A Comparative Web-Based System for Evaluating React and Vue Through Architecture, Performance, and Developer Experience

Asher Griess

2025-12-11

Abstract: Choosing an appropriate front-end web solution is a common challenge for developers and organizations, particularly when evaluating popular JavaScript technologies such as React and Vue. Existing comparisons often emphasize subjective preferences, limited benchmarks, or incomplete/outdated technical explanations. This project presents a web-based comparative system that integrates React and Vue into a single static website to provide an objective, reproducible, and example-driven evaluation of the two solutions. The system is implemented using Vite, npm workspaces, and GitHub Pages and supports side-by-side explanations of architectural design, state management, routing, and performance behavior. A custom modular shell application handles navigation and rendering between independent React and Vue projects, enabling consistent styling and user experience. Evaluation focuses on functional correctness, deployability, and the successful integration of both frameworks within a shared environment. This is supplemented by performance analysis drawn from scholarly and industry benchmarks. Results demonstrate that the system could effectively support comparative learning, local reproducibility, and transparent sourcing while also highlighting known trade-offs in scalability, optimization effort, and ecosystem maturity. The project consists of a reusable architectural pattern for framework comparison and provides a foundation for future expansion across additional metrics and front-end technologies.
Keywords: React, Vue, Vue.js, Comparison, Web, Application

Contents

I Introduction

The problem that I intend to solve with this project is to give a more decisive answer to whether to choose React or Vue.js web solutions. These are two of the most popular web solutions that are often seen compared in the industry. It is important to mention that Vue is a framework, while React is not, and as such I will call them solutions when referring to them as a group [1]. These are both JavaScript/TypeScript web solutions that allow for developers easier creation of websites. When looking at industry documentation that compares the two, I typically feel that they lack depth and objectivity. They fail to either consider certain aspects of a solution, such as performance, or tend to focus on a single one. This project is to help frontend developers, technical leads, and students decide which solution to choose.

My project will entail creating a website that will host a more thorough comparison of the two frameworks. The basic objective is create a static site that integrates both React and Vue to provide runnable examples. This will include pages that focus on specific aspects or features of each framework. Included in this will be working examples; this allows developers to get a better overall sense of how they work. It will also make it easy for developers to clone the code and edit it on their local machine to get a more thorough understanding. This will have a higher focus on explaining how things work vs. subjective comparisons.

This is an improvement because I believe it will not only give a more objective comparison of the two solutions but a more in-depth one as well. If done this way, developers will at least be more informed on the two solutions and thus more capable of making the choice between the two. Getting a better understanding of the frameworks should help developers understand the implications their design choices will have on their projects. This will hopefully increase the number of correct choices and thus save developers and companies more time and money. It is also important to mention that if you know one of the solutions, this comparison could also be used to help learn and understand the other. Additionally, I hope to bridge the gap between academia and industry in terms of accessibility, methodology, and information.

II Background and Related Work

2.1 Qualitative vs Quantitative

A lot of what makes this comparison difficult is due to its generally qualitative nature. This is due to the multifaceted nature of how frameworks allow users to do something in many ways for many real-life scenarios. Where one solution may work well, in another situation it may not. This is why many of the sources I have for this project are documentation, as they seek to explain.

There are exceptions to this rule, as I found academic and industry sources that dealt with more quantitative results. These categories mostly had to do with two qualities that

tend to involve more numbers, which are performance and community. These will be the bulk of my non-documentation sources, as they work best for these types of comparisons.

2.2 General Findings From Literature

Most of the explanation for the finding is on the actual site itself, which you can find in Section 7. But one of the main things that we can find is that React and Vue handle the Virtual DOM very differently [2], [3]. What I'll discuss here are some more general things about the two and what I found out about handling sources themselves.

Overall, I found that React is the more popular option than Vue when it comes to web development [1], [4]. This is critical to consider since React is not considered a framework and expects you to use outside libraries to create software. Since React is more popular, there are many options for libraries to complement React and specific use cases, making it generally more flexible. We also need to consider that Vue provides more out-of-the-box support for creating a fully functional web application. Not only that, but Vue is widely considered easier to learn than React [1], [4].

React tends to be better for larger applications where its modularity and large community create better solutions to large problems [4]. However, because of how it handles the Virtual DOM, it tends to fall behind in performance when compared to Vue [2]. We do need to consider that there are many developer optimizations and libraries to help increase performance. This is demonstrated in the results of some studies where measurements that were not granular sometimes did better than Vue. This gap may be confusing as it fails to fully show how and why React performed the way it does and the extra effort it may take to increase performance.

Vue overall was considered best for small to medium applications due to its simplicity and completeness [4]. It does, however, without any extra libraries, handle large amounts of state data with better performance and effectiveness. But it tends to fall short when it comes to large applications due to its lesser amount of libraries and compared flexibility [1], [4]. It can be used for larger applications but will require much more thought and consideration.

There is a lack of examples when it comes to more general comparisons in these studies in most areas except performance. Even then, performance measurement methods were different enough that they tended to reveal diversified results. We also need to consider the updates that have been happening frequently and often with these frameworks that may make data in these studies obsolete. Taking these sources and giving them a deeper analysis while cross-referencing them with official documentation should clear up some inconsistencies. This would also bring the data into a more singular source, giving readers a better idea which solutions to choose.

2.3 Scholarly vs Industry

Overall, scholarly sources agree on most things about React and Vue when it comes to more direct comparisons. However, they tend to add crucial details that add much more context or have outdated information that has changed. These articles also tend to focus on a group or single aspect of each solution, which may fail to give an idea of the bigger picture. Additionally, scholarly sources tend to update less frequently, which makes it more likely to be out of date as these web solutions update. When it comes to actual testing,

transparency seems to be much better in scholarly papers, which give a better idea of why we got certain numbers.

Industry sources tend to be more useful in certain forms; most of the time this is documentation. This also aligns better with my methodology of explanation over conclusions. Additionally, it is often updated frequently, especially considering the popularity of the two frameworks [1], [4]. When it comes to testing most industry docs have a tendency towards bias and lack of transparency. This is because testing methods have vary largely producing very different results, if not transparent it can be hard to determine what it says about the web solution.

III System Design and Implementation

3.1 Plan Overview

My plan was to use GitHub to help host my site as a static. This allows me to not only host my project for free, but using GitHub Actions, I am able to automatically deploy it as well. This would include both React and Vue.js sites into a single static site through various methods, which I will discuss. Additionally, I will include both the libraries prettier and eslint to have more uniform code in terms of styling and formatting.

The first thing that I need to do in terms of general structure is create a home page. This will include an overview of what is contained in the website and what it should be used for. This will include a bunch of subpages that include a bunch of pages that compare certain features and contain links to working examples. Some of these subpages may also be more general overviews of certain aspects, such as performance. This can be seen in Figure 3.1, where I provide some examples of basic things I need to create examples of, such as state management or routing. While in the real website these pages will likely be more specific, this should give you an idea of what it will look like. You can also note that performance doesn't have examples since we will be using benchmark number comparisons instead.

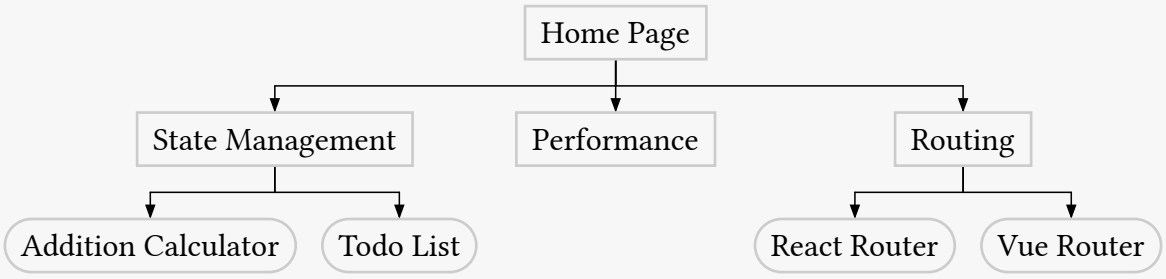

Figure 3.1: General Website Structure with Examples

3.2 Project Creation

I started by creating both the react and vue project directories inside of GitHub repository for version control. I did this using a package called create-vite which allows you create multiple types of web projects using the vite building tool. This tool allows us to not only start the projects, but also have start with some vite configuration which we will use to build our website distribution. I did this by running the following commands resulting in the partial setup of the project [5].

Setup Commands Bash

```
mkdir shell
npm create vite@latest react-app -- --template react
cd react-app
npm install
cd ..
npm create vite@latest vue-app -- --template vue
cd vue-app
npm install
```

Listing 3.1: Bash command that were used to create web application.

3.3 NPM & Vite

After this I would go about creating the shell app, which would deal with grabbing the pages from each project depending on the URL. It serves as the connection between React and Vue so that we can have them on a singular static site. When we build the final solution, we will have a single dist directory that contains all the projects built, where the shell app is the entry that calls the other projects. This meant that I had to mess with our package.json and vite configuration files. In the root directory of the project, which had its own package.json along with all the project subdirectories. I used what are called workspaces in npm to allow for our npm script to be executable from the root directory. This also allows us to install all npm packages with one npm install command instead of having to do so in each directory.

For the shell app, I had to go out of my way to make sure that when building and running the local dev server, the shell app would be able to see and use the builds of both projects. I was able to achieve this in each of the projects vite configuration files so that we had a single dist folder that the shell could access, which you are to see in Listing 3.2.

shell/vite.config.js JavaScript

```
16    server: {
17      port: 5173,
18      fs: {
19        // allow access to project root and sub-app builds
20        allow: [
21          "..",
22          path.resolve(__dirname, "../dist/react-app"),
23          path.resolve(__dirname, "../dist/vue-app"),
24          path.resolve(__dirname, "../dist/shell"),
25        ],
26      },
27    },
28    build: {
29      outDir: "../dist",
30      emptyOutDir: false,
31      rollupOptions: {
32        external: [
33          "../dist/vue-app/vue-entry.js",
34          "../dist/react-app/react-entry.js",
35        ],
36      },
37    },
```

Listing 3.2: Vite Config for the shell for vue and react dists.

3.4 React & Vue Integration (Routing)

Overall I will not be able to go over every single change that I made to make this happen between the two apps. I will go over the main things that I did, since there are many smaller things that I had to in order to get this function that it would likely fill most the report. Most of the magic is done within the shell app, I did this by using the vue-router package to manage all the routing between the two separate projects. I did in conjunction with the Vue project as I ended up using Vue for all the non example pages.

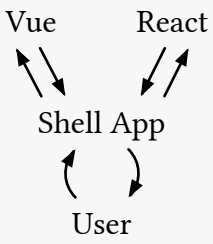

Figure 3.2: General Website Architecture

Since we are using a static site, we can't have multiple pages in the sense they have a different normal URL. Instead we use what is called a hash routing, which uses a special

hash character before the actual URL that is used internally Listing 3.3-h1 . This means the URL is never sent to the server, but managed by the page script itself [6]. This is done in the shell app to show the user what view we are looking at which behaves similar to having multiple pages. You can see how this will generally work as the shell app with handle requests from the user and give Vue or React as seen in Figure 3.2.

shell/src/main.js JavaScript

```
1  import { createApp } from "vue";
2  import { createRouter, createWebHashHistory } from "vue-router";
3
4  const VueWrapper = {
5    template: '<div id="vue-container"></div>',
6    async mounted() {
7      await import("../../dist/vue-app/vue-entry.css");
8      const { mount } = await import("../../dist/vue-app/vue-entry.js");
9      const hash = window.location.hash;
10     const match = hash.match(/^#\/vue(\/.*)?$/);
11     const vueSubPath = match?.[1] || "/";
12     this.vueUnmount = await mount("#vue-container", vueSubPath);
13   },
14   unmounted() {
15     if (this.vueUnmount) this.vueUnmount();
16   },
17 };
...                                    ...
31 const routes = [
32   { path: "/", redirect: "/vue/" },
33   { path: "/vue/:pathMatch(.*)*", name: "vue", component:
     VueWrapper },
34   { path: "/react/:pathMatch(.*)*", component: ReactWrapper },
35 ];
36
37 const router = createRouter({
38   history: createWebHashHistory h1("/react-vs-vue/"),
39   routes,
40 });
41
42 createApp({ template: "<router-view />" }).use(router).mount("#app");
43 });
```

Listing 3.3: A code block showing the shell app routing

Next, we have to consider that the Vue app itself will have routing of some sort to go between pages. The solution I came up with was to allow the Vue app to change the hash route based on its own route. This means that in the Vue app we have to use the memory mode, which is similar to hash but doesn't change anything in the URL [6]. Additionally, we

can have the Shell app send the URL to the Vue app so it can sync up to the requested hash URL. This way, if I type the URL to go to a certain page, it will take me straight there and not to the homepage Listing 3.4.

vue-app/src/main.js JavaScript

```
export function mount(selector, initialPath = '/') {
  appInstance = createApp(App)
  // Add Prime Vue Theming and Ripple Effect
  appInstance.use(PrimeVue, {
    theme: {
      preset: customPreset,
    },
  })
  appInstance.use(router)
  router.push(initialPath).catch(() => {})
  let updatingFromShell = false

  // Whenever Vue navigates, update shell hash
  router.afterEach((to) => {
    if (updatingFromShell) return
    const desiredHash = `#/vue${to.fullPath}`
    if (window.location.hash !== desiredHash) {
      window.location.hash = desiredHash
    }
  })

  // Sync Shell URL → Vue router
  hashListener = () => {
    const hash = window.location.hash || ''
    if (!hash.startsWith('#/vue')) return

    const match = hash.match(/^#\/vue(\/.*)?$/)
    const vueSubPath = match?.[1] || '/'

    updatingFromShell = true
    router.push(vueSubPath).catch(() => {})
    updatingFromShell = false
  }
  // Adds listener to trigger on hash change
  window.addEventListener('hashchange', hashListener)
  appInstance.mount(selector)
  return () => unmount()
}
```

Listing 3.4: A code block showing the integration of vue-app and shell app

3.5 GitHub Pages

Using npm I was able to create a package-lock.json that allows the GitHub Actions pipelines to deal with installing all packages. Additionally, we use npm scripts to build our project in the pipeline, which is then hosted on GitHub Pages automatically after each commit with changes. These scripts not only make our deployment possible but also make it easy for someone to view the site and run the project locally. After we create a build in the GitHub pipeline, we upload this as an artifact to GitHub Pages after some setup and finally deploy the page.

3.6 Styling & Components

For the actual content pages that I will have on my website, we will be using Vue. This seems to work best since you can easily use the Vite router without using Vite. I did this in the shell app, and it made it easier to interface with Vue over React. For this reason I had to find a component library for Vue in order to save time on page styling and also give my website a more uniform feel. For this I used a NPM component library called primevue, this included some of their other packages, primeicons and @primeuix/themes. The first thing this called for was me setting the theming and a custom ripple directive to use its ripple effects to make the UI more interactive. The component I really ended up using was the Menubar to allow users to easily navigate between sites from any screen. I ended up modifying some example code and creating a new solution that makes it easy to add new pages and their icons. I additionally added compatibility for navigating to React pages since it follows a different behavior.

In the shell app, I had to add some code to handle loading CSS. In Vite, I set cssCodeSplit to false so it became a separate file in the dist. This way I could use the path to load the CSS file in the shell application when loading it. This also means that we are only using the Vue app for CSS since there tends to be some conflict when loading multiple CSS files and also because JavaScript makes it harder to unload them. It also works in my favor, as we want the React pages to have the same CSS as the Vue examples.

3.7 Citations & Code Highlighting

Since I wanted to add citations to my website, I thought it best to find a plugin that could output BibTeX to IEEE format in JavaScript. For this I found a package called citation-js which can be used to generate formatted reference sections from BibTeX. The above approach works well since I use BibTeX in my documentation to define sources. This also gives the project transparency on where I got my information. This resulted in me making a few functions so that I could easily create an ordered reference list and also in-text citations with the addition of @citation-js/plugin-bibtex and @citation-js/plugin-csl.

vue-app/src/functions/bib.js JavaScript

```
import { Cite } from '@citation-js/core'
import '@citation-js/plugin-bibtex'
import '@citation-js/plugin-csl'

const rawEntries = "" // BibTeX Content here

async function init() {
  const cite = await Cite.async(rawEntries)
  const entries = cite.data

  formattedEntries = cite
    .format('bibliography', { nosort: true, template: 'ieee' })
    .split('\n')
    .filter(Boolean)
    .map((text, index) => ({
      id: entries[index].id,
      number: index + 1,
      text,
    }))
}

export default {
  init,
  getCitationNumber(id) {
    const item = formattedEntries.find((e) => e.id === id)
    return `[${item ? item.number : '?'}]`
  },
  getCitationNumbers(ids) {
    const items = ids.map((id) => formattedEntries.find((e) => e.id
    === id))
    let processedItems = []
    for (let i = 0; i < items.length; i++) {
      processedItems.push(`[${items[i] ? items[i].number : '?'}]`)
    }
    const resultString = processedItems.join(', ')
    return resultString
  },
  getBibliography() {
    return formattedEntries
  },
}
```

Listing 3.5: A code block showing the implementation of citation-js

In Listing 3.5, we can see that there are a few functions that I implemented to create a flexible solution for each page. First we have the init function, which I use to load the reference text that will be used on every page that requires sources. This just ensures that we have the list of entries loaded before we render any pages. It also ensures that the order is correct by adding the nosort option, which just goes in order of the BibTeX entries.

vue-app/src/views/metrics/PerformanceView.vue Vue

```
2  <script setup>
3  import GenericPage from '@/components/templates/GenericPage.vue'
4  import vueRender from '@/assets/VueRender.svg?raw'
5  import { inject } from 'vue'
6  const citations = inject('citations')
7  </script>
...                                    ...
12 <p>
13   One of the metrics that both React and Vue are often compared on is
     performance. Both
14   solutions use a virtual DOM solution to more efficiently render
     changes to the page. This
15   generally means faster modification times. This is because the
     virtual DOM tends to be more
16   lightweight and additionally allows batched updates
17   {{ citations.getCitationNumbers(['Trends', 'ComparPopWeb',
     'VueRendering']) }}.
18 </p>
...                                    ...
36 <p>
37   React tends to have a more simplistic approach when it comes to
     handling the virtual DOM.
38   Most of the time, React handles renders/changes by walking through
     the subtree of the
39   component, which starts the render loop. It is able to do this
     because in React, components
40   are only able to directly share their state with their descendants
41   {{ citations.getCitationNumber('RendPerf') }}. This on its own can
     increase performance
42   since we don't have to walk through the whole tree if we don't
     initialize the render loop
43   from the root node.
44 </p>
```

Listing 3.6: Citation Example

On the exported item in bib.js we have 3 additional functions. The first function getBibliography returns a list of text for each formatted source. The second function getCitationNumber takes the input of the BibTeX label and outputs the reference text by its number. The second function getCitationNumbers takes a list of BibTeX labels and outputs their

reference numbers in order as text. This allows me to easily reference items from the bibliography by their BibTeX label, which I can set, similar to LaTeX.

3.8 Code Highlighting

Another thing that needed to be incorporated into the site was code highlighting (syntax highlighting). This allows us to better display code on the site, which will be necessary when presenting how different web solutions work. For this I used a npm package called highlight.js along with a plugin for Vue @citation-js/plugin-bibtex. This allowed me to first register the languages that I wanted to use, as seen in Listing 3.7-h1 . Secondly, it allowed me to use the plugin to create a component that can be used through the app, as seen at Listing 3.7-h2 .

vue-app/src/main.js JavaScript

```
9  import hljs from 'highlight.js/lib/core'
10 import bash from 'highlight.js/lib/languages/bash'
11 import javascript from 'highlight.js/lib/languages/javascript'
12 import hljsVuePlugin from '@highlightjs/vue-plugin'
13
...                                    ...
18 hljs.registerLanguage('bash', bash) h1
19 hljs.registerLanguage('javascript', javascript)
20
...                                    ...
47 export async function mount(selector, initialPath = '/') {
...                                    ...
57   await citations.init()
58   appInstance.provide('citations', citations)
59   appInstance.directive('ripple', Ripple)
60   appInstance.use(router)
61   appInstance.use(hljsVuePlugin) h2
...                                    ...
71 }
```

Listing 3.7: Code highlight import in main.

Once we have gone about registering the new highlightjs component, we can then use it in our Vue files as seen in Listing 3.8.

codeExample.vue Vue

```
1 <highlightjs
2         language="javascript"
3         code="Code to be Displayed"
4 />
```

Listing 3.8: Code highlight example for JavaScript.

IV Evaluation and Results

When it comes to this specific type of project, evaluation is quite difficult due to its complexity and qualitative nature. However, there are some more simple things that we can assess when it comes to outcomes. For example, the website is hosted and able to be visited; additionally, you are able to navigate between React and Vue. This is a large addition to the project, as it allows me to showcase the result of functional code for both solutions. Developers are also able to run these examples locally on their machine and add changes. I was able to achieve this result locally by using the `npm run dev` command, which allowed me to navigate to both sites. Additionally, I successfully deployed the application and got this cross-solution routing functional, as seen in Table 4.1.

Target	Measurement
Build Size (MB)	3.33 MB
Build Time	7.325 Seconds
Number of Dependencies	340
Successful static export?	Yes
Static refresh on route loads correct page?	Yes
Does React route Loads?	Yes
Does Vue route Loads?	Yes

Table 4.1: Functionality Metrics using NodeJs 24.11.1, and npm veriosn 11.6.2.

Most of the work that I have done so far has been on the performance comparison page. This has seemed to give me some results that have given me some more insight into how performance compares between the two. My first task was explaining the process of how React and Vue both deal with rendering and dealing with changes to the DOM. In lots of ways they were both very similar in how they dealt with the Virtual DOM; however, Vue seems to do a bunch more things that help improve performance in certain common situations. There were exceptions, however, as React gives some options for developers to use that function similar to how Vue's built-in systems handle reactivity.

I felt like this content better explained how these systems work and lined up fairly well with results I have found from sources. Two of the scholarly sources seem to be much more specific about how testing worked and lined up with industry testing. Comparing these sources seemed to give more uniform results that made sense together and in the context of the rendering systems used by the solutions. Overall, however, I found that Vue generally performed better than React, especially given certain situations (more details on my site linked in Section 7) [2], [7], [8]. I tried to format the page similar to IEEE, so I would just check the content for more specifics as it is already on the page.

4.1 Success Metrics

Target	Met?
Cover most common features in both solutions and comparing them.	No
Create examples for most features that allow more developer incite.	Yes
Compare both solutions on more general aspect such as performance or scalability.	Yes
Make the project easy to run locally to allow developers to modify the code.	Yes
Actually host the website through GitHub for people to view.	Yes
Functional React and Vue Integration.	Yes
Host website through GitHub Pages.	Yes

Table 4.2: Previous & new set expected targets and if they were met.

Most of the metrics that I had defined in previous assignments were specific things that I wanted to have done. For example, I wanted to actually host my website using GitHub so that people could view it, and I ended up getting it working. Additionally, I made it easy for developers to run the project locally on their machines. Most of the other objectives I have partially satisfied. I wanted to cover the most common features for both solutions, but I only really ended up starting on the state features. I also didn't get much of a chance to start examples, but I created some to show that it is possible to do. Finally, I got mostly done with the performance section that focuses on a more general aspect of the solution.

Overall, I would consider that this project was successful, as it shows that this style of solution can be achieved. I would say that there are some things that could be added to create a fully working and used version. Adding some sort of feedback embedded on the site itself may be helpful in getting an idea of what users thought and if it achieved its goal. Additionally, having multiple people contributing and talking about how each feature works could help address some issues with content being accurate. I also never got around to feature parity; I did show it was possible using multiple examples in both frameworks. If I continued to work on this project, it seems that it would likely be quite successful with a few small additions.

4.2 Discussion and Contributions

The results that I have so far seem to demonstrate that when complete, this could be a useful tool for developers in a multitude of ways. Firstly, we have the set objective of helping developers choose a web solution. If a developer wanted to better understand a solution in a certain aspect, this site could help them with either solution. Additionally, if a developer already knew one of these explanations, it could help them understand the other, as they are framed as a comparison.

I also think that this helps bridge the gap between scholarly and industry. It is an easily accessible website that would attract developers but at the same time is formatted similar to IEEE. It also uses a combination of scholarly and industry sources that provide a more complete and updated overview of the two solutions. I also think that the way I analyze the content is much more scholarly but more accessible to industry in terms of access and acknowledgement. This unique solution takes some of the best ideas from both to create this bridge that can be useful to both.

So far most of what I have learned is specific either to the actual implementation of the project or to how React and Vue compare in terms of performance. It would also be

important to mention that I had never used Vue before this project, but I learned it fairly quickly due to my experiences with a variety of web solutions. Most of what I learned in terms of content for React had to do with built-in solutions to certain performance issues. This included certain hooks and functions that stop components from being re-rendered or memoized functions. This included a new official tool in the React ecosystem called the React compiler that tries to automatically add the aforementioned performance solutions during build time. Although I had little to no data on its success, I mentioned that it should be considered and may improve the performance of React over time as it improves.

I learned a lot more working on this about Vue than I did React, since it has a lot more complex system than React. One of these systems is Vue's limited reactivity system, which helps track dependencies to reduce the amount of re-renders. This system made some performance results more explainable, as the concepts seemed to directly apply to the testing situations.

Overall, I think that my project is plagued by limitations due to the situational nature of comparisons. This made it so most of what I had written has no clear-cut answers; there were or are exceptions to most of the things discussed. In this sense I wish there were more large-scale studies that gave more data-driven answers for average developers. I also think that to some degree there will always be a case in which the points I make are not true. In this sense I don't believe it to be possible to completely 100% of the time help developers choose the right options. This could even depend on the people on the team, which I have no sense of when creating the project.

Additionally, I think time was a real limiting factor so far; trying to dig this deep into the concepts takes a large amount of time for even one aspect of comparison. This will likely limit the amount of content that I will have in the final result. Most of the future improvement would just be more refinement of already written material and examples and just adding more comparison to the project.

Overall I would say that this project demonstrated that this type of solution as an education comparison platform is possible. It also gives us a better idea of the impact that it could have if better perfected with time spent and some additional features. It showed us that this single-page front-end styled integration is not only possible but also able to be built in a way that allows for a reusable static deployment pattern.

4.3 Future Work

Overall, the main thing that would be needed for future work is adding and expanding the amount of features and metrics talked about. This would allow developers to have a better overall mindset of how each framework does and works in certain areas. The more added, the better picture that we would be able to construct for developers. This would result in a more complete outcome that would be useful for comparison and learning. If you already knew one of the solutions, this would make it easier to learn by comparison, as you would get to see the other solutions equivalents.

Another thing that would add a lot of value would be some sort of community feedback. This could be either through GitHub as an open-source project or a feedback forum that would be a part of the website. The GitHub part is mostly already set up, as we have everything deployed through GitHub. If we wanted to add a feedback forum, I would need to get some sort of database or server to store the data since we currently have a static site.

V Ethical and Societal Considerations

There are few ethical and user-impact considerations that I need to make with this project. I need to realize that developers could use this project to help choose a framework that will have real-world impacts on businesses and individual developers. I need to consider my own bias when writing about the two frameworks, which is why I mostly focus on explantation, as it eliminates some of this bias. Additionally, I acknowledge multiple times in the text that you should be assessing your individual situation, as I am unable to account for all variables.

The main ethical and societal consideration that I will need to worry about when it comes to this project is misuse. For example, if a user only decides to read one section of the website, they may get an incomplete idea of the solutions. This means that I need to be careful with my wording and be clear about avoiding a black-and-white view on aspects of each framework.

Overall, I do need to consider that my decisions in this project may result in choices being made about choosing either framework. This could mean someone who doesn't pick well could lose either time or money. My job here is not to make everyone select the correct option but to more heavily consider their options. By providing information that could be useful in informing these decisions, I hope to improve the number of "correct" decisions.

In terms of transparency and feedback, GitHub provides ways for users to either request changes or give me feedback. Additionally, they are able to run and see how the site was created. I also use IEEE formatting for the sources to better allow for checks on how I found and curated information for this project. Part of my hope is that there will be some community support and collaboration to help this project be kept up to date and more accurate to real-life situations.

VI Conclusion

This project has the foundation laid to create a final usable result. It successfully integrates Vue and React as a single-page application and hosts them as a static site on GitHub Pages. Additionally, using npm I have created a result that can easily be replicated and modified locally. This allows developers to go view the actual code used to create each example. There is a page that showcases how each framework should be compared, even if we have hard numbers on the performance page. This page explains how each framework deals with the whole rendering process, shows code examples, and provides comparison numbers from industry and scholarly sources. This should give a good sense of direction for the complete project if the project were to be continued.

VII Supplementals

7.1 Links

- GitHub: react-vs-vue

- Website: React + Vue Site

VIII References

[1] T. Uppal, S. Srivastava, and K. Saini, "Web Development Framework : Future Trends," in *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2022, pp. 2181–2184. doi: 10.1109/ICAC3N56670.2022.10074105.

[2] R. Ollila, N. Mäkitalo, and T. Mikkonen, "Modern Web Frameworks: A Comparison of Rendering Performance," *Journal of Web Engineering*, vol. 21, no. 3, pp. 789–813, 2022, doi: 10.13052/jwe1540-9589.21311.

[3] Vue Developers, "Rendering Mechanism." [Online]. Available: https://vuejs.org/guide/extras/rendering-mechanism

[4] G. Kaur and R. G. Tiwari, "Comparison and Analysis of Popular Frontend Frameworks and Libraries: An Evaluation of Parameters for Frontend Web Development," in *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2023, pp. 1067–1073. doi: 10.1109/ICESC57686.2023.10192987.

[5] create-vite Developers, "create-vite - npm." [Online]. Available: https://www.npmjs.com/package/create-vite

[6] Vue Developers, "Different History Mode | Vue Router." [Online]. Available: https://router.vuejs.org/guide/essentials/history-mode.html

[7] S. Krause, "js-framework-benchmark results for Chrome 142.0.7444.60— krausest.github.io." 2025.

[8] R. N. Diniz-Junior *et al.*, "Evaluating the performance of web rendering technologies based on JavaScript: Angular, React, and Vue," in *2022 XVLIII Latin American Computer Conference (CLEI)*, 2022, pp. 1–9. doi: 10.1109/CLEI56649.2022.9959901.

[9] J. Tong, R. R. Jikson, and A. A. S. Gunawan, "Comparative Performance Analysis of Javascript Frontend Web Frameworks," in *2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, 2023, pp. 81–86. doi: 10.1109/ICE3IS59323.2023.10335250.

[10] J. Sianandar and I. B. Kerthyayana Manuaba, "Performance Analysis of Hooks Functionality in React and Vue Frameworks," in *2022 International Conference on Information Management and Technology (ICIMTech)*, 2022, pp. 139–143. doi: 10.1109/ICIMTech55957.2022.9915183.

[11] A. Donvir, P. K. Saraswathi, and A. Jain, "End-to-End Application and Backend State Management: A Comprehensive Review," in *2024 IEEE 15th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2024, pp. 31–43. doi: 10.1109/UEMCON62879.2024.10754779.

[12] L. Qianqian and D. Yuxiao, "A Comprehensive Study on State Management Patterns of React," in *2023 International Conference on Applied Physics and Computing (ICAPC)*, 2023, pp. 769–772. doi: 10.1109/ICAPC61546.2023.00150.

[13] React Developers, "Component - React." [Online]. Available: https://react.dev/reference/react/Component

[14] single-spa Developers, “The Reccommended Setup | single-spa.” [Online]. Available: https://single-spa.js.org/docs/recommended-setup/

[15] Vue Developers, “Provide / Inject| Vue.js.” [Online]. Available: https://vuejs.org/guide/components/provide-inject

[16] React Developers, “useMemo - React.” [Online]. Available: https://react.dev/reference/react/useMemo

[17] React Developers, “memo - React.” [Online]. Available: https://react.dev/reference/react/memo

[18] React Developers, “useContext - React.” [Online]. Available: https://react.dev/reference/react/useContext

[19] React Developers, “React Compiler - React.” [Online]. Available: https://react.dev/learn/react-compiler

[20] N. Makarevich, “I tried React Compiler today, and guess what....” 2024.

[21] Vue Developers, “Reactivity in Depth | Vue.js.” [Online]. Available: https://vuejs.org/guide/extras/reactivity-in-depth.html