# Angular vs Vue

Cornelia Langsenlehner, Sandra Hartlauer, Michael Nußbaumer


BACHELORARBEIT

eingereicht am

Fachhochschul-Bachelorstudiengang

Mobile Computing

in Hagenberg

im Mai 2024

Advisor:

FH-Prof. Dr-Ing. Krösche Jens

# Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed copy is identical to the submitted electronic version.

Hagenberg, May 22, 2024

Cornelia Langsenlehner, Sandra Hartlauer, Michael Nußbaumer

# Contents

# Preface

This is version **2023/11/06** of the LaTeX document template for various theses at the School of Informatics, Communication and Media at the University of Applied Sciences Upper Austria in Hagenberg. We are pleased to learn that this document collection is meanwhile also used at various other institutions in Austria and abroad.

The document was initially created in response to requests from students after the 2000/01 academic year when an official LaTeX introductory course was offered in Hagenberg for the first time. The fundamental idea was to "simply" convert the already existing *Microsoft Word* template for diploma theses to LaTeX and possibly to add some unique features. This quickly turned out to be not very useful since LaTeX, especially concerning the handling of literature and graphics, requires a substantially different way of working. The result is— rewritten from scratch and much more extensive than the previous document—a manual for writing with LaTeX, supplemented with additional (meanwhile removed) hints for *Word* users. Technical details of the current version can be found in Appendix.

While this document was initially intended exclusively for the preparation of diploma theses, it now also covers *master theses*, *bachelor theses*, and *internship reports*. The differences between these documents have been deliberately kept small.

When creating this template, an attempt was made to work with the basic functionality of LaTeX and—as far as possible—to achieve this without additional packages. This was only partially successful; several supplementary "packages" are necessary, but only common extensions have been used. Of course, there is a large number of additional packages which can be helpful for further improvements and refinements. Everyone is encouraged to experiment with these as soon as they have the necessary self-confidence and sufficient time to experiment. Many details and tricks are not explicitly mentioned in this document but can be explored in the underlying source code at any time.

Numerous colleagues have provided valuable support through careful proofreading and constructive suggestions for improvement. We thank Heinz Dobler for consistently improving our "computer slang" and Elisabeth Mitterbauer for her proven "orthographic eye".

Usage of this template is free without any restrictions and not bound to any mention. However, when used as a basis for one's work, one should not simply start working on it, but at least *read* the essential parts of the document and, if possible, take them to heart. Experience has shown that this improves the quality of the results significantly.

This document and the associated LaTeX classes have been available since November

2017 on CTAN[1] as package `hagenberg-thesis`,

https://ctan.org/pkg/hagenberg-thesis.

The current source code, as well as additional materials—such as a wiki with instructions for the integration of often requested functionalities and extensions—can be found at

https://github.com/Digital-Media/HagenbergThesis.[2]

Despite great efforts, a document like this always contains errors and shortcomings. Comments, suggestions, and helpful additions are welcome. Ideally, as comments or issues on GitHub.

By the way, here, in the preface (which is common in diploma and master theses but dispensable for bachelor's theses), you may briefly describe the genesis of the document. This is also the place for any acknowledgments (e.g., to the supervisor, the examiner, the family, the dog, etc.) as well as dedications and philosophical remarks. These should be balanced and limited to a maximum of two pages.

W. Burger (em.) and W. Hochleitner

University of Applied Sciences Upper Austria
Department of Digital Media, Hagenberg
https://www.fh-ooe.at/campus-hagenberg/

[1] Comprehensive TeX Archive Network

[2] https://github.com/Digital-Media/HagenbergThesis/blob/main/CHANGELOG.md contains a list of chronological changes (formerly included in the appendix of this document).

# Abstract

Here goes an abstract of the work, with a maximum of 1 page. Unlike other chapters, the abstract is usually not divided into sections and subsections. Footnotes are also not used here.

By the way, abstracts are often included in literature databases with the author and title of the work. It is, therefore, essential to ensure that the information in the abstract is coherent and complete in itself (i.e., without other parts of the work). In particular, *no literature references* are typically used at this point (as is the case also in the *title* of the thesis and the German *Kurzfassung*)! If such is needed—for example, because the paper is a further development of a particular, earlier publication—then *full* references are necessary for the abstract itself, e.g., [Zobel J.: *Writing for Computer Science – The Art of Effective Communication*. Springer, Singapore, 1997].

It should also be noted that special characters or list items are usually lost when records are added to a database. The same applies, of course, to the German *Kurzfassung*.

In terms of content, the abstract should not be a list of the individual chapters (the introduction chapter is intended for this purpose). However, it should provide the reader with a concise summary of your thesis. Therefore, the structure used here is necessarily different from that used in the introduction.

# Kurzfassung

Dies sollte eine maximal 1-seitige Zusammenfassung Ihrer Arbeit in deutscher Sprache sein.

The German “Kurzfassung” should contain the same content as the English abstract. Therefore, try to translate the abstract precisely but not word for word. When translating, remember that certain idioms from English have no counterpart in German or must be formulated differently. Also, word order in German is very different from English. Without knowledge of the German language, it is acceptable to resort to translators. Nevertheless, hiring a skillful person for proofreading is recommended even with the highest confidence in one’s German knowledge.

The correct translation for “diploma thesis” is *Diplomarbeit*, a “master thesis” is called *Masterarbeit*. For “bachelor’s thesis”, *Bachelorarbeit* is the appropriate translation.

By the way, for this section, the *language setting* in LaTeX should be switched from English to German to get the correct form of hyphenation. However, the correct quotation marks must be set manually.

# Chapter 1

# Introduction - M

## 1.1 Motivation

The motivation for this analysis lies in the increasing importance of web applications in the modern digital landscape. Businesses and services are increasingly migrating to online platforms, thereby increasing the demand for efficient, user-friendly, and powerful web applications. The platform independence of web applications enables developers to reach a broad user base without needing to create separate applications for each operating system or device [8].

## 1.2 Challenges

### 1.2.1 Hardware Platforms

Developing web applications must support a variety of hardware platforms, including desktop computers, laptops, tablets, and mobile devices. The challenge is to ensure that applications perform consistently and efficiently across different devices [5].

### 1.2.2 Frameworks

The selection and management of frameworks significantly influence the efficiency and scalability of a web application. Different frameworks offer various advantages and disadvantages in terms of performance, maintainability, and user-friendliness [9].

### 1.2.3 Evaluation Methods

Evaluation methods are crucial for assessing the performance, security, and user-friendliness of web applications. This includes benchmarks, test scenarios, and other empirical methods to verify application quality [9].

## 1.3 Goals

This analysis aims to provide a comprehensive overview of both frameworks and serve as a decision-making guide by offering valuable insights into the strengths and weak-

nesses 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.

## 1.4 Structure

# Chapter 2

# Theoretical Background - S

## 2.1 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 [4]. Pros and Cons of Web Application Frameworks

Web frameworks offer many advantages, but they also come with some disadvantages.

**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 [4].

**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 [4].

## 2.2 JavaScript for Complex Web Applications 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 [3].

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 [6]. Leveraging JavaScript frameworks and libraries facilitates the manipulation of the DOM, streamlining the creation of dynamic web content.

### 2.2.1 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.

### 2.2.2 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 [7].

By leveraging JavaScript and jQuery for dynamic web content, websites can offer personalized, engaging experiences that cater to modern user expectations [12].

**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 [13].

## 2.3 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.

### 2.3.1 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 [14].

**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.

### 2.3.2 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 [11].

**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.

## 2.4 Comparison Criteria - M

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.

### 2.4.1 Performance

Angular provides robust performance and scalability for complex applications. In contrast, Vue stands out for its fast loading times and low overhead, making it particularly attractive for smaller projects [9].

### 2.4.2 Developer-friendliness

Angular offers a robust structure and comprehensive features, which come with a steeper learning curve but are supported by extensive documentation and a strong community. Vue, on the other hand, is characterized by its easy integration and intuitive syntax, facilitating quicker onboarding and flexibility for smaller projects [1, 10].

### 2.4.3 Community Support

Active and engaged community support is crucial for framework development and bug fixing. Referring to the study by Jelica Cincović and Marija Punt titled "Comparison: Angular vs. React vs. Vue," which examined community support through GitHub repository comparisons, Figure illustrates the following:

The graph indicates that React enjoys the largest community support, followed by Angular in second place and Vue in third [2].

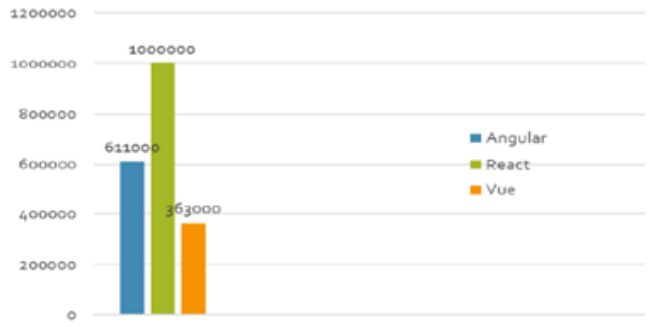


**Figure 2.1:** Number of GitHub repositories by frameworks

## 2.5 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.

### 2.5.1 Performance

**Vue:**

- Uses a virtual DOM for efficient updates
- Generally faster in rendering due to its lightweight nature

**Angular:**

- Real DOM, which can be slower in certain scenarios
- Optimized for performance with techniques like ahead-of-time (AOT) compilation

**Comparison:**

- Vue.js often has better performance for simple to moderately complex applications.
- Angular provides robust performance optimizations suitable for enterprise-level applications.

### 2.5.2 Developer-Friendliness

**Vue:**

- Easy to learn and integrate
- Simple syntax and flexible structure
- Comprehensive documentation

**Angular:**

- Steeper learning curve due to its complexity
- Requires understanding of TypeScript
- Extensive documentation and resources

**Comparison:**

- Vue.js is generally more accessible for beginners.

- Angular is more suitable for developers with experience in TypeScript and complex frameworks.

### 2.5.3 Community Support

**Vue:**

- Growing community with increasing adoption
- Numerous plugins and integrations
- Active development and support

**Angular:**

- Large and established community
- Extensive resources and third-party libraries
- Backed by Google, ensuring long-term support

**Comparison:**

- Angular has a larger and more established community.
- Vue.js is rapidly growing and has a supportive community.

### 2.5.4 Ecosystem

**Vue:**

- Flexible and integrates well with other libraries
- Vue CLI for project scaffolding
- Vue Router and Vuex for state management

**Angular:**

- Comprehensive suite of built-in tools
- Angular CLI for efficient development
- Angular Material for UI components

**Comparison:**

- Vue.js offers flexibility in choosing tools and libraries.
- Angular provides a more integrated and consistent development experience.

## 2.6 Conclusion

Choosing the right framework depends on the specific needs of a project. Vue.js offers simplicity and flexibility, making it ideal for smaller projects and quick integration. 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.

# Chapter 3

# Case Study - C

## 3.1 Introduction

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.

### 3.1.1 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.

### 3.1.2 Application Description

#### 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.

#### 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.

## 3.2 Technical Implementation

### 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.

### 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.

### 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.

## 3.3 Performance and Efficiency

### 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.

## 3.4 Challenges and Solutions

### Common Challenges

Hier ist die umformulierte Version des Textes auf Englisch:

*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.

### 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.

## 3.5 Comparison and Analysis

### 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.

# Chapter 4

# Conclusion and Recommendations - C

## 4.1 Summary

Angular and Vue each have unique strengths and are suitable for different types of projects. Angular excels in large-scale, complex applications, while Vue is ideal for smaller, agile developments.

## 4.2 Recommendations

Based on this case study, Angular is recommended for enterprise-level applications requiring robust architecture and extensive features. Vue is recommended for projects needing quick prototyping and flexibility.

## 4.3 Future Outlook

Future research could explore the evolving best practices in both frameworks and their impact on development efficiency and performance.

# References

## Literature


[1] Angular. *Getting started with Angular*. https://angular.io/docs. Accessed: 2024-06-24. 2024 (cit. on p. 6).

[2] Jelica Cincovi'c and Marija Punt. "Comparison: Angular vs. React vs. Vue. Which framework is the best choice". *Universidad de Belgrade* (2020) (cit. on p. 6).

[3] Ionos. "Die beliebtesten JavaScript-Frameworks und -Bibliotheken" (2023). Abgerufen am 31.05.2024. URL: https://www.ionos.at/digitalguide/websites/web-entwicklung/beliebte-javascript-frameworks-und-bibliotheken/ (cit. on p. 4).

[4] Ionos. "Webframeworks – Überblick und Klassifizierung" (2022). Abgerufen am 31.05.2024. URL: https://www.ionos.de/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/ (cit. on pp. 3, 4).

[5] Xiang Mao and Jiannong Xin. "Developing Cross-platform Mobile and Web Apps". *CIGR Proceedings* (2014) (cit. on p. 1).

[6] MDN Web Docs. "Introduction to the DOM". *MDN Web Docs* (2024). URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (visited on 06/25/2024) (cit. on p. 4).

[7] OpenJS Foundation and jQuery contributors. "jQuery" (2024). URL: https://jquery.com/ (cit. on p. 5).

[8] United Nations Conference on Trade and Development. "Digital Economy Report 2019" (2019). Abgerufen am 31.05.2024. URL: https://unctad.org/publication/digital-economy-report-2019 (cit. on p. 1).

[9] Dhruv Verma. "A comparison of web framework efficiency: performance and network analysis of modern web frameworks" (2022) (cit. on pp. 1, 6).

[10] Vue. *Introduction to Vue.js*. https://vuejs.org/v2/guide/. Accessed: 2024-06-24. 2024 (cit. on p. 6).


## Online sources


[11] Vue.js Contributors. *Vue.js*. Accessed: 2024-06-25. 2024. URL: https://vuejs.org/ (visited on 06/25/2024) (cit. on p. 5).

[12] MoldStud. *Creating Dynamic Web Content with JavaScript and jQuery*. Feb. 2024. URL: https://moldstud.com/articles/p-creating-dynamic-web-content-with-javascript-and-jquery (visited on 06/25/2024) (cit. on p. 5).

[13] Mozilla Developer Network. *JavaScript Guide*. Accessed: 2024-06-25. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (cit. on p. 5).

[14] Angular Team. *Angular*. Accessed: 2024-06-25. 2024. URL: https://angular.io/ (visited on 06/25/2024) (cit. on p. 5).