CS39440: Major Project - Choosing a Language

WORKING TITLE: "VIDEO GAME BACKLOG APPLICATION"

Department of Computer Science,
Aberystwyth University

Last updated: **12th February, 2024**

v1.0 - Release

Produced by:

Kal Sandbrook
kas143@aber.ac.uk
on *Computer Science - G400 BSc*

Supervised by:

Dr. Edore Akpokodje
eta@aber.ac.uk
Lecturer in Computer Science

This document discusses various programming languages and frameworks that were considered for the development of this project. It lists the pros and cons of each language and gives a potential final decision.

The following requirements and needs were considered when picking a shortlist of languages:

- The language must have good support for **object oriented programming**, due to the nature of the application.
- There should be a good API available for **interfacing with a database**, as the application will need to store and retrieve persistent data.
- Support for **HTTP requests and responses** is required, as the application will need to communicate with an external API. This means the language should also be able to support **asynchronous programming**.
- A desired feature would be to have good support for **native UI development** (such as Qt), to ensure a performant result.

Based on these requirements, the following languages were considered:

Language	**Characteristics**	**Pros**	**Cons**
C# (`.NET`)	An object-orinted programming language with static typing. Very similar to Java. Has good UI support for Windows but not so much for other operating systems.	• Good support for HTTP Requests with the `HttpClient` library. • Entity framework to simplify database interaction. • Supports **WPF** for native UI development. Doesn't support Qt.	• Windows-only support for native UI development. • `.NET` Libraries are not easy to work with.
C++	An object-orinted language with static typing.	• Supports Qt for native UI development. • Good performance. • Has an API for HTTP requests and database interaction.	• Not as easy to work with as other languages. • **No prior experience**, and no time to learn.
Python	A high level language that *can* be object-oriented. Dynamically-Typed.	• Support for HTTP requests with the `requests` library. • Supports databases with `sqlite3`. • Good support for async programming. • PyQt for UI.	• Slow performance. • Dynamic typing is messy and can lead to bugs. • No visibility control for object oriented. (No private/public/ protected)

Java	An object-orinted language with static typing.	• Good support for HTTP Requests with the `HttpClient` library. • Supports databases with `JDBC`. • Supports **JavaFX** for native UI development. Also has Qt bindings with QtJambi.	• Slow performance.
JavaScript	A high-level interpreted language primarily used for web development.	• HTTP requests with the `fetch` API. • Supports databases with `IndexedDB`. • Good support for async programming. • Electron for UI. Easy to design.	• Slow performance. • Dynamic typing is messy and can lead to bugs. • Electron is not a native UI.

Based on the above table, I think that there are two main options going forward: **Java and Python**.

Both languages have support for HTTP requests and databases, and both have good support for native UI development. The main difference is that Python is a dynamically typed interpreted language, mainly used for scripting - whilst Java is a statically typed compiled language, mainly used for enterprise applications. I am familiar with both of these languages, so experience is not a major factor in the decision.

Going forward, I will be considering both languages in more detail by write some small test applications in each language to see which is more suitable for this project.