## Research report JavaScript framework

**­­­­­­­­­­­**



Student name: Rowan van der Weel

Student number: 4924576

Class: DB-04

Teacher: Tim Kurvers

Version: 1.0

Date: 21-09-2023

Table of contents

[**Introduction** 3](#_Toc146204405)

[Problem description 3](#_Toc146204406)

[Main question 3](#_Toc146204407)

[Sub questions 3](#_Toc146204408)

[**Results** 3](#_Toc146204409)

[Sub question 1 3](#_Toc146204410)

[Sub question 2 3](#_Toc146204411)

[Sub question 3 3](#_Toc146204412)

[Sub question 4 3](#_Toc146204413)

[**Resolution** 3](#_Toc146204414)

[Conclusion 3](#_Toc146204415)

[Recommendation 3](#_Toc146204416)

[**References** 3](#_Toc146204417)

[**Version History** 3](#_Toc146204418)

# **Introduction**

## Problem description

A JavaScript framework is a pre-built collection of tools and libraries that provides a structured way to develop web applications. In today’s landscape, it’s essential to use JavaScript frameworks to build the front-end of a web application. They offer pre-written code and patterns to handle common tasks, making it faster and easier for developers to create robust and efficient web applications.   
  
There are many JavaScript frameworks that exist, and they keep evolving, each having their own strengths and weaknesses and none of them is the **best**. The choice of a framework is very important, as it directly impacts the efficiency, performance, maintainability and scalability of web applications is impacted.

For my project “Gamify” it was recommended to use either: Angular, React or Vue.js for the front-end. I don’t have any insight about JavaScript frameworks, so with this research report I hope to gain a good understanding of these frameworks and determine which is most suitable for my project.

## Main question

Which of the following JavaScript frameworks is the most suitable for GamifyWork: Angular, React or Vue.js?

## Sub questions

1. What are the strengths and weaknesses for the chosen JavaScript frameworks?
   1. **Literature study:** I’ll search for academic papers, articles, and blog posts that discuss the chosen JavaScript frameworks. Summarize the key findings regarding their strengths and weaknesses.
   2. **SWOT analysis:** Conduct a SWOT analysis for each framework. Identify the Strengths, Weaknesses, Opportunities, and Threats associated with using each framework. This will provide a structured framework for evaluating their respective advantages and disadvantages.
2. What is the level of community support and availability of resources for learning and troubleshooting for Angular, React, and Vue.js?
   1. **Community research:** Engage with online communities, forums, and social media groups dedicated to each of these frameworks. Observe discussions, queries, and the level of activity within these communities. Take note of the number of members, frequency of posts, and responsiveness to inquiries.
   2. **Survey:** Create a survey targeting developers who have experience with Angular, React, and Vue.js. Ask questions about their perception of community support, availability of resources, and their experiences with troubleshooting. Include questions about their participation in online communities, especially on platforms like Reddit.
3. What security features and vulnerabilities exist in each framework?
   1. **Document Analysis:** I’ll review the official documentation for each framework. Look for sections or documents that explicitly outline the strengths and weaknesses of the framework.
   2. **Problem Analysis:** Research what the most popular vulnerabilities and attacks are towards frameworks. And how different frameworks find solutions for these problems.
4. What is important for GamifyWork?
   1. **Best good and bad practices:** Research and compile a list of best practices for customization in each of these frameworks. These practices should highlight effective ways to extend or modify the behaviour of the framework while avoiding common pitfalls.
   2. **Observation:** Observe and analyse real-world applications or projects built using each of these frameworks. Pay attention to how developers have customized the framework to meet specific requirements.

# **Results**

## Sub question 1

**What are the strengths and weaknesses for the chosen JavaScript frameworks?**  
All 3 of the frameworks have their own advantages and drawbacks. If you want to choose a framework, it’s recommended to pick one you like. I’ll zoom in at every one of them and then I’ll give a conclusion.   
  
**Angular**

This framework is the most widely framework among the three, it includes support for a lot of things. It is built for mainly large and complex projects. “In its core, Angular is all about building re-usable user interface components which you then control with Angular and which you can combine with other components to build an entire user interface from those Angular-controlled components” (Schwarzmüller, 2023). Angular uses Typescript. Angular is very popular amongst developers.

**React**

React is a JavaScript library that is all about components. It’s a very small and simple library, which provides you tools to render whatever you want in your circumstances. React usually uses JavaScript (a special JavaScript feature called “JSX”). React has been downloaded the most in the last past 2 years, according to the graph above the conclusion section. React has the most fans and the job offering is extremely wide (best as a safe pick).

**Vue.js**

The last framework sits between React and Angular if you look at its size. Vue uses regular JavaScript. The framework is a very popular and the community keeps growing, but there are not as many job offerings as with React and Angular.

**Conclusion**

The more tools/support a framework contains, doesn’t immediately mean it’s better than the others. It simply relies on your project, if it’s a very complex project I would recommend using Angular. However, if you want to focus on simplicity and components, React would be a better option for that. Typescript includes all the features of JavaScript and more. “TS is a typed superset of JS that compiles to plain JS. In Contrast, JavaScript is a dynamic language that doesn't support types.” If you are new to JavaScript, it would be better to learn that first and later you can evolve to Type (Olszewska, 2023). “Searching on indeed on February 21, 2022, showed the following results: React has 67,301 job offerings, Angular has 24,508 job offerings, and Vue.js has 3,857” (Vue vs React vs Angular: What Framework Would You Choose?, 2022). If you want to start simple and have a wide job offering, you should start with React.

GamifyWork is simple and small, so Angular is out of the question. For GamifyWork I would choose either React or Vue, due to their suitability for smaller and less complex projects.

A screenshot of a computer

Description automatically generated

Figure 1 Downloads per year (Vue vs React vs Angular: What Framework Would You Choose?, 2022)

A blue and white table with text

Description automatically generated

Table 2 Vue vs React vs Angular – a comparative table (Vue vs React vs Angular: What Framework Would You Choose?, 2022)

|  |  |  |
| --- | --- | --- |
| **React** | **Positive** | **Negative** |
| **Intern** | * Large community. * Strength in components. | * State management complexity. * Lack of built-in features. |
| **Extern** | * Big demand for React developers. * Integration with other tools. | * Competition with other frameworks. * Fragmentation in the ecosystem. |

|  |  |  |
| --- | --- | --- |
| **Angular**  Angular JS full logo transparent PNG - StickPNG | **Positive** | **Negative** |
| **Intern** | * Full featured. * Typescript integration. | * Steep learning curve. * Larger file sizes. |
| **Extern** | * Support for big and complex applications. * Integration with tools. | * Competition with other frameworks. * Demotivate developers to work with Angular |

|  |  |  |
| --- | --- | --- |
| **Vue.js**  A green and blue letter v  Description automatically generated | **Positive** | **Negative** |
| **Intern** | * Simple for beginners. * Flexible and progressive. | * Smaller ecosystem. * Less adoption in large and complex applications. |
| **Extern** | * Growing community. * Market adoption. | * Competition with other frameworks. * Depends on one person. |

## Sub question 2

**Social media**

After carefully looking into social media platforms like Twitter, Facebook, Instagram, and LinkedIn, it's clear that React gets more attention compared to other frameworks. I made detailed graphs in Excel to show this. React is the favourite among users, showing strong activity levels. Its user-friendly and interactive features make it a top choice in the development world.

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

Figure 2

Afbeelding met tekst, schermopname, diagram, Perceel

Automatisch gegenereerde beschrijving

Figure 3

Afbeelding met tekst, schermopname, diagram, ontwerp

Automatisch gegenereerde beschrijving

Figure 4

Afbeelding met tekst, schermopname, diagram, nummer

Automatisch gegenereerde beschrijving

Figure 5

**Forum**

I delved into Stack Overflow for insights on popular frameworks. I found a whopping 467,894 questions tagged with [reactjs], while [angular] boasted 300,224 questions, and [vue.js] had 105,789. Switching gears to the GitHub community, the last week saw a surge of activity with 16 discussions for Vue, while Angular and React had only 2 each. This data underscores Reacts widespread usage and popularity.

**Survey**

I asked experienced programmers on Reddit about their usage of Angular, React, or Vue.js on different Reddit communities. Surprisingly, over 60% of them had the most experience with React. This shows that React is highly favoured among experienced developers.

Figure 6 Click [here](https://www.reddit.com/r/webdev/comments/16ug9yd/with_which_javascript_framework_have_you_worked/) for poll

Afbeelding met schermopname, tekst, software, Multimediasoftware

Automatisch gegenereerde beschrijving

­­­

Figure 7, Click [here](https://www.reddit.com/r/learnjavascript/comments/16ug7hz/with_which_javascript_framework_have_you_worked/) for the poll

Afbeelding met schermopname, tekst, software, Multimediasoftware

Automatisch gegenereerde beschrijving

## Sub question 3

**Attacks**

I’ve looked at how these different frameworks handles XSS and injection attacks. “[Cross-site scripting (XSS)](https://en.wikipedia.org/wiki/Cross-site_scripting) enables attackers to inject malicious code into web pages. Such code can then, for example, steal user and login data, or perform actions that impersonate the user. This is one of the most common attacks on the web” (Angular, 2023). “JavaScript code injection attacks are common on applications that accept user input and execute it on the server side. Orchestrating such attacks require attackers to use the developer tools console or an input field to supply the malicious script, which is parsed to the server for dynamic code execution” (Sengupta, 2022). In the following paragraph I explain what security feature(s) every framework has and how they handle the attacks.

**Angular**

This JavaScript has a built-in feature against XSS attacks. Angular incorporates security features like Content Security Policy (CSP) to block XSS attacks. CSP governs which content sources can be loaded into a webpage, bolstering security. Angular also employs a sandbox, which acts as a barrier between untrusted code and the rest of the application. This prevents malicious code from tampering with sensitive data or causing harm. While Angular is generally secure, it's crucial to use it properly to avoid potential security vulnerabilities in applications. (Srivastava, 2023)

The best practices to keep in mind (according to Srivastava) are:

* Use InnerHtml Property Binding
* Use XSS Sanitization

The first example works like the following:

import { Component, OnInit } from '@angular/core';

import { MyService } from './data.service';

import { DomSanitizer, SafeHtml } from '@angular/platform-browser';

@Component({

selector: 'app-root',

template: `<div [innerHtml] = "safeValue"></div>`,

providers: [MyService]

})

export class AppComponent implements OnInit {

safeValue: SafeHtml;

constructor(private secure: MyService) {

this.safeValue = this.secure.getSafeHtml("<h1>Sanitization Success</h1>");

}

ngOnInit() {

}

}

The second example is a bit easier to understand:

<h1>{ someUserInputValue }</h1>

Angular has features that help protect against two common internet security problems: cross-site request forgery (CSRF) and cross-site script inclusion (XSSI). While these issues are mostly dealt with on the server side, Angular has tools that make it simpler to handle them on the client side. Angular has built-in protection against harmful code in web applications through features like CSP, which helps prevent XSS attacks by controlling what scripts can run on a page. CSP in Angular allows developers to specify which external resources can be loaded, reducing the risk of unauthorized script execution and enhancing the overall security of the application (Angular, 2023). Angular provides clear and easily understandable documentation on how to implement its security features.

**React**

According to Reacts official documentation, it doesn’t come with built-in protections against XSS attacks. Additionally, React does not offer built-in support for CSRF protection, developers must implement appropriate measures themselves. While CSP is not specific to React, it can be enforced at the server level to mitigate certain types of attacks.

However, according to some forms/blogs/articles React does have some features to prevent XSS attacks. “I would say React is “pretty safe” from XSS attacks. There are many safety measures such as auto-escaping variables included in React.js to overcome these attacks” (Vindula, 2023). Here is the example he uses:

<h1>{ someUserInputValue }</h1>

One may think that an attacker could input a malicious code to the DOM and perform XSS as follows.

const someUserInputValue = '<script>alert(“This is an XSS attack”)</script>';  
return <h1>{ someUserInputValue }</h1>

Typically, this attack would work in traditional HTML, JS based application. But React is clever! Instead of executing the malicious code, it would interpret the code as a string and it would look like this.

<script>alert(“This is an XSS attack”)</script>

There are numerous online resources, articles, and advice available that offer guidance on where to concentrate efforts and how to proactively address security concerns in React applications.

**Vue.js**

In Vue, some vulnerability reports regarding XSS arise from situations where developers intentionally render unsensitized user-provided content as Vue templates, which is inherently risky and not something Vue can prevent. Additionally, mounting Vue on a page containing server-rendered and user-provided content can lead to similar vulnerabilities. It's advised as a best practice to avoid mounting Vue on nodes with such content (Vue.js, 2023) Vue employs a security measure called "escaping" to counteract XSS attacks. This ensures that any input provided is treated as plain text and not executed as code (C, 2013).

To prevent XSS attacks you can do this, Mathias explains:  
I recommend you use libraries such as sanitize-html, vue-sanitize, or vue-3-sanitize, to sanitize the HTML content. Translating it to code, go to the “XSSStoredView.vue “file where you will see this content.

<template>  
 <div class="posts">  
 <div v-for="post in posts" :key="post.name">  
 <div v-html="post.content"></div>  
 </div>  
 </div>  
</template>

As you can see, the content of the posts enters without a previous preprocessing or sanitization to the v-html directive. To fix this problem you just need to use the functions injected to Vue through some of the mentioned libraries (vue-3-sanitize library in this case).

<div v-html="$sanitize(post.content)"></div>

These sanitization libraries have a system of rules such as allow lists and deny lists. You can configure them to allow or ignore some HTML tags, properties, CSS, JavaScript, and others.

“HTTP security vulnerabilities, such as CSRF and XSSI, are primarily addressed on the backend, so they aren't a concern of Vue's. However, it's still a good idea to communicate with your backend team to learn how to best interact with their API, e.g., by submitting CSRF tokens with form submissions” (Vue.js, 2023). The official documentation site warns you about potential danger [here](https://vuejs.org/guide/best-practices/security.html#potential-dangers).

## Sub question 4

**GamifyWork**

GamifyWork is an application that I make for my individual project. One of the most important challenges with video games is that they create a sense of progress within the game, yet this progress doesn’t translate into real-life accomplishments. This application will help that by offering an enjoyable experience that also contributes to progress in real-life.

The purpose of this application is to give users a platform to view and handle tasks in a way that suits their likes and interests. Through the game experience, it aims to make task management enjoyable, motivating users to achieve their goals efficiently. This app brings together entertainment and productivity, proving that one can have fun while still making significant progress towards real-life.

The important stuff for this application will be:

* User Experience (UX) Design: Ensuring that the app is intuitive and easy to use will be crucial. Users should be able to navigate through tasks and game elements seamlessly.
* Gamification Mechanics: This is important for this project. The game elements should be engaging and motivating, providing a sense of accomplishment as users complete tasks.
* Task Management Functionality: The core functionality and most important thing of GamifyWork should allow users to add, edit, and mark tasks as complete. It should also provide features like setting due dates and priority levels.
* Time Management: Since I don’t have a lot of time, managing time effectively will be crucial. I can’t spend too much time for learning new things and it doesn't have to be perfect. I try to work from most important to least important.

The less important stuff will be:

* Advanced Security Measures: Since security isn't a major concern for my project, I'll focus on implementing some small and basic security practices.
* Extensive User Authentication: Since I'm working on an individual project, I don't need complex user authentication systems like multi-factor authentication. I’ll try to work with key cloak, but ill focus on that later.
* JavaScript Expertise: While it's great that I'm willing to learn JavaScript for this project, I understand that I don't need to become an expert right away. That’s why I want to pick one with a low learning curve.
* Large Scale Application: Since my app won't be very big and user handling isn't a major concern, I'll start with a basic scale application.

**Applications**

There are some RPG that focus on self-improvement, but the most of them are made for mobile. The application that looks the most like mine application is [Habitica](https://habitica.com/). That application uses Vue.js. Luckily, it’s an open-source application and I could read through the code, and I understand how it works. However, sites like Trello and Jira are using React. And the main important thing for GamifyWork is task Management. There are minimal projects that exist of Angular. And projects that exist of Angular (like Microsoft Planner) aren’t exactly what I need for GamifyWork.

# **Resolution**

## Conclusion

## Recommendation

# **References**

# **Version History**

|  |  |
| --- | --- |
| **When?** | **What?** |
| 21/09/2023 | First start, initialized it and finished the introduction. |
| 28/09/2023 | Chosen different methods from the DOT framework for the sub questions. |
| 05/10/2023 | Result sub question 1 finished. |
| 06/10/2023 | Result sub question 2 finished. |
| 12/10/2023 | Sub question 3 finished. |