Java Lectureflow

Module-1) SE - Overview of IT Industry	5

- Introduction of students
- Career in IT
- Understanding Student Login of TOPS ERP
- Using Lab
- What is Program
- What is programming?
- Types of Programming Language
- World Wide Web
- How Internet Works
- Network Layers on Client and Server
- Client And Servers
- Types of Internet Connections
- Protocols
- Application Security
- Software Applications and its types
- Software Architecture
- Layers in Software Architecture
- Software Environments
- Types of Programming Languages
- Source Code
- Github and introductions
- Student Account in Github
- Types of Software
- Introduction of Software
- Application software
- Software development process
- Software Requirement
- Software Analysis
- System Design
- Software Testing
- Maintenance
- Development
- Web Application
- Designing
- moble application
- DFD
- Desktop Application
- Flow Chart

Module 3) SE - Introduction to Programming - May 2024	15

- Overview of C Programming - What is C Programming? History and Evolution, Importance and Applications
- Setting Up Environment -Installing a C Compiler (e.g., GCC), Choosing an IDE (DevC++. VS Code, Codeblocks, etc) Writing Your First Program
- Basic Structure of a C Program - Structure of a C Program ,Comments in C, Data Types and Variables, Constants, Keywords and Identifiers
- Operators - Arithmetic Operators, Relational Operators, Logical Operators, Assignment Operators, Increment and Decrement Operators, Bitwise Operators, Conditional Operator
- Control Flow Statements - Decision-Making in C - If Statement, If_Else statement, If_elseif (Elseif Ladder), Nested if-else statement, Switch statement
- Looping in C - While Loop, For Loop, Do-While Loop
- Loop Control Statements - Break, Continue, Go to
- Functions in C - Introduction to Functions, Function Declaration and Definition, Function Call and Return
- Arrays in C - Introduction to Arrays, One-Dimensional Arrays, Multi-Dimensional Arrays
- Pointers in C - Introduction to Pointers, Pointer Declaration and Initialization
- Strings in C - Introduction to Strings, String Handling Functions, strlen, strcpy, strcat, strcmp, strcmpi,strchr, String Input and Output
- Structures in C - Introduction to Structures, Structure Declaration and Initialization, Array of Structure Nested Structures
- File Handling in C, File Operations (Opening, Closing, Reading, and Writing), File Pointers, File Handling Functions

Module-4) Introduction to OOPS Programming	8

- Introduction to C++ - Understanding the Basics of Programming, Introduction to C++ Language, POP Vs OOP, Advantages of OOP, Setting Up C++ Development Environment, Writing and Running Your First C++ Program, Input and Output Operations in C++
- Variables, Data Types, and Operators - Variables and Constants in C++, Data Types and Size Specifiers, Assignments, Arithmetic, Relational, Logical, and Bitwise Operators, Type Conversion in C++, Constants and Literals
- Control Flow Statements - Conditional Statements: if, if_else, else if ladder, nested if, Switch Statement, Loops: while, do-while, for, Break and Continue Statements, Nested Control Structures
- Functions and Scope - Introduction to Functions ,Function Prototypes and Definitions, Parameters and Return Values, Scope of Variables
- Arrays and Strings - Introduction to Arrays, Single-Dimensional and Multi-Dimensional Arrays, Array Initialization, Accessing Elements, and Manipulation, Introduction to Strings in C++, String Operations and Functions
- Introduction to Object-Oriented Programming -Understanding the Basics of Object-Oriented Programming, Advantages of OOP Paradigm, Key Concepts: Classes, Objects, Inheritance, Polymorphism, Encapsulation, Introduction to C++ as an Object-Oriented Language

- Classes and Objects - Declaring Classes and Objects in C++, Class Members: Data Members and Member Functions, Constructors and Destructors, Access Specifiers: Public, Private, Protected, Class Member Functions: Inline and Outside Definitions
- Inheritance - Concept of Inheritance and Reusability, Types of Inheritance: Single, Multiple, Multi-level, Hierarchical, Base and Derived Classes in C++, Access Control and Inheritance, Constructors and Destructors in Inheritance
- Polymorphism - Understanding Polymorphism in OOP, Compile-Time and Runtime Polymorphism, Function Overloading and Operator Overloading, Virtual Functions and Dynamic Binding, Abstract Classes and Pure Virtual Functions, Scope resolution operator, Static Keywords, Inline Function
- Encapsulation - Understanding Encapsulation and Information Hiding, Access Specifiers: Public, Private, Protected, Encapsulation in C++ Classes, Benefits of Encapsulation in OOP, Friend Functions and Friend Classes
- File Handling - Introduction to File Handling in C++, Opening, Closing, Reading, and Writing Files, Error Handling in File Operations, Working with File Streams

Module-5) SE - Introduction to DBMS	**9**

- Introduction to SQL - SQL Overview, Definition of SQL, Importance of SQL in Database Management, DBMS-RDBMS
- SQL Syntax - Basic SQL Syntax, Structure of SQL Statements
- SQL Constraints - Types of Constraints (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY), Implementing Constraints in Tables
- Main SQL Commands and Sub-commands - Data Definition Language (DDL), CREATE Command, Creating Tables, Specifying Column Names, Data Types, and Constraints
- ALTER Command - Modifying Existing Tables, Adding, Modifying, or Dropping Columns
- DROP Command - Deleting Tables from the Database
- Data Manipulation Language (DML) - INSERT Command, Adding Data into Tables Specifying Column Names and Values
- UPDATE Command - Modifying Existing Data in Tables, Changing Values in Specific Columns
- DELETE Command - Removing Data from Tables, Deleting Specific Rows with WHERE Clause
- Data Query Language (DQL) - SELECT Command, Retrieving Data from Tables, Filtering Data with WHERE Clause, Sorting Data with ORDER BY Clause, Limiting Results with LIMIT or FETCH FIRST Clause
- Data Control Language (DCL) -GRANT Command, Granting Privileges to Users or Roles, Granting SELECT, INSERT, UPDATE, DELETE Permissions
- REVOKE Command - Revoking Privileges from Users or Roles, Removing SELECT, INSERT, UPDATE, DELETE Permissions
- Transaction Control Language (TCL) - COMMIT Command, Saving Changes Permanently to the Database
- ROLLBACK Command - Reverting Uncommitted Changes, ?SAVEPOINT Command - Creating Intermediate Points in a Transaction for Rollback
- SQL Joins - Inner Join, Left Join, Right Join, Full Outer Join
- SQL Group By - Grouping Data in SQL Queries

- SQL Stored Procedure - Definition and Purpose of Stored Procedures, Creating and Executing Stored Procedures
- SQL View - Creating Views in SQL, Advantages of Using Views
- SQL Trigger - Introduction to Triggers, Types of Triggers (INSERT, UPDATE, DELETE)
- Introduction to PL/SQL - Definition and Purpose of PL/SQL, Benefits of Using PL/SQL
- PL/SQL Syntax - Structure of PL/SQL Blocks, Variables, Constants, and Data Types in PL/SQL
- PL/SQL Control Structures - IF-THEN, IF-THEN-ELSE, CASE Statements ,Loops (WHILE, FOR)
- SQL Cursor - Introduction to Cursors in PL/SQL, Implicit vs. Explicit Cursors
- Rollback and Commit Savepoint - Transaction Management in PL/SQL

Module 12) - Java -Introduction	**1**

- Introduction Lecture
- Introduction of students
- Understanding Student Login of TOPS ERP
- Working on Project and Assignment
- Using Lab
- Career in IT

Module 13) - Java - Core Java	**15**

- Conditional Statements (If, If Else, Nested If Else If)
- Introduction of Core Java
- Practical Example : 1. Odd-Even, 2. Prime Number, 3. Max out of three, 4. Student's grade system
- Eclipse IDE
- (Switch Case)
- JVM,JDK,JRE
- Practical Example : 1. Mini Calculator
- Class, Object Constructor
- Loops (While, Do While, For)
- Class, Object, Method
- Practical Example : 1. Sum of n numbers, 2. patterns, 3. prime numbers for a range
- Constructor
- Break and Continue
- Garbage Collection
- Practical Example : 1. exit or continue from loop using break & continue
- Finalize
- SDLC Process
- Project Analysis
- Source File Layout
- Analysis In Details
- DFD (with practical)
- Package Management, Modifiers- Public, Private, Protected, Default
- Introduction of DFD

- Import Statement
- Rules for Drawing DFD
- Context Level
- Data types
- First Level
- Primitive Types
- Second Level
- Reference Types
- Array Introduction
- Data Dictionary
- Modifiers - Public, Private, Protected, Default
- Why Array? Advantages
- Flow Chart
- Types of Array
- Resizing Array
- Copying Array
- Primitive types and Reference type Arrays
- Encapsulations
- Advantages of Inheritance
- Types of Inheritance
- Practical of Inheritance
- Practical of Inheritance with Constructor
- Polymorphism
- Types of Polymorphism
- Method Overloading and Method Overriding
- Abstract and Interface - Introduction and Difference
- Keywords - This, Static, Final, Super
- Classes
- Object Class(only Important Methods)
- String Class (Only Important Methods)
- String Buffer & String Builder
- Wrapper Classes
- Exceptions
- Introduction - Why Exceptions
- Types of Exceptions
- Try catch and Finally Block
- Multi catch Exceptions
- Throw and Throws keywords
- Method Overriding with Exceptions
- Custom Exceptions
- FILE I/O
- What is Stream and Types of Stream
- File Input Output Streams and Its Methods
- File class

- Command Line Arguments
- Thread-Introduction
- Thread Life Cycle
- Creating Threads
- Thread Class Methods (Only Important Methods)
- Runnable Interface
- Synchronized block and Synchronized Methods
- Collection Framework - Introduction
- Collection API
- Hierarchy of Collections
- List and Set and Map Collections
- Array list, vector and Other Classes
- Generics
- Comparator and Comparables
- JAVA GUI
- AWT (Introduction only) & Swing (in Details)
- Components, Containers, Frame, Window, Panel, Layout
- All Components
- Events, Event Handling
- Practical Example : 1. Create class box with three variable height, width, depth. Create default, parameterized and copy constructor. Create one method called volume to show width*height*depth. Call all constructor and volume method for all constructor
- Practical Example : 1. Create class named student with variable rno, fname, lname, email, mobile. create 2 methods to get student data and print them.
- Practical Example : 1. One dimensional array(get data by scanner and print it). 2. Array array elements in asceding and desceding order
- Practical Example : 1. Perform constructor chaining
- Practical Example : 1. Print the current thread that is by default available and then change it's name and again print it. 2. Create a thread usinf Runnable interface. 3. Create a thread using Thread class. 4. Create multiple thread and execute it in main method. 5. Create multiple thread and execute them simultaneouly and achive synchronization. 6. Create two synchronized thread and perform deadlock
- Practical Example: 1. Create 2 two dimensional array and perform matrix addition, subtractio and multiplication
- Practical Example: 1. Create abstract class RBI with one abstract method interest rate and extend this class in three class SBI, HDFC, Kotak to implement abstract method. 2. Create interface with 2 method. Implement this method in two class. 3. Create program for inheritance of interface. 4. Create program to implment static method in interface and call it in a class
- Practical Example: 1. Create custome excepton insufficientfund. Create class named bank and create two methods deposite and withdraw. If withdrawal amount is greater than balance then throw user defined exception and handle it.
- Practical Example: 1. Create swing GUI with id, fname, lname & email and perform CRUD operation with mysql datanase.
- Practical Example: 1. Demonstrate the devide by zero, input mismatch exception and arrayindexoutofbounds exception in a multi catch and multi try statement. 2. Create a method called

demo and enter user defind integer value at runtime, if user enters negative value ask again to put value using recursion others wise throw an exception and handle it. 3. Create above program using throws clause without recusrion. 4. Demonstrate the finally block. 5. How to use exception in method override

- Practical Example: 1. Make a ArrayList with different type of data and perform it's different method. 2. Iterate ArrayList data in both direction from first to last and last to first. 3. Demonstrate HashSet with it;s method. 4. Demonstrate HashMap and iterate it's data. 5. Perform enumeration with Vector class. 6. Create generic method to print different types of array of diffent wrapper classes. 7. Demonstrate Comparator 8. Demonstrate Comparable.
- Practical Example: 1. Method overloading. 2. Method overriding 3. Dynamic method dispatch to solve method override
- Practical Example: 1. Pass the 2 integer values through command line and print the maximum number from this.
- Practical Example: 1. Perform singleinheritance. 2. Multilevel. 3. Hierarchical.
- Practical Example: 1. String class & its method 2.Perform StringBuffer class methods
- Practical Example: 1. Write a progra to write 1 string data into the file using FileOutputStream and read that file using FileInputStream.
- Practical Example: 1. Write a program to show the use of this keyword in assigning values to variable, as argument in constructor and method, call the default constructor in parameterized constructor using this, and call the method using this. 2. Write a program to demonstrate the difference between static and non static variable. 3. Write a program to to create a static method and static block. 4. Demonstrate teh use of final variable, mthod and class. 5. Access the variable, methods and constructor from d
- Practical Example: 2. Do above operation using FileWritter & FileReader
- Practical Example: 3. Create one class name student with rno, fname, lname & email and store values of variable into object and then write that object into file and read it.
- Practical Example: 4. Print all the basic property of file that is available in your c:\ drive. You create tops1.txt and put some text into it.

Module 14) Java - RDBMS & Database Programming With JDBC	**6**

- Database
- DBMS and RDBMS
- Introduction MYSQL
- Mysql IDE
- Query Types
- DDL, DML, DQL, DCL
- Constraints : Primary Key, Foreign Key, Unique Key
- Normalizations: 1NF 2NF 3NF
- Joins: All Joins Types
- Advance Database: Indexers Views Procedures Functions Cursor, Triggers
- JDBC (Insert, Update, Select, Delete)
- Introduction of JDBC

- Driver Types
- Steps for Creating Connections
- Types of Statements (Statements, prepared Statements and Callable Statements)
- Result Set Interface
- Database Metadata
- Result Set Metadata
- Practical Examples: SQL Queries
- Practical Example : 1. Create swing GUI with id, fname, lname & email and perform CRUD operation with mysql datanase. 2. Demonstrate callble statement in & out parameter.

Module 15) Java - Web Technologies In Java	**6**

- HTML UL, Tag LI, Tag a, Tag IMG, tag Table, TR, TD, tag
- Form tags with Attributes
- All input tags CSS
- Types of CSS Pseudo- Classes Margins and Puddings
- CSS background
- CSS using ID and Class
- JavaScript Events
- Validations with Regular Expressions
- Firebug Template Integration
- Practical Example: 1. Basic HTML Tags 2. Create Registration form and perform required and regular expression validation for firstname(only alphabets allowd), email(standard email id), mobile number(only 10 digits). 3. Perform all type of css, class & id, pseudo code.
- Introduction of Client Server Architecture
- HTTP Protocol overview with Request and Response header explanation
- J2EE Architecture Overview
- Web Component Development In Java CGI Programming Process Advantage and Disadvantage
- Servlet Programming Introductions Advantage and Disadvantage
- Servlet Versions, Types of Servlets
- Difference between HTTP Servlet an Generic Servlet
- Servlet Life Cycle
- Creating Servlets Servlet Entry in web.xml
- Logical URL Servlet Config Interface
- Request Dispatcher Interface Forward and Include Methods
- Request Dispatcher Interface
- Servlet Context Interface Web Application Listener Scope of Objects, Request and Response Application (Context)
- Practical Example: 1. Fetch data from web.xml to particular servlet using ServletConfig. 2. Fetch data from web.xml to multiple servlet using ServletContext. 3. Create one registration form in jsp and send data to servlet, from servlet again send data to jsp using RequestDispatcher. 4. Create login form in jsp and after login send uname & password to servlet, check data if not blank go forward and if blank then include login.jsp page to servlet.

- Java Filters - Introduction What are the needs Filter Life Cycle Process of Execution Filter Applying Filter Entry in web.xml URL Pattern with Filter
- Practical Example: 1. Perform server side validation using filter.
- Action JSTL Custom Tags
- Comments
- Declaration Implicit Objects
- Directives - Scriplets
- Expression
- JSP Life Cycle
- JSP Translation
- Practical Example: JSP Translation JSP Life Cycle Comments Directives Scrip lets Expression Declaration Implicit Objects Action JSTL Custom Tags
- Cookies Session
- Hidden Form Fields
- Session Management - Introduction
- Session Tracking Technique
- URL Rewriting
- What are needs?
- Practical Example: 1. Create registration form, after validation insert data to database and redirect to login form, if successfull login manage session data and logout. 2. Create complete CRUD operation for user profile management.

Module 16) Java - Software Design Pattern And Project	**7**

- Software Design Pattern and Project MVC + DAO
- Session Management (Session, Cookie, Hidden Form Field , URL Rewriting)
- Project Cover Topics: 1) Template Integration 2) Image Up/Dwn 3) Mail Integration 4) OTP Via Mail Integration 5) Online Payment Integration 6) AJAX

Module 17) Java - Hibernate Framework	**5**

- Introduction Hibernate Architecture
- Hibernate Relationship(one to one, one to many , many to one , many to many)
- Hibernate CRUD Example

Module 18) Java - Spring	**5**

- Introduction
- BeanFactory and application Context
- Container Concepts
- Spring Data JPA Template
- MVC

Module 19) Java - Spring Boot	5

- Introduction to STS
- MVC (Template Integration, CRUD , Form Validation , Pagination), AOP
- Security, Role based Authentication, OAuth2, Token based authentication

Module 20) Java - Spring WebServices	6

- Introduction to WebService
- Basics of REST APIs
- MVC, AOP
- Spring REST(CRUD API, Pagination, Fetch from Multiple Table , Image Up/API)

Module 21) Java - Microservices with Spring Boot, Spring Cloud	6

- Microservices with Spring Boot, Spring Cloud
- Introduction to MicroService architecture
- Advantages with MicroService over Monolithinc architecture
- Develop and Deploy Microservice application in localhost Introduction to Service Discovery ,Eureka server
- Client side Discovery pattern
- Server side Discovery pattern
- Load Balancing configuration

Module 3) WD - HTML	10

- Student Intro , Career Center Login ,What is Internet, HTTP/HTTPS, WWW, Domain name and Top Domain name
- SEO, What is HTML, What is Text Editor, Web Browser, Downloading Text Editor , HTML Structure, First Program in HTML
- 1) HTML Introduction 2) HTML Getting Started 3) HTML Elements 4) HTML Attributes 5) HTML Basic Tags
- 1) HTML Doctypes 2) HTML Layout 3) HTML Head 4) HTML Meta 5) HTML Scripts
- Practical Examples: 1) Create any simple web page to display your name. 2) Importance of meta tag and Doctypes
- Tags and self Closing Tags, Basic Tag , Attribute and Events, Marquee Tag
- HTML - Meta Tags, HTML - Comments, HTML - Images, HTML - Tables, HTML - Lists, HTML - Text Links, HTML - Image Links
- HTML Headings HTML Paragraphs HTML Links HTML Text Formatting HTML Styles HTML Images
- HTML - Frames, HTML - Iframes, HTML - Blocks, HTML - Backgrounds, HTML - Colors, HTML - Fonts
- Anchor Tag, Img Tag, Image Mapping

- HTML - Fonts, HTML - Forms, HTML - Embed Multimedia ,HTML - Marquees, HTML - Header, HTML - Style Sheet, HTML - Javascript ,HTML - Layouts
- List Tag, Tables, Forms
- HTML - Tags Reference, HTML - Attributes Reference, HTML - Events Reference, HTML - Fonts Reference, HTML - ASCII Codes, ASCII Table, Lookup, HTML - Color Names, HTML - Entities, HTML - Fonts, Ref HTML - Events, Ref MIME Media Types, HTML - URL Encoding Language, ISO Codes HTML - Character Encodings, HTML - Deprecated Tags
- PRactical Examples: 1) Create simple Doc and display your name using different heading tag 2) Create link for open google. 3) Create document using all text formatting tags
- HTML online editor
- HTML Tables HTML Lists HTML Forms HTML Iframes
- Practical Examples: 1) Create simple table 2) Create time table for your school 3) Create table with colspanrowspan example 4) Create invoice using table 5) Create hotel menu. 6) Create index page for your book. 7) Create list with different categories.
- PRactical Examples: Create registration form with all fields and validation

Module 4) WD - CSS and CSS 3	**20**

- 1) CSS 2) In-line CSS Internal Style External Style Sheet @import Style Sheet 3) CSS Class CSS ID
- What is CSS How to Implement CSS Class and ID Width and Height Css Unit Box Model (Margin,padding,Border) and create basic template design
- Practical example : Create page with difference color text
- CSS Selectors , Pseudo Classes and Elements , Float and Clear and Alignment , Font Styling , Opacity and Visibility , Line Height
- 1) CSS Text 2)CSS Font 3) CSS Background 4) CSS Links 5) CSS Lists 6) CSS Display 7) CSS Visibility
- Creating Header of Website , Outline , Background , Counter increment , Counter reset ,Cursor , Overflow
- PRactical Example : Create layout for your project
- Position , Creating Submenu , Border Radius, Transform , Animation , Font Awesome Icons
- 1) CSS Layout Model 2) CSS Border 3) CSS Margin 4) CSS Padding 5) CSS Outline
- Font Family Through Google Font , import fontface rule ,FlexBox
- 1) CSS Float 2) CSS Align 3) CSS Position 4) CSS Element Size 5) CSS Layer
- Practical Example : Create image gallery
- 1) CSS Pseudo Class Selector 2) CSS Pseudo Element Selector
- CSS Properties 1) Background, 2) border 3) bottom 4) caption-side 5) clear 6) clip 7) color 8) content
- Practical Example: Create Menu with logo at left side and contact info at right side using clear effect
- 1) counter-increment 2) counter-reset 3) cursor 4) direction 5) display 6) empty-cells
- Practical Example: 1) Create submenu list using counter
- 1) float 2) font 3) height 4) left 5) letter-spacing 6) line [height, style, style-7) image, style-position, 8) style-type] 9) margin 10) outline 11) overflow 12) padding
- 1) page-break 2) position 3) quotes 4) right 5) table-layout 6) text 7) top 8) vertical-align 9) visibility 10) white-space 11) width 12) word-spacing 13) z-index
- Practical Example: wireframe layout for your template using div

- Responsive Design Principles, Media Query (For Responsive Website) , Creating a Responsive Website
- Validate a Website, Hosting a website with free domain name, Column , Clippath , Gradient Color , Filter, Border Image

Module 5) Website Designing - HTML5	**5**

- HTML5 Tags, HTML5 Input and Attribute
- Audio and Video, Semantic Element in HTML5
- Canvas, Svg
- Display Grid

Module 5) WD – Advance CSS/CSS Preprocessors	**10**

- Sass Variables
- strings · numbers
- colors · booleans
- lists
- nulls Sass Nested Rules
- Sass Importing Files
- Sass Mixins
- Sass @extend Directive
- ?Introduction to Sass: Explaining the basics of Sass, its syntax, and how it enhances traditional CSS.
- Variables and Variable Scope: Discussing the use of variables in Sass and how variable scope works.
- Mixins and @include: Explaining mixins and how they can be used for reusable styles, along with the @include directive.
- Nested Rules: Discussing nested rules in Sass and how they improve code organization and readability.
- Control Directives: Covering Sass control directives such as @if, @for, and @each, and how they can be used for conditional styles and loops.
- ?Functions: Explaining the use of functions in Sass for tasks like color manipulation, math operations, and more.
- Partials and @import: Discussing the use of partials to split Sass code into smaller, modular files, and how to import them using the @import directive.
- Extend/Inheritance: Explaining how to use the @extend directive in Sass for extending styles, and discussing its benefits and potential pitfalls.
- Operators: Covering Sass operators for arithmetic, logical operations, and string manipulation.
- Sass Maps: Introducing Sass maps and how they can be used for managing key-value pairs and organizing data.
- Mixins Libraries: Introducing popular Sass libraries like Bourbon or Compass and discussing their features and benefits
- ?Best Practices: Providing tips and best practices for writing clean, efficient, and maintainable Sass code.
- Less

- Variables
- Mixins
- Nesting
- Operations
- calc() exception
- Escaping
- Functions
- Namespaces and Accessors
- Maps
- Scope
- Comments
- Importing
- Introduction to Less CSS: Explaining the basics of Less CSS, its syntax, and how it extends traditional CSS
- Variables: Discussing the use of variables in Less CSS and how they contribute to code maintainability and reusability.
- Mixins and Functions: Explaining how mixins and functions work in Less CSS, and how they can be used to encapsulate styles and perform operations.
- Nesting: Discussing the nesting feature in Less CSS and its benefits for organizing styles and improving readability.
- Operations and Functions: Covering the various operations and functions available in Less CSS for arithmetic, color manipulation, and more.
- Imports and Partials: Explaining how imports and partials work in Less CSS to split stylesheets into smaller, more manageable files.
- Control Directives: Covering control directives like @if, @for, and @each in Less CSS and how they can be used to create dynamic stylesheets.
- Extend: Discussing the @extend directive in Less CSS and how it allows styles to be inherited from one selector to another.
- ?Nested Rules and Parent Selectors: Explaining nested rules and the use of the & selector in Less CSS for generating complex selectors.
- ?Mixins Libraries: Introducing popular Less CSS libraries like LessHat or Bootstrap Less and discussing their features and benefits.
- ?Best Practices: Providing tips and best practices for writing clean, efficient, and maintainable Less CSS code.
- ?SCSS (Sassy CSS) : - · Introduction to SCSS: Explaining what SCSS is, how it extends CSS, and its benefits. ·
- Variables and Mixins: Discussing the use of variables and mixins in SCSS to improve code reusability and maintainability.
- Nesting: Explaining how nesting works in SCSS and its advantages in organizing styles.
- Partials and Importing: Discussing the use of partials to split SCSS code into smaller files and the process of importing them into a main SCSS file.
- Control Directives: Covering the use of control directives like @if, @for, and @each to create more dynamic and efficient stylesheets.

- Functions: Explaining how functions in SCSS can be used to perform calculations, manipulate colors, and more.
- Inheritance and Extends: Discussing how inheritance can be achieved in SCSS using the @extend directive and its implications.
- Operators: Covering the various operators available in SCSS for performing arithmetic, logical, and other operations. ·
- Mixins Libraries: Introducing popular SCSS libraries like Bourbon or Compass and discussing their features and benefits.
- ?Best Practices: Providing tips and best practices for writing clean, efficient, and maintainable SCSS code.

Module 7) WD - JQuery Basic, Effects & Advanced	**8**

- jQuery Basic a) jQuery Introduction b) jQuery Getting Started c) jQuery Syntax d) jQuery Selectors e) jQuery Events
- What is JQuery , Downloading JQuery File , First Program in JQuery
- Practical Example: Change CSS
- JQuery Syntax , Query Selector, Hide , Slide , Fade Effect in JQuery
- JQuery Effects 1) jQuery Show/Hide 2) jQuery Fade 3) jQuery Slide 4) jQuery Animation 5) jQuery Stop 6) jQuery Chaining 7) jQuery Callback
- How to Apply CSS Using JQuery, How to Add Class and Remove Class in Jquery , JQuery Animation
- Practical Example: Create slider with animation
- Filter using JQuery , JQuery Slider Plugin , Validation Plugin
- JQuery Advanced 1) jQuery Traversing 2) jQuery Ancestors 3) jQuery Descendants 4) jQuery Siblings 5) jQuery Filtering 6) jQuery Load 7) jQuery No-Conflict
- Zoom Plugin, Now Make Your Existing Website Dynamic with Javascript and JQuery
- Ajax and ajax get post

Module 8) JavaScript Essentials And Advanced	**25**

- Basic JavaScript, Js comment, Js variables , Understanding var, let and Const, JS switch, if, else,JS loop , Js global variables, Js data types, Js operators, Js Functions
- Functions - Function Declaration in JS - Arrow Functions - Higher Order Functions - Map, Reduce and Filter
- Javascript Objects, Js object , Js Array , Js string, Js Date, Js Math, Js number, Js Boolean
- Javascript BOM ,Broswer Objects , Window object, History object, navigator object, Screen object
- Javascript DOM, Document object, getElementById, getElementByName, getElementByTagName, JS innerHTML property, JS innerTEXT property
- Javascript OOPS, JS class, JS object, JS prototype, JS constructor method, JS static method, JS encapsulation, JS inheritance, JS polymorphism, JS abstractions
- Javascript Exception Handling, JS exception handling , Javascript try-catch
- Javascript MISC, JS this keyword , JS Debugging , JS Hoisting , JS Strict Mode, JS promises, JS typeof , JS ternary operator, JS reload() method, JS setAttributes () method, JS setInterval() method,

JS setTimeout() method.
- Javascript Events, Javascript Events, Javascript AddEventListener,(), jsOnclick event, jsdbclick event, JS onload event, JS onresize event.
- Array in JS, Creating Array, Array methods, The Spread & Rest operators, Destructuring
- JS Async, Callbacks, Promises, Async/Await
- ES6 Basics and Babel, New features in ES 6, Arrow functions, The . Operator, For/of , Map Objects, Set Objects, Promises, Functions Rest parameter, String.includes(),String.starts.With(), String.endWith(), Array.form(), Array.keys(), Array find(), Array findIndex(), javascript Modules
- Small Project using ES6
- Introduction to JavaScript Frameworks Overview of popular frameworks (React, Angular, Vue) Why use a framework?

Module-9) React - Components, State, Props	**8**

- Installation - Add React to a HTML Website - Create New React App - Hello World
- •Different ways to installation – Create first project in react - Add React to a HTML Website - Create New React App - Hello World
- Getting started in React
- •React Js Introduction •Project Structure Overview
- JSX
- •React Dom – Render HTML •SPA vs Traditional Web app
- Components
- Component Composition
- •Convert HTML to JSX
- JSX - Why JSX? - Embedding Expressions in JSX - Attributes with JSX - Children with JSX
- Props & Prop Types
- Types of Components, Difference between Function Components and Class components
- Event Handlers
- Class Lifecycle
- State
- Import - Export Components
- React Web App
- Working with Multiple components in same file, components nesting and composition
- Components, State, Props - Function Component - Class Component - Props - State - Class Component Lifecycle
- Conditional Rendering
- •Ternary , if, if..else , switch case
- State and its importance
- •Commonly used events (onclick, onchange..)
- •Event Handlers
- •UseState – Hook introduction
- •Setstate in class components
- •Practical : accept value from user and display in h1 , create calculator app using state , hide – unhide div on button click

- •Prop and prop types
- •Updating Array in state
- •Updating object in state
- Basic understanding of Css – style

Module 10) Lists , Hooks , Localstorage , Api Project	6

- Conditional Rendering - Lists and Keys - Forms - Handling Events - Lifting State up
- •Rendering List in React
- Hooks - Introduction - Using the State hook - Using the Effect hook - Rules of Hook - Custom Hook
- •Use of Map() function
- Rendering Lists inside components
- •Keys and importance in lists
- React Keys
- •useRef
- Using keys wit component
- •React Refs
- Uniqueness of keys among siblings
- •Uses of React Refs
- React refs
- •useEffect practical
- Uses of react Refs
- •useContext
- How to access of Refs
- •useReducer
- Refs current properties
- •useCallback
- Add Refs to DOM elements
- •useMemo
- Add refs to class components
- •Custom hook
- Callback refs
- •Forms and Form handling
- Forwarding Ref from one component to another component
- •Form submission and validation
- React with useRef
- •React Hook form
- React conditional rendering
- •Project : Perform CRUD operations using array or map , render all records from array or map and display in table apply edit and delete operations – don’t use any databases
- React if, logical & operator, Ternary operator, switch case operator, Conditional Rendering with Enum, Preventing components from rendering
- •JSON.parse and JSON.
- •Project : Perform CRUD operations using local storage database

- Project : Task manager application , pending task , completed task , select all , deselect all functionality

Module-11) React -Advance React- Styling , Routing	5

- Creating the first App
- •React bootstrap
- Understanding the App
- •Tailwind in React
- Styling the App
- •Material UI in react js
- Inspecting & Debugging styles
- •Advantages of material ui in react js
- Built-in components
- material UI components implementation
- Working with Images
- •Buttons, Grid system , stack , typography , icons
- ListViews
- Customize style using material ui theme
- TextInput
- Styling React Components - CSS stylesheet - Inline Styling - CSS Modules - CSS in JS Libraries (styled components)
- •Appbar
- •Layout components
- Creating Views (Scenes)
- Drawer , grid layout
- Conditional Rendering - Lists and Keys - Forms - Handling Events - Lifting State up
- •Form Components
- Hooks - Introduction - Using the State hook - Using the Effect hook - Rules of Hook - Custom Hook
- Advance Concepts - Context, useContext() - Working with Refs and useRefs() - Fragments - Performance optimization with useMemo() - Styling React Components - CSS stylesheet - Inline Styling - CSS Modules - CSS in JS Libraries (styled components)
- •Textfields , checkboxs , Form validation
- Bootstrap with React
- •Dialogs and modals
- React Router - Browser - Router - Link - Route - Template integration - Http Request in React - Get and Post data
- •React Router – Browser – Router - Link , navlink
- •Route Parameters – useParams
- •Nested Routes
- Programmatic Navigation – history
- •Lazy loading (performance optimization)
- •Authentication – authorization
- •Redirecting unauthorized used to login page

- Using css transitions or animation library like React transition group.
- •Error Handling – 404 Page not found
- •Project : Project using API Or FakeJsonAPI
- Api testing in Postman

Module 12)React – JSON-server and Firebase Real Time Database	8

- React Router
- •Installing Firebase SDK in react project
- B r o w s e r - R o u t e r - Lin k - R o u t e
- Need of react router
- •NoSql Database in firebase
- •Creating collection and documents
- T e m pla t e in t e g r a tio n - H t t p R e q u e s t in R e a c t - G e t a n d P o s t d a t a
- React router installation
- •Authentication with firebase
- React router, react-router-native, react-router-Dom
- Use of firebase storage
- Component in react router , Browser Router , HashRouter
- Project : Create app which contain registration , login and profile updation using firebase , user can post image and other user's can like dis-like post
- What is Route
- Project2: Web app using JSON-server
- What is Link component , Adding navigation using Link component
- Link vs NavLink
- React Router Switch , React Router redirect
- Nested Routing in React
- Template integrations Using Browser Router , Routes , Route , Link and Hash Router
- Advantages of react Router

Module-13) React - Applying Redux	8

- State
- State storage problem
- Redux Basics
- Redux Principles
- Implementing Redux
- React-Redux
- •Redux Core concepts – Actions , Reducers , Store React-Redux
- Middleware
- Counter App Demo
- Redux - Complexity of Managing state - Understand the Redux Flow - Setting up Reducer and store - Dispatching Actions - Passing and Retrieving Data with Action - Combining Multiple Reducers - Adding Middleware - Redux Dev tools

- •Redux-thunk middlerware
- Project : Perform CRUD operations using React-Redux

Module 30) Java - Rest Framework - Industry	10

- Design Pattern
- MVC Design Pattern with Example
- AJAX Programming With Example
- Practical Example: 1. Perform dynamic search operation in project using AJAX. 2. Register user with unique email using AJAX
- Introduction to Distributed Technologies RMI, EJB and WEB Services Introduction Types of Web Services What is Restful Web Services? Restful Web Services Annotations Restful Web Services with Exampl
- ractical Example: 1. Restful web service CRUD operation
- RESTful API: Representational State Transfer (REST) is a widely used architectural style for building web services. Understanding REST principles and being able to create RESTful APIs is essential. CRUD API: CRUD stands for Create, Read, Update, and Delete, which are the basic operations performed on data. Creating APIs that allow these operations is fundamental to backend development. Authentication and Authorization API: Knowing how to implement user authentication and authorization mechanisms is crucia
- OpenWeatherMap API: This API provides weather data for various locations worldwide. You can retrieve current weather conditions, forecasts, and historical weather data.
- Google Maps Geocoding API: This API allows you to convert addresses into geographic coordinates (latitude and longitude) and vice versa. You can use it to retrieve location data, calculate distances between points, and display maps.
- GitHub API: GitHub provides an API that enables you to interact with repositories, issues, pull requests, and more. You can perform actions like retrieving repository information, creating issues, and accessing user data.
- Twitter API: Twitter offers an API that allows you to integrate Twitter functionality into your applications. You can fetch tweets, post tweets, retrieve user information, and perform searches.
- REST Countries API: This API provides information about countries, including details like population, languages spoken, currencies, and more. You can retrieve country-specific data and use it for various applications.
- Social authentication (For eg; Login with Google, Login with Facebook...etc)
- Email sending APIs (For eg; Mailchimp, Mailgun...etc)
- SMS sending APIs (For eg; Twilio)
- Normal payments (For eg; Paypal, Stripe)
- - Google Maps API

Module 31) java - Frameworks - Industry	16

- All Core Interface Query and Criteria Named Query
- Relationships Many to Many
- Relationships One to Many

- Relationships Many to One
- Hibernate Introduction
- Relationships One to One
- Hibernate Architecture
- All Database Operations with hibernate
- Practical Example: 1. CRUD Operation with hibernate using xml files. 2. CRUD operation with hibernate using annotation. 3. Perform onetoone relationship(Employee class with eid, uname and password & EmployeePersonalINfo class with epid, fname, lname,email). 4. Perform onetomany & manytoone relationship(Employee class with eid, fname,lname,email & Department class with deptno, dname,location). 5. Perform manytomany relationship(Student class with sid, sname & Course class with cid, cname)
- Introduction of Spring Framework Architecture
- Overview Of Spring Framework
- Core Container AOP
- Spring DAO (Data Integration)
- Spring Using IDE, Using Library Spring Hello World Example
- Practical Example: 1. Hello world spring app to introduce spring framework
- 1) Spring IOC Container 2) Bean Factory 3) Application Context Spring Bean Definition 4) Configuration 5) Life Cycle 6) Inheritance 7) Scopes
- Practical Example: 2. Perform spring inheritance, life cycle & abstraction. 3. Perform singleton & prototype scope to use spring beans variable
- 1) Spring Dependency Injection 2) Constructor based 3) Setter Getter based 4) Inner Beans , Aliases and ID-ref Collections and References 5) Auto Wiring
- Practical Example : 4. Demonstrate spring dependency injection by setter method.5. Demonstrate spring dependency injection by constructor. 6. Demonstrate spring dependency injection by object. 7. Perform inner bean concept in xml file. 8. Use all type of collection refences in spring xml file. 9. Minimize spring xml file using spring auto wire concept
- 1) Spring AOP 2) AOP Term 3) Write the Aspects 4) Configure Where the Aspects
- Practical Example : 10. Perfomr AOP(aspect oriented programming concept(login, perform, logout sequence)
- Spring ORM
- Practical Example : 11. Perform CRUD operation in spring web using hibernate integration
- 1) Spring MVC Web Forms 2) Spring Form Handling 3) Spring Form Tags 4) Spring Controller XML and Annotation Based
- Practical Example : 12. Create spring MVC pattern using dispatcher servlet. 13. CRUD operation using Spring MVC+ORM
- Spring MVC with Session Management
- Practical Example : 14. Spring MVC+ORM+Session