# Node.js + Express Training Program (2 Weeks)

## Week 1 – Foundations

---

### Day 1 – Node.js Fundamentals

**Why Learn:**
Understand Node.js architecture, event loop, and asynchronous programming to build scalable apps.

**Tasks:**

- Build a CLI tool that prints system info (OS, memory, uptime).
  Create a basic HTTP server without Express.

---

### Day 2 – Express Basics

**Why Learn:**
Express simplifies web development with routing, middleware, and request/response handling.

**Tasks:**

- Create routes for users and posts.
- Write custom middleware for logging request details.

---

### Day 3 – REST API Design

**Why Learn:**
Follow REST principles for scalable APIs and standard error handling.

**Tasks:**

- Build CRUD API for posts.
- Add a centralized error handler for invalid routes and bad requests.

---

## Day 4 – Async Patterns

**Why Learn:**
Node.js apps rely heavily on asynchronous programming for performance.

**Tasks:**

- Compare callback vs Promise vs async/await with examples.
- Implement a worker thread for CPU-heavy tasks.

---

## Day 5 – Database Integration

**Why Learn:**
Persisting and querying data is essential for any real-world application.

**Tasks:**

- Connect Node.js with MongoDB or PostgreSQL.
- Build CRUD operations for posts using the database.

---

## Day 6 – Authentication (Part 1)

**Why Learn:**
Apps need secure user login and authentication.

**Tasks:**

- Build signup and login APIs.
- Protect private routes with JWT authentication.

---

## Day 7 – Authentication (Part 2) + File Uploads

**Why Learn:**
Handle role-based access (admin/user) and manage media uploads.

**Tasks:**

- Implement role-based authorization (only admin can delete posts).

- Upload profile picture to AWS S3.

---

### Day 7.5 – File Processing

**Why Learn:**
Work with large files efficiently (streams, image processing).

**Tasks:**

- Implement file streaming for large text files.
- Resize and compress an image before saving.

---

## Week 2 – Advanced Features

---

### Day 8 – Emails + SMS + Background Jobs

**Why Learn:**
Enable notifications and asynchronous job handling in production apps.

**Tasks:**

- Send verification email using Nodemailer.
- Send SMS with Twilio API.
- Use BullMQ to schedule a background job (e.g., welcome email).

---

### Day 9 – Error Handling + Logging

**Why Learn:**
Improve debugging and monitoring for production reliability.

**Tasks:**

- Create centralized error middleware.
- Implement structured logging with Winston and log rotation.

---

## Day 10 – Security Best Practices

**Why Learn:**
Protect applications from real-world security threats (XSS, CSRF, SQL Injection, DoS).

**Tasks:**

- Add Helmet middleware for HTTP security headers.
- Implement rate limiting & request validation.

---

## Day 10.5 – Event-Driven Architecture

**Why Learn:**
Build scalable systems using events and messaging.

**Tasks:**

- Use EventEmitter to trigger a welcome email on signup.
- Implement Redis Pub/Sub for event broadcasting.

---

## Day 11 – Pagination + Filtering

**Why Learn:**
Handle large datasets with efficient pagination and filtering.

**Tasks:**

- Implement API pagination for posts.
- Add filtering and sorting options.
- Cache responses with Redis.

---

## Day 12 – Project Structure + Git

**Why Learn:**
Maintain clean architecture and collaborate with Git workflows.

**Tasks:**

- Refactor project into MVC structure.
- Learn Git branching & PR workflow.

---

### Day 12.5 – Clean Code Practices

**Why Learn:**
Readable and maintainable code is key to long-term success.

**Tasks:**

- Refactor duplicate code into reusable modules.
- Apply ESLint rules and format code.
- Conduct peer code reviews.

---

### Day 13 – Documentation + Testing

**Why Learn:**
Testing ensures reliability, and docs help developers use APIs.

**Tasks:**

- Generate Swagger docs for your APIs.
- Write unit tests with Jest for critical routes.

---

### Day 14 – Deployment + Capstone Project

**Why Learn:**
Deploy projects to the cloud and apply all learned skills in a real-world scenario.

**Tasks:**

- Deploy app to Render/Heroku/AWS.
- Capstone: Build a **Blog/Events Platform** with auth, CRUD, file upload, notifications, and deployment.

## Backend Libraries

### 1 express

- **Purpose:** Minimal and fast web framework for Node.js.

**Install:**

```
npm install express
```

**Usage:**

```
import express from "express";
const app = express();
app.get("/", (_, res) => res.send("Hello"));
app.listen(3000);
```

- **Used for:** Building REST APIs.

---

## 2 dotenv

- **Purpose:** Loads environment variables from `.env` file.

**Install:**

```
npm install dotenv
```

**Usage:**

```
import dotenv from "dotenv";
dotenv.config();
console.log(process.env.PORT);
```

- **Used for:** Configuration management.

---

## 3 cors

- **Purpose:** Enables cross-origin requests.

**Install:**

```
npm install cors
```

**Usage:**

```
import cors from "cors";
```

```
app.use(cors());
```

- **Used for:** Frontend-backend API connections.

---

## 4 eslint & prettier

- **Purpose:** Code linting and formatting.

**Install:**

```
npm install eslint prettier --save-dev
```

- **Usage:**
  Add `.eslintrc.json` and `.prettierrc` files to configure.
- **Used for:** Clean, consistent code.

---

## 5 express-validator

- **Purpose:** Validate user inputs in routes.

**Install:**

```
npm install express-validator
```

**Usage:**

```
import { body, validationResult } from "express-validator";
app.post("/user", body("email").isEmail(), (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) return res.status(400).json(errors.array());
});
```

- **Used for:** Input sanitization.

---

## 6 joi

- **Purpose:** Object schema validation.

**Install:**

```
npm install joi
```

**Usage:**

```
import Joi from "joi";
const schema = Joi.object({ name: Joi.string().min(3).required() });
schema.validate({ name: "Pooja" });
```

- **Used for:** Validating API request bodies.

---

## 7 body-parser

- **Purpose:** Parse incoming request bodies.

**Install:**

```
npm install body-parser
```

**Usage:**

```
import bodyParser from "body-parser";
app.use(bodyParser.json());
```

- **Used for:** Accessing `req.body`.

---

## 8 multer

- **Purpose:** Handle file uploads.

**Install:**

```
npm install multer
```

**Usage:**

```
import multer from "multer";
const upload = multer({ dest: "uploads/" });
app.post("/upload", upload.single("file"), (req, res) =>
res.send("Uploaded"));
```

- **Used for:** Handling file uploads.

---

## 9 bcrypt / bcryptjs

- **Purpose:** Hash passwords securely.

**Install:**

```
npm install bcrypt
# or
npm install bcryptjs
```

**Usage:**

```
import bcrypt from "bcryptjs";
const hash = await bcrypt.hash("password", 10);
```

- **Used for:** Secure authentication.

---

## 10 jsonwebtoken

- **Purpose:** Create and verify JWT tokens.

**Install:**

```
npm install jsonwebtoken
```

**Usage:**

```
import jwt from "jsonwebtoken";
const token = jwt.sign({ userId: 1 }, "secret");
```

- **Used for:** Auth middleware and session control.

---

## 11 lodash

- **Purpose:** Utility library for arrays, objects, functions.

**Install:**

```
npm install lodash
```

**Usage:**

```
 import _ from "lodash";
console.log(_.capitalize("hello"));
```

- **Used for:** Data manipulation.

---

## 12 mongoose

- **Purpose:** ODM for MongoDB (schema + model-based).

**Install:**

```
 npm install mongoose
```

**Usage:**

```
 import mongoose from "mongoose";
mongoose.connect("mongodb://localhost:27017/test");
```

- **Used for:** MongoDB data modeling.

---

## 13 sequelize

- **Purpose:** ORM for MySQL, Postgres, SQLite.

**Install:**

```
 npm install sequelize mysql2
```

**Usage:**

```
 import { Sequelize } from "sequelize";
const sequelize = new Sequelize("db", "user", "pass", { dialect:
"mysql" });
```

- **Used for:** SQL database abstraction.

---

## 14 prisma

- **Purpose:** Modern ORM for SQL/NoSQL with schema generation.

**Install:**

```
npm install prisma --save-dev
npx prisma init
```

- **Usage:**
  Define models in `schema.prisma`, then run `npx prisma generate`.
- **Used for:** Clean DB management.

---

## 15 jest

- **Purpose:** Testing framework for JS/Node.

**Install:**

```
npm install jest --save-dev
```

**Usage:**

```
test("sum", () => expect(1 + 2).toBe(3));
```

- **Used for:** Unit and integration testing.

---

## 16 nodemailer

- **Purpose:** Send emails easily from Node.js.

**Install:**

```
npm install nodemailer
```

**Usage:**

```
import nodemailer from "nodemailer";
const transporter = nodemailer.createTransport({ service: "gmail",
auth: { user: "you@gmail.com", pass: "pass" } });
transporter.sendMail({ to: "test@gmail.com", subject: "Hello", text:
"Hi!" });
```

- **Used for:** Email verification, password resets.