Draw It or Lose It

CS 230 Project Software Design Template

Version 1.4

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	07/21/24	Eric Foster	Updated summary, requirements, constraints, and domain model.
1.2	08/03/24	Eric Foster	Updated the evaluation section and table.
1.4	08/22/24	Eric Foster	Updated the Recommendation Section.

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.

<u>Executive Summary</u>

The client (The Gaming Room) is desiring to expand their game called Draw It or Lose It, to not only be available as an Android app, but is seeking to develop a web-based version that serves multiple platforms. CTS has been hired to assist in setting up the environment and streamlining the development of the game. The game will include the ability to have one or more teams involved, with each team having multiple players assigned to it. The game and team names must be unique, and only one instance of the game can exist in memory.

<u>Requirements</u>

The game must have the following requirements meant:

- A game will have the ability to have one or more teams
- Each team will have multiple players
- The game and team names must be unique
- Only one game instance should exist in memory

<u>Design Constraints</u>

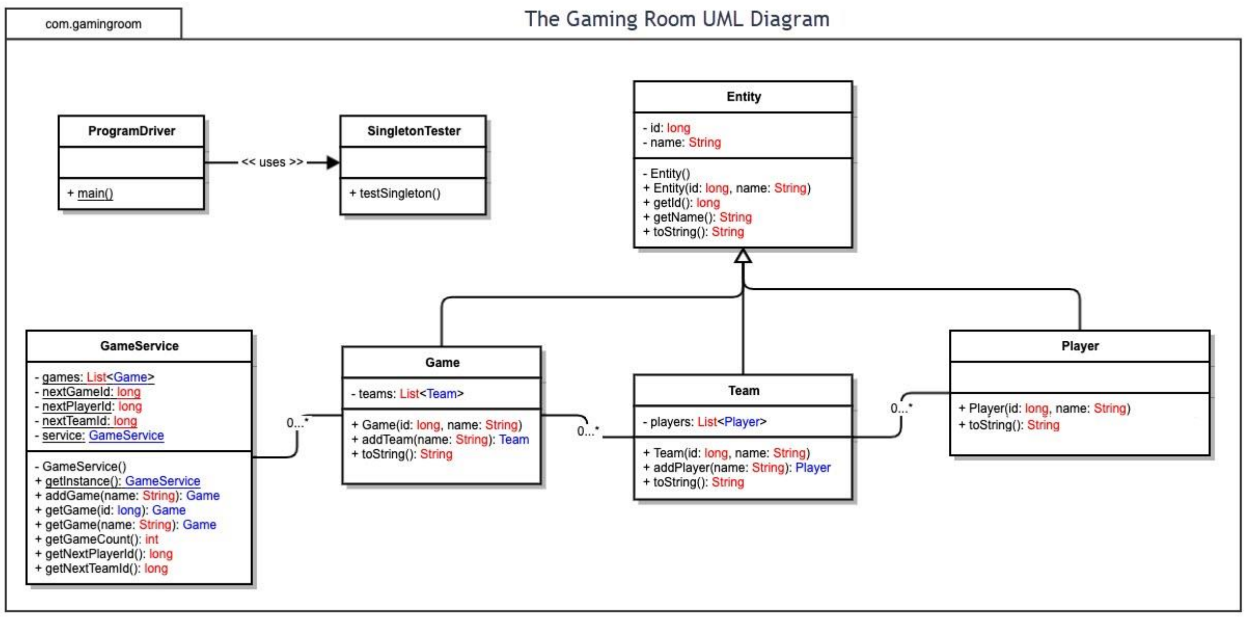
The primary design constraint that we will face is multi-platform development. We must ensure that we either have a team that has experience and knowledge in cross-platform development or will need to ensure that all specialized teams are working in conjunction with each other to ensure that development on each platform is the same outcome no matter the platform being utilized.

<u>System Architecture View</u>

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.

<u>Domain Model</u>

Starting with the ProgramDriver Class, this class uses the SingletonTester Class to test our singleton pattern and ensure that only one game instance exists in memory. The arrow from the ProgramDriver Class to the SingletonTester Class is a direct association relationship, indicating that these classes directly communicate with each other. Moving on to the Entity Class and the classes below, in the UML, the Entity Class is a parent class to the Game, Team, and Player Classes, with those three classes inheriting from the Entity Class.



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 the indicated information.

Development Requirements	**Mac**	**Linux**	**Windows**	**Mobile Devices**

Server Side	Mac has quite a lot to offer. With Mac being Unix based, this allows for more security to be present. With that, Mac is heavily known for being very performance heavy, meaning that Apple strives to provide product solutions with high performance. Mac is also known to be very user-friendly, more so than windows as they focus more on a simple UI structure, allowing for learning the system to be easy. With the above advantages stated, comes some disadvantages that follows. Mac is known for being more costly to utilize. On top of that, Mac is also known for being less reliable when assessing for scalability due to Mac not being designed to host large-scale solutions. Finally, Mac may have limited resources, making it harder to develop a high performing and scalable solution.	Linux is an open source, Unix like OS. With that comes some favorable advantages. First, because Linux is open source, it is free to use, thus eliminating the licensing cost and making it more cost effective than other options. Also, because Linux is a Unix like OS, Linux is highly known for being very secure, allowing for more heavily managed and monitored security. Linux is also known for being reliable and stable due to an exceptionally large community of active developers constantly updating and working on Linux to make it better. Once again, where there are advantages, there are also disadvantages. Linux has a very steep learning curve, making knowledge of the use of the command line (CLI) a requirement as that is how most configurations are done. With that, hosting and maintaining a server on Linux	Windows is a popular option for many users due to its high customizability. Because of it's high popularity, you can take advantage of the familiarity to reach out to more users. Utilizing windows allows for easy integration of Microsoft based solutions such as MySQL, .NET applications, and more. Windows also utilizes a strong Graphical Interface (GUI) that makes management of a server and the configurations easier as it does not rely as heavily on command line as other OS options, thus lowering the labor cost needed to maintain the server. With all that being said, the cost of licensing can be quite high as you must not only consider the costs for licensing, but also the cost of licensing for client access as well, making this option more expensive for all	Mobile devices do offer some advantages. Some of those advantages might include factors such as portability, low initial costs, and ease of use. However, for hosting a server, this option has severe limitations. First, due to size, mobile devices are known for having limited resources available such as limited storage, limited power, and so on. Mobile devices also rely on the use of a battery in which those batteries are not designed for long-term operation that a server may require. Finally, the UI of a mobile device is small and may hinder a persons ability to effectively manage a server without running into speed limitations due to size, thus creating a bottleneck in productivity and efficiency.

Client Side	When assessing the client side, we must consider a few factors. When assessing costs for Mac development, there may be additional costs in client side development associated with development tools such as a dev account for the app store or if utilizing a cross-platform program, costs may be included with the use of tools such as VSCode or JetBrains products. Time can be quite complex for this type of project if utilizing native only options, meaning if you utilize only the tools that are available to develop with for that specific platform, it may involve having to create multiple versions on each system, thus increasing the time of development as well as the need for specialty knowledge in developing in each OS system. This can be avoided by utilizing tools such as React Native or Flutter, thus eliminating the need to develop	Associated costs for Linux is similar to Mac. It all depends on the tools utilized to host and maintain the codebase. Some tools may be free but costs may be associated with the use of cloud infrastructures and other aspects of development such as CI/CD automation and other development aspects. Time can also once again be quite long and complex. Depending on if you are utilizing system specific development or cross platform development tactics will all play a role on how fast the development team can get the codebase running and how fast they may be able to respond to issues. If going with a system specific approach, it may take more time by nature when handling multiple Linux environments and may cause the process to become harder to maintain due to the added complexity. If	Once again, associated costs for windows will also be similar to Mac and Linux. It all depends on the tools utilized to develop and maintain the codebase. Also additional costs may be involved when utilizing windows servers and other Microsoft solutions or enterprise solutions for development. Once again, time spent is dependent on whether you choose to do native development (system specific) or cross platform development. If choosing the native route, the development team will need knowledge in developing and managing systems within the windows environment, not to mention that they would need to develop for multiple windows environments. If you choose to utilize a cross platform development approach, then the dev team would need	As stated with the other three OS systems, Costs can be associated and involved with the tools we utilize. For system specific development, IOS development has tools like XCode that is free but requires a Dev account for app store distributions which costs money. For android development, you can use tools such as Android Studio which is free but additional costs may be involved when utilizing libraries and other resources. In addition to those tools, costs may be associated with licensing and hosting of applications and the application infrastructure. As for time, once again it depends on the path of development you choose to go with. If going the native route, it may take more time and may cost more to manage multiple versions and environments. If going with the cross platform approach, you may be able to save on time and money by working off one master codebase that works for multiple

Development Tools	Mac provides access to utilize many different programming languages for use in development. Some may include, but are not limited to: C++, C#, Objective-C, Java, JavaScript, and Python to just name a few. Mac also supports many Integrated Development Environments (IDE's) including XCode, AppCode, VSCode, and CLion to name a few. Mac also provides support for various package managers. The most popular are Homebrew and Swift Package Manager. Some of these tools offer both free and paid versions based on the need.	Linux provides support for development in multiple programing languages such as Java, JavaScript, C/C++, Go, and Rust to name a few. Linux also provides support for many IDE's such as VSCode, Eclipse, CLion, and NetBeans to name a few. Linux also has support for additional tools such as Docker and a few debugger tools to help assist with the development of an application. Some of these tools offer both free and paid versions based on the need.	Windows provides heavy support for development utilizing various programming languages such as Java, JavaScript, C#, C++, Rust, and Go to name a few. Windows also provides support for various IDE's such as Visual Studio, VSCode, Eclipse, and CLion to name a few. Windows also has support for additional tools such as Windows subsystem for Linux, Docker, and Process Explorer to name a few. Some of these tools offer both free and paid versions based on the need.	For mobile development, if choosing to go the native route, IOS offers Swift and Objective-C, and Android offers Java and Kotlin. For cross platform development (Desktop and mobile development), Dart, and React Native are just a few of many options that can be utilized. As for IDE's, you can utilize XCode for IOS and Android Studio for Android. If utilizing cross platform development, then you can utilize a tool called flutter in conjunction with Dart. Additional tools may include the app stores for mobile applications, and Firebase for backend database usage. Some of these tools offer both free and paid versions based on the need.

<u>Recommendations</u>

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 on the Analysis above, my recommendation would be to utilize windows as the primary OS for development of the "Draw It or Lose It" game. The reason for this is for multiple factors. One major reason for this recommendation is that the number of users for windows is substantially larger than that of other operating systems. This means that not only is the client base larger, but that also means that most users will have an easier time on this OS due to familiarity, there will be a stronger pool of developers with a diverse set of skills, and windows offers a diverse list of development tools that may lack in other operating systems. It's these reasons that make windows the best option for cross-platform development.

2. **Operating Systems Architectures**: In an article documentation published by Microsoft, Windows 10 introduces the Universal Windows Platform, or UWP for short. With the development and implementation of UWP on windows 10 or better, apps that were designed for UWP could call both Win32 APIs and the .NET Framework, as well as call the windows RT APIs. Because of this, it makes this option the most optimal as it allows for development of a single app that can run on all devices (wwlpublish, n.d.).

3. **Storage Management**: My recommendation for the game "Draw It or Lose It" is to utilize cloud storage. By utilizing cloud storage, this will centralize all data in a safe location while also utilizing an optimized server that is designed to hold large amounts of data. This will help with the organization of all data being stored while also providing some security to that data.

4. **Memory Management**: For this project, the utilization of garbage collection and caching effectively will ultimately help drive a performance and optimization focused approach, ensuring that the game will run smoothly for all users without having performance issues due to incorrect usage of the allotted memory space.

5. **Distributed Systems and Networks**: My recommendation for ensuring that communication can happen upon all platforms is to integrate a centralized server that can manage factors such as game state, synchronize actions, and provide consistency across all platforms. With that in mind, my recommendation to combat outages and connectivity issues is to utilize two mechanics/methods. You can utilize a failover mechanism to handle server outages by having backup servers in place just in case the primary server goes out. The other method is to utilize data replication methods by implementing multiple databases so that data is consistently ready and accessible.

6. **Security**: Security in any application is very important and must be taken seriously. Regarding our project, there are multiple ways we can approach this topic. My recommendation is to implement a few different aspects. For user security, an authentication system is a must to ensure that all information is secure and is being utilized by the correct user. With that, we should focus on only storing the data necessary for the game to run. This will help minimize any risks that may be involved if a data breach does happen. Windows by default comes with some very strong built-in security measures, but to add on to that, we should also do security audits and regular updates to ensure that we stay on top of and up to date with any security

vulnerabilities that may be present. All of these methods will help to ensure that the user is safe and that we are doing our due diligence to ensure that our clients data does not get exposed.

References

wwlpublish. (n.d.). *Examine the Windows client architecture - Training*. Learn.microsoft.com. https://learn.microsoft.com/en-us/training/modules/explore-windows-architecture/3-examine-windows-client-architecture