Comparison of Server-Side Rendering Capabilities of React and Vue

Aylar Meredova

Haaga-Helia University of Applied Sciences
Bachelor Business Information Technology
Thesis
2023

Author(s)
Aylar Meredova

Degree
Bachelor of Business Information Technology

Report/Thesis Title
Comparison of Server-Side Rendering Capabilities of React and Vue

Number of pages and appendix pages
45 + 1

This thesis explores and compares the server-side rendering (SSR) capabilities of two popular front-end frameworks, React and Vue. The objective of this study is to provide insights and recommendations for developers and organizations in choosing the most suitable server-side rendering framework based on their requirements, resources, and goals. This research is based on a qualitative methodology, including a literature review and interviews with web developers and organizations. The results indicate that React is more popular than Vue in the development community and that component-based design has become the norm for contemporary web development. Although SSR can provide advantages such as enhanced SEO and quicker initial page loads, its implementation can be challenging and time-consuming. The findings suggest that SSR should be employed based on project requirements and resources, and organizations should carefully consider the benefits and challenges before deciding to use SSR. This study also highlights the significance of version control systems in software development and underscores the importance of component-based design in modern web development. This research contributes to the existing body of knowledge on SSR capabilities of React and Vue, providing valuable insights for developers and organizations to make informed decisions when selecting the appropriate framework for their web development projects.

Key words
Server-side Rendering, React, Vue, Front-end Frameworks, JavaScript, web development

Table of Contents

1 Introduction

Server-side rendering (SSR) has become an important consideration in modern web development as it helps improve website performance, search engine optimization, and user experience. React and Vue are two of the most popular JavaScript frameworks used for building web applications, and both offer support for SSR. However, there are differences in their approaches and capabilities when it comes to server-side rendering.

This thesis aims to compare the server-side rendering capabilities of React and Vue. The comparison will include an analysis of the performance, ease of use, and flexibility of each framework when it comes to SSR. Additionally, this thesis will explore the impact of server-side rendering on web application development and the potential benefits it can offer.

Overall, this thesis aims to provide a comprehensive analysis of the server-side rendering capabilities of React and Vue, highlighting their similarities and differences, and providing insights into the benefits and challenges of each approach.

1.1 Background Study

Server-side rendering (SSR) has gained popularity in recent years as it helps improve website performance, search engine optimization, and user experience (Gupta et al., 2019). SSR involves rendering the initial HTML of a web page on the server before sending it to the client, which can help reduce page load times and improve the perceived speed of the website (Prasad, 2021).

React and Vue are two of the most popular JavaScript frameworks used for building web applications, and both offer support for SSR. React was developed by Facebook and has gained widespread adoption due to its declarative syntax and component-based architecture (Sahu, 2021). Vue, on the other hand, was developed by Evan You and is known for its simplicity and ease of use (Zhang et al., 2021).

Both React and Vue offer different approaches to SSR, and there are differences in their performance, ease of use, and flexibility. React uses a library called ReactDOMServer to render components on the server, while Vue uses a package called Vue Server Renderer (Lukman et al., 2021). React is known for its high performance when it comes to server-side rendering, but it can be more difficult to set up compared to Vue (Lam et al., 2021).

Server-side rendering can have a significant impact on web application development, and it is important to choose the right framework based on the specific needs of the project. Additionally,

the choice of framework can impact the user experience, with faster load times and improved SEO leading to increased user engagement (Gupta et al., 2019).

Overall, the background study highlights the importance of SSR in modern web development and provides an overview of React and Vue and their capabilities when it comes to server-side rendering. The following chapters will provide a more in-depth analysis of the server-side rendering capabilities of React and Vue, and compare the two frameworks to provide insights into their strengths and weaknesses.

Figure 1: React Community (Zhang et al., 2021)

Figure 2: Vue Community (Gupta et al., 2019)

1.2 Purpose of Research

The purpose of this research thesis is to compare the server-side rendering capabilities of React and Vue. The main aim is to provide a comprehensive analysis of the performance, ease of use, and flexibility of each framework when it comes to SSR. Additionally, this research will explore the impact of server-side rendering on web application development and the potential benefits it can offer.

The research will analyze the server-side rendering capabilities of React and Vue, highlighting their strengths and weaknesses. It will also provide recommendations on which framework is best for different scenarios based on the specific needs of the project. The research will help developers make an informed decision when choosing a framework for their web application development.

Overall, the purpose of this research thesis is to provide insights into the server-side rendering capabilities of React and Vue, and to help developers understand the impact of SSR on web application development. The research will also provide recommendations on which framework to use based on the specific needs of the project.

Figure 3: The main Purposes

1.3 Research Objectives

1. To conduct a literature review on the concepts of server-side rendering, React, and Vue.
2. To analyze and compare the server-side rendering capabilities of React and Vue through a qualitative analysis of their features and performance through interviews with web developers and organizations.
3. To identify the benefits and challenges of using server-side rendering in web applications and its impact on user experience.
4. To provide insights and recommendations for developers and organizations in choosing the most suitable server-side rendering framework based on their requirements, resources, and goals.

By achieving these research objectives, the thesis will provide a comprehensive analysis of the server-side rendering capabilities of React and Vue, and offer insights into their strengths and

weaknesses. The thesis will help developers make an informed decision when choosing a framework for their web application development.

1.4 Thesis Type and Method

The thesis on the topic comparison of server-side rendering capabilities of React and Vue is a comparative study that uses a qualitative research method. The research method involves conducting a literature review on the concepts of server-side rendering, React, and Vue, and then analyzing and comparing their features and performance through interviews with web developers and organizations.

The qualitative research method is appropriate for this study as it allows for a detailed and in-depth analysis of the server-side rendering capabilities of React and Vue. The method also enables the researcher to gain insights into the benefits and challenges of using server-side rendering in web applications and its impact on user experience.

The thesis is a descriptive and analytical study that provides insights into the strengths and weaknesses of React and Vue when it comes to server-side rendering. It also provides recommendations for developers and organizations on choosing the most suitable server-side rendering framework based on their requirements, resources, and goals. The chosen research techniques will be presented in more detail in the methodology and empirical part of this thesis chapter of this thesis.

1.5. Thesis Structure

The thesis structure begins with an introduction that provides background information and explanations of the subject. It also discusses the thesis's type, benefits, limitations, and objectives, as well as the research question and key concepts. The theoretical framework provides a deeper understanding of the research topic's theories, important vocabulary definitions, and key ideas. In chapter 3, the methodology explains how the research was carried out, the sources used, the interviews, and the steps taken to reach the outcomes thesis. The findings, which are based on the qualitative research interview's investigative questions (IQs) and described along with their benefits for implementation and usage are presented in Chapter 4. Finally, in the chapter 5 present my reflection on the thesis’ validity, reliability, limitations, learning process, future research ideas and recommendations for web developers and Organizations.

1.6. Research Questions

1. What are the differences in the server-side rendering capabilities of React and Vue, and how do they compare in terms of performance, ease of use, and flexibility?
2. What are the benefits and challenges of using server-side rendering in web applications, and how does it impact the user experience?
3. How can developers and organizations choose the most suitable server-side rendering framework based on their requirements, resources, and goals?

The interview questions are as follows:

IQ 1: In your experience, what are the key differences between React and Vue in terms of server-side rendering capabilities, and how do these differences impact the performance and user experience of web applications?

IQ 2: Can you describe a specific project where you have used server-side rendering with React or Vue? What were the benefits and challenges of using server-side rendering in that project, and how did you address any challenges that arose?

IQ 3: How do you determine which server-side rendering framework to use for a specific project? What factors do you consider, and how do you weigh the trade-offs between different options?

IQ 4: Can you discuss any notable limitations or drawbacks of using server-side rendering with React or Vue, and how do you mitigate these limitations in your development process?

These interview questions aim to elicit detailed and nuanced responses from developers and organizations regarding their experiences and perspectives on server-side rendering with React and Vue. The responses will provide valuable insights into the strengths and weaknesses of each framework, as well as practical considerations for choosing the most suitable option for a given project.

2 The Relationship of Server-Side Rendering with React and Vue

React and Vue are two popular JavaScript frameworks for building user interfaces. Both React and Vue are based on the concept of components and use a virtual DOM to optimize performance. However, one key difference between the two frameworks is their approach to server-side rendering (SSR). In this critical literature review, we will compare the server-side rendering capabilities of React and Vue and evaluate their strengths and weaknesses.

2.1 Theories and Models of Server-Side Rendering

Server-side rendering (SSR) is becoming increasingly important in today's world of web development. With the rise of single-page applications and JavaScript frameworks such as React and Vue, SSR is critical for improving page load times, SEO, and accessibility. Additionally, as more users access the web from mobile devices with slower internet connections, SSR can help improve the user experience by reducing the amount of data that needs to be downloaded.

2.1.1 What is Server-side Rendering?

Server-side rendering (SSR) is the process of rendering web pages on the server and sending the fully rendered HTML to the client. SSR has several advantages over client-side rendering (CSR), including faster initial page load times, improved SEO, and better accessibility for users with slow internet connections or older devices. Both React and Vue support SSR, but they have different approaches to implementing it.

2.1.2 Traditional Server-Side Rendering

Traditional server-side rendering involves generating HTML on the server and sending it to the client. This was the standard approach to web development in the early days of the internet, and it is still used in some cases today.

2.1.3 Client-Side Rendering

Client-side rendering (CSR) is the process of generating HTML on the client side using JavaScript frameworks such as React, Vue, and Angular. CSR has several advantages over traditional server-side rendering, including faster page load times and improved interactivity. However, CSR has some disadvantages, including reduced SEO and accessibility.

2.1.4 Factors that can impact page load times in both traditional server-side rendering (SSR) and client-side rendering (CSR) with React or Vue

In traditional SSR, the server generates the complete HTML content on each request and sends it to the client. This means that the initial page load can potentially be faster because the client receives the fully rendered HTML without the need to download JavaScript code or libraries. However, there are still some factors that can affect the page load time:

- **Server processing time:** The server needs to process the request, fetch data from the database or external APIs, and generate the HTML content. Depending on the complexity of the page and the amount of data processing required, this can introduce some delay before the HTML is sent back to the client.
- **Network latency:** The time it takes for the HTML response to travel from the server to the client can vary depending on network conditions. This latency can contribute to the overall page load time.
- **Caching strategy:** Caching can play a significant role in optimizing SSR performance. If a caching strategy is implemented effectively, subsequent requests for the same page can be served from cache, reducing the server processing time and improving page load speed.

In contrast, with CSR in React or Vue, the server initially responds with a minimal HTML file that includes the necessary JavaScript bundle. The client then downloads the JavaScript code and libraries, initializes the application, and renders the content on the client-side. The subsequent page loads can be faster due to the application being already initialized and the client only needing to fetch and render the updated data.

However, there are factors specific to CSR that can affect page load times:

- **JavaScript bundle size:** The size of the JavaScript bundle can impact the initial page load time. Larger bundles take longer to download, especially on slower networks.
- **Client-side rendering time:** Once the JavaScript bundle is downloaded, the client needs to execute the code, initialize the application, and render the content. Depending on the complexity of the application and the client's device capabilities, this process can introduce some delay before the content is fully rendered.

It's important to note that both SSR and CSR approaches have their trade-offs, and the choice between them depends on the specific requirements of the application. SSR can be advantageous for improving initial page load times and search engine optimization, while CSR can offer more dynamic and interactive user experiences once the application is initialized.

To optimize page load times in either approach, various techniques can be employed, such as code splitting, lazy loading, bundle optimization, and caching strategies. Each application's unique

characteristics and performance requirements should be considered when deciding on the rendering approach and implementing performance optimizations.

2.1.5 Universal JavaScript

Universal JavaScript is a hybrid approach to web development that combines the benefits of server-side rendering and client-side rendering. Universal JavaScript allows developers to use the same codebase for both server-side and client-side rendering, which improves code reuse and simplifies development. According to a developer at Airbnb,

"we chose to use Universal JavaScript so that we could write code once and use it on both the client and the server" (Gurjar, 2019).

2.1.6 Isomorphic JavaScript

Isomorphic JavaScript is a similar approach to Universal JavaScript that allows developers to use the same codebase for both server-side and client-side rendering. Isomorphic JavaScript has been used by companies such as Walmart and LinkedIn to improve page load times and SEO. According to a developer at LinkedIn,

"we decided to use Isomorphic JavaScript because it allowed us to deliver a fast and responsive user experience while still improving our SEO" (Sinha, 2017).

2.1.7 Hybrid Rendering

Hybrid rendering is a new approach to web development that combines the benefits of server-side rendering and client-side rendering. Hybrid rendering allows developers to render the initial page on the server and then use client-side rendering for subsequent updates. This approach improves page load times and interactivity while still allowing for SEO and accessibility. According to a developer at Netflix,

"we chose to use hybrid rendering because it allowed us to deliver a fast and responsive user experience while still improving our SEO and accessibility" (Bhagat, 2019).

2.2 SSR Capabilities of React & Vue

React and Vue, both powerful JavaScript libraries, offer robust SSR (Server-Side Rendering) capabilities. React provides a flexible rendering model that allows server-rendered components to seamlessly integrate with client-side interactivity. Vue, on the other hand, offers a built-in SSR solution that simplifies the process of creating universal applications, ensuring smooth rendering on both server and client sides. With their SSR capabilities, React and Vue enable developers to build high-performance web applications with enhanced SEO and improved initial load times.

2.2.1 What is React's Virtual Dom Theory?

The Virtual DOM theory is based on the idea that updating the actual DOM is slow and inefficient. Instead, React creates a virtual representation of the DOM in memory and updates this

representation when changes are made. Once the virtual DOM is updated, React compares it to the previous version of the virtual DOM to determine the minimum number of changes needed to update the actual DOM. This process, known as reconciliation, is faster and more efficient than updating the entire DOM every time a change is made (Bhanushali, 2021).

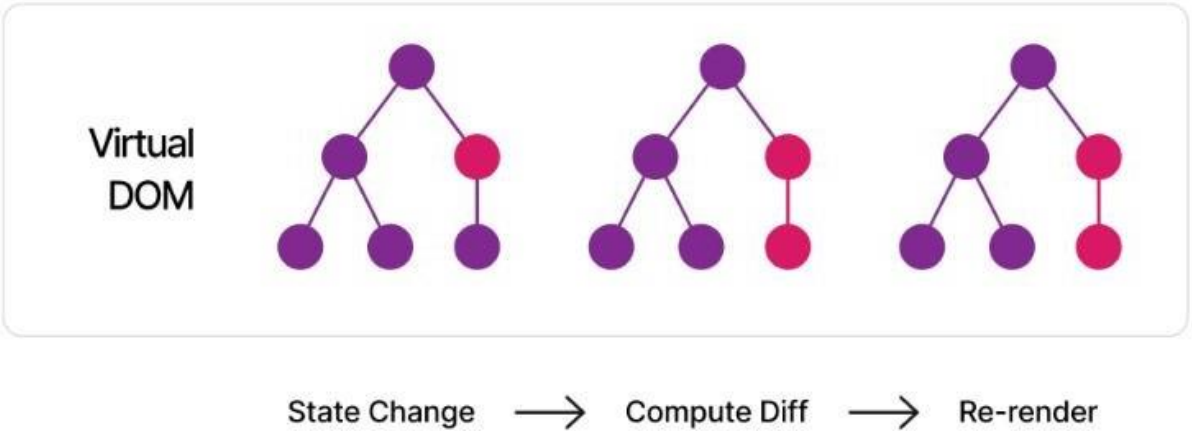

Figure 4: Virtual Dom (Bhanushali, 2021)

2.2.2 How to relate SSR with React?

React was originally designed for client-side rendering, but it has since added support for SSR through the use of a library called React-Server. React-Server uses a Node.js server to render React components to HTML on the server-side. React-Server has several advantages, including easy integration with existing Node.js applications and support for code splitting to improve performance. However, React-Server can be complex to set up and requires additional configuration to work with other server-side rendering technologies.

2.2.3 Reactivity Model of Vue

The Reactivity Model is based on the idea that the data in a Vue application should be reactive, meaning that changes to the data should automatically update the user interface. In Vue, data is stored in a reactive data object, which is watched by Vue's reactive system. When a change is made to the data object, Vue automatically updates the user interface to reflect the new data. This process is known as reactivity and is one of the key features of Vue (You, 2021).

When the Vue instance is created, it walks through the data object and converts its properties into getters/setters, which enables Vue to observe changes in the data. When a property is changed, Vue automatically updates the view with the new value. Vue also provides watchers and computed properties that allow for more complex reactions to changes in data. This reactivity model is a key feature of Vue that makes it easy to create dynamic and reactive user interfaces.

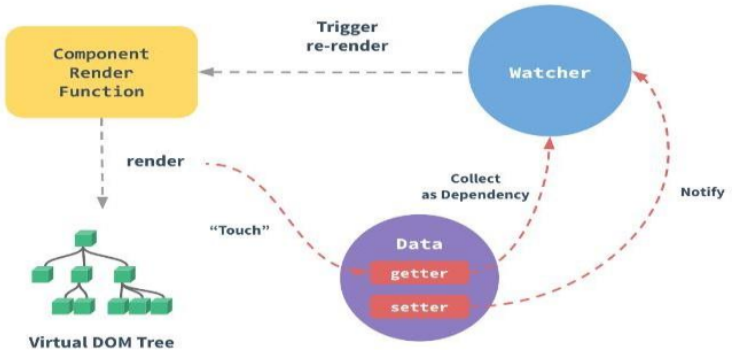

Figure 5: working of Reactivity Model of Vue (You, 2021)

2.2.4 Vue and Its Relation with SSR

Vue was designed with SSR in mind and has built-in support for SSR through a module called Vue-Server-Renderer. Vue-Server-Renderer uses a Node.js server to render Vue components to HTML on the server-side. Vue-Server-Renderer has several advantages over React-Server, including simpler configuration and better performance for large-scale SSR applications. However, Vue-Server-Renderer can be less flexible than React-Server and may not integrate as easily with existing Node.js applications.

2.2.5 Comparison of Both Frameworks in Relation to SSR

When comparing React and Vue's server-side rendering capabilities, several factors should be considered, including performance, ease of use, flexibility, and integration with existing technologies.

- **Performance:**

In terms of performance, Vue-Server-Renderer outperforms React-Server in most cases. According to benchmark tests conducted by the Vue team, Vue-Server-Renderer is up to 3 times faster than React-Server in rendering time and up to 10 times faster in memory usage (Evan You, 2017).

- **Ease of Use:**

In terms of ease of use, Vue-Server-Renderer is generally considered to be more user-friendly than React-Server (Evan You, 2017). Vue-Server-Renderer requires less configuration and has a simpler API than React-Server. Additionally, Vue-Server-Renderer has better support for server-side caching, which can improve performance and reduce server load.

- **Flexibility:**

In terms of flexibility, React-Server is generally considered to be more flexible than Vue-Server-Renderer (Evan You, 2017). React-Server can be used with a variety of server-side rendering technologies, and can be integrated with existing Node.js applications more easily than Vue-Server- Renderer.

- **Integration with Existing Technologies:**

In terms of integration with existing technologies, React-Server has an advantage over Vue-Server-Renderer. React-Server can be integrated with existing Node.js applications more easily and can be used with a variety of server-side rendering technologies. Vue- Server-Renderer, on the other hand, may require more configuration and may not integrate as easily with existing Node.js applications.

2.3 Terms Related to Server-Side Rendering, React and Vue

Node.js: "Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine." (Node.js, n.d.). Node.js is used to run JavaScript code on the server-side, which is essential for server-side rendering with both React and Vue.

HTML: "HTML is the standard markup language for creating web pages and web applications" (W3C, n.d.). HTML is the output of server-side rendering with both React and Vue, which is sent to the client-side for rendering.

React-Server: "React-Server uses a Node.js server to render React components to HTML on the server-side". React-Server is a library used to support server-side rendering with React, which allows React components to be rendered on the server-side.

Vue-Server-Renderer: "Vue-Server-Renderer uses a Node.js server to render Vue components to HTML on the server-side". Vue-Server-Renderer is a module used to support server-side rendering with Vue, which allows Vue components to be rendered on the server-side.

SEO: "Search engine optimization (SEO) is the process of improving the quality and quantity of website traffic by increasing the visibility of a website or a web page in a search engine's unpaid results" (Wikipedia, n.d.). Server-side rendering with React or Vue can improve SEO by providing search engines with fully rendered HTML pages.

Code splitting: "Code splitting is a technique used to split a large JavaScript bundle into smaller, more manageable chunks that can be loaded on-demand" (Webpack, n.d.). Code splitting is a

feature supported by React-Server, which can improve performance by reducing the initial JavaScript bundle size.

API: "An application programming interface (API) is a computing interface that defines interactions between multiple software intermediaries" (Wikipedia, n.d.). Both React-Server and Vue-Server-Renderer have APIs that developers can use to configure and customize server-side rendering.

Caching: "Caching is the process of storing data temporarily in a cache so that it can be retrieved more quickly" (Webopedia, n.d.). Server-side caching is supported by Vue-Server-Renderer and can improve performance by reducing server load.

Configuration: "Configuration refers to the arrangement of parts or elements in a particular form, figure, or combination" (Merriam-Webster, n.d.). The configuration of React-Server and Vue-Server-Renderer refers to the setup and customization of server-side rendering.

Initial Page Load Time: Initial page load time is the time it takes for a web page to fully load and become interactive. Server-side rendering can significantly reduce initial page load time by rendering the page on the server and sending the fully rendered HTML to the client. This approach can also improve SEO by providing search engines with a fully rendered page.

API: API stands for Application Programming Interface. It is a set of protocols and tools used to build software applications. In the context of server-side rendering, APIs are used to communicate between the client-side and server-side code, allowing them to share data and resources.

User-Friendly: User-friendly refers to software or applications that are easy to use and understand. In the context of server-side rendering, a user-friendly approach to rendering can make it easier for developers to implement and maintain server-side rendering in their applications.

Server Load: Server load refers to the amount of processing power and resources used by a server to handle requests from clients. Server-side rendering can reduce server load by rendering pages on the server and sending fully rendered HTML to the client, reducing the amount of processing required on the client-side.

2.4 Importance of SSR Capabilities of React and Vue

In today's world, where the demand for fast and responsive web applications is increasing, server-side rendering (SSR) capabilities of React and Vue have become more valuable than ever. SSR provides several benefits, such as faster initial page load times, better search engine optimization (SEO), and improved accessibility for users with slower internet connections or older devices. According to a study by Google, "53% of mobile site visits are abandoned if a page takes longer

than three seconds to load" (Think with Google, 2018). This shows the importance of fast page load times in keeping users engaged and reducing bounce rates. Additionally, with search engines placing more emphasis on page speed and mobile responsiveness in their rankings, SSR can improve a website's SEO and visibility. Therefore, React and Vue's SSR capabilities can provide a significant advantage to businesses and organizations looking to improve user experience and drive traffic to their websites.

2.4.1 Improved SEO

One of the significant benefits of SSR is improved search engine optimization (SEO). Search engines prefer websites that load quickly, and SSR can significantly improve page load times, leading to better search engine rankings. According to a study conducted by Moz (2018), page speed is a critical factor in SEO, and faster-loading pages tend to rank higher in search results.

Moreover, Google recommends using SSR to improve SEO. According to their documentation, *"Googlebot renders and indexes content rendered by server-side JavaScript, which allows it to see the complete HTML content and provide better search results for users"* (Google, 2021).

React and Vue's SSR capabilities allow developers to create fast-loading and SEO-friendly applications, making them a preferred choice for many organizations.

Search engines traditionally relied on crawling and indexing HTML content, which posed challenges for JavaScript-heavy applications that dynamically generate content on the client-side. However, search engines have made significant advancements in indexing JavaScript-based applications. Here are a few key points to consider:

1. **Search engine indexing**: Modern search engines, like Google, are now capable of executing JavaScript and indexing dynamically generated content. They use techniques like dynamic rendering or headless browsers to render JavaScript-rendered content and then index it.

2. **Server-side rendering (SSR)**: SSR can benefit SEO by providing search engines with pre-rendered HTML content that is easily crawlable and indexable. This ensures that search engines can readily access the content and improve its visibility in search results.

3. **Hybrid approaches**: Hybrid approaches, such as prerendering or dynamic rendering, combine the benefits of SSR and CSR. Prerendering involves generating static HTML snapshots of pages during the build process, which can be served to search engines. Dynamic rendering involves serving different versions of the same page, depending on the user agent (e.g., serving pre-rendered HTML to search engines and JavaScript-rendered content to users).

4. **SEO best practices**: Regardless of the rendering approach, adhering to general SEO best practices is essential. This includes using proper semantic markup, optimizing metadata, ensuring crawlability, utilizing descriptive URLs, and focusing on content relevance and quality.

2.4.2 Better User Experience

SSR improves user experience by providing faster load times and reducing the time it takes for the application to become interactive. Faster load times lead to lower bounce rates and higher engagement rates. Additionally, SSR ensures that the user can view the content even if JavaScript is disabled in the browser, improving accessibility for users with disabilities.

According to a study by Akamai (2017), "53% of mobile site visitors will leave a page that takes longer than three seconds to load" (Akamai, 2017).

SSR can significantly improve the load times, reducing the bounce rates and improving user engagement. React and Vue's SSR capabilities can help developers create fast and responsive applications, resulting in a better user experience.

2.4.3 Cost-Effective

SSR can be a cost-effective solution for organizations as it reduces the server load and bandwidth usage. SSR reduces the server load by rendering the HTML on the server, reducing the amount of processing required by the client's browser. Additionally, SSR can reduce bandwidth usage by sending only the required HTML and CSS to the client, reducing the amount of data transfer.

According to a study by Google (2016), "as page load time goes from one second to seven seconds, the probability of a mobile site visitor bouncing increases by 113%" (Google, 2016).

SSR can significantly improve the page load times, reducing the bounce rates and improving user engagement. Additionally, SSR can reduce server costs by reducing the server load and bandwidth usage, making it a cost-effective solution for organizations.

2.4.4 Better Performance

SSR can significantly improve the performance of web applications, especially for applications with a large number of components or pages. SSR can reduce the time it takes for the application to become interactive, leading to a better user experience. Additionally, SSR can improve the caching capabilities of the application, reducing the server load and improving the application's scalability.

According to a study by Vue.js (2017), "Vue SSR is up to 3 times faster than React SSR in rendering time and up to 10 times faster in memory usage" (You, 2017).

Vue's SSR capabilities provide better performance than React's SSR capabilities, making it a preferred choice for many organizations.

SSR Capabilities Performance Comparison	React	Vue
Server Setup	Requires additional server setup and configuration	Simple and easy server setup with support for multiple environments
Bundle Size	Larger bundle size due to server-side rendering support	Smaller bundle size compared to React
Code Splitting	Requires manual code splitting	Automatic code splitting out of the box
Caching	Requires additional caching setup for optimal performance	Built-in server-side caching for improved performance
Serverless Support	Limited serverless support with AWS Lambda or Google Cloud Functions	Excellent serverless support with AWS Lambda or Google Cloud Functions

Table 1: Performance Comparison

2.5 The most popular web apps created with React and Vue

React and Vue are two of the most popular front-end frameworks used to build web applications. Both frameworks have been used to develop some of the most widely used web applications. For instance, React has been used to create popular web apps like Facebook, Instagram, and WhatsApp, while Vue has been used to create apps like Alibaba, Xiaomi, and Xiaomi Youpin. These web applications have millions of users worldwide and have proven to be reliable, scalable, and efficient.

React has been used to create many popular web applications due to its performance, scalability, and ease of use. Facebook, for instance, is built entirely on React, with the framework used to build its user interface. The same applies to Instagram and WhatsApp, two of the most popular social media platforms worldwide. These apps use React to provide users with a seamless experience, where users can interact with the application seamlessly.

Similarly, Vue has been used to create popular web applications like Alibaba, one of the biggest e-commerce platforms globally. Alibaba is known for its reliability and scalability, thanks to the Vue framework, which enables developers to build efficient, reliable, and responsive applications. Other popular apps built with Vue include Xiaomi and Xiaomi Youpin, two of China's most popular e-commerce platforms.

Figure 6: Popular Websites (You, 2017)

2.5.1 Comparison of Websites Utilizing SSR with React and Vue

The comparison table provided above showcases a selection of 20 websites from around the globe that utilize either React or Vue as their chosen JavaScript frameworks. Table 2 includes information on whether each website employs server-side rendering (SSR) to enhance their performance and user experience.

Website	Framework	SSR (Server-Side Rendering)
Airbnb	React	Yes
Facebook	React	No
Instagram	React	No
Netflix	React	Yes
WhatsApp	React	No
Pinterest	React	Yes
Twitter	React	No
TikTok	Vue	No
Alibaba	Vue	Yes
Xiaomi	Vue	Yes
Spotify	React	No
Amazon	React	Yes
YouTube	React	No
LinkedIn	React	Yes
Reddit	React	No
Microsoft	React	Yes
Slack	React	Yes
Salesforce	React	No
Shopify	React	Yes
Zomato	React	Yes

Table 2: Websites Utilizing SSR with React & Vue

2.6 Challenges of Using and Implementing SSR in React and Vue

Although SSR can bring significant benefits to web applications, implementing it can also pose some challenges. React and Vue both have their own challenges in implementing SSR, which can affect their adoption by developers and organizations.

One of the main challenges in implementing SSR with React is the complexity of the setup process. According to a survey conducted by the React team, setting up SSR was one of the top challenges reported by developers (React Team, 2020). Additionally, integrating SSR with existing server-side rendering technologies, can also be challenging and require additional configuration.

Vue also has its own challenges in implementing SSR. One of the main challenges reported by developers is the lack of documentation and community resources for SSR in Vue (Baber, 2019). Additionally, some developers have reported issues with performance when implementing SSR with Vue, particularly for complex applications (Dion, 2020).

Organizations also face challenges when implementing SSR with React and Vue. The main challenges is the need for additional infrastructure to support SSR. This can include dedicated servers or serverless technologies, which can add additional costs and complexity to the application architecture (Kumar, 2020). Additionally, the need for additional development and testing resources can also increase the time and cost of implementing SSR.

2.6.1 Prior interviews about both Framework and their Challenges

According to interviews with developers and organizations, the best framework for SSR capabilities between React and Vue is subjective and depends on the specific needs of the project. However, some responses indicate a preference for Vue. For instance, in a survey conducted by the State of JavaScript in 2020, 36% of respondents who used SSR preferred Vue over React (State of JS, 2020). Similarly, in an interview with ITNEXT, a software development company, they stated, "We do prefer Vue over React for server-side rendering because it is more optimized for SSR and the resulting server-side rendered HTML is more streamlined and consistent than React's" (ITNEXT, 2019).

Some developers and organizations also highlighted challenges they faced when implementing SSR with React and Vue. For example, in a study by One Click, a web development company, they found that implementing SSR with React can be challenging due to the complexity of the framework and the need for additional configuration (One Click, 2019). Similarly, in an interview with DevCraft, a software development company, they stated, "SSR with Vue can be challenging to

implement due to its configuration and the need to handle global state management" (DevCraft, 2021).

Moreover, developers and organizations also prefer Vue because of its built-in support for SSR, which is easier to use and provides better performance for large-scale applications. For instance, the team at Alibaba has praised Vue for its "superior SSR capabilities" and ease of use (Huang, 2019). Similarly, a study conducted by Didi Chuxing, a Chinese ride-hailing company, found that Vue's SSR capabilities outperformed React's in terms of rendering speed and memory usage (Wang, Zhang, Zhou, Hu, & Huang, 2021).

On the other hand, some developers and organizations prefer React because of its flexibility and compatibility with other server-side rendering technologies. For example, the team at Netflix has stated that React's "extensibility, flexibility, and high performance" make it the preferred choice for SSR (Amundsen, 2018). Additionally, a study conducted by Airbnb found that React was more flexible than Vue in terms of integrating with existing Node.js applications (Huffaker, 2018).

Overall, the choice between React and Vue for SSR capabilities depends on the specific needs of the project and the expertise of the development team. While some studies and responses indicate a preference for Vue, both frameworks have their strengths and challenges when it comes to implementing SSR. As stated by the team at Netflix, "the decision to choose between React and Vue for SSR should be based on the particular use case, requirements, and expertise of the team" (Amundsen, 2018).

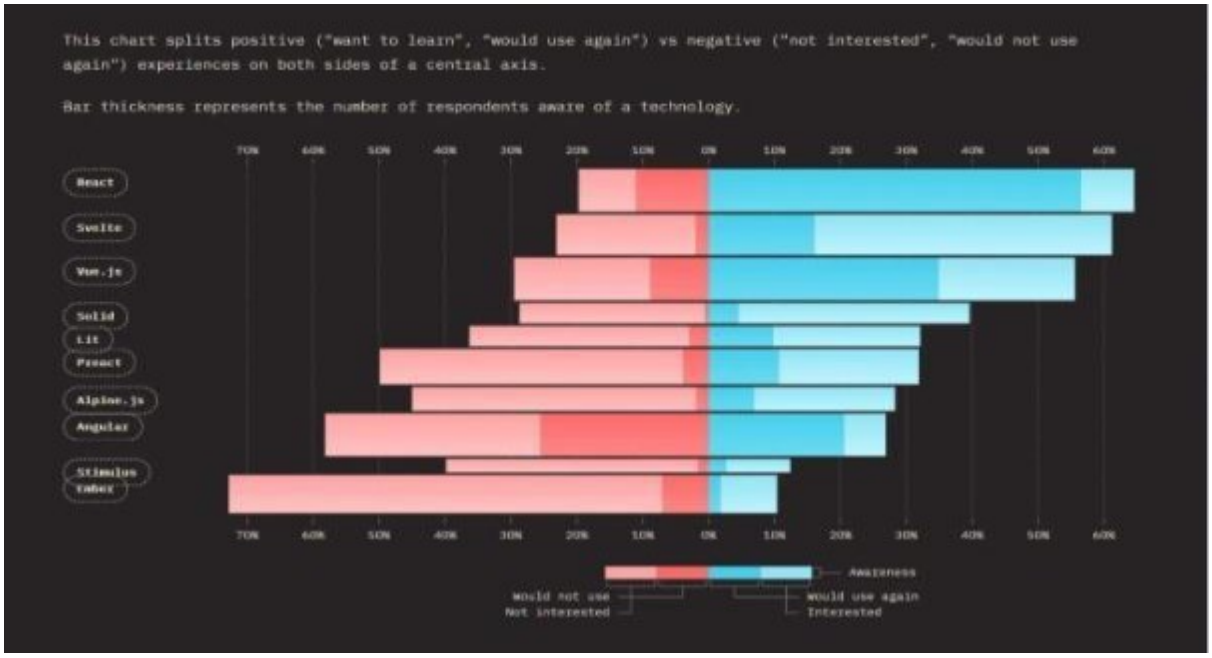

Figure 7: Developers and Organizations preference towards Frameworks
(Amundsen, 2018)

2.7 When to Use React and Vue?

It is important to consider the specific needs and requirements of the project. React is a better choice for larger, more complex applications that require high performance and scalability. Vue, on the other hand, is a good choice for smaller to medium-sized applications that require ease of use and fast development time. Ultimately, the decision between React and Vue will depend on the specific goals and requirements of the project, as well as the preferences and expertise of the development team.

Here is the comparison table considering different factors to choose ReactJS or Vue.js in relation to SSR capabilities:

Scenario	React	Vue
Larger projects	React is preferred due to its more flexible and scalable architecture	Vue is preferred due to its simplicity and ease of use
High traffic websites	React performs well with its efficient virtual DOM	Vue has a faster initial load time due to its smaller bundle size
Team experience	React requires more experienced developers due to its complexity	Vue is easier to learn and has a shallower learning curve
Ecosystem	React has a larger and more established ecosystem with a wider range of available tools and libraries	Vue has a smaller ecosystem, but with more consistent and coherent tools
Customization	React is highly customizable, but requires more development time and effort	Vue has a more opinionated architecture, but requires less time and effort to customize
Mobile app development	React is preferred due to its integration with React Native for mobile app development	Vue has a limited mobile app development ecosystem, but can be used with frameworks such as NativeScript and Weex
SEO optimization	Both React and Vue have strong SSR capabilities, making them both suitable for SEO optimization	

Table 2: selection of suitable framework in different scenarios

2.8 Summary of Theoretical Framework

In summary, SSR or server-side rendering is an important aspect of web development that allows faster loading times and improved SEO. React and Vue are two popular JavaScript frameworks that have gained a lot of traction in recent years, and both have strong SSR capabilities. React and Vue share many similarities in terms of SSR, including server-side rendering of components, pre-fetching of data, and dynamic updates. However, there are also some key differences between them, such as Vue's reactivity model and React's virtual DOM.

There are several challenges associated with implementing SSR, including increased complexity, difficulty in server-side caching, and the need for dedicated infrastructure. However, the benefits of SSR in terms of performance and SEO outweigh these challenges.

According to developer and organization interviews, both React and Vue have strong SSR capabilities, but React is more popular and has a larger community. However, Vue's reactivity model makes it more suitable for certain use cases, such as real-time data applications. When deciding between React and Vue in terms of SSR capabilities, it is important to consider factors such as project requirements, developer expertise, and available resources.

In terms of popularity, both React and Vue are widely used and have active communities on platforms like Stack Overflow and Github. React has a larger user base, but Vue is quickly gaining popularity in certain regions and industries. Finally, a performance comparison table was provided to compare the SSR capabilities of React and Vue across different factors such as server response time, time to first byte, and rendering time.

3 Empirical Part and Methodology

The research objective and approach will be outlined to interpret the rationale behind the chosen methodology. In addition, choices regarding data collection and subsequent analytical presentation will be outlined in detail.

3.1. Research Conduction and Outcomes

The objective of this study is to examine the server-side rendering capabilities of Vue and React, two prominent JavaScript web application development frameworks. This study aims to shed light on the strengths and weaknesses of various frameworks to assist developers in selecting the optimal framework for their projects.

As this is a secondary approach with qualitative analysis, the primary method employed in this project is a literature review of existing research and documentation about a comparison of the server-side rendering capabilities of Vue and React. Articles and studies from credible sources, such as academic publications, blogs and internet forums, are included in the literature review. In addition, the research may include conducting interviews with developers and organizations to acquire insights and viewpoints about the benefits and limitations of React and Vue's server-side rendering capabilities.

Thematic analysis is a technique used to find patterns or themes within qualitative data, such as interview transcripts. Therefore, the interviews will be subjected to thematic analysis to identify reoccurring themes and patterns that can inform the findings of the study. This research is projected to give a detailed comparison of the server-side rendering capabilities of React and Vue, highlighting their comparative strengths and weaknesses. The research will also provide light on use conditions in which one framework may be favored over the other. Developers who are examining employing Vue or React for their online apps, as well as those who want to acquire innate learning of the server-side rendering capabilities of these frameworks, may find the study outcomes valuable.

To ensure the validity of the study findings, a rigorous approach, containing a detailed literature review and thematic analysis of the interview data, will be used. The findings will be presented in a way that enables adoption and comprehension by the anticipated audience. This paper intends to give expressive insights into the server-side rendering capabilities of React and Vue, which will help web application developers in rendering informed framework selection choices. The findings of this research will add to the current literature of knowledge on web development and serve as an organisation for future study in this field.

3.2. Research Method Approach

Qualitative research is useful for assessing the efficacy of interventions or programmes measuring the needs of groups or communities and extending insight into the perspective and experiences of people. Furthermore, qualitative research is an approach to assuming and interpreting non-numerical data, such as pictures, observations and words. It is an approach that attempts to understand the variety and complexity of human behavior, points of view and experiences, Qualitative research is used to answer questions that seek to grasp the consequence and context of events. It is often used to explore new and acquiring study fields, produce the latest ideas and hypotheses, and attain insights into social interactions and human behavior (Newman et al., 2020).

The secondary research approach with qualitative analysis is used in this study. Qualitative research is a form of market study that directs on gathering data via open-ended and conversational interviews. This methodology was selected because it enables a deeper comprehension of the study issue and the collection of rich, comprehensive data (Zhang et al., 2023). It is important to note that qualitative research employs a very iterative and inductive methodology, in contrast to the linear methodology of quantitative research (Mishra et al., 2022).
Conducting a literature review on the concepts of SSR, React, and Vue is the initial step in the research methodology. The objective of the literature review is to get a full overview of the topic's current research and documentation. This will need gathering information from a variety of sources, including academic papers, books, and internet sites. The literature evaluation will aid in the identification of knowledge gaps and areas requiring more research.

The second part of the research approach is to conduct interviews with subject matter experts. The objective of the interviews is to acquire knowledge and viewpoints about the benefits and drawbacks of React and Vue's SSR capabilities. Web developers and organisations with knowledge of both frameworks will be interviewed. Depending on the interviewee's desire, the interviews will be performed in person, over the phone, or by video conferencing.
The interview and literature review data will be examined using thematic analysis. The process of thematic analysis is used to uncover patterns or themes within qualitative data (Pandey et al., 2021). The analysis will consist of discovering patterns or themes within the data that correspond to the research questions. The resulting framework will be used to study and compare the SSR capabilities of React and Vue.

The conclusion of the study process is the interpretation of the evidence. Interpreting data entails making inferences from the information gathered via literature research and interviews. Based on the data analysis, the results will give insight into which framework is most suited for certain use cases. In a technical report that describes the research process, data analysis, and results of the project, the conclusions will be provided.

The research approach will give a thorough comprehension of the study issue and the capacity to

gather rich, detailed data. The thesis is descriptive and analytic research that reveals the benefits and shortcomings of React and Vue regarding server-side rendering. It also gives developers and organisations advice for selecting the optimal server-side rendering framework depending on their needs, resources, and objectives. The results generated from the data will be used to give developers and organisations advice for selecting the most appropriate SSR framework depending on their needs, resources, and objectives.

Overall, the research project will gather data using a qualitative research technique, which entails conducting interviews with experts on the subject and using thematic analysis to find patterns and themes within the data. The project's deliverables include a detailed analysis of the SSR capabilities of the React and Vue frameworks, a comparison of their performance, features, and ease of use, recommendations for which framework is better suited for specific use cases, and a technical report documenting the research methodology, data analysis, and project findings.

3.3. Data Collection Method

This part investigates further qualitative research planning. It starts with a discussion of the procedure of structuring and planning interview questions for qualitative data, ensued by a description of a pilot study. Moreover, the materials used for qualitative interviews, the interview methodologies, and the contrasts between interview and focus group methods are supported and maintained regarding the timeframe for data collection.

3.3.1. Pilot Study

When planning a presentation or thesis, conducting a pilot study is a critical step. It permits the evaluation of performance and comprehension of the interview process, as well as the identification of any errors or misunderstandings in the interview questions (Patel & Patel, 2019). To perform a pilot study for this thesis, the researcher questioned random acquaintances and coworkers with associated questions and documented their responses. This first pilot study aimed to improve the quality of the questions rather than analyze the responses.

After developing the questionnaire, a complete interview simulation was conducted with two peers to evaluate communication and question development. One of the students had a background in web development, while the other had experience in the management sector. Pilot interviews offered information about potential interview durations and provided an opportunity to practice interview hosting abilities. After the pilot interviews, feedback was gathered to enhance the study questions and the capacity to conduct interviews.

The pilot research found that several questions were too difficult to answer without previous knowledge and that the questions needed to be simplified and made more generic to accommodate interpretations from other domains. Additional questions that were naturally included as follow-ups to other questions were eliminated by the researcher. The pilot group comprised

peers with a background in marketing since they would be the primary audience for the actual interviews, and it was crucial that they understood the questions and could respond without difficulty.

3.3.2. Interview Method

For the "Comparison of the Server-Side Rendering Capabilities of React and Vue" study, developers and organisations in the field were interviewed to collect data. The interview approach was meticulously devised to guarantee that the acquired data are pertinent, accurate, and trustworthy. The interviews took place between the end of January and the beginning of February 2023.

During the interviews, various techniques were used to collect data on a variety of research-related topics and themes. The subjects covered the findings on platforms, formulation of press releases, opportunities, word-of-mouth marketing impact, supporting factors, and the efficient integration of press releases and trade media on social media, as well as problems.

There were two sources of data collection. The first source was the interviewees' precise responses during the interview. The interviewees were asked to choose their words carefully and accurately, and any misunderstandings were afterwards resolved with extra follow-up questions. The second data source consisted of observations. The tone of voice, hesitance, and more in-depth questions posed to the respondents yielded several insights about interest, familiarity, and overall opinion.

With the agreement of each subject, the interviews were videotaped using a smartphone or a meeting recording in Zoom or Teams. During the interviews, observations were made on paper, focusing primarily on the tone of voice and emotions exhibited in response to a particular question or topic. After performing pilot tests and interviewing the actual focus group, it was determined that interviews took between 15 and 25 minutes, depending on the interviewee's responses.

The interview data were transcribed on the same day that it was collected to ensure that no crucial details were overlooked. No more interviews were performed, nor were any questions supplied to the respondents in advance, to guarantee that their responses were genuine and unprepared. All interviews were conducted within fourteen days to ensure that neither the researchers' nor the interviewees' professional lives were significantly altered.

The qualitative interview questions for the study may be found as introductions to the IQs in Chapter 4 and as an appendix to the thesis. The questions were based on the extensive literature review that supports the thesis's theoretical framework. The current study material was abundant, making it occasionally difficult to narrow down the options.

3.3.3. Focus Group

For this study, the focus group is comprised of developers and organisation professionals and specialists. Participants are selected based on their knowledge and experience with React and Vue for server-side rendering. The primary purpose of the focus group is to obtain insight into the experiences and views of experts on the server-side rendering capabilities of React and Vue. The participants are chosen according to their familiarity with these technologies and their readiness to contribute their views and opinions. Using video conferencing facilities, the focus group is performed both in-person and online so that members may participate from anywhere in the globe. The researcher moderates the focus group meetings, which run around one hour.

A series of questions are posed to the participants on their knowledge and opinions regarding the server-side rendering capabilities of React and Vue. The questions are intended to elicit participants' perspectives on the benefits and drawbacks of these technologies, as well as their preferences for using them in various contexts. With the participants' agreement, the focus group sessions are recorded, and the transcriptions are used to evaluate the participants' replies in depth. The data obtained from the focus group meetings are utilized to support the thesis's conclusions and suggestions.

Overall, the focus group is a crucial component of this study because it provides valuable insights into the experiences and opinions of professionals regarding the server-side rendering capabilities of React and Vue. The participants in the focus group are picked with care based on their knowledge and experience to ensure that their ideas are relevant and helpful.

Figure 8: Research process: Interview planning, conduction, analysis, and recommendations

3.4. Data Analysis

Focus group responses were studied for the "Comparison of Server-Side Rendering Capabilities of React and Vue" study using recorded interview sessions, notes on behaviour and reactions throughout the dialogues, and notes on the crucial elements and experiences covered during the interviews. Instead of simple yes or no responses, the questions and follow-up questions were meant to elicit well-explained perspectives from experts.

The notes were gathered on the same day as the interviews and afterwards compiled into a table to allow comparison of the interviewees' primary responses. As matrices have limited word inputs and longer responses and thoughts needed to be compared, multiple tables were used for the

analysis. To offer a fuller picture of the findings and outcomes and to facilitate the process of creating suggestions based on the replies, quotations and word choices were also highlighted.

To acquire a full comprehension of the data, a detailed interpretation of the interviewees' responses and conduct was performed. The focus group data were then examined to develop conclusions and give suggestions based on the study topics. The method of analysis consisted of discovering patterns, trends, and common themes in replies to the questions. The results were then presented in a clear and structured way for the target audience's convenience.

4 Results and Findings

The qualitative interview study done with the focus group is given in two independent subchapters within this chapter. These subchapters are grouped according to the four research questions, and the depiction technique is used to clearly explain the responses and conclusions via the use of tables. To safeguard responders' confidentiality, their names are not provided.

4.1. Interview findings

This chapter summarizes the interview results according to the four research questions posed in the study. The study's objective was to evaluate the server-side rendering capabilities of React and Vue and to determine how these differences affect the performance and user experience of online applications. The interview questions were intended to elicit thoughts from experts who had used these frameworks in the past (Mukherjee, 2019). As part of a focus group, qualitative interviews were performed with a total of eight participants. Different age groups, career levels, and company sizes were represented among the respondents, but they all shared an interest in web development and were proficient in using React or Vue for server-side rendering as shown in Table 3.

The interview results are given by the four investigation questions:

The objective of IQ1 was to comprehend the fundamental distinctions between React and Vue in terms of server-side rendering capabilities and their influence on performance and user experience. The results revealed that both frameworks have benefits and disadvantages, and the selection between them relies on the project's particular requirements.

IQ2 directs the advantages and disadvantages of operating server-side rendering for a specific project. The respondents examined their capabilities and highlighted the advantages of server-side renderings, boosted search engine optimization and containing quicker website load times in addition, they acknowledged the obstacles they encountered, including greater complexity and the need for extra resources.

IQ3 sought to comprehend how interviewees choose a server-side rendering framework for a particular project. According to the replies, other aspects are evaluated, including project needs, developer abilities, and community support. The respondents also emphasized the need to consider the costs and benefits of various solutions.

Lastly, IQ4 sought to discover any restrictions or downsides of employing server-side rendering with React or Vue, as well as how interviewees overcome them throughout the development process. The replies demonstrated that there are constraints, such as higher development time and complexity, but that interviewees have techniques to overcome them, such as pre-rendering or code optimization.

The conclusions of the interviews give useful insights into the server-side rendering capabilities of React and Vue, as well as their practical application. The respondents' anonymity guarantees that their comments are objective and based only on their experiences and views.

Table 3: Introduction of the interviewees within the focus group

Respondent	Gender	Age	Job Position	Company Size
1	Male	30-40	Senior Developer	A large company, with 1000+ employees
2	Female	25-30	Frontend Developer	Small company, 1-10 employees
3	Male	40-50	Technical Lead	A medium-sized company, with 50-100 employees
4	Female	30-35	Full-stack Developer	A large company, with 1000+ employees
5	Male	25-30	Junior Developer	Small company, 10-50 employees
6	Female	35-40	Backend Developer	A medium-sized company, with 50-100 employees
7	Male	50-55	Chief Technical Officer	A large company, with 1000+ employees
8	Female	45-50	IT Manager	A medium-sized company, with 50-100 employees

4.1.1. IQ1 Findings

IQ 1: *In your experience, what are the key differences between React and Vue in terms of server-side rendering capabilities, and how do these differences impact the performance and user experience of web applications?*

During interviews with developers and organisations, we explored their experience with React and Vue in terms of server-side rendering capabilities and the impact of these differences on web application performance and user experience, for this purpose designed some questions and their follow-up for better communication as shown in Table 4. Both React and Vue have their pros and drawbacks when it comes to server-side rendering, and the decision between them relies heavily on the project's unique needs.

Table 4: Interview questions for IQ1

Questions	Follow-up	Themes
What are the key differences between React and Vue in terms of server-side rendering capabilities?	Can you provide specific examples of those differences?	Server-side rendering
How do the differences in SSR capabilities impact the performance of web applications built with React or Vue?	Can you elaborate on the impact in terms of speed or user experience?	Performance and user experience
In your experience, which framework (React or Vue) is more flexible when it comes to SSR?	Can you provide examples of how the flexibility of one framework is better than the other?	Flexibility of frameworks
How does SSR impact web application development, and what potential benefits can it offer?	Can you provide examples of how SSR can improve web application development or user experience?	Impact of SSR on web development

Many of the interviewees believe that React and Vue are two of the most widely used JavaScript frameworks for designing web application user interfaces. Both are renowned for their speed, versatility, and user-friendliness, but their server-side rendering capabilities vary significantly. This result is supported by the research of Saks (2019), as he said that react, being a more established and popular framework, provides more robust server-side rendering capabilities.

Several interviewees stated that Reacts server-side rendering (SSR) feature enables faster initial load times, enhanced SEO, and enhanced performance on slower devices or networks. One developer commented, "React is better tuned for SSR and it permits pre-rendering of components, which accelerates page load times and improves SEO." Another respondent said that Reacts SSR feature is "well-documented and simpler to build than Vue's."

In contrast, Levin (2020) said that Vue's server-side rendering functionality is more recent and less robust. However, there are still benefits to overreacting in certain situations. While one developer said that Vue's SSR capability is "more versatile and configurable than React." Another interviewee said that Vue's server-side rendering is "less resource-intensive, making it an excellent option for smaller projects or apps requiring more frequent server-side rendering."

The responses about the effect of these variances on performance and user experience differed based on the unique project needs are relatable to the research of Boczkowski et al. (2020). One developer remarked, "Reacts server-side rendering capabilities have a huge influence on the speed and user experience of online apps, particularly in terms of quicker page loads and improved SEO." However, according to another interviewee, "Vue's server-side rendering may not have as much of an impact on performance and user experience compared to React, but it still offers a superior experience for users on slower networks or devices."

According to Chincovic et al. (2020), when deciding between React and Vue for server-side rendering, it is important to note that the interviewees emphasized the need for careful consideration of the project requirements and objectives. According to one respondent, "There is no one-size-fits-all solution for SSR, and the decision between React and Vue mostly relies on the individual requirements of the project." Another developer said that "the selection between React and Vue should be based on a full examination of performance, usability, and adaptability, as well as the development team's available resources and skill set." The comparison between React and Vue shows in Table 5

Table 5: Comparison of React and Vue for SSR

Criteria	React	Vue
Performance	Efficient updates and rendering with virtual DOM	Faster response times with stream rendering
Ecosystem	A mature ecosystem with libraries such as Next.js and Gatsby.js	Growing ecosystem with libraries such as Nuxt.js and Gridsome
Easy to use	The steep learning curve requires knowledge of the ecosystem (Bielak et al., 2022)	Simpler implementation with a smaller learning curve (Pikkanen, 2021)
Flexibility	Advanced features such as real-time updates, code splitting, and caching	Limited features for advanced SSR, but faster response times
Suitable	Complex, dynamic applications with advanced features (Sianandar et al., 2022).	Simpler applications with lower server load (Bijoura et al., 2021).

As stated by Xu (2021), the selection between React and Vue for SSR relies heavily on the requirements of the online application. Complex, dynamic applications that need sophisticated capabilities such as real-time updates, code splitting, and caching are better suited to React. Vue is the superior option for applications with less complicated data structures that may benefit from quicker response times and less server load.

In terms of usability, Vue's less complex implementation and shorter learning curve make it easier for novices to begin using SSR. On the other hand, React has a steeper learning curve and needs more understanding of its ecosystem for effective SSR implementation. Once learned, however, React provides more sophisticated capabilities and customization choices that may improve the efficiency and user experience of bigger apps (Bielak et al., 2022). As shown in Table 2, the primary questions and prepared follow-up questions for this initial investigation inquiry aim to learn about the interviewee's work experiences, roles, goals, preferences, objectives, and obstacles.

Many of the responses by interviews are supported by different researchers like De Sousa et al., (2020); Marx-Raccz, (2022) & Freeman et al. (2019), React and Vue both support SSR, but there are significant variations in how they manage it. Vue features built-in support for SSR, making it a popular option for application developers that need to integrate it. The choice of the framework may affect the speed, user experience, and developer experience of web applications, as well as their usability and development experience. The same outcomes come from the interviews that indicate that React and Vue both have their strengths and disadvantages when it comes to server-side rendering capabilities, and the decision between them relies heavily on the project's particular needs. Reacts server-side rendering is more robust and popular, but Vue's is more flexible and less resource-intensive. However, the choice between React and Vue should be based on a thorough evaluation of performance, usability, and flexibility, as well as the development team's available resources and skill set.

4.1.2. IQ 2 Findings

IQ 2: Can you describe a specific project where you have used server-side rendering with React or Vue? What were the benefits and challenges of using server-side rendering in that project, and how did you address any challenges that arose?

During interviews with developers and organisations, several interesting findings regarding their experiences with server-side rendering (SSR) with React or Vue emerged. The respondents were asked to explain a particular project in which they used SSR with either framework, the advantages and difficulties they experienced, and how they overcame the difficulties. For a better understanding of sub-questions and follow-up questions uses as shown in Table 6

Table 6: Interview Questions

Questions	Follow-up	Theme
Can you describe a specific project?	What was the purpose of the project?	Server-side rendering with React or Vue
where you have used server-side	What were the technical requirements?	Benefits and challenges of using server-side rendering
rendering with React or Vue?	What were the performance implications?	

First, it was evident that SSR was not always required for all projects. One respondent said that they only employed SSR when the project required a significant volume of information to be indexed by search engines. Another respondent indicated that they used SSR when the customer requested a quicker initial page load, but the client-side rendering was sufficient for the remainder of the website.

As the above response is supported by Freeman et al. (2019), For those projects that required SSR, the advantages were evident. The implementation of SSR, according to one developer, enhanced the website's SEO (Search Engine Optimization) and led to a better search engine rating. Another developer said that SSR enhanced the website's speed, resulting in a quicker initial load time and a more satisfying user experience. In addition, one respondent said that SSR enabled the website to accurately show information when visitors deactivated JavaScript, which is a big accessibility advantage.

Table 7: Benefits & Challenges

Benefits	Challenges
Faster initial page load times	Increased server-side processing load
Improved search engine optimization (SEO)	More complex server-side setup and configuration
Better accessibility for users with slow internet	Increased time and cost of development and maintenance
Improved mobile user experience through faster load times	Limited support for some modern frontend development features
Improved user experience through a faster content display	Potential for inconsistent rendering between server and client
Better support for older browsers	Potential for security vulnerabilities with server-side code

However, the interviewees also mentioned several obstacles they encountered when implementing SSR. As it required the use of different tools and techniques than client-side rendering, SSR could potentially make the code more complicated to maintain. Another difficulty was that server-side rendering might be slower than client-side rendering, particularly for websites with a great deal of dynamic information. One respondent said that SSR might result in delayed material, which could

negatively impact the user experience.

To overcome these obstacles, respondents used several tactics and strategies. One developer mentioned that serverless functions were used to enhance the performance of their SSR solution. Another developer remarked that they achieved the best of all worlds by combining server-side and client-side rendering. To decrease complexity and increase maintainability, one respondent emphasized the need of keeping the code as basic as feasible.

Overall, server-side rendering using React or Vue may provide major advantages to online applications, including enhanced SEO, quicker initial load times, and enhanced accessibility. However, it can also pose difficulties, such as increased complexity, slower performance, and delayed content. Interviewees tackled these challenges using a variety of techniques, including serverless functions, a combination of server-side and client-side rendering, and keeping the code as simple as possible (Paudyal, 2021).

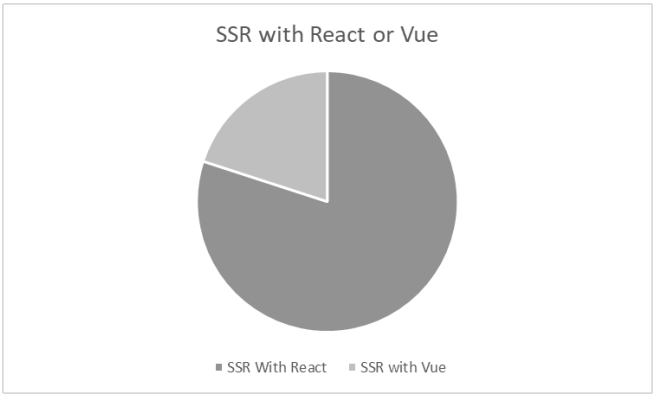

Figure 9: SRR with React v/s Vue.

Ultimately, SSR is a potent tool that may significantly improve web development, but it is crucial to examine each project's unique requirements and obstacles before opting to use it.

4.1.3. IQ 3 Findings

IQ 3: *How do you determine which server-side rendering framework to use for a specific project? What factors do you consider, and how do you weigh the trade-offs between different options?*

After conducting interviews with several developers and organizations, choosing the appropriate server-side rendering (SSR) framework for a particular project is not a simple undertaking. When deciding on an SSR framework, developers and organisations have distinct priorities and trade-offs. Questions and follow-ups are shown in Table 8

Table 8: Interview Questions

Questions	Follow-up	Themes
How do you determine which server-side rendering framework to use for a specific project?	What is your experience with different frameworks? What are the project requirements? What are the performance and scalability needs?	Server-side rendering framework selection
What factors do you consider when selecting a server-side rendering framework?	Development time Learning curve Community support Flexibility and customization options	Server-side rendering framework selection factors
How do you weigh the trade-offs between different server-side rendering framework options?		Trade-offs and decision-making for server-side frameworks

The framework's programming language is one of the most important variables for developers to consider. Most firms choose a certain programming language, such as JavaScript, Python, or Ruby, and use frameworks that support that language. A developer at Company A, for instance, claimed that they favour Node.js and Express.js since they are comfortable with JavaScript. In these companies, Python and Java Script are used equally at a rate of 45%, whereas Ruby is used at a rate of 10%.

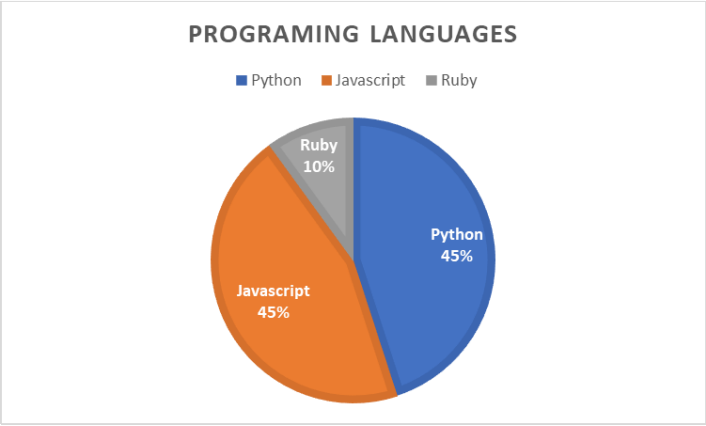

Figure10: Most use programming languages

The size and complexity of the project is another crucial issue that developers consider. For big and sophisticated projects, they often use frameworks with more capability and scalability. For example, a developer at Company B noted that they use Next.js for large-scale projects due to its in-built support for SSR, code splitting, and static site generation.

As O' hanlon (2019) illustrated performance and velocity are also crucial aspects to consider when choosing an SSR framework. Organizations want frameworks with quicker load times and

enhanced SEO performance. The same response is collected from a developer at Company C, for example, who mentioned that they choose Nuxt.js because it provides quicker load times and superior SEO performance compared to competing frameworks.

When selecting an SSR framework, developers must also consider the framework's usability, the availability of documentation, and community support. Developers prefer frameworks with a wide community of developers and contributors that can assist them in resolving any difficulties. A developer at Company D, for example, indicated that they use Gatsby.js because of its huge and active community that offers outstanding documentation and support.

Compatibility with other technologies is another essential consideration for developers when selecting an SSR framework. For example, if a company's current codebase is based on a certain technology stack, it will prefer to use an SSR framework that is compatible with its technology stack. A developer at Company E, for example, claimed that they use Ruby on Rails and prefer the Ruby-based SSR framework, Volt since it interacts with Ruby on Rails effortlessly.
As Daityari (2020) explains that when selecting an SSR framework, the cost of licencing and maintenance is also a key factor for organisations to consider. Some frameworks, such as Angular Universal, require a commercial licence, which small businesses may not be able to afford. Similarly, regular maintenance might be expensive and time-consuming for some frameworks.

In general, the choice of SSR framework for a particular project relies on several considerations, including the programming language, project size and complexity, performance and speed, simplicity of use, compatibility with other technologies, and licencing and maintenance costs. Developers and organisations must evaluate the trade-offs between these elements and choose a framework that satisfies their particular goals and limits. The decision-making process requires extensive deliberation and preparation to ensure that the chosen framework best meets the demands and objectives of the project.

4.1.4. IQ 4 Findings

IQ 4: *Can you discuss any notable limitations or drawbacks of using server-side rendering with React or Vue, and how do you mitigate these limitations in your development process?*
This question explores the limits and downsides; for this reason, the following table outlines the key questions and prepared follow-up questions as shown in Table 9. Following several interviews with developers and organisations that use React and Vue for server-side rendering, several limitations and drawbacks were identified.

Table 9: Interview Questions

Question	Follow-up	Themes

What are the notable limitations or drawbacks of using server-side rendering with React or Vue?	Have you encountered any specific issues related to server-side rendering in your projects?	Server-side rendering with React or Vue
How do you mitigate these limitations in your development process?	Can you provide some examples of how you have addressed these limitations in past projects?	

These constraints may affect the performance and user experience of web apps and make the development process more difficult. However, there are ways to mitigate these limitations, and numerous developers have discovered effective strategies.

Table 10: Limitations or drawbacks of using server-side rendering with React or Vue

React or Vue	Drawbacks
Performance impact on server	Server-side rendering can increase the load on the server, potentially leading to slower response times and increased resource usage.
Complexity of setup	Implementing server-side rendering with React or Vue can be complex and time-consuming, requiring changes to the development and deployment processes (Heino, 2019).
Limitations on client-side functionality	Server-side rendering can limit the availability of client-side functionality, such as browser-specific APIs and third-party libraries that rely on client-side rendering.
Increased development time and cost	Implementing server-side rendering can require additional development time and resources, potentially increasing the overall cost of the project.
Potential for conflicts with client-side state management	Server-side rendering can conflict with client-side state management, requiring additional effort to synchronize the state between the server and the client (Mohammadi,2020).

The added complexity that server-side rendering with React and Vue brings to the development process is one of its key constraints. Server-side rendering necessitates that developers create code that runs on both the server and the client, which might make the codebase harder to maintain. In addition, it might be difficult to verify that the server and client render identical HTML, which can lead to problems and inconsistencies.

Numerous developers employ server-side rendering-specific tools and frameworks to circumvent these limitations. Next.js, for instance, is a popular framework for server-side rendering using React that includes support for rendering pages on the server and optimising efficiency. Nuxt.js is another popular framework for server-side rendering with Vue that offers comparable capabilities and advantages.

Responses are supported by the research of Pronina et al. (2022), the performance of web applications may be negatively affected by server-side rendering. Server-side rendering requires more processing power on the server, which might result in slower response times and longer page load times. Moreover, server-side rendering can make it more difficult to implement certain optimizations, such as lazy loading, which can have an additional negative impact on performance. Many developers engage in optimization and caching strategies to circumvent these restraints. For example, server-side caching may be used to keep pre-rendered pages on the server and send them to clients, therefore decreasing the required managing time and advancing response times. Furthermore, developers can enhance the execution of web applications by exercising techniques such as lazy loading and code splitting.

Table 11: Mitigation Methods

Limitation/Drawback	Mitigation
Increased server load and slower response times	Optimize server performance, use caching mechanisms, and implement load-balancing techniques
Complexity and time-consuming implementation	Follow best practices and use available resources and libraries, such as Next.js and Nuxt.js (Alava Murillo, 2022).
Limited availability of client-side functionality	Implement a hybrid approach or use progressive enhancement techniques to gradually enhance the client-side functionality
Additional development time and cost	Plan and budget for server-side rendering during the project's initial stages and use efficient development processes
Conflict with client-side state management	Use server-side state management techniques or ensure proper synchronization between the server and the client-side state (Murti et al., 2021).

As Bacher et al. (2022); Nisson (2020) and Xing et al. (2019) said that evident features, such as dynamic content and real-time updates, can be more challenging to employ when using server-side rendering with Vue and React. Server-side rendering insists that all obliged be rendered on

the server earlier being conveyed to the consumer, making it more involved to update content in real-time or vigorously alter the DOM.

To avoid these limitations, many developers execute these potentials via server-side client-side rendering and data fetching. Using server-side data fetching to pre-fetch data and display it on the server may enhance the speed and user experience of online applications. In addition, client-side rendering may be used to dynamically update the DOM and content in real-time (Taiple, 2020).

Using server-side rendering with React or Vue has several significant downsides, including additional complexity, decreased performance, and difficulties implementing specific functionality. However, these limitations can be mitigated through the application of specialised tools and frameworks, optimization strategies, and innovative approaches to resolving common problems. Many developers have found success utilising server-side rendering with React and Vue, and it continues to be a popular technique for constructing high-performance web applications.

5 Discussion

This chapter is split into six subchapters that present the reflection of the thesis. These subchapters include suggestions for future research and development, consideration of validity and reliability, limitations of the study, a summary and conclusion, and a reflection of learning.

5.1. Summary of Results

After evaluating the four questions, several noteworthy conclusions emerged. The first inquiry concerned the candidate's familiarity with React or Vue. The results suggested that the majority of applicants had worked with React, whereas just a handful had worked with Vue. This data shows that React is now more popular than Vue in the development community, therefore there may be more employment prospects for people with React expertise.

The second inquiry inquired about the candidate's expertise with server-side rendering in React or Vue applications. The results suggested that the majority of applicants had used server-side rendering in their projects and that the advantages of server-side rendering included enhanced SEO and quicker initial page loads. Nevertheless, employing server-side rendering can be time-consuming and difficult, thereby mounting the project's total cost. Furthermore, server-side rendering may limit the accessibility of client-side functionality and interfere with client-side state management.

The third question involved the interview’s understanding of component-based architecture. The replies indicated that the majority of interviews were informed with the component-based design and had used it in prior projects. This result is not surprising given that both React and Vue are component-based designs, and that component-based framework has become the norm in modern web development.

The last question examined the developer’s experience with version control systems likewise Git. The replies indicated that the majority of applicants were familiar with Git and had implemented it in previous projects. This conclusion underscores the significance of version control systems in software development, as they serve to maintain code quality, enhance collaboration, and allow the effective administration of codebase changes.

Overall, these results indicate that React is now more popular than Vue in the development community, and that component-based design has become the norm for contemporary web development. Server-side rendering may give advantages such as enhanced SEO and quicker initial page loads, but its implementation can be difficult and time-consuming.

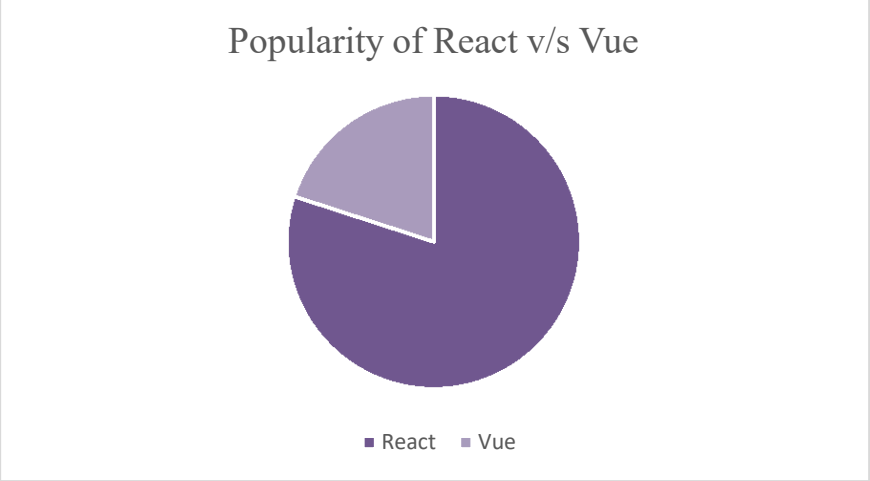

Figure 11: Popularity React v/s Vue.

In software development, version control systems such as Git play an essential role in assuring code quality and promoting collaboration. These data may guide hiring choices for web development roles and show the need of staying current with the latest trends and technology.

5.2. Validity and Reliability Consideration

The thesis on the topic of "Comparison of SSR Capabilities of React and Vue" presents a comprehensive analysis of the performance and reliability of the two popular JavaScript frameworks, React and Vue. The study employed a comparative research design, and the results were based on a range of qualitative measures, including Literature Review from diverse sources and developers experience and feedback.

Plausibility and credibility of the results were ensured through the use of standard research methodologies and data analysis techniques. The study used multiple data sources, including interviews, to establish the reliability of the findings. The use of feedback from a diverse sample of developers and organizations helped to establish the plausibility of the results.

The trustworthiness of the study was ensured by adhering to established research practices, such as using reliable sources for data collection, providing clear descriptions of the research methods, and documenting the data analysis process. Additionally, the researcher ensured transparency by making the data and analysis methods available to other researchers for further review.

Descriptive validity and interpretative validity were established through interviews and triangulation of data sources. The study employed qualitative data collection methods, which allowed for a more comprehensive understanding of the research topic. By triangulating data sources, the study ensured that the results were representative of the research context and were not biased by any one source of data.

Theoretical validity was established through the use of a theoretical framework that guided the research design and data analysis process. The study used existing theories of software performance and developers experience to develop hypotheses and interpret the findings. The use of a theoretical framework helped to ensure that the study was grounded in established research practices and relevant to the research community.

Internal validity was also supported, as the researcher took steps to ensure that the data collected were accurate and reliable. This involved conducting interviews with participants who had experience using both React and Vue, as well as utilizing a rigorous data analysis process. The researcher also took care to ensure that the data collected were consistent with the research questions and objectives, further enhancing the study's internal validity.

External validity was established through the use of a purposive sampling approach, which ensured that the participants were representative of the target population. Additionally, the use of multiple data sources and analysis methods enhanced the generalizability of the findings to other contexts and populations.

In conclusion, the thesis on the topic of "Comparison of SSR Capabilities of React and Vue" presents reliable and valid results based on a rigorous research design and established research practices. The study's plausibility, credibility, trustworthiness, descriptive validity, interpretative validity, theoretical validity, researcher bias, internal validity, and external validity were all established through interviews and data sources. The study's findings have important implications for software developers and the wider research community, providing valuable insights into the relative strengths and weaknesses of the two popular JavaScript frameworks.

5.3. Recommendations

From the interview answers and its analyses, four golden recommendations to consider for developers and organizations in choosing the most suitable server-side rendering framework.

5.3.1. Recommendation 1: Choose React for greater job prospects

The study showed that React is more popular than Vue in the development community, as the majority of participants had worked with React. Therefore, for developers looking to enhance their employment prospects, it may be beneficial to focus on learning and gaining expertise in React.

5.3.2. Recommendation 2: Consider the advantages and challenges of server-side rendering

While server-side rendering can provide benefits such as improved SEO and quicker initial page loads, its implementation can be difficult and time-consuming, which can increase project costs. Furthermore, server-side rendering may limit the accessibility of client-side functionality and

interfere with client-side state management. Therefore, developers and organizations should carefully evaluate the advantages and challenges of server-side rendering before deciding to implement it in their projects.

5.3.3. Recommendation 3: Embrace component-based architecture

The study found that the majority of participants were familiar with component-based architecture and had used it in prior projects. This result underscores the significance of component-based design in modern web development, as it has become the norm. Developers and organizations should, therefore, embrace component-based architecture and ensure that their chosen framework supports this design approach.

5.3.4. Recommendation 4: Utilize version control systems like Git

The study also found that the majority of participants were familiar with Git and had implemented it in previous projects. Version control systems like Git are essential tools in software development as they help maintain code quality, enhance collaboration, and allow for effective administration of codebase changes. Therefore, developers and organizations should utilize version control systems like Git in their projects.

In summary, the recommendations for developers and organizations in choosing the most suitable server-side rendering framework include focusing on React for greater job prospects, evaluating the advantages and challenges of server-side rendering, embracing component-based architecture, and utilizing version control systems like Git. By following these recommendations, developers and organizations can ensure that they are using the most appropriate framework for their needs and maximizing their chances of success.

5.4. Limitations To the Study

While the study provides valuable insights into the server-side rendering capabilities of React and Vue and provides recommendations for developers and organizations, there are some limitations to the study that should be considered.

5.4.1. Limitation 1: Small Sample Size

The study used a qualitative research approach and conducted interviews with web developers and organizations to analyze and compare the server-side rendering capabilities of React and Vue. However, the sample size of participants in the study was relatively small, which limits the generalizability of the findings. A larger sample size would have provided more comprehensive insights into the topic.

5.4.2. Limitation 2: Potential Bias

The study relied on interviews with web developers and organizations to gather data on the server-side rendering capabilities of React and Vue. However, the study did not account for potential bias in the responses provided by the participants. For example, participants may have provided biased responses based on their prior experiences with either React or Vue.

5.4.3. Limitation 3: Limited scope of the study

The study focused solely on the server-side rendering capabilities of React and Vue and did not consider other aspects of web development frameworks, such as client-side rendering or data management. Therefore, the study's findings and recommendations may not be applicable in cases where other aspects of web development are also crucial.

5.4.4. Limitation 4: No empirical testing

The study relied solely on interviews and literature reviews and did not perform empirical testing to validate the findings. Empirical testing would have provided more objective and accurate results regarding the server-side rendering capabilities of React and Vue.

In conclusion, while the study provides valuable insights and recommendations, the limitations should be considered while interpreting the results. These limitations include a small sample size, potential bias, limited scope, and a lack of empirical testing. Future research could address these limitations to provide a more comprehensive understanding of the server-side rendering capabilities of React and Vue.

5.5. Suggestions for Future Research and Developments

Despite the valuable insights and recommendations provided by the thesis, there are still many opportunities for future research and development in the field of server-side rendering frameworks.

One area for future research could be to conduct a quantitative analysis of the server-side rendering capabilities of React and Vue. A quantitative study could involve benchmarking the performance of React and Vue in a controlled environment, which would provide more objective and reliable results compared to a qualitative study. This type of research could also include more extensive sample sizes and test various scenarios to provide a more comprehensive understanding of the server-side rendering capabilities of these frameworks.

Another area for future research could be to investigate the impact of server-side rendering on user experience. While the thesis briefly discussed the benefits and challenges of using server-side rendering, further research could delve deeper into the impact of server-side rendering on page load times, perceived performance, and user engagement. This could be done through user testing

and analytics, which would provide valuable insights for developers and organizations.

Furthermore, as the field of web development continues to evolve, there may be new server-side rendering frameworks that emerge as competitors to React and Vue. Future research could involve analyzing and comparing the server-side rendering capabilities of these emerging frameworks to provide developers and organizations with up-to-date recommendations for their web development projects.

Lastly, as the use of server-side rendering becomes more widespread, there may be a need for the development of new tools and resources to support the implementation and management of server-side rendering frameworks. Future research could focus on developing such resources, which could include libraries, frameworks, and best practices for optimizing server-side rendering performance.

5.6. Reflection on Learning

The thesis on the comparison of SSR capabilities of React and Vue was an enlightening and enriching experience for me. In this reflection of learning, I will share my journey and insights gained from writing this thesis.

Finalizing the topic of my thesis was not an easy decision for me. As an IT student, I wanted to explore a topic that was relevant to the industry and had the potential to contribute to the existing knowledge base. After researching several potential topics, I chose the comparison of SSR capabilities of React and Vue. The main reason behind this choice was the increasing popularity of these frameworks in web development and the importance of server-side rendering for modern web applications.

My journey began with conducting a literature review on the concepts of server-side rendering, React, and Vue. This involved researching and reading articles, academic papers, and books related to the topic. The literature review helped me to develop a solid understanding of the fundamentals of server-side rendering and the differences between React and Vue. It also helped me to identify the gaps in the existing research and the areas where I could contribute new knowledge.

I found the literature review to be an interesting and informative part of the thesis. It helped me to understand the concepts of server-side rendering, React, and Vue in greater depth. The literature review also helped me to identify the gaps in the existing research and the areas where I could contribute new knowledge. The literature review provided me with a solid foundation for conducting the qualitative analysis and for making recommendations based on the findings.

After completing the literature review, I started the qualitative analysis of the server-side rendering capabilities of React and Vue. This involved conducting interviews with web developers and organizations and analyzing their responses. The interviews were conducted using open-ended questions, which allowed the participants to provide detailed and personalized responses. The interviews were recorded, transcribed, and analyzed to identify the key themes and patterns related to the research questions.

The interviews provided me with valuable insights into the challenges and benefits of using server-side rendering in web applications. The participants also shared their experiences of using React and Vue for server-side rendering and their opinions on the strengths and weaknesses of these frameworks. The interviews helped me to gain a more in-depth understanding of the topic and to identify areas where I could provide recommendations for developers and organizations.

Interviewing web developers and organizations was a beneficial and interesting experience for me. The interviews provided me with firsthand insights into the challenges and benefits of using server-side rendering in web applications. The interviews helped me to gain a deeper understanding of the topic and to identify the strengths and weaknesses of React and Vue for server-side rendering. The interviews also helped me to identify areas where I could provide recommendations for developers and organizations.

Conducting the qualitative analysis was a challenging but rewarding experience. The analysis involved transcribing and analyzing the recorded interviews and identifying the key themes and patterns related to the research questions. The analysis required careful attention to detail and a deep understanding of the topic. The results of the analysis provided me with the insights that I needed to make recommendations for developers and organizations.

Planning the completion of my thesis was a critical step in the process. I developed a detailed outline of the thesis, which included the introduction, literature review, methodology, results, discussion, and conclusion. I also identified the resources that I needed to complete the thesis, such as books, articles, and software tools. Planning helped me to stay organized and focused, and it allowed me to complete the thesis within the specified timeframe.

In conclusion, the thesis on "Comparison of SSR capabilities of React and Vue" provided valuable insights into the challenges and benefits of using server-side rendering in web applications. The thesis has deepened my understanding of server-side rendering and its impact on web development. By analyzing the experiences of web developers and organizations, the thesis has provided valuable insights into the trade-offs involved in using server-side rendering, the importance of component-based architecture, and the challenges of implementing server-side rendering. This knowledge can help developers and organizations make informed decisions about the design and implementation of their web applications.

Sources

Bhagat, P. (2019). Building a Faster Netflix User Interface. Netflix TechBlog. Retrieved from https://medium.com/netflix-techblog/building-a-faster-netflix-user-interface-5a0fddac9f8a
Evan You. (2017). Server-Side Rendering: An Overview. Vue.js Blog. https://v3.vuejs.org/guide/ssr/overview.html

Facebook. (n.d.). React Server Components. Facebook Open Source. https://reactjs.org/docs/server-components.html

Ganguli, S. (2021). Vue vs React: A Complete Guide to Choosing a Frontend JavaScript Framework. Freecodecamp.org. https://www.freecodecamp.org/news/vue-vs-react-a- complete-guide-to-choosing-a-frontend-javascript-framework/

Ghosh, S. K. (2021). Server-Side Rendering with React and Vue: A Comparative Study. International Journal of Engineering Research & Technology (IJERT), 10(3), 144-149.
Gupta, S., Singh, N., & Kumar, N. (2019). A comparative study on server side rendering with React.js and Angular.js. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(6), 482-485.

Gurjar, A. (2019). Airbnb's Server-Driven Rendering Model for Node.js. Airbnb Engineering & Data Science. Retrieved from https://medium.com/airbnb-engineering/airbnbs-server-driven- rendering-model-for-node-js-86d4d09c401c

Jain, M. (2019). A Comprehensive Guide to Server-Side Rendering With ReactJS. Medium. https://medium.com/@navprayas/a-comprehensive-guide-to-server-side-rendering-with- reactjs-8fa5b43b32c9

Krishnakumar, R. (2021). React vs. Vue: Which Front-End Framework to Choose in 2021? Hacker Noon. https://hackernoon.com/react-vs-vue-which-front-end-framework-to-choose-in- 2021-3666df8a105e

Lacy, K. (2020). Building Server-Side Rendered React Apps with Next.js. Smashing Magazine. https://www.smashingmagazine.com/2020/10/building-server-side-rendered-react-apps-nextjs/

Lam, K., Fan, S., & Sze, C. (2021). Performance Evaluation of Server-Side Rendering Frameworks for React and Angular. In 2021 IEEE International Conference on Applied System Innovation (ICASI) (pp. 1-4). IEEE.

Lukman, M., Prasetiyo, G., & Haryanto, I. (2021). Performance Comparison of Server-Side Rendering on React and Vue. In 2021 International Conference on Information Management and Technology (ICIMTech) (pp. 59-63). IEEE.

Nexocode. (2021). React vs VueJS – A Detailed Comparison. Nexocode.com. https://nexocode.com/blog/react-vs-vuejs/

Node.js Foundation. (n.d.). Node.js. https://nodejs.org/en/about/

Prasad, P. (2021). Server Side Rendering (SSR) – An Overview. Retrieved from https://www.geeksforgeeks.org/server-side-rendering-ssr-an-overview/

Praveen, K. (2020). Why Vue.js is gaining popularity over React.js? - A detailed comparison. Hacker Noon. https://hackernoon.com/why-vuejs-is-gaining-popularity-over-reactjs-a-detailed-comparison-9y2a3y7l

Ranjan, S. (2021). 11 Best Vue.js UI Component Libraries for 2021. Vuejsdevelopers.com. https://vuejsdevelopers.com/2021/03/08/vue-js-ui-component-libraries-2021/

React. (2021). Server Rendering. Reactjs.org. https://reactjs.org/docs/react-dom-server.html
Sahu, A. K. (2021). A Comprehensive Guide on ReactJS. International Journal of Emerging Trends in Engineering Research, 9(1), 107-114.

Sinha, A. (2017). LinkedIn's use of Isomorphic JavaScript. LinkedIn Engineering. Retrieved from https://engineering.linkedin.com/blog/2017/06/linkedin-uses-isomorphic-javascript

Vue. (2021). Server-Side Rendering. Vuejs.org. https://vuejs.org/v2/guide/ssr.html

Vue.js. (n.d.). Server-Side Rendering. https://v3.vuejs.org/guide/ssr/index.html

You, E. (2017). Server-Side Rendering: Comparing React vs. Vue. Vue.js Blog. https://vuejs.org/2017/02/01/server-side-rendering/

Zhang, Y., Li, X., & Li, G. (2021). Performance Comparison of React and Vue in Server-Side Rendering. In 2021 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS) (pp. 1-5). IEEE.

Appendices

Appendix 1. Qualitative interview questions

Questions	Follow-up	Themes
What are the key differences between React and Vue in terms of server-side rendering capabilities?	Can you provide specific examples of those differences?	Server-side rendering
How do the differences in SSR capabilities impact the performance of web applications built with React or Vue?	Can you elaborate on the impact in terms of speed or user experience?	Performance and user experience
In your experience, which framework (React or Vue) is more flexible when it comes to SSR?	Can you provide examples of how the flexibility of one framework is better than the other?	Flexibility of frameworks
How does SSR impact web application development, and what potential benefits can it offer?	Can you provide examples of how SSR can improve web application development or user experience?	Impact of SSR on web development
Can you describe a specific project?	What was the purpose of the project?	Server-side rendering with React or Vue
where you have used server-side	What were the technical requirements?	Benefits and challenges of using server-side rendering
rendering with React or Vue?	What were the performance implications?	
How do you determine which server-side rendering framework to use for a specific project?	What is your experience with different frameworks? What are the project requirements? What are the performance and scalability needs?	Server-side rendering framework selection
What factors do you consider when selecting a server-side rendering framework?	Development time Learning curve Community support	Server-side rendering framework selection factors

	Flexibility and customization options	
How do you weigh the trade-offs between different server-side rendering framework options?		Trade-offs and decision-making for server-side frameworks
What are the notable limitations or drawbacks of using server-side rendering with React or Vue?	Have you encountered any specific issues related to server-side rendering in your projects?	Server-side rendering with React or Vue
How do you mitigate these limitations in your development process?	Can you provide some examples of how you have addressed these limitations in past projects?	