Draw It or Lose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	07/15/2023	Mary Sanders	Updates to executive summary and design constraints, Domain model
1.1	7/26/2023	Mary Sanders	Added detail to Evaluations, Server/Client side and Development tools
1.2	08/11/2023	Mary Sanders	Update to Recommendations

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 has a game, Draw It or Lose It, currently available exclusively on Android OS. They would like to release a web-based version capable of serving multiple platforms. Like Win, Lose, or Draw, the game consists of teams that must guess a puzzle within a timeframe, using only stock-photos as clues. Staff currently do not know how to set up the environment and need assistance with streamlining development.

Requirements

- Web-based game must work on multiple platforms
- Game must allow for one or more teams, each with multiple players
- Game and team names must be unique
- Only one instance may exist in memory, requiring unique identifiers for game, player, team instance

Design Constraints

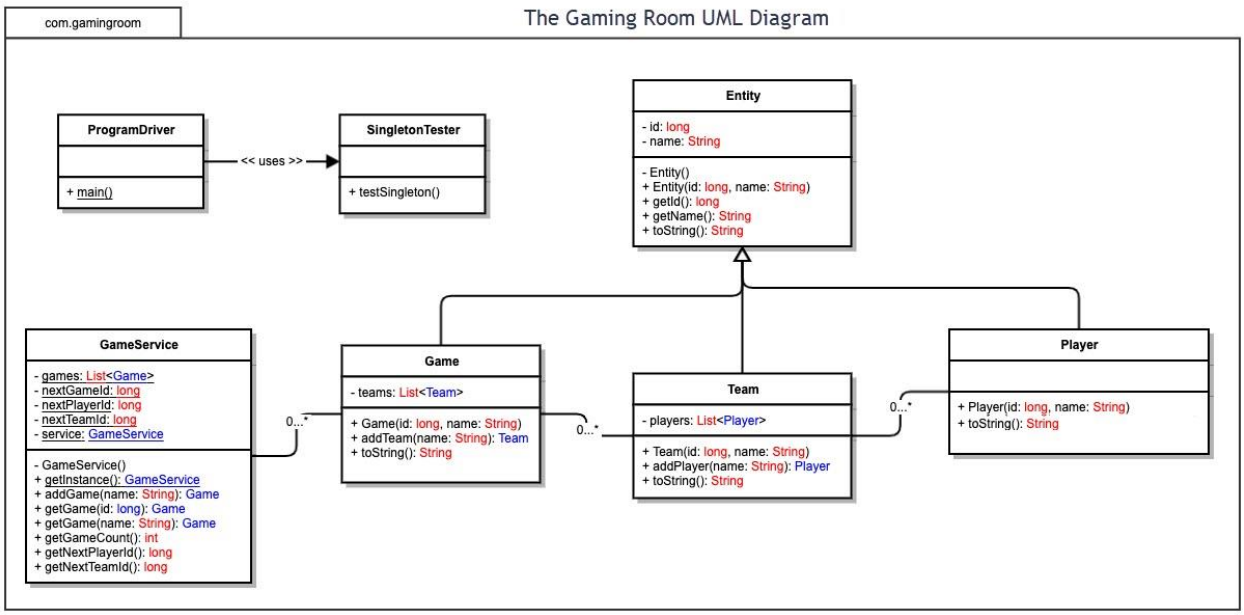
Since the game needs to work on multiple platforms, user experience across each should be consistent and like the Android OS UI already in place. Stock photos are being used for clues, so licenses will need to be acquired. One or more teams must be allowed per instance of a unique game, with each team allowing for multiple, unique players.

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 UML diagram shows how the Game, Team, and Player classes are derived from the Entity base class, an example of Inheritance. The GameService class is used to organize the game, team, and player instances, and can have anywhere from 0… *(many) unique games. Using the singleton pattern ensures that each instance is unique.



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	• Stability, crashes are infrequent • Easy to use, user friendly • Robust technical support • Licensing is expensive • Only available on Apple products	• Inexpensive • Open-source • Widely used • Secure • Not as user-friendly	• User-friendly • Robust technical support • Widely used, compatible with Windows tools • Licensing is expensive • Less stable option, known to crash	• Inexpensive • Convenient • Each device is different • Not compatible across different mobile OS • Less computing power

Client Side	• Good tech support • User-friendly UI • Stability • Licensing fees can be expensive • Requires knowledge of MacOS • Requires Apple products	• Inexpensive • Open-source • Works with many browsers • Stability • Not very user-friendly, learning curve involved • No customer service support	• Widely used, familiar • Customer service support available • Compatible with many browsers • Licensing may be expensive	• Convenient and portable • Cost is variable based on mobile device model • Difficult to implement • Requires knowledge of mobile web implementation
Development Tools	• Xcode IDE • Swift, Object-C, C++ programming languages	• Eclipse, PyCharm IDEs • Python, Java, C++, Ruby programming languages	• Visual Studio, Eclipse, PyCharm IDEs • Python, Java, C++, Ruby, PHP, HTML, CSS programming languages	• Android Studio, Xcode IDEs, depending on the OS • Swift, Python, Java, etc., depending on the OS

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**: I would recommend using Windows, as this is the OS that is most widely used. Even though there are licensing costs, there would be a negligible learning curve for the average player or developer.

2. **Operating Systems Architectures**: Windows is an operating system that has a GUI (graphical user interface) that allows the user to access applications and files within the file management system. Windows OS also has a kernel mode that works to execute code and manage memory usage without user involvement.

3. **Storage Management**: Microsoft Azure Storage Explorer is a cloud-based storage management service that offers pay-as-you-go flexibility and is easily scalable.

4. **Memory Management**: The Windows OS kernel manages how memory is allocated between the physical drive(s) and virtual memory.

5. **Distributed Systems and Networks**: This game is web-based, so is distributed to any user with an internet connection and a browser capable of loading the site. Hosting the site on a cloud-based server can help reduce the likelihood of connection outages.

6. **Security**: Windows Security with antivirus and firewall services in addition to the security offered by Azure.