# Full Stack Developer Course Syllabus

## Core Skills

- JavaScript
- TypeScript
- Python

## Frontend Development

### UI/UX Design

- Tools:
- Figma

### For Web Development

- HTML
- CSS
- Tailwind CSS
- React

### Backend Development

- Node.js (Express)
- NestJS

### Database

- MongoDB

### Version Control

- Git

### Machine Learning

- LLM (Large Language Models)
- Tensorflow
- Pytorch
- Mojo
- mindsDB

### For Mobile Development

- React Native
- Flutter
- Flutter Flow (Tool)

### Big data

- Kafka
- Elastic Search

**Testing tools**

- Postman
- Selenium

**UX/UI Syllabus**

Tools: Figma, Adobe XD

**1.Research**

Creating a research plan,Desk research,Competitor evaluation,Business value identification,Domain research

**Sources:**

- How to Create a Research Plan in 7 Steps (Maze)

https://maze.co/guides/ux-research/plan/

- Creating a UX Research Plan

https://www.youtube.com/watch?v=VKl8X16GoiY

- UX Research Methods Overview

https://www.youtube.com/watch?v=UYV-aPsSzI0

- UX Showcase - Benefits of desk research

https://www.youtube.com/watch?v=C8JCoV0-6XE

- Domain research and how to do one

https://uxplanet.org/ux-design-domain-research-and-how-to-do-one-1d7ead522c3f

**2. Qualitative methods**

User Interviews, Usability Testing

**Sources:**

- User Interviews: How, When, and Why to Conduct Them

https://www.nngroup.com/articles/user-interviews/

- Quantitative User-Research Methodologies: An Overview

https://www.nngroup.com/articles/quantitative-user-research-methods/

**3.UX DESIGN**

**Design Patterns and Guidelines**

General UX-patterns and guidelines, Patterns and Guidelines, Psychology (Laws, Principles), User engagement, Conceptual design, Wireframing,

**Interactive prototyping, UX Writing**

**Sources:**

- The Laws of UX - 19 Psychological Design Principles

https://www.youtube.com/watch?v=fYs2Mdyasuc

- Personas vs Customer Segments vs Jobs to be Done vs... in UX

https://www.youtube.com/watch?v=76LyRKts1Ac

- 50 UI tips (UX-patterns)

https://rattibha.com/thread/1368851471660421120

- Field Studies

https://www.nngroup.com/articles/field-studies/

**4.Visual Design**

Data visualization,Typography,Coloristics,Composition,Visual theory,Iconography,Illustration, Promo-materials, Graphic design

**Sources:**

- Understanding visual design

https://www.youtube.com/watch?v=yNDgFK2Jj1E

- Principles of graphic design

https://www.youtube.com/watch?v=aQI95mVeDXw

**5.Design Operations**

**Processes and Optimization**

Design process models,Methodology: Agile / Kanban / Waterfall / Scrum / Lean UX, Design management (UX debt, Backlog, Planning), Design processes, Design systems (Kits, Guidelines, Libraries, etc.), Design optimization, Design documentation.

**Sources:**

- Lean vs Agile vs Design Thinking vs... YOU

https://www.youtube.com/watch?v=OCL6RkUOShI

**HTML Study Topics:**

### 1. HTML Structure & Document Setup:

- <!DOCTYPE html> declaration
- The <html> element
- The <head> element:
    - <title> tag
    - <meta> tags (viewport, charset, etc.)
    - Linking to external resources (CSS, scripts)
- The <body> element

### 2. Basic HTML Elements:

- Headings (<h1> to <h6>)
- Paragraphs (<p>)
- Line breaks (<br>)
- Horizontal rules (<hr>)
- Emphasis:  (italicized text),  (bold text)
- Generic containers: , 
- Comments (<!-- ... -->)

### 3. Lists:

- Unordered lists (<ul>) and list items (<li>)
- Ordered lists (<ol>) and list items (<li>)
- Description lists (<dl>, <dt>, <dd>)

### 4. Links and Navigation:

- The anchor tag (<a>)
- href attribute:
    - Internal links (same page, different page within website)
    - External links (other websites)
    - Email links (mailto:)
- Navigation elements: <nav>

### 5. Images and Media:

- The image tag (<img>)
    - src attribute (source of the image)
    - alt attribute (alternative text for accessibility)
    - Relative and absolute paths
- Basic Audio (<audio>) and Video (<video>) Embedding

### 6. Tables:

- <table>, <tr> (table row), <th> (table header), <td> (table data)
- colspan and rowspan attributes
- <thead>, <tbody>, <tfoot>

### 7. Forms:

- The <form> element
    - action attribute (where to send the form data)
    - method attribute (GET or POST)
- Input elements (<input>) and their type attributes:
    - type="text" (text input)
    - type="password" (password input)
    - type="email" (email input)
    - type="checkbox" (checkbox)
    - type="radio" (radio button)
    - type="submit" (submit button)
    - type="reset" (reset button)
    - type="number" (number input)
    - type="date" (date input)
    - type="file" (file input)
- <textarea> element (multi-line text input)
- <select> and <option> elements (dropdown menus)
- <button> element
- Form attributes: placeholder, required, value, name, id
- Label element (<label>) for accessibility

### 8. Semantic HTML:

- Understanding the purpose of semantic HTML
- Semantic elements: <header>, <nav>, <main>, <article>, <aside>, <footer>
- Benefits for SEO and accessibility

### 9. Accessibility:

- Importance of accessibility
- Using semantic HTML
- Providing alt text for images
- Using labels for form fields
- Understanding ARIA attributes

## CSS Study Topics:

### 1. CSS Fundamentals:

- What is CSS and its purpose
- Ways to include CSS:
    - Inline styles
    - Internal styles (<style>)
    - External stylesheets (<link rel="stylesheet">)
- CSS syntax: selectors, properties, values
- Comments in CSS (/* ... */)

### 2. CSS Selectors:

- Element selectors (e.g., p, h1)
- Class selectors (e.g., .my-class)
- ID selectors (e.g., #my-id)
- Attribute selectors (e.g., [type="text"])
- Universal selector (*)
- Descendant selectors (e.g., nav ul li)
- Child selectors (e.g., nav > ul > li)
- Adjacent sibling selectors (e.g., h2 + p)
- General sibling selectors (e.g., h2 ~ p)
- Pseudo-classes (e.g., :hover, :focus, :active, :nth-child())
- Pseudo-elements (e.g., ::before, ::after)
- Combining selectors

### 3. CSS Properties:

- **Text Properties:** color, font-family, font-size, font-weight, font-style, line-height, text-align, text-decoration, letter-spacing, word-spacing, text-transform, text-shadow.
- **Background Properties:** background-color, background-image, background-repeat, background-position, background-size
- **Box Model Properties:** width, height, padding, margin, border (and its sub-properties: border-width, border-style, border-color), box-sizing
- **Layout Properties:**
    - display (block, inline, inline-block, flex, grid, none)
    - position (static, relative, absolute, fixed, sticky)
    - float and clear
    - visibility
    - overflow
    - z-index

### 4. CSS Box Model:

- Content, padding, border, margin
- Understanding how the box model impacts size and spacing
- box-sizing property (content-box, border-box)

### 5. Flexbox Layout:

- Flex container and flex items
- display: flex
- flex-direction (row, column, row-reverse, column-reverse)
- justify-content (space around, center, flex start, flex end, etc.)
- align-items (center, flex start, flex end, stretch)
- flex-wrap
- flex-grow, flex-shrink, flex-basis
- align-self
- Creating common layouts with Flexbox

### 6. Grid Layout:

- Grid container and grid items
- display: grid
- grid-template-rows, grid-template-columns
- grid-gap
- grid-column, grid-row
- grid-area
- Creating common layouts with CSS Grid

### 7. Positioning:

- position: static (default)
- position: relative
- position: absolute
- position: fixed
- position: sticky
- Using top, right, bottom, left properties

### 8. Responsive Web Design:

- Viewports and the <meta name="viewport"> tag
- Media queries (@media)
- Breakpoints
- Flexible grids
- Responsive images (using max-width: 100%)

### 9. CSS Specificity:

- Understanding CSS specificity rules
- Inline styles, ID selectors, class selectors, element selectors, attribute selectors
- How to handle specificity conflicts

### 10. CSS Transitions and Animations:

- Basic transitions with the transition property
- Basic animations using the @keyframes rule

## TypeScript Syllabus

## I. TypeScript Fundamentals

- **1.1 Introduction to TypeScript**
  - What is TypeScript?
  - Why use TypeScript? (Type Safety, Maintainability, Scalability)
  - Setting up a TypeScript project
- **1.2 Basic Types**
  - Primitives: number, string, boolean, null, undefined
  - Arrays: number[], string[], any[], Array<T>
  - Tuples
  - Enums
  - Any
- **1.3 Interfaces**
  - Defining contracts for objects
  - Optional properties
  - Read-only properties
  - Extending interfaces
- **1.4 Classes**
  - Class declaration
  - Constructors
  - Methods
  - Access modifiers (public, private, protected)
  - Inheritance
  - Abstract classes
- **1.5 Functions**
  - Function signatures
  - Optional and default parameters
  - Rest parameters
  - Generics
- **1.6 Type Assertions**
  - as syntax
  - **<> syntax**

## II. TypeScript in Practice

- **2.1 TypeScript with JavaScript Libraries**
  - Integrating TypeScript with existing JavaScript libraries (e.g., React)
  - Handling JavaScript libraries without type definitions
- **2.2 TypeScript in Node.js**
  - Creating Node.js applications with TypeScript
  - Using TypeScript with popular Node.js frameworks (e.g., Express)
- **2.3 Advanced TypeScript Concepts**
  - Union types

    - Intersection types
    - Conditional types
    - Type guards
    - Utility types (e.g., Partial, Required, Readonly)

**III. TypeScript in React**

- **3.1 Setting up TypeScript in React**
    - Creating a React project with TypeScript
    - Configuring TypeScript with Create React App
- **3.2 TypeScript in React Components**
    - Typing props
    - Typing state
    - Using interfaces with components
    - Handling events with TypeScript
- **3.3 TypeScript with React Hooks**
    - Typing custom hooks

**JS Study Topics:**

**I. JavaScript Fundamentals**

- 1.1 Introduction to JavaScript
    - What is JavaScript?
    - Where is JavaScript used? (Frontend, Backend, Mobile)

    - Basic Syntax (Variables, Data Types, Operators)
- 1.2 Control Flow
    - Conditional Statements (if/else, switch)
    - Loops (for, while, do-while)
- 1.3 Functions
    - Function Declaration
    - Function Expressions
    - Arrow Functions
    - Scope (Global, Local)
    - Closures
- 1.4 String methods
    - charAt() charCodeAt() concat()
    - indexOf() lastIndexOf() slice()
    - substring() substr() replace()
    - toUpperCase() toLowerCase() trim()
    - trimStart() trimEnd() padStart()
    - padEnd() split() includes()
    - match() matchAll() Find()
    - startsWith() endsWith()
    - repeat()
- 1.5 Array methods
    - Push() Pop() Shift()
    - Unshift() Splice() Sort()
    - Reverse() Fill() Concat()
    - Slice() Join() indexOf()
    - lastIndexOf() Includes() forEach()
    - Map() Filter() Reduce()
    - Every() Some() Find()
    - findIndex()
- 1.5 Objects
    - Object Literals
    - Object Methods
    - Object-Oriented Programming (OOP) Concepts (Classes, Inheritance, Polymorphism)
- 1.6 Date Object
    - Creating a Date Object
    - Setting the Date & Time by a Single String
    - Separating Variables with Commas
    - Displaying the Date & Time
    - Time Zones
    - Extracting the Date
    - Extracting the Hrs
    - Set Date Method
    - Set Time
    - Non-Data Object Functions
    - Form Objects
    - Button Objects
    - Text Objects
    - Text Area Objects
    - Hidden Objects
    - Check Box Objects

    - Radio Button Objects
    - Selecting Objects
    - Onchange Events
- 1.7 DOM Manipulation
    - Selecting Elements (getElementById, querySelector, etc.)
    - Modifying HTML and CSS
    - Handling Events (click, mouseover, etc.)

## 2. Advanced JavaScript

- Closures
    - Lexical Scope
    - Private Variables with Closures
- Prototypes and Inheritance
    - Prototypal Inheritance
    - ES6 Classes
- Async JavaScript
    - Callbacks
    - Promises
    - async/await
- Error Handling
    - try, catch, finally
    - Custom Errors
- Event Loop
    - Microtasks vs. Macrotasks
    - Understanding setTimeout, Promise, and async/await

## 3. DOM Manipulation

- Selecting Elements
    - getElementById, querySelector, querySelectorAll
- Modifying Elements
    - innerHTML, textContent, style
    - Adding/Removing Classes
- Events
    - Event Listeners (click, keyup, change)
    - Event Delegation
- Forms and Validation
    - Form Submission
    - Validating Email and Password

**Tailwind CSS Syllabus**

### 1. Introduction to Tailwind CSS

- **What is Tailwind CSS?**
- **Benefits of using Tailwind CSS**
- **Installation and Setup**
    - Using a CDN
    - Installing via npm/yarn
- **Understanding Utility-First CSS**

## 2. Core Concepts

- **Utility-First Design Philosophy**
- **Responsive Design with Tailwind**
- **Mobile-First Workflow**
- **Customizing the Configuration File**

## 3. Basic Utilities

- **Text and Font Utilities**
  - Text size, alignment, and color
  - Font family and weight
- **Background Utilities**
  - Background color and gradients
  - Background position and size
- **Border Utilities**
  - Border width, color, and style
  - Border radius
- **Spacing Utilities**
  - Margin and padding
  - Width and height

## 4. Layout and Positioning

- **Flexbox Utilities**
  - Aligning items and content
  - Justify content
  - Flex grow, shrink, and basis
- **Grid Utilities**
  - Creating grid layouts
  - Controlling rows and columns
- **Position Utilities**
  - Static, relative, absolute, fixed, and sticky
  - Z-index

## 5. Responsive Design

- **Breakpoints and Media Queries**
- **Tailwind's Predefined Breakpoints**
- **Applying Utilities Conditionally**

## 6. Advanced Styling

- **Pseudo-Classes and Pseudo-Elements**
  - Hover, focus, active, and visited states
  - First, last, and nth-child selectors
- **Transition and Animation Utilities**
  - Adding transitions and durations
  - Creating custom animations
- **Typography Plugin**
  - Styling prose content

## 7. Customization

- **Extending Tailwind's Default Theme**
  - Adding custom colors, fonts, and spacing

- **Using Plugins**
  - Official and third-party plugins
- **Purging Unused Styles**
- **JIT (Just-in-Time) Mode**

# React Syllabus

1. **Introduction to React**

**What is React?**

- Overview of React and its advantages.
- Single Page Applications (SPA) concept.
- Setting up the Environment
- Installing Node.js and npm/yarn.
- Creating a React app using Create React App and Vite.
- Understanding JSX
- What is JSX?
- Embedding expressions in JSX.
- JSX vs. HTML.

2. **Core React Concepts**

- React Components
- Function vs. Class Components.
- Component structure and lifecycle.
- Props and State
- Passing props to components.
- Managing state with useState.
- Props vs. State: Key differences.
- Event Handling
- Handling events in React (onClick, onChange, etc.).
- Passing arguments to event handlers.
- Conditional Rendering
- Using if, ternary operators, and && for rendering logic.

3. **Styling in React**

- CSS in React
- Inline styles.
- CSS Modules.
- Global stylesheets.
- Third-party Libraries
- Styled-components.
- Tailwind CSS.
- Emotion.

**4. React State Management**

- useState Hook
- Managing local state.
- Updating and resetting state.
- useReducer Hook
- Understanding reducers.
- When to use useReducer over useState.

**5. Advanced React Hooks**

- useEffect
- Lifecycle methods in functional components.
- Managing side effects and cleanup.
- useRef
- Accessing DOM elements.
- Persistent values across renders.
- useMemo & useCallback
- Performance optimization.

- Memoizing computations and functions.
- Custom Hooks
- Creating and using custom hooks.

6**. Working with Forms**

- Forms and Inputs
- Controlled vs. uncontrolled components.
- Handling form submissions.
- React Hook Form
- Simplifying form handling.
- Validation with libraries like yup and zod.

7. **Routing**

- React Router
- Setting up React Router.
- Creating routes with Route and Switch/Routes.
- Handling 404 pages.
- Dynamic Routing
- Route parameters.

**8. React Context API**

- Global State Management
- Creating and providing context.
- Consuming context with useContext.
- 9. Redux for State Management
- Introduction to Redux
- Core concepts: Store, Actions, Reducers.
- Redux Toolkit
- Setting up Redux Toolkit.
- Creating slices and async thunks.
- Integration with React
- Using useSelector and useDispatch.

10. **Fetching Data**

- API Integration
- Using fetch and axios to call APIs.
- Handling loading and error states.
- React Query
- Data fetching and caching.
- Background updates and polling.

11**. Component Patterns**

- Higher-Order Components (HOC)
- Render Props
- Compound Components
- Controlled and Uncontrolled Components

**12. Performance Optimization**

- React.memo
- Preventing unnecessary re-renders.
- Code Splitting
- Using React's lazy and Suspense.
- Dynamic imports.

13. **Testing in React**

- Unit Testing
- Testing components with Jest and React Testing Library.
- Integration Testing
- Mocking APIs.
- Testing user flows.

14. **TypeScript with React**

- Setting up TypeScript
- Adding TypeScript to a React project.
- Typing Components
- Props, state, and hooks with TypeScript.

15. **Working with Third-party Libraries**

- UI Libraries
- Material-UI, Ant Design, Shadcn/UI.
- Charts and Graphs
- React ApexCharts, Chart.js.

16. **Next.js for SSR and SSG**

- Introduction to Next.js
- Server-side rendering (SSR).
- Static site generation (SSG).
- Dynamic Routing
- API routes and middleware.

17**. Deployment**

- Building and Optimizing React Apps
- Using npm run build and analyzing bundles.
- Hosting

19. **Advanced Concepts**

- React Fiber Architecture
- Understanding rendering and reconciliation.
- Error Boundaries
- Handling runtime errors gracefully.
- Rendering components outside the DOM hierarchy.

# Node.js Syllabus:

1. **Introduction to Node.js:**
    - What is Node.js and its uses
    - Event-driven, non-blocking I/O model
    - Node.js vs. Browser JavaScript
    - Setting up the Node.js environment
    - Running Node.js scripts and REPL (Read-Eval-Print Loop)
    - Overview of Node.js architecture and single-threaded nature
2. **JavaScript Fundamentals for Node.js:**
    - Variables, Data Types, and Control Structures

- Functions, Closures, and Scope
- Understanding the Event Loop
- Asynchronous Programming: Callbacks, Promises, and Async/Await
- Error Handling and Exception Management

3. **Node.js Modules and Package Management:**
    - Understanding CommonJS and ES Modules
    - Creating and Importing Custom Modules
    - Using npm (Node Package Manager) and yarn
    - Managing Dependencies: package.json and package-lock.json
    - Semantic Versioning and Dependency Updates
4. **File System and Path Operations:**
    - Reading and Writing Files
    - Working with Directories
    - Using the Path Module
    - Streams and Buffers for File Handling
    - Watching for File Changes with fs.watch

5. **HTTP and Web Servers:**
    - Creating an HTTP Server
    - Handling Requests and Responses
    - Understanding HTTP Methods and Status Codes
    - Building RESTful APIs
    - Routing Basics and URL Parameters
6. **Working with Databases:**
    - Connecting to Databases (e.g., MongoDB)
    - CRUD Operations and Query Building
    - Database Transactions and Indexing
    - Connection Pooling and Optimizations
7. **Authentication and Authorization:**
    - Implementing Authentication Strategies (e.g., Passport.js)
    - Managing User Sessions and Cookies
    - JWT (JSON Web Tokens) for Stateless Authentication
    - Role-Based Access Control

- OAuth2.0 and Third-Party Authentication

8. **Testing and Debugging:**
    - Writing Unit Tests and Integration Tests
    - Using Testing Frameworks (e.g., Jest)
    - Debugging Techniques: Node.js Debugger and VS Code Integration
    - Writing Mock Services and End-to-End Tests
9. **Performance Optimization and Security:**
    - Profiling and Monitoring Applications with tools like PM2 and New Relic
    - Implementing Security Best Practices: Helmet.js, CORS, and Rate Limiting
    - Securing Sensitive Data with Environment Variables
    - Handling Errors, Exceptions, and Graceful Shutdowns

## NestJS Framework Syllabus:

1. **Introduction to NestJS:**
    - Philosophy and Architecture of NestJS
    - Comparison with other Node.js Frameworks
    - Setting up a NestJS Project with the CLI
    - Overview of Monolithic and Microservice Architectures in NestJS
2. **Core Concepts:**
    - Modules and Modular Design
    - Controllers and Routing (Request-Response Lifecycle)
    - Providers, Services, and Dependency Injection
    - Using Lifecycle Events and Application Lifecycle Hooks

3. **Middleware, Guards, and Interceptors:**
    - Creating and Using Middleware for Request Handling
    - Implementing Guards for Authorization
    - Using Interceptors for Request/Response Transformation
    - Extending Core Features with Custom Decorators
4. **Data Persistence:**
    - Integrating with Databases using Mongoose
    - Creating and Using Entities and Repositories
    - Data Transfer Objects (DTOs) and Validation
    - Handling Database Transactions
5. **Validation and Error Handling:**
    - Validating Requests with Pipes (Class Validator and Class Transformer)
    - Custom Exception Filters for Advanced Error Handling
    - Global Validation and Error Management
6. **Authentication and Authorization:**
    - Implementing JWT Authentication with Guards
    - Passport.js Strategies for OAuth2.0 and Social Logins
    - Role-Based Access Control (RBAC) and Permissions
    - Securing Endpoints with Guards and Interceptors
7. **Microservices Architecture:**
    - Building Microservices in NestJS
    - Using Message Brokers (e.g., Kafka) and Patterns
    - Managing Inter-Service Communication with Transport Layers
    - Service Discovery and Load Balancing
8. **Real-Time Applications:**
    - Setting up WebSocket Gateways and Events
    - Handling Real-Time Notifications and Messaging
    - Integrating Third-Party APIs for Real-Time Data Feeds
9. **Testing in NestJS:**
    - Writing Unit Tests for Controllers, Services, and Modules
    - End-to-End Tests with Test Utilities
    - Mocking Dependencies for Isolated Testing
    - Using Testing Frameworks like Jest and Supertest

- Automating Tests with CI/CD Pipelines

10. **API Documentation**
    - Document the API's with Swagger (@nestjs/swagger).
    - Versioning APIs.
11. **Deployment and Best Practices:**
    - Configuring Application Environments with ConfigModule
    - Logging and Monitoring with NestJS Logger and External Tools
    - Implementing Application Security (Helmet.js, Rate Limiting)
    - Dockerizing NestJS Applications for Deployment
    - Performance Optimization and Scaling Applications

**MongoDB Syllabus**

**1. Introduction to MongoDB**

- Overview of NoSQL Databases
- Features and Benefits of MongoDB
- MongoDB Architecture (Documents, Collections)

**2. Installation and Setup**

- Installing MongoDB on Windows/macOS/Linux
- Working with MongoDB Shell & Compass
- Introduction to MongoDB Atlas (Cloud)

**3. Basic CRUD Operations**

- Creating and Dropping Databases/Collections
- Insert, Read, Update, Delete Operations (CRUD)
- Querying with Filters and Projections

**4. Data Modeling**

- Schema Design Approaches (Embedding vs. Referencing)
- Understanding BSON Data Types
- Best Practices for Data Structuring

**5. Indexing**

- Types of Indexes (Single Field, Compound, Multikey, Text, Geospatial)
- Creating and Using Indexes
- Performance Optimization with Indexes

**6. Aggregation Framework**

- Aggregation Pipeline Stages ($match, $group, $project)
- Using Operators ($sum, $avg, $sort, $limit)
- Common Use Cases of Aggregation

**7. Relationships in MongoDB**

- One-to-One, One-to-Many, Many-to-Many Relationships

- Lookup Operations with $lookup

**8. Working with Arrays**

- Array Operators ($push, $addToSet, $pop)
- Querying Arrays ($elemMatch, $size, $all)
- Unwinding Arrays with $unwind

**9. Transactions in MongoDB**

- Introduction to ACID Transactions
- Multi-Document Transactions
- Use Cases and Limitations

**10. Data Validation**

- Schema Validation Rules
- Using JSON Schema for Data Validation
- Enforcing Data Integrity

**11. Replication and Sharding**

- Overview of Replication (Primary-Secondary)
- Setting Up Replica Sets
- Introduction to Sharding for Horizontal Scaling

**12. Backup and Restore**

- Taking Backups with mongodump
- Restoring Databases with mongorestore
- Backup Strategies and Best Practices

**13. Security in MongoDB**

- Role-Based Access Control (RBAC)
- Authentication and Authorization
- Security Best Practices

**14. Performance Optimization**

- Profiling Queries with explain()
- Analyzing Query Execution Plans
- Improving Read and Write Performance

**15. MongoDB with Programming Languages**

- Connecting MongoDB with Node.js,
- Writing Queries and CRUD Operations
- Handling Errors and Connection Pools

**16. Working with MongoDB Atlas**

- Setting Up and Managing Cloud Clusters
- Monitoring and Alerts
- Scaling and Backups in Atlas

**GIT & Version Control**

**Module 1: Introduction to Version Control and Git**

- **Description**: Understand the concept of version control and why it is essential for modern software development. This module introduces Git, its role in the development lifecycle, and how it compares to other version control systems.
    - Topics:
        - ♣ What is Version Control?
        - ♣ Introduction to Git
        - ♣ Centralized vs. Distributed Version Control
        - ♣ Installing Git & Setting Up Your First Repository

**Module 2: Git Basics – Essential Commands and Workflows**

- **Description**: Learn the fundamental Git commands necessary to interact with repositories. You'll be introduced to common workflows that help keep your codebase organized.
    - Topics:
        - ♣ Cloning Repositories
        - ♣ Creating and Switching Branches
        - ♣ Staging Changes (git add)
        - ♣ Committing Changes (git commit)
        - ♣ Viewing History (git log)
        - ♣ Managing Remote Repositories (git push, git pull)

**Module 3: Branching and Merging**

- **Description**: Explore branching strategies in Git, enabling you to develop features independently and then merge changes seamlessly into the main codebase.
    - Topics:
        - ♣ Branching Basics
        - ♣ Merging Branches (git merge)
        - ♣ Resolving Merge Conflicts
        - ♣ Rebasing (git rebase)
        - ♣ Cherry-Picking Commits

**Module 4: Collaboration with Git – Working with Teams**

- **Description**: Understand how Git facilitates collaboration in team-based projects, including best practices for managing pull requests, reviewing code, and syncing changes across different contributors.
    - Topics:
        - ♣ Forking Repositories
        - ♣ Creating and Managing Pull Requests
        - ♣ Reviewing Code and Resolving Conflicts
        - ♣ Using Git in Team-based Workflows (e.g., GitLab, GitHub Flow)

**Module 5: Advanced Git Concepts**

- **Description**: Deep dive into more advanced Git features that will help you manage complex project workflows, track changes over time, and ensure your codebase is always in a clean state.
    - Topics:
        - ♣ Stashing Changes (git stash)
        - ♣ Rewriting History (git rebase, git filter-branch)
        - ♣ Tags and Releases
        - ♣ Submodules and Subtrees

**Module 6: Git Best Practices and Troubleshooting**

- **Description**: Learn how to avoid common pitfalls and adopt Git best practices to ensure efficient and smooth development. This module includes tips for troubleshooting and maintaining a clean history.
    - Topics:
        - ♣ Git Commit Message Conventions
        - ♣ Maintaining a Clean Git History
        - ♣ Undoing Changes (git revert, git reset)
        - ♣ Fixing Broken History (git fsck, git reflog)
        - ♣ Using Git with CI/CD Pipelines

**Python**

**Python Introduction and Setting up the Environment**

- Introduction to Programming
- Why Python for Data Science?
- Different Python IDEs
- Downloading and Setting up the Python Environment

**Python Basic Syntax and Data Types**

- Python Input and Output Operations
- Comments
- Variables, Rules for Naming Variables
- Basic Data Types in Python
- Typecasting in Python

**Operators in Python**

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Identity Operators
- Membership Operators
- Bitwise Operators

**Strings in Python**

- Creating Strings
- String Formatting
- Indexing
- Slicing
- String Methods

**Lists**

- Creating Lists
- Properties of Lists
- List Indexing
- List Slicing
- List of Lists
- List Methods
- Adding, Updating & Removing Elements from Lists

**Tuples**

- Syntax to Create Tuples
- Tuple Properties
- Indexing on Tuples
- Slicing on Tuples
- Tuple Methods

**Sets**

- Syntax for Creating Sets
- Updating Sets
- Set Operations and Methods
- Difference between Sets, Lists, and Tuples

**Dictionaries**

- Syntax for Creating Dictionaries
- Storing Data in Dictionaries
- Dictionaries Keys and Values
- Accessing Elements of Dictionaries
- Dictionary Methods

**Python Conditional Statements**

- Setting Logic with Conditional Statements
- If Statements
- If-Else Statements
- If-Elif-Else Statements

**Loops in Python**

- Iterating with Python Loops
- While Loop
- For Loop

- Range
- Break
- Continue
- Pass
- Enumerate
- Zip
- Assert

**List and Dictionaries Comprehension**

- Why List Comprehension
- The Syntax for List Comprehension
- The Syntax for Dict Comprehension

**Functions**

- What are Functions
- Modularity and Code Reusability
- Creating Functions
- Calling Functions
- Passing Arguments
- Positional Arguments
- Keyword Arguments
- Variable-Length Arguments (*args)
- Variable Keyword Length Arguments (**kargs)
- Return Keyword in Python
- Passing Function as an Argument
- Passing Function in Return
- Global and Local Variables
- Recursion

**Anonymous Function**

- Lambda
- Lambda with Filter
- Lambda with Map
- Lambda with Reduce

**Generators**

- Creating and Using Generators

**Modules**

- Creating Modules
- Importing Functions from a Different Module
- Importing Variables from Different Modules
- Python Built-In Modules

**Packages – Data Analysis**

- Creating Packages
- Importing Modules from the Package
- Different Ways of Importing Modules and Packages
- Working on NumPy, Pandas, and Matplotlib

**Exceptions and Error Handling**

- Syntax Errors
- Logical Errors

- Handling Errors using Try, Except, and Finally

**Classes and Objects (OOPS)**

- Creating Classes & Objects
- Attributes and Methods
- Understanding init Constructor Method
- Class and Instance Attributes
- Different Types of Methods
- Instance Methods
- Class Methods
- Static Methods
- Inheritance
- Creating Child and Parent Class
- Overriding Parent Methods
- The super() Function
- Understanding Types of Inheritance
    - Single Inheritance
    - Multiple Inheritance
    - Multilevel Inheritance
- Polymorphism
- Operator Overloading

**Date and Time**

- Date Module
- Time Module
- DateTime Module
- Time Delta
- Formatting Date and Time
- strftime()
- strptime()

**Regex**

- Understanding the Use of Regex
- re.search()
- re.compile()
- re.find()
- re.split()
- re.sub()
- Meta Characters and Their Use

**Files**

- Opening Files
- Opening Different File Types
- Read, Write, Close Files
- Opening Files in Different Modes

**Machine Learning**

**Introduction to Machine Learning**

- What is Machine Learning?
- Types of Machine Learning: Supervised, Unsupervised, Reinforcement
- Applications of Machine Learning

**Programming Basics**

- Introduction to Python for Machine Learning
- Libraries: NumPy, Pandas, Matplotlib, Seaborn
- Working with Jupyter Notebooks

**Data Preprocessing**

- Understanding Data: Cleaning, Handling Missing Values
- Feature Scaling: Standardization and Normalization
- Encoding Categorical Variables: One-Hot Encoding, Label Encoding
- Data Splitting: Train/Test/Validation Sets

**Exploratory Data Analysis (EDA)**

- Descriptive Statistics
- Data Visualization
- Correlation and Covariance

**Supervised Learning - Basics**

- Linear Regression
- Logistic Regression
- Evaluation Metrics: MSE, RMSE, Accuracy, Precision, Recall, F1-Score

**Unsupervised Learning - Basics**

- K-Means Clustering
- Principal Component Analysis (PCA)

**Advanced Topics in Machine Learning**

**Advanced Supervised Learning**

- Decision Trees
- Random Forest
- Gradient Boosting Algorithms: XGBoost, LightGBM, CatBoost

**Unsupervised Learning**

- Hierarchical Clustering
- DBSCAN
- Anomaly Detection

**Neural Networks and Deep Learning**

- Introduction to Neural Networks
- Activation Functions
- Training Neural Networks: Forward Propagation, Backpropagation
- Deep Learning Frameworks: TensorFlow, PyTorch

**Natural Language Processing (NLP)**

- Text Preprocessing: Tokenization, Lemmatization, Stemming
- Sentiment Analysis and Text Classification

## Core LLM Concepts

**LLM Architecture**

- Evolution from encoder-decoder to decoder-only architectures (e.g., GPT).
- Tokenization: Converting text to numbers.
- Attention Mechanism: How self-attention helps with long-range dependencies.

- Sampling Techniques: Text generation approaches like greedy search and nucleus sampling.

**Pre-training Models**

- Data Preparation: Cleaning and tokenizing large datasets.
- Distributed Training: Efficient use of multiple GPUs for model training.
- Training Optimization: Adaptive learning rates, gradient clipping, optimizers like AdamW.
- Monitoring: Tracking training progress.

**Post-training Datasets**

- Chat Templates: Different formats for storing and interacting with data.
- Synthetic Data Generation: Creating instruction-response pairs using models.
- Quality Filtering: Removing low-quality data using automated techniques.

**Supervised Fine-Tuning (SFT)**

- Fine-tuning: Adjusting model parameters on specific tasks.
- Training Parameters: Learning rate, batch size, etc.
- Distributed Training: Scaling across GPUs.

**The LLM Engineer - Building and Deploying Applications**

1. **Running LLMs**
   - APIs: Using models from OpenAI, Hugging Face, etc.
   - Prompt Engineering: Structuring inputs for better outputs.
2. **Building a Vector Storage**
   - Ingesting Documents: Loading and processing various document types.
   - Vector Databases: Storing and retrieving text embeddings.
3. **Retrieval Augmented Generation (RAG)**
   - Orchestrators: Connecting LLMs to external tools.
   - Evaluation: Measuring retrieval and generation quality.
4. **Advanced RAG**
   - Query Construction: Translating user instructions to structured queries.
   - Program LLMs: Automating prompt optimization.
5. **Deploying LLMs**
   - Local & Server Deployment: Running models on local servers or the cloud.

- Edge Deployment: Deploying models on devices like smartphones or web browsers.

## Kafka Syllabus

### 1. Introduction to Apache Kafka
- What is Apache Kafka?
- Kafka vs. Traditional Messaging Systems
- Core Kafka Components: Topics, Producers, Consumers, Brokers
- Use Cases of Kafka (Real-Time Analytics, Event Streaming)

### 2. Kafka Architecture Overview
- Kafka Clusters and Brokers
- Partitions and Replication
- Message Retention and Offsets
- Producers and Consumers Overview

### 3. Installing and Setting Up Kafka
- Installing Kafka on Windows/macOS/Linux
- Zookeeper and Kafka Setup
- Configuring Brokers and Partitions
- Running Kafka Commands (Creating Topics, Producing and Consuming Messages)

### 4. Kafka Topics and Partitions
- Creating and Managing Topics
- Partitioning Strategies
- Replication Factor and Fault Tolerance
- Data Retention Policies

### 5. Producers in Kafka
- Understanding Kafka Producers
- Producer Acknowledgments and Delivery Guarantees
- Configuring Producers (Batch Size, Compression, Retries)
- Key-Value Pair Messaging

### 6. Consumers in Kafka
- Kafka Consumer Groups and Offsets
- Polling vs. Push-Based Consumption
- Auto vs. Manual Offset Management
- Consumer Rebalancing and Parallelism

### 7. Kafka Internals
- Log Segments and Indexing
- ISR (In-Sync Replicas) and Leader Elections
- Kafka Controller and Quotas
- Log Compaction and Cleanup Policies

**8. Kafka Streams and Processing**

- Introduction to Kafka Streams API
- Stateless vs. Stateful Processing
- Windowing, Joins, and Aggregations
- Processing Guarantees (At-Least-Once, Exactly-Once)

**9. Kafka Connect (Data Integration)**

- Introduction to Kafka Connect
- Source and Sink Connectors
- Common Connectors (JDBC, Elasticsearch, S3)
- Running and Monitoring Connectors

**10. Schema Registry and Serialization**

- Avro, JSON, and Protobuf Formats
- Role of Schema Registry
- Schema Evolution and Compatibility

**11. Security in Kafka**

- Authentication (SSL/SASL)
- Authorization (ACLs and Role-Based Access Control)
- Data Encryption in Transit and at Rest
- Best Security Practices

**12. Monitoring and Logging**

- Kafka Metrics (JMX Monitoring)
- Tools: Prometheus, Grafana, Confluent Control Center
- Analyzing Kafka Logs for Debugging
- Performance Tuning Techniques

**13. Kafka Reliability and Scaling**

- Replication and High Availability
- Load Balancing and Partition Distribution
- Handling Failures and Recovery Strategies

**14. Kafka Performance Optimization**

- Producer and Consumer Tuning
- Partitioning Strategies for Scalability
- Broker Configuration Optimizations
- Best Practices for Throughput and Latency

**15. Kafka with Microservices**

- Event-Driven Microservices Architecture
- Using Kafka with Spring Boot
- Building Resilient Microservices with Kafka
- Handling Eventual Consistency

**16. Kafka in Cloud Environments**

- Running Kafka on AWS, Azure, GCP
- Using Managed Kafka Services (Confluent Cloud, AWS MSK)
- Deployment Strategies in Cloud

**React Native**

**React Core Concepts**

- JSX
- Functional and Class Components
- Props and State
- Lifecycle Methods (Class Components)
- Hooks

**Rendering and Event Handling in React**

- Conditional Rendering
- Lists and Keys
- Event Handling

**Forms and Styling in React**

- Forms
- Component Styling

**Routing and Error Management**

- React Router
- Error Handling

**Performance and Optimization**

- Lazy Loading and Code Splitting

**Advanced React Concepts**

- Custom Hooks
- Context API with Reducer

**React Native Specifics**

**Core Concepts**

- Environment Setup
- Core Components (View, Text, Image, etc.)
- Flexbox Layout
- React Navigation
- Styling in React Native

**Device Features**

- Native Device Features (Camera, Geolocation, Notifications, etc.)
- Networking (HTTP Requests)
- State Management (Context, Redux)
- Storage (AsyncStorage etc..)

**Additional Functionalities**

- Gesture Handling (React Native Gesture Handler, Reanimated)
- Push Notifications (Firebase, React Native Push Notifications)
- Deep Linking
- App Permissions
- Theming (Light/Dark Modes)

- Web Integration (React Native Web)

**Advanced Topics**

- Native Modules (iOS/Android)
- Secure Storage (Encrypted Storage, Keychain)
- WebSockets and Real-Time Communication
- Animations (Animated API, Reanimated)
- Testing (Unit Testing)

**Flutter**

**Flutter Basics**

- Introduction to Flutter
- Flutter Architecture Application
- Setting Up Android Studio for Flutter Development
- Setting Up Flutter for Application Development
- Flutter Development on Ubuntu

**Flutter Widgets**

- Introduction to Flutter Widgets
- Stateful vs Stateless Widgets
- Container Class in Flutter
- Scaffold Class in Flutter
- MaterialApp Class in Flutter
- BottomNavigationBar Widget in Flutter
- ClipRRect Widget in Flutter
- Drawer Widget in Flutter
- ClipRect Widget in Flutter
- Opacity Widget in Flutter
- RotatedBox Widget in Flutter
- RichText Widget in Flutter
- OctoImage Widget in Flutter
- AppBar Widget in Flutter
- Visit Flutter-Widgets to view more widgets

**Flutter UI Components**

- Flutter – Carousel Slider
- Flutter – Staggered Grid View
- Flutter – Circular & Linear Progress Indicators
- Alert Dialog Box in Flutter
- Flutter – Dialogs
- Icon Class in Flutter
- Expanded Class in Flutter
- Analog Clock in Flutter
- Flutter – Handling Videos
- Flutter – Expansion Tile Card
- Flutter – Tabs
- Flutter – Horizontal List
- Flutter – Working with Charts
- Flutter – Convex BottomBar

- Flutter – Slidable

**Flutter Design & Animations**

- Customizing Fonts in Flutter
- Flutter – Skeleton Text
- Flutter – Animation in Route Transition
- Flutter – Ripple Effect
- Flutter – UI Orientation
- Flutter – Physics Simulation in Animation
- Flutter – Themes
- Flutter – Radial Hero Animation
- Flutter – PhotoHero Class
- Flutter – Hinge Animation
- Flutter – Lottie Animation
- Flutter – Using Google Fonts
- Flutter – Auto-Size Text
- Flutter – Rotate Transition
- Flutter – Lazy Loader
- Flutter – Animated Splash Screen
- Flutter – Shimmer
- Rive Animations in Flutter
- ProgressIndicator in Flutter

**Flutter Forms & Gestures**

- Form Validation in Flutter
- Designing a Form Submission Page in Flutter
- Flutter – Gestures

**Flutter Navigation & Routing**

- URLs in Flutter
- Multi-Page Applications in Flutter
- Routes and Navigator in Flutter
- Retrieve Data From TextFields in Flutter
- Flutter – WebSockets
- Flutter – Avoiding Jank
- Flutter – Named Routes
- Flutter – Updating Data on the Internet
- Flutter – Fetching Data From the Internet
- Flutter – Deleting Data on the Internet
- Flutter – Sending Data to the Internet
- Flutter – Arguments in Named Routes
- Flutter – Return Data from Screen
- Flutter – Send Data to Screen
- Flutter – Send Data to the Screen using RouteSettings

**Flutter Accessing Device**

- Gallery Access in Flutter
- Camera Access in Flutter
- Background Local Notifications in Flutter
- Restrict Landscape Mode in Flutter

**Flutter Advanced Concepts**

- Flutter – Read and Write Data on Firebase
- Mail and SMS in Flutter
- Making Calls in Flutter
- FAB – Speed Dial in Flutter
- Flutter – Wakelock
- Implementing REST API in Flutter
- HTTP GET Response in Flutter

**Flutter flow (Tool)**

**Visual App Builder**
- Drag-and-drop UI creation
- Flexible layouts and themes

**Widgets and Customization**
- Pre-built UI components
- Custom widgets and animations

**Firebase and Backend Integration**
- Authentication
- Firestore Database
- Firebase Storage
- API Integration

**Navigation and Routing**
- Screen navigation setup
- Actions and events
- Conditional and deep linking

**State Management**
- Local and global state
- Data binding

**Custom Code**
- Custom actions and logic
- Modify exported Flutter code

**Real-time Preview**
- Test app in real-time
- Preview across devices

**Exporting Code**
- Export production-ready Flutter code
- Further customization with code

**User Roles and Access Control**
- Role-based access
- Show/hide content based on roles

**App Deployment**
- iOS and Android deployment
- Progressive web app (PWA)

**Prototyping**
- Quickly build app prototypes

# Elasticsearch

**Introduction and Fundamentals**

- What is Elasticsearch?
- How Elasticsearch Works (Distributed Search Engine, Lucene)
- Key Concepts
- Documents
- Indexes
- Shards and Replicas
- Clusters
- Use Cases for Elasticsearch
- Log Analysis
- Full-Text Search
- Site Search
- Analytics

**Installation and Setup**

- Downloading and Installing Elasticsearch
- Starting and Managing an Elasticsearch Cluster
- Basic Configuration Options

**Data Modeling and Mapping**

- Creating Indexes and Defining Document Structures
- Mapping Data Fields
- Text
- Keyword
- Number
- Date
- Understanding Data Types and Their Impact on Search

**Indexing Documents**

- Sending Data to Elasticsearch Using the REST API
- JSON Document Format
- Bulk Indexing for Efficient Data Loading

**Basic Querying**

- Simple Search Queries
- Keyword Matching
- Phrase Matching
- Filtering Data Based on Specific Field Values
- Using Query String Syntax

**Advanced Search Features**

- Wildcards and Regular Expressions
- Fuzzy Matching
- Faceting and Aggregations
- Geo-Spatial Search

**Kibana and Visualization**

- Introduction to Kibana
- Creating Dashboards and Visualizations
- Exploring Data Using Kibana Discover