🎓 Introductory Node.js Course

📚 Section 1: What is Programming? What is Node.js? Setup and Hello World

🔰 BASIC LEVEL

🧠 Concepts:

- What is programming?
- Introduction to JavaScript & Node.js
- What is the difference between client-side JS and server-side JS?
- What can Node.js do?
- Setting up Node.js
- Writing and running your first Node.js program

💡 Explanation:

> Programming is giving a computer instructions to perform tasks.
> Node.js is a **runtime environment** that allows you to run JavaScript **outside the browser**, especially on servers.

⚙️ Setup Guide:

✅ Step-by-Step

1. **Install Node.js:**
    - Go to https://nodejs.org

- Download **LTS version** for stability
- Install it (click through defaults)

2. **Verify Installation:**

```
node -v
npm -v
```

3. **Create a Project Folder:**

```
mkdir my-first-node
cd my-first-node
```

4. **Create your first file:**

`hello.js`

```
console.log("Hello, World from Node.js!");
```

5. **Run the file:**

```
node hello.js
```

✅ You've just run your first Node.js program!

✍️ Classwork:

1. Create a file `welcome.js` that prints:
   - A greeting message
   - Your name
   - Today's date using `new Date()`
2. Run the file and take a screenshot of the output.

🌍 Real-World Use Case:

> Logging service boot-up messages on the server when the app starts.

🟡 INTERMEDIATE LEVEL

🧠 Concepts:

- Understanding `require` and modules
- Working with the file system
- Creating your first web server
- Introduction to package.json and npm

💡 Explanation:

> Node.js is modular. You can import built-in modules like `fs`, `http`, or third-party ones via `npm`.
> A web server is just a program that listens to HTTP requests and sends responses.

✅ Example: Simple Web Server

```
// server.js
const http = require("http");

const server = http.createServer((req, res) => {
    res.writeHead(200, { "Content-Type": "text/plain" });
    res.end("Hello from your first Node.js server!");
});

server.listen(3000, () => {
    console.log("Server is running on http://localhost:3000");
});
```

Run with:

```
node server.js
```

✍️ Classwork:

1. Create a server that:
    - Responds with your name and favorite quote
    - Listens on port 4000
2. Test it in your browser or Postman.

🌍 Real-World Use Case:

> This is how basic APIs and backend services are created for apps.

🔴 ADVANCED LEVEL

🧠 Concepts:

- Environment variables with `.env`
- Installing external packages (e.g., Express.js)
- Creating routes
- JSON responses and API basics
- Nodemon for development

✅ Setup: Install Express

```
npm init -y
npm install express dotenv
npm install -D nodemon
```

Update `package.json` scripts:

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

✅ Example: Simple API with Express

```javascript
// index.js
require('dotenv').config();
const express = require('express');
const app = express();
const PORT = process.env.PORT || 5000;

app.get('/', (req, res) => {
    res.json({ message: "Welcome to my API!" });
});

app.get('/about', (req, res) => {
    res.json({ developer: "Joel", stack: "Full Stack Developer" });
});

app.listen(PORT, () => {
    console.log(`Server running on port ${PORT}`);
});
```

Create a `.env` file:

```
PORT=5000
```

Run the app:

```
npm run dev
```

✍️ Classwork:

1. Create two new routes:
   - `/greet` → returns your name, a message, and current time
   - `/skills` → returns a list of 5 skills in JSON

🌍 Real-World Use Case:

> This is how APIs power your Flutter, React, or mobile apps — sending data like users, products, etc.

🧠 BONUS: Summary Table

Level	Topics Covered	Tools Used
Basic	Hello World, Setup, Console log	Node.js only
Intermediate	Modules, File system, Basic Server	`fs`, `http`, Postman
Advanced	Express, API routes, dotenv, nodemon	Express, Dotenv, Nodemon