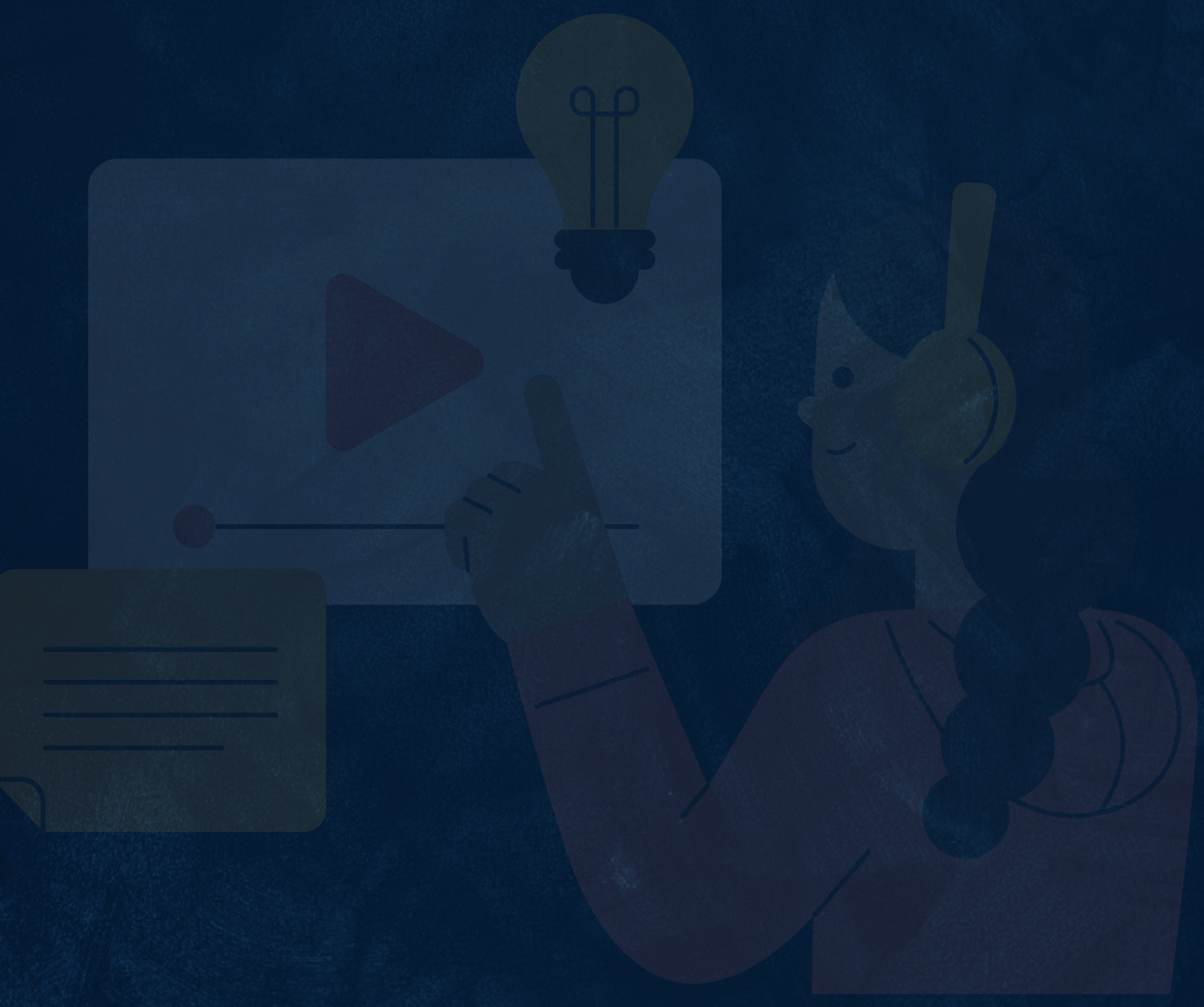
Paperlive Learning | Microsoft

Full Stack Development program in collaboration with Microsoft



Paperlive Learning

www.paperlive.in

Module 1 : Web Development Fundamentals

HTML5
- o Structure of a webpage
- o Semantic HTML tags
- o Forms and inputs
- o Media embedding (audio, video, iframe)

CSS3
- o CSS selectors & properties
- o Box model & positioning
- o Flexbox & Grid layout
- o Animations & transitions
- o Responsive design (media queries)

JavaScript (Core)
- o Variables (var, let, const)
- o Data types & operators
- o Functions & scope
- o DOM manipulation (querySelector, events)
- o ES6+ features (arrow functions, destructuring, template literals)
- o Promises, async/await
- o Error handling

Module 2: Version Control & Collaboration

Git & GitHub
- o Installing and configuring Git
- o Git workflow (clone, branch, commit, merge, push, pull, rebase)o Resolving conflicts
- o Pull requests & code reviews
- o GitHub Actions (basics of CI/CD pipelines)

Module 3: Version Control & Collaboration

React.js
- o Components (functional & class)
- o Props & State
- o Hooks (useState, useEffect, useContext, useReducer)
- o React Router (SPA navigation)
- o Forms and validation
- o Context API

State Management
- o Redux (actions, reducers, store)
- o Redux Toolkit

UI Libraries
- o Material-UI / Tailwind CSS / Bootstrap

Testing
- o Jest & React Testing Library

Module 4: Continuous Integration Using Jenkins

Node.js
- o Event loop
- o Modules & npm
- o File system, streams

Express.js
- o Setting up a server
- o Middleware
- o Routing
- o RESTful APIs
- o Error handling

Authentication & Securit
- o JWT authentication
- o OAuth2
- o Password hashing (bcrypt)
- o Helmet.js & CORS

Module 5: Databases

SQL (Relational Databases)
- o PostgreSQL/MySQL basics
- o CRUD operations
- o Joins, indexing, stored procedures

NoSQL (MongoDB)
- o Documents & collections
- o CRUD operations
- o Aggregation framework
- o Indexing

ORMs
- o Sequelize (SQL)
- o Mongoose (MongoDB)

Module 6: DevOps & Cloud Fundamentals

Linux Basics
- o Commands, permissions, process management

CI/CD
- o GitHub Actions / Jenkins pipelines

Containerization
- o Docker basics
- o Docker Compose

Cloud
- o AWS (EC2, S3, RDS basics)
- o GCP/Azure overviewo Deploying full-stack applications

Monitoring
- o Logging basics
- o Intro to Grafana & Prometheus

Module 7: System Design & Architecture

Design Principles
- o SOLID principles
- o Design patterns

System Architecture
- o Monolithic vs Microservices
- o Load balancing, caching
- o REST vs GraphQL

Scalability
- o Database scaling (sharding, replication)
- o CDN
- o Queue systems (RabbitMQ/Kafka)

Module 8: Generative AI for Full-Stack

Introduction to AI & LLMs
- o What is Generative AI?
- o Large Language Models (LLMs)

Prompt Engineering
- o Writing effective prompts
- o Zero-shot, few-shot prompting

Integrating AI into Applications
- o Using OpenAI API (GPT models)
- o Chatbot development with Node.js & React
- o AI-powered search & recommendations

Practical Projects
- o AI-powered resume buildero AI chatbot integrated into a web app
- o Sentiment analysis dashboard

Responsible AI
- o Bias & fairness in AI
- o Ethical considerations

Projects

1. Personal Portfolio Website

Tech Stack: React, TailwindCSS, React Router, Netlify/Vercel

Steps

1. Initialize project with npx create-react-app or vite.
2. Setup folder structure: components/, pages/, assets/.
3. Build sections → Home, About, Skills, Projects, Contact.
4. Use React Router for navigation between pages.
5. Add contact form with form validation using React Hook Form.
6. Optimize with lazy loading & SEO tags.
7. Deploy using Netlify/Vercel → connect to GitHub repo for CI/CD.

Best Practice: Add analytics (Google Analytics) + dark mode toggle.

2. Blogging Platform (Medium Clone)

Tech Stack: MERN (MongoDB, Express, React, Node), JWT Auth, Cloudinary

Steps

1. Setup User Model (username, email, password), Post Model (title, content, tags).
2. Implement JWT Auth → login, register, logout.
3. Build API routes: /posts, /comments, /users.
4. Frontend with React + Redux for state management.
5. Add rich text editor (Quill.js or Draft.js).
6. Deploy backend on Render/Heroku, frontend on Netlify.
7. Add file uploads (Cloudinary for images).

Best Practice: Add rate-limiting on API to prevent spam.

3. E-Commerce Store

Tech Stack: MERN + Stripe + AWS S3

Steps

1. Create schema → Product (name, price, stock), Order, User.
2. Product listing & filtering with pagination.
3. Shopping cart → add/remove/update items.
4. Payment integration (Stripe/PayPal sandbox).
5. Admin dashboard for product & order management.
6. Deploy backend on AWS EC2, images on S3.
7. Add caching with Redis for frequently viewed products.

Best Practice: Apply indexing in MongoDB for faster product search.

4. Chat Application

Tech Stack: Node.js, Express, Socket.io, MongoDB, Redis

Steps

1. Initialize Node app with Socket.io for real-time chat.
2. Create chat rooms (1-1 + group chat).
3. Store messages in MongoDB → schema: { sender, receiver, text, timestamp }.
4. Use Redis for temporary message caching.
5. Add typing indicator & online status.
6. Deploy on Heroku/AWS with HTTPS enabled.

Best Practice: Encrypt messages in DB, enable JWT auth for sockets.

5. Task Management Tool (Trello Clone)

Tech Stack: React, Node.js, PostgreSQL, Drag-and-drop (React DnD)

Steps

1. Design schema: Boards → Lists → Tasks (with foreign keys).
2. Implement CRUD operations for tasks.
3. Add drag-and-drop for moving tasks across lists.
4. Setup WebSockets for real-time task updates.
5. Add email/notification service with Nodemailer.
6. Deploy using Docker containers on AWS.

Best Practice: Use RabbitMQ/Kafka for scalable event-driven updates.

Projects

6. Resume Builder (AI-Powered)

Tech Stack: React, Node, MongoDB, OpenAI API, React-PDF

Steps

1. Create form for user details (education, skills, experience).
2. Store details in MongoDB.
3. Use React-PDF to generate downloadable resumes.
4. Integrate GPT API → auto-suggest role-based skills & achievements.
5. Add ATS score checker (keywords vs job description).

Best Practice: Allow export in multiple formats (PDF, DOCX).

7. Job Portal

Tech Stack: MERN + ElasticSearch + AI Matching

Steps

1. Roles: Candidate, Recruiter, Admin.
2. Candidate → profile, resume upload.
3. Recruiter → post jobs, shortlist candidates.
4. Admin → approve/reject job posts.
5. AI → Compare resumes & job descriptions (TF-IDF/NLP similarity).
6. Use ElasticSearch for fast searching/filtering.

Best Practice: Add email triggers when job is posted/shortlisted.

8. Social Media App (Instagram Lite)

Tech Stack: MERN + Cloudinary + Redis

Steps

1. Implement authentication (JWT + OAuth via Google).
2. Upload photos/videos (Cloudinary/S3).
3. Likes, comments, follow/unfollow system.
4. Real-time notifications using WebSockets.
5. Feed algorithm → sort posts by timestamp & engagement.

Best Practice: Add AI moderation (flagging NSFW content).

9. SaaS Expense Tracker

Tech Stack: React, Node.js, MongoDB, Stripe, Recharts

Steps

1. User login & dashboard.
2. Add daily expenses → categories (food, travel, rent).
3. Generate graphs (Recharts).
4. Export CSV/PDF reports.
5. SaaS billing with Stripe (free & premium tiers).
6. Deploy on AWS Lambda + S3.

Best Practice: AI forecasting of monthly spend trends.

10. AI Knowledge Assistant

Tech Stack: LangChain, Pinecone, OpenAI API, React

Steps

- Upload PDFs using Multer → store in MongoDB GridFS.
- Generate embeddings using OpenAI & store in Pinecone.
- Build search API → fetch relevant chunks.
- UI: User asks Q → API retrieves + GPT generates answer.
- Deploy backend as serverless functions (AWS Lambda).

Best Practice: Add role-based access (free vs premium users).

CASE STUDY

1. Redesign a Legacy Website

- Steps: Audit old site → Migrate to React SPA → Optimize with code splitting.
- Solution: Reduced page load from 5s → 1.2s.

2. Scalability of Blogging Platform

- Steps: Load testing (JMeter) → DB sharding → Redis caching.
- Solution: 70% performance improvement under 100k users.

3. E-Commerce Security

- Steps: Run OWASP ZAP → Identify SQL injection/XSS → Fix with sanitization.
- Solution: Added JWT + parameterized queries → Security Grade A

4. Cloud Deployment Strategy

- Steps: Deploy app on AWS/GCP/Azure → Compare cost/performance.
- Solution: AWS cheapest compute, GCP strong ML, Azure enterprise-ready.

5. AI Chatbot for Customer Support

- Steps: Integrate GPT → Measure response time → Compare vs human.
- Solution: AI reduced time by 80% but needed escalation pipeline.

6. Resume Shortlisting Automation

- Steps: Build NLP model → Compare recruiter vs AI shortlists.
- Solution: AI matched 87% recruiter decisions.

7. Multi-Tenant SaaS System

- Steps: Tenant DB design → Deploy with schema-per-tenant → Test scaling.
- Solution: Scaling bottleneck solved with DB sharding.

8. Database Optimization in Social Media

- Steps: Profile slow queries → Add indexes → Restructure schema.
- Solution: Query response improved 10x.

9. Microservices vs Monolithic in Job Portals

- Steps: Build prototypes both ways → Compare deployment + cost.
- Solution: Monolith cheaper <10k users, microservices necessary >100k.

10. Responsible AI in Applications

- Steps: Identify risks (bias in resume filtering) → Add fairness testing.
- Solution: Added transparency reports + bias detection pipeline.