Bachelor's Thesis

Information and Communications Technology

2025

Pham Duy Quang

A Comparative Study of React and Vue.js in Single Page Website (SPW) Development

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Degree programme in Information and Communications Technology

2025 | 45 pages

Pham Duy Quang

A Comparative Study of React and Vue.js in Single Page Website (SPW) Development

This thesis has researched and compared the practical applications of React and Vue, which are quite popular and widely used frameworks today. An experimental single-page website project has been implemented and developed in both frameworks with the same core functions, including product listing, search, filter, sort, rank, wishlist model. The method used in the experiment was to develop two versions of React and Vue in parallel to gain multi-dimensional perspectives and evaluations of both frameworks. Both versions applied component-based design principles. Finally, each version was evaluated by the author of the thesis through Google Lighthouse. Each version was evaluated based on performance, code complexity, author experience, and community support. The results of the thesis showed that Vue.js had a superior score to React in terms of metrics such as performance and especially loading speed through Google Lighthouse evaluation criteria. However, besides that, React offered a more diverse ecosystem and greater flexibility in logic and expansion. In conclusion of this thesis, Vue.js provided a faster and easier development experience than React, which required more time and experience to master. The information that has been compared and analyzed in the thesis has provided developers and development teams with a comprehensive view as well as a practical understanding to be able to make the most appropriate and correct choices for different SPW development needs.

Keywords:

React, Vue.js, Single Page Website (SPW), JavaScript Frameworks, Front-End Development, Performance Comparison, State Management, Routing.

Contents

Figures

Tables

List of abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Transfer Protocol
JSX	JavaScript XML
SPW(s)	Single Page Website(s)
SEO	Search Engine Optimization
XML	eXtensible Markup Language

1 Introduction

1.1 Background and Context

Over the years of web development and building websites. With the advent of countless new technologies and JavaScript frameworks to support and improve the quality of websites. Above all, the two front-end frameworks React and Vue.js are still the most prominent, especially in developing and building Single Page Websites (SPWs) - this is a method of developing a website in which most of the content is loaded dynamically without refreshing the entire page. SPWs will allow web applications to load a single HTML document and update the content dynamically, thereby bringing a better user experience.

With the rapid growth of Single Page Websites (SPWs), they have become popular and widely used for websites such as portfolios, business landing pages, blogs, and many interactive contents. Therefore, developers often face the challenge of choosing the right JavaScript framework. Among them, React and Vue.js are the two names that are often mentioned. Both frameworks provide developers with solutions and utilities to create high-performance and interactive SPWs, but they also have differences in the learning curve, development experience as well as accompanying ecosystem support. Technically, React is not a framework but it is, more precisely, a third-party library for rendering UI components. React is developed by Meta (formerly Facebook) and is used in combination with other libraries to create complete applications. On the other hand, Vue.js is a JavaScript framework used to build user interfaces. It is mainly based on modern and standard JavaScript. Vue.js was created by Evan You (MDN Web Docs, n.d.b). Therefore, choosing the right framework for a project is very important.

This thesis will conduct comparisons between React and Vue.js frameworks to provide different perspectives and aspects to help developers make the right choice in using the framework in their projects. In this thesis, a simple Single Page Website will be implemented identically by the two frameworks, React and

Vue.js, to compare aspects such as performance, complexity, state management, and ecosystem support.

1.2 Problem Statement

With the popularity and widespread of the two frameworks React and Vue.js, there have been many comparisons and discussions on choosing the right framework to implement for a Single Page Website project to suit the needs, performance, and flexibility, but few comparisons and provide a comparison based on a real application, website built similarly.

In this thesis paper, the research questions that need to be addressed are:

- Which framework is better and provides better performance for SPW?
- Which framework is easier to learn, use, and approach for developers?
- How will the issues of state management and routing be compared?
- What is the role of community support and ecosystem of the two frameworks in long-term projects?

After answering the above questions, this thesis will provide a clear comparison between React and Vue.js based on the author's real-life development experience.

1.3 Research Objectives

This thesis will focus and target the following key points:

- Provide a clear comparison between React and Vue.js in the context of the rapidly growing and constantly changing Single Page Website.
- Provide discussions and evaluations of the two frameworks based on performance, ease of access and use, state management, and sustainability in community support of both frameworks.

- Author implement simple, sample projects using both React and Vue.js frameworks to provide practical, hands-on comparison and clear comparison perspectives.
- Research and give recommendations and advice for developers to choose between React and Vue.js for suitable SPW projects.

1.4 Structure of the Thesis

The thesis is structured as follows:

Chapter 2: Literature Review focuss on discussing the fundamentals of the two frameworks React and Vue.js as well as other frameworks, Single Page Website, and previous comparisons of React and Vue.js

Chapter 3: Methodology provides an outline of the thesis, the scope of the project, tools, and software used in the thesis, and the metrics used for comparison.

Chapter 4: Implementation and Functional Analysis provides detailed information on the process of practicing and developing a simple SPW using React and Vue.js. This chapter also includes the challenges encountered and the solutions implemented, providing real-life experiences when implementing a similar SPW with two different frameworks.

Chapter 5: Evaluation and Results presents the thesis results, performance comparison, complexity analysis, and community support comparison of the two frameworks.

Chapter 6: Discussion interprets the main findings, highlights the implications for developers, and identifies the limitations of the thesis and the author of this thesis.

Chapter 7: Conclusion and Future Work summarizes the thesis, and provides conclusions, recommendations, and suggestions for developers, as well as new directions for future.

2 Literature Review

This chapter will focus on discussing the fundamentals of the two frameworks React and Vue.js as well as other frameworks, Single Page Website, and previous comparisons of React and Vue.js.

2.1 Overview of JavaScript Frameworks for Front-End Development

Web development has been revolutionized and developed by leaps and bounds with the advent of JavaScript frameworks. These frameworks, when launched, have provided developers, programmers, and investors with an extremely useful and superior tool for building and creating dynamic, responsive and diverse websites and applications. In recent years, the transition from traditionally rendered websites to frameworks has changed the way users approach and interact with applications and the web. Additionally, two front-end frameworks that have been growing strongly and widely used in recent years are React and Vue.js, which are often the top two choices of developers for projects to deploy a Single Page Website, because these two frameworks always offer and provide component-based architecture, in that applications are built with reusable components, and most of all, there will always be a strong and sustainable support ecosystem. Meanwhile, when we look at another framework called Angular, this framework often offers subjective approaches along with a somewhat rigid structure, therefore React and Vue.js will always be more flexible (MDN Web Docs, n.d.a).

2.2 Introduction to Single Page Websites (SPWs)

Along with the strong development of modern JavaScript frameworks in recent years, the concepts and theories of Single Page Websites (SPW) are no longer strange to developers. Moreover, it is growing strongly and gradually becoming popular and widely used in the web development industry, when investors and businesses are looking for and deploying faster websites, saving time and

costs, more interactive, and more user-friendly. An SPW will be able to load all content in a single HTML page, thereby helping to reduce page loading time and improve user experience. Regarding current applications of SPW, it is widely applied in the deployment and development of websites such as portfolios, landing pages, blogs, event websites, and small businesses with limited services. Where the continuity and seamlessness of user experience are very important and need to be cared for and improved. However, along with these benefits, SPW also poses significant challenges in managing initial load status, routing, and performance at project initiation, requiring effective solutions to mitigate these issues (Pretorius, 2024).

2.3 Key Features of React (React Documentation, n.d.)

Developed and published by Meta (formerly Facebook), React is known as a component-based JavaScript library used to build fast and optimized user interfaces, designed around a virtual DOM. Key features will include:

- Reusable components - They can make project maintenance and scaling or project changes easier and more flexible, bringing many benefits and time savings for developers.
- Efficient rendering - As mentioned earlier, React is designed around a virtual DOM, so it can improve performance.
- React has a strong ecosystem - It is supported by many libraries and tools as well as a large React user community.
- React Hooks - This is a feature introduced by React to provide a functional approach to state and lifecycle management without requiring class components.
- JSX Syntax - React can combine JavaScript and HTML to get a more readable and accessible structure.

Some of the real-world applications of React are big companies like Netflix, Airbnb, and Uber, which have widely used this framework in their programs. However, the approach and learning can be challenging for beginners, especially when starting to work and get in touch with advanced management tools like Redux or Context API.

2.4 Key Features of Vue.js (Vue.js Documentation, n.d.)

Vue.js was created and developed by Evan You and is widely known as a progressive, lightweight, and accessible framework for developers and programmers. While React uses JSX syntax, Vue.js uses a more HTML-friendly template syntax, making it easier for developers familiar with traditional HTML and CSS to approach, learn, and apply. Key features will include:

- Easy to learn and approach - Since Vue.js uses template-based syntax that is quite similar to HTML, it is easier for beginners to use.
- Lightweight and flexible - This is a framework that can be used as a complete framework or slowly integrated into an existing project.
- Powerful state management and two-way data binding - Vue.js has state management libraries like Vuex and Pinia that provide good and efficient state management solutions. Vue.js also helps synchronize UI data and logic seamlessly and cleanly.

In real-world projects, Vue.js is favored by developers and applied to small and medium-sized projects due to the flexibility of the framework. However, Vue.js does not have a large user community like React, so there may be some situations where there are fewer third-party integrations.

2.5 Comparison of React and Vue.js in Previous Studies

Numerous studies have compared aspects of React with Vue.js, including their ease of use, complexity, and performance. According to analysis, both frameworks provide a strong solution for developing one-page applications; yet,

the market has seen distinct developments due to variations in architecture, developers, and ecosystem expertise (Emmanni, 2023; Tadesse, 2023).

Important findings from earlier research:

In terms of performance, React makes use of virtual-Domage, which usually offers quick UI updates for large-scale apps and contributes to scalable applications' excellent performance. On the other hand, Vue.js has demonstrated efficacy in small and medium-sized applications and is intended to maximize component results through a comprehensive feedback system (Monterail, 2024).

Vue.js is nearly as simple to use as HTML and CSS due to its model-based syntax. React has a steeper learning curve for managing the lifespan of states and components, and users must be familiar with JSX, despite having higher flexibility and modularity (Monterail, 2024).

Community & Ecosystem: React and Vue.js both enjoy huge community support, but their ecosystems differ in size and engagement. With over 230,000 stars on GitHub and 481,000 questions on Stack Overflow, React has proven to be popular and supported by companies, most notably Meta (formerly Facebook). React also benefits from frameworks like Next.js and Gatsby, which increase server visibility and scalability, making React an ideal choice for large applications. In contrast, Vue.js’s community, while smaller, remains engaged and developer-friendly, with 208,000 stars on GitHub and around 108,000 questions on Stack Overflow. Vue's ecosystem is well-established, including utilities such as Vue Router for navigation and Vuex or Pinia for state management, decreasing the need for third-party solutions. The community is actively participating through forums, GitHub, and social media channels, making it easy to seek assistance and cooperate. While React's extensive ecosystem provides flexibility and extensibility, Vue.js prioritizes simplicity and accessibility, making both frameworks good alternatives for SPW development (Monterail, 2024). (Refer to Table 1)

Table 1. Vue.js vs React Overview (Monterail, 2024).

Feature	Vue.js	React
Virtual DOM	✓	✓
Virtual Component-Based UI Development DOM	✓	✓
Official Component Library for Mobile Apps	✓ (NativeScript)	✓(React Native)
Open Source	✓	✓
Creator	Evan You	Meta(Facebook)/Jordan Walke
Initial Release	2014	2013
Written In	JavaScript	JavaScript
Syntax	HTML(default), JSX	JSX
Live Websites (2024)	3 million	12 million
Top Features	- Elegant programming style and patterns - Easy learning curve - Thorough documentation	- Elegant programming style and patterns - Rich package ecosystem - Widespread usage
Popular Applications	Behance, Google, Facebook, Wizz Air, Nintendo, Upwork, Alibaba, Vice, Trustpilot, Netflix, Euronews	Atlassian, Instagram, Airbnb, Pinterest, Netflix, Dropbox, Uber, Reddit
License	MIT	MIT

3 Methodology

Methodology - Provides an outline of the thesis, the scope of the project, tools, and software used in the thesis, and the metrics used for comparison.

3.1 Research Design

The research methodology in this thesis is based on a parallel experimental comparison model of react and vue.js in a pilot project.

- Step 1: React and Vue.js Research, Analysis of key features.
- Step 2: I will create a website (SPW) with React and a similar version with Vue.js.
- Step 3: Evaluate key factors such as performance, development complexity, state management system, and community support.
- Step 4: Summarize data, Analyze results, Objective comments.

3.2 Project Scope and Use Cases

A test project is a simple web page (SPW) with the following common features:

- Display a list of data generated with dummy data.
- Add/remove functions to the list (minimal, no backend).
- Use local storage or state management to save data state in your browser.
- Optimize loading speed for performance and page testing.

This project is useful for reviewing React and Vue.js data processing, optimizing interfaces, managing state, and evaluating the suitability of each framework.

3.3 Metrics for Comparison:

To objectively evaluate React and Vue.js, this thesis used four main criteria:

3.3.1 Performance

Performance for each framework is based on

- Side Load Time (page load time) - Using Google Lighthouse Tool, we can use First Contentful Paint (FCP) measures the time it takes for a browser to render the first piece of content after a user navigates to a webpage, and Largest Contentful Paint (LCP) when the user loads the main content of the page, LCP is the measurement of the time it takes for the largest content element in the viewport to load. and other metrics (Google, 2024; Chrome Developers, 2024).
- DOM Performance (DOM Performance) - Check the speed of updating when data changes (Alter Solutions, 2024).

3.3.2 Complexity (Easy or Hard to Develop)

React vs Vue.JS Development complexity is based on:

- Project Structure - Compare the architecture of React (JSX) and Vue.js (template-based syntax)
- Ease of learning - based on the popularity of educational documentation and frameworks.
- Development Time (Development Time) - Evaluate the time to complete the basic functions of all frameworks.

3.3.3 State Management Routing

- React and VUE.JS State management and routing (routing) are analyzed effectively by
- Redux (React) vs. VuEx/Pinia (Vue .JS). How does a react router work compared to a Vue router? (Hasan, 2023)
- Data update speed and scalability on the front-end.

3.3.4 Community Support and Ecosystem (Monterail, 2024)

Community and ecosystem support for each framework is based on:

- The number of learning resources. (docs, blogs, video tutorials).
- Number of supported libraries.
- Developers on GitHub, and Stack Overflow.

3.3.5 Tools and Technologies Used

To conduct this thesis using tools and technologies, the following tools and technologies are used:

- React.js (React Router, Redux) - Create SPW instances in React.
- Vue.js (Vue Router, Pinia) - Create SPW instances using Vue.JS
- Google Lighthouse, WebPagetest - Website testing for the website.
- GIT and GITHUB - Source control and tracking process.

4 Implementation and Functional Analysis

4.1 Project Overview: Use Case Description

The project in the thesis paper is built and designed as a simple Single Page Website (SPW) with the main purpose of performing the following comparisons in the thesis paper. The website is completely new and similar in core functions to both frameworks, React and Vue. The website is implemented as a website displaying information of electronic products, including the following functions: searching for products, filtering by different categories, filtering by product ratings/reviews, sorting by price, viewing product details via a modal, and adding/removing products from the Wishlist.

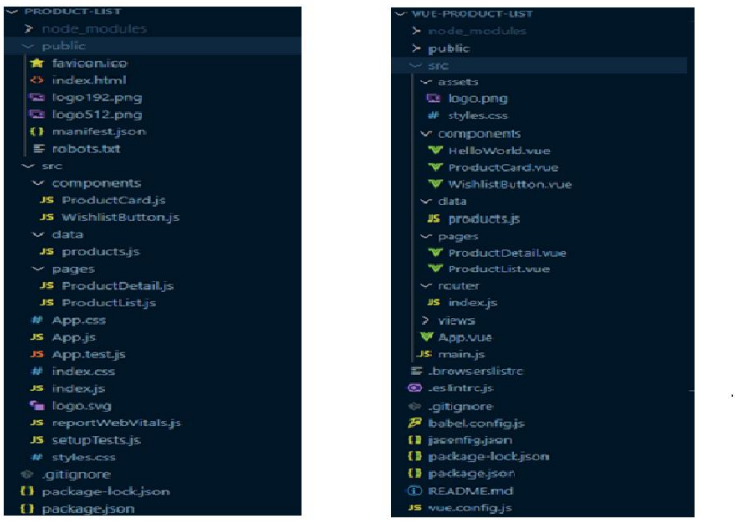
The main features of the project include: (Refer to Table 2 and Figure 1)

- Displaying a list of products with images, prices, categories, descriptions, and star ratings.
- Searching for products by name.
- Filtering products by category.
- Filtering products by star ratings.
- Sorting by price ascending/descending.
- View product details via a modal without leaving the main page.
- Add/remove products from wishlist.
- Save wishlist using Local Storage.

Furthermore, through this project, the author of this thesis also gained more knowledge and learned more about Vue.js, which the author of this thesis had not been exposed to much before.

Table 2. Table of technologies used in the project.

Stack	React Version	Vue Version
Framework	React (18.x)	Vue 3
CSS	styles.css	styles.css (reuse from React)
Data handling	useState, props	reactive, computed, props
Routing	react-router-dom	vue-router
State	useState, LocalStorage	reactive, LocalStorage
Structure	Component-based	Component-based



React Version Vue Version

Figure 1. Illustration: Project folder structure for both version React and Vue.

4.2 React Implementation:

This is the part where the React version of the project is developed and deployed. The author of this thesis chose to implement the React version before because this is the framework that the author of this thesis has learned and practiced throughout the learning process. From there, it is easier to approach both the mindset and the way of building code. Then, it is possible to implement the Vue version better and faster because, for the author of this thesis, Vue is a newer framework that needs time to learn.

4.2.1 Key Features Used

The project with the React version is developed based on the component-based structure of React, using React hooks to manage state easily and clearly, and also combining with Local Storage to be able to store state from Wishlist. The author of this thesis used useState for state management, handling side effects with useEffect. And the project is initialized with a command familiar to developers working with React: **npx create-react-app** PRODUCT-LIST. Below are the main features of the deployed React version.

Component-Based Architecture.

The project is divided into separate components to help the source code be clean, easy to maintain, and easy to extend and reuse. The structure is divided as follows:

- ProductList.js: The main page displays the product list.
- ProductDetail.js: displays product details in the modal.
- ProductCard.js: the child component representing each product.
- WishlistButton.js: handles add/delete wishlist operations.

Handling Search, Filter & Sort in the project: Project features like filtering by category, searching by name, sorting by price, and sorting by rating are handled via useState and **.filter()** / **.sort()** in React. (Figure 2)

```
const filteredProducts = products
  .filter((product) =>
    product.name.toLowerCase().includes(searchTerm.toLowerCase())
  )
  .filter((product) => category === "All" || product.category === category)
  .filter((product) => ratingFilter === 0 || rating[product.id] >= ratingFilter)
  .sort((a, b) => {
    if (sortOrder === "asc") return a.price - b.price;
    if (sortOrder === "desc") return b.price - a.price;
    return 0;
  });
```

Figure 2. Filtering, Searching, and Sorting Logic in ProductList.js.

Wishlist Modal with LocalStorage: Products can be added or removed from the Wishlist, The Wishlist will be stored in LocalStorage to avoid website reload, which is to keep the nature of a Single Page Website. The Wishlist, instead of using a route,r is replaced and displayed in a separate modal, and users can add and remove products from the Wishlist easily. (Figure 3)

```
useEffect(() => {
  const storedWishlist = JSON.parse(localStorage.getItem("wishlist")) || [];
  setWishlist(storedWishlist);
}, []);

useEffect(() => {
  localStorage.setItem("wishlist", JSON.stringify(wishlist));
}, [wishlist]);

useEffect(() => {
  localStorage.setItem("rating", JSON.stringify(rating));
}, [rating]);

const toggleWishlist = (id) => {
  if (wishlist.includes(id)) {
    setWishlist(wishlist.filter((item) => item !== id));
  } else {
    setWishlist([...wishlist, id]);
  }
};
```

Figure 3. Wishlist Logic with useEffect and LocalStorage.

Product Detail as Modal: The details of the product will be designed and built in the form of a modal, this was changed by the developer during the development process because he realized that if using a router, the website would reload the page and use a new page, which according to the author of this thesis would lose the nature of a Single Page Website, so he decided to use a modal to ensure that it was still a Single Page Website as originally planned for the project and thesis. (Figure 4)

```
      {isDetailOpen && selectedProduct && (
        <div className="wishlist-modal">
          <div className="wishlist-content">
            <span className="close-btn" onClick={() => setIsDetailOpen(false)}>❌</span>
            <img src={selectedProduct.image} alt={selectedProduct.name} className="product-detail-image" />
            <h2>{selectedProduct.name}</h2>
            <p className="category">Category: {selectedProduct.category}</p>
            <p className="price">${selectedProduct.price}</p>

            <p className="description">{selectedProduct.description || "No description available."}</p>

            <div className="rating">
              <label>Rating: </label>
              {[1, 2, 3, 4, 5].map((star) => (
                <span
                  key={star}
                  className={`star ${rating[selectedProduct.id] >= star ? "filled" : ""}`}
                  onClick={() => handleRatingChange(selectedProduct.id, star)}
                >
                  ★
                </span>
              ))}
            </div>

            <button
              className={`wishlist-btn ${wishlist.includes(selectedProduct.id) ? "added" : ""}`}
              onClick={() => toggleWishlist(selectedProduct.id)}
            >
              {wishlist.includes(selectedProduct.id) ? "❤️ Remove from Wishlist" : "🤍 Add to Wishlist"}
            </button>
          </div>
        </div>
      )}
    </div>
  );
}

export default ProductList;
```

Figure 4. Product Detail Modal Implementation.

Product Detail Modal Implementation: This is the part that will be used to display the rating, specifically from 1 star to 5 stars. The data will be taken from product.js. This feature helps users to view and give ratings about the product. (Figure 5)

```
<div className="rating">
  {[1, 2, 3, 4, 5].map((star) => (
    <span
      key={star}
      className={`star ${star <= product.rating ? "filled" : ""}`}
    >
      ★
    </span>
  ))}
</div>
```

Figure 5. Star Rating Display in ProductCard Component.

4.2.2 Challenges and Solutions

Challenge 1: Managing product detail display without using React Router.

- Challenge: In the initial steps, I designed and implemented the project with the intention of having a separate product overview page and a separate product detail page. But when working and delving into the concepts, the author of this thesis realized that there was a certain difference between a Single Page Website and a Single Page Application, and what I needed was an SPW, not a SPA, so using React Router to navigate to another page was not suitable for the original purpose of the thesis. Because the ProductDetal page was initially defined as a separate router (/product/:id), up to this point it was the structure of an SPA, not purely about SPW anymore.
- Solution: To ensure the correct technical orientation and original goal of the thesis, the author of this thesis changed the product detail page to be displayed in a modal overlay format. Modal overlay allows when the user clicks on a product, the ProductDetail component will be rendered internally right in ProductList, and the modal will appear overlaying the existing product list without having to redirect to another page. (The code after fixing will be Figure 4. Product Detail Modal Implementation).

Challenge 2: Handling Wishlist Events Without Affecting the Modal

- Challenge: The problem that arises in this part is that when the user clicks on the ❤️‍🩹 button (which is the icon when the user adds a product to the Wishlist) to add to the Wishlist in a product card, the onClick from **div.product-card** will also be triggered at the same time, leading to the opening of the detail modal - this is something the author of this thesis does not want and needs to be fixed.
- Solution: In the ProductCard.js code file, the developer used **e.stopPropagation()** to prevent the click event from propagating from the Wishlist button to the product card. (Figure 6)

```
        <button
          className={`wishlist-btn ${
            wishlist.includes(product.id) ? "added" : ""
          }`}
          onClick={(e) => {
            e.stopPropagation();
            toggleWishlist(product.id);
          }}
        >
          {wishlist.includes(product.id) ? "❤️" : "🤍"}
        </button>
      </div>
    );
  }

  export default ProductCard;
```

Figure 6. Preventing Modal Trigger from Wishlist Button Using e.stopPropagation().

Challenge 3: Display Wishlist product list in Modal

- Challenge: The problem here is to ensure that all the features in Wishlist need to work smoothly and keep the original purpose and definition of a Single Page Website, that is, not redirecting to a new page but still being able to manage and perform the inherent operations to manage favorite products in Wishlist.
- Solution: The proposed solution is that users can open the list of favorite products from the ❤️‍🩹 My Wishlist button, from there they can delete

products right in the modal. All data will be saved in localStorage to maintain the state across page reloads. (Figure 7)

```
{isWishlistOpen && (
  <div className="wishlist-modal">
    <div className="wishlist-content">
      <span className="close-btn" onClick={() => setIsWishlistOpen(false)}>❌</span>
      <h2>My Wishlist</h2>
      {wishlist.length === 0 ? (
        <p className="no-results">Your wishlist is empty.</p>
      ) : (
        <div className="product-grid">
          {products
            .filter((product) => wishlist.includes(product.id))
            .map((product) => (
              <div key={product.id} className="product-card">
                <img src={product.image} alt={product.name} className="product-image" />
                <h2>{product.name}</h2>
                <p className="price">${product.price}</p>
                <button
                  className="wishlist-btn added"
                  onClick={() => toggleWishlist(product.id)}
                >
                  ❌ Remove
                </button>
              </div>
            ))}
        </div>
      )}
    </div>
  </div>
)}
```

Figure 7. Wishlist Modal with In-Place Product Removal and Local Persistence.

Challenge 4: Styling product details and Wishlist in modal

Challenge: The content appearing in the modal will mostly overflow the screen, greatly affecting the aesthetics and user experience. This problem is especially big for the Wishlist display when the list contains too many products.

Solution: The author of this thesis added the max-height and overflow-y: auto attributes to the CSS file so that the modal can scroll the content contained inside when it is too long, which will solve the above error. (Figure 8)

```
.wishlist-modal {
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
    z-index: 1000;
    width: 90%;
    max-width: 500px;
    max-height: 80vh;
    overflow-y: auto;
}
```

Figure 8. CSS rules for wishlist modal with scrollable overflow.

4.3 Vue.js Implementation:

This is the part where the project's Vue version is deployed and developed. The Vue version will also have the same core features as the React version to ensure consistency and provide more accurate results when comparing the two frameworks.

4.3.1 Key Features Used

In the Vue version, the project is written with the latest version of Vue 3 and is almost the default when you install the Vue environment with the command **npm install -g @vue/cli** then install the project with the command **vue create** vue-product-list then the project will be initialized with the version of Vue 3. This version of Vue is still a website listing electronic products that reflects the same functions as the React version, while using the strengths and idioms of Vue 3 with Composition API.

Component-Based Architecture with Composition API

Vue differs from React in that Vue requires and encourages you to use **<script setup>** syntax in .vue files to simplify the logic and improve the readability of the framework, compared to the flexibility and freedom of the React framework.

The Vue version of the application is structured into reusable and easily extensible components similar to the React version:

- ProductCard.vue: Displays individual product information.
- WishlistButton.vue: Handles the logic and UI for converting the wishlist.
- ProductList.vue: The parent view that comprises the card, filter, and wishlist dialog.

Reactive State Management using **ref** and **computed**: In Vue framework, Composition API's ref and reactive will be used in ProductList.vue code file to easily manage filters, selected products and wishlist state reactively. (Figure 9)

```
const searchTerm = ref("");
const category = ref("All");
const sortOrder = ref("default");
const ratingFilter = ref(0);
const wishlist = ref([]);
const selectedProduct = ref(null);
const isWishlistOpen = ref(false);

const filteredProducts = computed(() => {
  return products
    .filter((p) =>
      p.name.toLowerCase().includes(searchTerm.value.toLowerCase())
    )
    .filter((p) => category.value === "All" || p.category === category.value)
    .filter((p) => p.rating >= ratingFilter.value)
    .sort((a, b) => {
      if (sortOrder.value === "asc") return a.price - b.price;
      if (sortOrder.value === "desc") return b.price - a.price;
      return 0;
    });
});
```

Figure 9. Reactive product filters modal.

Template Directives and Dynamic Binding: In Vue framework, there is a system of vue directives including **v-for**, **v-if**, **:class**, **@click** to help link UI and logic as concisely and clearly as possible. A specific example in that code is the Wishlist method that uses **v-if** to display conditional methods and **v-for** to iterate over products. (Figure 10)

```vue
<div v-if="isWishlistOpen" class="wishlist-modal">
  <div class="wishlist-content">
    <span class="close-btn" @click="isWishlistOpen = false">❌</span>
    <h2>My Wishlist</h2>
    <div v-if="wishlist.length === 0" class="no-results">
      Your wishlist is empty.
    </div>
    <div v-else class="product-grid">
      <ProductCard
        v-for="product in wishlistProducts"
        :key="product.id"
        :product="product"
        @select="selectProduct"
      />
    </div>
  </div>
</div>
```

Figure 10. Wishlist modal rendering with directives.

Local Storage Synchronization: Similar to the React version, this Vue version will also use local storage for storage. Vue will keep track of the wishlist array so that it can synchronize it with localStorage whenever there is a change. To make this logic persistent, it is necessary to implement **watch()**. (Figure 11)

```vue
watch(
  wishlist,
  (newVal) => {
    localStorage.setItem("wishlist", JSON.stringify(newVal));
  },
  { deep: true }
);
```

Figure 11. Vue watch for syncing wishlist to localStorage.

CSS Styling & Modular Layouts: As for CSS, this will be a reused component from the React version to ensure visual consistency and design is applied to all modals such as wishlist and detail. However, the Vue version is improved to support scrolling inside the modal with **max-height** and **overflow-y**. At the same time, the grid layouts are designed separately to accommodate product previews and hover effects using scoped styles in the Vue SFC. (Figure 12)

```css
.wishlist-modal {
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);
    z-index: 1000;
    width: 90%;
    max-width: 500px;
    max-height: 80vh;
    overflow-y: auto;
}


.wishlist-content .product-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
    gap: 10px;
}
```

Figure 12. Vue wishlist modal with scroll styling.

4.3.2 Challenges and Solutions

Challenge 1: Strict ESLint & Prettier Integration:

Challenge: When developing a project with Vue version and specifically Vue 3, during the process of setting up the project's default features using Vue CLI, the author of this thesis chose the default feature including strict Prettier and ESLint rules to help the code easy to read and easy to look at. However, this causes quite a few compilation errors due to seemingly small formatting or text issues. For example: unwanted line breaks, missing semicolons or extra whitespace. All of them will be detected by the Prettier and ESlint systems and warned as an error, making the project unusable until you fix it.

Solution: When the notification system detects errors in Prettier and ESLint, the developer will execute a command to use the Prettier system to refresh and beautify the segments to the format that Prettier wants, which is the command **npx prettier --write "path to the file to be edited"**, for example: **npx prettier --write src/data/products.js**. There are also some ways such as running **npm**

run lint --fix or **.eslintrc.cjs** to temporarily prevent error notification rules. Although this causes some discomfort, this system will help your code become cleaner and more standardized.

Challenge 2: Prop Management and Emit Events:

Challenge: For the Vue framework, **defineProps** and **defineEmits** are used in the **<script setup>**, which is significantly different from how props and callbacks are handled in the React framework. If a developer forgets to define an emit event or misnames a prop, it will often cause silent errors and no response from the code.

Solution: To overcome this, the developer just needs to note that all props must be declared carefully using **definePropS()** and communicated back to the parent component using **defineEmits()**. Developers can and will be encouraged to use support tools such as Volar for Vue 3 and Vetur for Vue 2 to help detect these errors early.

Challenge 3: Working with Local Storage Reactively

Challenge: Vue framework does not automatically synchronize data with localStorage. So updates to wishlist in code will not be reflected and saved in localStorage unless explicitly and manually tracked.

Solution: **watch(wishlist, callback, { deep: true })** has been implemented to automatically persist and update changes. Also, initialization is done via **ref(JSON.parse(localStorage.getItem('wishlist')) || [ ])** to restore the state.

4.4 Comparison of Code Structures/Complexity and Development Processes

4.4.1 Project Folder Structure Comparison

Both React and Vue projects have adopted the same logical folder structure, including folders such as components, pages, data, and clearly built .js and .vue files, besides App.js and App.vue files as the main logical entry point. However,

due to the characteristics of each framework, this structure also exists some differences. (Refer to Table 3)

Table 3. Project Folder Structure Comparison.

Emplacement	React	Vue
Entry point	index.js uses ReactDOM.render	main.js uses createApp(App).use(router)
Component	.js files (JSX)	.vue Single File Components (SFC)
Routing	react-router-dom declared in App.js	vue-router declared in router/index.js
CSS Styling	Use 1 global styles.css file	Import global CSS + scope support in .vue
Assets	Set external image link	Similarly, use external links in data files.

4.4.2 Component Communication and State Management

There is a very notable and recognizable difference between the two frameworks transfer and processing of data between components.

React uses props to transfer data and employs useState to manage states. It employs the callback function passed down to the child component for handling activities such as adding products to the wishlist.

For example: **<ProductCard product={product} onSelect={setSelectedProduct} />**

Meanwhile, Vue 3 will require the use of defineProps() to receive data and defineEmits() to emit events from the child to the parent. It should improve transparency at the same time, but still, all the props and emits will need to be declared explicitly.

For example:

const props = defineProps(['product'])
const emit = defineEmits(['select'])

4.4.3 Modal Implementation and Routing Approach

Although the approaches followed by both versions are those used for Single Page Applications (SPAs), what seems different in them is the technical setup for displaying product details in the modal. (Refer to Table 4, Figure 13 and Figure 14)

Table 4. Modal Implementation and Routing Approach Comparison.

Features	React	Vue
Modal Control	**useState()** and condition **{selectedProduct && (...)}**	**ref()** and **v-if="selectedProduct"**
Routing	**react-router-dom** is not used for detail modal	**vue-router** declares path but does not use it
Wishlist display	Modal with condition **isWishlistOpen**	Similarly, use **v-if="isWishlistOpen"**
UI Handling	JSX uses inline logic	Vue uses directives **v-if**, **v-for**, **:class**

```
160          {isDetailOpen && selectedProduct && (
161            <div className="wishlist-modal">
162 >            <div className="wishlist-content">···
190              </div>
191            </div>
192          )}
193        </div>
194      );
195    }
```

Figure 13. React Modal Implementation.

```
<div v-if="selectedProduct" class="wishlist-modal">
  <div class="wishlist-content">
    <span class="close-btn" @click="selectedProduct = null">❌</span>
    <img ···
    />
    <h2>{{ selectedProduct.name }}</h2>
    <p class="category">Category: {{ selectedProduct.category }}</p>
    <p class="price">${{ selectedProduct.price }}</p>
    <p class="description">{{ selectedProduct.description }}</p>
    <div class="rating">···
    </div>
    <WishlistButton···
    />
  </div>
```

Figure 14. Vue Modal Implementation.

4.4.4 CSS and UI Considerations

While both versions share the same styles.css file, Vue provides the ability to restrict CSS to individual .vue components as needed. However, to keep things consistent, all styles are maintained in a common styles.css file.

A few interface enhancements like scrollable modals **overflow-y: auto** are applied consistently.

The "ElectroWorld" page header uses the same CSS neon light effect for both versions.

4.4.5 Developer Experience

(Refer to Table 5)

- Project initialization and environment configuration:

Project initialization for React is quite simple, developers only need to use Create React App to avoid creating a project from scratch. Developers only need to run the command line **npx create-react-app** and the system will automatically execute and create a React project ready to run and develop. As well as the pre-defined directory structures.

As for Vue, according to a developer, Vue will provide more options during the project initialization phase. Because when the developer initializes the project, the Vue framework will have a list of questions in a step-by-step order including: choosing the Vue 2 or Vue 3 version, choosing the configuration of Babel, TypeScript, Router, Vuex and especially the Prettier and ESLint systems. This can bring some flexibility but at the same time also create difficulties for beginners if they do not know which setup is suitable for the project.

- Syntax and control logic

For the React framework, it will use JSX - allowing developers to simultaneously write JavaScript nested in HTML. This makes the structure of the code very powerful but sometimes becomes confusing if you do not know how to allocate it properly, especially when there is a lot of control display logic. Writing UI and logic in the same block makes it easier to follow for JavaScript programmers and makes React more flexible but at the same time can also cause difficulties in maintenance for beginners.

As for the Vue framework, this framework uses pure HTML template layouts and combines those templates with **v-if**, **v-for**, **:class**, **@click**... to make writing code more intuitive and easier to follow. The **<script setup>** section will provide complete but very concise syntax, easy to learn, easy to read and clear for the logic. But this makes Vue more rigid than React because it must always follow a certain pattern.

- State management

The React framework will use **useState**, **useEffect**, **useCallback**... to manage state. This will require the developer to have a certain understanding of hooks and dependency arrays. Although it is somewhat powerful, it sometimes causes many errors that are difficult to debug if there is no strict control.

The Vue framework will use refs, reactive, computed, watch in the Composition API to manage state. For **ref()**, tracking and updating state is very simple and

fast. Even more special is the use of **watch()** to track changes and synchronize with localStorage to simplify a lot of logic.

- Beginner-Friendly

React will be somewhat more accessible to beginners, especially those who already know and have experience with JavaScript, but the JSX syntax and logic hooks will sometimes be difficult for beginners; but overall React is still an accessible framework with its flexibility.

Vue will be friendly to beginners learning frontend programming because the syntax is closer to pure HTML, clearer. However, when using Vue 3 with the Composition API, it can cause certain confusion when beginners have not yet distinguished and mastered **ref()**, **reactive()**, or **setup()**.

Table 5. Overall assessment after developing two projects.

Criteria	React	Vue
Initial Setup	Fast, Simple (CRA)	Flexible but easy to mess up (Vue CLI)
Syntax	JSX logic-rich, a bit confusing if there are many conditions	Separate, easy-to-read templates
Reactivity	Hooks (manual control)	Built-in with ref(), reactive()
Debug	Good React Devtools	Powerful Vue Devtools + Volar
Beginner-Friendly	Intermediate – needs in-depth understanding of hooks	High – easy to learn with syntax close to HTML

5 Evaluation: Performance Comparison with Lighthouse

In this performance evaluation, the author of this thesis used the Lighthouse tool to evaluate the overall performance of both website versions. The tool is an open-source tool developed and published by Google, which helps the analysis become more qualitative and reputable based on criteria such as Performance, Accessibility, SEO, and Best Practices.

Below are the results obtained after performing the test with Lighthouse with both versions having the same content and structure. (Refer to Table 6, Figure 15 and Figure 16)

Table 6. Lighthouse audit results for React and Vue apps.

Evaluation criteria	React SPW	Vue SPW
Performance	80	88
Accessibility	85	76
SEO	100	100
Best Practices	100	91

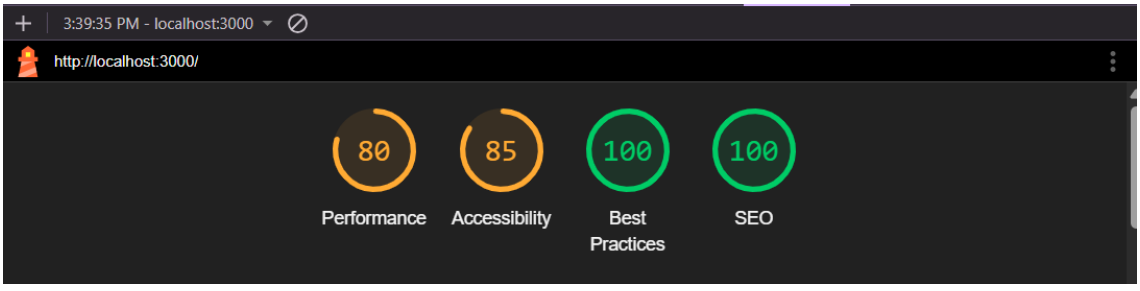


Figure 15. Lighthouse audit results for React app.

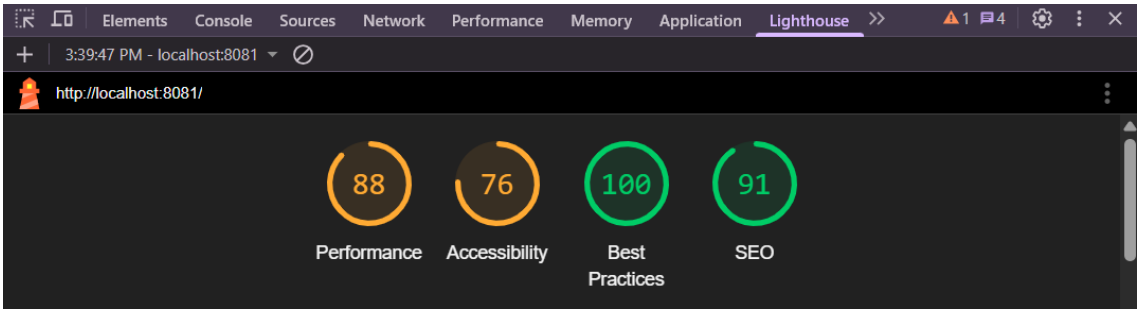


Figure 16. Lighthouse audit results for Vue app.

Performance: When testing the performance, the results show that the React version scored 80 points and the Vue version scored 88 points, 8 points higher than the React version on a scale of 100.

The metrics obtained from the Lighthouse report:
First Contentful Paint (FCP): Vue was recorded at a speed of 0.8 seconds while React was 1.3 seconds, which shows that the Vue version will have a better start and display content faster while React will be a little slower than Vue.

Latest Contentful Paint (LCP): Vue was recorded at a very good speed of 0.9 seconds, which is a good response time and meets most of the standards, while React was 2.9 seconds, which is not a very good response speed and needs further improvement.

Cumulative Layout Shift (CLS): Vue absolutely nails it with a perfect 0 seconds for user experience, while React comes in at 0.023 seconds, which is not too remarkable, yet overall positive.

Speed Index: For the version of Vue, the recorded content rendering speed is 7.6 seconds, which is a very sluggish response time that absolutely needs to be addressed, and for the React version, 1.3s which shows that the page's content is rendered almost instantaneously, offering a much superior experience.

Accessibility: Vue scored 76 while React performed remarkably, earning a grade of 85.

Vue’s penalty:

Not having the lang="en" attribute in the <html> tag creates inability for screen readers to identify the language.

Some selectors are missing a <label> tag which negatively impacts access to assistive technology.

Users with disabilities with the vision also are at a disadvantage due to a lack of adequate contrast between elements with text and the background.

In comparison, React’s use of standard HTML and good reuse of clear WAI-ARIA elements allow them to meet the requirements.

Best Practices: Both scoring 100 indicates that there are no security weaknesses in the source code, the script does not include JavaScript that might affect performance, and all requests are made via HTTPS. So both React and Vue are well set up:

Third party render blocking scripts are absent.

Obsolete APIs are not present.

All images were optimized and properly formatted.This indicates the two frameworks support contemporary web development by observing strict programming standards of reliability and efficiency.

SEO (Search Engine Optimization): Vue has a score of 91 while React scored superbly with a 100.

The SEO reasons Vue was fined:

A meta description remains absent, limiting the search meta display capabilities in Vue's index.html file.

Lack of descriptive alt content for images in some components (e.g., in ProductCard).

We should point out, however, that the SEO score is a mere primary technical outcome. Implementing SSR (Server-Side Rendering) via Nuxt or Next.js enables both frameworks to achieve a professional SEO standard.

6 Discussion

Developing two instances of the same application using React and Vue simultaneously gives a good picture of the features of these two popular frameworks. Starting from architecture designing to execution and programming experience, both frameworks possess points of strength along with points to observe while practically using them.

As for performance, Google Lighthouse test results show that Vue performs much better (88 compared to 80 for React) due mainly to its small bundle size and load time. React is considerably better in Speed Index (1.3 seconds compared to 7.6 seconds for Vue), which indicates speed in content rendering first content to screen. This may be because React wants to render important components first and load resources optimally.

Under Best Practices and Accessibility, React still leads at 85 and 100, respectively, whereas Vue has 76 and 91, respectively. Maybe this is because Vue hasn't fully implemented the semantic HTML tags or included some infinitesimally small errors in structure within the document. Still, they both rank 100 for SEO, implying that their search abilities are on par with each other and extremely great for public and broadly accessible applications.

There are some notable differences to note when describing the developer experience when working with both React and Vue. When working with React, the framework provides clear rules for working with JSX, useState, useEffect, and hooks that are organized and consistent. However, the framework requires a logical and strict separation of files so that it can be easily improved and extended later. In contrast, for Vue, the <script setup> syntax will help developers shorten steps and make the code easier to read, but also requires familiarity with refs, computed, watch as well as managing props/emits in the Composition API environment.

One of the biggest obstacles that programmers encounter when working with Vue for the first time is the strictness of the ESLint and Prettier systems built

into the project when initializing the project with Vue. This system will help programmers make the code neater, easier to read and improve, but will cause more compilation errors due to incorrect formatting, thereby making the code unusable. But this is an extremely useful tool once you get used to it, this is something that in the React framework sometimes depends on the individual settings of each programmer.

In case we consider the UI logic, both frameworks actually use modals to display product details and wishlists as well as the website interface, following the original Single Page Website design guidelines. However, handling modal state and data transfer in Vue version will be somewhat easier to extend thanks to the framework's default reactive properties, while React needs to pay more attention to prop drilling and manual interface updates.

In summary, both frameworks can be used to build complete applications and websites at all levels clearly and efficiently. From the above thesis and experience, Vue will be suitable for small to medium projects with short deployment time requirements and quick learning needs. On the contrary, React will play a huge role and is suitable for large and massive projects, where complex architectures and scalability play an important role. Ultimately, the choice of which framework depends largely on the project goals, the skills of the development team, and the ecosystem that you want to leverage in the future.

7 Conclusion

This thesis has compared and evaluated two frameworks, React.js and Vue.js, in the context of developing and implementing a single-page website with the same core functionalities and the same user interface. The thesis has implemented two identical projects on the same display content, which is electronic products with core features, which are search, filtering, sorting, product detail modals, and wishlist functionality. The purpose of the thesis was to comprehend, investigate, and analyze how the two models deal with a set of provided requirements, and thereby their relative strengths, weaknesses, and experience in development.

Both frameworks were determined to meet all the functional and technological demands as per the parallel project-based methodology. stood out for its usability, simplicity, and effective use of the Composition API to achieve modular development, whereas React offered flexibility, stability, and scalability. Utilization of various coding approaches and patterns throughout the development process led to disparate developer experiences for both frameworks.

The Lighthouse performance test showed that overall, Vue was better performing in load speed and performance. When the issue of accessibility and speed to first paint is also taken into account, React performed better. Real-world experience also supported these results, which also reflected React's robust codebase and control, and Vue's benefit for rapid prototyping.

A combination of the two in complexity, development process, and structure was achieved by developing the two projects side by side. The result is that neither framework is inherently superior to the other, but instead, the choice depends on the project's size and the developer’s familiarity with the framework, long-term goals, and team size.

For developers, teams, and organizations seeking to apply a suitable front-end framework for modern SPW development, this comparative thesis presents a handy guide.

References

Alter Solutions. (2024). Performance tuning with Google Lighthouse: a guide to better metrics. Available at: https://www.alter-solutions.com/articles/google-lighthouse-metrics (Accessed: 27 February 2025).

Chrome Developers. (2024a). First Contentful Paint (FCP). Available at: https://developer.chrome.com/docs/lighthouse/performance/first-contentful-paint (Accessed: 27 February 2025).

Chrome Developers. (2024b). Largest Contentful Paint (LCP). Available at: https://developer.chrome.com/docs/lighthouse/performance/lighthouse-largest-contentful-paint (Accessed: 27 February 2025).

Emmanni, P.S. (2023). Comparative Analysis of Angular, React, and Vue.js in Single Page Application Development. International Journal of Science and Research (IJSR), 12(6), pp. 2971–2974. Available at: https://www.ijsr.net/archive/v12i6/SR24401230015.pdf (Accessed: 18 February 2025).

Hasan, M. R. (2023). Vuex vs Pinia: A Comparison of Vue State Management Libraries. Available at: https://www.linkedin.com/pulse/vuex-vs-pinia-comparison-vue-state-management-libraries-hasan (Accessed: 27 February 2025).

MDN Web Docs. (n.d.a). Frameworks and libraries. Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries (Accessed: 12 February 2025).

MDN Web Docs. (n.d.b). Introduction to web development. Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Introduction (Accessed: 12 February 2025).

Monterail. (2024). Vue vs React: Which One Should You Choose for Your Next Project? Available at: https://www.monterail.com/blog/vue-vs-react (Accessed: 18 February 2025).

Pretorius, L. (2024). Single-Page Websites: When to Use Them. Available at: https://bird.marketing/blog/digital-marketing/guide/web-design-trends-best-practices/single-page-websites-best-practices (Accessed: 14 February 2025).

React Documentation. (n.d.). Introduction to React. Available at: https://react.dev/ (Accessed: 14 February 2025).

Tadesse, H. (2023). A Comparative Analysis of ReactJS and Vue.js. Available at: https://medium.com/@hiwibesty/an-important-part-of-frontend-development-choosing-the-right-framework-or-library-can-affect-the-63548041576f (Accessed: 18 February 2025).

Vue.js. (n.d.a). Comparison with Other Frameworks. Available at: https://vuejs.org/v2/guide/comparison.html (Accessed: 18 February 2025).

Vue.js Documentation. (n.d.b). Introduction to Vue.js. Available at: https://vuejs.org/guide/introduction.html (Accessed: 18 February 2025).