VISVESVARAYA TECHNOLOGICAL UNIVERSITY JNANASANGAMA, BELAGAVI - 590018



An Internship Project Report on

## LIFE STYLE STORE

*Submitted in partial fulfilment for the award of the degree*

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

*Submitted By*

**DEEKSHITH M RAI**

**4SN21CS032**

Internship Carried Out at

INTERNSHALA

HARYANA

Internal Guide External Guide

Prof. Alwyn Mr. KRISHNA V

Assistant Professor Design Engineer

SIT, Mangaluru Internshala, Haryana

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SRINIVAS INSTITUTE OF TECHNOLOGY

**(Accredited by NAAC)**

**MANGALURU - 574143, KARNATAKA 2023-2024**

**SRINIVAS INSTITUTE OF TECHNOLOGY (Accredited by NAAC) MANGALURU - 574143, KARNATAKA DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

# CERTIFICATE



Certified that the Internship work entitled “**LIFE STYLE STORE**” is a bonafide work carried out by **DEEKSHITH M RAI**, bearing USN **4SN21CS032**, in partial fulfilment for the award of degree of Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University, Belagavi during the year 2023-2024. It is certified that all corrections/ suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Internship report has been approved as it satisfies the academic requirements prescribed for the degree of Bachelor of Engineering.

|  |  |
| --- | --- |
| Prof. Jayashree J | Dr. Suresha D Dr. Shrinivasa Mayya D |
| Internship Co-ordinator | Head of the Department Principal  EXTERNAL VIVA |
| Name of Examiners | Signature with Date |
| 1. | 1. |
| 2. | 2. |

# CERTIFICATE





**ACKNOWLEDGEMENT**

I would like to take this opportunity to thank a lot of eminent personalities, without whose constant encouragement and support, this endeavor of mine would not have been possible.

First and foremost, I would like to thank VTU University, Belagavi for having internship as part of our curriculum, which gave us a wonderful opportunity to work on recent technologies of IT industries. I thank Srinivas Institute of Technology for providing us facilities to reach IT companies, without which, this internship could not have acquired the orientation, it has been now.

I take this opportunity to express my profound gratitude and deep regards to our Internship Guide, **Prof. Alwyn ,** Assistant Professor, Department of Computer Science and Engineering**,** for his support and guidance. I thank him for her guidance and constant encouragement throughout the course of this project.

I also take this opportunity to express my profound gratitude and deep regards to our

Internship Coordinator, **Prof. Jayashree J,** Assistant Professor, Department of Computer Science and Engineering**,** for her support and guidance.

I express a deep sense of gratitude to **Dr. Suresha D ,** Head of the Department, Computer Science and Engineering, for his cordial support, valuable information and guidance, which helped me in completing this project through various stages.

I also express my heartfelt gratitude to our Principal **Dr. Shrinivasa Mayya D.** for his kind co- operation and encouragement which helped me in the completion of this project.

**DEEKSHITH M RAI 4SN21CS032**

# DECLARATION

I **DEEKSHITH M RAI**, bearing USN **4SN21CS032**, student of Sixth Semester B.E in Computer Science and Engineering, Srinivas Institute of Technology, Mangaluru, hereby declare that the Internship work entitled "**LIFE STYLE STORE**" has been carried out and duly executed by me at Internshala, Haryana, in online mode under the guidance of **Prof. Jayashree J**, Assistant Professor, Department of Computer Science and Engineering, Srinivas Institute of Technology, Mangaluru and **Mr. KRISHNA V**, Design Engineer, Internshala, Haryana, submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Engineering in Computer Science and Engineering by Visvesvaraya Technological University, Belagavi during the academic year 2023- 2024.

Date: 13-11-2023 DEEKSHITH M RAI

Place: Mangaluru 4SN21CS032

**EXECUTIVE SUMMARY**

This internship report explains and annotates the work experience I have gathered as an Intern in the company “Internshala Haryana” from October 15, 2023 to December 15, 2023.

**LIFE STYLE STORE** is a website where user can easily buy the Clothing, Camera and other Equipment’s . Users have to create their account, if they want to purchase any items. I implemented a comprehensive LIFE STYLE STORE project on Internshala platform, utilizing HTML, CSS, and BOOTSTRAP for frontend design, while employing DBMS, PHP, JAVASCRIPT, REACT for backend data management.

The internship also helped me to learn about the various development strategies of an organization and a peek into every minute details to be taken care of was highlighted. In addition to technical aspects, I

# TABLE OF CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter | No. | Title | Page No. |
| 1 |  | **INTRODUCTION** | **1-2** |
|  | 1.1 | Objective | 1 |
|  | 1.2 | Overview | 1-2 |
|  | 1.3 | Benefits Of Internship | 2 |
| 2 |  | **HTML** | **3-6** |
|  | 2.1 | HTML Tags | 3-4 |
|  | 2.2 | HTML Code | 4 |
|  | 2.3 | Creating Lists Using HTML | 4-5 |
|  | 2.4 | Creating Forms Using HTML | 5-6 |
|  | 2.5 | Conclusion | 6 |
| 3 |  | **CSS** | **7-12** |
|  | 3.1 | Types Of Styling | 7-10 |
|  | 3.2 | Types Of CSS Selector | 10-11 |
|  | 3.3 | CSS Property | 11 |
|  | 3.4 | Creating Responsive Website | 12 |
|  | 3.5 | Conclusion | 12 |
| 4 |  | **BOOTSTRAP** | **13-15** |
|  | 4.1 | Break Points | 13 |
|  | 4.2 | Bootstrap Code | 14-15 |
|  | 4.3 | Predefined Classes In Bootstrap | 15 |
|  | 4.4 | Conclusion | 15 |
| 5 |  | **DBMS** | **16-20** |
|  | 5.1 | Structured Query Language | 16-17 |
|  | 5.2 | Creating Database | 18 |
|  | 5.3 | SQL With Table | 18-19 |
|  | 5.4 | Creating Table | 19 |
|  | 5.5 | Database Relationships | 19-20 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.6 | Conclusion | 20 |
| 6 |  | **PHP** | **21-27** |
|  | 6.1 | Variables and Operators In PHP | 21-22 |
|  | 6.2 | Conditional Statements In PHP | 22-24 |
|  | 6.3 | Loops In PHP | 24-25 |
|  | 6.4 | Database Related Function | 25-26 |
|  | 6.5 | Cookies And Sessions | 26 |
|  | 6.6 | Conclusion | 27 |
| 7 |  | **JAVA SCRIPT** | **28-32** |
|  | 7.1 | Variables In JS | 28 |
|  | 7.2 | Operators In JS | 28-29 |
|  | 7.3 | Conditional Statements In JS | 29 |
|  | 7.4 | Loops In JS | 30 |
|  | 7.5 | Function In JS | 30-31 |
|  | 7.6 | Arrow Function In JS | 31 |
|  | 7.7 | Array Methods | 31-32 |
|  | 7.8 | Conclusion | 32 |
| 8 |  | **REACT** | **33-40** |
|  | 8.1 | Prerequisites To Use React | 33-34 |
|  | 8.2 | Creating HTML Elements | 34-35 |
|  | 8.3 | Introduction To JSX | 35-37 |
|  | 8.4 | Introduction To Props And States | 37-38 |
|  | 8.5 | Handling The Event | 38-39 |
|  | 8.6 | Deploy | 39 |
|  | 8.7 | Conclusion | 40 |
| 9 |  | **PROJECT SCREENSHOT** | **41-44** |
|  |  | **CONCLUSION** | **45** |
|  |  | **BIBLIOGRAPHY** | **46** |

**Chapter 1**

# INTRODUCTION

The journey of an engineering student is not only confined within the walls of a classroom but extends far beyond, encompassing diverse experiences that shape and mold them into well rounded professionals. The internship period serves as a critical juncture where theoretical knowledge is fused with practical application, providing a platform for students to bridge the gap between academia and industry. This internship report sheds light on my transformative experience during my engineering internship, delving into the various chapters that encapsulate my growth, learning, and contributions.

### Objectives

Internships are generally thought to be reserved for college students looking to gain experience in a particular field. However, a wide array of people can benefit from Training Internships in order to receive real-world experience and develop their skills. Internships hold immense significance in the journey of engineering students, playing a pivotal role in their overall education and professional development. The rise in global competition has prompted organizations to devise strategies to have a talented and innovative workforce to gain a competitive edge.

### Overview

During my fourth-semester internship at Internshala, I completed a one-month web development course covering HTML, CSS, Bootstrap, DBMS, PHP, JavaScript, React, and concluded with a project on LifeStyleStore, a website to find rented rooms in different cities.

* + - **HTML**: A markup language for creating the structure of web pages.
    - **CSS**: A styling language used to enhance the presentation of HTML documents.
    - **Bootstrap**: A front-end framework for building responsive and mobile-first websites.
    - **DBMS**: Database Management System, software for managing databases and handling data.
    - **PHP**: A server-side scripting language primarily used for web development.
    - **JavaScript**: A versatile programming language for adding interactivity and dynamic content to web pages.
    - **React**: A JavaScript library for building user interfaces, particularly for single-page applications.

### Benefits of Internship

A web development course offers a multitude of benefits. Not only does it equip you with marketable skills sought after by employers across various industries, but it also opens up a wide range of career opportunities. With the flexibility to work remotely and the continuous learning opportunities in a dynamic field, web development provides a creative outlet for expressing design and functionality. It hones problem-solving abilities, fosters collaboration, and empowers entrepreneurial ventures. Ultimately, by contributing to the digital world, web development allows you to make information and services more accessible to people worldwide.

Moreover, as you delve deeper into web development, you'll discover the satisfaction of transforming ideas into tangible digital experiences. Whether you're building sleek user interfaces or optimizing website performance, every project offers an opportunity for growth and innovation. Beyond personal and professional development, web development also plays a crucial role in shaping the digital landscape. By creating intuitive interfaces and seamless interactions, you enhance user experiences and facilitate access to information worldwide.

**Chapter 2**

# HTML

Hypertext Mark-up Language (HTML), serves as the backbone of the internet, providing the essential structure for web pages. Composed of tags enclosed in angle brackets ("< >"), HTML elements define various parts of a webpage, such as headings, paragraphs, images, and links. This markup language follows a hierarchical structure, allowing developers to nest elements and organize content effectively. Alongside CSS and JavaScript, HTML forms the core technologies used in web development, enabling the creation of visually appealing and interactive websites. Understanding HTML is crucial for anyone venturing into web development, as it lays the foundation for building responsive and user-friendly digital experiences.

Proficiency in HTML is indispensable for web developers and designers alike. With its straightforward syntax and universal compatibility, HTML facilitates the creation of accessible and well-structured content on the web. Developers utilize HTML tags to organize information logically, optimize pages for search engines, and ensure compatibility across different devices and browsers. As the internet continues to evolve, HTML remains a fundamental skill, empowering individuals to navigate the digital landscape and contribute to the ever-expanding world wide web.

### HTML Tags

Here's a brief overview of some common HTML tags:

* + - **<!DOCTYPE>:** Specifies the HTML version and document type.
    - **<html>:** Defines the root element of an HTML page.
    - **<head>:** Contains meta-information about the document, such as title, styles, and scripts.
    - **<title>:** Sets the title of the webpage displayed in the browser's title bar.
    - **<body>:** Contains the content of the webpage, including text, images, links, etc.
    - **<h1> to <h6>:** Defines headings of different levels, from most important (h1) to least (h6).
    - **<p>:** Indicates a paragraph of text.
    - **<a>:** Creates a hyperlink to another webpage or resource.
    - **<img**>: Embeds an image into the webpage.
    - **<div>:** Defines a division or section in the document, often used for styling purposes.
    - **<span>:** Inline container used for styling or scripting small portions of text.
    - **<ul>:** Defines an unordered list.
    - **<ol>:** Defines an ordered list.
    - **<li>:** Represents a list item in <ul> or <ol>.
    - **<table>:** Creates a table for organizing data.
    - **<tr>:** Defines a row in a table.
    - **<td>:** Represents a cell in a table.
    - **<form>:** Sets up a form for user input.
    - **<input>:** Defines an input control within a form.
    - **<button>:** Creates a clickable button.

### HTML Code

<!DOCTYPE html>

<html lang="en">

<head>

<title>Simple HTML Example</title>

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is a simple HTML <a href="https://[www.example.com](http://www.example.com/)">anchor tag</a> and a paragraph.</p>

</body>

</html>

### Creating Lists Using HTML

* + - **Ordered List (<ol>)**: HTML element used to create a list of items in a specific numerical order, presenting each item with a default numbering sequence.
    - **Unordered List (<ul>)**: HTML element utilized for creating a list of items without any particular sequence, displaying each item with a default bullet point.

### Creating Forms Using HTML

HTML forms facilitate user interaction on web pages by enabling input fields, buttons, and other elements for data submission and processing.

<form method="post" action="submit.php">

<input type="text" name="email"/>

<input type="password" name="password"/><br/>

<input type="radio" name="gender" value="male"/>Male

<input type="radio" name="gender" value="female"/>Female

<input type="radio" name="gender" value="other"/>Other<br/>

<input type="checkbox" name="interest" value="football"/>Football<br/>

<input type="checkbox" name="interest" value="chess"/>Chess<br/>

<select name="city">

<option value="Kolkata">Kolkata</option>

<option value="Delhi">Delhi</option>

<option value="Mumbai">Mumbai</option>

<option value="Banglore">Banglore</option>

<option value="Outstation">Outstation</option>

</select><br/>

<textarea name="address" rows="6" cols="50"> Write your adddress....

</textarea><br/>

<input type="submit" name="submit" value="Submit"/>

</form>

### Conclusion

HTML, as the foundational language of the web, empowers developers to structure and present content effectively, fostering user engagement and interaction through its versatile elements and attributes. Its simplicity and ubiquity make it an indispensable tool for creating accessible, standardized, and visually appealing web experiences, shaping the digital landscape and driving innovation in online communication and information dissemination.

**Chapter 3**

# CSS

CSS, or Cascading Style Sheets, is a critical language in web development that dictates the visual presentation of HTML and XML documents. It provides developers with the means to define the layout, appearance, and behaviour of elements within a webpage. At its core, CSS operates on selectors, which target specific HTML elements, classes, IDs, attributes, or relationships between elements. Through properties and values, CSS assigns visual attributes such as color, size, spacing, and positioning to these targeted elements. The concept of cascade and specificity governs how conflicting styles are resolved, ensuring the appropriate application of styles.

Moreover, CSS adopts a box model to represent HTML elements as rectangular boxes with content, padding, border, and margin areas, offering precise control over dimensions and spacing. Responsive design is facilitated by CSS through media queries, allowing styles to adapt based on factors like screen size and device orientation.

### Types of Styling in CSS

1. **Inline Styling/Style Attribute**

Inline styling in CSS allows developers to apply styles directly to individual HTML elements using the attribute within the HTML tag. This method offers a quick and straightforward way to define styles for specific elements without the need for external CSS files or additional markup. The syntax for inline styling follows the standard CSS property-value pair format enclosed within double quotes. While inline styles provide immediate results and have the highest specificity, overriding styles defined in external or internal stylesheets, they come with limitations.

style

Therefore, while inline styling is suitable for quick adjustments or prototyping, it's generally advisable to reserve its usage for specific scenarios.

**Example:**

<div style="color: blue; font-size: 16px;">This text is styled inline</div>

### Internal style sheet

An internal style sheet in HTML involves defining CSS styles directly within the element, typically placed in the section of an HTML document. Unlike inline styles applied directly to individual HTML elements, internal style sheets allow developers to consolidate and organize CSS rules within the HTML file itself.

**Example**

<!DOCTYPE html>

<html>

<head>

<style>

body { background-color: lightblue; } h1 { color: navy; }

</style>

</head>

<body>

<h1>Hello, CSS!</h1>

<p>This is an example of internal styling.</p>

</body>

</html>

### External styling:

External styling in HTML involves linking an external CSS file to an HTML document using the element within the section. This method provides a clean separation between content and presentation, enhancing maintainability and scalability in web development

projects. The element specifies the relationship between the HTML document and the external CSS file, allowing styles defined in the CSS file to be applied to elements throughout the HTML document.

**Example: HTML file:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial- scale=1.0">

<title>External Style Sheet Example</title>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<h1>Welcome to External Styling</h1>

<p>This is an example of external styling.</p>

</body>

</html>

**CSS file:**

/\* External Styles \*/ body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

}

h1 {

color: #333;

text-align: center;

}

p {

color: #666;

text-align: center;

}

### Types of CSS Selectors:

* + - **Element Selector**: Targets HTML elements based on their tag name.
    - **Class Selector**: Targets elements with a specific class attribute.
    - **ID Selector**: Targets a single element with a specific ID attribute.
    - **Universal Selector**: Targets all elements on the page.
    - **Attribute Selector**: Targets elements based on their attributes.
    - **Descendant Selector**: Targets elements that are descendants of another element.
    - **Child Selector**: Targets elements that are direct children of another element.
    - **Adjacent Sibling Selector**: Targets an element that is immediately preceded by another element.

### CSS property

A CSS property is a key-value pair that defines the visual appearance or behavior of an HTML element. CSS properties control various aspects of an element's styling, such as its color, size, spacing, position, and more. Here are some common CSS properties:

* + - color: Sets the text color.
    - font-size: Sets the size of the font.
    - font-family: Defines the typeface or font family used for text.
    - background-color: Sets the background color of an element.
    - border: Defines the border around an element, specifying its width, style, and color.
    - margin: Sets the margin space around an element.
    - padding: Sets the padding space inside an element.
    - width: Sets the width of an element.
    - height: Sets the height of an element.
    - display: Specifies how an element should be displayed (e.g., block, inline, inline- block).
    - position: Defines the positioning method of an element (e.g., static, relative, absolute, fixed).
    - float: Specifies whether an element should float to the left or right of its container.
    - text-align: Sets the horizontal alignment of text within an element.
    - text-decoration: Adds decorations such as underline, overline, or line-through to text.
    - box-shadow: Adds a shadow effect to an element.
    - border-radius: Sets the radius of the element's corners to create rounded corners.
    - transition: Specifies the transition effect when a property changes value over time.
    - z-index: Sets the stack order of positioned elements.

### Creating a responsive website

* + - **Plan Your Layout**: Start by sketching or wireframing your website layout, considering how elements will be arranged and prioritized across different screen sizes. Determine the overall structure of your website, including navigation menus, content sections, and call-to-action buttons.
    - **Use Fluid Grids and Flexible Layouts:** Instead of fixed-width layouts, use fluid grids and flexible layouts that adjust proportionally to the screen size.

CSS frameworks like Bootstrap and Foundation offer grid systems that make it easier to create responsive layouts.

* + - **Responsive Typography**: Ensure that text on your website is legible and easy to read across all devices. Use relative units like percentages, ems, or rems for font sizes to allow text to scale appropriately based on the screen size.
    - **Media Queries**: Use CSS media queries to apply different styles based on the device's screen width, height, orientation, or resolution.
    - **Flexible Images and Media**: Set max-width: 100% for images and media elements to ensure they scale down proportionally on smaller screens without overflowing or becoming distorted.
    - **Optimize Navigation**: Design navigation menus that are easy to use on both desktop and mobile devices.
    - **Touch-friendly Design**: Make sure interactive elements such as buttons, links, and form fields are large enough and spaced appropriately for touch input on mobile devices. Avoid using hover effects that rely on mouse interactions, as they may not work well on touchscreens.
    - **Performance Optimization:** Optimize your website's performance by minimizing HTTP requests, compressing images, and using CSS and JavaScript minification techniques.

### Conclusion

In conclusion, CSS serves as a cornerstone in modern web development, allowing developers to tailor the visual presentation and layout of HTML documents with precision and flexibility. Through its wide array of selectors, properties, and styling techniques, CSS empowers designers to create aesthetically pleasing and user-friendly websites across various devices and screen sizes. Whether it's through inline styling, internal style sheets, or external CSS files.

**Chapter 4**

# BOOTSTRAP

Bootstrap is a popular open-source framework for building responsive and mobile-first websites and web applications. It offers a grid system, pre-styled components, and JavaScript functionalities for easy development. With its cross-browser compatibility and Sass support, Bootstrap enables developers to create consistent and customizable designs. Its active community provides support and resources for easy adoption.

Moreover, Bootstrap is regularly updated to incorporate the latest web development trends and best practices, ensuring that projects built with Bootstrap remain modern and maintainable over time.

### Break Points

Bootstrap employs breakpoints to facilitate responsive web design, ensuring that websites and applications adapt gracefully to different screen sizes and devices. Breakpoints are predefined sizes at which the layout of a page is adjusted to accommodate varying viewport widths. Breakpoints are predefined sizes at which the layout of a page is adjusted to accommodate varying viewport widths. Bootstrap includes several default breakpoints:

* + - **Extra Small (xs):** Up to 576 pixels wide (e.g., smartphones in portrait orientation).
    - **Small (sm):** From 576 pixels up to 768 pixels wide (e.g., smartphones in landscape orientation and small tablets).
    - **Medium (md):** From 768 pixels up to 992 pixels wide (e.g., larger tablets and small laptops).
    - **Large (lg):** From 992 pixels up to 1200 pixels wide (e.g., desktops and larger laptops).
    - **Extra Large (xl):** 1200 pixels wide and above (e.g., extra-large desktop screens).

### Bootstrap Code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Bootstrap Breakpoints Example</title>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-sm-6 col-md-4">

<div class="bg-primary p-3 text-white">

Content for Small (sm) devices and Medium (md) devices

</div>

</div>

<div class="col-sm-6 col-md-4">

<div class="bg-secondary p-3 text-white">

Content for Small (sm) devices and Medium (md) devices

</div>

</div>

<div class="col-sm-6 col-md-4">

<div class="bg-info p-3 text-white">

Content for Small (sm) devices and Medium (md) devices

</div>

</div>

</div>

</div>

</body>

</html>

### Predefined Classes In Bootstrap

* + - **align-items**: The align-items class in Bootstrap is used to vertically align grid items within their container along the cross-axis. It offers options like align-items-start, align- items-center, align-items-end, align-items-baseline, and align-items-stretch. For example, align-items-center will vertically center the grid items within their container, while align-items-end will align them to the bottom.
    - **align-self**: The align-self class in Bootstrap allows individual control over the vertical alignment of specific grid items within their container. It complements align-items and accepts values such as align-self-start, align-self-center, align-self-end, align-self- baseline, and align-self-stretch.
    - **justify-content**: Bootstrap's justify-content class aligns grid items horizontally within their container along the main axis. Options include justify-content-start, justify- content-center, justify-content-end, justify-content-between, justify-content-around, and justify-content-evenly.
    - **Offset**: Bootstrap's offset classes create space between grid columns by horizontally shifting them. They are structured as offset-{size}-{number}, where {size} represent the breakpoint (xs, sm, md, lg, xl), and {number} denotes the number of columns to offset.

### Conclusion

In conclusion, Bootstrap stands as a versatile and widely adopted front-end framework for web development. Its rich collection of predefined CSS and JavaScript components, coupled with a responsive grid system, empowers developers to create professional-looking websites and web applications rapidly. Bootstrap's extensive documentation, active community support, and regular updates ensure its relevance and usability in modern web development practices. With Bootstrap, developers can streamline their workflow, maintain consistency across projects, and deliver engaging user experiences across various devices and screen sizes.

**Chapter 5**

# DBMS

A Database Management System (DBMS) serves as a fundamental component in modern information systems, facilitating the storage, retrieval, and management of data in a structured and organized manner. It acts as an intermediary between users and the database, enabling efficient access to data while ensuring data integrity, security, and concurrency control. DBMS software handles tasks such as data modeling, schema definition, query processing, optimization, and transaction management, providing users with a unified interface to interact with the underlying database.

One of the key advantages of a DBMS is its ability to support multiple users and applications concurrently, allowing for collaborative data access and manipulation without compromising data consistency or reliability. By enforcing ACID (Atomicity, Consistency, Isolation, Durability) properties, DBMS ensures that transactions are processed reliably and efficiently, even in the presence of concurrent access and system failures. Additionally, DBMS often incorporates features such as data backup and recovery, replication, and access control mechanisms to safeguard data integrity and protect against unauthorized access or data loss.

### Structured Query Language (SQL)

SQL (Structured Query Language) queries form the backbone of interacting with relational database management systems (RDBMS) in database management systems (DBMS). They allow users to perform various operations such as retrieving, manipulating, and managing data stored in a relational database. Here's an overview of SQL queries in DBMS:

#### Data Retrieval:

* + - SELECT statement: Used to retrieve data from one or more tables based on specified criteria.
    - FROM clause: Specifies the table(s) from which data is to be retrieved.
    - WHERE clause: Filters data based on specified conditions.
    - JOIN operations: Combines data from multiple tables based on related columns.

#### Data Manipulation:

* + - INSERT statement: Adds new records into a table.
    - UPDATE statement: Modifies existing records in a table based on specified conditions.
    - DELETE statement: Removes records from a table based on specified conditions.

#### Data Definition:

* + - CREATE statement: Creates new database objects such as tables, views, indexes, or stored procedures.
    - ALTER statement: Modifies the structure of existing database objects.
    - DROP statement: Deletes database objects such as tables, views, or indexes.

#### Data Control:

* + - GRANT statement: Assigns privileges to users or roles, allowing them to perform specific actions on database objects.
    - REVOKE statement: Removes previously granted privileges from users or roles.

#### Transaction Control:

* + - BEGIN TRANSACTION statement: Starts a new transaction.
    - COMMIT statement: Commits a transaction, making its changes permanent.
    - ROLLBACK statement: Rolls back a transaction, undoing its changes.

#### Data Querying and Aggregation:

* + - Aggregate functions (e.g., SUM, AVG, MAX, MIN, COUNT): Performs calculations on sets of values.
    - GROUP BY clause: Groups rows based on specified columns.
    - HAVING clause: Filters groups based on specified conditions.
    - ORDER BY clause: Sorts the result set based on specified columns.

### Creating database

Creating a database in SQL involves using the CREATE DATABASE statement followed by the name of the database you wish to create. For example:

CREATE DATABASE my databases;

This statement creates a new database named "my database" in the SQL server. Once created, you can then proceed to create tables, define relationships, and manage data within this database using SQL statements. It's worth noting that depending on the SQL implementation (e.g., MySQL, PostgreSQL, SQL Server), additional options such as specifying character sets, collations, and storage configurations may be available when creating databases.

### SQL queries with table

#### Data Types:

Data types define the type of data that can be stored in a column of a table. Common data types include:

* + - Numeric types (e.g., INTEGER, DECIMAL, FLOAT)
    - Character types (e.g., CHAR, VARCHAR, TEXT)
    - Date and time types (e.g., DATE, TIME, TIMESTAMP)
    - Binary types (e.g., BINARY, VARBINARY, BLOB).

#### Constraints:

Constraints are rules enforced on data within a table to maintain data integrity. Common constraints include:

* + - Primary Key: Ensures each row in a table is uniquely identified.
    - Foreign Key: Enforces referential integrity between two tables by requiring values in one table to match values in another table's primary key.
    - Unique: Ensures that values in a column (or a combination of columns) are unique across the table.
    - NOT NULL: Specifies that a column must have a non-null value.
    - Check: Enforces specific conditions on values allowed in a column.

#### Default Value:

Default values provide a predetermined value for a column if no value is explicitly specified during insertion.

* + - When defining a column, you can specify a default value using the DEFAULT constraint.
    - Default values can be constants, expressions, or special values such as NULL or CURRENT\_TIMESTAMP.

### Creating Table

Creating a table in a Database Management System (DBMS) involves defining the structure of the table, including its columns and the data types they will contain. This process is fundamental for organizing and storing data efficiently within a database. In SQL, the standard language for managing relational databases, the CREATE TABLE statement is used to create tables.

CREATE TABLE table\_name ( column1 datatype [constraints], column2 datatype [constraints],

...

[table\_constraints]

);

### Database Relationsips:

Database relationships in DBMS are fundamental for organizing and managing data effectively. These relationships define how data in one table is associated with data in another table, ensuring data integrity and consistency within the database.

In a **one-to-one relationship**, each record in the parent table corresponds to exactly one record in the child table, and vice versa. While this relationship is less common, it's useful for separating distinct data entities when necessary.

A **one-to-many relationship** is more prevalent, where each record in the parent table can be linked to multiple records in the child table, but each record in the child table is associated with only one record in the parent table. This model is often used to represent hierarchical structures or scenarios where one entity has multiple related entities.

**Many-to-many relationships** occur when each record in one table can be related to multiple records in another table, and vice versa. In such cases, a junction table is used to establish the connection between the two tables, containing foreign key columns that reference the primary keys of the related tables.

For instance, in a school database, a one-to-many relationship may exist between Students and Courses tables, where each student can enroll in multiple courses, but each course is associated with only one student. Additionally, a many-to-many relationship can exist between Students and Courses tables, as each student can enroll in multiple courses, and each course can have multiple students enrolled.

These relationships ensure data consistency, prevent redundancy, and enable efficient data retrieval and manipulation, making them essential in designing robust and scalable database schemas.

### Conclusion

In conclusion, Database Management Systems (DBMS) serve as foundational components in modern information systems, facilitating the storage, organization, and retrieval of vast amounts of data efficiently. DBMS enables users to interact with databases using Structured Query Language (SQL) or other query languages, allowing for seamless data manipulation, querying, and management.

The importance of DBMS lies in its ability to ensure data integrity, security, and concurrency control, thereby maintaining consistency and reliability in data operations. By enforcing constraints, establishing relationships between data entities, and providing mechanisms for transaction management, DBMS ensures that data remains accurate, accessible, and protected from unauthorized access or corruption.

**Chapter 4**

# PHP

PHP, which stands for Hypertext Pre-processor, is a powerful server-side scripting language widely used for web development. It is primarily used to create dynamic web pages and interact with databases, making it a fundamental component of many modern websites and web applications. PHP code is embedded within HTML documents, allowing developers to seamlessly mix dynamic content generation with static HTML mark-up. One of PHP's key strengths is its versatility and ease of use, with a syntax that resembles other popular programming languages such as C, Java, and JavaScript.

PHP can handle a wide range of tasks, including form processing, file manipulation, session management, and authentication, making it well-suited for building feature-rich and interactive web applications. Additionally, PHP boasts a large and active community of developers, along with extensive documentation and a rich ecosystem of frameworks, libraries, and tools that further accelerate the development process. With its robust capabilities and widespread adoption, PHP remains a vital technology in the ever-evolving landscape of web development

### Variables and operators in PHP

In PHP, variables can store values of various data types, including:

* + - Integer: Whole numbers without any decimal points.
    - Float: Numbers with decimal points or numbers in scientific notation.
    - String: Sequences of characters.
    - Boolean: Represents true or false values.
    - Array: Holds multiple values in an indexed or associative manner.
    - Object: Instances of classes containing properties and methods.
    - NULL: Represents a variable with no value or explicitly set to NULL.

#### Types of Operators in PHP

1. **Arithmetic operator:** Arithmetic operators in PHP are used for mathematical calculations.

**Example**: $result = 10 + 5; // Adds 10 and 5, assigns result to $result variable.

1. **Assignment operator:** Assignment operators in PHP are used to assign values to variables.

**Example: $**a = 10; // Assigns the value 10 to the variable $a.

1. **Increment/Decrement operator:** The increment/decrement operators in PHP are used to increase or decrease the value of a variable by one.

**Example:** $a++; // Increments the value of $a by 1.

1. **Comparison operator:** Comparison operators in PHP are used to compare two values and return a Boolean result.

**Example:** $is equal = ($a == $b); // Checks if $a is equal to $b

1. **Logical (Relational) operator:** Logical operators in PHP are used to combine conditional statements and return a Boolean result.

**Example:** $is true = ($a && $b); // Checks if both $a and $b are true.

### Conditional statements in PHP

1. **If statement:** The "if" statement in PHP is used to execute a block of code if a specified condition is true.

**Example:** if ($age >= 18) echo "You are an adult."; // Checks if $age is greater than or equal to 18.

1. **If-else statement**: The "if-else" statement in PHP allows for the execution of different blocks of code based on whether a specified condition is true or false.

**Example:** if ($age >= 18) {

echo "You are an adult.";

} else {

echo "You are not an adult.";}

1. **if else if:** The "if-else if" statement in PHP allows for the execution of different blocks of code based on multiple conditions.

**Example:** if ($age < 18) {

echo "You are a minor.";

} elseif ($age >= 18 && $age < 65) { echo "You are an adult.";

} else {

echo "You are a senior citizen.";

}

1. **Switch:** The "switch" statement in PHP allows for the selection of one of many blocks of code to be executed based on the value of an expression.

### Example:

switch ($day) { case 1:

echo "Monday"; break;

case 2:

echo "Tuesday"; break;

case 3:

echo "Wednesday"; break;

default:

echo "Other day";

}

### Loops in PHP

1. **while loop:** The "while" statement in PHP is used to execute a block of code repeatedly as long as a specified condition is true.

### Example:

$i = 0;

while ($i < 5) { echo $i;

$i++;

}

1. **do-while:** The "do-while" statement in PHP is used to execute a block of code once, and then repeat the execution as long as a specified condition is true. Unlike the "while" loop, which evaluates the condition before the execution of the block, the "do- while" loop executes the block first and then checks the condition.

### Example:

$i = 0; do {

echo $i;

$i++;

} while ($i < 5);

1. **for :** The "for" loop in PHP is a control structure used to execute a block of code repeatedly for a specified number of times.

### Example:

for ($i = 0; $i < 5; $i++) { echo $i;

}

1. **for each:** The "foreach" loop in PHP is used to iterate over arrays or objects and execute a block of code for each element. It is particularly useful when you need to iterate over the elements of an array or the properties of an object, allowing for concise and readable code.

**Example:**

$colors = array("red", "green", "blue"); foreach ($colors as $color) {

echo $color;

}

### Database related Functions:

1. The **mysqli\_connect()**function in PHP is used to establish a connection to a MySQL database server.

#### Syntax:

$connection = mysqli\_connect("localhost", "username", "password", "database");

1. The **mysqli\_connect\_error()** function in PHP is used to return a string description of the last connect error.

#### Syntax:

if (!$connection) {

die("Connection failed: " . mysqli\_connect\_error());

}

1. The **mysqli\_query()** function in PHP is used to perform a query against the MySQL database.

#### Syntax:

$result = mysqli\_query($connection, "SELECT \* FROM table\_name");

1. The **mysqli\_fetch\_assoc()** function in PHP is used to fetch a result row as an associative array from a result set returned by a query.

Syntax: $row = mysqli\_fetch\_assoc($result);

1. The **mysqli\_error()** function in PHP is used to return a string description of the last error occurred during the execution of a MySQL query.

#### Synatx:

if (!$result) {

die("Query failed: " . mysqli\_error($connection));

}

### Cookies and Sessions:

#### Cookies:

* + - Cookies are small pieces of data stored on the client's browser. They are sent by the server in the HTTP response headers and are subsequently included in subsequent requests to the same server.
    - Cookies are primarily used to store user-specific information, such as login credentials, language preferences, or user preferences.
    - They have an expiration date and can be persistent (stored on the client's browser even after the browser is closed) or session-based (deleted when the browser is closed).

#### Sessions:

* + - Sessions are a server-side mechanism for storing user-specific information. They are typically implemented using a session ID, which is a unique identifier assigned to each user session.
    - Unlike cookies, sessions store data on the server, typically in a temporary directory or database. The session ID is sent to the client as a cookie, allowing the server to associate subsequent requests with the correct session data.
    - Sessions are more secure than cookies because the data is stored on the server and cannot be tampered with by the client.

### Conclusion

In conclusion, PHP is a powerful server-side scripting language widely used for web development. With its rich set of features and extensive community support, PHP enables developers to build dynamic and interactive web applications quickly and efficiently.

From handling form submissions and interacting with databases to processing user authentication and generating dynamic content, PHP offers a wide range of functionalities essential for modern web development.

**Chapter 7**

# JAVA SCRIPT

JavaScript (JS) is a dynamic programming language primarily employed for enhancing web pages with interactive features. Its syntax is akin to other programming languages, simplifying the learning curve for those with programming backgrounds. JavaScript operates within the ECMAScript standard, with updates such as ES6 introducing new functionalities like arrow functions and class declarations. Typically used for client- side scripting, JavaScript enables developers to manipulate webpage elements, respond to user actions, and update content dynamically without necessitating page reloads.

### Variables In JS

In JavaScript, variables store data values. They are declared using var, let, or const. Variables declared with var have function-level scope, while those declared with let or const have block-level scope. Variables declared with const cannot be reassigned.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature** |  | **var** |  |  | **let** |  |  | **const** |  |
| Scope | Function | | | Block | | | Block | | |
| Hoisting | Yes (Declaration) | | | Yes (Declaration) | | | Yes (Declaration) | | |
| Reassignment | Yes | | | Yes | | | No (Immutable) | | |
| Redeclaration | Yes | | | No | | | No | | |

JavaScript is dynamically typed, allowing variables to hold values of any data.

### Operators In JS

**Arithmetic Operators** include symbols for addition (+), subtraction (-), multiplication (\*), division (/), remainder (modulus, %), as well as increment (++) and decrement (--).

**Assignment Operators** are used to assign values to variables, with symbols like **=**, **+=**, **-=**,

**\*=**, **/=**, and **%=**. These operators can combine assignment with arithmetic operations.

**Comparison Operators** are used to compare two values or expressions, returning a boolean value (true or false). They include equal to (==), not equal to (!=), strict equal to (===), strict

not equal to (!==), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=).

**Logical Operators** operate on boolean values and are used to combine multiple conditions. They include logical AND (&&), logical OR (||), and logical NOT (!).

**Bitwise Operators** perform bitwise operations on operands, manipulating individual bits within binary representations. Examples include bitwise AND (&), bitwise OR (|), bitwise XOR (^), bitwise NOT (~), left shift (<<), right shift (>>), and zero-fill right shift (>>>).

**Unary Operators** operate on a single operand. Examples include unary plus (+), unary minus (-), increment (++), decrement (--), logical NOT (!), typeof, and void.

The **Ternary (Conditional) Operator** is a shorthand for an if-else statement, with the syntax **condition ? expr1 : expr2**. It evaluates the condition and returns one of two expressions based on whether the condition is true or false.

The **String Operator** (+) is used for concatenating strings.

The **Comma Operator** (,) evaluates both of its operands and returns the value of the last operand. It is often used to include multiple expressions in a single statement.

### Conditional Statements In JS

Conditional statements in JavaScript are essential for controlling the flow of a program based on specified conditions. They allow developers to execute different blocks of code depending on whether a condition evaluates to true or false. The primary conditional statements in JavaScript include **if**, **else if**, **else**, and the **switch** statement.

The **if statement** is used to execute a block of code if a specified condition is true. If the condition evaluates to false, the code block is skipped. Optionally, an **else if statement** can be added to test additional conditions if the initial **if** condition is false. This allows for more complex decision-making within the code. Additionally, an **else statement** can be appended after an **if** or **else if** block to specify a block of code to be executed if none of the preceding conditions are true.

The **switch statement** provides an alternative way to handle multiple conditional branches based on the value of an expression. It evaluates an expression, matches the expression's

value to a case label, and executes the corresponding block of code. The **switch** statement can include multiple case labels and an optional default label, which is executed if no match is found.

### Loops In JS

**for...of loop**: The **for...of** loop is used to iterate over the values of an iterable object, such as an array or a string. For example:

const numbers = [1, 2, 3, 4, 5]; for (const num of numbers) { console.log(num);

}

**for...in loop**: The **for...in** loop is used to iterate over the enumerable properties of an object, including inherited ones. For Example:

const person = { name: 'John', age: 30,

city: 'New York'

};

for (const key in person) { console.log(`${key}: ${person[key]}`);

}

**forEach loop**: The **forEach** loop is a method available on arrays that allows you to iterate over each element of the array and perform an action. For example:

const colors = ['red', 'green', 'blue']; colors.forEach(color => {

console.log(color);

});

### Functions In JS

In JavaScript, functions are blocks of reusable code designed to perform a specific task. They can be declared using the **function** keyword followed by a name and a set of parentheses containing optional parameters. Functions can accept input parameters, execute a series of statements, and return a value using the **return** statement. They provide a way to modularize code, promote reusability, and improve maintainability in JavaScript applications. Additionally, functions in JavaScript are first-class citizens, meaning they can be assigned to variables, passed as arguments to other functions, and returned from other functions. Here's a simple example of a function declaration in JavaScript:

function greet(name) {

return "Hello, " + name + "!";

}

### Arrow Function In JS

Arrow functions in JavaScript, introduced in ECMAScript 6 (ES6), offer a concise syntax for defining functions. They are particularly handy for short, inline functions and preserve the lexical context of the **this** keyword. With a shorter syntax and lexical **this**, arrow functions simplify function declaration and usage in JavaScript code.

// Traditional function expression const multiply = function(x, y) {

return x \* y;

};

// Arrow function

const multiplyArrow = (x, y) => x \* y;

### Array Methods

In JavaScript, arrays come with a rich set of built-in methods for performing various operations efficiently. These methods can be used to manipulate arrays, iterate over elements, filter data, and more. Some commonly used array methods in JavaScript include:

* + - **push()**: Adds one or more elements to the end of an array and returns the new length of the array.
    - **pop()**: Removes the last element from an array and returns that element.
    - **shift()**: Removes the first element from an array and returns that element, shifting all subsequent elements to a lower index.
    - **unshift()**: Adds one or more elements to the beginning of an array and returns the new length of the array.
    - **concat()**: Combines two or more arrays and returns a new array without modifying the existing arrays.
    - **slice()**: Returns a shallow copy of a portion of an array into a new array object selected from begin to end (end not included).
    - **splice()**: Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.
    - **forEach()**: Executes a provided function once for each array element.
    - **map()**: Creates a new array populated with the results of calling a provided function on every element in the calling array.
    - **filter()**: Creates a new array with all elements that pass the test implemented by the provided function.
    - **find()**: Returns the first element in the array that satisfies the provided testing function.
    - **findIndex()**: Returns the index of the first element in the array that satisfies the provided testing function.
    - **indexOf()**: Returns the first index at which a given element can be found in the array, or -1 if it is not present.
    - **lastIndexOf()**: Returns the last index at which a given element can be found in the array, or -1 if it is not present.
    - **includes()**: Determines whether an array includes a certain value among its elements, returning true or false as appropriate.

### Conclusion

In conclusion, JavaScript (JS) stands as a versatile and indispensable language for web development, enabling developers to create dynamic and interactive websites. With its wide array of features, including functions, variables, loops, conditionals, and built-in methods for arrays and objects, JavaScript empowers developers to build robust applications capable of handling diverse user interactions and data manipulation tasks.

**Chapter 8**

# REACT

React is a powerful JavaScript library for building user interfaces, renowned for its simplicity, flexibility, and performance. Developed by Facebook, React enables developers to create dynamic and interactive web applications with ease. At its core, React utilizes a component-based architecture, allowing developers to break down complex UIs into reusable and modular components.

These components encapsulate both the structure and behavior of UI elements, promoting code reusability and maintainability. Additionally, React employs a virtual DOM (Document Object Model) to efficiently manage updates and render UI changes, resulting in improved performance and responsiveness. With its declarative approach to building UIs and robust ecosystem of tools and libraries, React has become a go-to choice for developers seeking to create modern web applications.

### Pre requisites to use React

* + - **HTML/CSS**: Familiarity with HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) is essential as React primarily deals with building user interfaces for web applications.
    - **JavaScript** : React is built using JavaScript, so a good grasp of JavaScript fundamentals is necessary. Additionally, knowledge of ES6 (ECMAScript 2015) and its features such as arrow functions, classes, destructuring, spread/rest operators, and modules is highly recommended.
    - **Git and Version Control**: Proficiency in using Git for version control is beneficial for collaborating on React projects, tracking changes, and managing codebase versions. Understanding basic Git commands for initializing repositories, branching, committing changes, and merging branches is essential for efficient project management.
    - **Node.js and npm/yarn**: React projects typically rely on Node.js for running JavaScript on the server-side and managing project dependencies using npm

(Node Package Manager) or yarn. Familiarity with npm/yarn commands for installing packages, managing dependencies, and running scripts is essential for setting up and managing React projects.

### Creating HTML element:

1. **Creating HTML element using render():**

In React, you typically don't create HTML elements directly using the render() method. Instead, you create React components that represent these elements and then render them using the render() method. However, if you want to render a basic HTML element directly using the render() method, you can do so by using JSX, which is a syntax extension for JavaScript that allows you to write HTML-like code within your JavaScript files.

**Example:**

import React from 'react';

import ReactDOM from 'react-dom';

class MyComponent extends React.Component { render() {

return (

<div>

<h1>Hello, world!</h1>

</div>

);

}

}

ReactDOM.render(<MyComponent />, document.getElementById('root'));

### Creating multiple HTML element using React:

In React, you create multiple HTML elements by composing them within your components using JSX. JSX allows you to write HTML-like syntax directly within your JavaScript code, making it easy to create and render multiple elements.

**Example:**

import React from 'react';

import ReactDOM from 'react-dom';

class MyComponent extends React.Component { render() {

return (

<div>

<h1>Hello, world!</h1>

<p>This is a paragraph.</p>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

</div>

);

}

}

ReactDOM.render(<MyComponent />, document.getElementById('root'));

### Introduction to JSX

JSX, which stands for JavaScript XML, is a syntax extension for JavaScript that allows you to write HTML-like code directly within your JavaScript files. It was introduced by React to provide a more intuitive and concise way to define the structure of UI components.

With JSX, you can seamlessly mix HTML-like tags and JavaScript expressions, making it easier to create and manipulate UI elements in React applications. Here's a brief introduction to JSX:

* + - **HTML-Like Syntax**: JSX resembles HTML syntax, allowing you to write tags such as <div>, <p>, <h1>, etc., directly in your JavaScript code.
    - **JavaScript Expressions**: You can embed JavaScript expressions within curly braces {} inside JSX tags. This allows you to dynamically generate content, compute values, or execute functions.
    - **Component Composition**: JSX allows you to compose components by nesting them within each other, similar to how you nest HTML elements.
    - **Attribute and Prop Handling**: JSX supports specifying attributes on elements using HTML-like syntax. You can also pass props to components using JSX attributes.
    - **No Browser Dependency**: JSX is not a requirement for using React, but it's a convenient and widely adopted syntax for defining React components. JSX code is transpiled into regular JavaScript code before it's executed in the browser.

**Example:**

import React from 'react'; const MyComponent = () => { const name = 'John Doe'; return (

<div>

<h1>Hello, {name}!</h1>

<p>Welcome to my React application.</p>

</div>

);

};

export default MyComponent;

### Introduction to Props and States Props:

In React, "props" (short for "properties") serve as a crucial mechanism for passing data from parent components to child components. This mechanism allows developers to customize and configure child components dynamically, enhancing their versatility and reusability across the application. Props are passed as attributes from parent to child components, accommodating various data types such as strings, numbers, arrays, objects, and functions. Immutable by nature, props ensure the stability and predictability of a component's behavior, as their values cannot be altered once received.

**Example:**

import React from 'react';

// Child component that receives props const Greeting = (props) => {

return <h1>Hello, {props.name}!</h1>;

};

// Parent component that renders the child component const App = () => {

// Define a name to be passed as a prop const userName = "John";

// Render the Greeting component and pass the 'name' prop return <Greeting name={userName} />;

};

export default App;

### States:

In React, "state" refers to the internal data and condition of a component, representing information that can change over time due to user interactions, system events, or data updates. Unlike props, which are passed from parent to child components and remain immutable within the child component, state is managed internally by a component and can be modified using method. States play a crucial role in building dynamic and interactive user interfaces, allowing components to manage their own data and update their appearance or behavior accordingly.

**Example:**

import React, { Component } from 'react'; class Counter extends Component { constructor(props) {

super(props);

// Initialize state with a counter value of 0 this.state = {

counter: 0

};

}

// Function to increment the counter incrementCounter = () => {

// Update the state by incrementing the counter value this.setState({ counter: this.state.counter + 1 });

};

render() { return (

<div>

<h2>Counter: {this.state.counter}</h2>

<button onClick={this.incrementCounter}>Increment</button>

</div>

);

}

}

export default Counter;

### Handling the Event

In React, event handling involves the process of responding to user interactions within components. This is achieved by defining event handler functions, which are methods within the component class responsible for executing specific actions in response to events like clicks, input changes, form submissions, and more. These event handler methods are then attached to corresponding event listeners within the JSX code using special attributes like onClick, onChange, onSubmit, etc. When the specified event occurs, React automatically calls the associated event handler function.

Event handling in React follows a pattern similar to traditional JavaScript event handling, but with JSX syntax. Event handler functions often interact with the component's state, updating it using the setState() method to trigger re-renders and reflect changes in the user interface. Accessing event data within the event handler functions allows components to respond dynamically to user interactions, enabling interactive and responsive user interfaces.

### Deploy

Deploying a React application involves making it accessible to users on the web, typically by hosting it on a web server or a cloud platform. The deployment process for React applications can vary depending on factors such as the hosting provider, the complexity of the application, and the desired deployment environment.

One common approach to deploying a React application is to create a production build of the application using tools like create-react-app or webpack. This process optimizes the application for performance and bundles all the necessary files into static assets (HTML, CSS, JavaScript) that can be served by a web server.

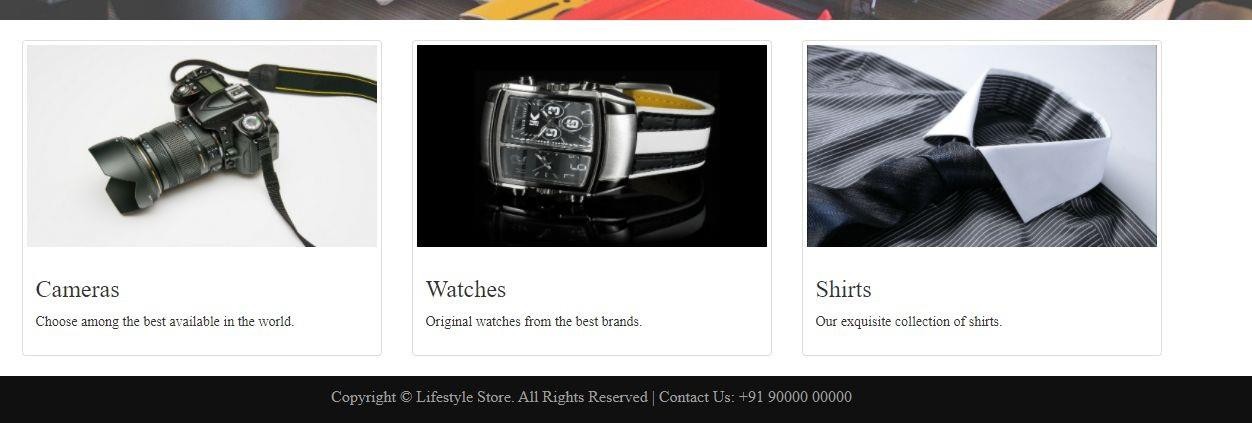
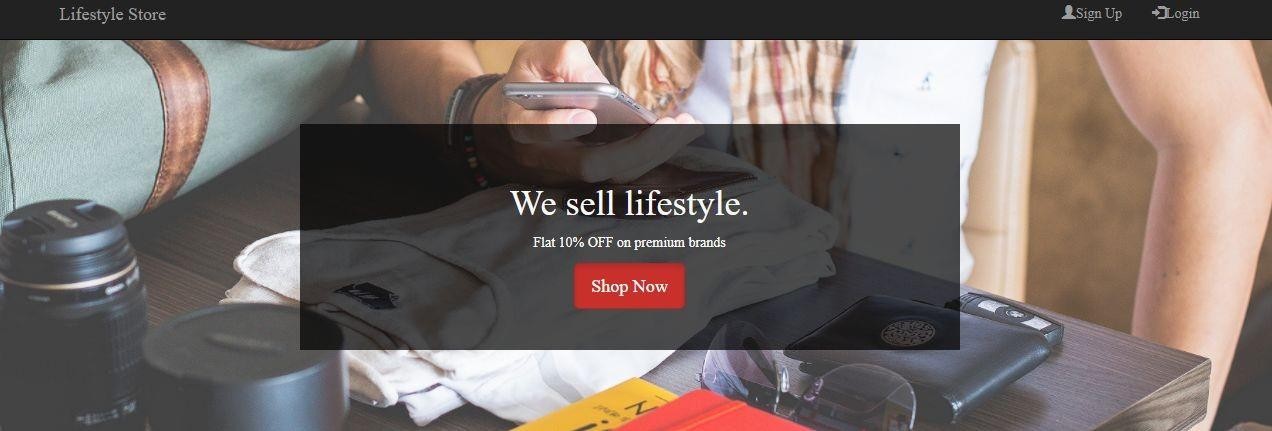
Once the production build is generated, developers can deploy the application to various hosting platforms such as Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, Netlify, Vercel, GitHub Pages, or traditional web hosting providers. Each platform may have its own deployment process and requirements, but the general steps typically involve uploading the production build files to the hosting environment, configuring any necessary server settings, and ensuring that the application is accessible to users via a public URL.

### Conclusion

React is a powerful JavaScript library for building dynamic and interactive user interfaces. Its component-based architecture promotes code reusability and maintainability. With JSX syntax and virtual DOM, React offers efficient rendering and enhanced performance. Its ecosystem of tools and libraries supports rapid development and scalability. React's popularity and robust community make it a top choice for modern web development projects.

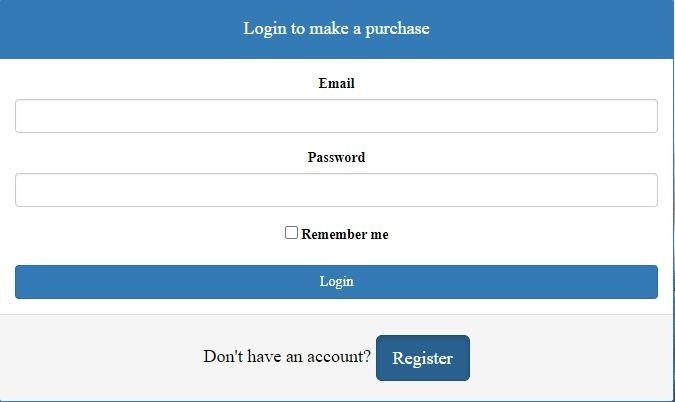
**Chapter 9**

1. **HOME PAGE :**



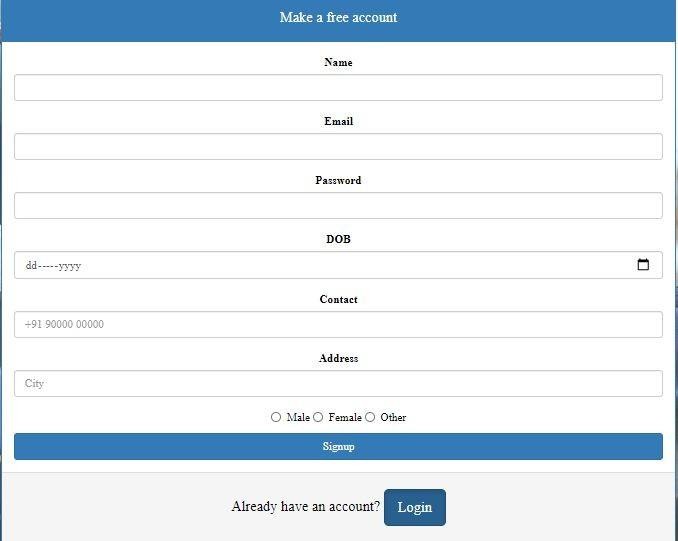
*Fig1: Lifestyle Store Index Page where customer can view different Products and Clothings*

1. **LOGIN PAGE :**



*Fig2: Lifestyle Store Login Page where customer can login to Buy or put the items into cart*

**3) SIGN UP PAGE :**



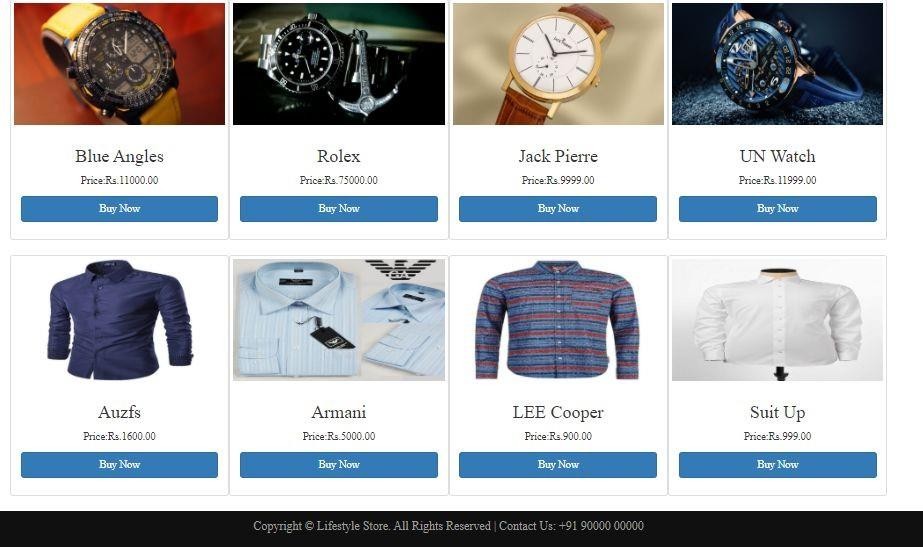
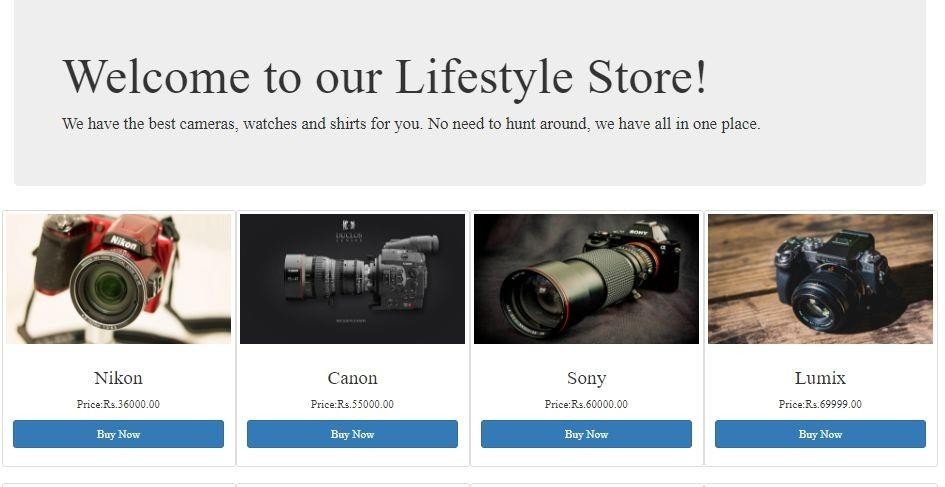
*Fig3: Lifestyle Store Signup page where customer can Sign up for the first time*

**4) SETTINGS PAGE :**



*Fig4: Lifestyle Store Settings page where customer can change the passwords*

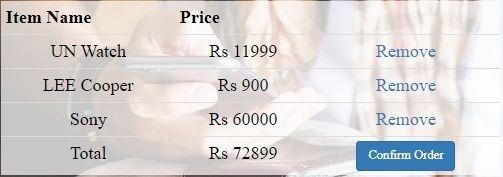
1. **PRODUCTS PAGE :**



*Fig5: Lifestyle Store Product page where customer can buy and put items into cart*

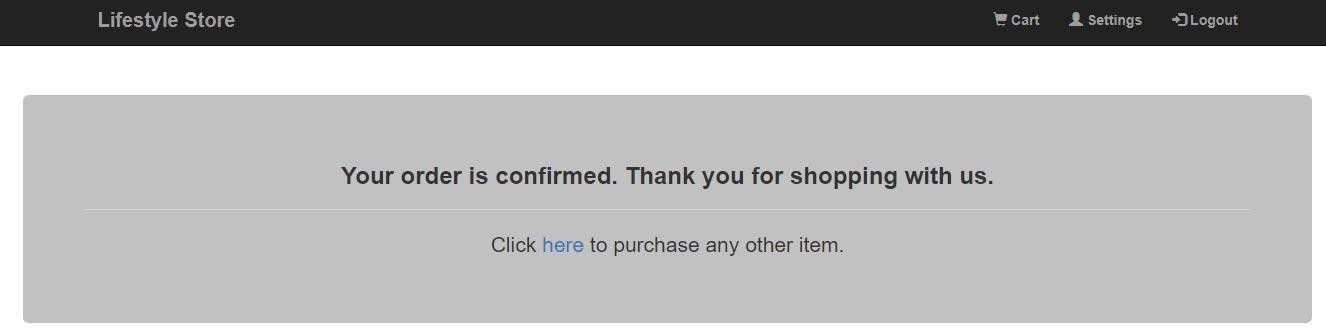
1. **CART PAGE :**





*Fig6: Lifestyle Store Cart page where customer can Confirm Order in the cart and Buy the Product*

1. **SUCCESS PAGE :**



*Fig7: Lifestyle Store Success page where customer gets the Confirmation message of the Order*

1. **NAVIGATION BAR :**



*Fig8: Lifestyle Store Success page where customer gets the Confirmation message of the Order*

# CONCLUSION

In conclusion, the web development internship experience, centered around the creation of the LifeStyleStore website, has been both enlightening and rewarding. Through this project, I've gained invaluable insights into various facets of web development, including front-end design, back- end functionality, and user experience optimization. Throughout the internship, I navigated real-world challenges, honed my problem-solving skills, and collaborated effectively with team members to bring the vision of LifeStyleStore to fruition. From initial concept to final implementation, the journey was marked by continuous learning and growth.

The LifeStyleStore website stands as a testament to the dedication, creativity, and technical expertise cultivated during the internship. It not only serves as a functional platform but also reflects a commitment to excellence in design and usability. Looking ahead, the skills and experiences gained from this internship will undoubtedly serve as a solid foundation for future endeavors in web development. As I embark on the next phase of my journey, I carry with me a deep appreciation for the opportunities afforded and the lessons learned during this transformative experience.

In essence, the web development internship, underscored by the creation of the LifeStyleStore website, has been instrumental in shaping my professional trajectory and fueling my passion for innovation in the digital landscapes.

# BIBLIOGRAPHY

* 1. Flanagan, David. "JavaScript: The Definitive Guide." O'Reilly Media, 2020.
  2. Duckett, Jon. "HTML and CSS: Design and Build Websites." Wiley, 2011.
  3. Welling, Luke, and Laura Thomson. "PHP and MySQL Web Development." Addison- Wesley Professional, 2016.
  4. Freeman, Eric, and Elisabeth Robson. "Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages." O'Reilly Media, 2012.
  5. Meyer, Eric A. "CSS: The Definitive Guide." O'Reilly Media, 2017.
  6. McFarland, David Sawyer. "JavaScript & jQuery: The Missing Manual." O'Reilly Media, 2014.
  7. Gauchat, Jean-François. "User Experience Design: A Practical Introduction." O'Reilly Media, 2019.