

Angular vs Vue

1st Cornelia Langsenlehner
Mobile Computing, BA)
FH Hagenberg, Upper Austria
Hagenberg, Austria
s2210237024@fhoee.at

2nd Sandra Hartlauer
Mobile Computing, BA)
FH Hagenberg, Upper Austria
Hagenberg, Austria
s2210237030@fhoee.at

3rd Michael Nußbaumer
Mobile Computing, BA)
FH Hagenberg, Upper Austria
Hagenberg, Austria
s2110237017@fhoee.at

Abstract—This paper presents a comparison between two popular JavaScript web frameworks: Angular and Vue. It aims to assist development teams in selecting the most suitable technology for their specific application requirements by analyzing the strengths and weaknesses of both frameworks.

The study covers theoretical aspects, such as the advantages and disadvantages of web frameworks, JavaScript for complex web applications, and performance metrics. A practical case study involving a project management tool, ProTrack, implemented in both Angular and Vue, highlights real-world applications. The analysis focuses on key functionalities like user registration, project creation, and time tracking, providing insights into performance, developer-friendliness, and community support.

The findings suggest that Angular is more suitable for large-scale, enterprise-level applications due to its comprehensive ecosystem and strong Google backing, whereas Vue offers simplicity and flexibility, making it ideal for smaller projects and rapid development.

Recommendations for future research and framework selection based on project scale and requirements are also discussed.

I. INTRODUCTION

A. Motivation

The increasing importance of web applications in the modern digital landscape motivates this analysis. According to the UNCTAD Digital Economy Report (2019), businesses and services are rapidly moving online, which increases the demand for efficient, user-friendly, and powerful web applications. The ability of web applications to operate across various platforms allows developers to serve a wide user base without the need to develop separate applications for each operating system or device [10].

B. Challenges

Web application development necessitates compatibility across multiple hardware platforms, including desktop computers, laptops, tablets, and mobile devices. Mao (2014) identifies the primary challenge as maintaining consistent performance and efficiency across these diverse devices [12].

Additionally, the selection and management of development frameworks significantly influence the efficiency and scalability of web applications. Verma (2022) points out that different frameworks exhibit unique strengths and weaknesses concerning performance, maintainability, and user-friendliness. Verma (2022) also stresses the importance

of evaluation methods in assessing the performance, security, and user-friendliness of web applications. These evaluations typically include benchmarks, test scenarios, and other empirical methods aimed at verifying the quality of the application [13].

C. Goals

This analysis aims to provide a brief overview of both frameworks and serve as a decision-making guide by offering valuable insights into the strengths and weaknesses of both technologies. These insights assist development teams in making informed decisions about the most suitable technologies for specific application requirements. Additionally, they help identify optimization opportunities to continuously enhance user experience and improve development efficiency.

D. Structure

Besides the structure, the motivation section also covers the challenges and goals of this study. The following three chapters include the theoretical background, which discusses web frameworks and compares Vue and Angular, a case study detailing a practical application and its analysis, and concludes with recommendations and future outlook.

II. THEORETICAL BACKGROUND

A. Web Frameworks

Frameworks already contain numerous functions in software development as proposed solutions for individual problems faced by various developers. This way, they do not have to start from scratch whenever they want to write new software. A framework is a program skeleton that serves as a foundation for software development.

When a framework serves as the foundation for web applications, it is referred to as a web application framework. A framework describes a collection of interacting classes and thus also fixes the design structure for software developed based on the framework [1].

1) *Pros and Cons of Web Application Frameworks:* Web frameworks offer many advantages, but they also come with some disadvantages.

2) *Advantages:*

Reduced Development Time and Cost: By using templates for basic functions such as database connections, caching, and security, development time is significantly reduced.

Open Source: Most web frameworks are available as open-source, eliminating licensing costs.

Clean and Maintainable Code: Frameworks promote the creation of clean and maintainable code, as developers can rely on proven building blocks.

Community Support: Regular improvements and quick fixes for security vulnerabilities are provided by the community [1].

3) *Disadvantages:*

Framework-Specific Requirements: Different frameworks have different design principles, potentially requiring compromises based on the project.

Code Bloat: Developers may not use all available functions, leading to unnecessary code.

Dependency Risks: Reliance on a framework and its provider can be risky if the framework is discontinued or its development slows.

Learning Curve: Familiarizing oneself with a framework's structure and use takes time.

Security Risks: Publicly accessible source code can pose security risks for enterprise applications [1].

B. Introduction to JavaScript Web Application Frameworks and Dynamic Web Content

JavaScript is a versatile programming language particularly suited for work in web browsers. Originally developed by Netscape, it has become one of the most widely used programming languages on the web, enabling developers to create dynamic and interactive content by manipulating the Document Object Model (DOM). Direct manipulation of the DOM can be challenging, which is where JavaScript frameworks and libraries come into play. They provide tools to simplify these and other aspects of programming, making JavaScript frameworks especially suitable for developing complex web applications. These frameworks offer a structured approach and pre-built components, enabling more efficient development once the concepts and guidelines of a particular framework are mastered [2].

The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM is crucial for creating dynamic web content, which is essential for modern web applications that need to be interactive and responsive [5]. Leveraging JavaScript frameworks and libraries facilitates the manipulation of the DOM, streamlining the creation of

dynamic web content.

1) *Benefits of Dynamic Web Content:* Personalization tailors content based on user preferences, demographics, or behavior, which can significantly boost engagement and conversion rates. Personalized experiences are crucial for retaining visitors and improving user satisfaction. Real-time updates provide timely information without refreshing the entire page, which is essential for news websites, social media platforms, and e-commerce sites. JavaScript enables real-time updates through asynchronous requests and data manipulation. Enhanced user engagement through interactive features such as live chat, dynamic forms, and multimedia sliders captivates users, extending session duration and reducing bounce rates.

2) *Implementing Dynamic Web Content:* To effectively implement dynamic web content using JavaScript and jQuery, it is important to clearly define goals, whether to enhance user experience, increase conversions, or improve engagement metrics. Utilizing user data by analyzing behavior with tools like Google Analytics helps in personalizing content and tailoring dynamic elements based on preferences and interactions. Choosing appropriate technologies, such as JavaScript frameworks or libraries (e.g., React, Vue.js) and jQuery plugins, streamlines development and ensures cross-browser compatibility. Continuously testing dynamic features, gathering user feedback, and refining implementations optimize performance and user satisfaction.

jQuery is a lightweight, fast, and feature-rich JavaScript library renowned for simplifying HTML document traversal and manipulation, event handling, animation, and Ajax interactions. It provides an intuitive API that seamlessly operates across various browsers. With its versatility and extendibility, jQuery has revolutionized JavaScript development for millions of users worldwide [17].

By leveraging JavaScript and jQuery for dynamic web content, websites can offer personalized, engaging experiences that cater to modern user expectations [7].

C. JavaScript for Complex Web Applications

JavaScript is well-suited for complex web applications due to its versatility and the powerful features of modern JavaScript frameworks. These frameworks provide a structured environment for development, making it easier to manage large codebases and maintain code quality.

1) *Alternatives to JavaScript:*

TypeScript: A superset of JavaScript that adds static types, making it easier to catch errors early.

Dart: Developed by Google, Dart is used in conjunction with the Flutter framework for building web and mobile apps.

Elm: A functional language that compiles to JavaScript, focused on simplicity and quality tooling [6]

D. Comparison Criteria

When evaluating web development frameworks, metrics play a crucial role as they enable an objective analysis of performance and efficiency. By comparing load times, responsiveness, and resource utilization, frameworks like Angular and Vue can be assessed in various scenarios to determine their performance.

1) *Performance*: Performance metrics are measured using the developer console, focusing on loading times and responsiveness. Loading times are assessed by recording the time taken from the start of the page load until the application is fully interactive. Responsiveness is evaluated by measuring the time taken for the application to react to user inputs and interactions after the initial load, ensuring minimal delay and smooth performance.

2) *Developer-friendliness*: Angular provides a structured environment with an array of features, complemented by detailed documentation and a well-established community, as outlined in the official Angular documentation. In contrast, Vue is noted for its straightforward integration and clear syntax, which enhances ease of use and adaptability in various project sizes, according to the official Vue.js documentation [15], [16].

3) *Community Support*: Community support is essential for the ongoing development and troubleshooting of software frameworks. This is underscored by a study conducted by Jelica Cincović and Marija Punt, which analyzed community engagement through GitHub repository statistics. According to their findings, presented in the graph below, React commands the largest community support, with Angular and Vue also showing significant but smaller communities [14].

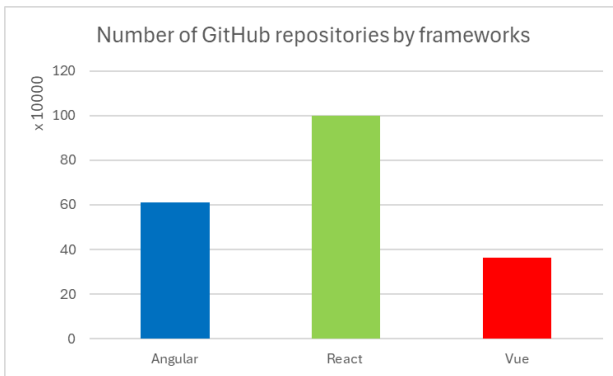


Fig. 1. Number of GitHub repositories by frameworks [14]

E. Relevant JavaScript Web Application Frameworks

Several JavaScript Web Application Frameworks are popular for developing web applications, each with its unique features and strengths. This section provides a brief overview

of two of the most relevant frameworks, Vue and Angular, and explains why they are examined in detail.

1) *Angular*: Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Developed by Google, Angular provides a comprehensive solution for complex and large-scale applications [8].

Key Features

Two-way data binding: Angular facilitates automatic synchronization of data between model and view components.

Dependency Injection: Angular's DI system promotes modular development and testing by injecting dependencies into components.

Comprehensive CLI (Command-Line Interface): Angular CLI offers powerful tools for project scaffolding, testing, and deployment.

TypeScript support: Angular is built with TypeScript, providing type safety, enhanced tooling, and improved developer productivity.

2) *Vue*: Vue is a progressive JavaScript framework designed for building user interfaces. It is characterized by its incremental adoptability, allowing developers to integrate it gradually into existing projects without the need to fully commit to the entire framework upfront. This modular approach makes Vue suitable for both small-scale projects and large-scale applications, offering flexibility in usage based on project requirements [9].

Key Features

Reactive data binding: Vue's reactive system ensures efficient updates to the DOM based on changes in data.

Component-based architecture: Vue promotes building UIs into modular, reusable components for easier maintenance and scalability.

Virtual DOM: Vue uses a virtual DOM for efficient rendering and updating of the actual DOM elements.

Flexibility and integration: Vue seamlessly integrates with existing projects and external libraries, offering flexibility in development approaches.

F. Vue vs. Angular: A Detailed Comparison

In this section, Vue and Angular will be compared based on several parameters to provide a clear understanding of their respective strengths and weaknesses.

1) Performance

Vue: Vue employs a virtual DOM, which enables efficient updates and generally faster rendering due to its lightweight design. Vue's reactivity system ensures only necessary components are updated, enhancing performance.

Angular: Angular uses a real DOM, which can be slower for updates due to extensive manipulation required. However,

Angular includes performance optimizations like ahead-of-time (AOT) compilation, which pre-compiles the application, reducing runtime overhead.

In comparison: Vue often achieves better performance for simple to moderately complex applications because of its virtual DOM and efficient reactivity system. Angular, on the other hand, provides robust performance optimizations suitable for enterprise-level applications through AOT compilation and advanced change detection strategies.

2) Developer-Friendliness:

Vue: Vue is known for its ease of learning and integration, featuring a simple syntax and flexible structure. Its comprehensive documentation and single-file components (where HTML, CSS, and JavaScript reside in a single file) make development straightforward.

Angular: Angular has a steeper learning curve due to its complexity and reliance on TypeScript, a statically-typed superset of JavaScript. Despite this, Angular offers extensive documentation and resources, promoting best practices and consistency in large teams and complex projects.

In comparison: Vue.js is generally more accessible for beginners due to its simpler syntax and flexibility. Angular, with its comprehensive documentation and enforced best practices, is better suited for developers experienced with TypeScript and large-scale applications.

3) Community Support:

Vue: Vue has a growing and active community with numerous plugins, integrations, and strong support. This community-driven nature ensures a wealth of tutorials, libraries, and tools, contributing to the framework's continuous improvement and rapid response to issues.

Angular: Angular features a large, established community backed by Google, providing extensive resources and a wide array of third-party libraries. The support from Google ensures long-term stability and robust development, making Angular a reliable choice for enterprise applications.

In comparison: Angular benefits from a larger, more established community and corporate backing, offering extensive resources and stability. Vue, while rapidly gaining traction, has a supportive community and is quickly growing in popularity.

4) Ecosystem:

Vue: Vue offers a flexible ecosystem that integrates seamlessly with other libraries. It includes tools like Vue CLI for project scaffolding, Vue Router for routing, and Vuex for state management, allowing developers to choose tools best suited to their project needs.

Angular: Angular provides a comprehensive suite of built-in tools designed for efficient development, such as Angular CLI for project management and Angular Material for UI components. This integrated toolset ensures a consistent development experience without the need for extensive

configuration or third-party tools.

In comparison: Vue excels in flexibility, allowing developers to tailor the development environment to their specific needs. Angular offers a more integrated and consistent development experience with its comprehensive toolset, beneficial for large-scale, enterprise-level applications.

III. CASE STUDY

A. Introduction of the Case Study

In this case study two versions of a web application called "ProTrack" will be compared that has been developed using two different web frameworks, Angular and Vue. ProTrack is a project management tool that allows users to track, manage and edit project work.

1) Objective: The objective of this case study is to compare Angular and Vue in the context of project management applications. The focus will be on three primary functionalities: user registration, project creation, and for each project. By developing and analyzing these applications, we aim to highlight the strengths and weaknesses of each framework in practical use.

2) Application Description:

a) User Registration and Login:

Implementation in Angular: In Angular, user registration and login were implemented using Angular Forms and an authentication service. Angular's template-driven forms were employed to handle user input, and validation was managed through built-in validators. Authentication was handled using JSON Web Tokens (JWT) to ensure secure access.

Implementation in Vue: In Vue, user registration and login were implemented using Vue Forms and Vuex for state management. The form components handled user input and validation within the components themselves. Vuex manages the user state, including authentication tokens.

b) Project Management:

Project Creation:

Angular: Project creation in Angular was achieved using Reactive Forms and services. Reactive Forms allowed for complex form validation and dynamic form control. The data handling was managed through Angular services, ensuring a clean separation of concerns.

Vue: In Vue, project creation utilized Vue Forms and Vuex. The form logic was encapsulated within the components, while Vuex managed the state of the project data, ensuring consistency across the application.

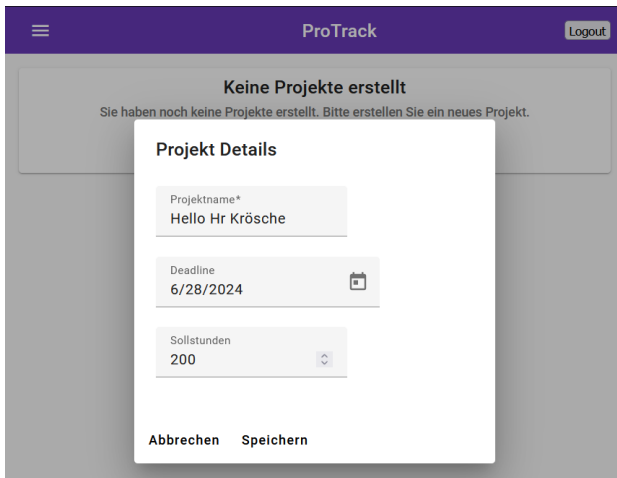


Fig. 2. Project Time running

Start and Stop Functionality:

Angular: The start and stop functionality for time tracking in Angular was implemented using a timer service and RxJS for managing time intervals. This approach allowed for reactive programming, enabling real-time updates and precise time tracking.

Vue: In Vue, the time tracking functionality was built using a timer component and Vuex for state management. The component handled the timer logic, while Vuex ensured that the state was updated accurately and consistently.

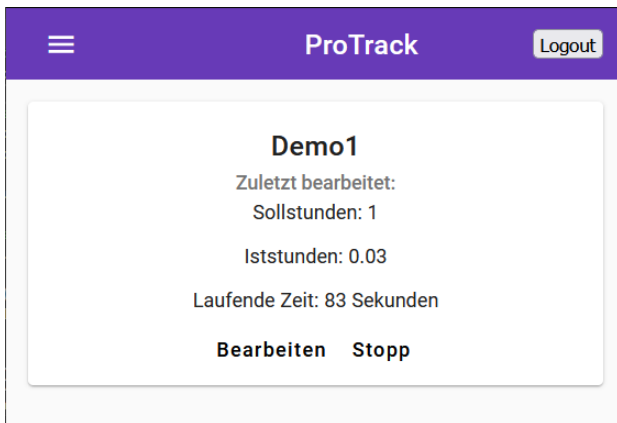


Fig. 3. Project Time running

B. Technical Implementation

a) Application Architecture:

Angular: The Angular application followed a component-based architecture with modules and services. This hierarchical structure ensured maintainability and scalability. Strong typing with TypeScript added an additional layer of robustness.

Vue: The Vue application also utilized a component-based architecture with Vuex for centralized state management. Vue's flexible structure allowed for rapid development and easy integration of new features.

b) State Management:

Angular: State management in Angular was handled using services and BehaviorSubjects from RxJS. This approach facilitated reactive updates and a clear separation of business logic.

Vue: Vuex store was used for state management in Vue, providing a single state tree and using mutations for state changes.

c) Routing:

Angular: The Angular Router was used for navigation, with lazy loading to improve performance. Route guards were implemented to secure routes and manage user access.

Vue: Vue Router was employed in the Vue application, with dynamic imports to optimize loading times. Navigation guards ensured that only authenticated users could access certain routes.

C. Performance and Efficiency

1) Loading Times and Responsiveness:

Angular: Performance metrics indicated that Angular had an up to two seconds longer initial load time due to its larger bundle size. However, once loaded, the application was highly responsive and efficient.

Vue: Vue exhibited faster initial loading times and smaller bundle sizes, contributing to a smoother user experience from the outset. Its lightweight nature made it particularly suitable for rapid development cycles.

D. Challenges and Solutions

1) Common Challenges:

State Management and Routing: Both frameworks encountered challenges in synchronizing state and managing navigation scenarios, such as displaying the specific data content of a particular project. Angular leveraged RxJS, while Vue utilized Vuex, both providing robust solutions to these issues.

2) Framework-Specific Challenges:

Angular: The main challenge with Angular was its complexity and boilerplate code, which required a steeper learning curve and more initial setup time.

Vue: Vue's flexibility sometimes led to inconsistencies. Ensuring code quality and maintaining state consistency were key challenges.

E. Comparison and Analysis

1) Advantages and Disadvantages:

Angular: Angular's strengths lie in its stability and comprehensive ecosystem, making it ideal for large-scale enterprise applications. However, its verbose syntax and higher initial complexity can be drawbacks.

Vue: Vue's simplicity and flexibility make it suitable for rapid development and smaller projects. Its main weakness is a less extensive ecosystem compared to Angular.

IV. CONCLUSION AND RECOMMENDATIONS

Choosing the right framework depends largely on the specific needs of a project. Vue.js offers simplicity and flexibility, making it ideal for smaller projects and rapid integration, while Angular provides a comprehensive solution suitable for large-scale applications with its robust features and strong support from Google. Understanding the strengths and weaknesses of each framework can help developers make informed decisions to best meet their project requirements.

A. Summary

Angular and Vue are tailored for different project scales and requirements. Angular is well-suited for large-scale, complex applications, benefiting from its comprehensive features and strong backing by Google. It offers a well-defined architecture, extensive libraries, and a highly structured environment that enhances maintainability and scalability. On the other hand, Vue is perfect for smaller, agile projects that require quick setups and less rigid structural constraints. Vue's simplicity, ease of use, and flexible integration make it highly attractive for rapid development cycles and dynamic project requirements.

B. Recommendations

For enterprise-level applications that demand robust architecture and extensive features, Angular is the recommended choice. It offers a structured environment, extensive tooling, and long-term support, making it suitable for applications that require high performance, security, and maintainability. Angular's strong typing with TypeScript, dependency injection, and comprehensive documentation support the development of large, complex applications with consistent coding practices.

For projects that prioritize rapid prototyping and require a high degree of flexibility, Vue is more appropriate. Its learning curve is shorter, and it offers ease of integration with existing projects, making it ideal for startups, small teams, and projects with evolving requirements. Vue's component-based architecture and reactivity system enable quick iteration and adaptation, which are crucial for fast-paced development environments. Additionally, developers should consider the following factors when choosing between Angular and Vue.

1) Team Expertise: If the development team has prior experience with TypeScript and prefers a strongly-typed language, Angular might be more suitable. TypeScript's static typing can reduce errors and improve code quality. Conversely, if the team is looking for simplicity and ease of use, Vue's intuitive design can be advantageous. Vue's flexibility and ease of learning make it accessible to developers with varying levels of experience.

2) Project Complexity: For highly complex projects with numerous components and dependencies, Angular's structured approach and built-in features like dependency injection can be beneficial. Angular's modularity and separation of concerns support maintainability in large applications. Vue's modularity and lightweight nature are better suited for simpler, less

complex projects. Its ease of integration with other libraries and tools can streamline development for smaller applications.

3) Community and Ecosystem: Angular's larger community and extensive ecosystem provide robust support and numerous third-party libraries, which can be crucial for large-scale projects. The strong corporate backing from Google ensures ongoing improvements and stability. Vue, while growing rapidly, offers a more community-driven ecosystem, which is advantageous for projects that benefit from rapid innovation and community contributions. Vue's community is highly active, contributing to a wealth of plugins and extensions that enhance its functionality.

C. Future Outlook

Future research should focus on the evolving best practices in using these frameworks and how they affect development efficiency and performance. This could provide deeper insights into which frameworks are best suited for various types of projects as technology and market demands continue to evolve.

1) Performance Improvements: Investigating how future updates to Angular and Vue impact their performance, particularly in terms of load times, responsiveness, and resource utilization, will help developers make more informed decisions. Performance benchmarks and comparative studies can provide practical insights into optimizing applications for better user experiences.

2) Tooling and Integration: As new tools and plugins emerge, understanding how they integrate with Angular and Vue can provide valuable information on enhancing development workflows and project outcomes. Evaluating the efficiency and ease of use of these tools can support more effective and streamlined development processes.

3) Case Studies and Real-World Applications: Conducting more case studies on different types of projects (e.g., e-commerce, social media, enterprise applications) can offer practical insights into the strengths and weaknesses of each framework in real-world scenarios. Detailed analyses of project implementations can highlight best practices and common pitfalls.

4) Developer Experience: Evaluating how changes in these frameworks impact developer productivity, satisfaction, and learning curves can help guide the choice of framework based on team dynamics and project requirements. Surveys and interviews with developers can provide qualitative data on the usability and developer experience of each framework.

By continuously monitoring and evaluating these aspects, developers can better adapt to the rapidly changing landscape of web development and make informed decisions that align with their project goals and organizational needs. Staying informed about advancements in these frameworks and their

ecosystems will enable developers to leverage the latest features and best practices, ensuring optimal performance and project success.

REFERENCES

- [1] Ionos, “Webframeworks – Überblick und Klassifizierung,” <https://www.ionos.de/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/>, 2022.
- [2] Ionos, “Die beliebtesten JavaScript-Frameworks und -Bibliotheken,” <https://www.ionos.at/digitalguide/websites/web-entwicklung/beliebte-javascript-frameworks-und-bibliotheken/>, 2023.
- [3] Vue.js, “Einführung,” <https://vuejs.org/guide/introduction.html>, abgerufen 31.05.2024.
- [4] Kinsta, “Angular vs. Vue: Ein Kopf-an-Kopf-Vergleich,” <https://kinsta.com/de/blog/angular-vs-vue/#angular-vs-vue-hnlichkeiten-und-gemeinsamkeiten>, 2023.
- [5] MDN Web Docs, “Introduction to the DOM,” https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction, 2024.
- [6] Mozilla Developer Network, “JavaScript Guide,” <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>, accessed 25.06.2024.
- [7] MoldStud, “Creating Dynamic Web Content with JavaScript and jQuery,” <https://moldstud.com/articles/p-creating-dynamic-web-content-with-javascript-and-jquery>, 2024.
- [8] Angular Team, “Angular,” <https://angular.io/>, 2024.
- [9] Vue.js Contributors, “Vue.js,” <https://vuejs.org/>, 2024.
- [10] United Nations Conference on Trade and Development, “Digital Economy Report 2019,” <https://unctad.org/publication/digital-economy-report-2019>, 2019.
- [11] D. P. Mishra, K. K. Rout, and S. R. Salkuti, “Modern tools and current trends in web-development,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 24, no. 2, pp. 978–985, 2021.
- [12] X. Mao and J. Xin, “Developing Cross-platform Mobile and Web Apps,” *CIGR Proceedings*, 2014.
- [13] D. Verma, “A comparison of web framework efficiency: performance and network analysis of modern web frameworks,” 2022.
- [14] J. Cincović and M. Punt, “Comparison: Angular vs. React vs. Vue. Which framework is the best choice,” *Universidad de Belgrade*, 2020.
- [15] Angular, “Getting started with Angular,” <https://angular.io/docs>, 2024.
- [16] Vue, “Introduction to Vue.js,” <https://vuejs.org/v2/guide/>, 2024.
- [17] OpenJS Foundation and jQuery contributors, “jQuery,” 2024.