JAVA FULL STACK

TABLE OF CONTENTS

CHAPTER-1

SPRING BOOT

1.1 Spring Boot

Spring Boot Tutorial provides basic and advanced concepts of Spring Framework. Our Spring Boot Tutorial is designed for beginners and professionals both.

Spring Boot is a spring module that provides the RAD (Rapid Application Development) feature to the spring framework.

Our Spring Boot Tutorial includes all topics of Spring Boot such, as features, project, maven project, starter project wizard, Spring Initializer, CLI, applications, annotations, dependency management, properties, starters, Actuator, JPA, JDBC, etc.

What is Spring Boot?

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

Spring Boot

Spring Boot Tutorial provides basic and advanced concepts of Spring Framework. Our Spring Boot Tutorial is designed for beginners and professionals both.

Spring Boot is a spring module that provides the RAD (Rapid Application Development) feature to the spring framework.

Our Spring Boot Tutorial includes all topics of Spring Boot such, as features, project, maven project, starter project wizard, Spring Initializr, CLI, applications, annotations, dependency management, properties, starters, Actuator, JPA, JDBC, etc.

1.2 Spring Boot Architecture

Spring Boot is a module of the Spring Framework. It is used to create stand-alone, production-grade Spring Based Applications with minimum efforts. It is developed on top of the core Spring Framework.

Spring Boot follows a layered architecture in which each layer communicates with the layer directly below or above (hierarchical structure) it.

Before understanding the Spring Boot Architecture, we must know the different layers and classes present in it. There are four layers in Spring Boot are as follows:

Presentation Layer

Business Layer

Persistence Layer

Database Layer

Presentation Layer: The presentation layer handles the HTTP requests, translates the JSON parameter to object, and authenticates the request and transfer it to the business layer. In short, it consists of views i.e., frontend part.

Business Layer: The business layer handles all the business logic. It consists of service classes and uses services provided by data access layers. It also performs authorization and validation.

Persistence Layer: The persistence layer contains all the storage logic and translates business objects from and to database rows.

Database Layer: In the database layer, CRUD (create, retrieve, update, delete) operations are performed.

1.3 Spring Boot Dependency Management

Spring Boot manages dependencies and configuration automatically. Each release of Spring Boot provides a list of dependencies that it supports. The list of dependencies is available as a part of the Bills of Materials (spring-boot-dependencies) that can be used with Maven. So, we need not to specify the version of the dependencies in our configuration. Spring Boot manages itself. Spring Boot upgrades all dependencies automatically in a consistent way when we update the Spring Boot version.

Advantages of Dependency Management

It provides the centralization of dependency information by specifying the Spring Boot version in one place. It helps when we switch from one version to another.

It avoids mismatch of different versions of Spring Boot libraries.

We only need to write a library name with specifying the version. It is helpful in multi-module projects.

Spring Boot Starter Web

Starter of spring web uses Spring MVC, REST and Tomcat as a default embedded server. The single spring-boot-starter-web dependency transitively pulls in all dependencies related to web development. It also reduces the build dependency count. The spring-boot-starter-web transitively depends on the following:

org.springframework.boot:spring-boot-starter

org.springframework.boot:spring-boot-starter-tomcat

org.springframework.boot:spring-boot-starter-validation

com.fasterxml.jackson.core:jackson-databind

org.springframework:spring-weborg.springframework:spring-webmvc

1.4 Spring Data JPA

Spring Data is a high-level Spring Source project. Its purpose is to unify and easy access to the different kinds of persistence stores, both relational database systems, and NoSQL data stores.

When we implement a new application, we should focus on the business logic instead of technical complexity and boilerplate code. That's why the Java Persistent API (JPA) specification and Spring Data JPA are extremely popular.

Spring Data JPA adds a layer on the top of JPA. It means, Spring Data JPA uses all features defined by JPA specification, especially the entity, association mappings, and JPA's query capabilities. Spring Data JPA adds its own features such as the no-code implementation of the repository pattern and the creation of database queries from the method name.

Spring Boot AOP

The application is generally developed with multiple layers. A typical Java application has the following layers:

Web Layer: It exposes the services using the REST or web application.

Business Layer: It implements the business logic of an application.

Data Layer: It implements the persistence logic of the application.

The responsibility of each layer is different, but there are a few common aspects that apply to all layers are Logging, Security, validation, caching, etc. These common aspects are called cross-cutting concerns. If we implement these concerns in each layer separately, the code becomes more difficult to maintain. To overcome this problem, Aspect-Oriented Programming (AOP) provides a solution to implement cross-cutting concerns.

Implement the cross-cutting concern as an aspect.

Define pointcuts to indicate where the aspect has to be applied.

Benefits of AOP

It is implemented in pure Java.

There is no requirement for a special compilation process.

It supports only method execution Join points.

Only run time weaving is available.

Two types of AOP proxy is available: JDK dynamic proxy and CGLIB proxy.

Spring Boot Thymeleaf

The Thymeleaf is an open-source Java library that is licensed under the Apache License 2.0. It is a HTML5/XHTML/XML template engine. It is a server-side Java template engine for both web (servlet-based) and non-web (offline) environments. It is perfect for modern-day HTML5 JVM web development. It provides full integration with Spring Framework.

It applies a set of transformations to template files in order to display data or text produced by the application. It is appropriate for serving XHTML/HTML5 in web applications.

The goal of Thymeleaf is to provide a stylish and well-formed way of creating templates. It is based on XML tags and attributes. These XML tags define the execution of predefined logic on the DOM (Document Object Model) instead of explicitly writing that logic as code inside the template. It is a substitute for JSP.

The architecture of Thymeleaf allows the fast processing of templates that depends on the caching of parsed files. It uses the least possible amount of I/O operations during execution.

1.5 Spring Boot Caching

Spring Framework provides caching in a Spring Application, transparently. In Spring, the cache abstraction is a mechanism that allows consistent use of various caching methods with minimal impact on the code.

Cache Abstraction

The cache abstraction mechanism applies to Java methods. The main objective of using cache abstraction is to reduce the number of executions based on the information present in the cache. It applies to expensive methods such as CPU or IO bound.

Every time, when a method invokes, the abstraction applies a cache behavior to the method. It checks whether the method has already been executed for the given argument or not.

Spring Boot EhCaching

EhCache

EhCache is an open-source, Java-based cache used to boost performance. The current version of Ehcache is 3. It provides the implementation of the JSR-107 cache manager. We can use it directly.

Features of EhCache

It is fast, lightweight, Scalable, and Flexible.

It allows us to perform Serializable and Object

It offers cache eviction policies such as LRU, LFU, FIFO,

It stores the cache in memory and disk (SSD).

It depends on SLF4J for logging.

It has a full implementation of JSR-107 and Jcache

CHAPTER-2

HTML

2.1 What is HTML

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

<article> This element is used to define an independent piece of content in a document, that may be a blog, a magazine or a newspaper article.

<aside> It specifies that article is slightly related to the rest of the whole page.

<audio> It is used to play audio file in HTML.

<dialog> It defines a window or a dialog box.

<footer> It defines a footer for a section.

<header> It defines a header for a section.

<main> It defines the main content of a document.

<nav> It is used to define the navigation link in the document.

<mark> It specifies the marked or highlighted content.

<meter> It is used to measure the scalar value within a given range.

<ruby> It defines ruby annotation along with <rp> and <rt>.

<section> It defines a section in the document.

<summary> It specifies a visible heading for <detailed> element.

2.2 HTML Ordered List | HTML Numbered List

HTML Ordered List or Numbered List displays elements in numbered format. The HTML ol tag is used for ordered list. We can use ordered list to represent items either in numerical order format or alphabetical order format, or any format where an order is emphasized. There can be different types of numbered list:
Numeric Number (1, 2, 3)
Capital Roman Number (I II III)
Small Romal Number (i ii iii)
Capital Alphabet (A B C)
Small Alphabet (a b c)

2.3 HTML Unordered List or Bulleted List displays elements in bulleted format . We can use unordered list where we do not need to display items in any particular order. The HTML ul tag is used for the unordered list. There can be 4 types of bulleted list:

- Disc
- Circle
- Square
- none

2.4 HTML Form

An HTML form is *a section of a document* which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.
An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc.
Text = Defines a one-line text input field
Password = Defines a one-line password input field
Submit = Defines a submit button to submit the form to server
Reset = Defines a reset button to reset all values in the form.
Radio = Defines a radio button which allows select one option.
Checkbox = Defines checkboxes which allow select multiple options form.
Button = Defines a simple push button, which can be programmed to perform a task on an event.
File = Defines to select the file from device storage.
Image = Defines a graphical submit button.

2.4.1The Target Attribute

The target attribute specifies where to display the response that is received after submitting the form.

The target attribute can have one of the following values:

Value	**Description**
_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window
framename	The response is displayed in a named iframe

The default value is _self which means that the response will open in the current window.

CHAPTER-3

CSS

CSS tutorial or CSS 3 tutorial provides basic and advanced concepts of CSS technology. Our CSS tutorial is developed for beginners.

CSS stands for Cascading Style Sheet.

CSS is used to design HTML tags.

CSS is a widely used language on the web.

HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

3.1 CSS

CSS tutorial or CSS 3 tutorial provides basic and advanced concepts of CSS technology. Our CSS tutorial is developed for beginners.

CSS stands for Cascading Style Sheet.

CSS is used to design HTML tags.

CSS is a widely used language on the web.

HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

3.1.1 CSS Syntax

A CSS rule set contains a selector and a declaration block.

Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

Declaration Block: The declaration block can contain one or more declarations separated by a semicolon.

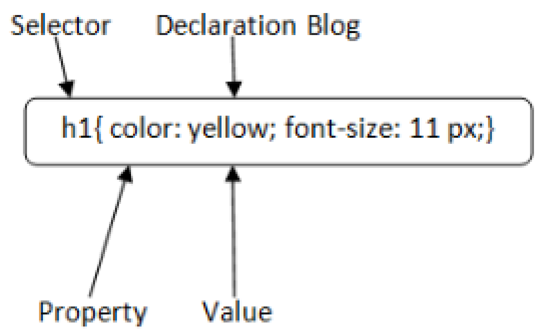
For the above example, there are two declarations:

color: yellow;

font-size: 11 px;

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.



3.2 CSS Grid

A grid can be defined as an intersecting set of horizontal lines and vertical lines. CSS Grid layout divides a page into major regions. It defines the relationship between the parts of a control built from HTML primitives in terms of layer, position, and size. Grid property offers a grid-based layout system having rows and columns. It makes the designing of web pages easy without positioning and floating.

3.3 CSS Layout

CSS layout is easy to design. We can use CSS layout to design our web page such as home page, contact us, about us etc.

There are 3 ways to design layout of a web page:

HTML Div with CSS: fast and widely used now.

HTML Table: slow and less preferred.

HTML Frameset: deprecated now.

CSS Table

We can apply style on HTML tables for better look and feel. There are some CSS properties that are widely used in designing table using CSS:

border

border-collapse

padding

width

height

text-align

color

background-color

3.4 Types of CSS

3.4.1 Inline CSS

Inline CSS is a method of styling where CSS properties are directly applied to HTML elements within the body section by using the “style” attribute. This method enables localized and specific styling for each element, which enhances the control over their presentation.

Example: This example shows the use of inline CSS with the help of an HTML document.

```html
<!DOCTYPE html>
<html>

<head>
    <title>Inline CSS</title>
    <style>
        p {
            color:#009900;
            font-size:50px;
            font-style:italic;
            text-align:center;
            }
    </style>

</head>

<body>
    <p>
        GNITC
    </p>
</body>

</html>
```

3.4.2 Internal CSS

This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e. the CSS is embedded within the <style> tag inside the head section of the HTML file.

Example: This example shows the use of internal CSS with the help of an HTML document.

```
<!DOCTYPE html>
<html>

<head>
    <title>Internal CSS</title>
    <style>
        .main {
            text-align: center;
        }

        .GFG {
            color: #009900;
            font-size: 50px;
            font-weight: bold;
        }

        .geeks {
            font-style: bold;
            font-size: 20px;
        }
    </style>
</head>

<body>
    <div class="main">
        <div class="GFG">GNITC</div>

        <div class="geeks">
            A computer science portal for geeks
        </div>
    </div>
</body>

</html>
```

3.4.3 External CSS

External CSS contains separate CSS files that contain only style properties with the help of tag attributes (For example class, id, heading, … etc). CSS property is written in a separate file with a .css extension and should be linked to the HTML document using a link tag. It means that, for each element, style can be set only once and will be applied across web pages.

Example: This example shows the use of external CSS with the help of an HTML document.

```html
<!DOCTYPE html>
<html>

<head>
    <title>Internal CSS</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="main">
        <div class="GFG">GNITC</div>

        <div class="geeks">
            A computer science portal for geeks
        </div>
    </div>
</body>

</html>
```

CHAPTER-4
ANGULAR JS

4.1 GENERAL

AngularJS tutorial provides basic and advanced concepts of AngularJS. Our AngularJS tutorial is designed for beginners and professionals.

Angular JS is an open source JavaScript framework by Google to build web applications. It can be freely used, changed and shared by anyone.

Our AngularJS tutorial includes all topics of AngularJS such as mvc, expressions, directives, controllers, modules, scopes, filters, dom, forms, ajax, validation, services, animation, dependency injection, views, w3.css etc. There are also given AngularJS interview questions to help you better understand the AngularJS.

AngularJS MVC Architecture

MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.

The MVC pattern is made up of the following three parts:

Model: It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.

View: It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.

Controller: It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

4.2 AngularJS Data Binding

Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

AngularJS follows Two-Way data binding model.

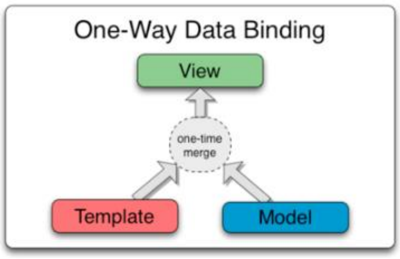
4.2.1 One-Way Data Binding

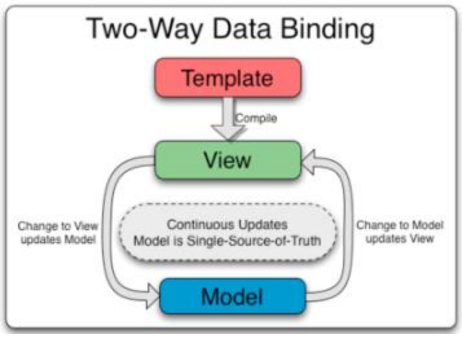
The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.

4.2.2 Two-Way Data Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.

Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.




Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates
Model is Single-Source-of-Truth
Change to Model updates View
Model

4.3 AngularJS Expressions

In AngularJS, expressions are used to bind application data to HTML. AngularJS resolves the expression, and return the result exactly where the expression is written.

4.3.1 AngularJS Directives

AngularJS facilitates you to extend HTML with new attributes. These attributes are called directives.

There is a set of built-in directive in AngularJS which offers functionality to your applications. You can also define your own directives.

4.3.2 AngularJS Controllers

AngularJS controllers are used to control the flow of data of AngularJS application. A controller is defined using ng-controller directive. A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts $scope as a parameter which refers to the application/module that controller is to control.

4.3.3 AngularJS Module

In AngularJS, a module defines an application. It is a container for the different parts of your application like controller, services, filters, directives etc.

4.3.4 AngularJS Scopes

The Scope is an object that is specified as a binding part between the HTML (view) and the JavaScript (controller). It plays a role of joining controller with the views. It is available for both the view and the controller.

4.4 AngularJS Dependency Injection

AngularJS comes with a built-in dependency injection mechanism. It facilitates you to divide your application into multiple different types of components which can be injected into each other as dependencies.

Dependency Injection is a software design pattern that specifies how components get holds of their dependencies. In this pattern, components are given their dependencies instead of coding them within the component.

AngularJS HTML DOM

In AngularJS, some directives can be used to bind application data to attributes of HTML DOM elements.

4.5 AngularJS Forms

AngularJS facilitates you to create a form enriches with data binding and validation of input controls.

Input controls are ways for a user to enter data. A form is a collection of controls for the purpose of grouping related controls together.

AngularJS Form Validation

AngularJS provides client-side form validation. It checks the state of the form and input fields (input, text area, select), and lets you notify the user about the current state.

It also holds the information about whether the input fields have been touched, or modified, or not.

Following directives are generally used to track errors in an AngularJS form:

$dirty - states that value has been changed.

$invalid - states that value entered is invalid.

$error - states the exact error.

AngularJS AJAX

AngularJS provides a $http service for reading data and remote servers. It is used to retrieve the desired records from a server.

AngularJS requires data in JSON format. Once the data is ready, $http gets the data form server in the **following manner:**

Here the file "data.txt" is employee's records. $http service makes an AJAX call and sets response to its property employees. This model is used to draw tables in HTML.

CHAPTER – 5
SQL TUTORIAL

5.1 SQL Tutorial

SQL tutorial provides basic and advanced concepts of SQL. Our SQL tutorial is designed for both beginners and professionals.

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.

SQL is not a database system, but it is a query language.

What is SQL?

SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.

This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDSMS.

You can easily create and manipulate the database, access and modify the table rows and columns, etc.

This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.

If you want to get a job in the field of data science, then it is the most important query language to learn.

Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

5.2 SQL Syntax

When you want to do some operations on the data in the database, then you must have to write the query in the predefined syntax of SQL.

The syntax of the structured query language is a unique set of rules and guidelines, which is not case-sensitive. Its Syntax is defined and maintained by the ISO and ANSI standards.

You can write the keywords of SQL in both uppercase and lowercase, but writing the SQL keywords in uppercase improves the readability of the SQL query.

- SQL statements or syntax are dependent on text lines. We can place a single SQL statement on one or multiple text lines.
- You can perform most of the action in a database with SQL statements.
- SQL syntax depends on relational algebra and tuple relational calculus.

5.3 SQL Data Types

Data types are used to represent the nature of the data that can be stored in the database table. For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

- String Data types
- Numeric Data types
- Date and time Data types

SQL Operators

Every database administrator and user uses SQL queries for manipulating and accessing the data of database tables and views.

The manipulation and retrieving of the data are performed with the help of reserved words and characters, which are used to perform arithmetic operations, logical operations, comparison operations, compound operations, etc.

What is SQL Operator?

The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query. In SQL, an operator can either be a unary or binary operator. The unary operator uses only one operand for performing the unary operation, whereas the binary operator uses two operands for performing the binary operation.

5.4 SQL Create Database

In SQL, the 'Create Database' statement is a first step for storing the structured data in the database.

The database developers and the users use this statement in SQL for creating the new database in the database systems. It creates the database with the name which has been specified in the Create Database statement.

Syntax of Create Database statement in SQL

In this syntax, **Database_Name** specifies the name of the database which we want to create in the system. We have to type the database name in query just after the 'Create Database' keyword.

SQL DROP Database

The SQL Drop Database statement deletes the existing database permanently from the database system. This statement deletes all the views and tables if stored in the database, so be careful while using this query in SQL.

This statement deletes all the data from the database. If you want to restore the deleted data in the future, you should keep the backup of data of that database which you want to delete.

Another most important point is that you cannot delete that database from the system which is currently in use by another database user. If you do so, then the drop statement shows the following error on screen:

5.5.1 SQL RENAME Database

In some situations, database users and administrators want to change the name of the database for some technical reasons. So, the Rename Database statement in SQL is used to change the name of the existing database.

Sometimes, the Rename Database statement is used because the developers think that the original name is not more relevant to the data of the database, or they want to give a temporary name to that database.

5.5.2 SQL SELECT Database

Suppose database users and administrators want to perform some operations on tables, views, and indexes on the specific existing database in SQL. Firstly, they have to select the database on which they want to run the database queries.

Any database user and administrator can easily select the particular database from the current database server using the USE statement in SQL.

SQL RENAME Database

In some situations, database users and administrators want to change the name of the database for some technical reasons. So, the Rename Database statement in SQL is used to change the name of the existing database.

Sometimes, the Rename Database statement is used because the developers think that the original name is not more relevant to the data of the database, or they want to give a temporary name to that database.

SQL SELECT Database

Suppose database users and administrators want to perform some operations on tables, views, and indexes on the specific existing database in SQL. Firstly, they have to select the database on which they want to run the database queries.

Any database user and administrator can easily select the particular database from the current database server using the USE statement in SQL.

5.6 SQL SELECT UNIQUE

Actually, there is no difference between DISTINCT and UNIQUE.

5.6.1 SELECT UNIQUE is an old syntax which was used in oracle description but later ANSI standard defines DISTINCT as the official keyword.

After that oracle also added DISTINCT but did not withdraw the service of UNIQUE keyword for the sake of backward compatibility.

In simple words, we can say that SELECT UNIQUE statement is used to retrieve a unique or distinct element from the table.

5.6.2 SQL DROP TABLE

A SQL DROP TABLE statement is used to delete a table definition and all data from a table.

This is very important to know that once a table is deleted all the information available in the table is lost forever, so we have to be very careful when using this command.

5.6.3 SQL DELETE TABLE

The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

SQL RENAME TABLE

In some situations, database administrators and users want to change the name of the table in the SQL database because they want to give a more relevant name to the table.

SQL DROP TABLE

A SQL DROP TABLE statement is used to delete a table definition and all data from a table.

This is very important to know that once a table is deleted all the information available in the table is lost forever, so we have to be very careful when using this command.

SQL DELETE TABLE

The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

DELETE FROM table_name [**WHERE** condition];

But if you do not specify the WHERE condition it will remove all the rows from the table.

DELETE FROM table_name;

There are some more terms similar to DELETE statement like as DROP statement and TRUNCATE statement but they are not exactly same there are some differences between them.

SQL RENAME TABLE

In some situations, database administrators and users want to change the name of the table in the SQL database because they want to give a more relevant name to the table.

Any database user can easily change the name by using the RENAME TABLE and ALTER TABLE statement in Structured Query Language.

The RENAME TABLE and ALTER TABLE syntax help in changing the name of the table.

Car Name	Car Color	Car Cost
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000
Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000
Swift Dezire	Red	7,95,000

5.7 SQL INJECTION

The SQL Injection is a code penetration technique that might cause loss to our database. It is one of the most practiced web hacking techniques to place malicious code in SQL statements, via webpage input. SQL injection can be used to manipulate the application's web server by malicious users.

SQL injection generally occurs when we ask a user to input their username/userID. Instead of a name or ID, the user gives us an SQL statement that we will unknowingly run on our database. For Example - we create a SELECT statement by adding a variable "demoUserID" to select a string. The variable will be fetched from user input (getRequestString).

Types of SQL injection attacks

SQL injections can do more harm other than passing the login algorithms. Some of the SQL injection attacks include:

Updating, deleting, and inserting the data: An attack can modify the cookies to poison a web application's database query.

It is executing commands on the server that can download and install malicious programs such as Trojans.

We are exporting valuable data such as credit card details, email, and passwords to the attacker's remote server.

Getting user login details: It is the simplest form of SQL injection. Web application typically accepts user input through a form, and the front end passes the user input to the back end database for processing.

Example of SQL Injection

We have an application based on employee records. Any employee can view only their own records by entering a unique and private employee ID. We have a field like an Employee ID. And the employee enters the following in the input field:

How to detect SQL Injection attacks

Creating a SQL Injection attack is not difficult, but even the best and good-intentioned developers make mistakes. The detection of SQL Injection is, therefore, an essential component of creating the risk of an SQL injection attack. Web Application Firewall can detect and block basic SQL injection attacks, but we should depend on it as the sole preventive measure.

Intrusion Detection System (IDS) is both network-based and host-based. It can be tuned to detect SQL injection attacks. Network-based IDSec can monitor all connections to our database server, and flags suspicious activities. The host-based IDS can monitor web server logs and alert when something strange happens.

Impact of SQL Injection

The intruder can retrieve all the user-data present in the database, such as user details, credit card information, and social security numbers, and can also gain access to protected areas like the administrator portal. It is also possible to delete the user data from the tables. These days all the online shopping applications, bank transactions use back-end database servers. If the intruder can exploit SQL injection, the entire server is compromised.

How to prevent SQL Injection attack

- We should use user authentication to validate input from the user by pre-defining length, input type, and the input field.
- Restricting the access privileges of users and defining the amount of data any outsider can access from the database. Generally, the user cannot be granted permission to access everything in the database.
- We should not use system administrator accounts.

TYPES OF OPERATORS

SQL operators are categorized in the following categories:

1. SQL Arithmetic Operators
2. SQL Comparison Operators
3. SQL Logical Operators
4. SQL Set Operators
5. SQL Bit-wise Operators
6. SQL Unary Operators

CHAPTER – 6
JAVA SERVLETS

6.1 GENERAL

- ➢ Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).Servlet technology is robust and scalable because of java language.
- ➢ Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below. There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, Servlet Response, etc. Java Servlets is a Java based web technology.
- ➢ Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side-without a face. Java servlets make many Web applications possible.
- ➢ Java Servlets comprise a fundamental part of the Java Enterprise Edition (Java EE). Please note that Java Servlets have to be executed inside a Servlet compatible "Servlet Container" (e.g. web server) in order to work.

MERITS

Servlets are platform independent as they can run on any platform.

• The Servlet API inherits all the features of the Java platform.

• It builds and modifies the security logic for server-side extensions.

• Servlets inherit the security provided by the Web Server.

• In Servlet, only a single instance of the requests runs concurrently. It does not run in a separate process. So, it saves the memory by removing the overhead of creating a new process for each request.

6.2 SERVLET CONTAINER

It is known as servlet engine which manages Java Servlet components on top of a web server to the request send by the client.

Servlet Container provides the following services:

• It manages the servlet life cycle.

• The resources like servlets, JSP pages and HTML files are managed by servlet container.

• It appends session ID to the URL path to maintain session.

• Provides security service.

• It loads a servlet class from network services, file systems like remote file system and local file system.

6.3 SERVLET CONTAINER CONFIGURATIONS

The servlet container can be configured with the web server to manage servlets in three ways listed below:

• Standalone container

• In-process container

• Out-process container

Standalone container: In this type the Web Server functionality is taken by the Servlet container. Here, the container is strongly
coupled with the Web server.

In-Process container: In this the container runs within the Web server process.

Out-Process container: In this type there is a need to configure the servlet container to run outside the Web server process. It is
used in some cases like if there is a need to run Servlets and Servlet container in different process/systems.

6.4 SESSION

It is a collection of HTTP requests between client and server. The session is destroyed when it expires and its resources are back to the servlet engine.

Session Handling

It is a means to keep track of session data. This represents the data transferred in a session. It is used when session data from one session may be required by a web server for completing tasks in same or different sessions. Session handling is also known assession tracking.

Mechanisms of Session Handling

There are four mechanisms for session handling: URL rewriting: The session data required in the next request is appended to the URL path used by the client to make the next request.

Session

It is a collection of HTTP requests between client and server. The session is destroyed when it expires and its resources are back to the servlet engine.

Session Handling

It is a means to keep track of session data. This represents the data transferred in a session. It is used when session data from one session may be required by a web server for completing tasks in same or different sessions. Session handling is also known assession tracking.

Mechanisms of Session Handling

There are four mechanisms for session handling: URL rewriting: The session data required in the next request is appended to the URL path used by the client to make the next request.

Query String: A string appended after the requested URI is query string. The string is appended with separator as '?' character.

Example 1): http://localhost:8080/newproject/login?user=test&amp;passwd=abcde

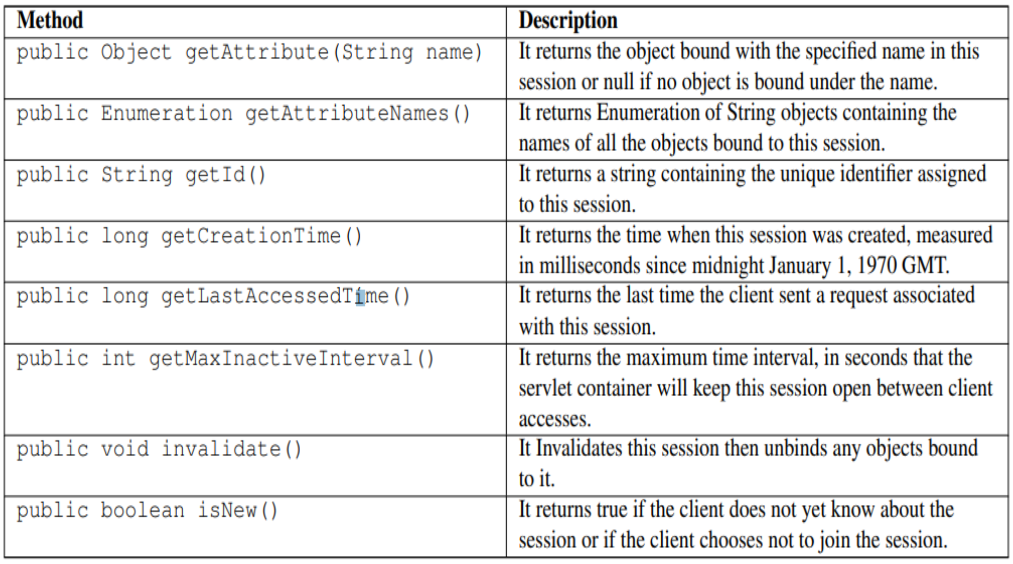
Path Info: It is the part of the request URI. Session data can be added to the path info part of the request URI.

Example 2): http://localhost:8080/newproject/myweb/login;user=test&amp;passwd=abcde

Hidden form field: A type of HTML form field which remains hidden in the view. Some other form fields are: textbox, password **etc.**

This approach can be used with form-based requests. It is just used for hiding user data from other different types of users.

HTTP session: It provides asession management service implemented through HttpSession object. Some HttpSession object methods are listed here;



Method	Description
`public Object getAttribute(String name)`	It returns the object bound with the specified name in this session or null if no object is bound under the name.
`public Enumeration getAttributeNames()`	It returns Enumeration of String objects containing the names of all the objects bound to this session.
`public String getId()`	It returns a string containing the unique identifier assigned to this session.
`public long getCreationTime()`	It returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
`public long getLastAccessedTime()`	It returns the last time the client sent a request associated with this session.
`public int getMaxInactiveInterval()`	It returns the maximum time interval, in seconds that the servlet container will keep this session open between client accesses.
`public void invalidate()`	It Invalidates this session then unbinds any objects bound to it.
`public boolean isNew()`	It returns true if the client does not yet know about the session or if the client chooses not to join the session.

6.5 EXCEPTION HANDLING

Exceptions are used to handle errors. It is a reaction to unbearable conditions. Here comes the role of web.xml i.e. deployment description which is used to run JSP and servlet pages. The container searches the configurations in web.xml for a match. So, in web.xml use these exception-type elements for match with the thrown exception type when a servlet throws an exception.

ERROR CODE CONFIGURATION

The /HandlerClass servlet gets called when an error with status code 403 occurs as shown below: Listing 7: For Error code 403.

6.6 DEBUGGING

Client-server interactions are in large number in Servlets. This makes errors difficult to locate. Different ways can be followed for location warnings and errors.

Message Logging

Logs are provided for getting information about warning and error messages. For this a standard logging method is used.

Servlet API can generate this information using log() method. Using Apache Tomcat, these logs can be found in TomcatDirectory/logs.Java Debugger Servlets can be debugged using JDB Debugger i.e.

Java Debugger

In this the program being debugged is sun.servlet.http.HttpServer.

Set debugger’s class path for finding the following classes:

• servlet.http.HttpServer

• server_root/servlets and server_root/classes: Through this the debugger sets breakpoints in a servlet.

Headers

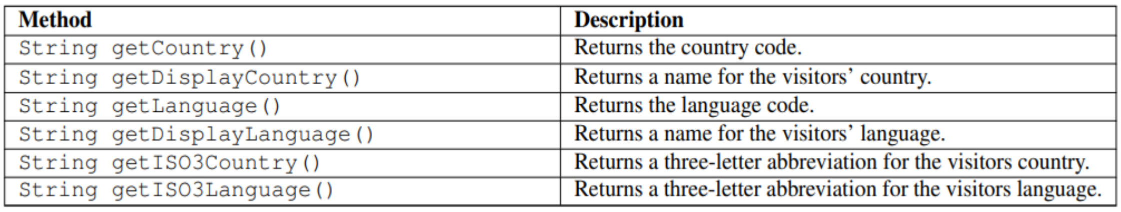
Users should have some information related to structure of HTTP headers. Issues can be judged using them which can further locate some unknown errors. Information related to HTTP headers can help you in locating errors. Studying request and response can help in guessing what is not going well.

Refresh

Refresh your browser’s web page to avoid it from caching previous request. At some stages, browser shows request performed previously.

6.7 INTERNATIONALIZATION

For building a global website, some important points are considered which includes language related to user’s nationality. Internationalization is enabling a website for providing content translated in different languages according to user’s nationality.



Method	Description
`String getCountry()`	Returns the country code.
`String getDisplayCountry()`	Returns a name for the visitors’ country.
`String getLanguage()`	Returns the language code.
`String getDisplayLanguage()`	Returns a name for the visitors’ language.
`String getISO3Country()`	Returns a three-letter abbreviation for the visitors country.
`String getISO3Language()`	Returns a three-letter abbreviation for the visitors language.

CHAPTER – 7

JAVA SCRIPT

7.1 GENERAL

- JavaScript is a language

- JavaScript is a light weight, Just-in-Time [JIT] compiled programming language.

- JavaScript is also an interpreted language.

Official Definition:

- It is a light weight interpreted and JIT compiled programming language.

- Interpreted language translates line by line of program.

- Compiled language translates all lines simultaneously at the same time.

- JavaScript supports both

a) Interpreted

b) Compiled

- Compiled languages are 2 types

a) JIT Compiled

b) AOT Compiled

- JIT is "Just-in-Time". It compiles the program in browser.

- AOT is "Ahead-Of-Time". It compiles the program in application and sends to Browser.

What is the role of JavaScript in HTML?

- JavaScript is used for DOM interactions and Manipulations.

- Adding elements into DOM

- Removing elements from DOM

Updating data into elements

- JavaScript is used to reduce burden on server by handling validations client side.
- Why it is called as Script?
- - A script can run individually or it can be embedded into another interface.
- How JavaScript is integrated into HTML

How JavaScript is integrated into HTML?

1. Inline

2. Embedded

3. External File

7.2.1. Inline JavaScript:

- Script is embedded into element.

- It is faster in access.

- It can't be re-used.
- Syntax:
- <button onclick="window.print()"> Print </button>

7.2.2. Embedded

- - JavaScript is defined in the page by using <script> container.
- - Script can be embedded into head or body section.
- Script MIME type is "text/javascript"
- Syntax:
- <head>
- <script type="text/javascript">
- </script>

- You can't access across pages.

7.2.3. Script in External File

- - JavaScript files have extention ".js"
- - JavaScript files are linked to any page by using "<script>" tag.
- home.js
- <script src="home.js" type="text/javascript"> </script>
- - You can access the script from any page.
- - Using external file will increase the number of requests for page and also the
- page load time.

7.3. JavaScript can refer HTML elements by using DOM hierarchy

Syntax:

window.document.images[0].src=

window.documents.forms[0].elements[1].value=

- It is faster in rendering as it is native to browser.
- Refering complete hierarchy is always complex.
- Refering by index is always an issue when element position changes.

7.3.1. JavaScript can refer elements by using "name".

— Every element in page can have a reference name.

- JavaScript can access elements by their name.

Syntax:

<img name="pic">

pic.src="path";

- If any element is a child element, you can't access directly without refering to its parent.

Syntax:

formName.elementName.property = "value";

"Name" attribute can be same for several elements.

yntax:

formName.elementName.property = "value";

"Name" attribute can be same for several elements.

7.3.2. JavaScript can refer HTML elements by using ID.

- JavaScript document object provides a method called "getElementByld()"
- It can access any element by using ID directly.

You can access any child element without refering to its parent.

Syntax:

<input type="button" id="btnRegister">

document.getElementByld("btnRegister").value = “VALUE”

- ID is used as selector in CSS, multiple elements can have same id.

7.4 Variables

Variables are storage locations in memory where you can store a value and use
it as a part of any expression.

Variable configuration comprises of 3 phases

a) Declaration

b) Assignment

c) Initialization

var x; Declaration

x=10; Assignment

var y=20; Initialization

Data Types

- Data Type defines data structure.

Data structure contains information about what type of data your can store and what range of data can be stored.

- JavaScript is not strongly typed language.
- JavaScript variables can't restrict data type. You can store any type of value.

Var x= 10;

x ="John";

// x is a number type

// x is changed to string type

- JavaScript data types are classified into 2 groups

a) Primitive Types

b) Non Primitive Types

a) Primitive types

Primitive types are immutable types.

You can't change the structure.

They have a fixed range for values.

Stack uses LIFO [Last-ln-First-Out]

varx= 10;

x = 20;

X = 30;

JavaScript Primitive types are

1. Number
2. String
3. Boolean
4. Null
5. Undefined

Number

It is used for numeric values JavaScript number can be

signed integer -20

unsigned integer- 20

floating point - 34.42

Double - 230.23

decimal - 2300.330

exponent - 2e3 [2 x = 2000]

binary - Ob1010

hexa - Of019

octa – 00746

avaScript allows to convert a string into number ifit is starting with numeric.

parselnt()

parseFloat()

Access data from API

- **API** [Application Interface] allows 2 applications running on 2 different machines to share information.
- Data is transported using 3 specifications

a) SOAP [Service Oriented Architecture Protocol]

b) REST [Representational State Transfer]

c) JSON [JavaScript Object Notation)

- SOAP

client => XML request => server sends => XML response

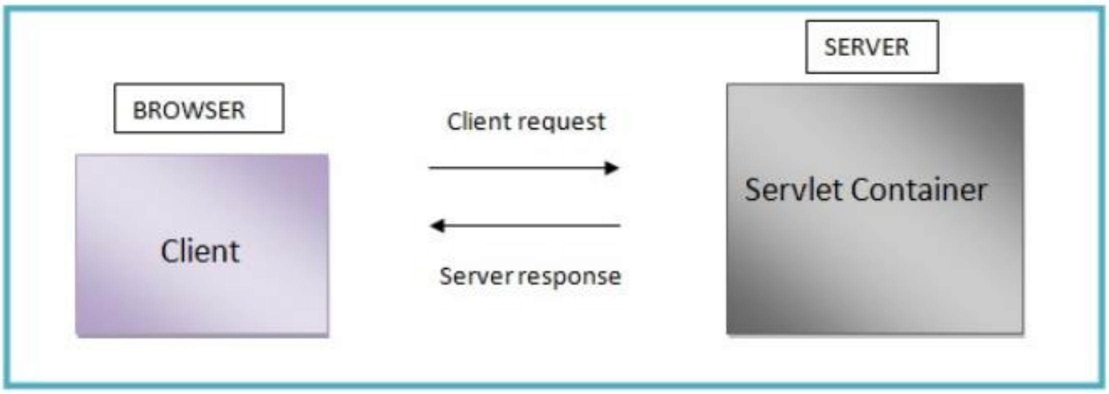
- REST

client => query request => server sends => XML response

- JSON

client => JSON request => sever sends => JSON response

7.5 SERVLET PROCESS



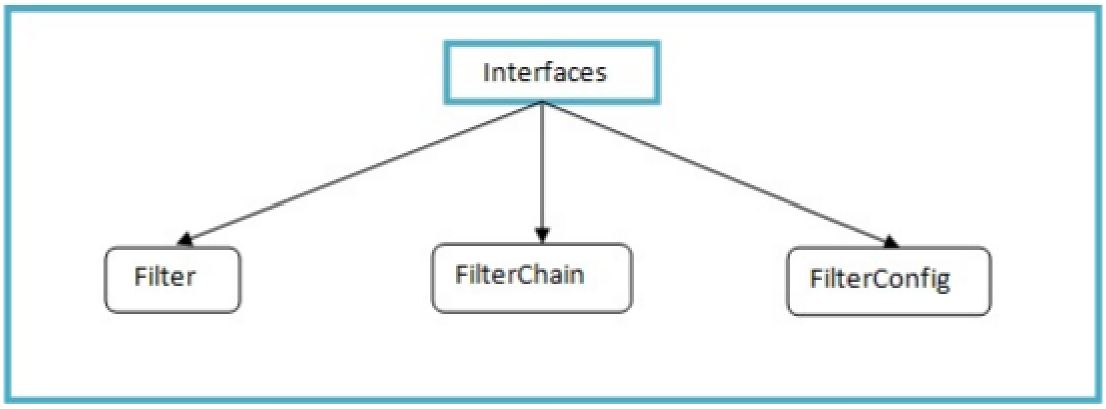
- A Request is sent by a client to a servlet container. The container acts as a Web server.
- The Web server searches for the servlet and initiates it.
  The client request is processed by the servlet and it sends the response back to the server.
- The Server response is then forwarded to the client.

FILTER

- Filters transform the content of requests, responses, and header information from one format to another. These are reusable codes.
- Filter class is declared in the deployment descriptor.
- It is used to write reusable components.
- The request is process before it is called using filters.
- It can be used under a web application for some tasks like:
- Validation
- Compression
- Verification
- Internationalization

INTERFACE

- This is the initial and basic interface which all filter class should implement.



CHAPTER – 8

IMPLEMENTATION

1.<u>Index.html:</u>

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./assests/CSS/style.css">
    <link rel="stylesheet" href="./assests/bootstrap/bootstrap.min.css">
    <script src="./assests/JS/jquery.slim.min.js"></script>
    <script src="./assests/JS/popper.min.js"></script>
    <script src="./assests/JS/bootstrap.bundle.min.js"></script>
    <script src="./assests/JS/jscript.js"></script>
    <script src="https://kit.fontawesome.com/2340707775.js"
crossorigin="anonymous"></script>

    <title>Dice Game</title>

</head>

<body class="bg-info">

    <nav class="navbar navbar-dark bg-dark justify-content-center">
        <h3 class="text-light">Dice Game</h3>
    </nav>

    <div class="container playerBoard">
        <div class="row justify-content-around">
            <div class="col-5">
                <div class="contents">
                    <label for="Player1">
                        <h2 id="player1">Player 1</h2></label>
                    <br>
                    <img src="./assests/images/dice-6.png" alt="dice" class="diceCss"
id="dice1">
                    <br>
                    <button type="button" id="rollPlayer1" class="btn-primary btn"
disabled onclick="rollDice1()">
                        Roll Dice
                    </button>
                </div>

            </div>
            <div class="col-5">
                <div class="contents">
                    <label for="Player2">
                        <h2 id="player2">Player 2</h2></label>
                    <br>
                    <img src="./assests/images/dice-6.png" alt="dice" class="diceCss"
id="dice2">
                    <br>
                    <button type="button" id="rollPlayer2" class="btn-primary btn"
disabled onclick="rollDice2()">
                        Roll Dice
                    </button>
                </div>
            </div>
        </div>
        <br>
        <div class="row justify-content-center">
            <div class="col-4">
                <h3 class="text-center">Score</h3>
                <div class="row justify-content-center">
                    <div class="col-6 border border-dark">
                        Player 1
                    </div>
                    <div class="col-6 border border-dark">
                        Player 2
                    </div>
                </div>
                <div class="row justify-content-center">
                    <div class="col-6 border border-dark">
                        <label for="p1Score" id="p1Score"> 0 </label>
                    </div>
                    <div class="col-6 border border-dark">
                        <label for="p2Score" id="p2Score"> 0 </label>
                    </div>
                </div>
            </div>
        </div>
        <br>
        <div class="row justify-content-center">
            <div class="col-6 text-center">
                <button type="button" id="start" class="btn-success btn" data-
toggle="modal" data-target="#playerName">
                    Start
                </button>
                <button type="button" id="restart" class="btn-success btn" disabled
onclick="window.location.reload();">
                    Restart
                </button>
                <div class="modal fade" id="playerName" data-backdrop="static" data-
keyboard="false" tabindex="-1"
                    aria-labelledby="adnm" aria-hidden="true">
                    <div class="modal-dialog modal-dialog-centered">
                        <div class="modal-content">
                            <div class="modal-header">
                                <h5 class="modal-title">Enter Player Name</h5>
                                <button type="button" class="btn" data-
dismiss="modal">
                                    <i class="fas fa-window-close"></i>
                                </button>
                            </div>
                            <div class="modal-body grid">
                                <label for="Player1">Player 1</label>
                                <input type="text" id="player1Name"
placeholder="Player 1 Name">
                                <label for="Player2">Player 2</label>
                                <input type="text" id="player2Name"
placeholder="Player 2 Name">
                            </div>
                            <div class="modal-footer">
                                Click Here to Start the Game -->
                                <button type="button" class="btn btn-primary"
id="nameSubmit" onclick="addName()" data-dismiss="modal">
                                    Submit
                                </button>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <br>
    </div>
</body>
</html>
```

2.**Style.css**

```css
.playerBoard {
    margin-top: 5%;
    height: 100%;
    align-self: center;
    background-color: #ffffff;
    border-radius: 8px;
    align-content: center;
}

.diceCss {
    width: 30%;
    margin-left: 35%;
}

.contents {
    display: grid;
    text-align: center;
    margin-top: 5%;
}

.btn {
    margin: 5px;
    padding: 5px;
}

.grid {
    display: grid;
    grid-template-columns: 25% 70%;
    grid-template-rows: 50% 50%;
    grid-gap: 10px;
}
```

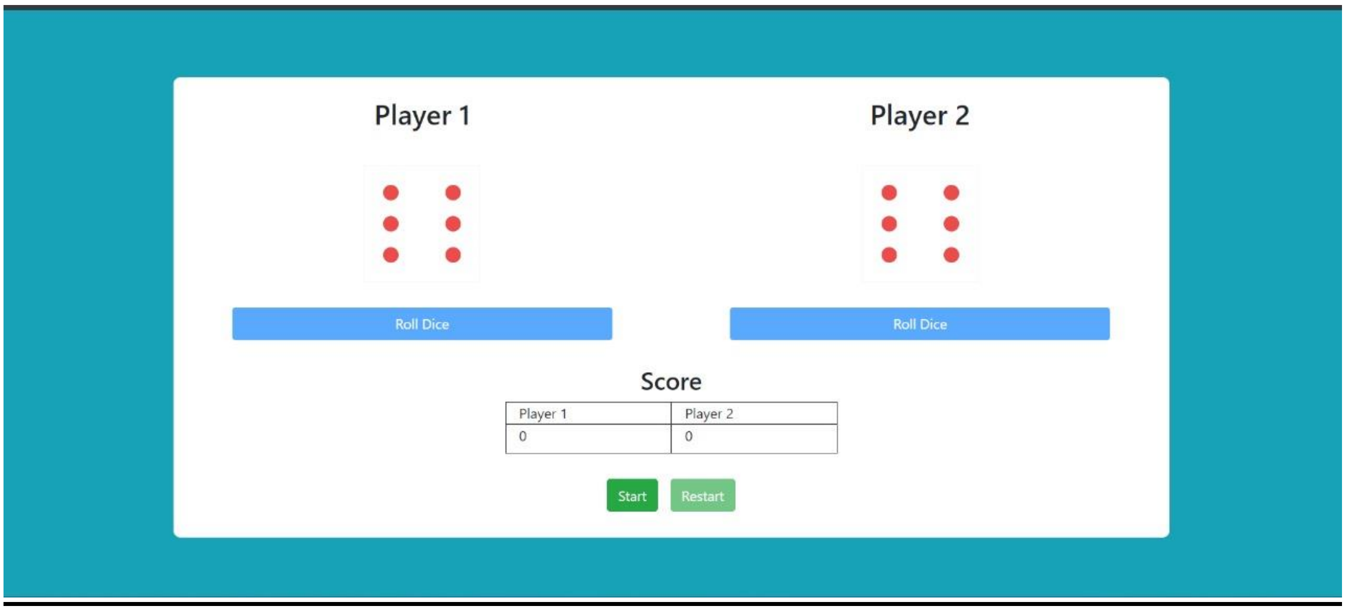
3. JSCRIPT.JS

```
var chance = 1,
    score1 = 0,
    score2 = 0,
    win = 0;


function addName() {
    var p1 = document.getElementById("player1Name");
    var p2 = document.getElementById("player2Name");

    player1.innerHTML = p1.value;
    player2.innerHTML = p2.value;

    document.getElementById("rollPlayer1").disabled = false;
    document.getElementById("start").disabled = true;
    document.getElementById("restart").disabled = false;


}


function rollDice1() {
    var rand1 = 0;
    for (let i = 0; i < 20; i++) {
        rand1 = Math.floor(Math.random() * 6) + 1;
        switch (rand1) {
            case 1:
                document.getElementById("dice1").src = "./assests/images//dice-1.png";
                break;
            case 2:
                document.getElementById("dice1").src = "./assests/images/dice-2.png";
                break;
            case 3:
                document.getElementById("dice1").src = "./assests/images/dice-3.png";
                break;
            case 4:
                document.getElementById("dice1").src = "./assests/images/dice-4.png";
                break;
            case 5:
                document.getElementById("dice1").src = "./assests/images/dice-5.png";
                break;
            case 6:
                document.getElementById("dice1").src = "./assests/images/dice-6.png";
                break;
            default:
                document.getElementById("dice1").src = "./assests/images/dice-2.png";
                break;
        }
    }
    score1 = score1 + rand1;
    if (score1 > 50 || rand1 == 6) {
        document.getElementById("rollPlayer2").disabled = false;
        document.getElementById("rollPlayer1").disabled = true;
        score1 -= rand1;
        return;
    } else if (score1 == 50) {
        p1Score.innerHTML = score1;
        win = 1;
        setTimeout(winAnnounce(),500);
    } else {
        p1Score.innerHTML = score1;
    }
}
function rollDice2() {
    var rand = 0;
    for (let j = 0; j < 20; j++) {
        rand = Math.floor(Math.random() * 6) + 1;
        switch (rand) {
            case 1:
                document.getElementById("dice2").src = "./assests/images/dice-1.png";
                break;
            case 2:
                document.getElementById("dice2").src = "./assests/images/dice-2.png";
                break;
            case 3:
                document.getElementById("dice2").src = "./assests/images/dice-3.png";
                break;
            case 4:
                document.getElementById("dice2").src = "./assests/images/dice-4.png";
                break;
            case 5:
                document.getElementById("dice2").src = "./assests/images/dice-5.png";
                break;
            case 6:
                document.getElementById("dice2").src = "./assests/images/dice-6.png";
                break;
            default:
                document.getElementById("dice2").src = "./assests/images/dice-2.png";
                break;
        }
    }
    score2 = score2 + rand;
    if (score2 > 50 || rand == 6) {
        document.getElementById("rollPlayer1").disabled = false;
        document.getElementById("rollPlayer2").disabled = true;
        score2 -= rand;
        return;
    } else if (score2 == 50) {
        p2Score.innerHTML = score2;
        win = 2;
        setTimeout(winAnnounce(),500);
    } else {
        p2Score.innerHTML = score2;
    }

}

function winAnnounce() {
    if (win == 1) {
        var p1 = document.getElementById("player1Name");
        window.alert(p1.value + " Wins");
        setTimeout(window.location.reload(), 5000);
    }

    if (win == 2) {
        var p2 = document.getElementById("player2Name");
        window.alert(p2.value + " Wins");
        setTimeout(window.location.reload(), 5000);
    }
}
```

CHAPTER - 9

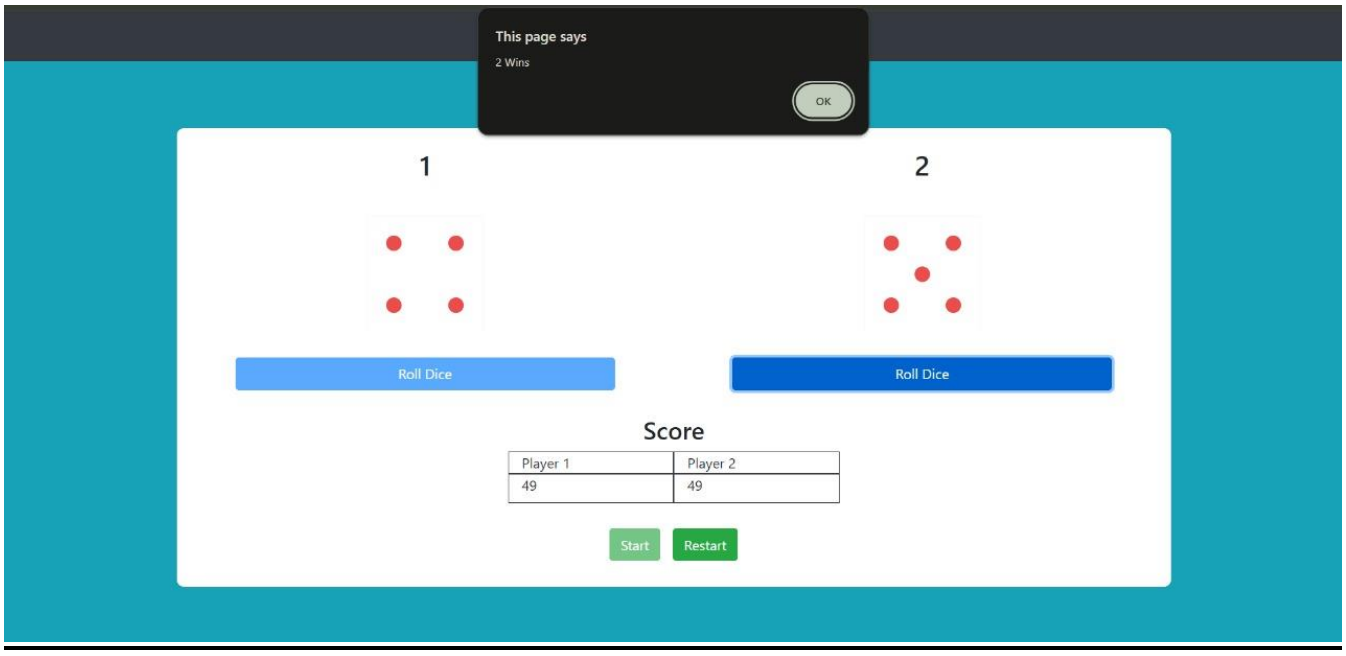
SNAPSHOTS

Screenshot-1: Home page of the Game



Player 1	Player 2
0	0

Screenshot-2 :- The winner of the game after the game play



This page says

2 Wins

Player 1	Player 2
49	49

CHAPTER – 10
CONCLUSION

In conclusion, this internship documentation reflects a holistic learning experience encompassing various technologies and tools crucial in modern web development. The exploration began with fundamental technologies such as HTML and CSS, providing a solid foundation for structuring and styling web content. The integration of JavaScript introduced dynamic behavior, enriching the user experience and adding interactivity to the web applications.

The journey extended to AngularJS, a powerful JavaScript framework, enabling the creation of robust and scalable front-end applications. The incorporation of AngularJS facilitated the development of dynamic, single-page applications (SPAs) with a modular and maintainable code structure.

On the server-side, SQL played a pivotal role in managing and querying databases. Understanding database concepts and implementing SQL queries became imperative for handling data efficiently in the back-end of web applications.

Throughout the internship, the focus was not only on individual technologies but also on the collaborative integration of these technologies to build cohesive and functional web solutions. The hands-on experience gained in each technology, along with the exploration of their synergies, has contributed to a well-rounded skill set in web development.

As the internship concludes, the knowledge and skills acquired in HTML, CSS, JavaScript, AngularJS, and SQL pave the way for continued growth in the dynamic and ever-evolving field of web development. The ability to create seamless, responsive, and interactive web applications is a testament to the comprehensive learning journey undertaken during this internship.

This documentation serves as a valuable resource, capturing the milestones achieved, challenges overcome, and the valuable insights gained throughout the internship. It stands as a testament to the commitment to continuous learning and the ability to adapt to emerging technologies in the field of web development. The skills acquired during this internship will undoubtedly serve as a solid foundation for future endeavors in the exciting realm of web development.