Creating APIs in Node JS with Vibe Coding

Course Duration: 40 Hours **Course code: C-APIs-N-JS**

1. Course Overview

This course provides a comprehensive guide to building APIs in Node.js using the Vibe Coding methodology, which emphasizes clarity, modular design, scalability, and clean architecture. Across multiple hands-on modules, learners will explore the fundamentals of RESTful APIs, dive deep into Express.js for handling routes and middleware, integrate databases, secure APIs with authentication and authorization, and learn to deploy production-ready applications. By the end of the training, participants will have the knowledge and skills to create robust, scalable, and secure APIs that follow industry best practices.

2. What you'll learn?

By completing this course, you will be able to:

- Describe RESTful API principles and Node.js fundamentals.
- Set up a Node.js environment and structure an API project using Vibe Coding.
- Build routes, controllers, and middleware with Express.js.
- Integrate MongoDB and SQL databases for data persistence.
- Implement authentication and authorization with JWT and OAuth2.
- Apply best practices in error handling, logging, and validation.
- Document APIs using Swagger / OpenAPI.
- Test APIs with Postman and automated frameworks.
- Deploy APIs to the cloud with monitoring and scaling strategies.

3. Target Audience

- Beginner and intermediate developers aiming to specialize in backend development with Node.js.

- Frontend developers seeking to become full-stack engineers.
- Professionals looking for a structured way to build APIs with Vibe Coding practices.

4. Pre-Requisites

Familiarity with:

- JavaScript ES6+ fundamentals
- Basic knowledge of HTTP and REST APIs
- Core concepts of SQL/NoSQL databases

5. Course content

Day 1: RESTful API Foundations with Node.js

1.1 Introduction to APIs

- What is an API?
- REST vs GraphQL
- API request/response flow
- Status codes, headers, and content types

1.2 Setting Up Node.js Project

- Initializing package.json
- Installing Express, nodemon, cors
- Project folder structure: routes/, controllers/, services/

1.3 Creating First REST API

- Basic routes: GET, POST, PUT, DELETE
- Parsing JSON payloads
- Using Postman to test endpoints

1.4 Error Handling and Middleware

- Centralized error handler

- Custom error messages
- Global middleware

Lab 1:

- Build a simple task-manager API with:
- Create, Read, Delete tasks
- Use middleware for logging and error handling

Day 2: API Security, Modularization, and Environment Setup

2.1 Environment Configuration

- Storing API keys securely with. env
- Using dotenv in Node.js

2.2 API Structuring Best Practices

- Creating modular routes and controllers
- Reusable services
- Organizing validation and constants

2.3 CORS, Rate Limiting & Security Basics

- Enabling CORS for frontend apps
- Adding basic security headers
- Rate limiting with express-rate-limit

2.4 Using Git & GitHub for Version Control

- Creating a GitHub repo
- Pushing project code
- Managing .env in .gitignore

Lab 2:

- Refactor the task-manager API:

- Move logic to controllers/ and services/
- Add .env, rate limiting, and GitHub versioning

Day 3: OpenAI API Integration (ChatGPT, DALL·E)

3.1 Introduction to OpenAI APIs

- Overview: GPT, DALL·E, Whisper
- OpenAI API documentation walkthrough
- Prompt engineering fundamentals

3.2 ChatGPT Integration (gpt-3.5-turbo)

- Install axios for HTTP calls
- Secure API key with .env
- Create /chat endpoint that accepts prompt and returns a GPT response

3.3 Image Generation with DALL·E

- Understanding DALL·E parameters (prompt, size, format)
- Creating /generate-image endpoint

3.4 Prompt Engineering Practice

- Writing system vs user prompts
- Controlling tone, format, creativity

Lab 3:

- Create a content-generator API with:
- /blog-summary – summarizes blog posts
- /image-cover – generates an image cover using DALL·E

Day 4: Google Cloud AI & Whisper API Integration

4.1 Using Google Cloud Vision API

- Enabling APIs in Google Cloud

- Creating service account and API key
- Analyzing:
- Text (OCR)
- Labels and objects

4.2 Using Google Natural Language API

- Analyzing sentiment, entities, syntax
- Creating /analyze-text endpoint

4.3 Uploading Files with Multer

- Handling audio/image uploads
- Saving to disk or using memory buffer

4.4 Whisper API for Speech-to-Text

- Upload .mp3 or .wav files
- Send to OpenAI Whisper API
- Create /transcribe-audio endpoint

Lab 4:

- Build a "Media Intelligence API":
- /analyze-image: returns labels/text from image
- /transcribe-audio: returns text from uploaded audio
- /summarize-transcript: sends transcribed text to ChatGPT for summary

Day 5: Final Project, Testing, and Deployment

- 5.1 Combining AI APIs in Workflow
- AI workflow: audio → text → analysis → summary
- Chaining multiple AI APIs in one request

- Create /smart-report:
  - Accepts audio
  - Transcribes → summarizes → returns JSON summary

5.2 Input Validation & Robust Error Handling

- Using express-validator or custom checks
- Handling timeouts, malformed inputs, API quota errors

5.3 API Documentation with Postman

- Creating Postman collections
- Documenting request/response schemas
- Exporting and sharing collections

5.4 Deployment

- Deploy to Render or Vercel
- Securing deployed API

Final Lab 5:

- Build a “Smart AI Assistant API”:
- Features:
- /chat: ChatGPT response
- /transcribe: Whisper audio transcription
- /analyze: Vision + NLP
- /smart-assist: Uploads audio/image, performs AI operations, and summarizes results