**A Project Report on**

**EXPLORING AND ASSESSING THE EXCELLENCE OF REACT JS FRAMEWORK**

Submitted in partial fulfillment for award of

**Bachelor of Technology**

Degree

in

**Computer Science and Engineering**

By

**V. Kotesh Nag (Y20ACS586) V. Naveen (Y20ACS587)**

**Viswajit Verma (Y20ACS585) P. Akash (Y20ACS530)**

A blue logo with a white background

Description automatically generated

Under the guidance of

**Dr. P. Pardhasaradhi, PhD,**

**Professor**

Department of Computer Science and Engineering

**Bapatla Engineering College**

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

**BAPATLA – 522 102, Andhra Pradesh, INDIA**

**2023-2024**

**Department of**

**Computer Science and Engineering**

A blue logo with a white background

Description automatically generated

**CERTIFICATE**

This is to certify that the project report entitled **EXPLORING AND ASSESSING THE EXCELLENCE OF REACT JS FRAMEWORK** that is being submitted by V. Kotesh Nag (Y20ACS586), V. Naveen (Y20ACS587), Viswajit Verma (Y20ACS585), P. Akash (Y20ACS530) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

**Signature of the Guide Signature of the HOD**

**Dr. P. Pardhasaradhi Dr. M. Rajesh Babu**

**Professor Assoc. Prof. & Head**

**DECLARATION**

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

**V. Kotesh Nag (Y20ACS586)**

**V. Naveen (Y20ACS587)**

**Viswajit Verma (Y20ACS585)**

**P. Akash (Y20ACS530)**

**Acknowledgment**

We sincerely thank the following distinguished personalities who have given their advice and support for the successful completion of the work.

We are deeply indebted to our most respected guide **Dr. P. Pardhasaradhi**, Professor, Department of CSE, for his valuable and inspiring guidance, comments, suggestions, and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending their cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly with their cooperation and encouragement.

**V. Kotesh Nag (Y20ACS586)**

**V. Naveen (Y20ACS587)**

**Viswajit Verma (Y20ACS585)**

**P. Akash (Y20ACS530)**

# Abstract

ReactJS is a prominent open-source JavaScript library used widely for building dynamic user interfaces for web and mobile applications. It excels in managing the view layer of single-page applications, offering speed, simplicity, and scalability. JSX, an optional JavaScript syntax extension that resembles HTML, is a key feature that improves code readability and maintenance, aiding in debugging with clear error messages.

React’s use of a virtual DOM optimizes rendering, enabling high-performance applications that can update dynamically without page reloads—ideal for large-scale applications needs frequent data updates. The library's component-based architecture promotes reusable code, simplifying development and testing. Components can manage their states and be nested to build complex interfaces while maintaining clear code separation, enhancing organization, and facilitating collaboration.

React is also compatible with other JavaScript libraries and frameworks, like AngularJS in MVC architectures, allowing for seamless integration into both new and existing projects. This flexibility makes React a versatile choice across different types of applications.

Overall, ReactJS stands out for its efficient handling of the view layer, enhanced by JSX, virtual DOM, and modular architecture, making it an excellent option for developers aiming to create responsive, scalable, and maintainable applications. These features, combined with strong community support and adaptability with other frameworks, position React not just as a contemporary tool but also as a sustainable choice for future front-end development endeavors.

**Table of Contents**

[Abstract v](#_Toc164246783)

[Table of Contents vi](#_Toc164246784)

[List of Figures viii](#_Toc164246785)

[List of Tables x](#_Toc164246786)

[1 Introduction 1](#_Toc164246787)

[2 React and its Prerequisites 3](#_Toc164246788)

[2.1 Introduction to React 3](#_Toc164246789)

[2.2 Key Features of React 3](#_Toc164246790)

[2.3 Required Prerequisites 4](#_Toc164246791)

[2.4 JavaScript Basics 5](#_Toc164246792)

[3 Benefits of Learning React 9](#_Toc164246793)

[3.1 Learning Curve 9](#_Toc164246794)

[3.2 React Introduced JSX 12](#_Toc164246795)

[3.3 React is Fast and Agile 12](#_Toc164246796)

[3.4 Development Community 12](#_Toc164246797)

[4 Environmental Setup 13](#_Toc164246798)

[5 Creating and Running a React Application 15](#_Toc164246799)

[6 React Core Architecture 17](#_Toc164246800)

[6.1 React Virtual DOM 17](#_Toc164246801)

[6.2 One-Way Data Flow 20](#_Toc164246802)

[6.3 Non-Functional Requirements 21](#_Toc164246803)

[6.4 Introducing JSX Syntax 22](#_Toc164246804)

[6.4.1 JSX characteristics 22](#_Toc164246805)

[6.4.2 Practical JSX 23](#_Toc164246806)

[7 React v/s Other frameworks 25](#_Toc164246807)

[7.1 React vs Angular 25](#_Toc164246808)

[7.2 React vs JQuery 26](#_Toc164246809)

[7.3 React vs ASP.Net 27](#_Toc164246810)

[7.4 React vs JSP 29](#_Toc164246811)

[8 Implementation 31](#_Toc164246812)

[8.1 Implementation of JSP 31](#_Toc164246813)

[8.2 Implementation of JQuery 33](#_Toc164246823)

[8.3 Implementation of React 34](#_Toc164246834)

[9 Results 35](#_Toc164246835)

[9.1 Page Reloading 35](#_Toc164246836)

[9.2 Component Based Architecture 36](#_Toc164246847)

[9.3 Code Reusability and Easy Debugging 38](#_Toc164246859)

[9.4 Single Page Application 40](#_Toc164246872)

[9.5 Results Page Responsiveness 41](#_Toc164246873)

[10 Conclusion and Future Enhancement 44](#_Toc164246874)

[11 Bibliography 45](#_Toc164246875)

**List of Figures**

[Figure 2.1 Traditional Syntax 6](#_Toc164124619)

[Figure 2.2 Arrow Function 7](#_Toc164124620)

[Figure 2.3 Objects 7](#_Toc164124621)

[Figure 2.4 Events 7](#_Toc164124622)

[Figure 2.5 Arrays 8](#_Toc164124623)

[Figure 3.1 Function written in the Ionic framework 1](#_Toc164124624)0

[Figure 3.2 Directive called ngRepeat written in AngularJS framework 1](#_Toc164124625)1

[Figure 3.3 Filter and map function in ReactJS 1](#_Toc164124626)1

[Figure 4.1 Node and node package manager versions 1](#_Toc164124627)3

[Figure 5.1 Syntax to create react application 15](#_Toc164124628)

[Figure 5.2 Changing to other directory 15](#_Toc164124629)

[Figure 5.3 Running the application 1](#_Toc164124630)5

[Figure 6.1 React Core Architecture 1](#_Toc164124631)8

[Figure 6.2 JSX Syntax 2](#_Toc164124632)2

[Figure 6.3 Basic Example of JSX 2](#_Toc164124633)3

[Figure 8.1 JSP implementation steps 3](#_Toc164124634)2

[Figure 8.2 JQuery implementation steps 3](#_Toc164124635)3

[Figure 8.3 React implementation steps 3](#_Toc164124636)4

[Figure 9.1 Page Reloading of JSP 3](file:///C:\Users\Windows%2010\Downloads\C5_Report%20(1).docx#_Toc164124637)5

[Figure 9.2 Page Reloading of JQuery 3](#_Toc164124638)6

[Figure 9.3 Refreshing a portion of the page in React 3](#_Toc164124639)6

[Figure 9.4 Component Based Architecture of JSP 3](#_Toc164124640)7

[Figure 9.5 Component Based Architecture of JQuery 3](#_Toc164124641)7

[Figure 9.6 Component Based Architecture of React 38](#_Toc164124642)

[Figure 9.7 Code Optimality in JSP](#_Toc164124643) 39

[Figure 9.8 Code Optimality in JQuery 39](#_Toc164124644)

[Figure 9.9 Code Optimality in React 39](#_Toc164124645)

[Figure 9.10 Single Page Application of JSP 4](#_Toc164124646)0

[Figure 9.11 Single Page Application of JQuery 41](#_Toc164124647)

[Figure 9.12 Single Page Application of React 41](#_Toc164124648)

[Figure 9.13 Page Responsiveness in JSP 42](#_Toc164124649)

[Figure 9.14 Page Responsiveness in JQuery 4](#_Toc164124650)2

[Figure 9.15 Page Responsiveness in React 43](#_Toc164124651)

**List of Tables**

[Table 7.1 React vs Angular 2](#_Toc164124749)6

[Table 7.2 React vs JQuery 2](#_Toc164124750)7

[Table 7.3 React vs ASP.Net](#_Toc164124751) 28

[Table 7.4 React vs JSP 29](#_Toc164124752)

# Introduction

ReactJS, a popular open-source front-end JavaScript library developed by Facebook, has revolutionized web development with its efficient rendering mechanism and component-based architecture. React simplifies the process of building interactive user interfaces by breaking them down into reusable, modular components. Its popularity among developers stems from its ability to minimize DOM manipulation through the use of a virtual DOM, resulting in faster UI updates and improved performance. This core feature allows developers to focus on creating dynamic user experiences without being hindered by performance constraints.

Central to React's design are its core concepts, including the component-based architecture, unidirectional data flow, and JSX syntax. Components serve as the building blocks of React applications, encapsulating both UI elements and their associated logic. The unidirectional data flow ensures predictable state management, with data flowing from parent to child components. JSX, a JavaScript extension, enables developers to write HTML-like code within JavaScript, enhancing code readability and maintainability. These foundational principles empower developers to create complex user interfaces with ease while maintaining code clarity.

In addition to its core features, React offers a wide range of functionalities and a robust ecosystem that further enhances its capabilities. React Hooks, introduced in React 16.8, simplify state management and side-effect handling in functional components, eliminating the need for class components. React Router facilitates declarative routing in single-page applications, enabling seamless navigation between different views. The Context API provides a centralized way to manage global state, reducing prop drilling and simplifying data sharing between components. Moreover, React's server-side rendering (SSR) capability improves SEO and initial page load times by rendering HTML on the server.

React enjoys a vibrant community of developers, contributors, and enthusiasts who actively contribute to its growth and evolution. The React ecosystem is rich with libraries, tools, and resources that cater to various development needs, including state management, styling, and testing. This thriving ecosystem, combined with React's popularity and ease of use, has led to widespread adoption across industries and domains. From startups to tech giants, organizations are embracing React for its efficiency, scalability, and performance.

Looking ahead, ReactJS remains poised to shape the future of web development. With ongoing advancements such as React Concurrent Mode and Suspense, developers can expect even greater capabilities and performance enhancements in future versions of React. As businesses increasingly rely on web and mobile applications to engage with users, React provides a solid foundation for building cutting-edge experiences that meet the demands of today's digital landscape. In conclusion, ReactJS continues to be a driving force in web development, empowering developers to create innovative and compelling user interfaces.

# React and its Prerequisites

React is a popular JavaScript library for building user interfaces, particularly for single-page applications and interactive web experiences. Before diving into React features, it's beneficial to have a good understanding of the understanding of React and certain prerequisites.

## Introduction to React

React is a JavaScript library developed by Facebook for building user interfaces, particularly for single-page applications where UIs need to update dynamically over time. It allows developers to create reusable UI components that manage their own state, making it easier to build complex and interactive web applications.

## Key Features of React

**Component-Based Architecture:** React follows a component-based architecture, where UIs are broken down into small, reusable components. These components encapsulate their own logic and state, making it easier to manage and maintain large applications.

**Virtual DOM:** React uses a virtual DOM (Document Object Model) to efficiently update the UI. Instead of updating the entire DOM tree when changes occur, React compares the virtual DOM with the real DOM and only updates the parts that have changed. This results in faster rendering and better performance.

**Declarative Syntax:** React uses declarative syntax, allowing developers to describe how the UI should look based on the current state. This makes it easier to understand and maintain code compared to imperative approaches.

**JSX:** JSX (JavaScript XML) is a syntax extension that allows developers to write HTML-like code within JavaScript. JSX makes it easier to create React components by combining UI elements and JavaScript logic in a single file.

**Unidirectional Data Flow:** React follows a unidirectional data flow, where data flows only in one direction—from parent to child components. This makes it easier to debug and understand how data changes propagate through the application.

Overall, React simplifies the process of building dynamic and interactive user interfaces, making it a popular choice for front-end development.

## Required Prerequisites

**HTML/CSS Fundamentals:** Understanding of HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) is fundamental. HTML provides the structure for web pages, while CSS is used for styling and layout.

**Proficiency in JavaScript:** JavaScript is the core language used in React development. You should have a strong grasp of JavaScript concepts such as variables, functions, arrays, objects, control flow, and more.

Additionally, familiarity with ES6 (ECMAScript 2015) and newer JavaScript features is essential as React heavily relies on these modern language features.

**Node.js and Package Managers (npm/yarn):** Node.js is a JavaScript runtime environment used for executing JavaScript code outside of a browser. It's often used in React development for running build tools, servers, and scripts.

npm (Node Package Manager) and yarn are package managers for Node.js. They are used to install, manage, and update packages and dependencies required for React projects.

**Version Control with Git:** Git is a distributed version control system used for tracking changes in code. Understanding Git basics such as cloning repositories, committing changes, branching, and merging is crucial for collaborating on React projects and managing code effectively.

**Command Line/Shell Proficiency:** Basic knowledge of the command line or shell is necessary for tasks like navigating directories, running scripts, installing dependencies, and executing Git commands.

**Familiarity with JSX Syntax:** JSX (JavaScript XML) is a syntax extension used in React to describe the structure of UI components. It allows developers to write HTML-like code within JavaScript files, making the creation of React components more intuitive. Understanding JSX syntax is essential for building React applications efficiently.

By mastering these prerequisites, you'll be well-equipped to start learning React and building powerful user interfaces for web applications.

## JavaScript Basics

JavaScript is a versatile programming language commonly associated with web development. While it traditionally executes within web browsers, Node.js extends its capabilities to server-side environments, enabling broader applications. Here's a structured breakdown:

**Variables:** JavaScript supports three variable declaration types:

**var**: Allows redeclaration and reassignment.

**let**: Prevents redeclaration but permits reassignment.

**const**: Disallows both redeclaration and reassignment.

**Data Types:** JavaScript comprises primitive and non-primitive data types:

**Primitive:** Includes numbers, booleans, strings, symbols, undefined, null, and bigint.

**Non-Primitive:** Primarily consists of objects.

**Functions:** Functions can be declared using traditional or arrow notation:

**Traditional Syntax:** Traditional syntax in JavaScript refers to the syntax and structure commonly used before the introduction of modern ES6 (ECMAScript 2015) features. While ES6 brought significant improvements and syntactic sugar to JavaScript, understanding traditional syntax is still valuable as it forms the foundation of the language. Here's an overview of traditional function syntax:

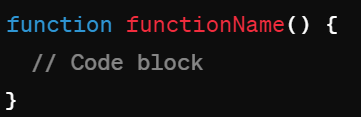


Figure 2.1 Traditional Syntax

**Arrow Function:** Arrow functions in JavaScript provide a concise way to write function expressions. They were introduced in ES6 (ECMAScript 2015) and have since become a popular feature due to their simplicity and compact syntax. Here's an overview of arrow function syntax:

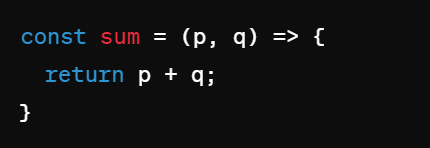


Figure 2.2 Arrow Function

**Objects:** In JavaScript, an object is a standalone entity, with properties and type. Objects in JavaScript can be compared to objects in real life. Just as the concept of objects in real life includes properties and behaviors, objects in JavaScript consist of properties, which are essentially key-value pairs. Here's an overview of object syntax:

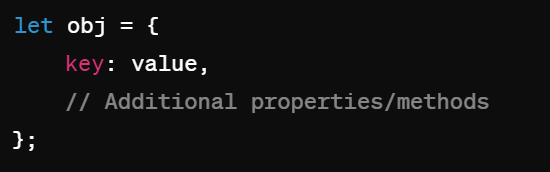


Figure 2.3 Objects

**Events:** Events in JavaScript are actions or occurrences that happen in the browser. These events can be triggered by user interaction, such as clicking a button, hovering over an element, or typing on the keyboard. They can also be triggered by the browser itself, like when a page finishes loading. Here's an overview of event syntax:

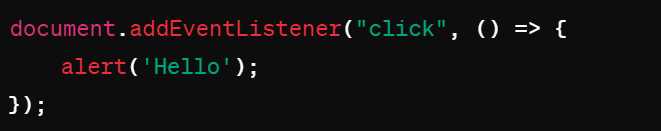


Figure 2.4 Events

**Popup Boxes:** JavaScript provides built-in popup boxes like alert(), confirm(), and prompt().

**Arrays:** Arrays in JavaScript provide a convenient way to store and manipulate collections of data. They are ordered, indexed collections of elements, where each element can be accessed by its index. Arrays in JavaScript are dynamic, meaning their size can be modified during runtime.

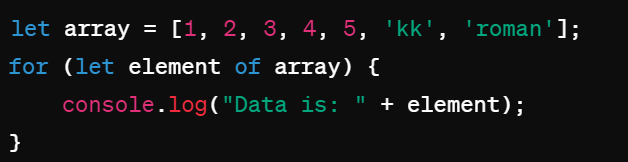


Figure 2.5 Arrays

**Asynchronous Behavior:** JavaScript operates asynchronously, utilizing callbacks for asynchronous tasks like ***setTimeout().***

**Operators:** Various operators include arithmetic, assignment, comparison, and logical operators.

**Comments:** Comments can be single-line **(//)** or multi-line **(/\* \*/).**

**Control Flow:** Control flow includes conditional statements (***if, if else, else if ladder, switch case***), ternary operator (***condition ? expression1 : expression2***), and loops (***for, for...in, for...of, while, do...while***).

**String to Integer Conversion:** Use ***Number.parseInt()*** to convert strings to integers.

This comprehensive overview provides a solid foundation for understanding JavaScript fundamentals.

# Benefits of Learning React

React was introduced to the world two years ago, and since then it has seen impressive growth, both inside and outside Facebook. New web projects at Facebook are commonly built using React in one form or another, and it is being broadly adopted across the industry. Developers and engineers are choosing React because it allows spending more time to focus more on the product development and less time spent on fighting and learning the framework.

A React application is a collection of discrete components, each representing a single view. The idea of every individual view component makes it easy to iterate on product development because to make changes on a single view or component, it is not necessary to consider the entire system. When an application is built with React, the code is generally predictable, it is because React wraps the DOM mutative, imperative API with a declarative one, which raises the level of abstraction and simplifies the programming model. Moreover, it is easier to scale the application built with React.

The combination of React and the rapid iteration cycle of the web, has enabled it to make some excellent products including many Facebook components. An amazing JavaScript framework called Relay has also been made on top of React, which helps simplifying data fetching on a large scale.

## Learning Curve

Unlike some other JavaScript libraries where it takes a lot of time to learn about the frameworks, in React it does not take much of an effort to start building an application. React has many strong features. Readability is one of the greatest strengths of React. It is easily readable even to those who are unfamiliar with it. While other frameworks require learning many concepts about the framework itself, ignoring the language fundamentals, React does the absolute opposite. For example, let’s consider how different it is in React and Ionic (AngularJS) rendering a portion of an employer’s list.

In Ionic, it requires the use of a directive called ngRepeat. Let’s assume an array of employers. Each of them contains the following fields: first\_name, last\_name, is\_married.

The target is to show only employers who are married. The following Figure 3.1 shows a screenshot of a function written in the Ionic framework.



Figure 3.1 Function written in the Ionic framework

As shown in Figure 3.1, a function is written called EmployerCtrl where it shows some specific information of the employers.

Figure 3.2 shows a screenshot of a directive called ngRepeat written in AngularJS framework.

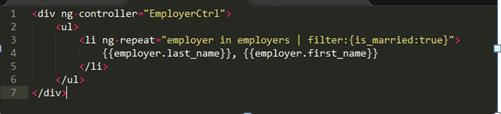


Figure 3.2 Directive called ngRepeat written in AngularJS framework

As shown in Figure 3.2, this is how a directive is written in the AngularJS framework.

If one is not familiar with Ionic/Angular, this code snippet may raise some immediate questions of what is the $scope and what is the specific syntax here for the filter.

But in React, one can use the existing knowledge of language fundamentals. The above functionality can be done using filter and map functions in React. The following Figure 3.3 shows how to write the function in ReactJS.



Figure 3.3 Filter and map function in ReactJS

Figure 3.3 Screenshot of how to use the filter and map function in ReactJS.

As shown in Figure 3.3, functions can be written in easy ways compared to other frameworks. A few questions may also arise regarding what React create class does and what render is but the rest of the code is regular JavaScript. This means it will be quite easy and understandable to those who do not even know much about React.

## React Introduced JSX

JSX is a language that lets you specify the DOM elements before the components right inside of JavaScript files. This means the logic behind the components and the visuals are all in one place. This is such a great idea when other frameworks are taking queues to place them.

## React is Fast and Agile

ReactJS is featured with a one-way unidirectional data flow between the states and layers in an application. This means data flows in a single direction between the application states and layers. In two-way data binding like Angular, if a model is changed, the view also changes and vice-versa. React renders the updates in the DOM much quicker than alternative frameworks and it is a much smaller library. DOM means document object model. Thus, it is easy to choose the tools to get the job done.

## Development Community

Big companies like the New York Times, Facebook, and Netflix are using React in production. They are continuously contributing to developing the React core and building amazing third-party libraries that work with any React applications.

# Environmental Setup

**Node.js Installation:**

**Download Node.js:** Visit the official Node.js website:[*Node.js Download*](https://nodejs.org/)

Download the appropriate installer for your operating system (Windows, macOS, Linux).

**Verify Installation:** After installation, open your terminal (Command Prompt, PowerShell, or Terminal).

Run the following commands to verify that Node.js and npm (Node Package Manager) are installed.



Figure 4.1 Node and node package manager versions

**Visual Studio Code (VSCode) Installation:**

**Download VSCode:** Visit the official VSCode website: [*VSCode Download*](https://code.visualstudio.com/)

Download the installer for your operating system (Windows, macOS, Linux).

**Enable All Checkboxes:** During installation, make sure to enable all checkboxes for optional features if prompted.

**Browse React Developer Tools:** Open your web browser (Chrome, Firefox, etc.).Search for "React Developer Tools" in the browser's extension marketplace.

Add the extension to your browser (e.g., Chrome: [React Developer Tools - Chrome Web Store](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi)).

**Visual Studio Code Extensions:**

**Open VSCode:** After installing, open Visual Studio Code.

**Install Extensions:** Open the Extensions view by clicking on the square icon in the sidebar or pressing **Ctrl+Shift+X** (Windows/Linux) or **Cmd+Shift+X** (macOS).

Search for the following extensions and install them:

**Thunder Client:** REST Client for VSCode.

**ES7 React/Redux/GraphQL/React-Native snippets:** Code snippets for React, Redux, GraphQL, and React Native.

**Bracket Pair Colorizer:** Matches brackets with the same color.

**Auto Rename Tag:** Automatically renames paired HTML/XML tags.

**Live Server:** Launches a local development server with live reload capability.

**Prettier - Code formatter:** Code formatter for JavaScript, TypeScript, CSS, and more.

# Creating and Running a React Application

**Initialise React App:** Use the following command to create a new React application named "firstapp"



Figure 5.1 Syntax to create react application

**Start Development Server:** Navigate into the newly created "firstapp" directory:



Figure 5.2 Changing to other directory

**Start the development server by running**

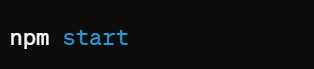


Figure 5.3 Running the application

**Directory Structure:**

The React application contains two main directories:

**public:** Contains the index.html file, which is the main HTML file for your React application.

**src:** Contains the source code of your React application, including components, styles, and other assets.

**Props and State:**

**Props:** Data that can be passed from parent to child components. Props are immutable.

**State:** Represents the details managed within a component. The State can be updated and trigger re-renders.

**Key Files:**

**public/index.html:** The main HTML file where your React app is rendered. This file includes a div with an id of "root", where React renders your components.

**src/index.js:** The entry point of your React application, where React is initialized and the root component is rendered.

**Running the Application:** After starting the development server with **npm start**, your React application will automatically open in your default web browser.

**Alternative Method** (Without **create-react-app**): You can also create a basic HTML file and serve it using a local development server like "Live Server" in VS Code. Simply create an HTML file with your desired content and right-click on it in VS Code. Choose "Open with Live Server" to launch the file in your browser. By following these steps, you'll have your first React application up and running, ready for further development!

# React Core Architecture

React has become one of the most popular JavaScript libraries for building user interfaces, thanks to its simplicity, performance, and reusability. At its core, React offers a component-based architecture that enables developers to create modular, maintainable, and scalable applications. Let's delve into the fundamental aspects of React’s architecture:

## React Virtual DOM

DOM stands for Document Object Model. DOM manipulation is very important for modern interactive web technologies. It is often called the heart of the modern web. It is an abstraction of the structured text. But it works slower than other JavaScript operations because most JavaScript frameworks usually update the DOM even if they do not need to do it. That means those updates are not necessarily required to perform the actions but they still do by default. For example, let us assume nine items have been put in a shopping basket in an online web store. Now let us say only the first item is needed to buy and proceed to checkout. Here, most technologies would rebuild the entire list that has been put in the basket. This means the framework has to unnecessarily work ten times more. Because of only one change the system has to rebuild the list exactly how it was before.

React did not invent Virtual DOM but uses and provides it to the developer community for free. Virtual Dom is simply an abstraction of HTML DOM. React has a corresponding virtual DOM object for every DOM object like a correspondent or a lightweight copy. Virtual DOM is also characterized by similar properties to a real DOM. However, it cannot make any changes directly to the view. DOM manipulation is quite a slow process. But manipulating Virtual DOM is faster because it has nothing to do with the view part and does not make any changes to the screen. Figure 13, reprinted from stackoverflow.com, is an illustration of Virtual DOM in the memory.

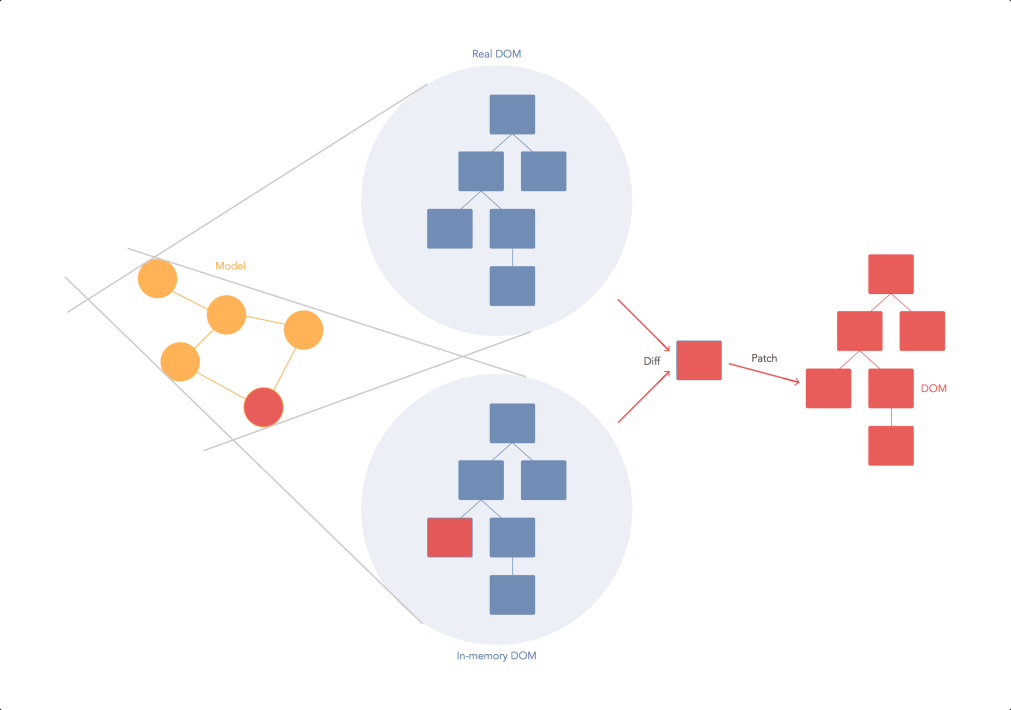


Figure 6.1 React Core Architecture

Figure 6.1 Screenshot of React virtual Dom in memory. Reprinted from stackoverflow.com (2017). As shown in Figure 6.1, a React virtual DOM in the memory is a lightweight copy of the real DOM. React uses a method called “diffing” which means rendering a JSX element gets every single Virtual DOM updated. This might sound inefficient but it costs nothing as Virtual DOM is quite fast to get updated and does not make any impact in the process. After the DOM gets updated React compares the updated DOM with a pre-updated state of the DOM and determines which virtual DOM has been changed. Once React detects the changed DOMs, React updates only those objects to the real DOM.

Thus, React makes the updates faster through Virtual DOM. In the above-mentioned example, React would have updated only the checked item from the list and left the rest of the items alone. This makes the difference when updating a page in an application while React can only make changes to the necessary parts of the DOM. This virtual DOM manipulation process is one of the main reasons why React is gaining much popularity among the developer communities.

**React Virtual DOM Pros and Cons**

Among the many advantages of the ReactJS library, a few of the key advantages are described here:

* The different algorithms written in React are quite fast and efficient
* The inclusion of JSX and hyperscript lets us build multiple frontends for the same application.
* It is very lightweight and capable to be run on every mobile device
* Lots of tractions and mindshare
* It can also be used without React as an independent engine
* A few disadvantages of React are as follows.
* It occupies quite much of the memory. Full in-memory copy of the DOM.
* Static and dynamic elements don’t make much of a difference.

## One-Way Data Flow

Frameworks like Angular and Ember use two-way data binding. In a two-way data binding for example in Angular, if a model is changed, the view also automatically gets changed and vice versa. An input field in the model can also mutate the model. It performs well in most of the applications but sometimes it may lead to cascading updates and changing to one model may cause updates in other models. Again, since the state is mutable by both view and controller, the data flow can be unpredictable in some cases. Flux or Redux with React can be a better solution to avoid those uncertainties since both architectures follow a one-way data flow. One-way data flow does not make cascading updates and changes in view.

One-way data flow ensures that data flows throughout the application in a single direction to offer more control between the states and models in an application. One-way data flow also makes the architecture less complicated and more understandable. Flux architecture is a functional approach. Here the view is considered as a function of the application state. Eventually, if the state gets some changes the view also gets re-rendered automatically. Moreover, a similar view is generated from the states and gives a better understanding and predictability to the application.

To make it more predictable, in an application, data from parent to child flows in a single direction. Any data can be updated from any view, and anytime in this approach. In case something goes wrong, debugging is also made less complicated in this way.

## Non-Functional Requirements

Non-functional requirements are the characteristics and properties that a system must possess to be considered effective, reliable, and usable. Here are some non-functional requirements of a drowsiness detection system:

**Accuracy:** The system must be highly accurate in detecting signs of drowsiness. It should have a low false-positive rate, meaning that it should not trigger an alert when the driver is not drowsy, and a low false-negative rate, meaning that it should not miss signs of drowsiness when they are present.

**Reliability:** The system must be reliable and consistent in its operation. It should function properly under varying conditions, including different lighting, weather, and driving environments.

**Responsiveness:** The system must be responsive, providing alerts to the driver promptly. It should detect signs of drowsiness quickly and alert the driver before they become a danger on the road.

**Usability:** The system must be easy to use and understand. The alerts provided to the driver should be clear and easily understandable, and the system should not require extensive training or expertise to operate.

**Scalability:** The system should be scalable, meaning that it can be easily adapted to different types of vehicles and driving scenarios. It should also be able to handle large amounts of data and be able to support multiple users at the same time.

**Security:** The system must be secure and protect the privacy of the driver. It should not collect or store any personal data that could be used to identify the driver, and it should be designed to prevent unauthorized access or tampering.

**Performance:** The system must perform well under a range of conditions and be able to handle large amounts of data. It should be designed to be efficient and minimize processing time, while also being able to handle multiple tasks at once.

Overall, these non-functional requirements are critical to the success of a drowsiness detection system. By meeting these requirements, the system can provide reliable, accurate, and timely alerts to the driver, helping to prevent accidents and improve road safety.

## Introducing JSX Syntax

JSX is neither a string nor an HTML. It is a statically typed syntax extension to JavaScript. It is similar to an object-oriented language which is designed to run on modern web browsers. JSX is recommended to be used with React to design and build the user interface. While it comes with the full power of JavaScript it might even seem as a template language too at first glance though it is not. The React element is produced by JSX. It can be rendered to the React Virtual DOM.



Figure 6.2 JSX Syntax

### JSX characteristics

JSX has got some unique features that made JSX quite popular among React and React Native developers. At the beginning, it may look difficult but with time adopting JSX is easy.

First of all, it is faster: While JSX source code is compiled into JavaScript, it shows a very optimized result. Compared to the equivalent code written in JavaScript, JSX-generated code runs faster. JSX has proved to be 12 % faster in iOS and 29 % faster in Android.

Secondly, it is safer: In contrast to JavaScript, JSX is statically typed and mostly typesafe. The quality of applications becomes higher when being developed using JSX since many errors will be caught during the compilation process. It also offers debugging features at the compiler level as well.

Thirdly, it is easier: JSX offers a solid class system much like Java, freeing the developers from working with the too-primitive prototype-based inheritance system provided by JavaScript. Expressions and statements, however, are mostly equal to JavaScript, so it is easy for JavaScript programmers to start using JSX. There are also plans for language services for editors / IDEs, for example, code completion to make coding easier.

### Practical JSX

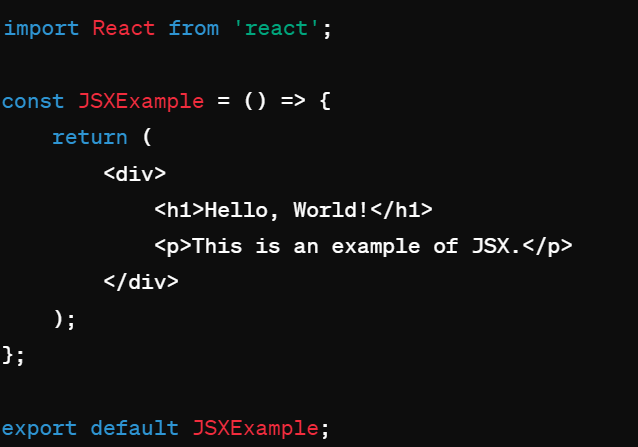


Figure 6.3 Basic Example of JSX

In the example:

JSX allows you to write HTML-like syntax directly within JavaScript code.

You can include HTML elements, attributes, and even JavaScript expressions inside curly braces {} within JSX.

The **return** statement within the component function returns JSX elements, which will be rendered as UI components when the component is used.

# React v/s Other frameworks

React, Angular, and Vue.js are frequently compared in frontend development circles due to their widespread usage and distinct characteristics. React shines with its flexible, component-based architecture, enabling developers to efficiently create reusable UI elements. In contrast, Angular offers a comprehensive framework with a structured approach, featuring functionalities like two-way data binding and dependency injection. Vue.js prioritizes simplicity and user-friendliness, making it an attractive option for smaller projects or those new to front-end frameworks. When selecting between these options, factors such as project complexity, team expertise, and specific feature requirements play crucial roles. By recognizing the unique strengths of each framework, developers can make well-informed decisions tailored to their project needs.

## React vs Angular

React and Angular are both popular choices for building dynamic web applications. React, developed by Facebook, is a JavaScript library known for its component-based architecture and virtual DOM, offering flexibility and performance optimizations. It's favored for its simplicity and extensive ecosystem. On the other hand, Angular, backed by Google, is a full-fledged framework with features like two-way data binding and dependency injection, providing a comprehensive solution for large-scale projects. While React has a shallower learning curve and offers more flexibility in choosing libraries and tools, Angular's built-in solutions and strong support make it suitable for enterprise-level applications. Ultimately, the choice between React and Angular depends on project requirements, development team expertise, and preference for framework vs. library approach.Top of Form

Table 7.1 React vs Angular

|  |  |  |
| --- | --- | --- |
| **Aspect** | **React** | **Angular** |
| Setup | Easy, using tools like Create React App | Requires Angular CLI, more initial configuration |
| Component Structure | Flexible. Components manage their own state, and communication between components involves passing props | Structured. Components are part of modules, and services handle state and communication |
| State Management | Uses one-way data flow with state and props. Can integrate external libraries like Redux for more complex apps | Built-in with services, dependency injection, and centralized state |
| Template Syntax | JSX (HTML in JavaScript) | Declarative HTML and TypeScript |
| Learning Curve | Easier for beginners, gradual learning curve | Steeper for beginners due to opinionated structure |
| Community | Large and active, plenty of libraries | Strong community |
| Flexibility | More flexible, can be integrated into existing projects. Allows choice of state management tools | Less flexible due to opinionated structure. |
|  |  |

## React vs JQuery

React and jQuery serve different purposes in web development. React is a JavaScript library for building user interfaces, focusing on component-based architecture and virtual DOM for efficient rendering. It promotes the creation of reusable and maintainable UI components, making it suitable for large-scale applications with complex user interfaces. On the other hand, jQuery is a fast and concise JavaScript library that simplifies HTML document traversing, event handling, animating, and Ajax interactions. It's primarily used for DOM manipulation and simplifying cross-browser compatibility. While React emphasizes declarative and efficient UI development, jQuery provides a lightweight solution for enhancing and manipulating HTML elements. In modern web development, React is often preferred for building dynamic and interactive web applications, while jQuery is commonly used for quick fixes or enhancements to existing projects.

Table 7.2 React vs JQuery

|  |  |  |
| --- | --- | --- |
| **Aspect** | **jQuery** | **React** |
| Code Organization | Tends to lead to messy and hard-to-maintain code as the application grows. | Encourages breaking down UI into smaller, reusable components, improving code organization. |
| State Management | Tricky, especially as the application gets larger. | Provides a built-in way to manage component state, enhancing robustness. |
| Performance | Directly manipulates HTML DOM, potentially leading to performance issues with complex UIs. | Uses a virtual DOM for efficient updates, resulting in better performance, especially for dynamic applications. |
| Debugging & Testing | Challenging due to the procedural nature of the code. | Component-based architecture makes debugging and testing easier, each component is isolated and self-contained. |
|  |  |  |

## React vs ASP.Net

React is a JavaScript library for building user interfaces, focusing on component-based development and efficient rendering through the use of a virtual DOM. It's commonly used for creating interactive and dynamic frontend experiences in web applications. On the other hand, ASP.NET is a web development framework developed by Microsoft, primarily used for building backend services and APIs. It provides a comprehensive set of tools and libraries for building robust and scalable web applications, with features like MVC architecture, routing, authentication, and data access. While React is primarily used for frontend development, ASP.NET covers both frontend and backend aspects of web development, offering a full-stack solution for building modern web applications. Choosing between React and ASP.NET often depends on the specific requirements of the project, such as the need for a frontend framework, existing technology stack, and development expertise.

Table 7.3 React vs ASP.Net

|  |  |  |
| --- | --- | --- |
| **Aspect** | **React** | **.NET (ASP.NET MVC)** |
| Language | Frontend: Primarily JavaScript, JSX for component syntax. Backend: This can be used with various backend technologies (Node.js, Python, etc.). | Entirely written in C#. |
| Rendering | Client-side: Components rendered in the browser. | Server-side rendering, where the server generates HTML. |
| Development Stack | Frontend: React for UI components. Backend: This can be connected to any backend technology. | Full-stack framework for building web applications in C#. Integrated with Visual Studio IDE. |
| State Management | Utilizes React Hooks for managing component state. Follows a unidirectional data flow. | Utilizes MVC pattern with models and controllers for state management. The server manages the application state. |
| Deployment | Deployed as static files. Works on any web server or cloud platform. | Deployed on ASP.NET servers. Often used in enterprise applications. |
|  |  |  |
|  |  |  |

## React vs JSP

React is a JavaScript library for building interactive user interfaces, developed by Facebook. It emphasizes component-based architecture, allowing developers to create reusable UI components for building dynamic web applications. React uses a virtual DOM to optimize rendering performance and offers JSX for writing HTML within JavaScript, enhancing code readability. On the other hand, JSP (JavaServer Pages) is a technology used for building dynamic web pages in Java. It enables embedding Java code within HTML to generate dynamic content on the server-side. While React focuses on client-side rendering and offers a more modern approach to building web applications, JSP is a server-side technology often used in Java web applications for generating dynamic content before it's sent to the client. Overall, React is preferred for building highly interactive and responsive user interfaces, while JSP is suitable for server-side rendering and integrating Java logic into web applications.

Table 7.4 React vs JSP

|  |  |  |
| --- | --- | --- |
| **Aspect** | **React** | **JSP (Java Server Pages)** |
| Language | Primarily JavaScript, JSX for component syntax. | Primarily Java for server-side logic and HTML for the view. |
| Rendering | Client-side rendering, where components are rendered in the browser. | Server-side rendering, where the server generates HTML. |
| Componentization | Encourages breaking down UI into reusable components, improving code organization and maintainability. | Supports server-side templating but may lack the modular componentization of React. |
| State Management | Utilizes React Hooks for managing component state. Follows a unidirectional data flow. | May require additional frameworks or libraries for managing server-side state effectively. |
| Performance | Uses a virtual DOM for efficient updates, resulting in better performance, especially for dynamic applications. | Performance can vary depending on server-side processing and caching mechanisms. |
| Learning Curve | May have a steeper learning curve, especially for developers new to JavaScript or modern frontend development practices. | Familiar for Java developers, potentially lowering the learning curve. |
| Ecosystem | Large and active community with extensive libraries and tools available. | May have a narrower ecosystem compared to JavaScript-based frontend frameworks. |
| Deployment | Deployed as static files, works on any web server or cloud platform. | Typically deployed on Java servers, requires Java-based infrastructure. |

# Implementation

The implementation of a project involves translating plans into action. It encompasses assigning tasks, setting milestones, and executing activities according to the project plan. Team members collaborate, monitor progress, and address challenges as they arise, ensuring that the project stays on track. Quality assurance measures are implemented to maintain high standards, and risks are managed to minimize disruptions. Documentation is kept throughout the process, and once deliverables are completed and approved, they are deployed. Training and support are provided to stakeholders, and the project is evaluated against its objectives, with lessons learned applied for future endeavors. Effective communication, flexibility, and adherence to best practices are crucial for successful project implementation.

## Implementation of JSP

The implementation of a JSP (JavaServer Pages) application involves translating the project specifications into functional web pages using Java-based technologies. This typically includes creating dynamic web content by embedding Java code within HTML templates. Developers utilize servlets to handle requests, process data, and generate dynamic content, which is then rendered as HTML to be displayed in the user's web browser. JSP allows for the seamless integration of Java logic with HTML, enabling the creation of interactive and data-driven web applications. Throughout the implementation process, developers focus on coding, testing, and refining the application to ensure its functionality, performance, and usability meet the project requirements. Additionally, deployment procedures are followed to make the application accessible to users, often involving the setup of a web server and database connectivity for full functionality.

Figure 8.1 JSP implementation steps

**URL to Access the Code:** **https://github.com/kingkotesh/Final\_Year\_Project.git**



## Implementation of JQuery

The implementation of a jQuery application involves integrating jQuery library into web pages to enhance client-side functionality and user interactions. After including the jQuery library, developers utilize its powerful features to manipulate HTML elements, handle events, make AJAX requests, and create animations, among other tasks, simplifying complex JavaScript operations. Implementation typically follows a structured approach, where developers identify specific elements to target with jQuery selectors, apply desired actions or effects using jQuery methods, and ensure compatibility across different browsers. jQuery's intuitive syntax and extensive documentation facilitate rapid development and deployment of interactive web applications, making it a popular choice for frontend development projects. Throughout the implementation process, thorough testing and optimization are essential to ensure smooth performance and a seamless user experience.

Figure 8.2 JQuery implementation steps

**URL to Access the Code:** **https://github.com/kingkotesh/Final\_Year\_Project.git**



## Implementation of React

The implementation of a React application involves translating design mockups and user requirements into functional components using React's declarative syntax. Developers write JSX to define the UI structure and logic, incorporating state management with tools like Redux or React's built-in state management capabilities. They integrate APIs for data retrieval and manipulation, ensuring seamless communication between the frontend and backend. Throughout implementation, developers leverage React's component-based architecture to build reusable UI elements, enhancing efficiency and maintainability. Testing, debugging, and optimizing performance are integral parts of the implementation process, ensuring a smooth user experience across devices and browsers. Finally, deployment involves bundling the application for production and configuring deployment pipelines for continuous integration and delivery.

Figure 8.3 React implementation steps

**URL to Access the Code: https://github.com/kingkotesh/Final\_Year\_Project.git**

# Results

## Page Reloading

In React, updating a portion of the page involves dynamically refreshing specific components or elements without reloading the entire page. This is made possible through React's virtual DOM, which efficiently updates only the parts of the UI affected by data changes or user interactions. By leveraging a component-based architecture and state management, React minimizes unnecessary re-renders and optimizes performance, resulting in a smoother and more responsive user experience. Additionally, client-side routing with libraries like React Router enables seamless navigation between different views or pages without triggering full page reloads, further enhancing the interactivity of React applications.

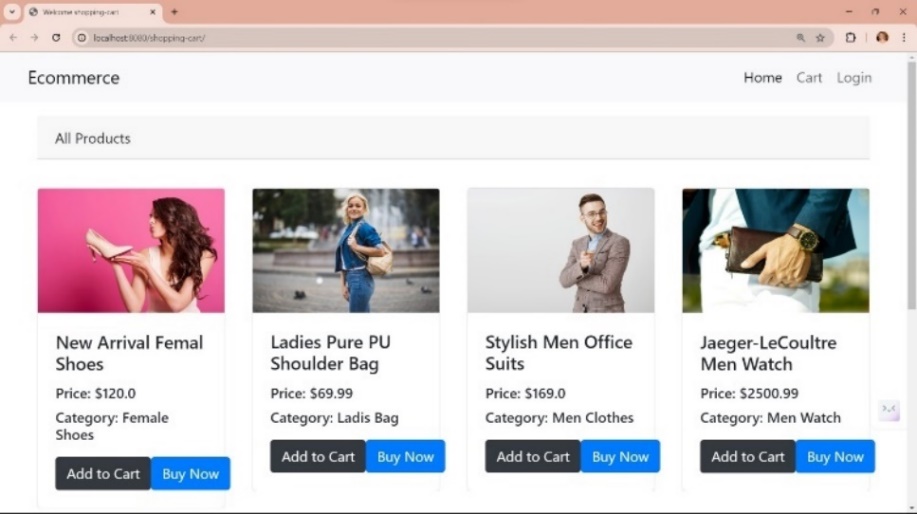
****

Figure 9.1 Page Reloading of JSP

**A screenshot of a website

Description automatically generated**

Figure 9.2 Page Reloading of JQuery

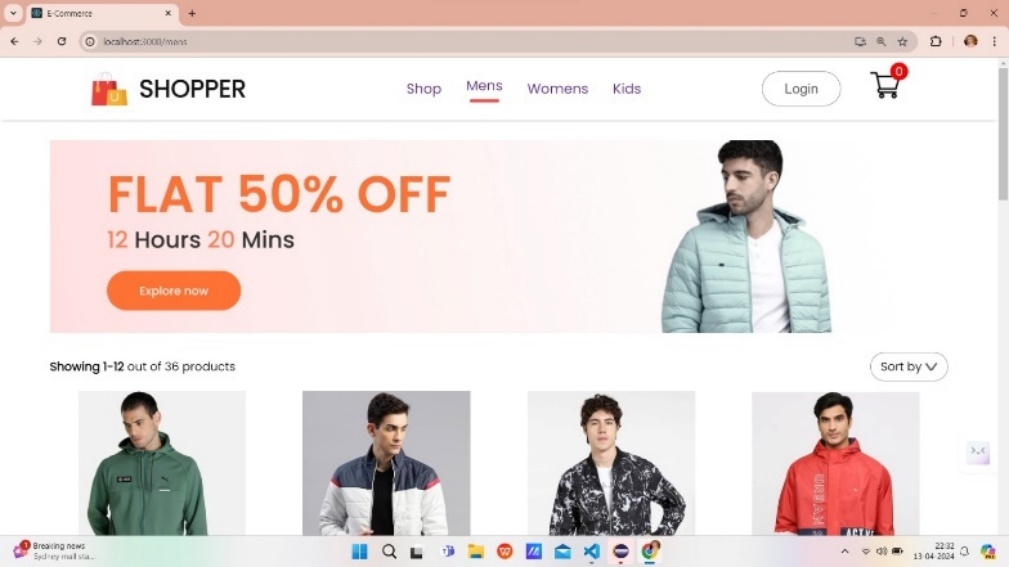
****

Figure 9.3 Refreshing a portion of the page in React



## Component Based Architecture

Component-based architecture is a software design approach where the system is built by breaking it down into self-contained, reusable modules called components. Each component encapsulates a specific piece of functionality and can be independently developed, tested, and maintained. Components communicate with each other through well-defined interfaces, allowing for modular and flexible system design. This approach promotes code reusability, scalability, and maintainability, as components can be easily composed and combined to create complex systems. Component-based architecture is widely used in modern software development, including web development frameworks like React and Angular, where UIs are composed of reusable UI components.

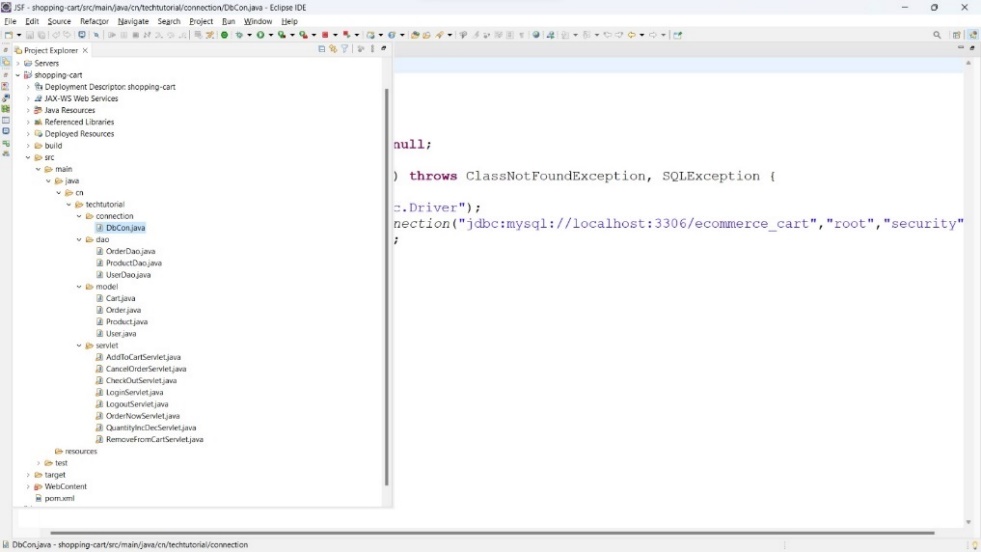
****

Figure 9.4 Component Based Architecture of JSP

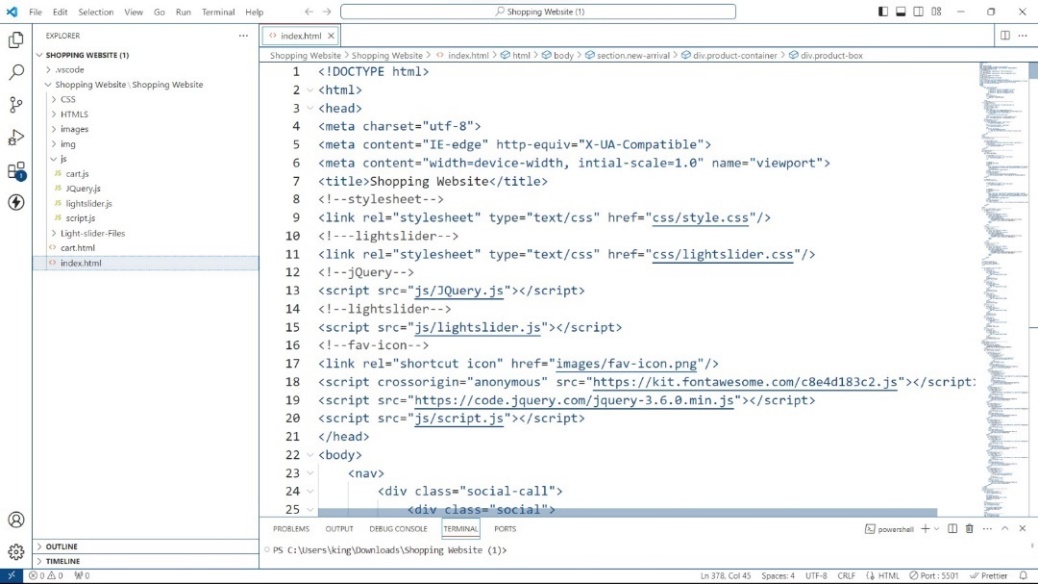
****

Figure 9.5 Component Based Architecture of JQuery

**A computer screen shot of a program

Description automatically generated**

Figure 9.6 Component-Based Architecture of React



## Code Reusability and Easy Debugging

Code reusability and easy debugging are two key benefits of React development. With React's component-based architecture, developers can create modular and reusable UI components that encapsulate their own state and behavior. This promotes code reusability across the application, as components can be easily composed and reused in different parts of the UI, leading to more maintainable and scalable codebases. Additionally, React's declarative and unidirectional data flow makes it easier to reason about the application's state and behavior, simplifying the debugging process. Developers can leverage tools like React DevTools and browser developer tools to inspect component hierarchies, track state changes, and debug potential issues more efficiently. Overall, React's emphasis on component reusability and clear separation of concerns enables developers to build robust and easily maintainable applications while streamlining to the debugging process.

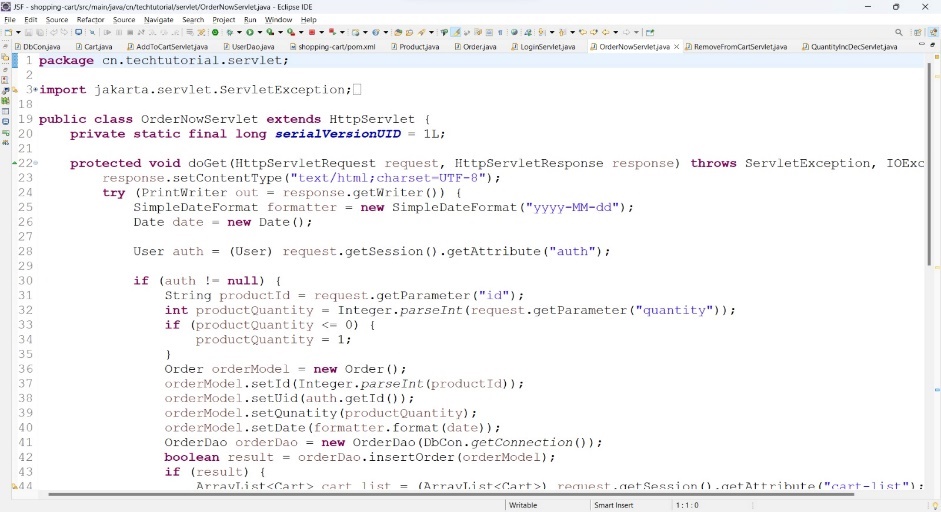


Figure 9.7 Code Optimality in JSP

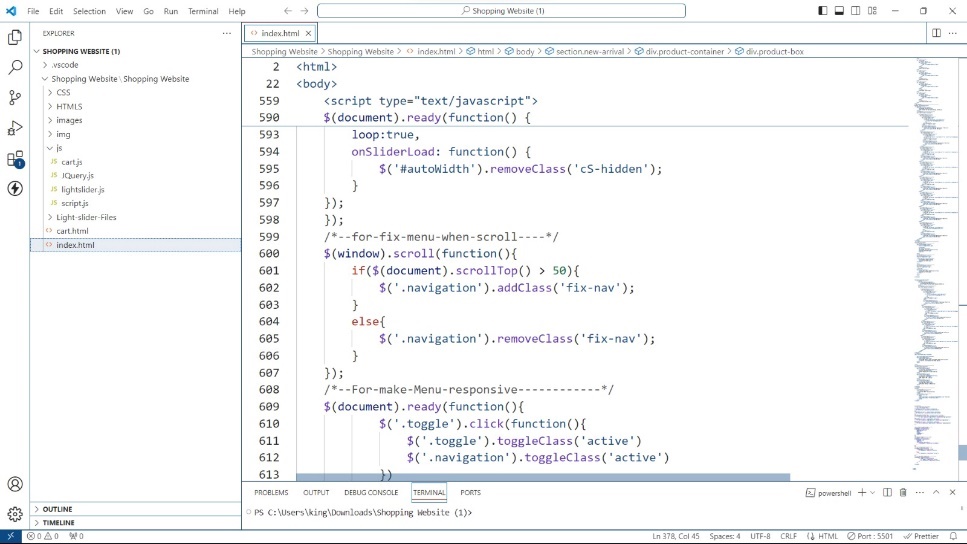


Figure 9.8 Code Optimality in JQuery

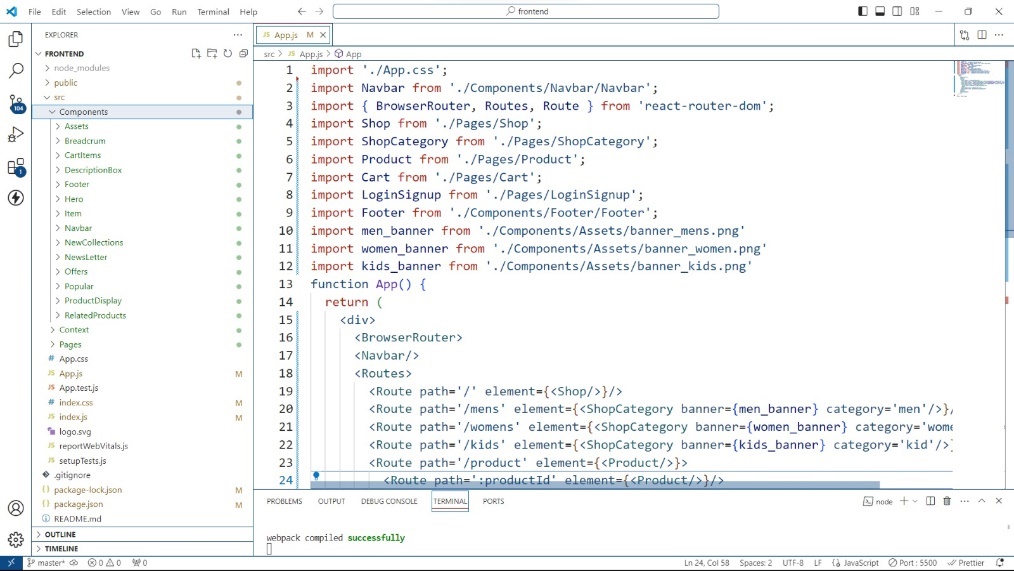


Figure 9.9 Code Optimality in React



## Single Page Application

A single-page application (SPA) is a web application that operates within a single web page, dynamically updating its content in response to user interactions without requiring the entire page to reload from the server. SPAs achieve this by loading the necessary HTML, CSS, and JavaScript resources when the application is initially loaded, and subsequently fetching data from the server via APIs. This approach provides a smoother and more responsive user experience compared to traditional multi-page applications, as navigation between different views or pages is handled client-side without the need for full-page reloads. SPAs are commonly built using modern JavaScript frameworks like React, Angular, or Vue.js, which facilitate the development of complex user interfaces and interactive web applications.

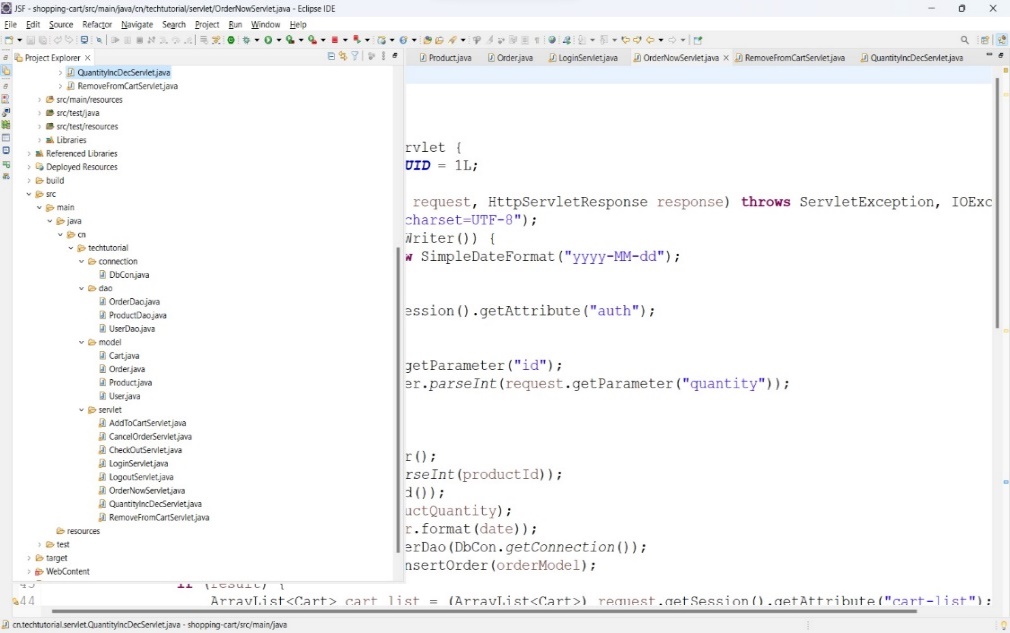
****

Figure 9.10 Single Page Application of JSP

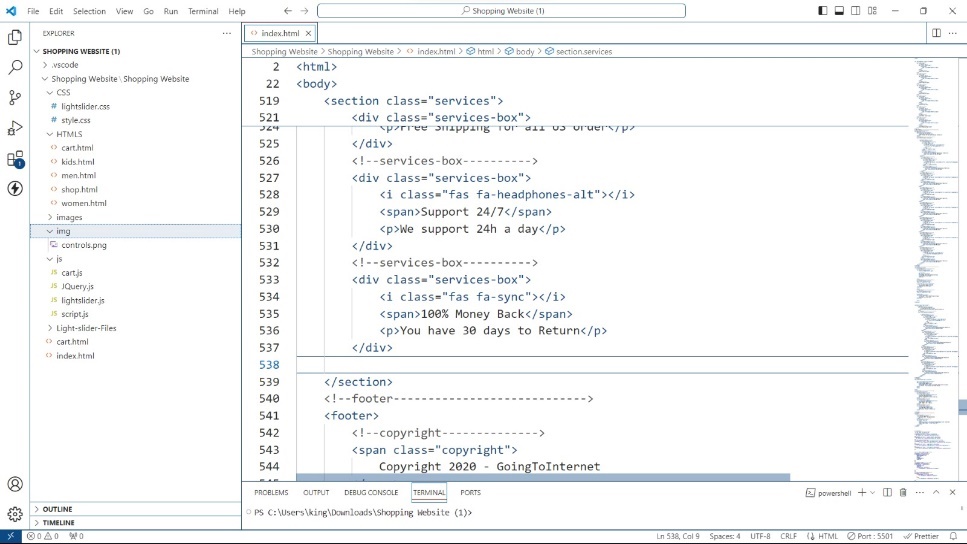
****

Figure 9.11 Single Page Application of JQuery

**A computer screen shot of a computer

Description automatically generated**

Figure 9.12 Single Page Application of React

## Results Page Responsiveness

Page responsiveness refers to a website's ability to adapt and display content appropriately across various devices and screen sizes, ensuring an optimal user experience. Responsive web design techniques, such as flexible grid layouts, media queries, and fluid images, enable web pages to dynamically adjust their layout and content presentation based on the user's device, whether it's a desktop computer, tablet, or smartphone. By prioritizing usability and readability on all devices, responsive pages enhance accessibility and engagement, ultimately leading to higher user satisfaction and retention.

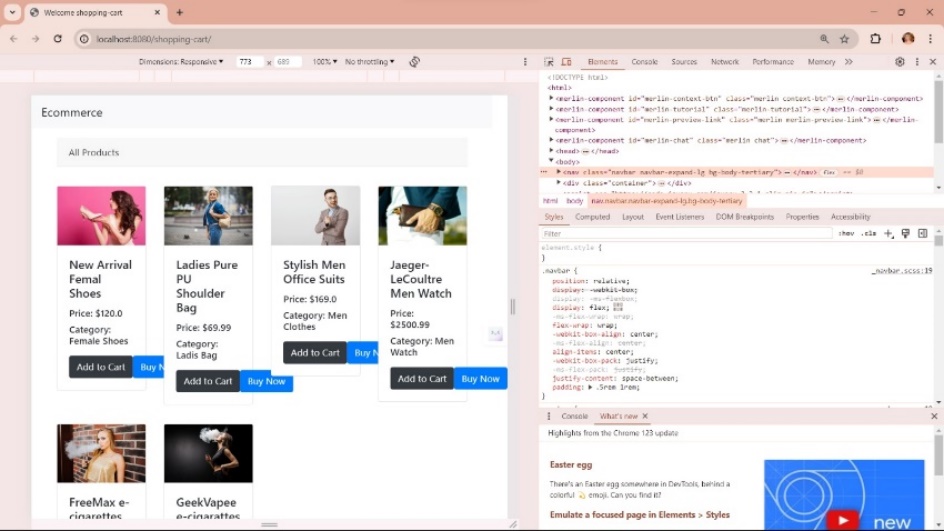


Figure 9.13 Page Responsiveness in JSP

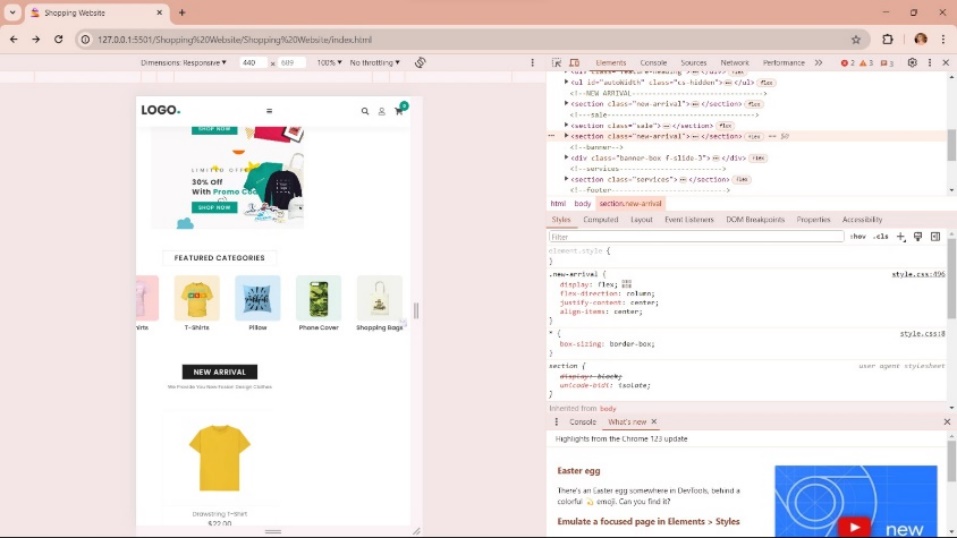


Figure 9.14 Page Responsiveness in JQuery

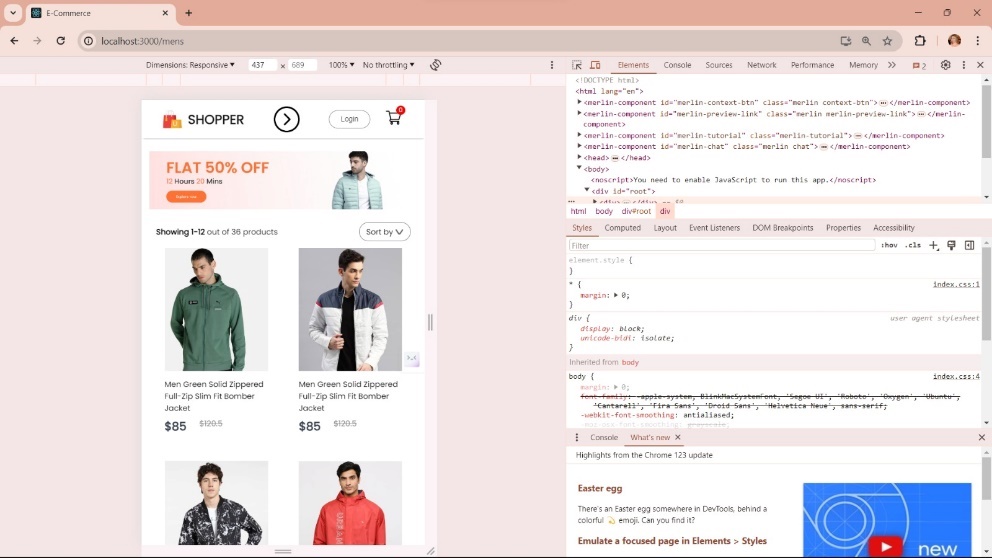
****

Figure 9.15 Page Responsiveness in React

# Conclusion and Future Enhancement

React.js has emerged as a cornerstone in modern web development, offering developers a robust framework for crafting dynamic and responsive user interfaces. Its component-based architecture streamlines development by breaking down complex UIs into reusable and modular components, fostering code organization and collaboration. Powered by the virtual DOM, React.js optimizes rendering efficiency, resulting in faster updates and smoother user interactions. This performance-centric approach, coupled with its robust state management capabilities, enables developers to create high-performing applications that deliver exceptional user experiences.

Furthermore, React.js benefits from a thriving community of developers and contributors, driving innovation and sharing best practices within the ecosystem. This collaborative environment not only provides developers with access to a wealth of resources and support but also facilitates continuous learning and growth. By actively engaging with the React.js community and staying abreast of emerging trends and advancements, developers can ensure their applications remain competitive, scalable, and aligned with industry standards.

# Bibliography

|  |  |
| --- | --- |
| [1] | G. m. B. C. K. Siddegowda C J, "React framework (creating a web application with react native)," *International Journal of Recent Trends in Engineering & Research (IJRTER),* 2018. |
| [2] | S. G. P. S. Archana Bhalla, "Present day web-development using reactjs," vol. 07, no. 05, 2020. |
| [3] | R. C. a. A. Sandeep, "React.js and front-end development," *IRJET,* vol. 07, no. 04. |
| [4] | M. Biswal, "React lifecycle methods," *Researchgate Article,* 2019. |
| [5] | N. I. Naim, "ReactJS: An Open-Source JavaScript Library for Front-end Development," *Teixeira, P., Professional Node. js: Building JavaScript-based scalable software,* 2012. |
| [6] | "State of JS 2020: JavaScript Flavors," May 2021. [Online]. Available: https://2020.stateofjs.com/en US/technologies/javascript-flavors/ . |
| [7] | V. Subramanian, "Modularization and Webpack," *Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node.* |
| [8] | M. Levlin, "DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte," 2020. |
| [9] | S. G. P. S. Archana Bhalla, "Present day web-development using reactjs," vol. 7, no. 5, 2020. |
| [10] | N. I. Naim, "ReactJS: An Open-Source JavaScript Library for Front-end Development". |
| [11] | K. G. D. G. Vipul Kaushik, "React Native Application Development," 2018. |
| [12] | G. C, "Introducing Flux: An Application Architecture for React.In: Introduction to React. Apress, Berkeley," *CA.* |
| [13] | B. Eisenman, "Learning React Native: Building Native Mobile Apps with JavaScript," *O'Reilly Media, Inc.,* 2015. |
| [14] | M. V. M. a. T. R. Ramos, "AngularJS performance ( A survey study )," *IEEE Software,* vol. 35, no. 2, pp. 72-79, 2018. |
| [15] | Z. Chen, " HTML5 Hybrid Mobile Application: Building mobile applications using web technologies with Ionic," 2018. |