Course Description and Outline

Intermediate Development with Node.js

Course Overview

This intensive 5-day course builds on the fundamentals of Node.js to help developers deepen their skills in server-side JavaScript. Students will explore advanced asynchronous programming techniques, performance optimization, security best practices, comprehensive testing strategies, and scalable application design.

Prerequisites

- Solid understanding of JavaScript fundamentals
- Basic experience with Node.js and npm
- Familiarity with REST APIs and HTTP concepts
- Basic command line proficiency
- Understanding of Git version control

Course Objectives

By the end of this course, students will be able to:

1. **Master Advanced Asynchronous Programming**

- Implement async iterators and generators

- Utilize worker threads for CPU-intensive tasks

- Design efficient queue-based processing systems

2. **Build Production-Ready APIs**

- Create secure REST APIs with proper authentication

- Implement comprehensive error handling strategies

- Apply security best practices and middleware

3. **Work with Advanced Database Patterns**

- Implement MongoDB transactions and aggregations

- Design efficient caching strategies with Redis

- Optimize database queries and connections

4. **Create Real-Time Applications**

- Build WebSocket-based chat applications

- Implement real-time data streaming

- Handle concurrent connections efficiently

5. **Optimize Application Performance**

- Conduct performance analysis and profiling

- Implement caching strategies at multiple levels

- Use clustering and load balancing techniques

6. **Implement Comprehensive Testing**

- Write unit, integration, and end-to-end tests

- Implement test automation and CI/CD pipelines

- Use advanced testing patterns and mocking

7. **Deploy and Monitor Applications**

- Containerize Node.js applications with Docker

- Implement logging, monitoring, and health checks

- Deploy to cloud platforms with proper DevOps practices

Daily Schedule

Day 1: Advanced Asynchronous Programming & Worker Threads

- Event loop deep dive and performance considerations
- Async iterators and generators
- Worker threads for CPU-intensive tasks
- **Labs**: Async Iterators, Worker Threads

Day 2: Authentication, Security & Database Transactions

- JWT authentication implementation
- Security best practices and middleware
- MongoDB transactions and advanced queries
- **Labs**: JWT Auth System, MongoDB Transactions

Day 3: Caching, Redis & Real-Time Communication

- Redis caching strategies and patterns
- Advanced Redis features (pub/sub, streams)
- WebSocket implementation and real-time chat
- **Labs**: Redis Caching, Advanced Redis, Real-Time Chat

Day 4: Performance Analysis & Comprehensive Testing

- Performance profiling and optimization
- Load testing and bottleneck identification
- Comprehensive testing strategies (unit, integration, E2E)
- **Labs**: Performance Analysis, Testing Suite Implementation

Day 5: Docker Deployment & Production Monitoring

- Docker containerization best practices
- Production deployment strategies
- Logging, monitoring, and health check implementation
- **Labs**: Docker Deployment, Logging & Monitoring

Assessment Methods

- Hands-on lab exercises (70%)
- Code review and best practices implementation (20%)
- Final project demonstrating course concepts (10%)

Course Materials

- Complete slide deck with all lecture content
- 11 hands-on lab exercises with detailed instructions
- Complete working solutions for all labs
- Additional resources and documentation links
- Setup and environment configuration guide

Technical Requirements

- Node.js 18+ installed
- Code editor (VS Code recommended)
- Docker Desktop
- MongoDB (local or cloud instance)
- Redis server
- Git for version control
- Terminal/command line access

Target Audience

This course is designed for:

- JavaScript developers with basic Node.js experience
- Backend developers looking to deepen Node.js skills
- Full-stack developers focusing on server-side development
- Software engineers preparing for senior-level positions
- Development teams adopting Node.js for enterprise applications