**Introduction to Backend Development**

**Part 1: Theory**

**1. What is a Backend?**

- Definition: The "server-side" of a web app that handles data, logic, and communication with databases.

**- Analogy:** Like the kitchen in a restaurant (frontend = dining area).

**- Key Responsibilities:**

- Data storage/retrieval (e.g., databases).

- Authentication/authorization.

- Business logic (e.g., payment processing).

**2. What is an API?**

**- Definition:** Application Programming Interface — a set of rules for communication between systems.

**- Example:**

- A weather app (frontend) asks an API for data, and the backend returns JSON like:

*```json*

*{ "city": "London", "temp": 22, "unit": "C" }*

*```*

**- REST API:** Uses HTTP methods (GET, POST, etc.) and URLs (endpoints) to perform actions.

**3. Node.js & Express.js Basics**

**- Node.js:** A JavaScript runtime for server-side code (non-blocking, event-driven).

**- Express.js:** A framework to simplify building web apps/APIs with Node.js.

**Part 2: Setup**

**Activity:** Install Node.js and Set Up Project

**1. Install Node.js**

- Download from *[nodejs.org](https://nodejs.org).*

- Verify installation:

*node -v # Should show version (e.g., v18.x.x)*

*npm -v # Node Package Manager version*

**2. Create Project Folder**

*mkdir my-first-backend*

*cd my-first-backend*

**3. Initialize Node Project**

*npm init -y # Creates a package.json file*

**Part 3: Hello World App**

Activity: Build a Simple Server with Node.js

**1.Create `server.js`**

***// Import Node's built-in HTTP module***

*const http = require('http');*

***// Create a server***

*const server = http.createServer((req, res) => {*

*res.writeHead(200, { 'Content-Type': 'text/plain' });*

*res.end('Hello World from Node.js!');*

*});*

***// Start the server on port 3000***

*server.listen(3000, () => {*

*console.log('Server running at http://localhost:3000');*

*});*

**2. Run the Server**

*node server.js*

- Visit `*http://localhost:3000*` in a browser. You should see "**Hello World from Node.js!**".

3. **Modify the Response**

- Change the text in `*res.end()*` and restart the server to see updates.

**Part 4: Express.js Basics**

Activity: Build a REST API with Express

**1. Install Express**

*npm install express*

**2. Create `app.js`**

*const express = require('express');*

*const app = express();*

*const port = 4000;*

***// Basic GET endpoint***

*app.get('/', (req, res) => {*

*res.send('Welcome to my first Express API!');*

*});*

***// GET endpoint returning JSON***

*app.get('/users', (req, res) => {*

*const users = [*

*{ id: 1, name: 'Alice' },*

*{ id: 2, name: 'Bob' }*

*];*

*res.json(users);*

*});*

***// Start server***

*app.listen(port, () => {*

*console.log(`Express server running at http://localhost:${port}`);*

*});*

3. Run the Express Server

*node app.js*

**- Test endpoints:**

- `*http://localhost:4000*` → Welcome message.

- `*http://localhost:4000/users*` → JSON user list.

**4.Add a POST Endpoint (Bonus)**

***// Middleware to parse JSON requests***

*app.use(express.json());*

*app.post('/users', (req, res) => {*

*const newUser = req.body;* ***// Assume client sends { name: "Charlie" }***

*res.status(201).json({ message: 'User created', user: newUser });*

*});*

- Test with Postman or curl:

*curl -X POST -H "Content-Type: application/json" -d '{"name":"Charlie"}' http://localhost:4000/users*

**Part 5: Recap & self activities**

**- Key Takeaways:**

- Backend handles logic/data; APIs enable communication.

- Node.js runs JS on the server; Express simplifies routing.

- Homework:

- Add a `/products` endpoint that returns a list of products.

- Experiment with POST requests to add dummy data.

**Troubleshooting Tips**

**- Port Conflicts:** Change the port number if you see `EADDRINUSE`.

**- Installation Issues:** Use `nvm` (Node Version Manager) if Node.js isn’t working.