WORLDLINE LEVEL -2

1. *C++*
2. *Java*
3. *C#.Net*
4. *ASP.Net*
5. *Software Testing*
6. *Angular*
7. *Android*

---

1. C++ LEVEL 2 TRAINING TOPICS

Class and Object Concepts: - Constructors and Destructors ·Definition & declaration ·Characteristics of constructors ·Overloaded constructors ·Copy constructor ·Destructors ·Array of objects ·Use of constructors ·Managing the heap ·Passing objects ·Objects & references ·Copy constructor ·Overloading copy constructor ·Returning objects ·Returning object references ·Passing & returning pointers to objects

POLYMORPHISM:- Runtime Polymorphism, Virtual Functions & Dynamic Binding · Pointers & Classes · Pointers to Class Members · Method overriding · Base class pointers & derived class objects · Virtual functions · Virtual functions & base class pointers · Virtual functions & base class reference · Dynamic Binding v/s Early Binding · Pure virtual functions · Virtual destructors · Abstract classes & applications · Generalization & Specialization

Exception Handling:- · Exception handling fundamentals · try, catch & throw · Using multiple catch · The 'catch all' exception handler · Nested try blocks · Rethrowing an exception & overruling a function's return value · Restricting Exceptions – the throw list · handling derived class exceptions · Setting the terminate and unexpected handlers · Application of Exception handling

Templates:- · Generic functions & Data abstraction · Function templates · Explicitly specializing a function template · Overloading Function Template · Using standard parameters · A Generic Sort algorithm · Generic Classes (Class template)

Generic Algorithms:- · mutating algorithms · non-mutating algorithms

Standard Template:- Library List - Map - Multi-Map - Vector - QueueSet

Design Patterns:- Singleton – Decorator - Visitor - Iterator – Proxy **Smart Pointers:-** Share pointer - Auto pointer

C++ 17/ 20: - Nested Namespaces - Variable declaration in if and switch if constexpr statement - Structured bindings - Fold Expressions - Direct list initialization of enums - Generic Lambda Array bounded/unbounded 3-way comparisons

C++ Use case

- Task is to develop a module for a training management application that helps organizations manage employee training and development. The module should allow organizations to create and manage courses, track employee progress, and generate reports on training performance.
- The module should include the following components:
- A class called Course that represents an individual training course. The class should have a constructor that initializes the course with a title, description, instructor, and other relevant details. The class should also have a destructor that deallocates any memory used by the object.
- An array of objects of type Course that stores the training courses. The array should be dynamically allocated on the heap using a smart pointer to manage memory.
- A class called Employee that represents an individual employee. The class should have a constructor that initializes the employee with a name, position, department, and other relevant details. The class should also have a destructor that deallocates any memory used by the object.
- An array of objects of type Employee that stores the employees. The array should be dynamically allocated on the heap using a smart pointer to manage memory.
- A class called Enrollment that represents an individual employee's enrollment in a course. The class should have a constructor that initializes the enrollment with a reference to the employee and a reference to the course. The class should also have a destructor that deallocates any memory used by the object.
- A class called Enrollment Manager that manages employee enrollments. The class should be implemented as a std::map container that stores objects of type Enrollment.
- A class called Report that generates reports on employee training performance. The class should use templates to support generating reports of different types.
- A class called Training Manager that brings together all the components of the system. The class should have a function for adding new courses, a function for adding new employees, a function for enrolling employees in courses, and a function for generating reports on employee training performance.
- A class called Logger that handles logging of errors and other messages generated by the system. The class should use the Singleton pattern to ensure that only one instance of the logger can be created.
- A class called Proxy that provides a simplified interface to the Training Manager class for external applications to interact with. The class should be implemented using the Proxy pattern.
- To test the person's knowledge, you can ask them to implement the above module and provide a demo of how the system works. They should be able to demonstrate their understanding of constructors, destructors, overloaded constructors, copy constructors, returning objects, passing objects, and passing and returning pointers to objects. They should also be able to demonstrate their understanding of runtime polymorphism, virtual functions, dynamic binding, pointers and classes, virtual destructors, abstract classes, generic functions and data abstraction, and the use of standard library containers such as std::map, std::deque, and std::vector. Finally, they should be able to demonstrate their understanding of exception handling, templates, mutating and non-mutating algorithms, and design patterns such as Singleton, Decorator, Visitor, Iterator, and Proxy.

2. JAVA LEVEL 2 TRAINING CONTENT

Object Oriented Programming: Methods and Constructors – Default - Constructors and Constructors with - Arguments, Abstract, Static, Final - Garbage Collection, String class idiosyncrasies, Cosmic Super class - Object Class, Exception Handling, FILE IO - Data Handling and Functions - Functions, String Handling - String, String Buffer Classes, serialization, string builder class, utility classes

Packages and Multi-Threading: Multi-Threading: Runnable Interface, Extending a Thread Class, Synchronization in Threads. Executor framework, callable interface, concurrency package.

Exception Handling: Try, Catch, Finally, Throw and Throws

Wrapper Classes and Inner Classes: Integer, Character, Boolean, Float etc **Collections:** Array List, Vector, HashSet, Tree Set, HashMap, Hash Table ,Wildcards

Java 8 Features: Base 64 encoding / decoding, default and static methods in Interfaces, Functional Interfaces and Lambda Expressions ,Streams and Optional , Predicates

Parser DOM Parser: Writing into an XML file and Parsing an XML file SAX Parser, XSL

JDBC: Introduction to SQL: Connect, Insert, Update, Delete, Select - Introduction to JDBC and Architecture of -Types of Drivers: Type 1/2/3/4 drivers Insert/Update/Delete/Select Operations - Batch Processing Transaction Management: Commit and Rollback

Servlets: Introduction to Web Technologies ,Type of Servlets: Generic and Http Servlet , Request Dispatchers: Forward and Include - 4 types of Session Tracking and Filters

Hibernate and JPA with annotation Introduction to Hibernate Architecture of Hibernate - Database Operations: Insert/Update/Delete/Select - Inheritance Collections - HQL and Restrictions - Caching in Hibernate

Testing : Junit, Mockito, Loggers, Build tools: Maven, Gradle

Spring: Introduction to Spring Framework Architecture - Display a Sample Message - IoC Containers - Bean Definition Bean Scopes - Bean Post Processors - Dependency Injection - Auto-Wiring

Spring Boot: Spring Boot Project Set Up - Create Spring Boot Starter Project - Developing JPA Entities - Create REST APIs with Spring Data - JPA Repositories and Spring Data REST, Thyme Leaf - Session Management with Spring, Session Management techniques - Stateful/Stateless, JWT tokens

Web Services: Introduction to Web Services - WSDL file - WSDL and UDDI SOAP, RESTful Web Service JAX-WS Implementation

Introduction of Design Patterns : Singleton , Factory Design pattern , Abstract Factory, Builder ,Decorator, Etc

Logging : Log4jxml , Log back

Java Use Case: Develop a Training Module Management System

- You have been tasked with developing a Training Module Management System for a company. The system should allow the company to create, manage, and track training modules for its employees. The system should be developed in Java, using object-oriented programming concepts and related topics.
- Functional Requirements:
    - Create Training Module: The system should allow a user to create a new training module. Each training module should have a name, description, and list of topics covered.
    - Manage Training Module: The system should allow a user to update, delete, and view training modules. The user should be able to search for training modules by name or topic.
    - Assign Training Module: The system should allow a user to assign a training module to one or more employees. The user should be able to view the progress of each employee for a particular training module.
    - Generate Report: The system should allow a user to generate a report that shows the progress of all employees for a particular training module.
- Non-Functional Requirements:
    - Exception Handling: The system should handle exceptions and provide meaningful error messages to the user.
    - File I/O: The system should store and retrieve data from a file.
    - Multi-Threading: The system should use multi-threading to improve performance.
    - Java Collections: The system should use Java collections such as ArrayList and HashMap to store and manage data.
    - Java 8 Features: The system should use Java 8 features such as streams and lambdas to process data.
    - Hibernate: The system should use Hibernate to store and retrieve data from a database.
    - Design Patterns: The system should use design patterns such as Singleton and Factory to improve the code quality and maintainability.
    - Web Services: The system should expose RESTful web services to allow external systems to access the data.
- Deliverables:
    - Source Code: The source code of the Training Module Management System.
    - Documentation: The documentation of the system, including user manual and technical documentation.
    - Test Cases: The test cases that validate the functionality of the system.
    - Deployment Package: The deployment package that can be used to deploy the system on a production environment.
    - Presentation: A presentation that explains the design and implementation of the system.
- The person should be able to implement this use case using the concepts and topics they have learned, such as Object-Oriented Programming, Constructors, Java Collections, Multi-Threading, Exception Handling, File I/O, Java 8 Features, Hibernate, Design Patterns, and Web Services.

---

3. C# L2 TOPICS

Understanding .NET: The C# Environment .NET Strategy .NET Framework CLR JIT & IL Automatic Memory management Framework Base Classes - System, Exception Visual Studio 2022 .NET Languages Benefits of .NET C# and .NET .NET Framework & .Net Core 3.1 with new features of .NET 6 Nunits.

Overview of C# : •Simple C# Program •Namespaces •Comments •Main function •Classes •Console.Writeline •CommandLine Arguments •Compile time errors •Program Structure •Coding Style

Application & Libraries: • Console class • Console Input and output • Formatted output •Numeric, string and Date formatting • Reusable Component - class library – dll

Literals, Variables & Data Types : •Literals • Variables • Data Types • var keyword • Value Types • Reference Types • Variable declaration & initalization • Default Values • Constant Variables • Scope of Variables • Boxing & Unboxing

Operators & Expressions: • Arithmetic Operators • Relational Operators • Logical Operators •Assignment Operators • Increment, Decrement Operators • Conditional Operators • Bitwise Operators •Evaluation of expressions • Precedence and Associativity • Type conversion • Math functions

Decision making and branching: • If else • switch • ? : Operator d**ecision making and looping:** •While • do while • for • foreach

Methods in C# :• Methods • Method Parameters • Pass by value • Pass by reference • Output parameters • Variable argument list • Method overloading

Arrays: • 1-D Arrays • 2-D Arrays • MultiDimensional Arrays •Jagged Arrays • System.Array Class • ArrayList Class

Strings: • Creating strings • common string methods • comparing strings • finding substrings •mutable strings • Array of strings • Equals vs ==

Structs & Enums: • Structs •Struct Methods • Struct vs Classes • Enums • Enum initialization •Enum Type conversion

Object Oriented Programming :• Oop Principles • Defining a class • Fields and methods • access modifiers •Creating objects • Accessing class members • Constructors • Overloaded Constructors •Static members • Static and private constructors • Copy Constructors • Destructors • Member initialization • this reference • constant members • readonly members • Properties, get and set •Inheritance and polymorphism • Classical Inheritance • Containment Inheritance • Defining a subclass • Subclass constructor • multilevel inheritance • Hierarchial inheritance • overriding methods • abstract classes and methods • sealed classes and methods • polymorphism •extension classes • Interfaces • Defining and interface • extending an interface • implementing an interface •interfaces and inheritance • abstract classes and interfaces Operator overloading : Overloadable Operators • Need for operator overloading • overloading unary, binary and comparison operators

Error & Exceptions: • What is Debugging • Types of errors • Exceptions • Exception Properties •Exception handling syntax • Multiple catch statements • Exception Hierarchy • general catch handler • finally statement • throwing exceptions • Custom exceptions

Debugging: • Debug Windows - Data Tips, Locals/Auto, Quick Watch, Watch, Call Stack, Immediate •BreakPoints & TracePoints • Glyphs • Add, Delete, Disable, Enable BreakPoints • Breakpoint Filter •Breakpoint Condition • BreakPoint HitCount • Code Stepping - Step Into, Step Over, Step Out • Edit Variable Value • Set the next statement

Collections: • Basic Collection Interfaces • Queue • I list interface • ArrayList Class • IDictionary •StringDictionary • Types of Collections

Windows Forms: Form class • MessageBox class • Button Control • Label & LinkLabel •TextBox •RichTextBox • CheckBox • RadioButton • NumericUpDown, PictureBox, ComboBox •ProgressBar, MonthlyCalendar And DateTimePicker Control • WebBrowser Control • OpenFile, SaveFile & Color Dialog Controls

Delegates and events : Delegates •Declaring delegates • delegate methods, instantiation and invocation •using delegates • Events • Events using custom delegates • Event using inbuild EventHandler delegate

File IO & Serialization: System.IO namespace • What is Stream? •FileStream class • File, Directory classes •StreamReader & StreamWriter class • BinaryReader & BinaryWriter class • Reading and writing data • What is Serialization? • Type of Serialization •Binary, SOAP, XML, JSON serialization • Serialize and deserialize data •

ADO.NET: Introduction to ADO.NET • ADO.NET Architecture • Data Providers • Configuring Connection String in Config file •Coding using Connected and Disconnected architecture • Manipulate Data using ADO.NET
LINQ: Understanding the LINQ Framework • LINQ Providers o LINQ to Objects o LINQ to SQL o LINQ to Entity o LINQ to Dataset oLINQ to XML • IEnumerable<T> and IQueryable<T> interfaces •System.Linq namespace •LINQ query with declarative and method syntax •LINQ query execution • Query operators - Restriction Operator - Projection Operator - Ordering Operators - Grouping Operators - Join Operators - Set Operators Aggregate Operators

Entity Framework: Introduction to Data Access Layer – ORM - Entity Framework architecture - EF Data model : DataFirst, CodeFirst, ModelFirst - EDM - Data retrieval in EF - Data manipulation in EF - Lazy loading vs eagar loading - Code First approach

Multi Threading : Multi Tasking OS • Process & Threads • What is a thread • A typical windows exe lifecycle • Single threaded model & drawbacks • Multi Threading • Main Thread & Worker Threads • Working with Threads in C# • Os Scheduler • Thread States • Stack & Heap in Threads • Foreground vs Background Threads •Debugging Threads in Visual Studio • Stopping or Pausing a thread • Shared Resources • Thread Scheduling •Access UI from Worker Thread • Thread Pool

Task Parallel Library • What is TPL? • Benefit of multi core • Task class • Task.Factory & Task.run •Async & Await Keywords

Use Case: Develop a Training Module Management System

You have been tasked with developing a Training Module Management System for a company. The system should allow the company to create, manage, and track training modules for its employees. The system should be developed in C#, using object-oriented programming concepts and related topics.

Functional Requirements:

Create Training Module: The system should allow a user to create a new training module. Each training module should have a name, description, and list of topics covered.

Manage Training Module: The system should allow a user to update, delete, and view training modules. The user should be able to search for training modules by name or topic.

Assign Training Module: The system should allow a user to assign a training module to one or more employees. The user should be able to view the progress of each employee for a particular training module.

Generate Report: The system should allow a user to generate a report that shows the progress of all employees for a particular training module.

Non-Functional Requirements:

Exception Handling: The system should handle exceptions and provide meaningful error messages to the user.

File I/O: The system should store and retrieve data from a file. **Multi-Threading**: The system should use multi-threading to improve performance. **C# Collections**: The system should use C# collections such as List and Dictionary to store and manage data.

C# Features: The system should use C# features such as LINQ and Lambda expressions to process data.

Entity Framework: The system should use Entity Framework to store and retrieve data from a database.

Design Patterns: The system should use design patterns such as Singleton and Factory to improve the code quality and maintainability.

Web Services: The system should expose RESTful web services to allow external systems to access the data.

Deliverables: **Source Code**: The source code of the Training Module Management System. **Documentation**: The documentation of the system, including user manual and technical documentation. **Test Cases**: The test cases that validate the functionality of the system. **Deployment Package**: The deployment package that can be used to deploy the system on a production environment.

Presentation: A presentation that explains the design and implementation of the system.

The person should be able to implement this use case using the concepts and topics they have learned, such as Object-Oriented Programming, Constructors, C# Collections, Multi-Threading, Exception Handling, File I/O, C# Features, Entity Framework, Design Patterns, and Web Services

---

4. ASP.NET L2 TRAINING

Introduction to Asp.Net: What is Web Application - Web Architecture - ASP.NET Architecture - HTTP Response / Request Model - Page life cycle - Introduction Visual Studio IDE - Asp.NET Web Form and code behind model
Web Forms Controls: Using HTML server controls - ASP.NET Server controls - IsPostBack Property - PostBack Control - List Controls - Auto Post Back Property

Validation Controls: Validating user input using validation controls - RequiredField Validator - Compare Validator - Range Validator - Required Field Validator - Validation Summery - Custom Validator (writing client side validation script) Validation Group

Themes and Master Pages: Creating Themes in skin files - Apply themes at the page level and application level - Master page - Content page - Accessing controls defined in master page

ADO.NET: Introduction to ADO.NET - Calling Data from class library and binding to control - Creating data application in ASP Web Form

Data Source Controls: SQL Data Source Control - Object Data Source Control - Example – Retrieve, Insert, update, Delete data using Data Source

Data Bound Controls : GridView - Detail View - ListView & Data Pager - Chart Control

Navigation Controls: XML Data Source - SiteMap Data Source - TreeView - Menu - SiteMap Control

State Management: How to manage the state in ASP.NET - Client side state management techniques - View State - Query String - Cookie - Server side state management techniques - Application Variable - Session Variable

Caching: What is Caching - Apply caching in aspx - Apply caching using Cache object - Page, Fragment, Parameter Caching option - Use if Substitution control in caching - Expiration policy - Absolute - Sliding - Dependency - File - SQL Data Cache invalidation

Security: ASP.NET Security Model - Authentication - Windows Authentication - Forms Authentication - Passport Authentication - Authorization - User level - Role based

.Net core and .Net 6: need to be included and taught in detail as project requirement is today to have candidate ready with this skills.

Use Case: Managing an Online Course with a Learning Management System

- Course content is created and organized in the LMS
- Students are enrolled in the course
- The instructor logs into the LMS and accesses the course management interface
- The instructor views the list of enrolled students and their progress in the course
- The instructor adds new course content or updates existing content in the LMS
- The instructor sets due dates for assignments and assessments
- The instructor creates and assigns quizzes and assessments for students
- The instructor sends announcements or messages to students through the LMS

- The instructor monitors student activity and engagement with the course materials and assessments
- The instructor grades student assignments and assessments within the LMS
- The instructor provides feedback to students on their work through the LMS
- The instructor generates reports on student progress, including grades and course completion
- Course content and student information is saved and updated in the LMS
- Students have access to updated course materials and assessments
- Instructor has an overview of student progress and course completion
- Students receive grades and feedback on their work9

---

5. SOFTWARE TESTING L2

Testing Throughout the Development Cycle: - SDLC models - Co relation between Development and Testing Activities - V Model - Testing Methodologies [V&V] - Testing levels [Unit/Integratio] Testing Throughout the **Development Cycle:** - Testing levels [System/UAT] - Maintenance Testing - Functional vs Non Functional Testing **Static Testing:** - Static Testing types - Walkthrough Process - Review Process - Work products for review White box testing techniques - Statement Coverage - Condition Coverage - Decision Coverage Test Design Techniques - Identification of Test scenarios, Test conditions - Writing effective test cases - Understanding Test case priority - Test case review Black box testing techniques - Equivalence Class Partitioning - Boundary Value Analysis - Decision Tables -State Transition -Use Case Other types - Error guessing - Exploratory Testing Test Management - Roles and Responsibilities of Test Team - Test planning - Contents of Test Plan document - Test strategy - Entry criteria - Execution Cycle - Exit criteria - Risk Analysis - Test monitoring and control - Test closure activities [reports and metrics]

Defect Management - What is a Defect - Defect Attributes - Defect Tracking - Defect Life Cycle - Defect severity / priority - Difference and as well co-relation between test case priority and defect priority Tool Support for Testing - Types of tools used - Test management tools - Defect tracking tools - Test automation tools - Other tools - Tool selection criteria - Success factors for tools Other type of Testing - User Experience Testing - Performance Testing - Security Testing - Configuration and Compatibility Testing Testing in Agile - Agile values and Principles - **Introduction to scrum** - Scrum artifacts - Scrum events and ceremonies - Scrum roles and responsibilities - Testing techniques in agile Testing Methodologies - Spiral Methodology - XP (Extreme Programming) Methodology

Use Case: Software Testing in the Development Cycle

Basic Flow:

The testing team reviews the SDLC models to understand the various phases of the development cycle and the testing activities that should occur in each phase.

The testing team collaborates with the development team to determine the co-relation between the development and testing activities.

The V model is used to plan the testing activities and ensure that testing occurs at each stage of the development cycle.

The testing team determines the appropriate testing methodologies (V&V) and testing levels (Unit/Integration) for the application.

The testing team conducts static testing, including walkthroughs and reviews of the work products, to identify defects and improve the quality of the application.

White box testing techniques are used, including statement coverage, condition coverage, and decision coverage, to ensure that all code paths are tested.

The testing team applies test design techniques to identify test scenarios, test conditions, and write effective test cases. Test case priority is also determined and reviewed.

Black box testing techniques are applied, including equivalence class partitioning, boundary value analysis, decision tables, state transition, and use cases to test the application from a user perspective.

Other types of testing are performed, including error guessing and exploratory testing, to identify defects that may have been missed.

Test management is performed by the test manager, including the roles and responsibilities of the test team, test planning, contents of the test plan document, test strategy, entry criteria, execution cycle, exit criteria, risk analysis, test monitoring and control, and test closure activities (reports and metrics).

Defect management is used to track and manage defects, including understanding what a defect is, defect attributes, defect tracking, the defect life cycle, defect severity/priority, and the difference between test case priority and defect priority.

Tool support is used for testing, including test management tools, defect tracking tools, test automation tools, and other tools. Tool selection criteria and success factors for tools are also considered.

Other types of testing are performed, including user experience testing, performance testing, security testing, and configuration and compatibility testing.

Testing in agile is performed using agile values and principles, including an introduction to scrum, scrum artifacts, scrum events and ceremonies, scrum roles and responsibilities, and testing techniques in agile.

The testing team uses various testing methodologies, including spiral methodology and XP (Extreme Programming) methodology, as appropriate.

6. ANGULAR TRAINING L2 TOPICS

Introduction Getting started:- What is Angular? Try it Setup Understanding

Angular:- Overview Components Overview Component lifecycle View encapsulation Component interaction Component styles Sharing data between child and parent directives and components Content projection Dynamic components Angular elements

Templates:- Overview Introduction Text interpolation Template statements

Binding:- Understanding binding Property binding Attribute binding Class and style binding Event binding Two-way **binding Pipes:-** Understanding pipes Using a pipe in a template Transforming data with parameters and chained pipes Template reference variables SVG as templates

Directives:- Built-in directives Attribute directives Structural directives Directive composition API

Dependency injection:- Dependency injection in Angular Understanding dependency injection Creating an injectable service Defining dependency providers Hierarchical injectors

Developer guides:- Standalone Standalone components Migrating to standalone

Change detection:- Zone pollution Slow computations Skipping component subtrees

Routing and navigation:- Common routing tasks Tutorial: routing in single-page applications Tutorial: creating custom route matches Tutorial: adding routing to Tour of Heroes Router reference

Forms:- Introduction Reactive forms Strictly typed reactive forms in depth Validate form input Building dynamic forms HTTP client Image optimization

Testing:- Intro to testing Code coverage Testing services Basics of testing components Component testing scenarios Testing attribute directives Testing pipes Debugging tests Testing utility APIs

Internationalization:- Overview Common internationalization tasks Example Angular application Optional internationalization practices

Animations:- Introduction Transition and Triggers Complex Sequences Reusable Animations Route transition animations

Use Case: Frontend design for a training management system

---

7. COURSE TITLE: ANDROID APP DEVELOPMENT WITH KOTLIN

Course Objective: This course aims to provide students with a solid understanding of Kotlin and its application in Android app development. By the end of the course, students will be able to design, build, and deploy Android applications using Kotlin.

Prerequisites: Basic programming knowledge (preferably Java or any other object-oriented language)

Course Outline:

1. **Introduction to Kotlin and Android Studio**
    - Introduction to Kotlin programming language
    - Setting up the Android Studio development environment
    - Kotlin syntax and basic programming concepts
    - Variables, data types, and operators in Kotlin
2. Kotlin Fundamentals and Control Structures
    - Conditional statements (if, when)
    - Loops (for, while, do-while)
    - Functions, parameters, and return types
    - Kotlin collections: arrays, lists, and maps
3. Object-Oriented Programming with Kotlin
    - Classes and objects in Kotlin
    - Properties, fields, and methods
    - Inheritance, polymorphism, and interfaces
    - Data classes and sealed classes
4. Android Basics and User Interface
    - Understanding Android architecture
    - Activities, intents, and the activity lifecycle
    - Designing UI with XML and Kotlin
    - Working with views, view groups, and layouts
5. Android Components and Navigation
    - Fragments and their lifecycle
    - Navigation and passing data between fragments
    - RecyclerView and ListView
    - Android storage options: Shared Preferences, Files, and Databases
6. Networking and Data Handling
    - Working with RESTful APIs and JSON data
    - Using Retrofit and OkHttp for network communication
    - Implementing LiveData and ViewModel
    - Basic understanding of Coroutines for asynchronous programming
7. Advanced Android Features
    - Notifications and PendingIntent
    - Location and Google Maps integration
    - Permissions and runtime permission handling
    - Material Design components and theming
8. Testing, Debugging, and Deployment
    - Unit testing and UI testing with JUnit and Espresso
    - Debugging tools and techniques in Android Studio

- Optimizing app performance and memory management
- Preparing and publishing your app to Google Play Store

Use Case App Overview:

The Event Management App allows users to discover, create, and manage events in their local area. The app includes features such as event browsing, registration, user authentication, and event organization.

Capstone Project: Event Management App

Introduction to Kotlin and Android Studio

Set up the development environment and create a new Android project using Kotlin

Design the basic structure of the app and plan the required features

Kotlin Fundamentals and Control Structures

Implement basic event browsing using conditional statements and loops

Parse event data and display it in a list format

Object-Oriented Programming with Kotlin

Create classes and interfaces for event objects, users, and organizers

Implement data classes for event data and user profiles

Android Basics and User Interface

Design the app's user interface, including event listing, event details, and user profile screens

Implement navigation between different activities and fragments

Android Components and Navigation

Implement RecyclerView for event listing and efficient scrolling

Use Shared Preferences to store user preferences and settings

Develop the registration and login system using fragments and navigation components

Networking and Data Handling

Integrate the app with a RESTful API to fetch and manage event data

Use Retrofit and OkHttp for network communication and JSON data parsing

Implement LiveData and ViewModel for efficient data handling and updating UI components

Use coroutines for handling asynchronous tasks, such as fetching data from the server

Advanced Android Features

Implement notifications for event reminders and updates

Integrate Google Maps to show event locations and provide directions

Handle runtime permissions for location access and other required permissions

Apply Material Design principles to improve the app's appearance and user experience

---