CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A learning curve comparison between React and Angular

Analyzing documentation, ease of use, community support, and modularity

Degree project report in Computer Engineering

Johan Franz
Milton Niklasson

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024
www.chalmers.se

Degree project report 2024

A learning curve comparison between React and Angular

Analyzing documentation, ease of use, community support, and modularity

Johan Franz
Milton Niklasson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

A learning curve comparison between React and Angular
Analyzing documentation, ease of use, community support, and modularity
JOHAN FRANZ, MILTON NIKLASSON

Supervisor: Jonas Almström Duregård, Dept. of Computer Science and Engineering
Examiner: Lars Svensson, Dept. of Computer Science and Engineering

Degree project report 2024
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: N/A

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

A learning curve comparison between React and Angular
Analyzing documentation, ease of use, community support, and modularity
JOHAN FRANZ
MILTON NIKLASSON
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

In the ever-expanding world of JavaScript frameworks, it can be challenging to determine which framework is best suited for university students venturing into web applications. This degree project report evaluates two of the most prominent frameworks, React and Angular, by developing a web application which needs to display graphs of scientific data. The study focuses on the most critical qualities for web development: documentation, ease of use, community support, and modularity. For graphing scientific data, the evaluation is based on user inputs, specifically from researchers, as well as the scalability and functionality each framework can provide. The study involved developing a web application using both frameworks, with the objective of creating the same application twice and then assessing the development process. The goal was to provide computer engineering students with a deeper understanding of the development and challenges associated with using two of the most prominent frameworks in the industry. The results highlighted that React outperformed Angular in several key areas, particularly in ease of use and community support, with React achieving an average score of 9.25 compared to Angular's 6.25. React's documentation and structured approach were more intuitive and beneficial for developers with a university-level background. While both frameworks required the use of D3.js for graphing capabilities, React's integration process was smoother and more efficient. These findings aim to guide students in choosing the most appropriate framework for their needs in web application development, emphasizing React as the more suitable choice for those seeking a straightforward and well-supported framework.

Keywords: Web development, Framework comparison, React, Angular

Acknowledgements

We would like to express our sincere gratitude to our product owners, Ebba Sandbecker, André Persson, and Sebastian Samuelsson, for providing us the opportunity to undertake our degree project under their guidance. We also wish to acknowledge the contributions of the electrical engineering partners, Jimmy Brevitz and Vincent Olofsson, who developed the hardware for the web application. Furthermore, we extend our thanks to the researchers, Munis Khan and August Yurgens, for their invaluable assistance in defining the graphing criteria.

Johan Franz
Milton Niklasson
Gothenburg, May, 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this degree project report listed in alphabetical order:

CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
JSX	JavaScript XML
RBAC	Role-Based Access Control
SVG	Scalable Vector Graphics
UI	User Interface
XML	Extensible Markup Language

Contents

List of Tables

1
Introduction

This chapter presents the background of the degree project and the purpose of this report. Additionally, relevant information regarding the scientific data collected from the graphene biosensor will be provided.

1.1 Background

This report is closely associated with a degree project conducted concurrently. The idea for the degree project originated from LayerLogic, a newly founded company supported by Chalmers Ventures, which focuses on detecting listeria in the fish production industry. Researchers at LayerLogic have developed an innovative biosensor that utilizes an atom-thick layer of graphene to measure the resistance of a substance, thereby determining the presence of listeria.

However, the company lacks a software solution to interface with the biosensor and store all test data. Consequently, the degree project aims to develop a website with an associated database that can read from and write to the biosensor, as well as store completed test results in the database.

Given that the scope of the degree project does not generate substantial report material, this report will instead focus on comparing two different frameworks for creating websites and determining the best way to present the test data collected from the biosensor to researchers.

1.2 Purpose

The purpose of this report is to address the following research questions:

- **Research Question 1 (RQ1):** "Which front-end JavaScript library, React or Angular, offers the most straightforward implementation process and presents the least challenging learning curve for website development?"
- **Research Question 2 (RQ2):** "Which front-end JavaScript library, React or Angular, provides the most efficient tools for intuitively visualizing scientific data?"

1.3 Research Context

To address the first research question (RQ1), we will establish our own criteria for comparison due to the absence of standardized methodologies. The criteria include:

1. **Relevant Features:** We will evaluate the features offered by each framework, acknowledging that a greater number of features does not necessarily equate to better usability.

2. **Data Collection and Presentation:** We will assess how effectively each framework can collect and present data. This criterion will also relate to RQ2 and will be evaluated by a group of potential users and customers.

3. **Learning Curve:** We will determine how easily university students can learn each framework, based on the quality of documentation and the complexity of the framework.

For the second research question (RQ2), the criteria will be focused on customer and user satisfaction. The web application must cater to both customers and researchers, ensuring that scientific data is presented in a comprehensible manner for both experts and non-experts. Feedback from research groups will serve as the primary criterion for evaluation. Given the limited time frame, this investigation will involve a specialized group to gauge their perception of the scientific data presentation.

1.4 Limitations

Due to the significant portion of the project's time frame dedicated to programming and learning the different frameworks, the project team will not conduct original research on what makes a framework easy to learn. Instead, the team will rely primarily on existing research and subjective opinions to guide their actions and comparisons.

Sara Abrahamsson's thesis on the comparison of React, Angular, and Vue provided a baseline for our report due to its relevance to RQ1 [1]. Additionally, we referenced existing research on comparing different frameworks [2].

1.5 Scientific Data from Graphene Biosensor

The scientific data collected from the graphene biosensor is crucial for detecting listeria. This sensor, developed by LayerLogic, utilizes a graphene layer to measure the electrical resistance of samples, allowing researchers to determine the presence of listeria accurately. The software solution we develop will enable seamless interaction with the biosensor and efficient storage and retrieval of test data, ultimately aiding researchers in their efforts to ensure food safety in the fish production industry.

2
Technical Background

This chapter will present the relevant technical background needed to understand the degree project and the comparison between the different frameworks.

2.1 Web Application

A web application is a software program that runs on a web server and is accessed by users through a web browser over the internet. Unlike traditional desktop applications, web applications do not need to be installed on the user's device and can be used on any device with a compatible browser. Web applications typically involve a combination of frontend and backend technologies to deliver interactive, dynamic, and responsive user experiences [3].

2.2 HTML

HTML, or HyperText Markup Language, is the standard language used for creating and structuring content on the web. It provides the basic building blocks for web pages, allowing developers to define elements such as headings, paragraphs, links, images, and other multimedia. HTML is the foundation upon which web pages are built, and it works in conjunction with CSS and JavaScript to create fully functional and visually appealing websites [4].

2.3 CSS

CSS, or Cascading Style Sheets, is a stylesheet language used for describing the presentation and layout of web pages. While HTML provides the structure and content, CSS is used to control the visual appearance, including colors, fonts, spacing, and positioning of elements. By separating content from design, CSS enables developers to maintain consistency across multiple pages and makes it easier to update and manage the look and feel of a website [5].

2.4 JavaScript

JavaScript is a high-level, interpreted programming language and one of the foundational technologies of the World Wide Web. It powers interactive web pages and

is crucial for web applications. JavaScript is dynamic, weakly typed, and prototype-based, supporting both object-oriented and functional programming styles. Originally designed to enhance web page interactivity, JavaScript has since evolved to run on the server side through platforms like Node.js. Its weak typing makes it more forgiving and easier to learn, though this can sometimes obscure problematic code [6].

2.5 TypeScript

TypeScript, developed by Microsoft, is a statically typed superset of JavaScript. It introduces optional static types, interfaces, and other enhancements to boost developer productivity and code reliability. TypeScript code compiles to plain JavaScript, ensuring compatibility with any JavaScript runtime environment. While TypeScript promotes robust and maintainable code, particularly in large-scale applications though its adoption may require some adjustment period [7].

2.6 JSX

JSX, or JavaScript XML, is a syntax extension for JavaScript. JSX allows developers to write HTML elements directly within JavaScript code, providing a more intuitive and seamless way to structure component-based UI development. By using JSX, developers can create elements that resemble HTML, but with the full power and flexibility of JavaScript, enabling dynamic and interactive user interfaces. Under the hood, JSX is transformed into regular JavaScript function calls by a compiler like Babel, facilitating efficient rendering and updates in the browser. This combination of HTML-like syntax and JavaScript logic simplifies the process of creating complex user interfaces and enhances code readability and maintainability [8].

2.7 Document Object Model (DOM)

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of HTML or XML documents as a tree-like structure, where each node represents a part of the document (e.g., elements, attributes, text). The DOM provides a way for programs to dynamically access and manipulate the content, structure, and style of web documents. It forms the backbone of client-side web development, enabling interactions such as dynamic content updates, event handling, and animation [9].

2.8 Virtual DOM

The Virtual DOM is a concept used in some modern web development frameworks. It is a lightweight, in-memory representation of the actual DOM tree. When changes are made to a component, instead of directly updating the real DOM, the framework first updates the Virtual DOM. The updated Virtual DOM is then compared with

the previous version to determine the minimal set of changes needed to update the real DOM. This process, known as reconciliation, minimizes DOM manipulations and improves performance [10].

2.9 React

React is an open-source JavaScript library for building user interfaces, particularly single-page applications. Developed by Facebook in 2013, it allows developers to create reusable UI components. React uses a virtual DOM to efficiently update and render components in response to data changes. Key features of React include JSX, a syntax extension for JavaScript that looks similar to HTML, and a component-based architecture, which promotes modularity and reusability in web development [11].

Ecosystem and Community Support
React's ecosystem includes tools like Create React App, React Router, and state management libraries like Redux. It boasts a large community and a vast array of third-party libraries and resources.

Performance Considerations
React's virtual DOM enhances performance by minimizing direct DOM manipulations. Techniques like code splitting and lazy loading are commonly used to improve application performance.

Learning Curve and Documentation
React has a relatively gentle learning curve for JavaScript developers and is supported by an abundance of tutorials and documentation.

Use Cases and Industry Adoption
React is widely adopted by companies like Facebook, Instagram, and Netflix, and is particularly well-suited for dynamic, high-performance user interfaces.

2.10 Redux

Redux is a predictable state management library for JavaScript applications, commonly used with React to manage the state of the application in a more structured and maintainable way. Redux centralizes the application's state in a single store, which serves as the single source of truth for all state changes. This approach simplifies state management, especially in large and complex applications where state needs to be shared across multiple components [12].

2.11 Angular

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Developed by Google and released in 2016, it provides a comprehensive solution for both the development and deployment of applications. Angular includes a robust set of tools and features, such as two-way data binding, dependency injection, and a modular architecture. It uses TypeScript as its primary language, enhancing the development process with strong typing and improved tooling [13].

Ecosystem and Community Support
Angular's ecosystem includes the Angular CLI, Angular Material for UI components, and a comprehensive set of tools for routing, forms, HTTP client, and more. The framework's extensive nature covers most aspects of web development out of the box.

Performance Considerations
Angular's Ahead-of-Time (AOT) compilation improves performance by compiling the application at build time. The framework's change detection and zones also contribute to efficient performance.

Learning Curve and Documentation
Angular has a steeper learning curve due to its complexity but is supported by extensive official documentation and learning resources provided by Google.

Use Cases and Industry Adoption
Angular is used by companies like Google, Microsoft, and Forbes, and is particularly suitable for enterprise-scale applications that require a robust, all-in-one framework.

2.12 GitHub

GitHub is a web-based platform for hosting repositories and facilitating collaboration among developers. It is used by developers to effortlessly share and work on the same code base over the internet. It offers features such as capability to have different branches from the main code and a history of changes made to the repository with the ability to revert to earlier versions, enabling developers to easily branch out and try new ideas out [14].

2.13 Visual Studios Code (VS Code)

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It provides a highly customizable and lightweight environment for coding, debugging, and version control. It supports a large number of programming languages and features an extensive ecosystem of extensions to enhance productivity and workflow customization [15].

2.14 Frontend

The frontend, also known as the client-side, is the part of a web application that users interact with directly. It encompasses everything that users experience in their web browsers, including the layout, design, and user interface elements such as buttons, forms, and content. The frontend is built using technologies like HTML, CSS, and JavaScript, and is responsible for rendering the visual components and ensuring a responsive and interactive user experience. Frontend development often involves frameworks and libraries such as React, Angular, or Vue.js to create dynamic and efficient web applications [16].

2.15 Backend

The backend, or server-side, is the part of a web application that operates behind the scenes and is not directly accessible by users. It is responsible for managing the application's data, business logic, authentication, and server configuration. The backend interacts with the frontend by processing requests, performing database operations, and sending appropriate responses back to the client. Backend development involves using server-side languages like Node.js, Python, Java, or PHP, along with databases such as MySQL, MongoDB, or PostgreSQL. It ensures that the application runs smoothly, securely, and efficiently [17].

2.16 Firebase

Firebase is a comprehensive platform developed by Google for building web and mobile applications. It provides a suite of tools and services designed to simplify backend development and streamline the integration of various functionalities. Key features of Firebase include a real-time NoSQL database (Firestore), user authentication, cloud storage, hosting, and serverless functions. Firebase enables developers to quickly set up and manage backend infrastructure without the need for extensive server-side coding, allowing them to focus more on building and refining the application's frontend and user experience. Its integration with Google Cloud services further enhances scalability, performance, and analytics capabilities [18].

3
Methods

This chapter describes the methods we will use to establish objective criteria for evaluating the usability of frontend JavaScript libraries.

The report aims to address two key research questions. To achieve this, we will follow the foundation specified by the company (LayerLogic). The company's specifications primarily focus on functionality and user control. To fulfill these specifications, we will conduct a comparison of different frameworks.

3.1 Framework Selection

The project will investigate two of the most popular and widely used frameworks, React and Angular [19]. Initially, the web application will be built using React, followed by its recreation with Angular to facilitate a comparative analysis. The development of the web application will adhere to the rules and constraints outlined in section 1.2. The comparison methodology will involve implementing identical features with both frameworks and presenting the outcomes to a test group. The feedback from the test group will provide two critical insights: the features they find most beneficial and the framework they perceive as the best fit for their experience.

3.2 Method for RQ1

The method for evaluating Research Question 1 (RQ1) will be based on the following criteria, listed in order of importance: Documentation, Ease of Use, Modularity, Community, and Lines of Code.

3.2.1 Documentation

We will assess documentation quality based on public perception and overall satisfaction with the framework's documentation. The documentation selected will be only the official documentation released by the frameworks. The evaluating will be conducted though our perception when using the framework and public's perception.

3.2.2 Ease of Use

Ease of use, though subjective, will be evaluated based on GitHub reviews and our experiences during the web application development. We will compare how

similar features are implemented in React and Angular, assessing the complexity and structure of the implementation. The ease of plotting a graph displaying the curvature of resistance over voltage will serve as a specific case study for comparison.

3.2.3 Modularity

We will evaluate modularity similarly to ease of use, incorporating public opinion and our experience in developing the web application. The ability to deconstruct code to make it modular and scale-able will be a key focus, highlighting the frameworks' capabilities in reducing duplicate code and enhancing project structure.

3.2.4 Community

Community support will be assessed through a research-based approach. We will evaluate the framework's popularity and community activity by analyzing the number of questions and answered or not accepted answers on stack overflow and ongoing community development efforts. YouTube will also be an aspect of the community due to its importance to developers.

3.3 Method for RQ2

3.3.1 Researcher

For Research Question 2 (RQ2), the primary focus will be on the visualization of scientific data from a researcher's perspective. Input from two professors, both specializing in electronics, will be highly valued. Their insights will be crucial because the developed tool will be used by them and other researchers, providing necessary parameters and inputs for creating the desired visualizations.

3.3.2 Scalability

Scalability will be a critical aspect, considering the need for future development of the web application. We will measure scalability through our development process and review relevant scientific reports. This will include evaluating how well the frameworks handle dynamic adaptation of graph axes based on incoming data points, ensuring the visualizations are accurate and responsive to changes in the data.

3.3.3 Functionality

Functionality will be compared based on how well each framework executes drawing, scaling, and presenting scientific data as specified in RQ2. React and Angular will be evaluated on their ability to dynamically adjust graph axes based on data points provided by a device developed by the company.

This methodology will provide a comprehensive framework for comparing the usability of React and Angular, focusing on critical aspects relevant to the development of the web application and its future scalability.

4

Implementation

This chapter provides a brief insight into the process of building the website using both React and Angular as the frontend frameworks.

4.1 Planning

The project commenced with a week of planning. During the initial meeting with the company, the project owners presented their ideas regarding the website's design, functionality, and the timeline they had in mind. While the project owners had a fairly clear vision of the website's functionality, no predetermined tools or frameworks were specified. Given our limited knowledge of hosting a website with a database, determining the most suitable framework and backend required extensive research.

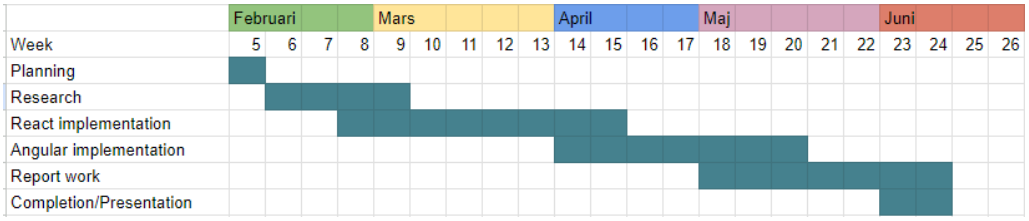

Figure 4.1: Proposed Gantt chart for the project

4.2 Research

Consequently, the following two weeks were dedicated to researching available options for website hosting and databases, as well as determining the appropriate framework for the website. During this period, we concluded that the simplest option for hosting the website was Firebase, Google's own backend cloud hosting service. Firebase allows websites to be hosted with an integrated database at no cost until significant traffic volumes are reached, making it the best candidate since the expected traffic was minimal [20]. Moreover, it has a low learning curve, enabling us to get started quickly.

After determining the backend, we decided to begin by building the website in React. The process was documented according to the different criteria defined in 1.2. Once

the website was completed, we replicated the result in Angular, documenting the process and comparing it to the React development experience.

4.3 Building the website with React

Since delivering a functioning version of the website was a high priority for the company, we immediately started learning React and building the website. The website needed to include the following features, which were implemented in the following order:

- Ensuring the website is only accessible to authenticated users.
- Allowing a user's account to access only tests from the factory to which the user is linked.
- Providing a blueprint of the factory layout with clickable zones.
- Enabling the sorting of test results depending on the selected zone.
- Rendering test data from the test device in a graph.
- Providing the ability to connect to the test device and start a test.

Setting up the website was done following the official documentation from React and Firebase [21] [22]. The process was smooth, and following the documentation was straightforward. Once the website was correctly set up and connected to the Firebase backend, we could start working on implementing the necessary features.

By following tutorials on YouTube and examples on Stack Overflow, setting up the authentication layer of the website was straightforward. The abundance of community documentation ensured that the most popular tutorials were easy to follow due to fierce competition.

Setting up the role-based access control (RBAC) for the website was more challenging. Since Firebase has no built-in mechanisms to ensure RBAC, we added an experimental extension called "Firestore Auth Claims" to the Firebase project [23]. Using the extension was easy due to the excellent documentation, even though it was in the experimental stage. The extension enabled us to keep track of which users had been assigned which roles through a document in the database. This meant that roles could not be manipulated by users, as the only way to change or add a role to an account was through the backend of the application.

The project owners wanted users to see a blueprint of their factory with each zone mapped out. The zones should be clickable and bring up a list of all completed tests in that specific zone. We implemented this feature using the component react-image-mapper [24]. This component highlights interactive zones in images, enabling users to click the blueprint to select a zone.

Sorting the test results by the selected zone was easily achieved since the database structure allowed easy access to zone-specific test results. However, this approach led to an excessive number of unnecessary reads from the database. To combat this,

we used the state management library Redux [12]. Once the app was loaded and a user authenticated, the app read all information from the database and stored it locally in the client's Redux store. This ensured that navigating between pages and re-loading tests would not result in unnecessary reads from the database.

Database	Collections and Subcollections
factory	- **map** - **tests** - zone 1 - testResults - cleaningLogs - cleaningInstructions - zone 2 - testResults - cleaningLogs - cleaningInstructions - zone 3 - testResults - cleaningLogs - cleaningInstructions

Table 4.1: Example database structure showing factories and their collections.

Rendering the test data as a graph was not as straightforward as implementing some of the other functionalities. The biggest obstacle to overcome was that we needed the graph to be plotted in the following sequence:

1. from 0 to the maximum value,
2. from the maximum value to the minimum value,
3. from the minimum value back to 0.

The problem was finding a React library that could aid in rendering graphs with a bidirectional path along the x-axis. After researching, we found only one suitable candidate, D3 [25]. D3 is a free and open-source JavaScript library used to visualize data. It offers powerful tools to render graphs but at the cost of complexity. We found using the library quite difficult, and the documentation was sometimes lacking.

Communication with the test device was through the USB ports on the computer using the serial interface. To enable the website to handle serial communication, we used Chrome's Web Serial API [26]. This API provides a way for websites to read from and write to a serial device with JavaScript. The API was chosen because it had good documentation and fit our needs well, as the test device could only output its data over the serial interface.

4.4 Replicating the website in Angular

Once the website was up and running with the desired functionality, we began the process of replicating the website using Angular as the frontend framework. Replicating the website was deemed the best and quickest way to conduct a fair comparison between the two frameworks. Since we already knew what functionality was needed and the logic to achieve it, the primary challenge was learning to use the Angular framework.

A new Firebase backend was instantiated and linked to an Angular project. The first thing that stood out in Angular compared to React was that the file structure for the root of the application, the app.jsx/ts file, was divided into three separate files: app.component.ts, app.config.ts, and app.routes.ts, instead of the single app.jsx file in React. This initially caused some confusion as it was not entirely obvious how the logic should be integrated into the different files. Following the official documentation to set up the project's root was also difficult because the documentation only provided steps on how to use Angular with an app.module.ts file instead of an app.component.ts file, which we had in our project. This was unexpected, considering that both Firebase and Angular are Google products, so we anticipated better integration guidance.

Finding solutions to this problem online was not easy either. Angular regularly receives significant updates that make impactful changes to its best practices. As a result, community documentation, such as YouTube videos and Stack Overflow threads, was often outdated and provided solutions using deprecated methods. The lack of up-to-date community documentation made learning Angular more challenging compared to React.

Once the project was correctly set up and we understood the root file structure, the next step was fixing authentication for users. This process was as seamless as it was with React, primarily because authentication is set up on the backend, meaning the frontend framework had no impact on this step.

The next step was to implement the clickable zones on a factory blueprint. Implementing this step was not possible within the project's timeframe, as there was no publicly available library or component for Angular that provided this feature.

Sorting the test results by zone was something we believed would be solved similarly to React, through a state management library. Angular features its own state management library called NgRx, which we hoped to use [27]. Using this extension was not as straightforward as using Redux. The official documentation was complex, and community resources were often lacking and outdated. We could not get this to work, so we sorted the tests by zone using a simple fetch from the database to retrieve data from a specific zone. This approach worked but would result in excessive reads from the database in a production build.

To render the received data from the test device as a graph, we encountered the same problem with Angular as with React. There were no built-in features to plot a graph with a bidirectional path, and no community resources were helpful. Therefore, we used D3 to visualize the data, as we did in the React build. Since D3 is a JavaScript library, not connected to any specific framework, the implementation was nearly identical to the React build. This allowed us to reuse existing code, which made our work easier but compromised the comparison for **RQ2** in 1.2.

When it came to setting up the connection between the website and the test device, we decided there was not enough time to implement this step thoroughly. The project timeline was nearly exhausted, and implementing the connection logic would take too long. This was not a significant problem since we had already gathered enough data to compare the frameworks.

5
Results

The following tables present evaluations of different criteria for Research Questions 1 (RQ1) and 2 (RQ2). Each criterion is scored on a scale from 1 to 10, where 1 represents the lowest score (e.g., no documentation) and 10 represents the highest (e.g., perfect documentation). The scoring system is inspired by the methodology used in [1]. It is important to note that these evaluations are made from the perspective of two university students with prior knowledge of several programming languages, including Java and JavaScript, but no prior experience in creating web applications with either React or Angular.

5.1 Criteria for RQ1

RQ1 focuses on the straightforwardness of implementation, with documentation being our top priority. If the documentation for a framework is easy to read and understand, implementation becomes significantly easier. The evaluation of documentation is restricted to official websites and their reviews on GitHub. Ease of use follows, where we assess how intuitive and complex the code is to implement. Complexity is essential for creating complex models and websites, but our experience showed that similar elements had varying complexity across different frameworks. Modularity is crucial for developing large websites, as it measures how intuitive it is to create modular components. The community is also a vital aspect of any framework, providing support and contributing to the framework's development. However, it is lower in priority compared to documentation, as good documentation can answer many questions. With poor documentation, the community becomes the primary source of answers, which can be problematic.

Criteria RQ1			
Criteria	React	Angular	Priority
Documentation	9	6	1
Ease of use	8	5	2
Community	10	6	3
Modularity	10	10	4
Backing	Facebook	Google	

Table 5.1: Criteria for RQ1

5.1.1 Documentation

Documentation is the most crucial part of any framework, provided by the framework's maintainers. When comparing the two official documentation pages, the target audience is an essential factor. Angular's documentation assumes users have prior programming experience, stating, "These docs assume that you are already familiar with HTML, CSS, JavaScript, and some of the tools from the latest standards, such as classes and modules" [28]. This perspective is geared towards more experienced programmers. In contrast, React's documentation is designed for learning from the ground up, offering pointers for common mistakes and step-by-step processes [21]. From the perspective of university students learning a new framework, React's documentation is significantly better. However, for experienced programmers, Angular's documentation might be more beneficial. For consistency in our criteria, we evaluated the documentation from the perspective of beginners, without negating the importance of advanced documentation for experienced users.

React's documentation has a more intuitive navigation and development flow. The ability to try code directly in the documentation makes learning highly effective. Users can edit code directly, which we found helpful for comparing our code with the documentation examples. Comments on common mistakes are beneficial for beginners, reducing the trial-and-error learning time. However, this can make it slower to skim through the documentation for specific problems.

Angular's documentation is more traditional, resembling a standard word document, making it less intuitive for users. It provides higher-level information, requiring prior knowledge of various programming languages and libraries. Consequently, Angular's documentation is less accessible to the general public. It is noteworthy that Angular is commonly used within companies with experienced programmers, making its documentation well-suited for them. The formal and rule-bound presentation can introduce a steep learning curve for beginners. Therefore, Angular's documentation has significant potential for improvement for users who are not well-versed in programming.

In conclusion, React's documentation is more user-friendly and accessible for new users, with a well-structured navigation system and effective learning tools, resulting in a high score of 9. Angular's documentation, while valuable for experienced programmers, is less accessible for beginners, with a more formal and less intuitive presentation, leading to a score of 6. For our purposes, React's documentation is superior due to its user-friendly nature and ease of navigation, while Angular's documentation is less approachable for newcomers.

5.1.2 Ease of Use

When handling the two frameworks, two things become apparent rather quickly. React is more intuitive in its design, and the coding process is smoother compared to Angular. This assessment is influenced heavily by the information available on the internet, including documentation and community support. The primary focus

when evaluating the ease of use is how easily one can develop and utilize the code. The complexity of the code's structure plays a significant role; excessive complexity can slow development, while insufficient complexity can render the framework inadequate for development. Therefore, ease of use is crucial for producing smooth and well-structured code.

The process of working with React was generally satisfactory, though not without some drawbacks. Overall, React performed well; the code was easy to understand and work with. The learning curve for React is relatively small, especially for those with prior knowledge of JavaScript or HTML, which contributes to its high ease-of-use score [29]. Even without prior knowledge of JavaScript or HTML, React is designed for fast learning with a high skill ceiling. The documentation for React simplifies problem-solving if any issues arise. The overall experience of learning and using React proved to be enjoyable.

The experience of using Angular, in contrast, was different from React. The ease of use for Angular was low, hindered by poor documentation, which resulted in a slow start. As previously mentioned, the complexity of the code was more of a hindrance than an asset. When replicating the same code for React due to RQ1, the same result was achieved but with varying levels of complexity. The primary issue with Angular is its tendency to overcompensate by creating a complex framework rather than a well-structured one. For example, when implementing a navigation bar for the web application, in React, you place the nav bar above other items in the file 'app.js', ensuring it is positioned above other components on the page. In Angular, you need to replicate the placement multiple times in different files, such as 'app.components' and 'app.routes'. This extensive use of multiple files and the hindrance of simplification made the code complex. This observation is supported by other papers stating, "On the other hand, the weaknesses are that Angular 2 contains too many files and it is not compatible with AngularJS" [30] and "The only exception being Angular, which has everything the same, but its components are also divided into modules that contain supporting code files related to components belonging to a certain module" [31].

The results for the ease of use criteria are as follows: React scored 8, while Angular scored 5. React demonstrated many qualities that facilitate ease of use, with its simple yet structured design making it intuitive to work with. However, in some aspects, React falters in terms of ease of use, occasionally feeling clumsy with an odd design approach. Overall, React was easy to use, with no major instances of difficulty. Angular's score of 5 is heavily influenced by its design and the over-complication of certain aspects. This assessment is from the perspective of beginners in both React and Angular. Ease of handling and working with a framework is a determining factor for its continued use in future projects. Our findings were supported by other sources indicating that significantly more people would use React again compared to Angular [32].

5.1.3 Community

RQ1 Community			
	React	Angular	Priority
YouTube videos	980,000	39,000	1
Stack overflow number of questions	478,312	304,939	2
Stack overflow number of unanswered questions	191,810	117,287	3
Percentage unanswered questions	40.1	38.5	4

Table 5.2: RQ1 Community

To quantify the community's impact, we considered subcriteria such as YouTube videos and Stack Overflow questions. These platforms are critical for developers seeking reliable information and problem-solving visualizations. YouTube allows experienced and professional programmers to teach and demonstrate correct coding practices, while Stack Overflow enables individuals to ask questions and receive answers from professional developers. It is important to note that this thesis reflects the perspective of two university students learning and developing a web application. This path is common for many in the field, making the abundance of information available from individual contributors, rather than companies, particularly valuable.

YouTube was our initial resource for both React and Angular development. There was a significant disparity in the availability of resources: React has over 25 times more videos than Angular, making it easier to find relevant and up-to-date information. The relevance of videos was made with YouTube's own relevance parameter when selecting videos. This observation is supported by the sheer number of videos available—980,000 for React compared to 39,000 for Angular. The large and active React community contributes to this disparity. A snowball effect is evident, where the extensive and current information available on YouTube attracts more learners and teachers, further growing the active community.

React also leads in the number of Stack Overflow questions, which is expected given its popularity [33]. The active community engaging in answering questions is larger for React, likely due to the greater number of people working with the framework. In contrast, Angular has 1.6% fewer questions than React [34]. While the percentages of unanswered questions for both frameworks are similar, the gap between them is smaller than anticipated. Larger frameworks tend to have more responses, and as the community size decreases, the number of answered questions typically drops. This trend holds true but on a smaller scale. When developing a product, users often seek help on the internet; a lower chance of finding solutions can significantly hinder development. Therefore, React scores higher in this criterion, with Angular falling short.

To summarize the community criterion, the availability of information is of utmost importance. If the documentation provided by the frameworks is lacking, the community must fill the gap to facilitate learning and development. In this regard, React is the clear winner, especially considering the extensive YouTube resources. While one could argue that Stack Overflow presents a closer comparison, the high number of answered questions for React suggests a greater likelihood of finding relevant answers. Consequently, the final scores for the community criterion are 10 points for React and 6 for Angular. Angular lacks in areas where React excels, from an active community to the availability of videos. The extensive community support for React provided a vast array of solutions, making it superior in this evaluation.

5.1.4 Modularity

Modularity is a crucial aspect of any framework or programming language, determining its potential for scalability and re-usability. Low modularity can render code obsolete, leading to wasted time and the potential necessity of rewriting the code in a new framework or language. In the current technological landscape, a framework is expected to offer some degree of modularity, and for a product to be successful, it must provide robust modularity. Thus, it was anticipated that both frameworks would offer modularity; the real question was how effectively they implemented this feature.

Our findings indicated that most of the modularity related to our project was influenced by the framework's structure. The key observations were connected to how each framework organizes its file system. This organization impacts the re-usability of code, requiring different methods of importing and implementing modular components. Both frameworks provided sufficient means for modularity for our project needs.

Our project's modularity requirements were relatively low, and the level of modularity implemented was adequate for a small-scale project. However, we could not personally evaluate the large-scale modularity necessary for more extensive projects. Consequently, both frameworks received a score of 10 for modularity. We encountered no issues, and the modularity provided was more than adequate for our purposes, with no need for improvement in this area.

5.2 Criteria for RQ2

The criteria for RQ2 are structured similarly to RQ1, from highest to lowest priority. When creating the web application, the visualization of scientific data was designed primarily for researchers rather than the clients of the web application. Consequently, the needs of the researchers were of the highest priority, as they provided essential insights into how to properly structure the data. Scalability in this context refers not to the scalability of the framework itself but rather to how well the frameworks implement scaling within graphing, such as adjusting the axis of the

graph for larger or smaller numbers and enabling zoom functionality. Functionality pertains to how effectively the tools provided by the framework can be implemented, including the available functions and the overall working environment.

During our research into developing the graph, we discovered that neither React nor Angular offered built-in visualization properties for graphing, as illustrated in Figure 5.1. Therefore, we utilized a React library called D3, which provided the necessary functionality for visualization and all the components required for our purpose. This leads to our research question RQ2. Given the nature of the frameworks, React and Angular did not provide direct visualization tools for our needs. Consequently, the evaluation will focus on what D3 achieved, with the results reflecting the performance of D3 rather than React and Angular.

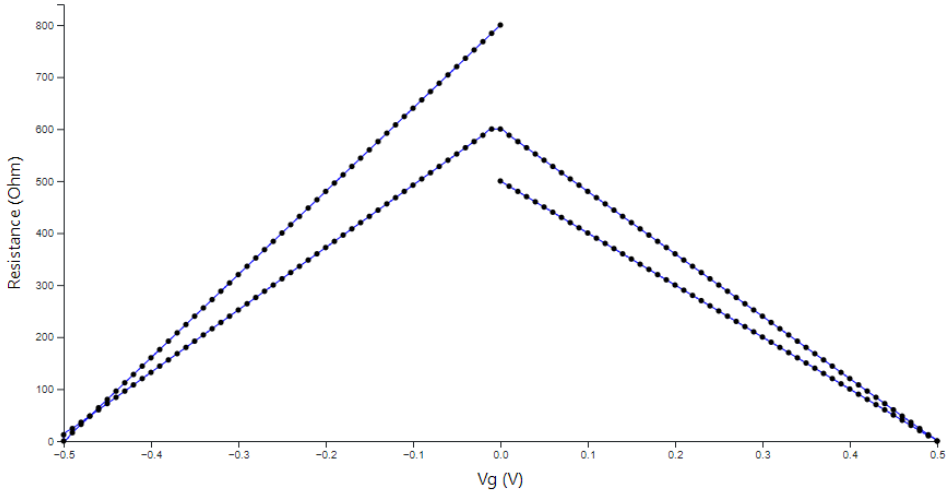

Figure 5.1: A graph visualizing the resistance measured from the biosensor.

Criteria RQ2		
Criteria	D3	Priority
Researcher	9	1
Scalability	9	2
Functionality	10	3

Table 5.3: Criteria for RQ2

5.2.1 Researcher

This criterion is influenced by several factors, the most important being the requests made by the researchers. These requests were clear and easy to understand. However, as the researchers progressed in their work on graphene, they provided additional input during our development process. This required us to rework the

product multiple times. Consequently, our development was initially conducted entirely in React and then recreated in Angular. Another factor influencing the researchers' criterion was the uncertainty regarding the final product, which only later in the development process met their satisfaction.

The final product was presented to the researchers, alongside electronics engineering students who were developing the electronics for the graphene project. The researchers were generally satisfied with the product but had some minor comments. These comments did not pertain to the graph itself but rather to the storage of the graphs. They requested that the graphs be accessible through a calendar with time-based organization, a feature we were unable to implement within our timeframe.

An important feature requested by the researchers was the high configurability of variables and inputs. Variables such as the time between points and the number of points were inputs the researchers wanted to control before the graph was generated. All these variables and inputs were successfully delivered to their satisfaction. A minor issue was the ability to zoom in on the graph for greater precision and better visualization, which we were unable to achieve. This limitation resulted in a slight reduction in points, leading to a final score of 9 for D3.

5.2.2 Scalability

The generation of the graph depended on various inputs, as discussed in the previous section (5.1.2.1). Therefore, the graph's x and y axes needed to scale for all possible inputs. This was essential to meet the requirements of RQ2, ensuring the criteria provided the most efficient tools for visualizing scientific data. In this context, we aimed to create the most effective way to observe the presented data. A graph without scaling is not flexible and thus fails to meet the aforementioned criteria.

The scaling provided by D3 was adequate, updating in real time on the web application. When new points were added, the graph automatically adjusted its x and y axes to accommodate the new values. This process, called Scalable Vector Graphics (SVG), ensured that even when zooming on a computer screen, the graph remained sharp. However, the scaling of the axes when zooming, as mentioned in section 5.1.2.1, was not achieved. Implementing this feature effectively would have been highly beneficial for D3. Consequently, the score was adjusted from a perfect 10 to a 9 due to this limitation.

5.2.3 Functionality

D3 is an outstanding library for data visualization, offering all the essential features for visualizing data. When importing a long string of variables, it efficiently determined where to cut based on a string output received from the device. It was easy to work with and well-planned for the purpose. The quality of this library was evident, and it met all our needs. The scaling, discussed in the previous section, was relatively simple to implement. A potential downside to its functionality is its

complexity; however, this complexity was of high quality, with smart solutions that facilitated an easy development process.

The tools provided by D3 were extensive, clearly delineating what the library could offer, including all the features we desired. However, the zooming feature was not implemented, which could be seen as a negative aspect of its functionality. Despite this, we decided against implementing it because it would disrupt our table when rendering new points. Therefore, while the functionality for zooming was present, it was not viable for our use. This might be considered a user error, and since the criteria are based on available features, it would be an oversight to reduce points. Consequently, the final score for functionality is a perfect 10, reflecting the library's excellent performance.

6
Discussion

This chapter presents a reflection on the degree project and discusses potential weaknesses in the methods used to compare the frameworks.

6.1 The Results

The results highlighted the strengths and weaknesses of both React and Angular, ultimately showing that React outperformed Angular in many of the criteria, as illustrated in Table 5.1. Several factors contributed to this outcome, primarily the target audience and popularity. React is widely recognized as one of the leading frameworks for front-end development, and for good reason; it performs exceptionally well, with its lowest score being an 8. This is not to undermine lesser-known frameworks, which can still be valuable for developers. However, in this comparison, Angular fell short in several areas, most notably in ease of use. React excelled here, offering simplicity and clarity that cater to new developers seeking a straightforward yet powerful framework. Angular, while offering a complex framework with many features, could not match React's better-structured approach.

The task for both frameworks was to create a website, developed by two university students. This condition influenced the results, as it became evident that Angular did not cater to this demographic. The structure of Angular's files and code was not as intuitive given our background, which is likely shared by many others in similar situations. When learning new frameworks or libraries, documentation is of utmost importance. As mentioned in the results, Angular's documentation did not meet our expectations, particularly from a learning perspective.

Both React and Angular lacked built-in graphing capabilities, leading us to use D3 for this purpose. D3 provided high-quality graphing functionalities that met our initial requirements. The only notable shortcoming was the absence of a zooming feature on the graph, but aside from that, D3 delivered an excellent graphing solution. The researchers, along with Jimmy and Vincent, were satisfied with the final graph once all elements were in place. The scalability feature of the graph was particularly well-received by the researchers. Without proper implementation of scalability, D3's evaluation would have suffered significantly.

In conclusion, while React and Angular were somewhat disappointing in certain aspects, they did succeed in providing the necessary framework for creating a web-

site. Working with both frameworks offered valuable insights into what each can and cannot deliver. When comparing the average scores from the RQ1 criteria, React achieved an average score of 9.25, while Angular scored 6.25. The larger-than-expected difference indicates that small changes and structural nuances can significantly impact the overall experience of using a framework.

6.2 Development Process

Building the website required multiple iterations and considerable flexibility, as the product owners frequently altered their requirements during the development process. This necessitated a high degree of adaptability in our approach, especially given the dual-framework strategy involving both React and Angular.

6.2.1 Challenges and Adaptations

Frequent Requirement Changes

The product owners' evolving vision meant that certain features had to be reimplemented multiple times. This highlighted the importance of an agile development approach, where incremental updates and continuous feedback are integral to the process. Both frameworks' component-based architecture facilitated this iterative process by allowing for easier modifications and testing of individual components.

Framework Selection

The decision to use both React and Angular was driven by a desire to compare their performance and suitability for the project. While React's simplicity and extensive community support proved beneficial, Angular's complexity and fragmented documentation presented significant hurdles. With better preparation or additional resources, we might have streamlined the Angular implementation more effectively.

Graphing and Data Visualization

Both frameworks lacked built-in graphing capabilities, necessitating the use of D3.js for data visualization. While D3 provided powerful tools for creating dynamic and interactive graphs, its complexity posed challenges. The decision to use D3 was ultimately justified by its flexibility and the quality of the final graphical representations, which met the project's requirements.

Role-Based Access Control (RBAC)

Implementing RBAC in Firebase was particularly challenging due to the lack of built-in support. The experimental Firestore Auth Claims extension was instrumental in overcoming this limitation. Although experimental, it provided the necessary functionality to manage user roles securely. Future projects might benefit from a more mature RBAC solution, potentially reducing the development effort and improving security.

Testing and Device Integration

Integrating the website with the test device via the Web Serial API was an ambitious but ultimately successful endeavor. This API allowed us to handle serial communication directly within the web application, demonstrating the potential for web technologies to interface with hardware devices. However, this added complexity underscored the importance of thorough testing and robust error handling to ensure reliable performance.

6.2.2 Reflections and Recommendations

Potential Framework Alternatives

Given the challenges faced with Angular, it might have been beneficial to explore other frameworks such as Vue.js. Vue offers a simpler learning curve compared to Angular while maintaining many of the benefits found in React. Its inclusion in the initial research phase could have provided a valuable additional comparison point.

Documentation and Community Support

The disparity in documentation quality and community support between React and Angular was a critical factor in the project's outcomes. Future projects should consider the availability and quality of learning resources as a key criterion when selecting frameworks.

Flexibility and Agile Methodologies

The project's success was heavily reliant on the flexibility to adapt to changing requirements. Implementing agile methodologies more rigorously, with frequent iterations and continuous feedback loops, could further enhance responsiveness to stakeholder needs.

Potential Framework Bias

It might be assumed that the results are biased towards React due to the additional time allocated for its development. However, we do not believe that our evaluation favored React because the extended time was necessary to ensure the website's functionality. This did not influence the scoring. The shorter timeline for Angular was due to the fact that we had already identified the required functionality and learned how to implement it. Consequently, replicating the website with Angular was significantly faster than the initial construction.

Original Timeline

We believe that we were able to adhere to the timeline outlined in 4.1. However, due to the project owners' demand for a rapidly functioning website, we were unable to commence the report writing as early as we would have preferred.

Final Product and Lessons Learned

Despite the challenges, the project delivered a functional website that met the core

requirements. The experience underscored the importance of choosing the right tools for the job, preparing for unexpected changes, and leveraging community resources effectively.

In conclusion, the development phase of the project provided invaluable insights into the strengths and limitations of both React and Angular. The flexibility and iterative nature of the development process were crucial in adapting to changing requirements and ultimately delivering a successful product. Future projects would benefit from a more comprehensive initial research phase, including a broader range of potential frameworks, and a stronger emphasis on agile development practices.

7
Conclusion

In conclusion, both React and Angular fulfill their promises of providing robust frameworks capable of designing web applications. Their performances in different areas varied significantly, with each framework taking different approaches to similar problems. Ultimately, React performed better in almost all the criteria, excelling in documentation, community support, and ease of use. Both frameworks demonstrated high modularity and a lack of built-in graphing capabilities. React's learning curve was considerably smaller than Angular's, which has a high level of complexity that posed challenges during development.

When the code was properly implemented, both frameworks produced equally structured and well-planned web applications. The data visualization was achieved using D3, which provided excellent development and functionality. The only minor issue encountered was the inability to create zooming functionality on the graph.

In summary, React offered the most straightforward implementation process and presented the least challenging learning curve compared to Angular. While both frameworks lacked intuitive tools for visualizing scientific data, therefor the usage of an external library (D3), effectively compensated for this limitation.

Bibliography

[1] S. Abrahamsson. *A model to evaluate front-end frameworks for single page applications written in JavaScript*. M.Sc. thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 2023. [Online]. Accessed: 4 March 2024. URL: https://www.diva-portal.org/smash/get/diva2:1758858/FULLTEXT01.pdf.

[2] A. Gerdessen. *Framework comparison method*. M.Sc. thesis, University of Amsterdam, Amsterdam, Netherlands, 2007. [Online]. Accessed: 21 April 2024. URL: https://homepages.cwi.nl/~jurgenv/theses/AntonGerdessen.pdf.

[3] *What is a Web Application?* Aws.amazon.com. [Online]. Accessed: 17 June 2024. URL: https://aws.amazon.com/what-is/web-application/.

[4] *HTML basics*. Developer.mozilla.org. [Online]. Accessed: 17 June 2024. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics.

[5] *What is CSS?* Developer.mozilla.org. [Online]. Accessed: 17 June 2024. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS.

[6] *What is JavaScript?* Developer.mozilla.org. [Online]. Accessed: 17 June 2024. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[7] *TypeScript is JavaScript with syntax for types.* Typescriptlang.org. [Online]. Accessed: 17 June 2024. URL: https://www.typescriptlang.org/.

[8] *Introducing JSX.* Legacy.reactjs.org. [Online]. Accessed: 17 June 2024. URL: https://legacy.reactjs.org/docs/introducing-jsx.html.

[9] *Introduction to the DOM.* Developer.mozilla.org. [Online]. Accessed: 17 June 2024. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.

[10] *Virtual DOM and Internals.* Legacy.reactjs.org. [Online]. Accessed: 17 June 2024. URL: https://legacy.reactjs.org/docs/faq-internals.html.

[11] *The library for web and native user interfaces.* React.dev. [Online]. Accessed: 17 June 2024. URL: https://react.dev/.

[12] *A JS library for predictable and maintainable global state management*. Redux.js.org. [Online]. Accessed: 17 February 2024. URL: https://redux.js.org/.

[13] *What is Angular?* Angular.dev. [Online]. Accessed: 17 June 2024. URL: https://angular.dev/overview.

[14] *What is GitHub? A Beginner's Introduction to GitHub.* Kinsta.com. [Online]. Accessed: 17 June 2024. URL: https://kinsta.com/knowledgebase/what-is-github/.

[15] *Code Editing. Redefined.* Code.visualstudio.com. [Online]. Accessed: 17 June 2024. URL: https://code.visualstudio.com/.

[16] *Front-end Development: The Complete Guide.* Cloudinary.com. [Online]. Accessed: 17 June 2024. URL: https://cloudinary.com/guides/front-end-development/front-end-development-the-complete-guide.

[17] *Backend: What is it and what is it used for?* Gluo.mx. [Online]. Accessed: 17 June 2024. URL: https://www.gluo.mx/en-US/blog/backend-que-es-y-para-que-sirve.

[18] *Make your app the best it can be with Firebase and generative AI.* Firebase.google.com. [Online]. Accessed: 17 June 2024. URL: https://firebase.google.com/.

[19] L. S. Vailshery. *Most used web frameworks among developers worldwide, as of 2023.* Statista.com. [Online]. Accessed: 24 May 2024. URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/.

[20] *Firebase Pricing*. Firebase.google.com. [Online]. Accessed: 24 May 2024. URL: https://firebase.google.com/pricing.

[21] *React documenation*. React.dev. [Online]. Accessed: 16 February 2024. URL: https://react.dev/learn.

[22] *Add Firebase to your JavaScript project*. Firebase.google.com. [Online]. Accessed: 16 February 2024. URL: Firebase.google.com/docs/web/setup.

[23] *Set Auth claims with Firestore*. Firebase.google.com. [Online]. Accessed: 16 February 2024. URL: https://github.com/FirebaseExtended/experimental-extensions/blob/next/firestore-auth-claims/README.md.

[24] *react-image-mapper*. Github.com. [Online]. Accessed: 18 February 2024. URL: https://github.com/coldiary/react-image-mapper.

[25] *The JavaScript library for bespoke data visualization*. D3js.org. [Online]. Accessed: 22 February 2024. URL: https://d3js.org.

[26] *Read from and write to a serial port*. Developer.chrome.com. [Online]. Accessed: 20 February 2024. URL: https://developer.chrome.com/docs/capabilities/serial.

[27] *Reactive State for Angular*. Ngrx.io. [Online]. Accessed: 24 February 2024. URL: https://ngrx.io/.

[28] *Angular documenation*. Angular.io. [Online]. Accessed: 5 March 2024. URL: https://angular.io/guide/cli-builder.

[29] S. Rathinam. *Analysis and Comparison of Di erent Frontend Frameworks*. B.Sc. thesis, Manipal Institute of Technology, Manipal, India, 2022. [Online]. Accessed: 20 April 2024. URL: https://www.suryaanshrathinam.com/static/media/SuryaanshRathinam_ComparisonOfFrontendFrameworks.a6d0c12b1b711c4cc8aa.pdf.

[30] F. Brännlund Stål E. Thorén. *Usage of Angular from developer's perspective*. Degree project report, Department of Computer Science and Engineering, Blekinge Institute of Technology, Blekinge, Sweden, 2017. [Online]. Accessed:

22 May 2024. URL: https://www.diva-portal.org/smash/get/diva2:1112464/FULLTEXT01.pdf.

[31] E. Saks. *JavaScript frameworks: Angular vs React vs Vue*. B.Sc. thesis. Business Information Technology, Haaga-Helia University of Applied Science, Helsinki, Finland, 2019. [Online]. Accessed: 24 May 2024. URL: https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf.

[32] N. Olsson N. Ockelberg. *Performance, Modularity and Usability, a Comparison of JavaScript Frameworks*. Degree project report, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden, 2020. [Online]. Accessed: 13 April 2024. URL: https://www.diva-portal.org/smash/get/diva2:1424374/FULLTEXT01.pdf.

[33] *Questions tagged [reactjs]*. stackoverflow.com. [Online]. Accessed: 4 March 2024. URL: https://stackoverflow.com/questions/tagged/reactjs.

[34] *Questions tagged [angular]*. stackoverflow.com. [Online]. Accessed: 4 March 2024. URL: https://stackoverflow.com/questions/tagged/angular?tab=Newest.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden
www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY