Curriculum

**.Net**

1. Introduction to .NET Framework: Overview of .NET, its history, Architecture and its role in modern software development.
2. .NET Common Language Runtime (CLR): Understanding how the CLR executes .NET applications and manages memory, exceptions, and security.
3. FCL
4. GC (Garbage collector)
5. CTS, CLS
6. Visual Studio IDE: Introduction to Visual Studio, the integrated development environment for .NET, and its key features for coding, debugging, and testing.
7. .NET Assemblies and the GAC: Understanding the concept of assemblies, how they are deployed, and the Global Assembly Cache (GAC).

**Visual Studio**

1. Introduction to Visual Studio IDE

- Overview of Visual Studio IDE

- History and versions

- Supported languages and platforms

2. Installing Visual Studio

- System requirements

- Installation options

- Configuring the development environment

3. Getting Started with Projects

- Creating a new project

- Opening existing projects

- Project templates and configurations

4. Understanding the IDE Layout

- Solution Explorer

- Code Editor

- Toolbox and Solution Navigator

- Properties Window

5. Navigating and Searching in Visual Studio

- Using the navigation bar and breadcrumbs

- Find and Replace in code

- Go to Definition and Find All References

6. Debugging in Visual Studio

- Setting breakpoints

- Step over, into, and out of code

- Watch and Locals windows

- Debugging tools and features

7. Version Control with Visual Studio

- Integrating Git repositories

- Cloning, branching, and merging

- Resolving conflicts

8. IntelliSense and Code Snippets

- Understanding IntelliSense

- Utilizing code snippets for faster coding

9. Refactoring and Code Analysis

- Renaming identifiers

- Extracting methods and properties

- Using code analysis to improve code quality

10. Project Configuration and Publishing

- Configuring project settings

- Publishing and deploying applications

11. Customizing Visual Studio

- Themes and fonts

- Extensions and add-ons

- Keyboard shortcuts customization

12. Testing and Test Explorer

- Writing unit tests in Visual Studio

- Running and debugging tests using Test Explorer

13. Performance and Profiling

- Performance analysis tools

- Profiling to optimize code performance

14. Azure Integration (Optional, if applicable)

- Deploying to Azure

- Azure DevOps integration

15. Tips and Tricks for Productivity

- Time-saving shortcuts and features

- Using snippets and templates effectively

16. Troubleshooting and Error Handling

- Common issues and their solutions

- Diagnosing and fixing errors

17. Visual Studio Code (Optional, as a comparison)

- Overview of Visual Studio Code

- Key differences between Visual Studio and Visual Studio Code

**C#**

1. Introduction to C#: Overview of C# language, its history, and its role in the .NET ecosystem.

2. C# Basics: Covering the fundamentals, data types, variables, operators, and basic input/output.

3. Control Flow: Understanding decision-making with if-else statements, switch statements, and loops (while, for, foreach).

4. Methods and Functions: Defining and calling methods, passing parameters, and returning values.

5. Arrays and Collections: Working with one-dimensional and multi-dimensional arrays, as well as collections like lists and dictionaries.

6. Object-Oriented Programming (OOP): Understanding the core principles of OOP, including classes, objects, inheritance, encapsulation, and polymorphism.

7. Constructors and Destructors: Creating and initializing objects using constructors, and managing resources with destructors.

8. Properties: Using properties to control access to class fields.

9. Exception Handling: Managing errors and exceptions with try-catch blocks.

10. File I/O: Reading from and writing to files.

11. Generics: Understanding generic classes, methods, and collections.

12. LINQ (Language-Integrated Query): Querying data using LINQ expressions.

13. Delegates and Events: Working with delegates and events for event-driven programming.

14. Asynchronous Programming: Understanding async and await for handling asynchronous operations.

15. C# Reflection: Learning about runtime type information and dynamic invocation.

16. C# Attributes: Using attributes to add metadata to types and members.

17. C# Design Patterns: Exploring common design patterns like Singleton, Factory, and Observer.

18. C# Memory Management: Understanding garbage collection and managing memory resources.

19. C# Best Practices: Learning coding conventions, performance optimization techniques, and clean code principles.

20. Unit Testing in C#: Introducing test-driven development (TDD) and writing unit tests with tools like MSTest or NUnit.

21. Introduction to ASP.NET Core: A brief introduction to web development with C# using the ASP.NET Core framework.

22. Introduction to Windows Forms or WPF: Learning about desktop application development with C# using Windows Forms or WPF.

23. Working with Databases: Connecting to databases, executing queries, and using Entity Framework or ADO.NET for data access.

24. Introduction to RESTful APIs: Creating and consuming RESTful APIs using C#.

25. C# and Multithreading: Understanding multithreading and synchronization in C#.

**Advance C#**

1. Delegates and Events:

- Understanding delegates and multicast delegates.

- Implementing custom events and event handlers.

- The role of delegates and events in event-driven programming.

2. Lambda Expressions and LINQ:

- Exploring lambda expressions and their benefits.

- Introduction to LINQ (Language Integrated Query) and its advantages.

- Querying data from various sources using LINQ.

3. Generics:

- Understanding generic types and methods.

- Creating generic classes and constraints.

- Benefits of using generics for type safety and code reusability.

4. Advanced OOP Concepts:

- Interfaces, abstract classes, and their use cases.

- Explicit interfaces implementation.

- Sealed classes and methods.

- Covariance and contravariance.

5. Advanced Exception Handling:

- Custom exception classes and exception handling strategies.

- Using the `finally` block and resource management.

- Handling and logging unhandled exceptions.

6. Multithreading and Asynchronous Programming:

- Synchronization and thread safety.

- Working with threads and thread pools.

- Asynchronous programming using async/await.

7. Advanced LINQ and PLINQ:

- Advanced LINQ operations and operators.

- Parallel LINQ (PLINQ) for data parallelism.

- Asynchronous LINQ queries.

8. Reflection and Attributes:

- Understanding reflection and its applications.

- Working with custom attributes and metadata.

- Using reflection to inspect and manipulate types and objects.

9. Design Patterns:

- Exploring common design patterns (e.g., Singleton, Factory, Observer).

- Applying design patterns to solve real-world problems.

10. Advanced File I/O and Serialization:

- Working with binary file formats and custom serialization.

- XML and JSON serialization using DataContractSerializer and JSON.NET.

- Using the File System Watcher for monitoring file changes.

11. Advanced Topics in Collections:

- Custom collection classes.

- Immutable collections.

- Working with large datasets using Big-O notation.

12. C# 9 and Future Features:

- New features introduced in C# 9 and later versions.

- Pattern matching enhancements.

- Records and init-only properties.

13. Unit Testing and Test-Driven Development (TDD):

- Advanced unit testing techniques.

- Test-driven development workflow.

- Using mocking frameworks for testing.

14. ASP.NET Core and Web API:

- Building advanced web applications using ASP.NET Core.

- Creating RESTful APIs with ASP.NET Core Web API.

- Securing APIs with JWT (JSON Web Tokens) authentication.

15. Entity Framework Core:

- Advanced querying using Entity Framework Core.

- Data migrations and database seeding.

- Performance optimization with EF Core.

**SOLD Principles**

1. Introduction to SOLID Principles

- Overview of SOLID principles: Single Responsibility Principle, Open/Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, and Dependency Inversion Principle.

- Importance of SOLID principles in writing maintainable and scalable code.

2. Single Responsibility Principle (SRP)

- Definition and explanation of SRP: A class should have only one reason to change.

- Real-world example: Implementing a logging service that handles logging responsibilities without mixing it with other functionalities.

3. Open/Closed Principle (OCP)

- Definition and explanation of OCP: Software entities should be open for extension but closed for modification.

- Real-world example: Designing a plugin system that allows adding new features without modifying existing code.

4. Liskov Substitution Principle (LSP)

- Definition and explanation of LSP: Objects of a superclass should be replaceable with objects of its subclasses without affecting correctness.

- Real-world example: Modeling different shapes (circle, square, rectangle) with a common interface and ensuring that all shapes can be used interchangeably.

5. Interface Segregation Principle (ISP)

- Definition and explanation of ISP: Clients should not be forced to depend on interfaces they do not use.

- Real-world example: Designing an interface for a printer that is specific to the required functionalities, allowing different types of printers to implement only relevant methods.

6. Dependency Inversion Principle (DIP)

- Definition and explanation of DIP: High-level modules should not depend on low-level modules; both should depend on abstractions.

- Real-world example: Implementing a data access layer with dependency injection to decouple the business logic from the data access implementation.

7. Applying SOLID Principles Together

- Demonstration of how the SOLID principles work together to create a flexible, maintainable, and scalable codebase.

- Example: Building a real-world application while adhering to SOLID principles and showcasing the benefits.

8. SOLID in Design Patterns

- Exploring how SOLID principles align with various design patterns like Factory Method, Strategy, and Dependency Injection.

- Real-world examples: Applying SOLID principles in combination with design patterns for more robust solutions.

9. Refactoring Legacy Code

- Techniques for refactoring legacy codebases to adhere to SOLID principles.

- Real-world example: Step-by-step refactoring of a legacy application to make it more maintainable and extensible.

10. Best Practices and Common Pitfalls

- Guidelines for effectively using SOLID principles in real-world projects.

- Common pitfalls to avoid when applying SOLID principles and their solutions.

11. Hands-On Projects

- Practical exercises and projects where participants can apply SOLID principles to solve real-world problems.

**SQL**

1. Introduction to Databases: Overview of databases, their types (relational, NoSQL, etc.), and their role in software development.

2. SQL Basics: Introduction to SQL (Structured Query Language), its purpose, and its syntax.

3. Creating and Managing Databases: How to create databases, tables, and manage their structure.

4. Basic CRUD Operations: Understanding how to perform Create, Read, Update, and Delete operations on database records.

5. SQL Constraints: Learning about constraints like Primary Key, Foreign Key, Unique, NOT NULL, etc., and their role in maintaining data integrity.

6. Filtering Data with WHERE: Using the WHERE clause to filter data based on specific conditions.

7. Sorting Data with ORDER BY: Sorting query results in ascending or descending order.

8. Querying Multiple Tables with JOIN: Understanding how to retrieve data from multiple tables using various types of joins (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN).

9. Aggregating Data with GROUP BY: Using GROUP BY to perform aggregate functions like SUM, AVG, COUNT, etc.

10. Subqueries: Learning about subqueries and their usage in SQL queries.

11. Working with Views: Creating and using views to simplify complex queries and provide a layer of abstraction.

12. Modifying Data with INSERT, UPDATE, DELETE: Performing data modification operations on the database.

13. SQL Functions: Exploring built-in SQL functions like String functions, Date functions, and Numeric functions.

14. Transactions: Understanding transactions, their properties (ACID), and how to manage them in SQL.

15. Indexes: Learning about indexes and their importance in optimizing query performance.

16. Data Integrity: Ensuring data integrity through constraints, triggers, and referential integrity.

17. Stored Procedures: Creating and using stored procedures for modular and efficient SQL code.

18. User-Defined Functions: Building custom functions for specific requirements.

19. SQL Performance Tuning: Techniques for optimizing query performance and troubleshooting common performance issues.

20. Working with Large Databases: Strategies for handling large datasets efficiently.

21. Advanced SQL Concepts: Advanced topics like Window Functions, Common Table Expressions (CTEs), and Recursive Queries.

22. Introduction to NoSQL: Overview of NoSQL databases and when to consider using them.

23. Database Administration: Basic database administration tasks like backups, restores, and user management.

24. Data Modeling: Introduction to database design and data modeling principles.

25. Database Security: Best practices for securing databases and preventing unauthorized access.

**Ado.Net**

1. Introduction to ADO.NET

- Overview of ADO.NET and its purpose in .NET applications.

- Understanding the ADO.NET architecture (Data Providers, DataSets, DataReaders, etc.).

2. Connecting to a Database

- Establishing database connections using Connection objects.

- Configuring connection strings for different database providers (SQL Server, MySQL, Oracle, etc.).

3. Executing SQL Commands

- Using Command objects to execute SQL queries, stored procedures, and other commands.

- Parameterized queries to prevent SQL injection attacks.

4. Data Readers

- Working with DataReader objects for efficient read-only access to data.

- Understanding forward-only and connected data reading.

5. DataSets and DataTables

- Exploring DataSets and DataTables for disconnected data access.

- Populating DataSets and DataTables using DataAdapters.

6. DataAdapters

- Using DataAdapters to fill DataSets and DataTables from databases.

- Updating databases with changes made in the DataSets.

7. Data Binding

- Binding DataSets and DataTables to UI controls for display and manipulation.

- Handling data-bound control events.

8. Error Handling and Transactions

- Implementing error handling mechanisms in ADO.NET.

- Using transactions to ensure data consistency and integrity.

9. Stored Procedures and Parameterized Commands

- Utilizing stored procedures for improved database performance and security.

- Working with parameterized commands to call stored procedures.

10. Entity Framework (EF)

- An introduction to Entity Framework, a higher-level ORM framework for data access.

- Comparing EF with traditional ADO.NET data access.

11. LINQ to SQL

- Understanding LINQ to SQL and its integration with ADO.NET for database access.

- Writing LINQ to SQL queries and updating data.

12. Data Access Best Practices

- Performance optimization techniques for data access in ADO.NET.

- Design patterns for data access layer and separation of concerns.

13. Database Connection Pooling

- Understanding connection pooling and its impact on application performance.

- Configuring and managing connection pooling in ADO.NET.

14. Asynchronous Data Access

- Performing asynchronous data access operations in ADO.NET.

- Utilizing async/await to improve application responsiveness.

15. Working with XML Data

- Using ADO.NET to read and write XML data.

- XML integration with databases and web services.

16. Working with JSON Data

- Introduction to JSON serialization and deserialization in ADO.NET.

- Consuming and producing JSON data in .NET applications.

17. Data Security

- Techniques for securing sensitive data in ADO.NET applications.

- Implementing data access permissions and encryption.

18. Data Caching

- Caching data to improve performance and reduce database load.

- Utilizing caching strategies in ADO.NET applications.

**.NET Entity Framework**

1. Introduction to .NET Entity Framework

- Overview of ORM (Object-Relational Mapping) and its role in data access.

- Understanding the benefits of using Entity Framework in .NET projects.

2. Setting Up Entity Framework

- Installing Entity Framework and its prerequisites.

- Configuring database connections and DbContext.

3. Code-First Approach

- Creating database models using POCO (Plain Old CLR Object) classes.

- Defining relationships between entities using attributes or fluent API.

- Generating and managing database schema from code.

4. Database-First Approach

- Creating Entity Data Model (EDM) from an existing database.

- Performing CRUD operations using generated entities.

5. Querying with LINQ and Entity SQL

- Using LINQ (Language Integrated Query) to query the database with Entity Framework.

- Understanding Entity SQL and its differences from LINQ.

6. Working with DbContext

- Performing database operations using the DbContext class.

- Managing entity state and tracking changes.

7. Eager Loading and Lazy Loading

- Understanding the concepts of eager loading and lazy loading.

- Controlling loading behavior to optimize performance.

8. Migrations and Database Updates

- Using Entity Framework Migrations to manage database schema changes.

- Handling database updates and versioning.

9. Database Seeding and Initialization

- Seeding the database with initial data during application startup.

- Implementing database initialization strategies.

10. Transactions and Concurrency

- Managing transactions in Entity Framework.

- Handling concurrency conflicts.

11. Performance Optimization

- Identifying performance bottlenecks in Entity Framework applications.

- Implementing caching and other performance optimizations.

12. Advanced Topics

- Implementing inheritance in Entity Framework models.

- Handling complex relationships like many-to-many and one-to-one.

- Working with stored procedures and raw SQL queries.

13. Testing with Entity Framework

- Strategies for unit testing applications that use Entity Framework.

- Mocking DbContext and repositories for testing.

14. Security Considerations

- Best practices for securing Entity Framework applications against common vulnerabilities.

15. Real-world Examples and Best Practices

- Building practical applications using Entity Framework.

- Showcasing best practices for structuring and organizing Entity Framework projects.

16. Entity Framework Core (Optional)

- An introduction to Entity Framework Core and its key differences from the classic Entity Framework.

17. Q&A and Troubleshooting

- Addressing common issues and answering questions related to Entity Framework.

**.Net MVC framework and Web API**

1. Introduction to ASP.NET MVC

- Overview of ASP.NET MVC architecture and its advantages

- Comparison with other ASP.NET frameworks (Web Forms, ASP.NET Core)

2. Model-View-Controller (MVC) Pattern

- Understanding the MVC architectural pattern

- Role of Model, View, and Controller in ASP.NET MVC

3. Setting Up an ASP.NET MVC Project

- Creating a new MVC project in Visual Studio

- Project structure and organization

4. Controllers

- Creating controllers and actions

- Handling HTTP requests and routing

- Controller actions and return types

5. Models

- Creating models to represent data

- Data annotations and validation attributes

- Working with Entity Framework for database interaction

6. Views

- Creating and rendering views

- Razor syntax and view templates

- Working with layout pages and partial views

7. Razor Views and HTML Helpers

- Using Razor syntax to build dynamic views

- Understanding HTML Helpers and their role in view rendering

8. Data Binding and Model Binding

- Binding data from models to views

- Handling user input and form submissions

- Model binding and validation

9. Routing and URL Management

- Understanding ASP.NET MVC routing system

- Customizing route configuration

- Route constraints and attribute routing

10. Action Filters and Middleware

- Implementing action filters for cross-cutting concerns

- Global filters and their usage

- Introduction to Middleware in ASP.NET Core

11. Client-Side Development with JavaScript and jQuery

- Using JavaScript and jQuery in ASP.NET MVC applications

- Ajax and asynchronous calls

12. Authentication and Authorization

- Implementing authentication and authorization in ASP.NET MVC

- Role-based security and claims-based authentication

13. Error Handling and Logging

- Handling exceptions and errors in ASP.NET MVC

- Logging techniques for troubleshooting and monitoring

14. Bundling and Minification

- Optimizing client-side assets with bundling and minification

- Improving application performance

15. ASP.NET Web API

- Introduction to Web API and its integration with ASP.NET MVC

- Building RESTful APIs to serve data

16. Deployment and Hosting

- Preparing an ASP.NET MVC application for deployment

- Hosting options and considerations

17. Unit Testing in ASP.NET MVC

- Writing unit tests for controllers and actions

- Testing view rendering and model validation

18. Real-World Project

- Encourage participants to work on a real-world project using ASP.NET MVC to apply their knowledge and skills gained during the bootcamp.

**Advance Asp.net MVC**

1. Review of MVC Basics

- Recap of the Model-View-Controller architectural pattern and its implementation in ASP.NET MVC.

2. Advanced Routing

- Custom route configurations and route constraints.

- Attribute routing and route prefixes.

3. Action Results and Filters

- Exploring various ActionResult types (PartialViewResult, JsonResult, FileResult, etc.).

- Understanding Action Filters and their usage for cross-cutting concerns (e.g., authentication, logging).

4. Advanced Model Binding

- Handling complex models and nested objects in model binding.

- Custom model binders for non-standard data types.

5. Asynchronous Programming

- Using async/await for asynchronous controller actions and asynchronous data access.

- Handling concurrency and avoiding race conditions.

6. Advanced Views

- Working with Display and Editor Templates for reusable view components.

- Extending Razor syntax with custom helpers and tag helpers.

7. Dependency Injection and IoC Containers

- Implementing dependency injection in ASP.NET MVC.

- Using popular IoC containers like Unity, Autofac, or Ninject.

8. Authentication and Authorization

- Implementing custom authentication and authorization logic.

- Using ASP.NET Identity for user authentication and role-based authorization.

9. Caching Techniques

- Caching strategies with OutputCache and caching partial views.

- Leveraging in-memory caching and distributed caching (e.g., Redis).

10. Real-time Web Applications

- Introduction to SignalR for building real-time features in MVC applications.

11. Bundling and Minification

- Optimizing client-side resources with bundling and minification.

- Working with custom bundles and controlling cache versions.

12. Error Handling and Logging

- Global error handling with custom error pages.

- Implementing logging using frameworks like Serilog or NLog.

13. Testing in MVC

- Writing unit tests for controllers, views, and models.

- Integration testing with Selenium or other testing frameworks.

14. Performance Optimization

- Identifying performance bottlenecks and using profiling tools.

- Techniques for optimizing database queries and reducing latency.

15. Web API Integration

- Integrating ASP.NET Web API with MVC for building RESTful services.

16. Security Best Practices

- Protecting against common web application security vulnerabilities (e.g., XSS, CSRF).

- Secure coding practices for MVC applications.

17. Deployment and DevOps

- Preparing applications for deployment to different environments.

- Automating the deployment process with CI/CD pipelines.

18. Single Page Applications (SPA) with MVC

- Integrating front-end frameworks like Angular, React, or Vue.js with MVC.

19. Microservices and Containerization

- Exploring microservices architecture and containerization using Docker.

20. Advanced MVC Project

- A hands-on project where participants apply the knowledge gained to build a complex MVC application with various advanced features.

**Rest API with simple CRUD operations**

1. Introduction to RESTful APIs

- Understanding the basics of REST architecture

- HTTP methods (GET, POST, PUT, DELETE) and their significance

2. Setting up the Development Environment

- Installing Visual Studio and .NET SDK

- Creating a new .NET Web API project

3. Building a Simple CRUD API

- Creating models for data entities

- Implementing controllers for CRUD operations

- Testing the API using tools like Postman

4. Handling HTTP Requests and Responses

- Parsing and validating incoming data

- Formatting and serializing data in responses

5. Working with Entity Framework Core

- Setting up Entity Framework Core for data access

- Creating a database context and configuring models

- Using migrations to create and update the database schema

6. Retrieving Data with GET

- Implementing GET endpoints to fetch data from the database

- Supporting query parameters for filtering, sorting, and pagination

7. Creating Data with POST

- Implementing POST endpoints to add new data

- Validating and handling input data

8. Updating Data with PUT and PATCH

- Implementing PUT and PATCH endpoints to update existing data

- Understanding the difference between PUT and PATCH

9. Deleting Data with DELETE

- Implementing DELETE endpoints to remove data from the database

10. Error Handling and Validation

- Implementing custom error responses

- Performing input validation and handling validation errors

11. Securing the API

- Basic authentication and authorization using tokens

- Role-based access control (RBAC)

12. Versioning the API

- Managing API versions to support backward compatibility

13. Best Practices for API Design

- Following RESTful design principles

- Properly structuring the API endpoints and resource URLs

14. Testing and Debugging

- Writing unit tests for the API controllers

- Debugging techniques for identifying and resolving issues

15. Deployment and Hosting

- Deploying the API to different environments (local, staging, production)

- Hosting options for the API (IIS, Azure, Docker)

16. Performance Optimization

- Caching strategies to improve API performance

- Identifying and resolving performance bottlenecks

17. Real-world Project

- Integrating all concepts to build a complete CRUD API for a specific use case

18. Additional Topics (Optional)

- Handling file uploads and downloads

- Implementing search functionality with filters

- Implementing advanced authentication and security features

**.Net Core**

1. Introduction to .NET Core

- Overview of .NET Core and its benefits

- Comparison with the traditional .NET Framework

- Setting up the development environment with Visual Studio or Visual Studio Code

2. Building a Basic Web Application

- Creating a new .NET Core project

- Understanding the project structure and files

- Running the application and testing the initial setup

3. MVC Architecture in .NET Core

- Introduction to Model-View-Controller (MVC) pattern

- Creating models, views, and controllers

- Handling requests and responses in MVC

4. Creating the Database

- Setting up a database using Entity Framework Core

- Using migrations to create and update the database schema

5. Implementing Create (C) Operation

- Creating a form to input data

- Handling form submission to add records to the database

- Validating user input and displaying error messages

6. Implementing Read (R) Operation

- Retrieving data from the database

- Displaying data in views using Razor syntax

- Implementing pagination for large datasets

7. Implementing Update (U) Operation

- Creating an edit form to update existing records

- Handling form submission to update records in the database

8. Implementing Delete (D) Operation

- Adding delete functionality to remove records from the database

- Implementing confirmation dialogs for delete actions

9. Working with Data Validation

- Understanding data validation techniques in .NET Core

- Using data annotations for server-side validation

- Implementing client-side validation using JavaScript libraries

10. Handling Errors and Exceptions

- Implementing global error handling in .NET Core

- Using custom error pages and error handling middleware

11. Implementing Search Functionality

- Adding search functionality to filter records

- Implementing search using both client-side and server-side techniques

12. Security and Authentication

- Overview of authentication and authorization in .NET Core

- Implementing authentication using Identity framework

- Securing CRUD operations based on user roles

13. Deployment and Publishing

- Preparing the application for deployment

- Publishing the .NET Core application to a hosting environment

14. Best Practices and Optimization

- Best practices for writing clean and maintainable code

- Techniques for optimizing performance and reducing overhead

15. Real-world Project

- Assigning a CRUD-based project to participants to apply their learning

- Guidance and assistance in building the project

**Angular**

1. Introduction to Angular

- What is Angular?

- Key features and benefits

- Setting up the development environment

2. Angular Architecture

- Understanding the component-based architecture

- Modules, components, templates, and metadata

3. TypeScript Fundamentals

- Introduction to TypeScript

- Data types, variables, and functions in TypeScript

- TypeScript classes and interfaces

4. Creating Angular Components

- Generating components using the Angular CLI

- Understanding component lifecycle hooks

- Building a basic component with template and styles

5. Data Binding and Interpolation

- Understanding data binding (property binding and event binding)

- Using interpolation to display dynamic data in templates

6. Angular Forms

- Template-driven forms vs. Reactive forms

- Creating and validating forms in Angular

7. Services and Dependency Injection

- Implementing services for business logic

- Using Dependency Injection to provide services to components

8. Routing and Navigation

- Configuring routes in Angular

- Creating navigation links and route parameters

9. HTTP Client Module

- Making HTTP requests to a backend server

- Handling API responses and error handling

10. Building a CRUD Application

- Creating a basic CRUD application with Angular components, services, and routing

- Implementing Create, Read, Update, and Delete operations

11. Angular Material (Optional)

- Introduction to Angular Material UI component library

- Using Material components for enhanced user interface

12. Forms and Data Validation

- Performing form validation for CRUD operations

- Handling user input and providing feedback

13. Error Handling and Notifications

- Implementing error handling for HTTP requests

- Displaying notifications to users for success and failure messages

14. Deployment and Hosting

- Preparing the application for deployment

- Hosting the Angular app on different platforms (e.g., GitHub Pages, Firebase)

15. Project Work

- Participants work on a CRUD-based Angular project using all the concepts covered in the bootcamp.