# ■ Node.js + Express.js Backend Roadmap 2025

Prepared by ChatGPT

August 13, 2025

Zero se deployment tak ek clear, step-by-step booklet jo aapko Node.js aur Express.js par professional REST APIs banane tak le jaye.

# Table of Contents

# 1. Introduction

Node.js ek JavaScript runtime hai jo browser ke bahar JavaScript run karne deta hai. Express.js ek lightweight framework hai jo Node.js ke upar fast aur maintainable backend APIs banana aasan banata hai. Is booklet mein hum step-by-step basics se production deployment tak chalenge.

**Prerequisites**: Basic JavaScript syntax (variables, functions, loops), terminal ki thodi practice.

# 2. Phase 1 – Foundation

## Goals

- JavaScript basics: Variables, Functions, Loops, Arrays, Objects
- Internet basics: HTTP, Request/Response, Server–Client model
- Install Node.js & npm (LTS recommended)
- Terminal basics (cd, ls/dir, mkdir, rm, npm commands)

## Quick Start

```
# Verify installations
node -v
npm -v

# Create a project folder
mkdir backend-roadmap && cd backend-roadmap

# Initialize a project (optional for later phases)
npm init -y
```

## Mini Project: Hello World Server (HTTP module)

### Steps

- project folder me hello.js file banayen
- neeche diya gaya code paste karen
- `node hello.js` run karen
- browser me http://localhost:3000 open karen

### Acceptance Criteria

- Browser me 'Hello, Backend World!' text nazar aaye
- Terminal me 'Server running on port 3000' log aaye

```
const http = require('http');
const server = http.createServer((req, res) => {
    res.end('Hello, Backend World!');
});
server.listen(3000, () => console.log('Server running on port 3000'));
```

# 3. Phase 2 – Node.js Core

## Focus Areas

- File system (fs) – files read/write
- Path (path) – paths resolve/join
- HTTP module basics
- npm packages install/use

## Mini Project: File Writer & Reader App

### Steps

- `app.js` create karen
- fs.writeFileSync se 'data.txt' me text likhen
- fs.readFileSync se content wapas read karke console me print karen

### Acceptance Criteria

- 'data.txt' file create ho
- Console me 'Hello Node.js' print ho

```
const fs = require('fs');
fs.writeFileSync('data.txt', 'Hello Node.js');
console.log(fs.readFileSync('data.txt', 'utf-8'));
```

# 4. Phase 3 – Express.js Basics

## Install & Setup

```
npm install express
# index.js
const express = require('express');
const app = express();
app.use(express.json());

app.get('/', (req, res) => res.send('Hello Express!'));
app.post('/data', (req, res) => res.json({ received: req.body }));

app.listen(3000, () => console.log('Server running on port 3000'));
```

## Concepts to Practice

- Basic routes: app.get(), app.post(), app.put(), app.delete()
- Middleware: app.use(), custom middleware (request time logger)
- Request & Response objects (req.params, req.query, req.body)

# 5. Phase 4 – REST API Mastery

## Core Skills

- CRUD operations: Create, Read, Update, Delete
- Route params (/:id) and query params (?page=1&limit;=10)
- Validation & error handling patterns
- Consistent response shapes and status codes

## Mini Project: To■Do API (In■Memory)

**Steps**

- Routes: POST /todos, GET /todos, GET /todos/:id, PUT /todos/:id, DELETE /todos/:id
- In■memory array me tasks store karen
- Validation: title required; completed default false
- Errors: 404 for missing id, 400 for bad input

**Acceptance Criteria**

- Saare endpoints Postman/curl se kaam karein
- Proper status codes (201 create, 200 ok, 404 not found, 400 bad request)

# 6. Phase 5 – Database Integration

## MongoDB + Mongoose

- MongoDB install ya cloud (connection string)
- Mongoose se connect
- Schema & Model define
- CRUD through Mongoose methods

```
npm install mongoose dotenv

// db.js
const mongoose = require('mongoose');
require('dotenv').config();
const MONGO_URI = process.env.MONGO_URI;

async function connectDB() {
  await mongoose.connect(MONGO_URI);
  console.log('MongoDB connected');
}
module.exports = connectDB;

// user.model.js
const { Schema, model } = require('mongoose');
const userSchema = new Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  age: { type: Number, min: 0 }
}, { timestamps: true });
module.exports = model('User', userSchema);
// server.js (snippet)
```

```
const express = require('express');
const connectDB = require('./db');
const User = require('./user.model');
const app = express();
app.use(express.json());
connectDB();

app.post('/users', async (req,res)=>{ const u = await User.create(req.body); res.status(201)
app.get('/users', async (_req,res)=>{ const u = await User.find(); res.json(u); });

app.listen(3000, ()=> console.log('API on 3000'));
```

## Mini Project: User Management API

### Steps

- User model (name, email, age) bnaen
- POST /users, GET /users, GET /users/:id, PATCH /users/:id, DELETE /users/:id
- Email unique + validation add karen

### Acceptance Criteria

- Duplicate email par 409/400 error mile
- CRUD operations DB me persist hon

# 7. Phase 6 – Authentication & Security

## Essentials

- Password hashing (bcrypt)
- JWT authentication (access token)
- Environment variables (dotenv)
- Security middleware (CORS, Helmet)

```
npm install bcrypt jsonwebtoken cors helmet

// auth.routes.js (snippet)
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const User = require('./user.model');
const router = express.Router();
const JWT_SECRET = process.env.JWT_SECRET;

router.post('/register', async (req, res) => {
  const { name, email, password } = req.body;
  const hash = await bcrypt.hash(password, 10);
  const user = await User.create({ name, email, password: hash });
  res.status(201).json({ id: user._id, email: user.email });
});

router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(401).json({ message: 'Invalid credentials' });
  const ok = await bcrypt.compare(password, user.password);
  if (!ok) return res.status(401).json({ message: 'Invalid credentials' });
  const token = jwt.sign({ sub: user._id }, JWT_SECRET, { expiresIn: '1h' });
  res.json({ token });
});

module.exports = router;

// server.js (security setup snippet)
const cors = require('cors');
const helmet = require('helmet');
app.use(cors());
app.use(helmet());
```

### Mini Project: Login/Register API with JWT

**Steps**

- Register + Login routes add karen
- Protected route /me jo sirf valid JWT par access ho
- Passwords hash store hon (plaintext nahi)

**Acceptance Criteria**

- Wrong password par 401
- Bearer token se /me accessible

# 8. Phase 7 – Advanced Features

- File uploads (multer)
- Emails (nodemailer)
- Request validation (Joi ya express-validator)
- Pagination & filtering (query params)

```
npm install multer nodemailer express-validator

// uploads (snippet)
const multer = require('multer');
const upload = multer({ dest: 'uploads/' });
app.post('/upload', upload.single('image'), (req, res) => {
  res.json({ filename: req.file.filename, size: req.file.size });
});

// validation (snippet)
const { body, validationResult } = require('express-validator');
app.post('/posts',
  body('title').isLength({ min: 3 }),
  (req,res)=>{
    const errors = validationResult(req);
    if(!errors.isEmpty()) return res.status(400).json({ errors: errors.array() });
    res.status(201).json({ ok: true });
  }
);
```

## Mini Project: Image Upload API

### Steps

- POST /upload endpoint (multer)
- Uploaded filename/URL response me return karein
- Accept only image/* MIME types

### Acceptance Criteria

- Non-image par 400
- Image save ho aur info return ho

# 9. Phase 8 – Deployment

- Git init, .gitignore (node_modules, .env)
- Environment variables for production
- Deploy (Render / Railway / Vercel)
- Health check endpoint & logs

```
# Git basics
git init
echo "node_modules
.env" >> .gitignore
git add .
git commit -m "Initial API"

# Common environment
PORT=3000
MONGO_URI=...
JWT_SECRET=...

# Start script in package.json
# "scripts": { "start": "node server.js" }
```

## Mini Project: Deploy To■Do API

### Steps

- Repo push karein (GitHub)
- Platform pe project connect karen
- Env vars set karen (.env ke values)

### Acceptance Criteria

- Public URL se endpoints reachable
- Logs me 'Server running' aur requests dikhain

# 10. Practice Challenges

- Notes API (tags + search)
- Blog API with authentication (roles: admin/user)
- E■commerce backend (products, carts, orders)

Tip: Pehle in■memory prototype banayen, phir database integrate karen, aur akhir me auth + deployment add karen.