Framework / Project

Justification

Oliver Badger

# Introduction

This document is to provide insight on the justificationfor my project developing an automatic route plotter for my project with the accelerate programme for Robert Walters.

# Project Template

## ASP.NET Core Web Application

ASP.NET Core is highly useful for building a route generator application due to its scalability, cross-platform capabilities, and excellent support for third-party API integration, such as the Google Maps API. Designed to handle high-performance, scalable web applications, ASP.NET Core can efficiently process multiple requests simultaneously, making it ideal for this project. Its cross-platform nature allows it to run on Windows, Linux, and macOS, offering flexibility in deployment environments. The framework supports the Model-View-Controller (MVC) architecture, allowing a clear separation of concerns, which is beneficial for maintaining and scaling the application. Additionally, Razor Pages simplify the development of page-focused scenarios, making it easier to create interactive and dynamic web pages. ASP.NET Core also provides comprehensive security features, including authentication and authorisation, essential for protecting user data. In relation to the project, the MVC pattern will effectively manage user interactions, route data processing, and presentation, while Razor Pages will enhance the user experience by providing interactive and responsive design elements for route suggestions and visualizations.

## ASP.NET Web Application (.NET Framework)

ASP.NET Web Application (.NET Framework) is highly useful due to its mature ecosystem and compatibility with existing libraries and legacy systems, making it ideal for complex web applications. The .NET Framework's long history and robust set of libraries and tools can be leveraged to build comprehensive solutions, particularly when integrating with other enterprise systems. This template features Web Forms, which allow for rapid development with drag-and-drop components, facilitating the quick construction of user interfaces. Additionally, it supports the MVC pattern, similar to ASP.NET Core, providing a structured development approach. The comprehensive tooling offered by Visual Studio ensures a rich development environment with powerful debugging and deployment capabilities. In relation to the project, this template would be advantageous if there is a need to integrate with existing .NET Framework applications or libraries. The MVC pattern can also provide a well-structured approach to developing the route generator application.

## Blazor App

Blazor App is highly useful for its ability to allow developers to use C# for client-side logic instead of JavaScript, promoting code consistency and reuse. Its component-based architecture facilitates the development of interactive and reusable UI components, significantly enhancing the user experience. The framework offers two distinct features: Blazor Server and Blazor WebAssembly. Blazor Server runs the app on the server with real-time updates to the client, making it ideal for scenarios requiring high security and consistent data synchronization. Blazor WebAssembly runs the app directly in the browser, providing a rich, interactive user experience without server round-trips. In relation to the project, the ability to use C# across both server and client-side code can streamline development and maintenance. Additionally, Blazor's component-based approach can efficiently handle the dynamic nature of route generation and map interactions.

## Node.js with Express

Node.js is exceptionally well-suited for handling multiple concurrent requests thanks to its event-driven, non-blocking I/O model, making it ideal for tasks such as fetching routes from the Google Maps API. Its lightweight and fast characteristics ensure quick response times and high concurrency, which is beneficial for applications that need to remain responsive and interactive. Additionally, the Express framework, a minimalist tool for building web applications and APIs, offers flexibility in setting up backend logic. The extensive library of packages available through Node Package Manager (NPM) further accelerates development. Together, these features allow Node.js to efficiently manage real-time API calls and deliver a seamless user experience.

## Python Django or Flask Web Application

Django and Flask, as high-level frameworks, streamline web development by offering abstractions that accelerate the development process. Both frameworks benefit from Python's rich ecosystem, which includes libraries for data processing, machine learning, and web scraping. Django is a full-featured framework that includes built-in ORM, authentication, and an admin interface, making it ideal for rapidly building complex applications. In contrast, Flask is lightweight and modular, offering flexibility to choose components as needed, which is perfect for smaller projects or microservices. For a project, Django’s robust features can efficiently handle user authentication, data management, and API integrations, while Flask’s flexibility allows for a more tailored approach, making it well-suited for developing specific application components as microservices.

## Final Analysis

For building a route generator application, ASP.NET Core Web Application is the best choice due to its scalability, cross-platform capabilities, and strong support for third-party API integrations like Google Maps. Unlike ASP.NET Web Application (.NET Framework), which, while robust, is less flexible in terms of cross-platform deployment and may involve more legacy system considerations. Blazor App, though innovative with its C# client-side logic and interactive components, adds complexity with its dual approach of Blazor Server and Blazor WebAssembly, which is not necessary for this project. Node.js with Express, while excellent for handling concurrent requests and real-time API interactions, lacks the structured MVC architecture and comprehensive security features provided by ASP.NET Core. Python frameworks like Django and Flask offer high-level abstractions and utilising Python’s ecosystem, but ASP.NET Core's extensive tooling and integration capabilities are more suited for a semi-complex, high-performance web application. Given its widespread use and abundant learning resources, ASP.NET Core ensures a smooth development process with ample support, making it the ideal choice for this project.