React Full Stack Developer with PostgreSQL

Duration: 10 days

Day 1: Modern JavaScript and React Fundamentals

Topics

- Full-stack application architecture
- Introduction to React and component-based development
- Project creation using Vite
- JSX syntax and expressions
- Functional components
- Props and component composition
- JavaScript concepts required for React:
    - Arrow functions
    - Destructuring
    - Spread and rest operators
    - Array methods
    - Modules
    - Promises and async/await

Hands-on Labs

- Create and configure a React application.
- Build reusable header, footer and content components.
- Create a product or employee card component using props.
- Render multiple records using map().
- Build a small dashboard interface using reusable components.

Capstone Progress

- Finalize the capstone problem statement.
- Identify major application modules.
- Create the React project structure.
- Design the initial application layout.

Outcome

Participants will be able to create React applications using reusable functional components.

Day 2: React State, Events, Forms and Hooks

Topics

- React state and the useState hook
- Event handling
- Conditional rendering
- Rendering lists and using keys

- Controlled components
- Form handling
- Basic form validation
- Introduction to the useEffect hook
- Component lifecycle behaviour using hooks
- Lifting state between components

Hands-on Labs

- Build an interactive task-management interface.
- Create add, edit and delete functionality using local state.
- Develop a registration form with validation.
- Display validation messages conditionally.
- Filter records using a search input.

Capstone Progress

- Create the main data-entry form.
- Build list and details components.
- Implement temporary CRUD operations using React state.

Outcome

Participants will be able to develop interactive React forms and manage component state.

Day 3: React Routing, API Integration and Application Structure

Topics

- Single-page application concepts
- React Router configuration
- Route parameters
- Nested routes
- Navigation and redirects
- Creating page and layout components
- Calling REST APIs using Fetch or Axios
- Loading, success and error states
- Environment variables
- Organizing components, pages, services and utilities

Hands-on Labs

- Create a multi-page React application.
- Configure home, login, listing, details and dashboard routes.
- Consume a public REST API.
- Create reusable API service functions.
- Implement loading indicators and error messages.
- Build search and filtering functionality.

Capstone Progress

- Configure capstone application routes.
- Create dashboard, listing, details and form pages.
- Prepare the frontend API service layer.

Outcome

Participants will be able to develop structured React applications and integrate external APIs.

Day 4: Node.js and Express.js Backend Development

Topics

- Role of Node.js in full-stack development
- Node.js project structure
- Node Package Manager
- Express.js application setup
- Routes and controllers
- Request and response handling
- Route parameters and query parameters
- Middleware
- Creating RESTful APIs
- HTTP methods and status codes
- Centralized error handling
- API testing using Postman

Hands-on Labs

- Create an Express.js server.
- Develop GET, POST, PUT and DELETE endpoints.
- Organize routes, controllers and middleware.
- Validate API requests.
- Test APIs using Postman.
- Implement centralized error responses.

Capstone Progress

- Create the capstone backend project.
- Develop the first set of REST API endpoints.
- Connect the React frontend to temporary backend data.

Outcome

Participants will be able to create and test REST APIs using Node.js and Express.js.

Day 5: PostgreSQL and Relational Database Design

Topics

Database Comparison

- Relational databases versus NoSQL databases
- PostgreSQL versus MySQL
- PostgreSQL versus MongoDB
- Tables and relations compared with collections and documents
- When to select PostgreSQL
- Benefits of PostgreSQL:
    - Data consistency
    - Relationships
    - Transactions
    - Constraints
    - Advanced SQL capabilities

PostgreSQL Fundamentals

- PostgreSQL installation and tools
- Creating databases and schemas
- Tables and data types
- Primary keys and foreign keys
- Constraints
- One-to-one relationships
- One-to-many relationships
- Many-to-many relationships
- Data normalization
- INSERT, SELECT, UPDATE and DELETE
- Filtering, sorting and grouping
- SQL joins

Hands-on Labs

- Create a PostgreSQL database.
- Design related tables.
- Apply primary-key and foreign-key constraints.
- Write CRUD SQL queries.
- Retrieve data using joins.
- Create reports using grouping and aggregate functions.

Capstone Progress

- Prepare the capstone entity-relationship design.
- Create the capstone PostgreSQL database.
- Define tables, relationships and constraints.
- Insert initial sample data.

Outcome

Participants will be able to design and query relational databases using PostgreSQL.

Day 6: Connecting Node.js with PostgreSQL

Topics

- PostgreSQL connectivity from Node.js
- Connection pooling
- Environment-based database configuration
- Executing parameterized SQL queries
- Preventing SQL injection
- Repository or data-access-layer pattern
- Performing CRUD operations through REST APIs
- Database error handling
- Transactions
- Pagination, search and sorting
- Database migrations and seed data
- Introduction to ORM and query-builder options

Hands-on Labs

- Connect an Express application with PostgreSQL.
- Create reusable database connection modules.
- Develop database-backed CRUD APIs.
- Implement search and pagination.
- Execute multi-step operations using transactions.
- Handle duplicate and foreign-key errors.

Capstone Progress

- Connect the capstone backend with PostgreSQL.
- Replace temporary data with database queries.
- Complete backend CRUD operations.
- Implement search, sorting and pagination.

Outcome

Participants will be able to build database-driven REST APIs using Express.js and PostgreSQL.

Day 7: Authentication, Authorization and Application Security

Topics

- User-registration workflow
- Login workflow
- Password hashing
- JSON Web Token authentication
- Authentication middleware
- Role-based authorization
- Protected backend routes
- Protected React routes
- Storing and managing authentication state
- Input validation and sanitization

- CORS configuration
- Secure environment variables
- Common web application security considerations

Hands-on Labs

- Create user-registration and login APIs.
- Hash passwords before database storage.
- Generate and verify authentication tokens.
- Protect API endpoints using middleware.
- Implement role-based access.
- Build login and logout functionality in React.
- Create protected frontend routes.

Capstone Progress

- Add user registration and login.
- Create user roles such as administrator and standard user.
- Protect sensitive pages and API endpoints.
- Display role-specific navigation and actions.

Outcome

Participants will be able to implement secure authentication and role-based authorization.

Day 8: Testing and Test Case Development

First Half: Unit and Integration Testing

Topics

- Purpose of software testing
- Testing pyramid
- Unit testing versus integration testing
- Introduction to Jest or Vitest
- Testing React components
- React Testing Library
- Testing rendering and user interactions
- Testing hooks and forms
- Mocking API calls
- Testing Express.js APIs
- Introduction to Supertest
- Database-related test considerations

Hands-on Labs

- Write unit tests for JavaScript utility functions.
- Test a React form component.
- Test button clicks and validation messages.
- Mock an API response.

- Write API tests for Express routes.
- Test successful and failed requests.

Second Half: Practical Test Cases and Application Quality

Topics

- Writing effective test scenarios
- Positive and negative test cases
- Boundary-value test cases
- Form-validation test cases
- Authentication and authorization test cases
- API test cases
- Database test cases
- Defect reporting
- Test execution and result documentation

Hands-on Labs

- Prepare a test-case document for the capstone project.
- Create test cases with:
    - Test case ID
    - Module
    - Preconditions
    - Test steps
    - Test data
    - Expected result
    - Actual result
    - Status
- Execute test cases against the capstone application.
- Record and fix identified defects.

Capstone Progress

- Add frontend and backend unit tests.
- Prepare a practical test-case document.
- Execute critical application test scenarios.
- Fix defects identified during testing.

Outcome

Participants will be able to write unit tests, API tests and structured practical test cases.

Day 9: Full-Stack Integration and Capstone Development

Topics

- End-to-end frontend and backend integration
- Managing API configuration
- Reusable API service modules
- Form submission and server-side validation

- Handling backend error responses
- Loading and empty states
- Pagination and filtering
- Reusable UI components
- Dashboard statistics
- Application debugging
- Code refactoring
- Production readiness checklist
- Basic deployment concepts

Hands-on Labs

- Integrate React pages with PostgreSQL-backed APIs.
- Implement complete CRUD workflows.
- Add server-side and client-side validation.
- Add search, sorting and pagination.
- Develop dashboard summary cards.
- Handle loading, error and empty states.
- Refactor repeated code into reusable modules.

Capstone Progress

- Complete all major capstone modules.
- Perform end-to-end integration.
- Improve user experience and error handling.
- Prepare the project for final testing and demonstration.

Outcome

Participants will be able to integrate and debug a complete React full-stack application.

Day 10: AI-Assisted Coding and Capstone Presentation

AI-Assisted Coding Topics

- Introduction to AI coding assistants
- Effective prompting for software development
- Using AI for:
    - React component generation
    - API endpoint generation
    - PostgreSQL query generation
    - Form-validation logic
    - Unit-test generation
    - Test-case generation
    - Code explanation
    - Debugging
    - Refactoring
    - Documentation
- Providing relevant project context to AI tools

- Reviewing AI-generated code
- Identifying incorrect or insecure suggestions
- Avoiding blind copy-and-paste development
- Privacy, security and intellectual-property considerations
- Developer accountability while using AI

AI-Assisted Coding Labs

- Generate a React component using an AI coding assistant.
- Generate a PostgreSQL query from a business requirement.
- Ask AI to create unit tests for an existing component.
- Diagnose and fix a backend defect using AI suggestions.
- Refactor duplicated code using AI assistance.
- Generate API and project documentation.
- Review AI-generated code for security and correctness.

Capstone Finalization

- Complete pending functionality.
- Execute final test cases.
- Fix critical defects.
- Improve code structure and documentation.
- Prepare the project demonstration.
- Present the final capstone project.
- Participate in technical review and feedback.

Outcome

Participants will be able to use AI coding assistants productively while maintaining code quality, security and developer ownership.
