The Gaming Room

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	<23/10/2021>	Jacqueline Amoah	Added information related to the software design.

Instructions

Fill in all bracketed information on page one (the cover page), in the Document Revision History table,and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

Executive Summary

The Gaming Room needs a web-based-game that can serve multiple platforms based on its popular Android game, Draw It or Lose It. This game would be like Win, Lose or Draw, a popular game show fromthe 1980s.

Design Constraints

- Game needs to be run on multiple platforms
- Multiple teams of multiple people
- Checks for unique team/game names must be present
- Only one instance of the game must be allowed at any one time

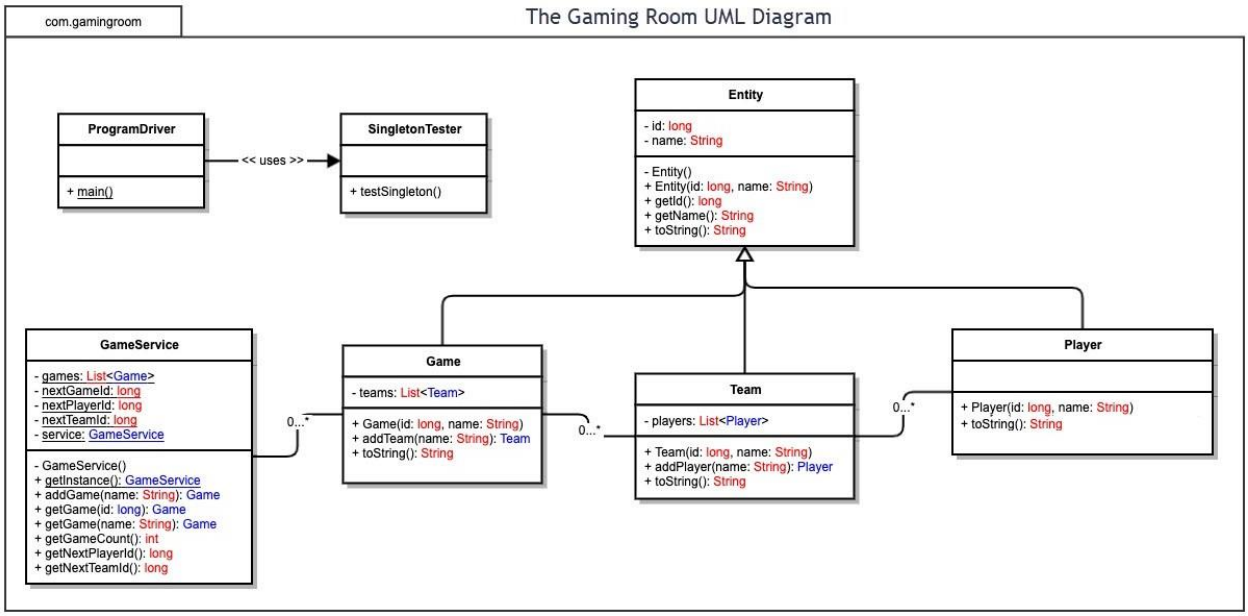
These constraints are what the game itself needs. In terms of development, The Gaming Room would like this game to be run on multiple platforms, so we would need to take the working Android code and port it over to the other platforms, which may require more developers to write this code in those other languages.

System Architecture View

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

Domain Model

The Entity class is our super class, and the Game, Team and Player classes all inherit directly from it. The GameService has a reference to the Game class, which has a reference to the Team class which has a reference to the Player class. All of these classes having a reference to another is done through aggregation. Also included are our SingletonTester and our ProgramDriver class. Our ProgramDriver classis where any executions of our application take place, and it uses our SingletonTester class. The SingletonTester class also has a use relationship with the ProgramDriver class. The way these classes are set up allows for one game to go on at a time with multiple teams (from a list) with multiple players (from a list).



Evaluation

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client's requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering theindicated information.

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	Mac does have Mac OS X server available for use. Per Apple's website, Mac OS X Server is only $20, so it would be inexpensive to implement. Mac, however, is not as popular as say Linux or Windows for performing these tasks.	Linux is interesting here as it has many distributions that have server capabilities. Linux Server would be low-cost, and open source (which provides a lot of resources). Not many users are savvy with Linux, so you would need someone who is familiar with Linux running the server.	Windows offers Windows Server. Looking at Microsoft's website, it might be costly to implement, but it is fully functioning. Windows is likely the most used operating system, so finding users to operate Windows server would be a lot easier.	Given that mobile devices do not necessarily have the power that computers do, hosting a fully-fledged server on one may not be the best option compared to computers. Running servers on mobile devices is the most advantageous in terms of cost, as there is little to none to get one started.
Client Side	Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Mac would need less time and someone who does not have as much experience with Mac would need more time.	Cost would be low (if there even is a cost) with Linux, as it is open source. Maximum time and experience would be necessary, as Linux is not commonly used and you would need someone who is apt with Linux and allow them time to work, as Linux can be difficult even for someone with experience.	Cost would be like a Windows setup, as these operating systems are not open source. Time would depend on expertise, as someone who has experience with Windows would need less time and someone who does not have as much experience with Windows would need more time.	Cost would not be too much of an issue with Mobile devices. Experience may not be too much of an issue, as mobile devices can be easier to work with. More time would be needed, as there are multiple operating systems and multiple mobile devices that would need to be worked on.
Development Tools	Swift would be the more common language used to write applications for Mac. There are multiple IDEs that can be used for Swift, such as Atom	Eclipse and Atom are commonly used IDEs on Linux. Eclipse is primarily used for Java, although it can support other languages like C+	Eclipse and Visual Studio are popular IDEs for Windows. Visual Studio can be used for developing in HTML, C# and JavaScript among	For iPhones, the development tools are like those for Mac, and iOS apps are typically writtenin Swift, though iOSand macOS are different in terms of

Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: Based upon all my research, I believe that Windows would be the best operating platform to use, as it is the most widely used operating platform, as well as the one that most people have a working knowledge of. There is an abundance of IDEs that can be used with Windows, and the total cost to utilize it is typically lower.

2. **Operating Systems Architectures**: The Windows architecture allows for applications to utilize the platform's kernel processes without directly affecting those processes. In other words, applications can utilize the power of Windows to have a GUI/window set up, access to memory and other vital processes that make the application without inadvertently affecting the processes that make the operating platform work.

3. **Storage Management**: Windows has Disk Management and Storage Sense built into the operating platform itself. Windows also has a Disk Cleanup tool that can be used. Disk Management is a Windows system utility that is mainly used for advanced storage tasks, while Disk Cleanup and Storage Sense are used to help maintain the storage on the system by deleting unnecessary files that are taking up space.

4. **Memory Management**: Windows has Memory Management built in as a system utility. We would need to build database for all the game's image files so that they can be easily be accessed by the application.

5. **Distributed Systems and Networks**: A client-server distributing system will be utilized here, as we will have each client application depend on the single server application for our game, so that each client application can be developed to that client's system's strengths. A strong server network would also be needed, as this game's success depends on multiple clients connecting toa single server to play one game altogether.

6. **Security**: Windows Defender is built into Windows as a security function for any Windows system. We would need to encrypt all of the data that is being sent back and forth through tried and true encryption methods.