DexNotePro: Backend Development Foundations

Welcome to Backend Development Foundations

In this course, you'll discover what powers your favorite websites and apps from behind the scenes — servers, APIs, databases, and logic that keep everything running smoothly. By the end, you'll understand how to build and deploy your own backend system.

---

1. Understanding the Backend

The **backend** is everything users don't see — the logic, databases, and servers that make a web app work.
When you send a message, make a post, or log in, it's the backend that processes your request.

Key Components:

- **Server:** Listens to requests and sends back responses.
- **Database:** Stores information like user data, posts, or transactions.
- **API (Application Programming Interface):** Connects frontend and backend.
- **Authentication:** Ensures users are who they say they are.

Try This:
Use a simple analogy — imagine the frontend as a restaurant waiter and the backend as the kitchen. Draw this as a small diagram in your notebook.

---

2. Introduction to Node.js

Node.js lets you use **JavaScript on the server**. It's fast, lightweight, and ideal for building modern APIs.

Basic Setup Steps:

1. Download and install Node.js from nodejs.org.
2. Create a folder for your project.
3. In your terminal, type:
4. `npm init -y`
5. `npm install express`
6. Create a file `server.js`:
7. `const express = require('express');`
8. `const app = express();`
9. `app.get('/', (req, res) => res.send('Hello from Backend!'));`
10. `app.listen(3000, () => console.log('Server running on port 3000'));`

Try This:
Run `node server.js` and visit http://localhost:3000. You've just launched your first backend!

---

3. Building REST APIs

An **API** (Application Programming Interface) allows the frontend and backend to communicate using **HTTP requests**.

Common Methods:

- **GET** – Read data
- **POST** – Create new data
- **PUT/PATCH** – Update existing data
- **DELETE** – Remove data

Example:

```
app.get('/users', (req, res) => {
  res.json([{ name: 'Aarav' }, { name: 'Diya' }]);
});
```

Try This:
Create a new route `/about` that returns a short JSON about your backend project.

---

4. Working with MongoDB

MongoDB is a NoSQL database that stores data as flexible **JSON-like documents**.

Steps to Use:

1. Create a free account at MongoDB Atlas.
2. Get your **connection string** (it looks like `mongodb+srv://...`).
3. Install MongoDB library:

```
npm install mongoose
```

5. Connect and define a model:

```
const mongoose = require('mongoose');
mongoose.connect('your-connection-string');
const User = mongoose.model('User', { name: String });
new User({ name: 'Ishaan' }).save();
```

Try This:
Add a new model for “Courses” with a name and completion status field.

---

5. Environment Variables and Secrets

Never expose your database passwords or API keys in public code.

Use a `.env` file:

```
PORT=3000
MONGO_URI=your_connection_string
```

And load it in your code:

```
require('dotenv').config();
```

Try This:
Add `dotenv` and configure your environment variables properly.

---

6. Authentication Basics

Authentication verifies user identity.

Popular Tools:

- **JWT (JSON Web Token):** Used for secure session management.
- **bcrypt:** For password hashing.

Example Flow:

1. User signs up → password hashed → saved in database.
2. User logs in → server checks hash → sends back JWT token.
3. User sends token in headers for future requests.

Try This:
Create a signup route that takes `username` and `password`, hashes it, and stores it in MongoDB.

---

7. Structuring a Real Project

A good backend is **organized**:

```
/backend
  ├── /models
  ├── /routes
  ├── /controllers
  ├── server.js
```

Each folder handles a part of the logic — models define data, routes handle requests, controllers define what happens.

Try This:
Split your `server.js` routes into separate folders and import them back.

---

8. Testing and Debugging

Use **Postman** or **Thunder Client** to test your APIs.
Look for:

- Response codes (200, 404, 500)
- Response time
- Data accuracy

Add helpful logs using:

```
console.log("User created:", user);
```

Try This:
Test all your API routes one by one and note the response status codes.

---

9. Deployment

Once your backend works locally, deploy it online.

Popular Options:

- **Render.com**
- **Vercel**
- **Railway.app**

Each can host Node.js servers with MongoDB integrations.

Try This:
Deploy your API and test it from your phone browser.

---

10. Where to Go Next

- Add more routes and connect with your frontend.
- Integrate authentication with your DexNotePro account system.
- Learn about advanced backend tools like **GraphQL**, **Redis**, and **Docker**.

✅ Congratulations!

You've completed **Backend Development Foundations** with DexNotePro.
Now, head to **ishaan7india.github.io/DexNotePro** and mark this course as **Complete** to track your progress and unlock your next challenge.