![A blue and green text

Description automatically generated]()

Cross Platform Development: CA1

John Devaney

L0016171

Applied Computing

1. **Introduction**

When making an app you will sometimes have to build it differently for each type of device. That can take up a lot of time and can be expensive. Now there are tools and methods that let you build one version of the app that can work on all those devices. This is called "cross-platform development." It saves time and money, and it attempts to make sure the app looks and works the same on every device.

There are several different cross-platform development tools that can be chosen from for this critical evaluation, such as Flutter, React, Adobe PhoneGap, Appcelerator, Xamarin, just to name a few. For my critical evaluation I will be focusing on React, QT, and .NET MAUI. (‘12 cross-platform development tools you should know in 2023’ 2023)

.NET MAUI is good if you already know a lot about Microsoft's code, tools, and frameworks, like C# and .NET. It can make your app feel like it's made for each device, which is great for the users. But it's not as popular as some other options since it not very well known.

React is another option, and it's popular with a big community of developers. This means there are a lot of extra pieces you can use to build your app, which is handy. But it can be slower for apps like EquityX that need to do lots of things quickly.

QT well-tested libraries to develop your code, it has modern UI features and uses wide set of libraries that are perfect for cross platform development.

Choosing the right toolkit is super important for making EquityX or any other app for that matter a successful app. You need to think about what your team knows, what the app needs to do, and how you want it to look. Each toolkit has its own good and not-so-good points. So, your job is to pick the one that's the best fit for your app's special needs.

1. **Technology Evaluation**
2. .NET MAUI

.NET MAUI is a framework for creating apps using C# and XAML. It allows developers to build apps for Android, iOS, macOS, and Windows using a single shared codebase. .NET MAUI is open source, mostly focusing on performance and extensibility. It allows you to apply most of your app's logic and user interface using a single codebase.

.NET MAUI is designed for developers who want to create cross-platform apps, share UI layouts, code, tests, and logic using C# and XAML. It makes it easier to use features from different types of devices, like Android and iOS. It's like a middleman that helps your app talk to these devices in a common language. So, you don't need to worry about all the different languages each device speaks.

It offers a collection of controls, a layout engine, data-binding support, customizable UI element handling, cross-platform APIs for device features, and a single project system for multi-targeting. It also supports hot reload, allowing you to make changes to code and UI in real-time without rebuilding the app which comes in very handy and saves plenty of time. (davidbritch 2023)

Pros:

* Familiar for Microsoft Developers: A good choice for those already comfortable with C# and .NET.
* User Interface Flexibility: It uses XAML, allowing for flexible UI design.
* Blazor Integration: Supports Blazor, which can be handy for web and mobile apps with shared components.

Cons:

* Newer Community: .NET MAUI is still relatively new with a smaller community, which means there are fewer resources and support.
* Platform-Specific Code: While it aims for native integration, sometimes, you might need to write platform-specific code.

1. React

React, is an open-source JavaScript library developed by Meta for building user interfaces. It simplifies the process of creating user interfaces in web applications.

React encourages a modular and component-based approach to designing interfaces, making it easier to build and maintain user interfaces. By breaking down interfaces into smaller, reusable components, developers can manage their projects more effectively. React applications consist of components, each with its specific purpose and functionality.

React also uses JSX (JavaScript XML), an extension of JavaScript, to embed HTML-like templates in JavaScript code. JSX is commonly associated with React and makes user interface elements simpler.

React components can be created as JavaScript functions or ES6 classes. Function components are simpler and more common, while class components offer more advanced features. (‘Overview of React.js’ 2023)

Pros:

* Community Support: React has a large and active community, which means more libraries, plugins, and components.
* Real-Time Changes: Its hot reloading feature lets you see changes as you make them.
* Code Sharing: It allows substantial code sharing, saving time.

Cons:

* Performance Issues: React apps can be slower for intricate applications like trading apps.
* Bridge Between JavaScript and Native Code: The bridge can slow down performance.
* Compatibility with Legacy Systems: Integrating React into existing projects or legacy systems can be complicated.

3. Flutter

Flutter is a framework based on the Dart language. It stands out by rendering the user interface directly on the device's canvas instead of relying on native frameworks. Flutter offers a range of widgets for creating modern applications, and its widget-based approach simplifies UI design. Widgets in Flutter are customizable and support animations and gestures.

A key feature of Flutter is its reactive programming approach. Widgets can have states, and by changing a widget's state, Flutter intelligently updates only what's necessary, avoiding unnecessary rendering. (‘Flutter - Introduction’ 2023; ‘Flutter architectural overview’ 2023)

Pros:

* High Performance: Flutter excels in performance because it uses Dart, a compiled language, and has its rendering engine.
* Real-Time Updates: It offers hot reloading, making the development process faster.
* Rich Widgets: Flutter's library of customizable widgets helps in creating impressive user interfaces.
* Easy to learn language: Flutter uses the Dart programming language, which is very easy to use and can be a great advantage compared to learning more complex languages.

Cons:

* Fewer Native Libraries: It has fewer Native libraries than other frameworks which can affect hardware and software integrations.
* Large App Size: Flutter apps can have larger file sizes because they include the Flutter engine. This can affect download times.

**Conclusion**

Cross-platform development is all about building one version of youre app that works on different devices, which will save time and money. You can pick from various tools to do this, like .NET MAUI, React Native, and Flutter.

NET MAUI is a good choice if you're already familiar with Microsoft technologies like C# and .NET. It allows your app to adapt well to many devices, but it's not as widely used at this point, so finding resources and support might be a bit more challenging.

React Native, benefits from a large and active community of developers. This means you have access to many libraries, plugins, and components. However, for more complex applications it might not provide the level of performance needed due to potential performance issues.

Flutter stands out in terms of performance because it utilizes Dart, a compiled language, and has its rendering engine. Flutter's widget-based approach simplifies UI design and offers real-time updates. The downside is that it may have fewer native libraries, which can impact hardware and software integrations, and Flutter apps can have larger file sizes, affecting download times.

Choosing the right one is important, and it depends on what you or your team knows and what your app needs to do.

1. **Architectural Design**

I plan to use the Model–view–viewmodel (MVVM) pattern when designing my project. This will make my code more easily manageable. The MVVM pattern will help to separate the business and presentation logic of an application from the UI.

**Model**

The model represents the data and business logic of the program and is responsible for managing, processing and sending data to the ViewModel, which then communicates to the view to display and interact with said data. (davidbritch 2021)

**View**

The View is responsible for displaying the structure, layout and appearance on screen. Each one of the views is written in XAML code, with code behind it that doesn’t contain business logic. All my pages will be in this folder. (davidbritch 2021)

**View model**

The view model creates properties and commands that can be linked to by the view for data binding. It also keeps the view informed about any alterations in its state by triggering change notification events. The View Model is directly connected to the View and the Model. (davidbritch 2021)

**Services**

The services are components or modules that are responsible for handling different aspects of an applications functionality other than what the view model handles. Services act as a man in the middle between the viewmodel and external data sources such as APIS.

**Visual Design**

Here I will show my designs for my EquityX app.

Home Page:

A screenshot of a phone

Description automatically generated

Market Page:  
  
A screenshot of a computer screen

Description automatically generated

Flyout Page:  
A screenshot of a login screen

Description automatically generated

Statistics Page:  
A screenshot of a graph

Description automatically generated

**References**

‘12 Cross-Platform Development Tools You Should Know in 2023’. Accessed 27 October 2023. <https://www.intuz.com/blog/cross-platform-development-tools>.

davidbritch. ‘What Is .NET MAUI? - .NET MAUI’, 30 January 2023. <https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui>.

‘Overview of React.Js’. Accessed 27 October 2023. <https://patterns.dev/posts/reactjs>.

‘Flutter - Introduction’. Accessed 27 October 2023. <https://www.tutorialspoint.com/flutter/flutter_introduction.htm>.

‘Flutter Architectural Overview’. Accessed 27 October 2023. <https://docs.flutter.dev/resources/architectural-overview>.

davidbritch. ‘The Model-View-ViewModel Pattern - Xamarin’, 8 July 2021. <https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>.