**Technologies for mobile app development**

For this particular mobile app, some of the choices we have would be XML, React Native, and Ionic (a framework using web languages like HTML, CSS, and Javascript) for front-end programming and Java, C#, and Kotlin for back-end programming. Our chosen languages will be determined based on the following criteria: ease of use, maturity as well as availability of resources online, domains covered, popularity, performance, scale, and speed of the code, and ability to test and debug.

Concerning front-end technology, since React Native is based on Javascript, a language that we are comfortable with, it should not take long to get up to grips with it. Similarly, we also have sufficient experience using Java and C#, particularly in app development. Kotlin is similar to Java, but is very expressive and much more concise, making it a strong contender for our back-end despite us not having much experience using it. XML is a markup language really similar to HTML, which we know, and is easy and intuitive . In contrast, Ionic is a bit more foreign to us, though it is important to note that it is an easy-to-use tool if we take the time to learn it.

Java and C# are both very mature languages with a wide array of resources available for consultation on the internet. XML, Kotlin, and React Native may not be as mature as Java and C#, but they still have a decent amount of resources available due to their similarities with certain popular languages. Ionic is mature and does have resources available on the internet, but they are not as abundant as for the other choices.

Java is one of those general-purpose languages that are capable of being used in almost any situation, whether it is used for numerical computing, creating video games, or developing a project’s back-end. Similarly, Kotlin is also a general purpose language that can be used in just about any situation where Java is appropriate. Although C# is not used in as many situations as these languages, it is, however, widely used for developing desktop and web applications as well as in game development in Unity. On the other hand, React Native and Ionic are specifically designed to help develop mobile applications. XML is also specifically designed for a single task in the sense that it is mostly used for encoding documents in a way that is readable to both humans and machines.

Unsurprisingly, Java is the most popular language out of our options followed by C#, which is also well-known. Due to their niche nature, React Native and Ionic are not as popular. Although Kotlin is quite similar to Java in many ways, it is not nearly as popular as the latter due to the fact that Java has been around for a much longer amount of time, thus allowing it time to get established in the programming community. While XML is well-known as a language that encodes documents, it is not nearly as popular as the other bigger languages such as Java and C#.

Java is an exceptionally fast and scalable language, being able to do many different operations (like memory allocation and deallocation, for example) efficiently, but it does require a lot more memory than most other languages. Like Java, C# is also fast and scalable, but is more efficient when it comes down to memory usage. React Native has adequate running speed (may be less than the other two) and its scalability depends on the situation, but it requires little time to build its code. Kotlin shares similar characteristics with Java, but it should be noted that Kotlin requires a bit more time to compile its code than Java does. XML has good performance, but, due to the nature of this small project, it would appear as very fast in this case. Ionic does perform fast, but the overall performance depends heavily on the language being used in it.

Troubleshooting in both Java and Kotlin when using the default modules is sufficient for the scope of this project. Similarly, using unit tests in C# is also enough for this case. Testing and debugging in React Native is possible as it has a few tools related to that, but, like all the other languages being considered, we would need to import a library to gain access to a wider array of testing tools if troubleshooting becomes more important for this project in the future. Testing capabilities in Ionic generally depends on the languages being used in it, so it can vary. XML is meant purely for displaying information on a document and as such it should be visually obvious which parts of the code represents which parts of the document, making it incredibly easy to fix bugs that may be present in the document.

After comparing these options in each of the six chosen categories, we have come to the conclusion that it is best to use XML and Java as the front-end and back-end languages respectively for this project. The key reasons behind this decision includes the fact that we had the most experience using Java than any of the other options, the time constraint that may prevent us from learning the other languages/frameworks, and the simple nature of the application that can be easily satisfied by these simple languages.

As well as everything above, we found that Android Studio uses XML and Java as the front-end and the back-end respectively, and having used it previously (in 207 and other side projects), we decided to give it a shot and make this project Android-only. Android Studio really gels everything up nicely, and makes it much easier to use XML and Java than any other language we might have used, per se.