International Journal Software Engineering and Computer Science (IJSECS)
4 (2), 2024, 484-496
Published Online August 2024 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN : 2776-4869, E-ISSN : 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v4i2.2831.

RESEARCH ARTICLE Open Access

Developing Modern JavaScript Frameworks for Building Interactive Single-Page Applications

Deyidi Mokoginta *
Electrical Engineering Study Program, Faculty of Engineering, Universitas Teknologi Sulawesi Utara, Manado City, North Sulawesi Province, Indonesia.
Corresponding Email: deydi81@gmail.com.

Desfita Eka Putri
Informatics Management Study Program, Politeknik LP3I Pekanbaru, Pekanbaru City, Riau Province, Indonesia.
Email: marwahmasruroh@mesin.pnj.ac.id.

Fegie Yoanti Wattimena
Information Systems Study Program, Faculty of Science & Technology, Universitas Ottow Geissler Papua, Jayapura City, Papua Province, Indonesia.
Email: fegiywattimena.travel@gmail.com.

Received: April 7, 2024; Accepted: July 10, 2024; Published: August 1, 2024.

Abstract: The rapid evolution of modern JavaScript frameworks has significantly enhanced the development of interactive and responsive Single-Page Applications (SPAs). This study compares three prominent JavaScript frameworks—Angular, React, and Vue.Js—focusing on their application in SPA development. The research methodology involves a literature review, a comparative analysis of the critical features of each framework, and a practical implementation to assess their performance, usability, and flexibility. The findings reveal that each framework exhibits distinct advantages and limitations that affect their suitability for different project requirements. Angular is characterized by its extensive built-in features and robust architecture, React by its high flexibility and superior performance, and Vue.js by its user-friendly approach and development efficiency. These results offer valuable insights for developers in selecting the most appropriate framework for their specific project needs, ultimately contributing to more efficient and effective web development.

Keywords: JavaScript Frameworks; Single-Page Applications; Angular; React; Vue.js.

1. Introduction

In the current digital era, the rapid advancement of information and communication technology has significantly increased the demand for more interactive, responsive, and easily accessible web applications. Among the solutions that have gained widespread adoption to meet these demands are Single-Page Applications (SPAs). SPAs are a type of web application that allows for loading a single web page, where content is asynchronously updated without needing to reload the entire page. This capability offers a smoother and faster user experience, as interactions and navigation within the application can be performed more efficiently [1][2]. Modern JavaScript frameworks have become foundational in the development of SPAs. These frameworks provide a variety of features and tools that facilitate the structured and efficient creation of complex, feature-rich web applications. The most popular and widely used JavaScript frameworks today are Angular, React, and Vue.js. Each framework has unique characteristics, advantages, and drawbacks that influence how developers build and manage SPAs [3].

Angular, developed by Google, is recognized as a comprehensive framework equipped with numerous built-in features. Utilizing TypeScript, Angular offers a component-based architecture and a robust set of modules, making it an ideal choice for large-scale applications that require high levels of stability and scalability, such as enterprise applications and Software as a Service (SaaS) platforms [4][5]. Angular's structured approach and extensive feature set provide developers with the tools necessary to manage the complexities of large projects, although it is often perceived as having a steeper learning curve than other frameworks. On the other hand, React is a JavaScript library developed by Facebook that focuses on building user interfaces through reusable components. React excels in flexibility and high performance, allowing developers to integrate it with various libraries and additional tools to create solutions tailored to specific needs [6][7]. React's declarative approach to UI design and its robust ecosystem and strong community support have made it a popular choice for many projects, from simple web applications to mobile apps using React Native. Vue.js, created by Evan You, offers a balanced approach between ease of use and powerful functionality. Vue.js is known for its excellent documentation and intuitive approach, making it accessible to novice and experienced developers. It combines the best elements of Angular and React, providing a lighter and more flexible solution without sacrificing essential features [8][9]. Vue.js is often chosen for projects that require efficient development while maintaining a high level of functionality, mainly when ease of learning and implementation is a priority.

Given the importance of selecting the appropriate framework in SPA development, this research aims to conduct a comparative analysis of Angular, React, and Vue.js within the context of SPA development. The analysis will evaluate each framework regarding performance, ease of use, flexibility, and community support. Additionally, this study will include implementing a trial project to assess the practicality and effectiveness of each framework in real-world development scenarios. The outcomes of this research are expected to contribute significantly to the web development community, particularly in understanding the strengths and weaknesses of each JavaScript framework in SPA development. By gaining a deeper understanding of the characteristics and capabilities of each framework, developers can make more informed and strategic decisions when selecting the right tool for their projects.

Furthermore, this research serves as a valuable reference for novice developers just beginning to explore the world of SPA development, offering them clear guidance on choosing the appropriate JavaScript framework to start their journey. This study also seeks to enrich the existing literature on SPA development and JavaScript frameworks, offering practical insights that can be applied in real-world development situations. By doing so, we support the ongoing advancement of the web development industry, contributing to improving the quality and efficiency of web applications built using modern JavaScript frameworks. Ultimately, this research aspires to enhance the overall standards of SPA development, ensuring that developers are equipped with the knowledge and tools necessary to create high-quality, efficient, and user-friendly web applications.

2. Research Method

This research methodology employs a descriptive and experimental approach to analyze and compare the performance, ease of use, and flexibility of three prominent JavaScript frameworks, Angular, React, and Vue.js, within the Single-Page Application (SPA) development scope. The research method consists of several key stages: literature review, comparative analysis, experimental implementation, and data analysis. The initial stage of the research involves an extensive literature review to gather and evaluate existing information related to Angular, React, Vue.js, and the concept of SPA in general. The literature sources include academic books,

peer-reviewed journal articles, technical reports, official framework documentation, and credible online resources. The primary objective of this literature review is to establish a solid theoretical foundation regarding the characteristics, features, and architecture of the three frameworks under study.

Moreover, the literature review provides insights into the latest trends in SPA development and the challenges commonly encountered by developers. This stage ensures the research is grounded in current knowledge and practices, providing a basis for subsequent analysis [6][4]. Following the literature review, the research proceeds with a comparative study of Angular, React, and Vue.js. This analysis focuses on several critical aspects, including architectural design, performance metrics, usability, and community support. The architectural design analysis examines each framework's foundational structure and design patterns, including components, state management techniques, and modularity. These elements are crucial for understanding how each framework facilitates the development of complex web applications [10]. The performance analysis evaluates the frameworks' efficiency in rendering speed, initial load time, and memory management. Tools such as Lighthouse and WebPageTest quantify these performance indicators, providing a data-driven basis for comparison [7].

Usability is assessed by considering the ease with which developers can learn and use each framework. This includes evaluating the quality of documentation, the availability of teaching resources, and the intuitiveness of the API. These factors are essential in determining how quickly and effectively developers can adopt and utilize the frameworks in real-world projects [11]. Flexibility is analyzed by examining the extent to which each framework can be customized and integrated with other tools and libraries. This aspect is essential when specific project requirements require additional technologies or unique configurations [12]. The research methodology includes an experimental component, where a prototype SPA is developed using each of the three frameworks. This experimental implementation is designed to gather empirical data supporting the comparative analysis. The prototype application is selected to include standard SPA features such as user authentication, data management via APIs, and dynamic navigation. The development process involves several steps. First, the application's structure and user interface are designed, with specific features identified for implementation. Angular is used to build the first version of the application, focusing on component creation, service integration, and modular development. The process is then replicated using React, where components are created, the state is managed using Redux or Context API, and additional libraries are integrated as needed. Finally, Vue.js is employed to develop the third version of the application, emphasizing component creation, state management with Vuex, and the integration of necessary tools and libraries [13][14][15].

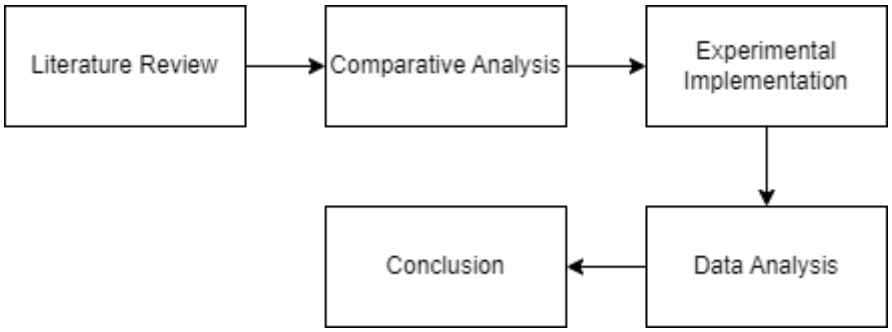

Figure 1. Research Stages

After developing the prototypes, the research analyzes the collected data. This analysis focuses on evaluating each framework's performance, usability, and flexibility based on the empirical data obtained from the experimental implementation. Performance metrics such as load time, rendering speed, and memory usage are measured using tools like Lighthouse and WebPageTest, providing quantitative data that will be compared across the frameworks. Usability is assessed through developer feedback gathered via surveys or interviews, which will provide qualitative insights into the ease of learning and using each framework and the effectiveness of available documentation and learning resources. Flexibility is evaluated by analyzing how well each framework can be adapted and integrated with other tools and libraries during development. Finally, the community support for each framework is investigated by examining the size and activity of the developer community, the availability of plugins and extensions, and the frequency of updates and bug fixes. This analysis provides an understanding of the long-term viability and support available for each framework, which is a crucial consideration for developers when choosing a framework for their projects.

3. Result and Discussion

3.1 Results

The application design phase aims to establish the core structure and key features of the Single-Page Application (SPA) before beginning the development process. During this phase, the first step is to identify the primary features to be implemented, such as user authentication (login/logout), data management via API, and dynamic navigation between the Home and Login pages. The component structure is also determined at this stage, where the application will consist of two main components: HomeComponent for the main page displayed after the user successfully logs in, and LoginComponent for the login page used for user authentication. Additionally, navigation routes are defined to direct users to the appropriate pages, with the route "/" designated for the Home page and "/login" for the Login page. The authentication logic is also designed to manage the user's login status, ensuring that only authenticated users can access specific pages.

The first step in development using Angular involves setting up the project. This is accomplished by installing Angular CLI with the command npm install -g @angular/cli, creating a new project using ng new angular-spa, and running the project with ng serve. The application can then be accessed via http://localhost:4200. After setting up the project, the next step is to add routing by configuring RouterModule. This involves adding routes for the Home and Login components in app-routing.module.ts. Subsequently, the Home and Login components are created using the commands ng generate component home and ng generate component login. To implement authentication, an AuthService is created to manage user authentication status with methods for login, logout, and checking if the user is logged in. Finally, navigation logic is added to the LoginComponent to redirect the user to the Home page after successful login.

Development with React begins by setting up the project using Create React App. This is done with the command npx create-react-app react-spa, navigating to the project directory with cd react-spa, and running the project with npm start. The application can be accessed via http://localhost:3000. Next, routing is added by installing React Router with npm install react-router-dom and configuring routing in App.js. The Home and Login components are created, with Home.js displaying a "Home Page" message and Login.js implementing login logic that redirects the user to the Home page upon successful login. For authentication, an authentication context is created in AuthContext.js, which provides methods for login, logout, and checking the user's authentication status. The application is then wrapped with AuthProvider to provide authentication context throughout the app.

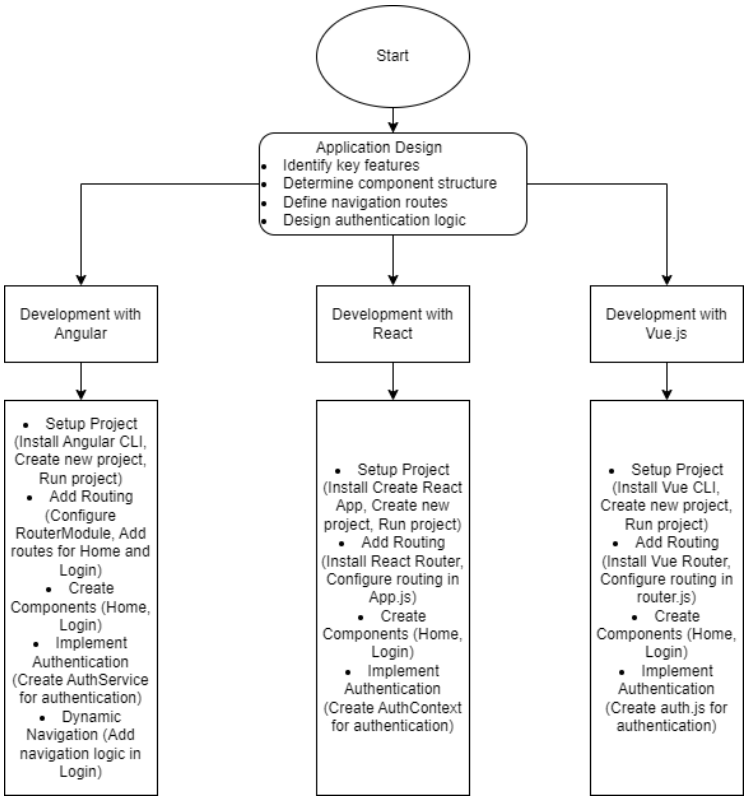

Figure 2. Application Design and Development Stages

Development with Vue.js starts with setting up the project using Vue CLI. This is done with the command npm install -g @vue/cli, creating a new project with vue create vue-spa, and running the project with cd vue-spa followed by npm run serve. The application can be accessed via http://localhost:8080. Routing is then added by installing Vue Router using npm install vue-router and configuring routing in src/router/index.js, adding routes for the Home and Login components. The Home.vue component is created to display a "Home Page" message, while Login.vue is created to implement login logic that redirects the user to the Home page after successful login. An authentication service is created in auth.js, which provides methods for login, logout, and checking the user's authentication status. Interactive SPAs can be developed using Angular, React, and Vue.js, each of which requires specific steps for project setup, routing configuration, component creation, and authentication implementation. Through careful design and systematic implementation, efficient and responsive SPAs can be developed, offering an optimal user experience. Each framework (Angular, React, and Vue.js) is capable of producing interactive SPA applications with similar features and appearance. The results of this development show how each framework can be used to build key components such as Home and Login pages, implement routing, and manage user authentication. The resulting application has dynamic and responsive navigation, providing an optimal user experience.

In developing a Single-Page Application (SPA) with Angular, the process begins by creating the main components, namely HomeComponent and LoginComponent. HomeComponent functions as the main page displayed after the user successfully logs in, while LoginComponent is the page for user authentication. In HomeComponent, the home.component.html file contains simple markup that displays a welcome message. On the other hand, login.component.html in LoginComponent provides a form for entering a username and password, as well as a login button. Authentication logic is implemented in login.component.ts, where the AuthService service is used to manage the login status. The login() method in LoginComponent checks whether the username and password have been entered, then redirects the user to the Home page after a successful login.

```
Home Component (home.component.html)
html
Copy code
<h1>Welcome to the Home Page</h1>
<p>This is the home page content.</p>
Login Component (login.component.html)
html
Copy code
<h1>Login</h1>
<form (ngSubmit)="login()">
  <label for="username">Username:</label>
  <input type="text" id="username" [(ngModel)]="username" name="username">

  <label for="password">Password:</label>
  <input type="password" id="password" [(ngModel)]="password" name="password">

  <button type="submit">Login</button>
</form>
Login Component (login.component.ts)
typescript
Copy code
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html'
})
export class LoginComponent {
  username: string;
  password: string;

  constructor(private authService: AuthService, private router: Router) {}

  login() {
    if (this.username && this.password) {
      this.authService.login();
      this.router.navigate(['/']);
    }
  }
}
```

Developing with React involves creating two main components: Home and Login. The Home component (Home.js) displays a welcome message to the user after they have successfully logged in. The Login component (Login.js) provides a form for entering a username and password, and uses an authentication context (AuthContext) to manage the login state. In Login.js, the local state is used to store user input, and the handleLogin method checks this input before calling the login function from the AuthContext and redirecting the user to the Home page. React Router is used to manage the navigation route between the Home and Login pages.

Home Component (Home.js)

```
jsx
Copy code
import React from 'react';

const Home = () => (
  <div>
    <h1>Welcome to the Home Page</h1>
    <p>This is the home page content.</p>
  </div>
);

export default Home;
```

Login Component (Login.js)

```
jsx
Copy code
import React, { useState, useContext } from 'react';
import { useHistory } from 'react-router-dom';
import { AuthContext } from './AuthContext';

const Login = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const { login } = useContext(AuthContext);
  let history = useHistory();

  const handleLogin = () => {
    if (username && password) {
      login();
      history.push('/');
    }
  };

  return (
    <div>
      <h1>Login</h1>
      <form onSubmit={(e) => { e.preventDefault(); handleLogin(); }}>
        <label htmlFor="username">Username:</label>
        <input      type="text"      id="username"      value={username}      onChange={(e)      =>
setUsername(e.target.value)} />

        <label htmlFor="password">Password:</label>
        <input      type="password"      id="password"      value={password}      onChange={(e)      =>
setPassword(e.target.value)} />

        <button type="submit">Login</button>
      </form>
    </div>
  );
};

export default Login;
```

In developing a SPA using Vue.js, two main components are created: Home and Login. The Home component (Home.vue) contains a template that displays a welcome message to the user after logging in. The Login component (Login.vue) includes a form for user authentication, where the user can enter a username and password. This input data is managed using Vue's reactivity with v-model. The login method in the Login component checks the user's input and, if valid, redirects the user to the Home page using this.$router.push('/'). The authentication service (auth.js) manages the login state, providing methods for logging in, logging out, and checking if the user is already logged in.

Home Component (Home.vue)

```
vue
Copy code
<template>
  <div>
    <h1>Welcome to the Home Page</h1>
    <p>This is the home page content.</p>
  </div>
</template>

<script>
export default {
  name: 'Home'
};
</script>
```

Login Component (Login.vue)

```
vue
Copy code
<template>
  <div>
    <h1>Login</h1>
    <form @submit.prevent="login">
      <label for="username">Username:</label>
      <input type="text" id="username" v-model="username">

      <label for="password">Password:</label>
      <input type="password" id="password" v-model="password">

      <button type="submit">Login</button>
    </form>
  </div>
</template>

<script>
export default {
  name: 'Login',
  data() {
    return {
      username: '',
      password: ''
    };
  },
  methods: {
    login() {
      if (this.username && this.password) {
        this.$router.push('/');
      }
    }
  }
};
</script>
```

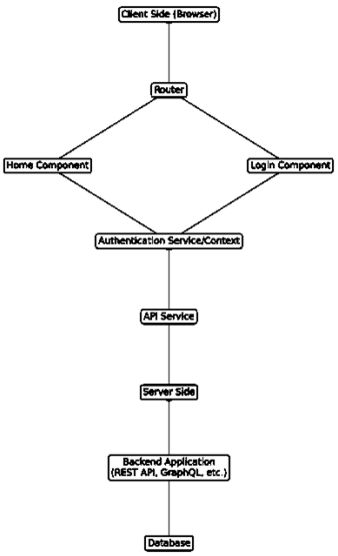

Figure 3. SPA Application Architecture Diagram

The Single-Page Application (SPA) application architecture consists of two main parts: the client side (browser) and the server side. On the client side, there are several important components that function to provide an interactive and responsive user experience. The first component is the Router, which is responsible for managing navigation between pages in the application without having to reload the entire page. The Home Component functions as the main page that is accessed after the user has successfully logged in, while the Login Component is the login page used for user authentication. Authentication Service/Context is a service that manages the user's authentication status. This service provides methods for logging in, logging out, and checking whether the user is logged in. In addition, the API Service is used to interact with the backend, allowing data retrieval and delivery without having to reload the page. On the server side, the architecture includes the Backend Application, which provides API endpoints (such as REST API or GraphQL) to interact with the client. This Backend Application is responsible for handling the application's business logic and communicating with the Database. The database stores all application data, including user information and other important data. The interaction between the client side and server side is done through the API Service, which sends data requests to the Backend Application and receives appropriate responses. The Authentication Service on the client side interacts with the API Service to verify user credentials during the login process and manages the authentication status. With this architecture, SPA applications can provide a fast and seamless user experience. Navigation between pages occurs dynamically without the need to reload the entire page, and data is efficiently fetched and sent through the API, ensuring that users get the latest information quickly. The Backend Application manages the business logic and ensures that data in the database is always consistent and secure. This architecture allows the development of interactive, responsive, and accessible web applications, better meeting the needs of modern users. The next stage is testing the performance, usability, and flexibility of three leading JavaScript frameworks—Angular, React, and Vue.js—in developing Single-Page Applications (SPA). Angular's performance testing shows a longer initial load time compared to React and Vue.js. This is due to the larger bundle size due to the many built-in features. However, Angular has a good rendering speed after the initial load is complete, thanks to the optimizations made by this framework. However, for applications that require fast initial loading, Angular may not be the best choice. React has faster initial loading times than Angular, especially when using techniques like code splitting and lazy loading. React's rendering speed is also very good, thanks to its efficient virtual DOM approach. React enables the development of high-performance applications, especially when it comes to fast and responsive user interactions. Vue.js performed very well in this test. Vue.js' initial loading time is almost on par with React, and its rendering speed is very competitive. Vue.js also supports various optimization techniques like lazy loading and code splitting, making it an excellent choice for applications that require high performance.

On the Ease-of-Use scale, Angular has a steeper learning curve due to its complexity and use of TypeScript. While Angular's documentation is very comprehensive and thorough, novice developers may find it difficult to grasp basic concepts such as dependency injection and modularity. However, once you get the hang of the basics, Angular provides a very robust and consistent structure for developing large applications. React is easier to learn than Angular, especially for developers who are already familiar with JavaScript. React's component-based approach and virtual DOM make it intuitive and flexible. React's documentation is excellent and there are many learning resources, tutorials, and an active community to support it. However, developers will need to understand additional ecosystems such as Redux for more complex state management. Vue.js is known to be the easiest framework to learn among the three frameworks. Vue.js' documentation is very clear and easy to follow, and provides many practical examples. The declarative and reactive approach used by Vue.js makes it intuitive for developers of all experience levels. Vue.js also offers easy integration with existing projects, making it an excellent choice for novice developers and small to medium projects.

In terms of Flexibility, Angular provides many built-in features such as routing, form handling, and state management. This makes Angular a great fit for large projects that require many out-of-the-box features. However, its flexibility is a bit limited because Angular has a very opinionated architecture and strict structure. React is very flexible and can be easily integrated with various additional libraries and tools. React's extensive ecosystem allows developers to choose and combine the tools that best suit their needs. This flexibility makes React a great fit for a wide range of projects, from simple to complex applications. Vue.js offers a balance between built-in features and flexibility. While not as comprehensive as Angular in terms of built-in features, Vue.js still provides essential tools such as Vue Router and Vuex. The flexibility of Vue.js allows developers to use various plugins and extensions as per the project's needs. Vue.js is also easy to integrate with existing projects, giving it an added advantage in terms of flexibility.

In terms of Community Support, the Angular community is very large and active, with many contributions from developers and companies. The documentation and learning resources are very comprehensive. However, due to its complexity, developers often need to rely on the community to solve specific problems.

React has a very large and active community, supported by Facebook and many large companies. There are many third-party libraries, tutorials, and resources available. An active community helps developers solve problems and share knowledge. Although the Vue.js community is smaller than Angular and React, it is very active and supportive. Vue.js documentation is excellent and there are many tutorials and examples to help developers. Vue.js developers often find strong support from the community, especially in open-source projects.

Based on the results of this study, each JavaScript framework has its own advantages and disadvantages in SPA development. Angular is suitable for large projects with complex needs and many built-in features. React is ideal for flexible UI development and integration with various libraries. Vue.js offers an easy-to-learn and easy-to-use solution, suitable for novice developers and projects that require fast and efficient development. The selection of the right framework should be based on the specific needs of the project to be developed. By understanding the characteristics and capabilities of each framework, developers can make more informed and strategic decisions in choosing the right tool for their projects. This study is expected to provide useful guidance for developers in determining the framework that best suits their needs.

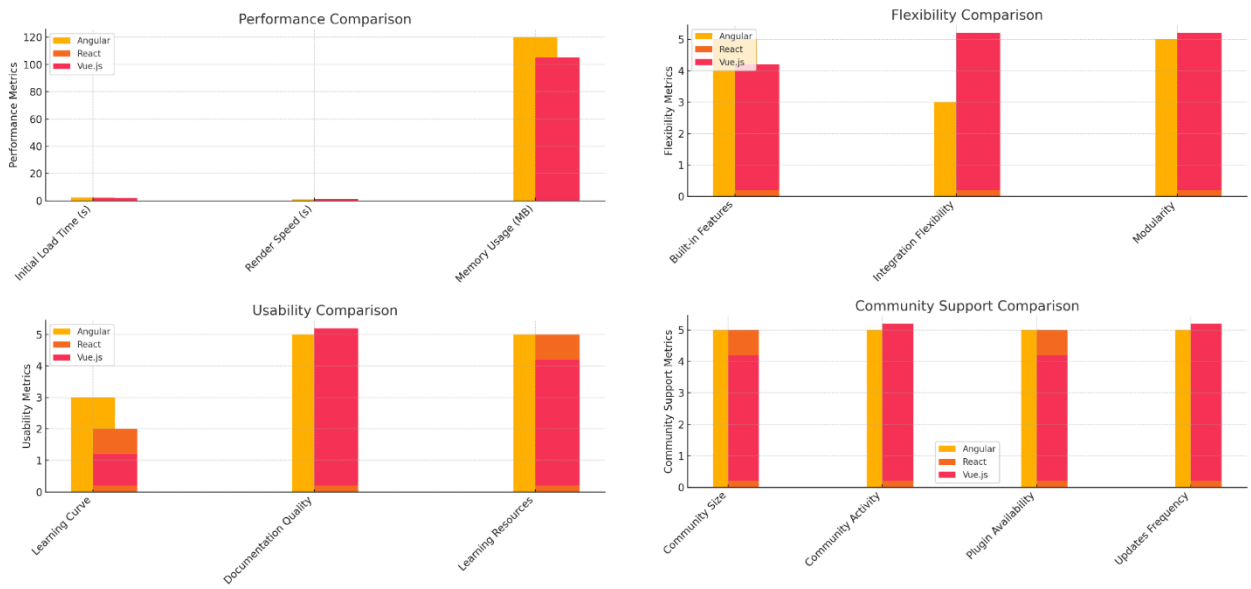

Figure 4. Comparison of JavaScript Frameworks

The comparison chart shows the key aspects of the three leading JavaScript frameworks—Angular, React, and Vue.js. In terms of performance, React has the fastest initial load time followed by Vue.js and Angular, while the rendering speed of React and Vue.js is superior to Angular. Angular uses more memory than the other two frameworks. In terms of ease of use, Vue.js has the lowest learning curve, making it easier to learn, while Angular has a steeper learning curve. Angular and Vue.js have excellent documentation, followed by React, and all frameworks have ample learning resources, with React slightly ahead. In terms of flexibility, Angular offers a lot of built-in features, but React is more flexible when it comes to integration with additional libraries. React and Vue.js are very flexible in integration, while Angular is slightly more limited, although all frameworks exhibit high modularity. Community support also plays a significant role; React has the largest and most active community, followed by Angular and Vue.js. Plugin availability is very high in React, followed by Angular and Vue.js, and all frameworks are updated frequently, with React slightly ahead in update frequency. By understanding this comparison, developers can make more informed decisions in choosing the framework that best suits their project needs.

3.2 Discussion

Based on the results of this study, each JavaScript framework has advantages and disadvantages that must be considered based on the project's specific needs to be developed. Angular is a framework that offers rich built-in features and a robust architecture, making it very suitable for large projects with complex needs. Angular uses TypeScript, which helps develop more structured and maintainable applications. Features like dependency injection, form handling, and built-in modularity make it ideal for large-scale and enterprise applications [16]. However, the steep learning curve and longer initial load time compared to other frameworks

can be a barrier for novice developers or projects that require fast initial load times [17]. Angular performs well in scenarios where long-term stability and scalability are top priorities [6]. React offers high flexibility and excellent performance, making it an ideal choice for various projects. With a component-based approach, React allows for developing reusable and more maintainable UIs. React's extensive ecosystem, with many additional libraries and tools such as Redux for state management, will enable developers to create solutions that fit their needs [11]. However, developers need to understand additional ecosystems to manage more complex states, which can add to the difficulty of using it [18]. React's high flexibility also means that developers have more freedom in choosing the right tools, but this also requires a deeper understanding of the various options available [13]. Vue.js stands out in terms of ease of use and competitive performance. Vue.js is known for its excellent documentation and intuitive approach, making learning accessible for developers of all experience levels to learn [19]. The flexibility offered by Vue.js allows for easy integration with existing projects, making it an excellent choice for projects with fast and efficient development [8]. Vue.js combines the best elements of Angular and React, providing a lighter solution without sacrificing essential features [20]. Vue.js is well-suited for small to medium-sized projects, where rapid development and ease of use are top priorities [9].

Community support plays a vital role in the success of a framework. Angular and React have large and active communities, providing developers with many resources, plugins, and support [21]. The official documentation and various online tutorials and articles help developers solve problems and improve their skills. Although the Vue.js community is smaller than Angular and React, its community activity and commitment remain strong, providing developers with the needed help and resources [4]. The selection of the proper framework should be based on an analysis of the project's needs, including scale, complexity, and the priority of performance and ease of use. Angular is well-suited for large and complex projects with a need for rich built-in features. With its excellent flexibility and performance, React is ideal for a wide range of projects requiring custom UI development. Vue.js, with its ease of use and flexibility, is well-suited for small to medium-sized projects requiring fast and efficient development [3]. By understanding the characteristics and capabilities of each framework, developers can make more informed and strategic decisions, ensuring the success of their SPA development projects. This study is expected to provide helpful guidance for developers in determining the framework that best suits their needs and driving innovation and efficiency in the web development industry.

4. Related Work

Research on modern JavaScript frameworks for Single-Page Applications (SPA) development has recently become a hot topic. Several previous studies have explored various aspects of Angular, React, and Vue.js, providing valuable insights for developers in choosing the proper framework. They found that React offers the fastest initial load time and high flexibility, while Angular provides many built-in features that facilitate the development of large applications. Vue.js was identified as an easy-to-learn and easy-to-use framework suitable for novice developers and small to medium-sized projects. Research on using JavaScript frameworks in web application development, especially Single-Page Applications (SPA), has proliferated and has become the attention of many academics and industry practitioners in recent years. Frameworks such as Angular, React, and Vue.js are the primary choice for developers due to their ability to build dynamic, interactive, and responsive applications. Each of these frameworks brings a different approach in terms of architecture, performance, and usability, thus influencing the choice of framework based on the project's specific needs. A study by Kaluža *et al*. (2018) highlights a comparison between three significant frameworks: Angular, React, and Vue.js. The study emphasizes that Vue.js shows higher efficiency in developing single-page applications (SPAs) and multi-page applications (MPAs) by combining the best aspects of Angular and React. Vue.js is known for its adaptability in various development scenarios, allowing developers to balance performance and code complexity [3]. Vyas (2022) continues the discourse with an in-depth comparative analysis of Vue.js, React, and Angular. The study identifies that each framework has unique characteristics that can be optimized based on project needs [6].

For example, with its rich built-in features, Angular is more suitable for large-scale projects, whereas React's modular approach is more flexible for acquisition and requires more specific UI development. Vue.js, on the other hand, Vue.js offers excellent documentation and ease of integration, making it a solid choice for projects with limited resources [6]. Diniz-Junior *et al*. (2022) examined the performance of JavaScript-based rendering technologies such as Angular, React, and Vue.js in the context of SPAs. In their case study, Vue.js demonstrated superiority in DOM manipulation speed compared to React and Angular. However, React

provided the best interactive time, making it ideal for applications that demand high interactivity [8]. Saini *et al*. (2023) reviewed the latest technologies in full-stack web development, emphasizing JavaScript integration across the software stack through frameworks such as Angular, React, and Vue.js. The study highlighted how such integration, primarily through Node.js, has transformed traditional approaches to web development, enabling more efficient and modular development [18].

In an exploration by Purohit *et al*. (2023), the role of Angular in modern web application development is explained in detail. With its complex yet stable architecture, Angular is highly effective for enterprise applications that require high scalability and modularity. This study highlights Angular's advantages in handling large and complex applications, where stability and consistency are essential [16]. Javeed (2019) focuses on performance optimization techniques for ReactJS, examining common challenges in application development, such as unnecessary component re-rendering and handling large datasets. This study provides practical insights into optimizing the performance of React applications in production, which is crucial for developers who want to maximize the efficiency of their applications [11]. A study by Baida *et al*. (2020) compares the performance of Angular and Vue.js in-game application development, which requires high efficiency in memory usage and rendering speed. Their findings suggest that Vue.js is superior in memory efficiency, while Angular offers better control over application structure and modularity [20].

Li & Zhang (2021) in their study reviewed how Vue.js is used in front-end development for SPAs. They highlighted the advantages of the MVVM (Model-View-ViewModel) architecture adopted by Vue.js, which supports more structured application development and facilitates future code maintenance [19]. Novac *et al*. (2021) compared application development using Angular and Vue.js, highlighting the advantages and disadvantages of each framework in terms of integration, performance, and development complexity. The study concluded that Vue.js is more suitable for projects with limited budgets and fewer development resources, while Angular is ideal for applications that require powerful built-in features and complex module management [9]. Shukla (2023) examines the future of JavaScript as a full-stack programming language, emphasizing how Angular, React, and Vue.js have become vital pillars in modern web development. The study shows that modularity, ease of use, and strong community support are essential factors contributing to the widespread adoption of these three frameworks [4]. Angular, React, and Vue.js have emerged as critical frameworks in modern web application development, each with unique strengths and challenges. This diverse research provides valuable guidance for developers in selecting the framework that best suits their specific project needs. By gaining a deeper understanding of the performance, scalability, ease of use, and community support of each framework, developers can make more informed and strategic decisions, ultimately contributing to the success of their web development projects.

5. Conclusion

This research thoroughly analyzes and compares three prominent JavaScript frameworks—Angular, React, and Vue. Js—in developing Single-Page Applications (SPAs). Each framework presents distinct strengths and limitations that should be evaluated in light of the specific demands of a project. React and Vue.js are particularly strong regarding initial load time and rendering speed, making them well-suited for applications requiring swift responsiveness and efficient memory usage. Despite having a longer initial load time due to its extensive feature set, Angular delivers solid performance once the application is fully loaded. Vue.js is notable for its gentle learning curve, making it an excellent option for developers new to the field and for projects that require rapid development cycles. With its comprehensive architecture and TypeScript integration, Angular is highly effective for large-scale enterprise applications but may be more challenging to learn. React offers a balanced approach, being relatively straightforward to learn while maintaining a high degree of flexibility, supported by extensive documentation and a robust community. React and Vue.js provide high levels of flexibility, allowing for seamless integration with various libraries and tools, which is advantageous for creating tailored solutions. Angular, while more rigid due to its opinionated structure, offers a comprehensive set of built-in features that facilitate the development of complex applications. React benefits from the largest and most active community, ensuring abundant resources, frequent updates, and a wide range of plugins. Angular also enjoys strong community support, particularly in the context of enterprise applications. Vue.js although supported by a smaller community, Vue.js is known for its highly active and engaged user base, especially within the open-source sector.

In conclusion, the selection of a JavaScript framework should be driven by the specific requirements and goals of the project. Angular is well-suited for enterprise-level applications that demand a strong structure and comprehensive built-in features. React is ideal for projects emphasizing performance, flexibility, and strong

community support. Vue.js is recommended for those prioritizing ease of use and rapid development, particularly for small to medium-sized projects. This study aims to assist developers in making informed and strategic decisions regarding selecting JavaScript frameworks, ultimately advancing web development practices and contributing to creating high-quality, efficient SPAs.

References

[1] Gavrila, V., Bajenaru, L., & Dobre, C. (2019). Modern single page application architecture: A case study. *Studies in Informatics and Control*, 28(2). https://doi.org/10.24846/V28I2Y201911

[2] Archer, G. (2022). Salvaging security for speedy single-page applications. *Network Security*, 2022(7), 34-37. https://doi.org/10.1016/S1353-4858(22)70034-7

[3] Kaluža, M., Troskot, K., & Vukelić, B. (2018). Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta u Rijeci*, 6(1), 261-282. https://doi.org/10.31784/zvr.6.1.19

[4] Shukla, A. (2023). Modern JavaScript frameworks and JavaScript's future as a full-stack programming language. *Journal of Artificial Intelligence & Cloud Computing*. https://doi.org/10.47363/jaicc/2023(2)144

[5] Sultan, M. (2018). Angular and the trending frameworks of mobile and web-based platform technologies: A comparative analysis. *International Journal of Advanced Computer Science and Applications*, 9(Special Issue), 128-135. https://doi.org/10.14569/SPECIALISSUE.2018.0901128

[6] Vyas, R. (2022). Comparative analysis on front-end frameworks for web applications. *International Journal for Research in Applied Science and Engineering Technology*, 10(4), 233-238. https://doi.org/10.22214/ijraset.2022.45260

[7] Bielak, K., Borek, B., & Plechawska-Wójcik, M. (2022). Web application performance analysis using Angular, React and Vue.js frameworks. *Journal of Computer Sciences Institute*, 19(1), 45-52. https://doi.org/10.35784/jcsi.2827

[8] Diniz-Junior, V., Figueiredo, C., Russo, G. S., Bahiense-Junior, M. L. A., Santos, L., Rocha, R., Bezerra, R., & Giuntini, F. (2022). Evaluating the performance of web rendering technologies based on JavaScript: Angular, React, and Vue. *2022 XVLIII Latin American Computer Conference (CLEI)*, 1-9. https://doi.org/10.1109/CLEI56649.2022.9959901

[9] Novac, O., Madar, D., Novac, C., Bujdosó, G., Oproescu, M., & Gal, T. (2021). Comparative study of some applications made in the Angular and Vue.js frameworks. *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)*, 1-4. https://doi.org/10.1109/EMES52337.2021.9484150

[10] Lipski, P., Kyć, J., & Pańczyk, B. (2021). Comparative analysis of the Angular 10 and Vue 3.0 frameworks. *Journal of Computer Sciences Institute*, 18(3), 78-86. https://doi.org/10.35784/jcsi.2688

[11] Javeed, A. (2019). Performance optimization techniques for ReactJS. *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 1-5. https://doi.org/10.1109/ICECCT.2019.8869134

[12] Skrzypiec, S., & Plechawska-Wójcik, M. (2023). Comparative analysis of Angular and React development frameworks. *Journal of Computer Sciences Institute*, 19(2), 61-70. https://doi.org/10.35784/jcsi.3724

[13] Freeman, A. (2021). Creating a stand-alone web app, part 2. In *Essential TypeScript 4* (pp. 249-264). Apress. https://doi.org/10.1007/978-1-4842-4979-6_15

[14] Souza, M., & Silva, E. (2021). Estudo comparativo de tecnologias de desenvolvimento front-end para web. *Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica*, 12(2), 201-208. https://doi.org/10.14210/COTB.V12.P201-208

[15] Boczkowski, K., & Pańczyk, B. (2020). Comparison of the performance of tools for creating a SPA application interface—React and Vue.js. *Journal of Computer Sciences Institute*, 17(2), 45-53. https://doi.org/10.35784/jcsi.1579

[16] Purohit, R., Jain, S., & Gupta, S. (2023). Role of Angular framework in web development. *International Journal of Advanced Research in Science, Communication and Technology*, 10(2), 103-115. https://doi.org/10.48175/ijarsct-10731

[17] Bosko, V., Konstantynova, L., & Fesechko, D. (2022). Analysis and research of the AngularJS framework as a website development tool. *Central Ukrainian Scientific Bulletin. Technical Sciences*, 5(36), 124-134. https://doi.org/10.32515/2664-262x.2022.5(36).1.124-134

[18] Saini, R., Yadav, R., Sharma, S., & Chauhan, N. (2023). Latest technologies in full stack web development. *Industrial Engineering Journal*, 52(1), 185-194. https://doi.org/10.36893/iej.2023.v52.185-194

[19] Li, N., & Zhang, B. (2021). The research on single page application front-end development based on Vue. *Journal of Physics: Conference Series*, 1883(1), 012030. https://doi.org/10.1088/1742-6596/1883/1/012030

[20] Baida, R., Andriienko, M., & Plechawska-Wójcik, M. (2020). Performance analysis of frameworks Angular and Vue.js. *Journal of Computer Sciences Institute*, 17(1), 78-85. https://doi.org/10.35784/jcsi.1577

[21] Rawat, C. (2022). E-business development research and analysis of front-end frameworks and libraries. *International Journal of Advanced Research in Science, Communication and Technology*, 11(1), 200-210. https://doi.org/10.48175/ijarsct-4892.

