Name:

Instructor:

Course:

Date:

Why I Chose C#

For the vast majority of programmers, a decision to invest time and effort into advancing one's skills and knowledge of a certain programming language is motivated by the utilitarian reasoning of applicability and financial prospects associated with it—as well it should be. An example of Javascript—a language that is immensely popular and promising despite being infamous for its rushed design and often clumsy syntactic decisions—showcases that the use of a programming language is often caused by external factors of its popularity and exposure as opposed to its internal soundness or orderliness. By applying this reasoning to my personal aims and interests as a programmer, I willingly chose to make C# my main language of computer programming and to learn it deeply beyond a basic familiarity. In reaching such a decision, I evaluated the areas where every language I considered is used and tasks for which it appears to be an optimal tool, and realized that the opportunities provided by C#, such as its specific use in .NET networks, graphical design, and game engineering, attracted me the most.

In analyzing the advantages of C#, it is natural to start with the advantages and disadvantages of the environment for which it was tailor-made. .NET Framework is a software

framework that basically serves as a buffer between the programming languages and operational system, assuring their interoperability. .NET Framework was developed by Microsoft, thus making it a leading framework for both the majority of stationary computers that operate on Windows, as well as a diverse series of mobile programs and other features represented within Windows CE. C# was specifically designed as a programming language that is best suited for being interpreted within the .NET Framework and adapted into commands within pre-established task libraries. In practice, the framework is integrated into a software development kit (SDK)—a package that includes "the compiler, runtime execution engine, the framework of pragmatically accessible functionality that the runtime can access (see API later in the chapter), along with any additional tooling (such as a build engine for automating build steps) that might be bundled with the SDK" (Michaelis, 17). While the freedom to choose between available SDK options provides some flexibility regarding advanced programming, the fact of SDK usage makes C# an attractive option for web application development.

While a structural connection between C# and .NET frameworks appears to be one of the 'internal' features of the language and its intended purpose, what defines the popularity of C# and its high demand for programmers with advanced knowledge of C# is its popularity within the realm of modern development. First and foremost, C# is unprecedentedly widespread in game development at least for three reasons. Firstly, C# happens to be a core language of the Unity game engine—one of the most popular game engines in the world (Mkhitaryan). Secondly, the environment of Microsoft Visual Studio—the integrated development environment that

operates with .NET framework structures—turns out to be an efficient method of programming, as it provides a legitimate shortcut for many routine processes. Thirdly and finally, as Microsoft remains the leading provider of operating systems for personal computers in the world, the relevance of C# prevails, and the popularity of the language in this day and age allows to reliably predict that even in case of the sudden death of Microsoft, the language will outlive the company at least for some time.

Lastly, some personal reasons for choosing C# help in choosing it as a 'native language' for an aspiring programmer. Being a powerful object-oriented language in the likes of Javascript and Python, it provides a little bit more of a challenge compared to these two alternatives, yet does not overwhelm a beginner with its complexity (Utley). However, while C# is slightly more difficult to learn, it provides many more possibilities in implementing complex projects, cooperating with other programmers over the same task, and integrating other languages within its programming environment. Moreover, within Microsoft Visual Studio, it is possible to earn tangible achievements as an aspiring programmer soon after starting to adopt the language. Due to the contribution of the IDE, the resulting programs end up much more appealing and interactive with much less effort. The ease with which one's first results are earned in C# might explain the warmth which many programmers have towards the C# language.

To conclude, C# appears to be a proper choice for an aspiring programmer who aims at transitioning from a beginner to an expert. It is appealing, highly demanded, and contains some structural features that would come handy with an increase of complexity of tasks faced by a

programmer. This does not mean, however, that my personal choice of a programming language would be somehow limited to C#, especially as there are many useful methods of combining the strength of C# and complementary languages such as Javascript and SQL. Nevertheless, it is C# that appears to me as the most appealing, engaging, and useful means by which a deeper and more complex understanding of computation and programming can be achieved—thus making it a perfect choice for a 'native' programming language.

Works Cited

Michaelis, Mark. *Essential C# 7.0*. 6th ed., Addison-Wesley Professional, 2018.

Mkhitaryan, Armina. "Why Is C# Among The Most Popular Programming Languages in The World?" *Medium*, 13 Oct. 2017, medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb.

Utley, Craig. "Why You Should Move to C#." *TechRepublic*, 4 Nov. 2002, www.techrepublic.com/article/why-you-should-move-to-c/.