This document serves as a means to justify the technology choices in Advanced Software semester through an evaluation matrix

Contents

[Contenders 2](#_Toc159925591)

[Technology A 2](#_Toc159925592)

[Technology B 2](#_Toc159925593)

[Technology C 2](#_Toc159925594)

[Why not anything else? 2](#_Toc159925595)

[No-Frameworks approach 2](#_Toc159925596)

[Code generators/Aggregators 2](#_Toc159925597)

[High-level full-stack frameworks 2](#_Toc159925598)

[Low and no-code generators 2](#_Toc159925599)

[Evaluation matrix 3](#_Toc159925600)

[Explanations 3](#_Toc159925601)

[Range of applicability 3](#_Toc159925602)

[Development speed 4](#_Toc159925603)

[Flexibility & Manageability 4](#_Toc159925604)

[Community & Documentation 4](#_Toc159925605)

[Load support 5](#_Toc159925606)

[Familiarity 7](#_Toc159925607)

[Conclusion 8](#_Toc159925608)

[References: 8](#_Toc159925609)

# Contenders

Keep in mind that all of the listed technologies and frameworks listed below are **narrow-focused frameworks**. This is because they are the most suitable for building enterprise level software fast, as well as having enough flexibility to satisfy most needs within the business context.

Technology A: JavaScript/ExpressJS

Technology B: C#/ASP.NET

Technology C: Java/Spring

# Why not anything else?

No-Frameworks approach:

Choosing to work with no frameworks means starting from the ground up and re-inventing the wheel. For an enterprise-grade web application narrow-focused framework have more than enough flexibility while providing a vastly faster development speed. No-frameworks are better suited for very specific embedded systems where computational speed is key.

## Code generators/Aggregators

Code generators and aggregators are meant to speed the bootstrapping of the project while providing no benefit afterwards. In this case, each technology besides JavaScript has a code generator to get a head start.

## High-level full-stack frameworks

These frameworks feature a short learning curve and fast development time. However, they suffer from flexibility and customization issues. This means that while this style of framework can satisfy the initial needs of the project, the requirements can quickly outgrow these frameworks’ flexibility, which leads to needing to rebuild the entire project.

## Low and no-code generators

This technology choice is even worse than high-level full-stack frameworks. In this case, there is little to no flexibility and customization beyond pre-defined points is impossible. These generators are the fastest way to configure simple applications, they are very unfit for an enterprise-grade solution.

## Evaluation matrix

The evaluation matrix judges a technology choice based on the weight of the criteria.

* If a contender comes first, all points are awarded
* Second place scores half of the points
* Third place scores no points

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Weight | Javascript/Express | Java/Spring | C#/ASP.NET |
| **Range of Applicability** | 3 | 1.5(2nd) | 3(1st) | 3(1st) |
| **Development Speed** | 3 | 3(tie) | 3(tie) | 3(tie) |
| **Flexibility & Manageability** | 3 | 3(1st) | 3(1st) | 3(1st) |
| **Community & Documentation** | 5 | 5(1st) | 2.5(2nd) | 0(3rd) |
| **Load support** | 10 | 5(2nd) | 5(2nd) | 10(1st) |
| **Familiarity** | 10 | 10(1st) | 5(2nd) | 0(3rd) |
| **Total score:** | **33** | **27.5/33 (1st)** | **21.5/33 (2nd)** | **19/33 (3rd)** |

# Explanations

## Range of applicability

This category scores a small 3 points due to the fact that all of the contenders are narrow-focused frameworks mostly used for RESTful services and web development.

In this category, Javascript/Express takes a 2nd place to Spring and ASP.NET due to the fact that it is mainly used for HTTP/RESTful services.

Spring and ASP.NET include features like

* Dependency injection
* Aspect-Oriented programming
* Transaction management
* Batch processing

That allow them to gain an edge over Javascript and Express.

However, due to the requirements of this semester’s project being a large-scale social-media oriented web application, all of the contenders are a suitable choice.

## Development speed

Here all the technologies score the same since they are narrow-focused frameworks. Technically speaking, Spring and ASP.NET can take advantage of their respective code generators to bootstrap a project quickly.

However, since creating a server in express only takes a few files and around 10 minutes (as described in MDN: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction>) all of the contenders score the same.

## Flexibility & Manageability

Again, here Spring and ASP.NET have advantage due to them being more flexible that Javascript since they have a slightly larger range of applicability.

However, Javascript/Express is far easier to manage due to the whole NodeJS system it resides in. From personal experience (and community opinions) it is highly agreed that NodeJS has the best package management because it is simple to work with and highly supported.

Java’s Maven/Gradle repositories are highly convoluted and not always easy to find/install/manage. The same thing can be said about C#’s NuGet packages as seen in this comparison:

A screenshot of a phone

Description automatically generated

## Community & Documentation

Here Javascript/Express undeniably takes first place over the other choices. The node ecosystem, as seen in the previous criteria, is highly present online with a large community to back it up. Documentation is also a lot more palatable and easier to understand. Below can be seen examples of the ***array.length()*** function for each of the technologies.

#Javascript

const clothing = ['shoes', 'shirts', 'socks', 'sweaters'];

console.log(clothing.length);

// Expected output: 4

#Java

// Java program to illustrate

// how to get the length of the array

public class Test {

public static void main(String[] args)

{

// Here str is the array name

// of String type.

String[] str = { "GEEKS", "FOR", "GEEKS" };

System.out.println("The size of "

+ "the array is " + str.length);

}

}

public static void Main()

{

// Declare a single-dimensional string array

String[] array1d = { "zero", "one", "two", "three" };

ShowArrayInfo(array1d);

// Declare a two-dimensional string array

String[,] array2d = { { "zero", "0" }, { "one", "1" },

{ "two", "2" }, { "three", "3"},

{ "four", "4" }, { "five", "5" } };

ShowArrayInfo(array2d);

// Declare a three-dimensional integer array

int[, ,] array3d = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },

{ { 7, 8, 9 }, { 10, 11, 12 } } };

ShowArrayInfo(array3d);

}

It is clear that C# loses here due to Microsoft’s unclear documentation and convoluted examples.

## Load support

In this category, C#/ASP.NET is the clear winner. All of the comparisons found online point towards C# being faster than Java and having the faster computing power in every scenario.

A graph with numbers and a bar

Description automatically generated

However, comparing Javascript/Express with Java/Spring gets a bit more complicated. As it can be seen in a simple hello world application:

A graph with green rectangles and numbers

Description automatically generated

When it comes to simple web traffic and taking in a lot of requests Express tends slower than spring due to its single-core nature. Java can take advantage of multi-threading capabilities and JVM optimizations, which gives it an edge in handling heavy loads and calculations.

However, if a Node cluster is utilized, express becomes vastly faster than Spring. A node cluster creates multiple workers based on the amount of CPU cores available.

A graph of a number of green and yellow bars

Description automatically generated with medium confidence

Due to the requirements of this semester’s project, both Javascript and Java are a valid choice and are capable of building scalable enterprise applications. Javascript’s single-threaded nature can be remedied by either using a Node cluster or Kubernetes. Since Kubernetes is a learning outcome of this semester, this issue becomes invalid.

However, Java would be more useful if there are complex computations to take advantage of multithreading, which is not the case in this project.

## Familiarity

This category is purely based on personal experience with the technologies. To put it simply, my familiarity with the technologies is as follows:

* Javascript/Express: Used for passion projects and Internship, experience of 2y+
* Java/Spring: Used in semester 3, experience of 6 months
* C# Used in semester 2, experience in building RESTful services: none

Having this in mind, my research of this semester has shown that there is no time waste and development time needs to be as fast as possible, since there is a lot of material to cover. If one’s desire is to explore new possibilities rather than to use something that they know would work, it is highly possible to be short of time.

This is why Javascript/Express takes first place in this category.

# Conclusion

In conclusion all of these technologies have the potential to support a large number of users, requests, and data volume.

Javascript/Express is the winning technology choice that is used for this semester’s project. This choice is based on factors such as the development expertise, compatibility requirements, and specific performance goals. Javascript is mainly used for RESTful services and web development, which makes is slightly less robust that C# and Java. Both technologies feature multithreading, which gives them an edge in handling large amounts of data and calculations.

However, due to Javascript’s large community, simple package management, easy-to-understand documentation, asynchronous nature, as well as my personal familiarity, it gains an advantage over its competitors.

# References:

Prema, P. (2022, January 5). Spring Boot vs ASP.NET Core: A Showdown - Putu Prema - Medium. *Medium*. <https://medium.com/@putuprema/spring-boot-vs-asp-net-core-a-showdown-1d38b89c6c2d>

Choubey, M. (2023, November 8). Node.js Express Cluster vs Spring Boot: Hello World performance | Tech Tonic. *Medium*. https://medium.com/deno-the-complete-reference/node-js-express-cluster-vs-spring-boot-hello-world-performance-8ccc78dcafcd

Stukalov, A. (2018, July 2). *Classification of development frameworks for enterprise applications*. Java Code Geeks. <https://www.javacodegeeks.com/2018/06/classification-development-framework.html>

*Express/Node introduction - Learn web development | MDN*. (2024, January 1). MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction>

*Gradle vs npm vs NuGet | What are the differences?* (n.d.). StackShare. <https://stackshare.io/stackups/npm-vs-nuget-vs-gradle>

*Array: length - JavaScript | MDN*. (2023, September 12). MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/length>

GfG. (2023, July 17). *How to determine length or size of an Array in Java?* GeeksforGeeks. <https://www.geeksforgeeks.org/how-to-determine-length-or-size-of-an-array-in-java/>

Dotnet-Bot. (n.d.). *Array.Length Property (System)*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/api/system.array.length?view=net-8.0