Day Wise Schedule

Days	Contents Outline	Details
Day 1	Intro and **Project Specs**	**Team of 3 participants to is formed and assigned one project. The team needs to finish this project by end of the course.**
Day 2	UML	• What is Object-Oriented Development? • Key Concepts of Object-Oriented Design • Cohesion and coupling • Analyzing a System • UML Overview • Building Blocks (4+1 view) • Modeling Types • Basic Notations • Standard Diagrams • Structural Diagrams o Class Diagram o Object Diagram o Component Diagram o Deployment Diagram • Behavioral Diagrams o Use Case Diagram o Interaction Diagram o State chart Diagram o Activity Diagram o Sequence Diagram • Creating diagrams with examples • Hands on using a scenario (Hands-on should cover the diagrams highlighted above) UML design as per project given (minimum coverage of 4+1 Architecture views for the project)
Day 3 to Day 5	MYSql	• Keys, Constraints and Indexes • Privileges • Administration (Logins, Users)

		• Programming (Functions, Procedures, Variables, Constants, Loops) • Comparison Operators • SQL Server Queries (CRUD with examples) • Joins (Inner, Left, Right, Self-Join) • Aliases • Clauses (Distinct, From, Where, Order by, group by, having) • SQL Server aggregate Functions (count, sum, avg, etc.) • Conditions (and, or, like, in) • Tables and Views (CRUD examples) • Data Types • Creating Indexes • Understanding Index Creation Options • Maintaining Indexes • Introducing Views • Defining and Using Views • Using Views to Optimize Performance • Introducing User-Defined Functions • Implementing User-Defined Functions • Introducing Triggers • Creating, Altering, and Dropping Triggers • Working with Triggers • Implementing Triggers Project Work: • Creation of database tables • Normalization of database • Putting table constraints • Adding sample data to Tables • Analyzing relationships and joins • Basic Stored procedures
Day 6 To Day 15	Core Java (JDK 1.8), Unit testing, Code Quality and Test Design (Classroom with everyday	**Java Basics , OOPs , Classes , Constructor , Methods , Static , Inhetritance , Polymorphism ,**

	exercises) • All this to be done using TDD • Core Java Language & Unit testing using JUNITs • Effective test case design • Writing good quality code • Maven • Git & Jenkins • SONAR, quality gates and build integration. Code quality aspects including testability, complexity, coupling etc. • Writing effective unit test cases and code coverage including mocking	**Overloading , Overriding , Abstract classes , Interfaces , InstanceOf , Cloning , Comparator , HashCode , Equals, toString , Packages , ExceptionHandling , MultiThreading , Collections , IO , Serialization , DeSerialization , Transiant , Volatile , Lambda , Functional Interface , Predicate , Streams , Method Reference, Property etc.** **TDD , JUnit Unit testing Writing test cases.** **Mocking** **Maven , Local Repository , Central Repository , Remote Repository , POM , Build.** **Sonar Lint quality gates and build Integration.** **Git** **Jenkins**
Day 16 to Day 18	Object oriented JS, CSS3, HTML5, Bootstrap, AJAX (Classroom)	Object oriented JS JavaScript basics, technologies overview, Introduction, Built-in Object, Expressions and operators, statements and declarations, Functions, Classes, Errors, OOPs Concepts. CSS3 Introduction, Syntax, Inclusion, Measurement Units, Colors, Backgrounds, Fonts, Text, Images, Links, Tables, Borders, Margins, Lists, Padding, Cursors, Outlines, Dimension, scrollbars, Positioning, Layers, Pseudo Classes/ elements, Text Effects, Media types etc. HTML5 Overview, Syntax, Attributes, Events, SVG, MathML, Web Storage, Web SQL, Server-Sent Events, WebSocket, Canvas, Geolocation, Drag and Drop etc. Ajax

		Introduction, Action, XMLHttpRequest, database Operations, Security etc. BootStrap Grid and CSS and Buttons
Day 19 to Day 26	Web App Development (project basedclassroom with exercises. All this to include TDD and unit test case development and SONAR integration. Maven to be used for build.) • JEE understanding • JSP, Servlets • JAX-WS and JAX-RS, JMS • Annotations, Dependency Injection • JPA and Hibernate	**Extending e-commerce application to web application and enterprise application. Here, we are focused on writing web UI interfaces and binding business logic as REST / SOAP web services**
Day 27 to Day 36	Spring Framework Version 5 (project-based classroom with exercises. All this to include TDD and unit test case development and SONAR integration. Maven to be used for build.) • Core framework – DI, IoC, AOP, JPA, Template, Transactions • Spring Boot • Servlet Stack (Servlet containers, Spring Security, MVC, Spring Data – JDBC/JPA/NoSQL) • Reactive Stack (Servlet 3.1+ containers, reactive Stream Adapters, Spring Security Reactive, Spring WebFlux, Spring Data Reactive Repositories) • Swagger • Functional automation using SOAP UI for services developed both REST and SOAP)	Modify your previous web application to use: 1. Spring Servlet Stack 2. Spring Reactive Stack Expose our business logic as services for Heterogeneous clients. Testing services with SOAP UI
Day 37 to Day 43	Angular 4 – Classroom • NPM, Gradle/Gulp • Unit testing in JS using Jasmine/Karma	Angular Introduction to Angular - What can we build ?, Coding in Angular 2, Building Blocks of

	• Functional automation using Protractor • Integration with SONAR	Angular – Modules, Components, Templates, Metadata, Data Binding – Interpolation, One way binding (Property Binding), Event binding, Two way binding, Angular Structural Directives - *ngFor, *ngIf and *ngSwitch, Pipes - Using built-in pipes, Date Pipe, Numeric Pipe, Services - Purpose of Services, Dependency Injection, How to set the providers ?, Component Lifecycle hooks, Data with Http - Using the HttpModule, Exception handling with Http, Routing - Introducing Routing, Routing Essentials, Adding the Routing module to an App, Adding routing to templates Node.js Introduction, Setup, REPL Terminal, NPM, Callbacks Concept, Event Loop, Event Emitter, Buffers, Streams, File System, Global Objects, Utility Modules, Web Modules, RESTFul API. JS Testing Using Jasmine Introduction, Setup, describe, The Spec Runner, Before and After, Spies, spyon() and createSpy() functions. Karma – Test Runner overview Protactor How Protractor Works Understanding Promises and promise-based tests Jasmine Locating Elements Automatically Interacting with the Page Debugging Protractor Tests Mocking HTTP Requests

		Integration with SONAR and implementation of Quality Gate
Day 44 to Day 47	UI for project using Angular/HTML5 (Project Work) Unit tests for Angular code (Project Work)	As per Project selection. Follow TDD approach (First write Unit tests, followed by code). All code to pass SONAR code quality gates.
Day 48 to Day 49	BDD • Cucumber/jBehave (classroom & everyday exercises)	Cucumber based acceptance tests
Day 50	DONE Workshop	