THE GAMING ROOM

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	03/20/2024	Summer Bernotas	Added initial executive summary and design constraints

Executive Summary

The Gaming Room has decided to implement their android game “Draw It or Lose It” into a web-based application. This game is based off of the 1980’s television game “Win, Lose, or Draw” where teams must compete against each other to guess what is being drawn. This game will consist of teams and slowly rendered images from a stock library where the teams must guess what the image is. Currently, the game is only accessible as an android application but, The Gaming Room would now like this game to be accessible on multiple platforms via web-based application.

Design Constraints

- Must limit game and team names to be unique. Teams and games should not share the same name as this will create naming conflicts and issues when joining games
- Game must be developed for a web-based application and be compatible with multiple operating systems and browsers
- Only one instance of the game can exist in memory at a given time

System Architecture View

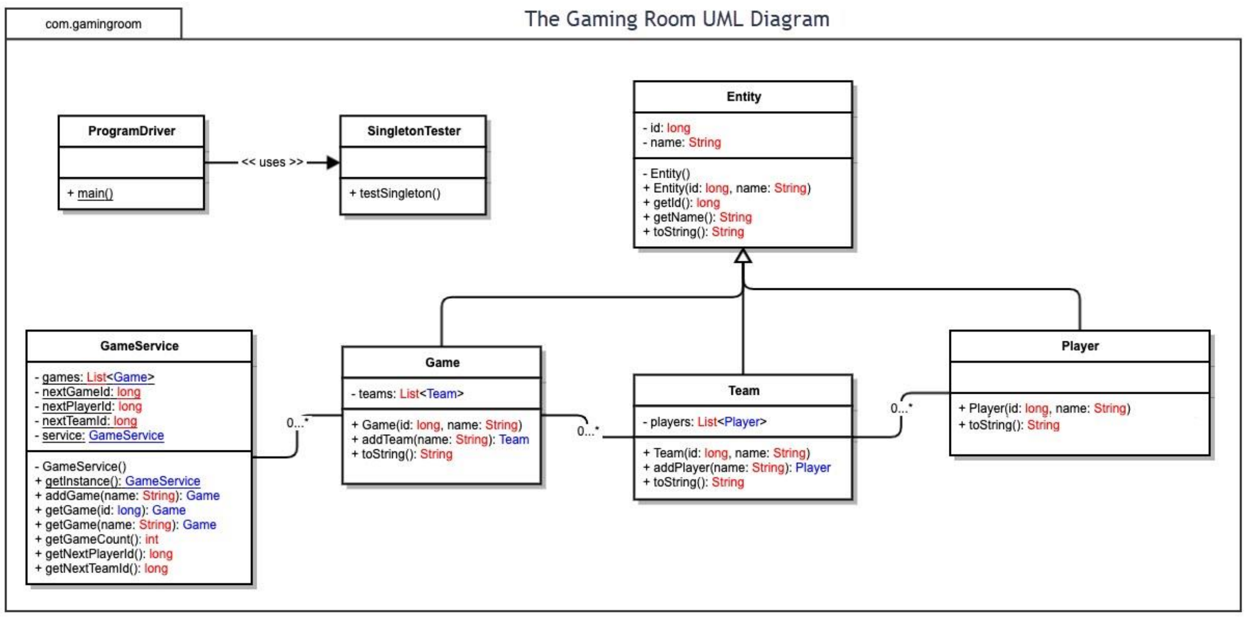
Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and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

Domain Model

The provided UML Diagram shows a base framework of the classes involved in the “Draw It or Lose It” game development. The *Entity* class serves as the superclass for the program. All variables declared here, “id” and “name”, are accessible to the *Entity* classes subclasses which are *Player, Team, Game,* and *GameService,* showing inheritance*.* The *Entity* class works as the “parent” to its subclasses, holding the game instance, teams within that game, as well as players on each team. We also see the GameService itself which is initializing the instance of the game.

The *GameService* class and *Game* class have a composition relationship. *GameService* will create and handle the games instances as well as basic functions to operate the instance itself as it inherits the “teams” variable. From *Game* we have the *Team* class which initializes the players together from *Player* to form the team, which then leads to the *Game.*

The *ProgramDriver* is where the “main” function resides and is the main executable for the application. In the *ProgramDriver* class we have the creation of the *GameService* singleton instance. This will help us allow only one instance of the game. This class, *ProgramDriver,* will handle the declaration of the teams and players as well.



<u>Evaluation</u>

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 servers are built on Unix which provides a relatively stable platform for web-based applications. Although stable, the scalability and hardware configurations are quite limited as compared to other operating systems and the hardware itself can be quite expensive. Mac does have good built-in commands in comparison for configuring and accessing the server itself and has good command line access using python.	Linux servers are built on an open-source OS which offers the most customization and flexibility than any other operating system. They handle scalability very well and have good security features as compared to other OS. Between Mac and Windows, Linux holds as the most affordable system since it is open-source and therefore free.	Windows servers offer the most amount of software compatibility and variety. It is also the most familiar among users which shows ease of use compared to other operating systems. Aside from this, Windows does have a few issues with security features due to the amount of web extensions, programs, and other, non-assured features that can be added. Aside from this though, it supports SQL which is vital when handling databases for any application and/or program. Although not the most affordable, it is easier to access than Mac and some other operating systems.	Mobile devices offer the best transportation capability as they are portable and offer an array of touchscreen and gesture-based interactions. Though mobile devices do come with a limited screen-size which can cause run-time issues along with many others when looking at functionality. Mobile devices do support many different features though in all areas but can get costly. This could cost less than Windows or greater than Mac.

Client Side	Mac is definitely the most expensive compared to others. The interface is relatively user-friendly but does offer a few learning curves for some features. Developing and maintaining is limited for multiple users. It is also most ideal for more creative and office type of users, as users in development, etc. will find that it is harder to use unless solely focusing on IOS and/or Mac specific development	Linux, due to its open-source system, is definitely the cheapest and most ideal for any software and/or web developers. It supports multiple clients and can be altered quite easily but the learning curve is much greater than that of Mac and Windows and will take a maximum amount of time to learn and understand	Windows offers a lot of plug-ins and tools to help users get the best possible experience. It fully supports multiple kinds of web-based applications and development as well as all rounded software including editing, writing, and gaming. Although not the most expensive though, licensing can get expensive as compared to Linux.	Mobile devices allow users to access applications and programs from anywhere as long as they have the proper connection. The most important aspect to think about is the screen compatibility and the limitations of connection. Mobile devices are also very limited as compared to other operating systems and laptop/computers. Only limited versions of application are commonly offered, and development and editing can be very hard due to its screen capabilities.
Development Tools	Most common programming languages used for MacOS are swift, C#, and basic web development languages and scripting – JavaScript, HTML, & CSS, which are paired with Node.js and React most commonly. IDE's will consist of primarily Xcode and VScode	Linux primarily uses OOP languages like C, C#, and C++, as well as the common language Python. You will also see basic web development languages – JavaScript, HTML, & CSS – alongside Node.js, React and React Native. Linux also utilizes MySQL. As far as IDE's go, you will commonly see Visual Studio, Atom, and git.	Windows uses the most variety of languages, but most software is commonly built with C, C++, Java, .NET, as well as basic web-development languages – JavaScript, HTML, CSS, & Python, alongside many different frameworks. The go-to IDE will be git, and/or Visual Studio and VSCode	Most mobile devices utilizes different languages and IDE's depending on the type of OS that they are functioning with. IOS will primarily use swift and Objective-C, while Kotlin or Android will primarily use Java. This is the same with IDE's. Android and Kotlin you will see Android Studio, Visual Studio, and Eclipse being used very often while for IOS you will see primarily all Xcode

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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

6. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>