Optimizing Web Application Development: Comparing Frameworks, Architectures, and Components

Boris Brnkaľák, Ján Janech, Patrik Hrkút

***Abstract* —** Web application development offers diverse methodologies, tools, and standards, each suited to specific requirements. This paper explores the two predominant architectures for web applications: multi-page applications (MPAs) and single-page applications (SPAs). It delves into their respective advantages, challenges, and appropriate use cases. The study also examines the selection of backend and frontend technologies, emphasizing frameworks such as Laravel, Node.js, React, Angular, and others, which impact application performance, scalability, and maintainability. Furthermore, the evaluation of component libraries, including Ant Design and PrimeReact, highlights their role in streamlining development. By comparing frameworks and component packages based on criteria like speed, complexity, support, and popularity, this research provides actionable insights for choosing optimal technologies tailored to application-specific goals.

***Keywords* —** Web application development, single-page applications, multi-page applications, backend frameworks, frontend frameworks, component libraries, technology comparison.

I. Introduction

When creating web applications, there are multiple approaches that we can use and there is no single way to develop one. There is a wide variety of different tools, methods and standards that we can use. This diversity is a positive factor, as it ensures that there are suitable options for different types of applications. However, the crucial problem is how to choose the right methodologies, tools and standards for application development. It is essential to make the right decision at the initial stage, as this will affect the further course of application development.

In the initial phase of web application development, we must consider many different aspects, such as choosing the basic architecture. We can decide to create a multi-page application (MPA), which functions like traditional web applications, consisting of separate pages, with each page dedicated to one specific functionality or content. An alternative is a single-page application (SPA), which works by dynamically rendering and loading content on a single page. The advantage of SPA is the ability to update content without having to refresh the entire page. Both architectures have their own advantages and disadvantages, and it is important to weigh them against your established preferences to find the right development method for your application.

Once we have chosen a certain development process, the next step is to choose a language for the backend part of the application. When developing SPA applications, an equally important step is choosing a frontend framework/library that will affect the user experience of the developed application. Therefore, our article will analyze what type of architecture, technologies and programming languages to choose [1].

Boris Brnkaľák, University of Žilina, Slovak Republic (e-mail: boris.brnkalak@fri.uniza.sk)
Ján Janech, , University of Žilina, Slovak Republic (e-mail: jan.janech@fri.uniza.sk)
Patrik Hrkút, University of Žilina, Slovak Republic (e-mail: patrik.hrkut@fri.uniza.sk)

II. ARCHITECTURES FOR WEB APPLICATION DEVELOPMENT

As we mentioned earlier, there are two main approaches to building web applications: SPA and MPA. The architecture is chosen depending on the type of application and specific requirements. These requirements may include various criteria, such as performance, SEO (search engine optimization), ease of implementation, popularity among users, and other advantages and disadvantages, which will be analyzed in more detail later [1].

A. Multi-page applications

MPAs work by reloading the entire page content every time a link is clicked, or a form is submitted. An MPA consists of multiple web pages, unlike an SPA, where there is only one page that dynamically loads and switches content.

The advantages of using MPA include robust support for SEO, allowing web browsers to easily search for content on the site. Another advantage is *developer experience*, with the MPA architecture being considered easier to implement compared to SPA. Finally, there is also the fact that JavaScript, which creates HTML elements, is not necessarily required to display the application, as the server will provide the generated HTML.

MPA also has its drawbacks. One of them is its lower speed compared to SPA, because with each redirection the entire application content is reloaded, even if some of the content has not changed, which also increases the load on the server. Another known disadvantage is the *user experience* aspect, especially if many users use mobile devices and expect a greater level of interactivity and faster content display, which is not so common with MPA [2] [3].

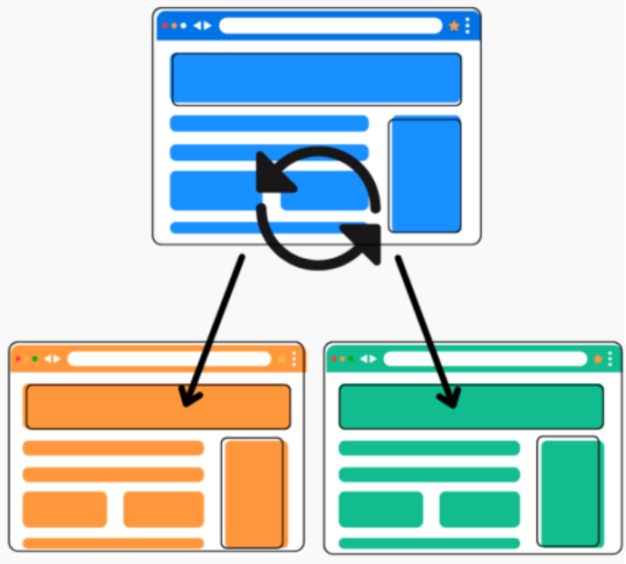

Fig. 1 Multi-page web application architecture

B. Single Page applications

SPA applications operate within a single browser session and allow the user to seamlessly navigate between components without having to reload the entire content each time they switch to a different page. SPA is built on dynamically rewriting the page content using JavaScript. Instead of loading the entire page, only the content that is currently needed to be displayed or has been changed is requested.

The most significant advantages include speed, because the application downloads the necessary content in advance when it is first loaded, which will be displayed later. This eliminates the need to load all the data and significantly increases the overall speed of the

application. Another advantage is data caching, which means less load on the server. Data is loaded from the server only when there is a change in the content, which contributes to optimizing the performance of the entire application.

Disadvantages of SPA include limited SEO support, which is more difficult to implement. This is because browsers do not have efficient tools to collect information from a single dynamic page compared to multiple pages. Another disadvantage is the necessity of using JavaScript for SPA to function properly. Without JavaScript, the application cannot take advantage of dynamic reloading, so if the user has JavaScript disabled in the browser, the application will not work at all [2] [3].

Figure 2 Single-page web application architecture

III. Parts of a Web Application

A web application is usually divided into two parts, which are separate and perform predetermined functions. Each part performs defined tasks, which are usually specific to it.

A. Frontend

The frontend is responsible for the visual aspect of the application, including styles, animations, and data display that are visible to the user. When developing the frontend part applications often we put emphasis on specific design elements such as colors, images, and graphs, creating a significant impact on the UX (user experience). The development process itself includes the implementation of various interactive elements such as scroll bars, navigation links, and switching between different subpages [4].

B. Backend

Backend managed by logic applications on server side. Its task is to modify, insert, delete and retrieve data from the database. It also provides authentication and authorization of users and if necessary, the backend communicates with external application programming interface (API). At the same time, it takes care of the validation of requests received from forms. The security of the backend is very important because it communicates with the database, where all the application data is stored [5].

After deciding on the appropriate architecture for the application being developed, it is necessary to choose the language, libraries and frameworks in which the application will be developed. The frontend will provide the visual part of the application, the backend takes care of the logic of the entire system, and finally the database, where all the data will be stored. This

choice is equally important because it allows us to customize the tools according to the specific requirements of our application.

IV. Frameworks and libraries

A framework can be described as the foundation on which we build our software projects. It provides a basis for creating a project, where it is not necessary to start from scratch, because some functions are already created and prepared in advance. The purpose of a framework is to simplify and speed up the software development process. Frameworks are often associated with specific programming languages and exist for both frontend and backend areas, some of which we will discuss in more detail with specific examples. Using frameworks brings several advantages, including saving time, reducing the risk of errors, and the ability to avoid creating each file from scratch. A framework can provide the basic structure of an application at the initial stage of its development. The most significant benefit of frameworks is that they are based on components. Components represent the encapsulation of various functionalities and content that can be reused as needed. Creating components effectively prevents the creation of duplicate code and minimizes the risk of errors. Another advantage is that they allow us to achieve clean and easily customizable code, which is especially important for long-term development, where multiple developers may work on a project over time.

Frameworks should also be chosen based on popularity, as a popular framework will have a larger user base and therefore an active community. This means that there will be more tutorials, documentation, libraries that we can deploy into the framework without any more complicated work. Another benefit is that more popular frameworks are up to date, have security patches, and are sustainable for the future. Especially when choosing a framework, it is important to consider the following criteria – characteristics, simplicity, shortcomings, and disadvantages. There are also more criteria [6], which we will describe later.

A. Backend frameworks and libraries

Backend frameworks and libraries contain pre-built code and file structures that provide programmers with a good foundation for creating application logic and server-side code. They offer pre-built files with a given code structure, modules, and libraries with generated code sections so that developers don't have to create them from scratch. If developers follow certain conventions for a particular framework, development itself will be much easier, and additional functionality will also be easier.

At the same time, they improve code maintainability. A big advantage of using a backend framework makes it easier to communicate with the database, create tables and individual attributes in tables using pre-prepared commands.

At the backend frameworks we most often encounter languages such as: PHP, JavaScript, Ruby, Java and many others. When choosing such a framework, the web hosting itself can play a big role, where most companies offer the PHP language, which is very widespread. On the other hand, we have JS, Ruby and other web hosting, which are not so widespread and therefore the application must operate on the cloud in a container environment (Docker), which is significantly more expensive than PHP web hosting [5].

B. Laravel

Laravel is one of the most popular PHP frameworks. It is popular for its features that make developers' work easier. The framework is built on the MVC architecture, which takes care of the order in the code. Laravel is an elegant framework, offering features such as: routing, authentication, authorization and database management. Laravel is one of the best choices among PHP frameworks because it has a large user base, which eliminates the lack of information when solving a problem. It has detailed documentation and there are also many

portals where we can find various procedures to solve a specific problem that may occur during development. It also uses up -to- date technologies and has support for creating reactive sections, where we do not have to refresh the entire content from scratch, but only a certain part (component) is refreshed. An example of such an option is the use of *Livewire* technology.

Laravel was developed to support the development of MPA applications and uses blade *templating system* to generate HTML. The syntax allows you to combine HTML with PHP functions, which allows you to create loops, conditions and other functionalities. The resulting source code is finally converted to PHP code. Its configuration is easily customizable and allows for easy combination with some of the frontend frameworks. For example, it is possible to integrate Laravel and React, which provides several advantages, such as using Laravel for efficient and fast creation of the logic of the entire application and React for creating a dynamic and interactive user interface [5] [7].

C. Node.js (Express)

Express is a JavaScript framework for building web applications. It is one of the most popular options for building APIs. It is easy to use and fast. Express is particularly well-suited for applications that are used to mediate real-time communication because Express uses a non-blocking, event-driven architecture. It supports routing, can create templates for dynamic HTML generation, and supports HTTP methods such as POST, PUT, GET, DELETE, and other methods for building REST APIs. Express is popular, which has led to a large user base, ongoing support, and support for many third-party libraries.

As mentioned above, Express is very minimalistic, so we don't need a lot of code to create application logic. One disadvantage of Express is that there is no predefined way to organize files, which means we must create our own file organization structure [5] [8].

D. Ruby on Rails

Ruby on Rails is a framework for developing web applications, based on the Ruby language. Using the Rails framework reduces the amount of code written. Rails uses the main principle of Don't Repeat Yourself, which says that the same functionality must not be written again. If we reuse the given logic in another place, we create one component and use it everywhere we need the given logic. This will guarantee us a more maintainable and extensible code. Ruby on Rails is built on the MVC architecture. The advantages of the Ruby on Rails framework include simple and fast implementation. Another advantage is the wide community, which means that there are many third-party libraries that can make our development more efficient and easier [5] [9].

V. FRONTEND FRAMEWORKS AND LIBRARIES

Frontend frameworks can be defined as a collection of tools, libraries, and pre-designed components and architectures. All frontend frameworks are built on JavaScript, which is essential for the framework to function at all. A huge advantage is that we can reuse these components without creating new parts of the code, we just need to dynamically change their content and render the component we want to display. Examples are tables, buttons, links, which streamline the overall development process and improve UX. However, today's trend is to use the TypeScript language, which is more recommended, as it improves the efficiency of writing code, maintains greater clarity of the code, and reduces the risk of errors by using static types, where each parameter and variable must have a defined type of frontends nowadays frameworks. In our article, we will describe four of them in more detail.

A. React.js

React to be one of the most widely used and popular frontends. frameworks. The main reason for the creation of this framework was to simplify the complex process of creating interactive applications. We can imagine a user interface that is created using collections of components, where each component is responsible for a certain small part of the code that is re-usable. The parameters that we can send to the component are called props and facilitate easier transfer of large data. A parameter can also be a function that is called when a specific event is triggered, for example, clicking a button. React uses JSX code, which allows you to write HTML code directly in JavaScript and thus we can easily create dynamic content in the application. The content is always rendered when the state changes. In React, there are two approaches to writing and creating components. One is creating a class based components, where components are created as separate classes with different functions, but this approach is no longer recommended. Another approach is function-based approach, which is newer and more popular. Function creation based components consists of creating a function into which we write the necessary code, where the output should be JSX code.

React is a SPA framework, so its advantage is that whenever the uniform resources locator (URL) changes does not reload the entire application. It uses a client-server architecture to retrieve data from the server. Simply put, if we have a page containing a large amount of data, after invoking an action, only the data that has changed is loaded [10] [11].

B. Angular

Angular is an open-source JavaScript framework developed by Google. It was developed for the purpose of creating SPA applications and it is built on the MVC architecture. Each component we create can have its own HTML template, a separate file for testing, and finally a controller, where the complete logic assigned to a specific component is written.

One of the biggest advantages of Angular is two-way data binding, where the framework synchronizes the model using a two-way data binding technique. When the data in the controller changes, the view itself changes at the same time. This feature can reduce the development time for developers, as they do not have to write additional code to synchronize between the controller and the view [10] [12].

C. Vue.js

Vue.js is a JavaScript framework that is also designed for creating SPA user interfaces. It consists of HTML, JS, CSS (cascading style sheets) and provides a component model. Like React, Vue.js supports declarative rendering, which extends HTML itself, allowing us to describe the output in HTML based on the state of variables. The reactivity of the Vue.js framework tracks changes in the state of JS variables and then performs various operations on the DOM (document object model).

Vue.js is based on the MVVM pattern. The MVVM pattern connects the View and Model based on two-way data bindings. Like most architectures, MVVM helps organize code into different modules for easier development and maintenance [10] [13].

D. Solid.js

Solid is a reactive library inspired by React.js. Solid.js is not very well known, but it has good features and is quickly becoming popular. Solid has a smaller user community, which results in fewer packages, forum questions/answers, and overall support. It uses a similar component model with dynamic data binding to JSX in React. The bigger difference from React is that components are only executed once. If necessary, it is possible to re-execute a part of the code when the state changes. The result is a simpler mental model. It has a similar idea to React in one-way data flow, but it has a different implementation and changed syntax. Solid allows us

to create reactive variables called signals. Signals take their initial value from the parameter of the function that creates the signal. Solid automatically changes the content when the value in the signal changes, and thus the dynamic content that is rendered to the DOM changes.

Compared to React, Solid is faster. Unlike React, it uses a compiled DOM to render content instead of a virtual DOM. This means that after compilation, we achieve a smaller JavaScript size, which is reflected in its speed compared to React. This makes Solid significantly faster and more powerful [14].

VI. Frontend component packages

We define component packs as collections of ready-made user interface elements that help us speed up development. Among these elements, we can define various buttons, tables, modal windows, charts, and many more.

As we mentioned above, component packages are created to simplify and accelerate frontend development. The most well-known advantages of using component packages include speed, ease of use, where you just need to call a pre-prepared component without writing additional CSS code. Another advantage is adaptability, where many ready-to-use components are manipulable, and we can change them according to our specific needs. Finally, we include a unified and modern UI, where we do not have to design the design of the entire website, because component sets already provide unified styles. The last advantage we mention is the provision of so-called *headless components*, which means that the component is fully functional, but does not contain any styles. So, we can design such a component according to our preferences [15] [16].

A. PrimeReact

PrimeReact is a library that contains component sets for creating user interfaces. As the name suggests, it is a library developed for React. It is one of the popular libraries that offers modern UI. One of its capabilities is the creation of many components without installing additional packages. It supports the creation of tables, forms, text editors, icons, graphs, navigation links and especially supports *lazy loading*, which is used to read currently used code segments [17].

B. Ant Design

Ant Design is a very popular component library. It was developed for creating UIs in React. Currently, Ant Design has 89.4 thousand stars on GitHub and has an average of 1.2 million downloads per week. It supports the creation of modern UIs. This support is essential if we want to create enterprise-level applications. It has many components that we can use without installing additional packages, such as tables, forms, custom icons, trees, charting, custom calendars, and many others [18].

C. Mantine

Mantine is a fully functional open source library designed for creating UI in React. It has many components that we can edit according to our needs. The Mantine library is not as popular as the previous libraries. It currently has 23.5 thousand stars on the GitHub portal and has an average of 325 thousand downloads per week. It provides many components such as forms, calendars, text editors, carousels and many others. The only drawback may be the smaller user base [19].

D. Next UI

Next UI, like other libraries, is developed to create a user interface for the React library. It has a perfect and very modern design. Creating components using this library is easy. Next UI is based on the Tailwind CSS framework. It currently has 18.4 thousand stars on GitHub and

an average of 72 thousand downloads per week on the *node packet manager* (NPM) website. This number of downloads is much smaller compared to the libraries we are listing. Like the other libraries, it supports almost all the necessary components. The only disadvantage we see in the given library is the smaller user base [20].

VII. Choosing the Right Framework

As we mentioned, the choice has a fundamental impact on the development of web applications and its later change is often very complicated. That is why we decided to make a comparison of individual options. We selected individual criteria and evaluated them. The individual percentages that we assigned to them are based on our empirical experience, as well as the study of available information on the Internet.

A. Backend part selection

The backend performs part or all the application logic. A good solution is to choose a language and tool that is popular and supported by web hosts. The PHP language is still used and its support by web hosts is the best among all available languages for developing the backend part. *Laravel* is currently the most widely used PHP web framework, which greatly facilitates the work of developers. Among the various PHP frameworks, we recommend Laravel, which has a large user base, popularity among developers, ease of writing PHP functions, where many are already ready, and support for third-party libraries.

B. Frontend part selection

The choice of frontend framework is crucial because it will affect a significant part of the application development, and it is the part of the application that the user meets. It is essential to choose the right option based on certain appropriately chosen criteria. The criteria that we will choose must be set in such a way that we select the right library that will meet the requirements.

C. Determining evaluation criteria for choosing a frontend framework

As we mentioned earlier, deciding on the right framework or library depends on the specifics of our application. It is necessary to establish criteria according to which we will decide on a suitable framework to port our application to. We will assign a weight to each criterion, and the higher the weight, the more emphasis we place on the given criterion.

Speed – The main idea behind converting to SPA is to improve user experience. The speed criterion is of greater value to us to provide a good user experience. Therefore, the weight of the criterion is 25%.

Complexity – Choosing the complexity criterion speaks to how difficult it is to implement and create new functionalities within the framework. Along with complexity, the possibility of scaling in the future is also related. This criterion was not that important to us, so we assigned it a weight of 10%.

Support – Support was one of the most important criteria, where we require that the framework has strong and long-term package support, with the aim of not having to change the framework in the future due to unsupported packages. Since this criterion is one of the key ones, the weight of this criterion is 25%.

Popularity – The popularity of the framework is another criterion that we decided to evaluate, as it is important that the framework has a sufficiently large user base. We also evaluated this criterion using the GitHub portal, where we focused on the number of stars added by developers from all over the world who use the framework and express satisfaction with it. Therefore, if a serious error occurs, we can find more detailed information about the error, which is why we require that it has detailed documentation. We would be able to find many

answers on user portals and forums, and most importantly, various libraries and packages supported by the framework would be created. Therefore, we will evaluate this criterion with a weight of 20%.

Size – One of the smaller criteria we chose is the size of the entire framework. Our goal is not to make the framework unnecessarily large after compilation. This criterion is not that important to us, so we will evaluate it with a weight of 5%.

Ecosystem – The last criterion we will evaluate is the ecosystem. Within this criterion, we will focus on the number of packages and features that the community has created to accelerate development. Therefore, this criterion will have a weight of 15%.

D. Comparison of frameworks

We will compare frontend frameworks and libraries according to the established criteria. Based on these comparisons, we will select suitable candidates. The resulting evaluation of the frameworks will be presented in a table.

TABLE I
FRONTEND COMPARISON FRAMEWORKS BASED ON EVALUATION CRITERIA

	Frameworks and libraries			
	React.js	Angular	Vue.js	Solid.js
Speed (25%)	15%	19%	21%	25%
Complexity (10%)	9%	5%	9%	10%
Support (25%)	25%	25%	23%	10%
Popularity (20%)	20%	8%	14%	5%
Size (5%)	1.5%	1%	2.5%	5%
Ecosystem (15%)	15%	12%	11%	4%
Sum	**85.5%**	70%	80.5%	59%

E. Frontend comparison evaluation frameworks

After comparing various frameworks and libraries, we concluded that React.js [21] [22] was the most suitable framework for our needs. As can be seen in Table 1, React.js again scored the highest on our evaluation criteria. This framework scores highly in popularity and is developed and maintained by Facebook. Frameworks like React and Angular [23] [24] are products of large companies, which proves that they will have long-term support. Solid [25] [26] excelled in many technical criteria, but its weakness is its popularity and ecosystem, as it is relatively new. Although React and Vue [27] [28] scored similarly, we decided to recommend React due to its larger user base and rich integration with third-party packages. After selecting React as our framework, we analyzed the component packages and then selected the most suitable solution [29].

VIII. COMPONENT PACKAGE SELECTION

For the component package, we selected libraries that are popular and have a large user base. We will then select a component package based on the criteria we have set. Just as we set the criteria for selecting a frontend framework and subsequently selected React as the most effective, we will select component packages that were created specifically for the React.js frontend library.

A. Determining evaluation criteria for component package selection

We required the component package to include various tables and functions for sorting and filtering data. It was equally important to choose a package that had a modern UI. We will assign weight to each criterion, which we will consider when assessing suitability.

Basic requirements – The basic requirements include that the component package be open source with regular updates, guaranteed future support, high popularity and a wide user base. This requirement was key to maintaining design consistency in the future. Another requirement was that the component pack allowed localization for different languages. The last criterion was the possibility of theme customization, which means the ability to change colors and style according to your own preferences. This criterion will be evaluated with a weight of 35% due to its greatest importance in choosing a suitable component pack.

Datagrid – This component is usually the most used in most applications with an administrative interface, so this criterion is one of the most important. The datagrid must include filtering and sorting. We will evaluate this criterion with a weight of 20%.

Forms – As with tables, this criterion will be one of the main ones for us and we will pay increased attention to it. However, in this area, we require that forms support several functions, including automatic text completion. Furthermore, we want the component package to allow the expansion of various sections to maintain clarity, and therefore we expect support for the accordion type of component. It would be welcome if the library also provided validation as part of its functionality. This criterion will be evaluated with a weight of 20%.

Documentation – Another criterion was detailed documentation, where it will be easy to find various functions and their instructions, how we can create and edit individual components. It is essential that the documentation is detailed and includes various examples. We also require that the component package is popular and has a high number of NPM downloads. At the same time, we need the components to be easy to create and the documentation to be up to date. We evaluate this criterion with a weight of 15%.

Other (e.g. icons) – The last criterion we chose is custom icons. We chose this criterion so that we wouldn't have to download another package containing icons. We give this criterion a weight of 5%.

B. Comparing component packages

We will evaluate the component packages according to the established criteria and based on these comparisons we will select suitable candidates. The resulting evaluation of the component packages is presented in the following table.

TABLE II
COMPARISON COMPONENT PACKAGES ON BASIS EVALUATION CRITERIA

	Component packages				
	PrimeReact	MUI	Ant Design	Mantine	Next UI
Basic requirements (35%)	18%	29%	33%	18%	26%
Data grid (20%)	20%	15%	20%	8%	20%
Forms (20%)	17%	20%	18%	14%	17%
Documentation (15%)	13%	13%	13%	14%	14%
Other (icons) (5%)	3%	5%	4%	0%	0%
Sum	71%	82%	**88%**	54%	77%

C. Evaluation of the comparison of component packages

Based on a comparison of component packages, we chose Ant Design as our preferred choice. MUI, although of similar quality, was not chosen due to the paid components. PrimeReact was of good quality but not very popular, and Next UI, although of modern design, is relatively new but does not have a large user base. Mantine had the lowest score due to its low popularity, lack of support for filtering and sorting in the datagrid, and absence of form validation. Ant Design

met our requirements the most, was popular, and included custom icons, which convinced us that it was the best fit for our needs.

IX. Conclusion

In this paper, we analyzed the key considerations for web application development, focusing on the choice of architectures, frameworks, and component libraries. Multi-page applications (MPAs) and single-page applications (SPAs) were compared to highlight their suitability for different scenarios, based on performance, scalability, user experience, and SEO capabilities. Frameworks like Laravel, Node.js, React.js, Angular, and others were assessed for their contributions to backend and frontend development, emphasizing factors such as ease of use, community support, and flexibility.

Additionally, we evaluated component libraries such as Ant Design, PrimeReact, and Mantine to determine their effectiveness in accelerating development and enhancing user interfaces. Through a systematic comparison using weighted criteria, React.js and Ant Design emerged as the most suitable choices for robust and scalable web applications.

The findings underline the importance of selecting technologies that align with the specific requirements of a project while considering long-term maintainability and scalability. This study provides a comprehensive guide for developers to make informed decisions and optimize their web application development processes.

References

[1] JOHNSON, J., 2024. How To Build A Web App. In: Buildbase [online]. [cit. 2024-2-9]. Available at: https://budibase.com/blog/how-to-make-a-web-app/

[2] JACKSON-BARNES, S., 2023. SPA Vs. MPA Explained: Differences Between Single-Page Applications and Multi-Page Applications. In: Spa Vs Mpa [online]. [cit. 2024-2-9]. Available at: https://www.orientsoftware.com/blog/spa-vs-mpa/

[3] JACKOWSKI, M., 2022. Single Page Application (SPA) vs Multi Page Application (MPA) – Two Development Approaches. In: SPA Vs Mpa [online]. [cit. 2024-2-9]. Available at: https://asperbrothers.com/blog/spa-vs-mpa/

[4] What Is Angular, 2023. In: Angular [online]. [cit. 2024-2-9]. Available at: https://angular.io/guide/what-is-angular

[5] Introduction, s. a. In: Vuejs [online]. [cit. 2024-2-9]. Available at: https://vuejs.org/guide/introduction.html

[6] VARMA, S., s. a. Introduction To SolidJS [online]. [cit. 2024-2-9]. Available at: https://www.loginradius.com/blog/engineering/guest-post/introduction-to-solidjs/

[7] CHAWRE, H., s. a. Top 10 Backend Frameworks In 2024 [online]. [cit. 2024-2-9]. Available at: https://www.turing.com/resources/backend-frameworks

[8] 10.X, s. a. In: Laravel [online]. [cit. 2024-2-9]. Available at: https://laravel.com/docs/10.x

[9] Introduction, s. a. In: Developer Mozilla [online]. [cit. 2024-2-9]. Available at: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[10] MILLER, S., 2021. What Is Ruby On Rails [online]. [cit. 2024-2-9]. Available at: https://www.codecademy.com/resources/blog/what-is-ruby-on-rails/

[11] React, 2024, s. a. In: Npmjs [online]. [cit. 2024-2-21]. Available at: https://www.npmjs.com/package/react

[12] React, 2024, s. a. In: Github [online]. [cit. 2024-2-21]. Available at: https://github.com/facebook/react

[13] Core, 2024, s. a. In: Npmjs [online]. [cit. 2024-2-21]. Available at: https://www.npmjs.com/package/@angular/core

[14] Angular, 2024, s. a. In: Github [online]. [cit. 2024-2-21]. Available at: https://github.com/angular/angular

[15] Vue, 2024, s. a. In: Npmjs [online]. [cit. 2024-2-21]. Available at: https://www.npmjs.com/package/vue

[16] Core, 2024, s. a. In: Github [online]. [cit. 2024-2-21]. Available at: https://github.com/vuejs/core

[17] PrimeReact, s. a. In PrimeReact [online]. [cit. 2024-3-8]. Available at: https://primereact.org/

[18] AntDesign, s. a. In AntDesign [online]. [cit. 2024-3-8]. Available at: https://ant.design/

[19] Mantine, s. a. In Mantine [online]. [cit. 2024-3-8]. Available at: https://mantine.dev

[20] NextUI, s. a. In NextUI [online]. [cit. 2024-3-8]. Available at: https://nextui.org/

[21] Solid, 2024, s. a. In: Github [online]. [cit. 2024-2-21]. Available at: https://github.com/solidjs/solid

[22] Js Framework Benchmark 2024, s. a. In: Krausest [online]. [cit. 2024-2-21]. Available at: https://krausest.github.io/js-framework-benchmark/2024/table_chrome_121.0.6167.85.html
[23] INNOCENT, CH., 2023. Best 15 React UI Component Libraries Of 2024 [online]. [cit. 2024-2-21]. Available at: https://prismic.io/blog/react-component-libraries
[24] ACHARYA, D., 2023. React UI Components Libraries: Our Top Picks [online]. [cit. 2024-2-21]. Available at: https://kinsta.com/blog/react-components-library/
[25] MVC, 2023, s. a. In: Developer Mozilla [online]. [cit. 2024-2-28]. Available at: https://developer.mozilla.org/en-US/docs/Glossary/MVC
[26] GALLARDO, E., 2023. What Is MVVM Architecture [online]. [cit. 2024-2-28]. Available at: https://builtin.com/software-engineering-perspectives/mvvm-architecture
[27] Client Server Model, s. a. In: Geeksforgeeks [online]. [cit. 2024-2-28]. Available at: https://www.geeksforgeeks.org/client-server-model/
[28] Api Token Authentication, s. a. In Laravel [online]. [cit. 2024-3-8]. Available at: https://laravel.com/docs/10.x/sanctum#api-token-authentication
[29] Configure Store, s. a. In Redux Toolkit JS [online]. [cit. 2024-3-18]. Available at: https://redux-toolkit.js.org/api/configureStore