Backend

The **backend** is the part of an application that works behind the scenes to handle data, run logic, and communicate with the frontend. We use **Node.js** to build the backend because it is fast, lightweight, and uses JavaScript, a language many developers are already familiar with. The **server** listens to requests and sends responses, the **database** stores information, and **APIs** provide a way for the frontend and backend to communicate.

What is Backend Development?

The backend is like the **kitchen of a restaurant**. When you order food (make a request), the chef (backend) prepares it, and the waiter (frontend) brings it to your table. It ensures data is stored, processed, and delivered correctly.

Why Do We Use Node.js?

Node.js helps us build the backend efficiently. It's:

- **Fast and Non-blocking:** Like a multitasking chef who can handle multiple orders without waiting for one to finish.
- **Uses JavaScript Everywhere:** You don’t need to learn a new language for backend development.

Real-life Example: When you use Uber, Node.js helps the app handle many users booking rides at the same time without slowing down.

What is a Server?

A **server** is like a **receptionist** that listens to requests (e.g., "Show me today’s specials") and sends the right response ("Here’s the list of specials").

Real-life Example: Netflix’s server sends you the list of movies you can watch when you log in.

What is a Database?

A **database** is a **digital notebook** where all app data is saved. It keeps everything organized and accessible.

- **MongoDB:** We'll use this database, which stores data in JSON-like objects, making it simple and flexible.
- **Real-life Example:** WhatsApp uses a database to store your chats, and Facebook uses it for posts and profiles.

What are APIs?

An **API** is like a **restaurant menu** that tells you what you can order (GET, POST, PUT, DELETE). APIs allow the frontend (what users see) to communicate with the backend.

Example:

- GET: "Show me all restaurants in Lagos."
- POST: "Add a new jollof rice recipe."
- PUT: "Update the price of fried rice."
- DELETE: "Remove a recipe from the menu."

Real-life Use Case: Jollof Rice Delivery App

If you were building a Jollof Rice Delivery app:

1. **Node.js:** Ensures the app can handle many users ordering jollof rice at the same time.
2. **Server:** Receives requests like "Find restaurants near me" and sends the list back.
3. **Database:** Stores restaurant details, menus, and customer orders.
4. **API:** Lets the frontend ask for data from the backend, like:
    - **GET:** "Show me all spicy jollof rice recipes."
    - **POST:** "Place an order for jollof rice and chicken."
    - **PUT:** "Update my delivery address."
    - **DELETE:** "Cancel my order."
