Web Development Training

Time Duration: 6 Months

Modules: 9 Modules

Lessons: 81 Lessons

Fees: 20000

Course Objective

The **Web Development Training Program** is designed to provide learners with a comprehensive, hands-on, and industry-ready foundation in full-stack web development. Over the span of **6 months**, students will master the core web technologies—**HTML, CSS, JavaScript, PHP, and React JS**—through structured lessons, practical projects, and real-world applications. This course includes **9 detailed modules and 81 sessions**, ensuring a gradual and in-depth understanding of both front-end and back-end development.

Participants will learn how to build responsive websites, manage databases, work with APIs, and implement modern frameworks like **React** and **Redux**. From simple form validation to building a fully functional e-commerce platform with an admin panel, this course focuses on skill-building through projects that simulate real industry demands.

The program also introduces essential development tools like **Git & GitHub**, API concepts, and even **Laravel**, enabling learners to transition smoothly into professional development environments. Whether you are a beginner or someone looking to solidify your skills, this course is tailored to turn your passion for web development into a valuable career path.

Upon completion, students will have built a strong portfolio, ready for internships, freelance opportunities, or junior developer roles, making this program an ideal choice for anyone aiming to enter the tech world with confidence

Mentors

1- Vivek Srivastava (Senior Software Engineer)

Work Experiences:	1) 10 Years of Experience in web development
	2) 3 Years of experience in Android Application Development
	3) 3 Years of experience in Apple Application Development
	4) Currently working in US Based company
Technologies:	1) PHP and it's Frameworks
	2) Java Script
	3) React JS
	4) React-Native
	5) RDBMS

2- Gaurav Mishra (Senior Full Stack Developer)

Work Experiences:	1) 10 Years of Experience in web development
	2) 5 Years of experience in Mobile Application Development
	3) Director of Stackvit LLP.
Technologies:	1) PHP and it's Frameworks
	2) React JS
	3) JAVA and It's Framework
	4) RDBMS
	5) Python
	6) AWS Services

2- Rajat K. (Full Stack Developer)

Work Experiences:	1) 2 Years of Experience in web development
	2) Currently working in Hyderabad as Software Engineer
Technologies:	1) PHP and it's Frameworks
	4) RDBMS
	6) AWS Services

✅ Module 1: Web Basics (Week 1)

🎯 Goal: Understand how the internet and web development works, and set up the required tools.

🧠 Lesson 1.1: What is Web Development?

- Difference between frontend, backend, full-stack
- Static vs Dynamic websites
- Client-server model
- Roles: Web Designer, Web Developer, Backend Developer, Full Stack Developer

💡Mini Task: Research 5 famous websites and identify what tech they may use (frontend/backend).

🧠 Lesson 1.2: How the Web Works

- Domain, IP Address, DNS
- HTTP vs HTTPS
- Browser, Server, Hosting
- Request → Response cycle
- Example: Browser requesting `index.html` from server

🧠 Lesson 1.3: Developer Tools & Setup

- Install VS Code
- Install Live Server Extension
- Install Google Chrome (or Firefox)
- Use browser DevTools (Inspect, Console, Network tab)

💡Mini Task: Open any site, inspect its HTML and try changing text/colors temporarily using DevTools.

🧠 Lesson 1.4: File Structure & First Web Page

- Create a new project folder
- Create `index.html` with boilerplate code
- Use Live Server to preview changes

🛠️ Mini Project (End of Week 1):

- Build a simple "About Me" page using only HTML
  - Headings
  - Paragraph
  - List of hobbies
  - Your photo (use a placeholder)

✅ Module 2: HTML – Structure of Web Pages (Week 2)

🎯 Goal: Learn to build structured, semantic, and accessible web pages using HTML5.

🧠 Lesson 2.1: Introduction to HTML

- What is HTML? (HyperText Markup Language)
- HTML Elements and Tags
- HTML Document Structure: `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`
- Creating your first HTML file

💡**Task:** Create a new file called `index.html` and replicate the above page.

🧠 Lesson 2.2: Headings, Paragraphs, and Text Formatting

- `<h1>` to `<h6>` (Importance of hierarchy)
- `<p>` – Paragraphs
- Text formatting tags: `<b>`, `<strong>`, `<i>`, `<em>`, `<u>`, `<mark>`, `<sub>`, `<sup>`

💡**Task:** Create a short article with heading, subheading, bold, italic, and marked words.

🧠 Lesson 2.3: Lists

- Ordered List: `<ol>`, `<li>`
- Unordered List: `<ul>`, `<li>`
- Nested Lists

💡**Task:** Create a grocery list with nested categories (Fruits, Vegetables, etc.)

🧠 Lesson 2.4: Links and Images

- `<a href="">` – Anchor tag
  - Open in same tab vs new tab (`target="_blank"`)
  - Link to external site, internal file, email
- `<img src="" alt="">` – Image tag
  - Width, height attributes
  - Online image vs local image

💡**Task:** Create a page with:

- A link to Google
- An image of your favorite food
- A clickable email link (`mailto:`)

🧠 Lesson 2.5: Tables

- `<table>`, `<tr>`, `<td>`, `<th>`
- Attributes: `border`, `colspan`, `rowspan`
- Semantic table formatting

💡**Task:** Create a 3x3 table showing student names, subjects, and marks.

🧠 **Lesson 2.6: Forms and Inputs**

- `<form>`, `<input>`, `<textarea>`, `<button>`
- Input types: `text`, `email`, `number`, `checkbox`, `radio`, `submit`, `password`, `file`, `date`
- `<label>` and `for` attribute
- Fieldsets and Legends

💡**Task:** Build a feedback form with name, email, rating (radio), and comments.

🧠 **Lesson 2.7: Semantic HTML5 Tags**

- Why semantic tags matter (SEO, accessibility)
- `<header>`, `<footer>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`

💡**Task:** Redesign your "About Me" page using semantic HTML layout.

🧠 **Lesson 2.8: HTML Media**

- `<audio>`, `<video>` tags
- Controls, autoplay, loop, poster
- Embedding YouTube using `<iframe>`

💡**Task:** Create a media gallery page with 1 video, 1 audio file, and 1 YouTube embed.

🛠️ Mini Project (End of Module 2): Personal Portfolio Page

Your portfolio should include:

- Header with your name and nav menu
- A profile image and intro paragraph
- A list of skills (unordered list)
- Table of completed courses
- A contact form
- Footer with links

✅ Module 3: CSS – Styling the Web (Weeks 3 & 4)

🎯 Goal: Understand how to style HTML elements using CSS and build responsive layouts using modern techniques like Flexbox and Grid.

🧠 Lesson 3.1: Introduction to CSS

- What is CSS? (Cascading Style Sheets)
- Three ways to apply CSS:
  - Inline (`style=""`)
  - Internal (`<style>` tag in `<head>`)
  - External (`style.css` file with `<link>`)
- CSS Syntax:

💡**Task:** Create a page with inline, internal, and external CSS and observe the priority.

🧠 Lesson 3.2: Basic Styling Properties

- `color`, `background-color`
- `font-size`, `font-family`, `font-style`
- `text-align`, `text-decoration`, `line-height`
- `padding`, `margin`, `border`

💡**Task:** Style an article with readable typography, spacing, and borders.

🧠 Lesson 3.3: Classes, IDs, and Selectors

- `.class`, `#id`, `element`

- Descendant, child, sibling selectors
- Grouping multiple selectors

💡**Task:** Create a layout with sections using different classes and IDs. Apply different background colors to each.

🧠 Lesson 3.4: Box Model

- Content → Padding → Border → Margin
- `width`, `height`, `box-sizing`
- `overflow`

💡**Task:** Create a card layout with padding, border, and spacing between cards.

🧠 Lesson 3.5: Positioning and Display

- `display`: block, inline, inline-block, none
- `position`: static, relative, absolute, fixed, sticky
- `z-index`
- Example of sticky header and fixed footer

💡**Task:** Create a sticky navigation bar at the top of the page.

🧠 Lesson 3.6: Flexbox

- Parent: `display: flex`
- Direction: `flex-direction`
- Alignment: `justify-content`, `align-items`
- `gap`, `flex-wrap`

- Responsive layout with Flexbox

💡**Task:** Create a 3-column responsive card layout using Flexbox.

🧠 Lesson 3.7: Grid Layout

- `display: grid`
- `grid-template-columns`, `grid-template-rows`
- `gap`, `grid-column`, `grid-row`
- Grid vs Flexbox: when to use which

💡**Task:** Build a 3x2 image gallery using CSS Grid.

🧠 Lesson 3.8: Pseudo Classes & Pseudo Elements

- `:hover`, `:focus`, `:active`, `:nth-child`
- `::before`, `::after`

💡**Task:** Create buttons that change appearance on hover and focus.

🧠 Lesson 3.9: Responsive Design with Media Queries

- `@media` rules
- Breakpoints: Mobile, Tablet, Desktop
- Adjust layout and text sizes based on screen width

💡**Task:** Make your Flexbox layout responsive for mobile screens.

🧠 Lesson 3.10: Animations & Transitions

- `transition` on hover
- Keyframe animations
- `@keyframes`, `animation-name`, `animation-duration`

💡**Task:** Animate a welcome message to fade in when the page loads.

🛠️ Mini Project (End of Module 3): Portfolio v2 – Styled Edition

Upgrade your HTML-only portfolio to include:

- A modern layout using Flexbox/Grid
- Themed color palette
- Styled navigation bar
- Hover effects on buttons and links
- Responsive design for mobile

✅ Module 4: JavaScript – Make Your Website Interactive (Weeks 5 & 6)

🎯 Goal: Learn JavaScript fundamentals and apply them to interact with HTML and CSS.

🧠 Lesson 4.1: Introduction to JavaScript

- What is JavaScript? What can it do?
- Adding JavaScript to HTML:
    - Inline (`onclick`)
    - Internal (`<script>` tag)
    - External (`script.js`)
- Your first `console.log()`

💡**Task:** Create a basic HTML file and log your name and age in the console.

🧠 Lesson 4.2: Variables and Data Types

- `var`, `let`, `const`
- Data Types: `String`, `Number`, `Boolean`, `Array`, `Object`, `null`, `undefined`

💡**Task:** Create variables for a user profile: name, age, isLoggedIn, hobbies.

🧠 Lesson 4.3: Operators and Expressions

- Arithmetic Operators: `+`, `-`, `*`, `/`, `%`

- Assignment Operators: `=`, `+=`, `-=`, etc.
- Comparison: `==`, `===`, `!=`, `>`, `<`
- Logical: `&&`, `||`, `!`

💡**Task:** Create a calculator that takes 2 numbers and logs the result of all operations.

🧠 Lesson 4.4: Conditional Statements

- `if`, `else if`, `else`
- `switch` statement

💡**Task:** Create a grading system that shows A/B/C/D based on marks.

🧠 Lesson 4.5: Loops

- `for`, `while`, `do...while`
- Loop through numbers, strings, arrays

💡**Task:** Loop through an array of fruits and display each one.

🧠 Lesson 4.6: Functions

- Function declaration vs expression
- Parameters, arguments, return values
- Arrow functions

💡**Task:** Write a function that calculates and returns the area of a rectangle.

🧠 Lesson 4.7: Arrays and Objects

- Creating and accessing arrays
- Array methods: `push()`, `pop()`, `shift()`, `unshift()`, `length`
- Objects: key-value pairs
- Accessing object properties

💡**Task:** Create an object for a car with properties: brand, model, year, and methods to start and stop the car.

🧠 Lesson 4.8: Events and Event Handling

- `onclick`, `onmouseover`, `onchange`, etc.
- `addEventListener()`
- Respond to user interaction

💡**Task:** Make a button that changes the background color of the page when clicked.

🧠 Lesson 4.9: DOM Manipulation

- `getElementById`, `querySelector`, `getElementsByClassName`
- Changing content: `innerText`, `innerHTML`
- Changing styles using JS
- Creating and removing elements

💡**Task:** Create a to-do list that adds items to a list dynamically using JS.

🧠 Lesson 4.10: Form Validation

- Prevent form submission
- Check for empty fields, valid email, min/max length
- Show error messages

💡**Task:** Build a contact form that validates name, email, and message before submitting.

🛠️ Mini Project (End of Module 4): Interactive Portfolio

Enhance your styled portfolio to:

- Show an alert on page load with a welcome message
- Toggle dark/light theme using a button
- Validate the contact form before submission
- Use JS to dynamically update your skills list

✅ Module 5: PHP – Backend Basics (Weeks 7 & 8)

🎯 Goal: Learn PHP fundamentals, handle forms, and connect to a MySQL database.

🧠 Lesson 5.1: Introduction to PHP

- What is PHP and how it works on the server
- `.php` file structure
- Mixing HTML + PHP
- Basic syntax: `<?php ?>`, echo, comments

💡**Task:** Create a basic `.php` file and output your name.

🧠 Lesson 5.2: Variables, Data Types & Constants

- `$variable` syntax
- Strings, integers, booleans, arrays, null
- `define()` for constants
-

💡**Task:** Create a PHP file to store user profile info and display it.

🧠 Lesson 5.3: Operators & Expressions

- Arithmetic, assignment, comparison, logical operators
- String concatenation (`.` operator)

💡**Task:** Create a calculator in PHP for two numbers.

🧠 **Lesson 5.4: Conditional Statements**

- `if`, `else if`, `else`
- `switch` statement

💡**Task:** Write a grading system based on PHP `if-else` logic.

🧠 **Lesson 5.5: Loops**

- `for`, `while`, `do...while`, `foreach`

💡**Task:** Display all months of the year using a `for` loop.

🧠 **Lesson 5.6: Functions**

- Creating and calling functions
- Parameters and return values

💡**Task:** Create a function that returns the square of a number.

🧠 **Lesson 5.7: Working with Forms (GET & POST)**

- HTML form + PHP backend
- `$_GET` vs `$_POST`

- `isset()` and `empty()` functions

💡**Task:** Create a form that takes user name and email and displays it using PHP.

🧠 Lesson 5.8: Arrays and Associative Arrays

- Indexed arrays
- Associative arrays
- `print_r()` and `var_dump()`

💡**Task:** Create an associative array of 3 students with name and score. Loop through it.

🧠 Lesson 5.9: File Handling

- Opening, reading, writing files
- `fopen()`, `fwrite()`, `fread()`, `fclose()`

💡**Task:** Write a form that saves user data into a `.txt` file.

🧠 Lesson 5.10: Introduction to PHP & MySQL

- What is a database?
- Introduction to MySQL
- Using PHP `mysqli_connect()` and `mysqli_query()`

💡**Task:** Connect to your local database and fetch user data from a `users` table.

🛠️ Mini Project (End of Module 5): Basic Contact Manager

- Create a form to input contact name, email, and phone
- Save the data to a MySQL database using PHP
- Display all saved contacts on another page

✅ 📌 Extended Module 5: Advanced PHP & MySQL (Weeks 9 & 10)

🎯 Goal: Build dynamic, data-driven applications using PHP and MySQL with best practices.

🧠 Lesson 5.11: Relational Database Concepts

- Primary Key, Foreign Key
- One-to-One, One-to-Many, Many-to-Many relationships
- Normalization: 1NF, 2NF, 3NF (Basic idea)
- ERD (Entity-Relationship Diagram)

💡**Task:** Design a schema for a student-course enrollment system.

🧠 Lesson 5.12: Advanced Form Handling

- Multi-step forms using sessions
- Retaining form data on validation errors
- File uploads: `enctype="multipart/form-data"`
- `$_FILES[]` and `move_uploaded_file()`

💡**Task:** Create a registration form that uploads a user's profile picture.

🧠 Lesson 5.13: PHP Sessions and Cookies

- `session_start()`, `$_SESSION`, `$_COOKIE`
- Use cases: user login, cart data
- Storing form data temporarily using sessions

💡**Task:** Create a multi-page form (Name → Address → Confirmation) using sessions.

🧠 Lesson 5.14: Full CRUD Application (Create, Read, Update, Delete)

- Insert data using prepared statements
- Display data in HTML tables
- Edit & update rows via ID
- Delete record with confirmation

💡**Task:** Build a full CRUD contact manager using `mysqli` and PHP forms.

🧠 Lesson 5.15: Using PHP with MySQL Securely

- SQL Injection and Prevention
- `mysqli_real_escape_string()` vs Prepared Statements
- Introduction to `PDO` (as optional best practice)

💡**Task:** Rebuild your earlier form using `prepared statements` for safety.

🧠 Lesson 5.16: Search, Pagination, and Sorting

- Search records via form input
- Display paginated results using `LIMIT` and `OFFSET`
- Sort table data by name, date, etc.

💡**Task:** Create a student directory with live search and page-wise display.

🧠 Lesson 5.17: User Authentication System

- Register & login form
- Password hashing (`password_hash()` and `password_verify()`)
- Session-based login
- Logout and session destroy

💡**Task:** Build a login/logout system with access-restricted dashboard.

🧠 Lesson 5.18: Admin Panel UI with PHP

- Create a basic dashboard using HTML + PHP
- Sidebar/Menu for managing pages (Users, Posts, etc.)
- Create dynamic table views and status badges

💡**Task:** Create a basic admin interface to manage blog posts.

🛠️ Capstone Project: Mini Blog with Admin Panel

- Admin Login / Logout
- Post Creation (title, content, image)
- Display posts on public page
- Edit/Delete options for admin
- Database Tables: users, posts

✅ 📌 Module 6: React.js (Weeks 11 to 14)

Goal: Learn to build modern and dynamic web applications using React.js, with an introduction to state management using Redux and Context API, and an in-depth look at React Hooks.

🧠 Lesson 6.1: Introduction to React

- What is React and why use it?
- Setting up the development environment (Node.js, npm, create-react-app)
- Understanding components, JSX, and the virtual DOM

🧠 Lesson 6.2: Components & Props

- Creating functional components
- Using props to pass data
- Component composition and reuse

🧠 Lesson 6.3: State and Event Handling

- Introduction to `useState`
- Handling user input and events
- Conditional rendering (`if`, `ternary`, `&&` rendering)

🧠 Lesson 6.4: useEffect and Component Lifecycle

- Introduction to `useEffect` hook
- Using side effects (e.g., fetching data)
- Cleaning up effects

🧠 Lesson 6.5: Forms in React

- Controlled vs uncontrolled components
- Managing form data with state
- Handling form submissions
- Basic form validation (required fields, length, etc.)

🧠 Lesson 6.6: Lists and Keys

- Rendering lists using `map()`
- Understanding the importance of `key` prop
- Using lists with dynamic state (e.g., delete or edit items)

🧠 Lesson 6.7: React Router (v6)

- Installing and setting up React Router
- Creating multiple pages (Routes and Route)
- Navigating between pages using `Link` and `useNavigate`
- Using `useParams` for dynamic routing

🧠 Lesson 6.8: Project Structure and Best Practices

- Organizing folders: components, pages, hooks, assets, etc.
- Code readability and maintainability
- Using ESLint and Prettier

🧠 Lesson 6.9: Fetching Data and APIs

- Using `fetch()` and `axios`

- Displaying data from a public API
- Handling loading and error states

🧠 Lesson 6.10: Redux Introduction

- What is Redux and why use it?
- Installing Redux and React-Redux
- Store, actions, and reducers
- Connecting React components to Redux store with `connect` and `useDispatch`
- Managing global state in larger applications

🧠 Lesson 6.11: Redux Deep Dive

- Setting up the Redux store with middleware (e.g., redux-thunk)
- Writing complex reducers
- Handling asynchronous actions in Redux (thunks)
- Understanding Redux DevTools
- Normalizing data and state structure for optimization

🧠 Lesson 6.12: Context API

- Introduction to Context API
- Creating a global state using `React.createContext()`
- Using `useContext` to consume context in functional components
- Best practices for Context API in large applications
- When to use Context vs Redux

🧠 Lesson 6.13: Advanced React Hooks

- `useReducer` for complex state logic
- `useMemo` and `useCallback` for performance optimization
- Custom Hooks: Creating reusable logic
- `useRef` for accessing DOM elements and persisting values across renders
- `useLayoutEffect` vs `useEffect`

🧠 Lesson 6.14: Final React Project – Task Manager App

- Add, edit, and delete tasks
- Store data using either `localStorage` or connect to a backend (PHP optional)
- Form handling and validation
- Routing between pages (e.g., Home, Add Task, Task Details)
- Clean UI with optional styling libraries like Bootstrap or Tailwind CSS
- Implement Redux or Context API for state management (optional)
- Using hooks for reusable logic across the app

✅ 📌 Extended Module 7: Redux Deep Dive (Weeks 14 to 15)

Goal: Master Redux state management with a focus on advanced patterns, asynchronous actions, middleware, and optimizations.

🧠 Lesson 7.1: Redux Middleware

- Introduction to middleware in Redux
- What is middleware and why is it useful?
- Installing and configuring middleware (e.g., `redux-thunk`, `redux-saga`)
- Using `redux-thunk` for asynchronous actions
- Creating custom middleware for specific use cases

🧠 Lesson 7.2: Redux Thunk for Asynchronous Actions

- Setting up `redux-thunk` middleware
- Understanding the difference between synchronous and asynchronous actions
- Dispatching async actions using `dispatch`
- Handling API requests (GET, POST) with Redux actions
- Example: Fetching data from an API (loading, success, error states)

🧠 Lesson 7.3: Redux Toolkit

- Introduction to Redux Toolkit for easier Redux development
- Installing and setting up Redux Toolkit
- Using `createSlice()` for reducers and actions
- Using `createAsyncThunk()` for async actions
- Benefits of Redux Toolkit over traditional Redux

🧠 Lesson 7.4: Normalizing Data in Redux

- What is data normalization and why is it important?
- Flattening nested data structures
- Storing data in normalized form (using ids)
- Example: Managing lists of users/products with normalized data
- Using `normalizr` library to normalize data automatically

🧠 Lesson 8.5: Optimizing Redux Performance

- Avoiding unnecessary re-renders in React by using `useSelector` efficiently
- Using `reselect` for memoized selectors
- Splitting Redux state into smaller slices
- Leveraging `React.memo` for preventing unnecessary rendering of components

---

🧠 Lesson 7.6: Advanced Patterns in Redux

- Managing complex state with multiple reducers (combineReducers)
- Handling state transitions with complex data models
- Using Redux for global state management (authentication, user settings)
- Benefits of Redux in large-scale applications

🧠 Lesson 7.7: Testing Redux Actions and Reducers

- Setting up testing tools for Redux (Jest, React Testing Library)
- Writing unit tests for actions and reducers
- Mocking API requests and testing async actions with `redux-thunk`

- Testing selectors and ensuring correct state updates

🧠 Lesson 7.8: Error Handling in Redux

- Managing errors in the Redux state
- Using `redux-thunk` to handle errors in async actions
- Dispatching error messages to Redux store
- Displaying error messages in the UI with React components

🧠 Lesson 7.9: Final Project Redux Integration

- Applying Redux in a real-world application (e.g., E-commerce or Task Manager)
- Managing state for products, cart, user authentication, and orders
- Combining Redux with React Router for global state management across pages
- Optimizing Redux state management for performance
- Final review of Redux implementation in the project

✅ 📌 Bonus Module: Advanced Topics (Optional)

🧠 Lesson 8.1: Git & GitHub Basics

Goal: Learn the fundamentals of version control with Git, and how to use GitHub for collaboration and project management.

- **What is Git?**
  - Introduction to version control
  - Git vs. SVN and other version control systems
  - Installing Git and setting up a repository
- **Basic Git Commands**
  - `git init`: Initializing a new Git repository
  - `git clone`: Cloning a remote repository
  - `git add`: Staging changes
  - `git commit`: Committing changes with meaningful messages
  - `git status`: Checking the status of changes
  - `git push` and `git pull`: Pushing and pulling changes from a remote repository
- **Branching and Merging**
  - Creating and switching branches (`git branch`, `git checkout`)
  - Merging branches (`git merge`)
  - Resolving merge conflicts
- **Collaborating with GitHub**
  - Setting up a GitHub account and creating repositories
  - Using `git remote` to connect a local repository to GitHub
  - Pushing code to GitHub and collaborating with others
  - Forking, Pull Requests, and Code Reviews
- **Best Practices**
  - Writing clear commit messages
  - Using `.gitignore` to exclude unnecessary files
  - Working with GitHub issues, labels, and projects for managing tasks

---

🧠 Lesson 8.2: API Concepts (REST)

Goal: Understand the core principles of RESTful API development and consumption.

- **What is an API?**
  - API definition and examples
  - Introduction to REST (Representational State Transfer)
  - Client-server architecture and statelessness
  - Common HTTP methods: `GET`, `POST`, `PUT`, `DELETE`
- **Building a Simple REST API (PHP Backend)**
  - Overview of REST API structure
  - Setting up routes and endpoints for CRUD operations
  - Returning data in JSON format
  - Handling HTTP requests and responses
  - Error handling in API responses
- **Consuming APIs with JavaScript**
  - Fetching data from an API using `fetch()` or `axios`
  - Handling asynchronous requests with `async/await`
  - Working with JSON data (parsing and stringifying)
  - Displaying API data dynamically in the frontend (React or HTML)
- **Best Practices for REST APIs**
  - RESTful URL conventions (e.g., `/users/{id}`, `/products`)
  - Status codes and their meanings (200 OK, 400 Bad Request, 404 Not Found, 500 Internal Server Error)
  - Pagination, sorting, and filtering in API responses
  - Authentication (JWT, OAuth, etc.)

🧠 Lesson 8.3: JSON Handling

Goal: Understand how to work with JSON (JavaScript Object Notation) for data storage, communication, and integration.

- **What is JSON?**
  - JSON syntax (objects, arrays, key-value pairs)
  - Why JSON is used for data exchange
- **Working with JSON in JavaScript**
  - Parsing JSON strings into JavaScript objects (`JSON.parse()`)
  - Stringifying JavaScript objects into JSON (`JSON.stringify()`)
- **Handling JSON in APIs**
  - Sending JSON in HTTP requests
  - Sending JSON with `POST` and `PUT` methods in API requests
  - Reading JSON responses from APIs
- **Best Practices**
  - Validating and sanitizing JSON data
  - Using tools like `JSONLint` for validation
  - Formatting JSON for readability

🧠 Lesson 8.4: Introduction to Laravel (PHP Framework)

Goal: Learn the basics of Laravel, a popular PHP framework, and understand how to build modern PHP applications using Laravel's features.

- **What is Laravel?**
  - Introduction to Laravel as a PHP MVC framework
  - Benefits of using Laravel for web development
  - Installing Laravel and setting up the development environment
- **Laravel Directory Structure**
  - Understanding key directories: `app`, `routes`, `resources`, `database`, and `public`
  - MVC architecture in Laravel (Models, Views, Controllers)
- **Routing in Laravel**

    - Defining routes in `routes/web.php`
    - Handling different HTTP request types (`GET`, `POST`, `PUT`, `DELETE`)
    - Route parameters and query strings
- **Controllers in Laravel**
    - Creating controllers using Artisan commands
    - Passing data from controllers to views
    - Handling form submissions and validation
- **Database and Eloquent ORM**
    - Setting up a database connection in `.env` file
    - Creating models for database tables
    - Performing CRUD operations using Eloquent ORM
    - Running database migrations with Artisan
- **Blade Templating Engine**
    - Creating views with Blade syntax
    - Using Blade directives (e.g., `@if`, `@foreach`, `@include`)
    - Sharing data between views and controllers
- **Authentication in Laravel**
    - Setting up Laravel's built-in authentication system
    - Registration, login, and password reset
    - Protecting routes with authentication middleware
- **Basic Laravel Application – Blog Example**
    - Creating a simple blog application with Laravel
    - Creating posts (CRUD operations)
    - Displaying posts using Blade templates
    - Protecting the creation of posts (admin access only)

✅ 📌 Bonus Module 9: E-commerce Project with Admin Panel (PHP, React, Redux, MySQL)

Goal: Build a small e-commerce platform from scratch, including an admin panel to manage products, users, and orders using PHP, HTML, CSS, JS, React, and Redux.

🧠 Lesson 9.1: Project Setup

- Overview of the project architecture (Frontend with React, Backend with PHP)
- Setting up the local development environment (XAMPP, PHP, MySQL, Node.js)
- Installing React and setting up the initial folder structure
- Introduction to PHP for backend APIs (CRUD for products, users, and orders)

🧠 Lesson 9.2: Frontend – Creating the React App

- Setting up React (Create React App)
- Folder structure for frontend (components, pages, services, redux)
- Building static pages (Home, Product List, Product Details)
- Styling the components using CSS or frameworks like Bootstrap/Tailwind CSS

🧠 Lesson 9.3: Backend – PHP & MySQL

- Setting up the PHP API backend
- Connecting to MySQL using PDO
- Creating tables for products, users, orders, and categories
- Writing CRUD operations for products (Create, Read, Update, Delete)

- Implementing user authentication (login, registration)
- Writing API routes for fetching product data and managing orders

---

🧠 Lesson 9.4: Admin Panel – PHP & MySQL

- Admin login and authentication
- Managing products from the admin panel (CRUD)
- Managing orders (viewing orders, changing order statuses)
- Managing users (viewing user details, deleting users)
- Implementing file uploads for product images

🧠 Lesson 9.5: Redux – Managing Global State

- Introduction to Redux and setting up the store
- Actions and reducers for products, cart, and user authentication
- Managing product list and cart state in Redux
- Implementing the shopping cart functionality (add to cart, remove from cart, view cart)
- Implementing user authentication state (login, logout)

🧠 Lesson 9.6: Implementing the Shopping Cart

- Building the shopping cart page (display cart items, total amount, checkout)
- Integrating the cart with the backend (creating an order, updating inventory)
- Storing cart state in Redux
- Checkout form (shipping details, payment options)

🧠 Lesson 9.7: Final Integration

- Integrating frontend React with backend PHP APIs
- Connecting Redux state with React components
- Validating form inputs (both frontend and backend)
- Error handling and displaying user-friendly error messages
- Testing the application and making improvements

🧠 Lesson 9.8: Admin Panel Enhancements

- Admin dashboard (order statistics, product count)
- Managing product categories, brands
- Improving user interface and user experience for the admin panel

---

🧠 Lesson 9.9: Final E-commerce Project – Deployment

- Preparing for production (minifying CSS, JS, setting up production-ready React app)
- Deploying the backend (PHP/MySQL) to a server (using services like Heroku, Bluehost, or a VPS)
- Deploying the frontend (React app) to a platform like Netlify or Vercel
- Final testing on both frontend and backend

🧠 Lesson 9.10: Final Project Review and Enhancements

- Reviewing the code for any issues or improvements
- Refactoring code for scalability and performance
- Optional: Adding features like product search, ratings, or user reviews
- Enhancing the admin panel for better usability

✅ 📌 Bonus Module 10: Career Development & Job Preparation

Lesson 10.1: Resume Building for Developers

Objective: Craft a professional, ATS-friendly resume that stands out in the tech industry.

- What is a Resume vs. CV – and which one to use
- Structure of a Software Developer Resume:
  - Contact Info
  - Objective/Summary
  - Skills (Languages, Frameworks, Tools)
  - Projects (with GitHub links)
  - Work Experience or Internships
  - Education
  - Certifications (if any)
- Action Verbs and Impactful Descriptions
- Common mistakes to avoid in developer resumes
- Resume design tips and free templates
- How to tailor your resume for specific job roles
- Creating a LinkedIn profile that matches your resume

Lesson 10.2: Interview Preparation (Technical + HR)

Objective: Prepare for both technical and non-technical interviews in the software development field.

A. Technical Interview Prep

- Types of interviews: phone screen, coding round, system design, project discussion
- Common Frontend Developer Questions (HTML, CSS, JS, React)
- Backend Developer Questions (PHP, MySQL, APIs)
- Debugging and code analysis questions

- Logic and problem-solving (basic DSA and real-world coding scenarios)
- Live coding tools: HackerRank, CodeSignal, etc.

B. Behavioral/HR Interview Prep

- Common HR questions and how to answer them:
    - Tell me about yourself
    - Strengths and weaknesses
    - Where do you see yourself in 5 years?
    - Why should we hire you?
- STAR method (Situation, Task, Action, Result) for story-based answers
- How to talk about academic or personal projects
- How to handle rejection and follow-up emails

Lesson 10.3: Job Search Strategies

Objective: Learn how and where to search for job opportunities, internships, and freelance work.

- Where to find job listings: LinkedIn, Indeed, Internshala, Stack Overflow Jobs, GitHub Jobs
- Writing cover letters and email proposals
- Networking tips: engaging with developers, recruiters, and alumni
- Creating and sharing a portfolio website
- Contributing to open-source projects to gain visibility
- Freelancing basics and platforms (Fiverr, Upwork)
- Applying for internships & junior developer roles strategically