring that the application's data handling is centralized and isolated from other functionalities, which aligns with microservices best practices.

Responsibilities:

* User registration and management.
* Water intake logging, updating, and deletion.
* Storing and retrieving historical data.

Endpoints:

* POST /users - Register a new user.
* GET /users/{username} - Retrieve user data.
* POST /intake - Log new water intake.
* PUT /intake/{id} - Update an existing intake record.
* DELETE /intake/{id} - Delete a specific intake record.

Analytics Service - Analytics Service

This service will analyze the water intake data and provide users with insights, helping them understand their hydration patterns better and making informed decisions about their water consumption habits.

Responsibilities:

* Generate daily, weekly, monthly, and yearly reports.
* Provide insights such as average intake, trends, and comparisons against hydration goals.

Endpoints:

* GET /analytics/daily/{username}
* GET /analytics/weekly/{username}
* GET /analytics/monthly/{username}
* GET /analytics/yearly/{username}

Notification Service(?) -

A service designed to engage users by reminding them to log their intake or alerting them when they are not meeting their hydration goals. This can help improve user adherence to desired hydration levels.

Responsibilities:

* Send reminders to users.
* Alert users if they haven’t logged water intake by a certain time.
* Provide motivational messages or hydration tips.

Endpoints:

* POST /notify/remind/{username}
* POST /notify/alert/{username}

Planning and Implementation Steps:

Define the Data Schema: Each service will need a clear data schema for the information it manages.

Choose Technology Stack: Decide on the technologies for each microservice (e.g., Node.js, Python Flask, databases like MongoDB or MySQL).

Develop Independently: Each microservice can be developed independently by different teams or at different times, as long as they adhere to the defined API contracts.

Testing and Integration: Ensure thorough testing of each microservice independently and then how they interact with each other