Angular vs Vue

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

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

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

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

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 [**?**].

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 [**?**].

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 [**?**].

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) 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 [**?**].

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 [?].

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 [?].

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 [?]. 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 [?].

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

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 [?]

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 [?], [?].

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 [?].

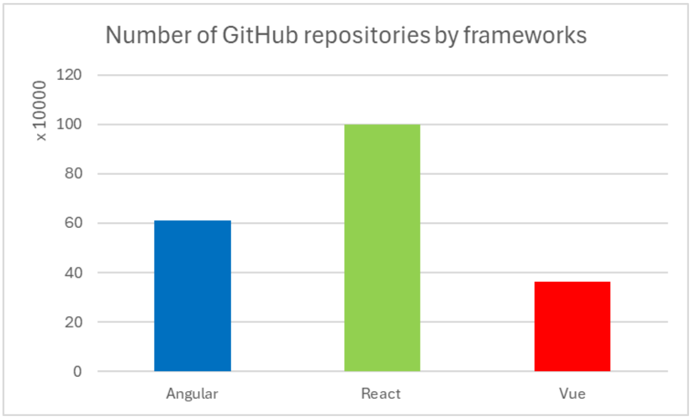

Fig. 1. Number of GitHub repositories by frameworks [?]

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.js 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 [?].

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 [?].

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 compoared based on several parameters to provide a clear understanding of their respective strengths and weaknesses.

1) Performance: Vue: uses a virtual DOM for efficient updates and is generally faster in rendering due to its lightweight nature.
Angular: utilizes a real DOM, which can be slower in certain scenarios, and is optimized for performance with techniques like ahead-of-time (AOT) compilation.
In comparison: Vue often achieves better performance for simple to moderately complex applications. When comparing the two frameworks, Angular provides robust performance optimizations suitable for enterprise-level applications.

2) Developer-Friendliness: Vue: is easy to learn and integrate, featuring a simple syntax and flexible structure along with comprehensive documentation.
Angular: has a steeper learning curve due to its complexity and requires understanding of TypeScript, but offers extensive documentation and resources.
In comparison: Vue.js is generally more accessible for beginners, while Angular is more suitable for developers with experience in TypeScript and complex frameworks.

G. Community Support

Vue: boasts a growing community with increasing adoption, complemented by numerous plugins and integrations. It benefits from active development and strong community support.
Angular: features a large and established community with extensive resources and a wide array of third-party libraries. Supported by Google, Angular ensures long-term support and robust development.
In comparison: Angular enjoys a larger and more established community base. Meanwhile, Vue is rapidly gaining traction and fostering a supportive community environment.

H. Ecosystem

Vue: offers flexibility and seamless integration with other libraries. It provides Vue CLI for streamlined project scaffolding, along with Vue Router and Vuex for effective state management.
Angular: features a comprehensive suite of built-in tools designed for efficient development. It utilizes Angular CLI for seamless project management and includes Angular Material for building sophisticated UI components.
In comparison: Vue excels in flexibility, allowing developers to choose from a variety of tools and libraries. Meanwhile, Angular offers a more integrated and consistent development experience with its comprehensive toolset.

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.

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.

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. On the other hand, Vue is perfect for smaller, agile projects that require quick setups and less rigid structural constraints.

B. Recommendations

For enterprise-level applications that demand robust architecture and extensive features, Angular is the recommended choice. For projects that prioritize rapid prototyping and require a high degree of flexibility, Vue is more appropriate.

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.

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first . . ."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.