

Backend Bootcamp

Session 1: Introduction to Backend Development

Slide 1: Welcome!

- 🙌 Welcome to Backend Bootcamp
 - 🖐️ What you'll learn:
 - Build scalable APIs using Node.js and Express
 - Connect and manage data with MongoDB
 - Structure projects using best practices
 - Deploy backend applications to the cloud
 - 🖐️ Tools: VS Code (IDE), Postman (API testing), GitHub (version control)
 - 😓 Goal: Equip you with real-world backend development skills
-

Slide 2: What is Backend Development?

- Backend = Behind-the-scenes logic of a web application
 - Responsibilities:
 - Handle user authentication and authorization
 - Communicate with databases to store/retrieve data
 - Perform calculations, enforce business logic
 - Send appropriate HTTP responses to frontend
 - Example: You enter login credentials → backend checks DB → returns token
-

Slide 3: Frontend vs Backend

Frontend	Backend
Runs in the browser	Runs on the server
Built with HTML, CSS, JavaScript	Built with Node.js, Python, etc.
Manages UI/UX	Handles data, logic, security
Fetches data from backend	Serves data to frontend
- Together they form a complete web application	

Slide 4: Fullstack Developer

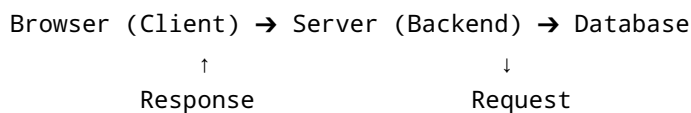
- A fullstack dev works on both frontend and backend
- Can build a complete application from UI to database
- Tech stack examples:

- MERN: MongoDB, Express, React, Node
 - MEVN: MongoDB, Express, Vue, Node
 - Useful in startups and small teams where devs wear multiple hats
-

Slide 5: What Happens When You Type `www.facebook.com`?

- 🤔 Step 1: You open your browser and type `www.facebook.com`
- 🧠 Step 2: The browser finds out which computer (server) runs Facebook
- 😊 Step 3: It sends a message (called a request) asking for the Facebook page
- 🤝 Step 4: Facebook's server receives the request and gets ready to respond
- 😊 Step 5: It sends back the web page (HTML, CSS, JavaScript, images, etc.)
- 😊 Step 6: Your browser displays the page using what it got
- 😊 Extra: The browser might keep asking the server for more data (like new messages)

Slide 6: Client-Server Architecture



- The client (browser/mobile app) makes a request - The server processes the request (e.g., fetches from DB) - The response (usually JSON) is sent back to the client - This interaction uses HTTP as the communication protocol

Slide 7: HTTP Basics

- HTTP = HyperText Transfer Protocol
 - It's stateless: each request is independent
 - Common HTTP methods:
 - **GET**: Read data (e.g., get a user profile)
 - **POST**: Create data (e.g., submit a form)
 - **PUT/PATCH**: Update data (e.g., change password)
 - **DELETE**: Remove data (e.g., delete an account)
 - Status Codes:
 - 200 OK: Request succeeded
 - 201 Created: New resource created
 - 400 Bad Request: Invalid input
 - 401 Unauthorized: Not logged in
 - 404 Not Found: Resource doesn't exist
 - 500 Server Error: Something went wrong
-

Slide 8: Common Backend Stack

- **Language**: JavaScript (runs on backend with Node.js)
- **Runtime**: Node.js

- Allows JavaScript to run on the server
 - Built on Chrome's V8 engine
 - Event-driven, non-blocking I/O for scalability
 - **Framework:** Express
 - Minimalist web framework built on Node.js
 - Helps define routes and middleware easily
 - Speeds up API development
 - **Database:** MongoDB
 - NoSQL database using documents (JSON-like format)
 - Scalable, flexible, and easy to integrate with Node.js
 - Uses Mongoose for schema and data modeling
 - **IDE:** Visual Studio Code (lightweight, extensions support)
 - **API Client:** Postman (for testing API endpoints)
 - This stack is popular due to its speed, scalability, and flexibility
-

Slide 9: Tools Setup

- 🖐️ Make sure to install:
 - **Node.js:** Runtime environment to run JavaScript server-side
 - **VS Code:** Writing, debugging, and navigating code
 - **Postman:** Send HTTP requests and inspect responses
 - **GitHub account:** Collaborate and store code in the cloud
 - Optional: Install MongoDB locally or sign up for MongoDB Atlas (cloud DB)
-

Slide 10: Hands-On: Test Public API

- Open Postman
- Send a GET request to:

```
https://jsonplaceholder.typicode.com/posts
```

- Inspect:
- Headers: metadata (content-type, auth)
- Body: actual data returned (posts in JSON format)
- Status: HTTP status (should be 200 OK)
- Try changing endpoint to:

```
/posts?userId=1
```

- This is how real frontend apps fetch filtered data from backend
-

Slide 11: Homework

- 🖐️ Install all required tools (Node.js, VS Code, Postman)
- 🖐️ Explore at least 2 endpoints from [JSONPlaceholder](https://jsonplaceholder.typicode.com/)

- 🖐️ Read article: "What is a REST API?" (link to be shared by instructor)
 - 🖐️ Bonus: Try creating a GitHub account if you don't already have one
-

Slide 12: Next Session Preview

- JavaScript Deep Dive:
 - ES6+ Features
 - Functions & arrow functions
 - Arrays and methods like `map`, `filter`, `reduce`
 - Callbacks and Promises (intro)
 - Quiz: Based on JavaScript basics
 - Hands-on: Write and run JavaScript code using Node
-

Thank You & Q/A

- 🙋 Any questions about today's material?
- 📅 Next class: JavaScript foundations
- 😊 Contact instructor for any setup help
- 😊 Stay curious, ask questions, and code daily!