Full Stack Development Syllabus

With Reach Node .js

Introduction to Full Stack Development

- ✓ **Overview of Full Stack Development**: What is Full Stack Development, roles of a full-stack developer, importance of front-end and back-end technologies, and the integration of both.
- ✓ **Tech Stack Overview**:
    - Front-End: React.js (UI components, routing, state management)
    - Back-End: Node.js (server-side logic, REST APIs, database interaction)
    - Database: MongoDB (NoSQL) or PostgreSQL/MySQL (SQL)
    - Version Control: Git, GitHub

Front-End Development with React.js

HTML5 & CSS3 Basics

- ✓ **HTML5**: Structure of a webpage, semantic tags, forms, and input elements
- ✓ **CSS3**: Styling techniques (layout, typography), Flexbox, Grid, Media Queries for responsive design
- ✓ **CSS Preprocessors**: Sass or less for better organization and maintainability

JavaScript (ES6+)

- ✓ **Fundamentals**: Variables, data types, operators, control structures
- ✓ **Functions and Objects**: Functions, arrow functions, ES6 classes, objects, and arrays
- ✓ **DOM Manipulation**: Selecting elements, event handling, updating the DOM
- ✓ **ES6 Features**: Arrow functions, promises, async/await, DE-structuring, template literals, spread/rest operators

React.js Essentials

- ✓ **Introduction to React**: What is React, component-based architecture
- ✓ **JSX**: Writing components using JSX, JSX syntax
- ✓ **Components and Props**: Creating functional and class components, passing data through props
- ✓ **State and Lifecycle**: Managing state in components, component lifecycle methods, and the useState and useEffect hooks
- ✓ **React Router**: Client-side routing for single-page applications (SPAs)
- ✓ **Event Handling**: Handling user inputs and actions in React components
- ✓ **Forms and Validation**: Handling form inputs, form validation in React

Advanced React Concepts

- ✓ **Context API**: Managing global state across components
- ✓ **React Hooks**: Custom hooks, useContext, useReducer, and performance optimization
- ✓ **State Management**: Introduction to Redux, connecting React to Redux for more complex state management
- ✓ **React Performance Optimization**: Memoization (React.memo, useMemo), lazy loading, and code splitting

Back-End Development with Node.js

Introduction to Node.js

- ✓ **What is Node.js?** Node.js fundamentals, non-blocking I/O, and asynchronous programming model
- ✓ **Setting up Node.js Environment**: Installing Node.js, npm, and basic project structure
- ✓ **Node.js Modules**: Using built-in modules like fs, http, path, url, etc.

Express.js

- ✓ **Introduction to Express**: Setting up an Express server, middleware, and routing
- ✓ **Routing**: Defining API routes, handling GET, POST, PUT, DELETE requests
- ✓ **Middleware**: Using middleware for error handling, authentication, logging, etc.
- ✓ **Template Engines**: Using EJS or Pug to render views dynamically

APIs & RESTful Services

- ✓ **REST API Basics**: REST principles, HTTP methods (GET, POST, PUT, DELETE), and status codes
- ✓ **Creating RESTful APIs with Express**: Setting up routes, controllers, and error handling
- ✓ **Authentication and Authorization**:
  - User authentication with JWT (JSON Web Tokens)
  - OAuth for third-party logins (Google, Facebook, etc.)
  - Role-based access control (RBAC)
- ✓ **CORS (Cross-Origin Resource Sharing)**: Handling cross-origin requests securely

Database Integration

Relational Databases (SQL)

- ✓ **Introduction to SQL**: Basics of SQL, SELECT, INSERT, UPDATE, DELETE queries
- ✓ **Database Design**: Normalization, relationships (one-to-many, many-to-many)
- ✓ **PostgreSQL/MySQL**: Installing and using PostgreSQL or MySQL with Node.js
- ✓ **ORM (Object-Relational Mapping)**: Using Sequelize or Knex.js to interact with databases via models

NoSQL Database (MongoDB)

- ✓ **Introduction to NoSQL**: Basics of NoSQL databases, document-oriented data
- ✓ **MongoDB**: Setting up MongoDB, collections, and documents
- ✓ **Mongoose**: Using Mongoose for schema definition and data validation in MongoDB
- ✓ **CRUD Operations in MongoDB**: Create, Read, Update, Delete operations with Mongoose

Authentication & Authorization

User Authentication

- ✓ **JWT (JSON Web Tokens)**: Securing API endpoints with JWT for user authentication
- ✓ **Session-Based Authentication**: Using sessions and cookies for managing login states
- ✓ **OAuth Authentication**: Implementing third-party login with Google, Facebook, or GitHub

Authorization

- ✓ **Role-Based Access Control (RBAC)**: Granting specific roles and permissions to users
- ✓ **Secure Password Storage**: Using bcrypt for hashing passwords and comparing them securely

Testing and Debugging

Unit Testing

- ✓ **Testing Frameworks**: Introduction to Mocha, Chai, and Jest for unit testing Node.js and React components
- ✓ **Testing APIs**: Writing tests for API endpoints, mock data, and error handling

Debugging

- ✓ **Debugging Node.js**: Using Node.js debugger and logging techniques
- ✓ **Debugging React**: React Developer Tools and browser debugging tools

Deployment

Deployment of React Application

- ✓ **Building for Production**: Using create-react-app to build a production-ready version of the React app
- ✓ **Deploying on Platforms**: Deploying React applications on platforms like Netlify, Vercel, or AWS

Deployment of Node.js Backend

- ✓ **Production Setup**: Setting up Node.js for production with tools like pm2, forever, or Docker
- ✓ **Server Deployment**: Deploying back-end Node.js applications on cloud providers like AWS, Heroku, or DigitalOcean
- ✓ **Environment Variables**: Managing sensitive information (API keys, database credentials) using environment variables