Chapter 2: Literature Review

2.1. Web application

First, we'll go over the definition of a Web application:

A web application is a software application that runs on a web server and can be accessed via the internet using a web browser. With the benefit of being accessible from any location with an internet connection, it is made to offer functionality and a user interface that are comparable to those of a desktop application.

Web applications can be accessed through any web browser-enabled device, including desktop computers, laptop computers, tablets, and smartphones. They are frequently built to be highly scalable, allowing them to easily handle large numbers of users and data. Aside from that, we can build with a variety of programming languages and frameworks, and they typically rely on server-side technologies like PHP, Java, Ruby, or Python to manage backend logic and database interactions.

As a result, we can see that web applications are widely used in today's society, particularly during the 4.0 period. It enables businesses, organizations, and individuals to use the internet's power to deliver services and information to everyone quickly, efficiently, and easily. In this section, I will discuss some of the factors that distinguish web applications:

* Digital Transformation: With the introduction of 4.0 technology, an increasing number of businesses are looking to digitize their operations and processes in order to improve efficiency and competitiveness. Web applications can assist them in making this transition possible by providing tools and services that streamline workflows and improve communication between teams and stakeholders.
* Remote Work: Because of the COVID-19 pandemic, which has fueled the trend of remote working, web applications have become critical in enabling remote communication and collaboration. Employees can access the information they require using web applications from any location with an internet connection, making it easier to work from home or other remote locations.
* Scalability: Web applications can be easily extended to meet the growing needs of users. Cloud-based infrastructure enables businesses to rapidly develop new versions of web applications as needed, ensuring that applications can handle increasing traffic and usage with no downtime or performance issues.
* Accessibility: Web applications can be accessed from any location with an internet connection, allowing us to expand our user base. This is especially true for companies that serve global markets or have customers in remote locations.

Overall, web apps are an important part of today's digital 4.0 landscape. It provides businesses and organizations with the tools they need to remain competitive and agile in an ever-changing business environment. After discussing the current importance of web applications, we will look at how they work:



*Figure 2: The flow diagram depicts how a Web application operates.*

* To begin, the user will launch a web browser such as Chrome, Microsoft Edge, or Safari and enter the URL or click a link to the web application.
* The browser sends the request to the web hosting server.
* The request is received by the web server and forwarded to the appropriate application server.
* The application server processes the request and returns a response.
* The response is returned to the web server, which in turn returns it to the browser.
* The response is received by the browser, which then displays the web page to the user.

We now have a better understanding of the web application's workings as well as its significance. As a result, the Motorcycle Rental Management System project will create such a system to assist customers in managing the process and motorbike rental invoices. I will also create a source that is divided into two parts: the front-end, which receives requests from the browser, and the back-end, which processes the requests and sends responses to the other side.

2.2. Interaction Design

Interaction Design is the design of the interaction between the user and the product. Typically, when people discuss interaction design, they are referring to software products such as apps or websites.

Interaction Design is a critical component of developing a web application, so we must thoroughly understand it. It is the process of designing interactive digital products such as websites, mobile applications, and software interfaces with the goal of creating engaging and user-friendly experiences. Understanding user needs and behaviors, designing interfaces and interactions to meet those needs, testing and refining the design based on user feedback are all part of the process.

Interaction designers create the look, feel, and functionality of digital products, which includes visual design, information architecture, navigation, and user flow. To create effective and engaging user experiences, they employ a variety of design tools and methods such as wireframe, prototyping, user testing, and design thinking.

The goal of interaction design is to create products that are simple to use, intuitive, and enjoyable for users while also achieving business objectives such as increasing user engagement and improving user retention and increase conversions.

Following that, I will discuss five dimensions of Interaction Design. They are more than just a useful concept to keep in mind when developing a digital product. We'll look at these parameters to get a full picture of how users interact with digital products and what interaction design entails.



*Figure 3: Five dimensions of Interaction Design*

1. Word: The meaning of words is represented by this dimension. Words are extremely powerful in any field. They can assist users in quickly grasping the meaning and influencing them through word expressions. As a result, words must be familiar and easy to understand, communicated in a tone appropriate to the context, and used consistently throughout the product to convey information to the end user. In order to prevent misunderstandings among users when they use our product, we must carefully select vocabulary and proofread the language.
2. Visual representations: This dimension discusses visual elements that the user interacts with, such as typography, diagrams, symbols, or images. They frequently supplement the words used to convey information to the user. These elements are just as effective as "Words" because they quickly convey meaning to the user. In certain situations, this improves the user experience. However, we must exercise control over the use of these elements. When using our software, avoid overloading to avoid confusing users.
3. Physical objects or space: This dimension refers to the physical objects with which users interact while using the product. For example, using a mouse or touchpad on a laptop, a finger on a smartphone, and so on. Apart from physical objects, this dimension also refers to a type of physical space in which users interact with software. For example, users can use the laptop at work or at home. All of these factors have an impact on how people interact with products. As a result, some electronic device developers will attempt to optimize all interactions in all conditions and spaces that users may encounter.
4. Time: This is a unique factor because it is related to the amount of time the user spends interacting with the previous three dimensions. It entails creating designs for specific situations, such as the first interaction, repeated use, or when the user encounters an error. Furthermore, it is regarded as a criterion by which users can measure and evaluate their progress. This is a broad but critical aspect of implementing interaction design. Nobody wants a design that is responsive or takes a long time to manipulate. In today's world, time is synonymous with money, so optimizing and saving time is a top priority for businesses or corporations looking to improve their products.
5. Behavior: This final dimension consists of actions, reactions, activities, and presentations that are adaptable and understandable to all users. They frequently include questions such as the following: How do users react to product actions? How do customers interact with the product? Developers must learn and design their own products based on societal trends to ensure that users, both new and old, can easily use their applications.

We now fully comprehend all interaction design concepts. The dimensions will be used in my Motorcycle Rental Management System project. With the first dimension, I will prioritize the use of basic English words so that users can understand the function and meaning of the website when they visit it. With the Visual representations element, I will consult some websites that provide bike rental services in order to select the appropriate fonts, colors, images, and charts for this project. Regarding the third factor, which is an objective factor, I can't completely control the user's space, so I built an application that can run on both laptops and phones to diversify users' needs so that they can use the product everywhere. In the fourth dimension, time, I will attempt to optimize the interaction time with the website, ensuring that all functions respond quickly enough to satisfy the basic user. Finally, with dimension behavior, I will listen to the user's request in conjunction with some other products to build appropriate and reasonable interactions for each function, ensuring user satisfaction.

2.4. MVC model

Knowing the MVC pattern is one of the basic elements of a web application product in the project. Model-View-Controller, or MVC, is a software development design pattern that divides an application's concerns into three interdependent parts: the model, the view, and the controller.

Diagram

Description automatically generated

*Figure 5: The MVC model*

* The Model represents the application's data and business logic. It is in charge of data retrieval, manipulation, and storage.
* The View represents the application's user interface. It is in charge of displaying data to the user and allowing the user to interact with the application.
* The Controller acts as a go-between for the Model and the View. It receives user input through the View, processes the user's requests, and updates the Model as needed. It also updates the View with any changes made to the Model.

The MVC pattern allows programmers to create code that is simpler to maintain, test, and modify by dividing an application's concerns into these three parts. It also contributes to the overall structure of an application and encourages the use of best practices. Upon learning about the MVC pattern, I believe it is quite appropriate and provides numerous benefits when used in web applications. Here are a few reasons why the MVC model is essential for web applications:

* Separation of Concerns: The MVC pattern divides the concerns of the application into three distinct components, each with their own set of responsibilities. This separation allows developers to work on different parts of the application without interfering with one another.
* Code Reusability: Separation of concerns makes code more reusable. Because the Model, View, and Controller are loosely coupled, changes to one component will not affect the others, making it easier to reuse code in other parts of the application.
* Better Code Organization: The MVC pattern encourages developers to write organized and modular code. It is easier to maintain, test, and modify code when the application's concerns are separated.
* Simpler Testing: MVC makes it easier to test individual application components. Separating the Model from the View and Controller during testing can make the process easier and more efficient.
* Improved User Experience: By dividing the application's logic and presentation, the MVC pattern contributes to an improved user experience. Developers can focus on making the user interface more user-friendly and responsive by keeping the application's business logic separate from its user interface.

Overall, the MVC design can assist developers in creating more robust, maintainable, and scalable web apps. At the Motorcycle Rental Management System project, I will use the MVC model with View as the Front-end source code (ReactJS) used to send requests to or receive responses from the Controller in the Back-end source. In addition, I have logically structured the Back-end source, including folders "Controller", "Entity", "Model", "Service", and "Specification" in accordance with the standard Spring Boot project structure. This structure follows the MVC pattern.

Chapter 3: Technology and Tools

3.2. ReactJS

**3.2.1. What is ReactJS?**

ReactJS (also known as React) is a well-known open-source JavaScript library for developing user interfaces (UI) for web applications. It was created by Facebook in 2011 and later released as an open-source project in 2013.

React enables developers to produce reusable UI components that can be used to quickly and easily build complex web applications. It employs a declarative syntax that allows developers to describe the desired UI state, with React handling DOM (Document Object Model) updates as needed. This method aids in improving the performance and efficiency of web applications.

React can be used with other libraries or frameworks, and it is frequently used in conjunction with other technologies such as Redux, GraphQL, and Next.js. React has a large user community, so I can easily find help to solve problems that arise during use. Furthermore, it supports a large number of libraries and frameworks that support the development of system interfaces.

Because of its performance, scalability, and ease of use, React has become one of the most popular JavaScript libraries for developing web applications. Following, I will list the advantages of this open-source:

* Improved performance: ReactJS employs a virtual DOM, which aids in the performance of web applications by reducing the number of direct manipulations of the actual DOM. This approach reduces unnecessary re-renders, improving the application's overall performance and speed.
* Reusable components: Developers can use ReactJS to create reusable components that can be used throughout the application. This reduces the amount of code that must be written, resulting in faster and more efficient development.
* Declarative syntax: ReactJS employs declarative syntax, which allows developers to more easily describe the desired UI state. This method allows developers to concentrate on what they want the application to do rather than how it should be done.
* Simple to learn: ReactJS has a simple API, making it simple for developers to learn and use. It also has a sizable and active user base that offers extensive documentation, tutorials, and support.
* SEO-friendly: ReactJS can be used to create search engine friendly web applications. Because React renders HTML on the server, search engines can easily crawl and index the content, resulting in higher rankings.
* Integration with other libraries: ReactJS integrates easily with other libraries and frameworks such as Redux, GraphQL, and Next.js, giving developers a flexible and extensible toolkit.

In general, ReactJS is a strong and flexible library that gives developers the resources they need to create scalable and fast web applications.

**3.2.2. Introduce about JSX**

One of the new skills acquired while working with ReactJS is JSX. In this section, I will introduce this concept. JSX (JavaScript XML) is a syntax extension for JavaScript that is used with ReactJS. It makes it simpler to create and work with the UI components of a React application by allowing developers to write JavaScript code that resembles HTML. The React compiler converts JSX code into regular JavaScript code that the browser can interpret.

Although JSX syntax is similar to HTML syntax, it is not the same. Instead, it's a syntax extension that lets developers write JavaScript code that generates React elements. However, it still enables us to use a familiar HTML-like syntax while utilizing JavaScript and ReactJS to their full potential.

I'll explain why it's preferable to use JSX instead of HTML in a ReactJS source code in this section. In fact, a React application can be created without the use of JSX. Developers can create React components in either pure JavaScript or plain HTML templates. However, using JSX will provide some of the following advantages:

* Improved developer experience: JSX enables developers to create components using familiar HTML-like syntax. This makes writing and reading code easier, particularly for those who are already familiar with HTML.
* Full JavaScript power: JSX enables developers to use the full power of JavaScript to create dynamic and interactive components. This includes constructing intricate UI components using loops, conditional statements, and other JavaScript features.
* Performance: To improve performance in React applications, JSX can be compiled. React can effectively update only the portions of the user interface (UI) that have changed rather than having to re-render the entire page by using a virtual DOM.
* Better React integration: JSX is designed to work in tandem with React components. It makes it simple to pass props and state between components and enables developers to easily create reusable components. Code is easier to write and read, especially for those who are already familiar with HTML.

To summarize, while it is possible to create React components using plain HTML or pure JavaScript, JSX provides several advantages that can make it easier and more efficient to create high-quality, performant applications.

**3.2.3. Virtual DOM**

The Virtual DOM (Document Object Model) is ReactJS's in-memory representation of the actual DOM. It is a lightweight and efficient copy of the actual DOM that contains all of the same properties and methods as the real DOM.

The Virtual DOM was developed as a technique for improving performance. When we make changes to the UI in a React app, the Virtual DOM is updated rather than the actual DOM. React then compares the updated Virtual DOM to the previous version to determine the specific changes that have been made.

After identifying the changes, React updates the actual DOM only where necessary, reducing the amount of work required to re-render the UI. This approach can result in significant performance improvements, especially for complex or large applications.

The Virtual DOM also enables developers to write declarative React components. Instead of writing complex code to manipulate the DOM directly, developers can simply describe the desired state of the UI and let React handle the updates. From there, it will assist us to improve the user interface's response time.

In conclusion, a crucial component of ReactJS, the Virtual DOM enables better performance and a more declarative approach to UI development. React can update only the necessary parts of the UI by using a lightweight copy of the actual DOM, resulting in faster and more responsive web applications.

**3.2.4. Introduce about Component**

A component in ReactJS is a reusable building block that encapsulates the user interface and its behavior. It is a component of a website or user interface that can be reused across multiple pages or applications. A component may include HTML, CSS, and JavaScript and may receive and pass data to other components.

Components are the fundamental building blocks of ReactJS applications and are an essential component of the ReactJS framework. The code is kept organized and understandable with the aid of components. They make it possible for us to create complex user interfaces by breaking them down into smaller, more manageable chunks.

Furthermore, components can be reused across multiple projects or pages, reducing the amount of code required and increasing the codebase's maintainability. We can save time and reduce the risk of introducing bugs or inconsistencies in the code by creating reusable components.

In conclusion, components are an important part of ReactJS projects because they help to organize, optimize, and maintain the code. They allow me to create complex user interfaces by breaking them down into smaller, reusable building blocks.

**3.2.5. Introduce about Props and State**

**3.2.5.1 What is Props in ReactJS project?**

Props (short for "properties") are a data transfer mechanism in ReactJS that allows data to be passed from a parent component to a child component. Props are read-only and the child component is unable to change them.

Props are essential in ReactJS projects because they enable developers to create reusable and modular components. A component can be customized to meet the needs of the parent component by passing data down through props. Because each component can be made to handle a particular set of data, it is simpler to design complex user interfaces with numerous components.

Using props also allows us to keep data flow in their application unidirectional, making it easier to reason about the application's state. Data flows from parent components to child components via props, and a child component can update the data by invoking a function passed down as a prop from the parent component.

Props are an important part of ReactJS projects because they enable the creation of reusable, modular components that can be easily customized and combined. We can create complex user interfaces that are easy to reason about and maintain by passing data down through props.

**3.2.5.2. What is State in ReactJS project?**

In ReactJS, state is an object that stores the data that a component requires to render and manage its behavior. Unlike props, which are read-only and are passed down from parent components, state is managed internally by the component and can be modified by the component.

Because it enables components to manage user interactions and react to changes in the application state, state is important for ReactJS projects. A form component, for example, could use state to store the values of the input fields and update them as the user types. A button component may use state to keep track of whether it is currently pressed or not.

State is also important in ReactJS projects because it enables efficient user interface updates. ReactJS will automatically re-render a component when its state changes, updating only the portions of the user interface (UI) that have changed. This allows for quick and efficient updates without the need to re-render the entire UI. It will enable us to make significant progress in UI optimization.

In general, state is an important factor of ReactJS projects because it enables components to be interactive and dynamic, responding to user input and changes in the application state. Internal state management allows components to be more efficient and responsive, resulting in a better user experience.

**3.2.5.3. Difference between Props and State:**

Props and state are both important concepts in ReactJS, but they serve different functions. Therefore, we must be able to distinguish between the two in order to develop the application more effectively. They have some significant differences:

* Ownership: Props are owned by the parent component and passed down to child components, whereas state is owned and managed by the component itself.
* Mutability: Props are read-only and cannot be changed by the child component, whereas state can be altered by the component.
* Initialization: Props are passed into a component when it is created, whereas state is initialized within the component's constructor or in a lifecycle method.
* Usage: Props are used to pass data from parent to child components, whereas state is used to manage internal data and handle user interactions.
* Data scope: Props are passed down from a parent component to a child component, limiting their scope to the component hierarchy, whereas state is only defined in the component in which it is defined.
* Triggers for updates: Changes to props are triggered by changes in the parent component, whereas changes to state are triggered by user interactions or other internal events within the component.

Props and state are used in a variety of ways within a ReactJS application, each serving a distinct purpose. Understanding the distinctions between the two is critical for developing effective and efficient components.

**3.2.6. Introduce Lifecycle methods of ReactJS**

Lifecycle methods in ReactJS are special methods that are called at specific points in a component's lifecycle. These methods enable we to perform specific actions at various stages of a component's existence, such as when it is first mounted, updated, or unmounted from the DOM.

In ReactJS, lifecycle methods fall into three broad categories:

* Mounting methods: When a component is first created and added to the DOM, these methods are invoked. This category includes constructor(), static getDerivedStateFromProps(), render(), and componentDidMount() .
* Updating methods: These methods are called when the state or props of a component change and the component needs to be re-rendered. Static getDerivedStateFromProps(), shouldComponentUpdate(), render(), getSnapshotBeforeUpdate(), and componentDidUpdate() are examples of methods in this category.
* Unmounting methods: When a component is removed from the DOM, these methods are invoked. The componentWillUnmount() is the only method in this category.

In addition to these main categories, there are a few less commonly used lifecycle methods, such as static getDerivedStateFromError(), componentDidCatch(), and shouldComponentUpdate().

Lifecycle methods are important in ReactJS because they give we hooks to perform actions at specific points in the lifecycle of a component. ComponentDidMount(), for example, can be used to retrieve data from an API when a component is first mounted, whereas componentWillUnmount() can be used to clean up any resources used by the component before it is unmounted.

We can control the behavior of our components at various points in their lifecycle using lifecycle methods, making it easier to create efficient and responsive user interfaces.

**3.2.7. Introduce about Hooks**

Hooks are a new feature in ReactJS 16.8 that allows us to use state and other React features in functional components that were previously only available in class components.

Hooks are useful in ReactJS projects because they make it easier to build and manage state logic within components, making code more readable and easier to maintain. Hooks are useful for managing state, dealing with side effects, and integrating with other React features like context and refs.

As we've seen, Hooks is new, but it's already demonstrated its significance. In this section, I will discuss a few advantages of hooks for a ReactJS project.

* Improved code organization: Hooks enable developers to combine related stateful logic into a single function, making it easier to understand and maintain.
* Simpler component hierarchy: Hooks allow us to create complex functionality within a single component, eliminating the need for multiple layers of nested components.
* Reusable logic: Hooks make it simpler to build and maintain large-scale applications by allowing logic to be encapsulated and reused across multiple components.
* Simple to understand and apply: Hooks are easier to understand and use than traditional class components, requiring less boilerplate code.
* Improved performance: Developers can make more efficient and performant components by using hooks to manage state and side effects.

In conclusion, Hooks are an important feature in ReactJS because they simplify the process of building and managing stateful logic within functional components. Hooks can assist us in creating better user interfaces and improving the overall performance of their applications by making it easier to create reusable and efficient components.

**3.2.8. Introduce about Redux**

Redux is a library for managing states in JavaScript programs, such as ReactJS. It provides a centralized store for managing the state of an application and allows for predictable and efficient handling of state updates and data flow.

The application state is stored in a single "store" object in a Redux-based ReactJS project, which is then accessed as needed by individual components. Changes to the state are handled by "actions" that are sent to the store and then processed by "reducers" to update the state.

Redux is useful in ReactJS projects because it provides a scalable and predictable way to manage an application's state. Redux makes it easier to manage complex data flows and maintain a clear separation of concerns between different parts of the application by centralizing the state in a single store. It also provides a standardized method for handling state updates, making debugging and testing application behavior easier. Then, I will discuss some benefits of redux:

* Managed by a centralized state: Redux makes application state management and maintenance simpler by allowing for a single source of truth.
* Predictable state updates: Redux makes it simpler to debug and predict the behavior of applications by adhering to a rigid pattern for handling state updates.
* Improved scalability: Redux makes complex data flows and state updates easier to manage, making it easier to build and maintain large-scale applications.
* Developer tools: Redux includes a number of developer tools and extensions to aid in debugging, testing, and maintaining the application.
* Interoperability: Redux can be combined with a variety of other frameworks and libraries, which makes it simpler to integrate with pre-existing codebases or create modular applications.

As we can see, Redux is an important tool for ReactJS projects because it can help improve the application's scalability, maintainability, and predictability. While it may add some complexity to the development process, it has the potential to make it easier to build high-quality, complex applications with a clear separation of concerns and predictable behavior.

In the Motorcycle Rental Management System project, I used redux for some intermediate variables that run in real-time, such as saving tokens for authorization and loading menus based on whether or not the user is logged in.

**3.2.9. Introduce about React Router**

One of the final terms I'd like to present in the ReactJS section is React Router. ReactJS programmers will be very familiar with this concept. We'll first discover what a React Router is.

React Router is a well-known ReactJS library that enables declarative routing in single-page apps. It enables programmers to specify routes for various application sites or components, and it manages navigation and rendering of these components based on the URL that is currently being used.

Because it offers a neat and reliable method to handle navigation within a single-page application, React Router is crucial for ReactJS projects. React Router makes it simpler to create complex applications with numerous views and user flows by defining routes and mapping them to particular components. Additionally, it makes handling browser history and URL parameters simpler, enabling more reliable and user-friendly navigation.

I'll then demonstrate how React Routing is useful.

* Route declaration: Developers can specify routes and navigation using React Router in a declarative manner, which makes it simpler to reason about application behavior.
* Component-based routing:  A modular and reusable strategy to developing complicated applications is made possible by React Router's ability to map routes to particular components.
* Changing routing: Applications with changing data and user flows can be built more easily thanks to React Router's ability to manage dynamic paths and URL parameters.
* Browser history management: React Router includes built-in management of browser history support, enabling seamless navigation and past tracking.
* Integration with other React features: React Router works in tandem with other React features such as context and hooks to provide a unified and uniform development experience.

React Router is an essential library for ReactJS projects because it enables declarative, component-based routing in single-page apps. React Router can help us create more robust and user-friendly applications by offering a predictable and modular approach to navigation and user flows. In this project, I use the primary React Router to render the pages defined in the App.js file.

3.5. Firebase

This part will go over a specialized tool for storing images pertaining to cloud computing. Firebase is the name of this application. It is an essential tool in the Motorcycle Rental Management System project.

**3.5.1. What is Firebase?**

Firebase is a Google-owned mobile and web application development tool that offers a variety of backend services such as authentication, real-time databases, hosting, cloud storage, and messaging. It helps us to create high-quality mobile and web apps rapidly, with minimal setup and maintenance.

Firebase provides a real-time database that can be used to store and synchronize data in real-time between numerous clients. It employs NoSQL technology, which makes it easier to store and retrieve data in a flexible way without requiring developers to build complex database schemas.

Furthermore, Firebase offers hosting services, making it simple to deploy web applications rapidly and securely. It also provides cloud storage for storing and serving information, as well as messaging services for sending tailored messages to users.

In the end, Firebase is a powerful platform for developing mobile and web apps that has grown in popularity among developers due to its ease of use and extensive feature set.

**3.5.2. Some benefits of Firebase**

* Scalability: Firebase provides cloud storage services that can scale automatically as our application's utilization grows. This means we won't have to think about managing our own servers or infrastructure to meet our application's storage requirements.
* Accessibility: Firebase enables us to view our stored images from any application or web browser. This can make managing our image data and sharing it with other users simpler.
* Security: Firebase includes built-in security features to safeguard our stored data, such as access controls and encryption. This can help to ensure that our image data is private and protected from unauthorized access.
* Integration: Firebase integrates easily with other Firebase services, such as authentication and real-time databases, making it easier to create a comprehensive application that utilizes multiple Firebase services.
* Cost: Firebase has a generous free tier that allows us to store and access a certain quantity of data each month for free. For smaller applications or initiatives with limited budgets, this can be a cost-effective solution.

**3.5.3. Configure Firebase in a ReactJS project.**

* Step 1 is creating a Firebase project: Go to the Firebase Console and start a new Firebase project.
* Step 2 is installing Firebase SDK: Install the Firebase SDK in our ReactJS app by running the following command in the terminal. For example, npm install firebase
* Step 3 is initializing Firebase: In our ReactJS project, start Firebase by creating a new Firebase configuration object with our Firebase project credentials.
* Step 4 is using Firebase services: We can use Firebase's services in our ReactJS components after we've started it.

Please keep in mind that based on the requirements of our application, we may need to configure Firebase authentication and other services separately. More information about setting and using Firebase services in ReactJS can be found in the Firebase documentation.