<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	<3/23/25>	<Jose Alonso>	<Web-based off the tv show>

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

< This document explains the plan for creating a web-based game called "Draw It or Lose It" for our client The Gaming Room. The game is based on the TV show "Win, Lose, or Draw," where teams try to guess what is being drawn. The idea is to use a library of stock drawings as clues, and multiple teams will play through four rounds. Making the game web-based will let more people play on different devices and improve the overall experience. The design aims to meet the client's needs and provide a fun, easy-to-use game.

>

Requirements

< The game must be accessible through a web-based platform making sure that it is compatible with every operating system.
The game should support the participation of more than one team at a time with multiple players on each team
Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.>

Design Constraints

- <**Web-Based Game**: The game needs to be built to run on the web, which comes with challenges like making sure it works properly across different browsers, devices, and ensuring the game is secure and communicates well over the internet.
- **Unique Names**: The system must make sure that every game, team, and player has a unique name to avoid confusion and make the game experience smooth when players create or join games.
- **One Game at a Time**: Only one version of the game can be running at any moment, so the design should make sure that the game doesn’t run more than once at the same time.

>

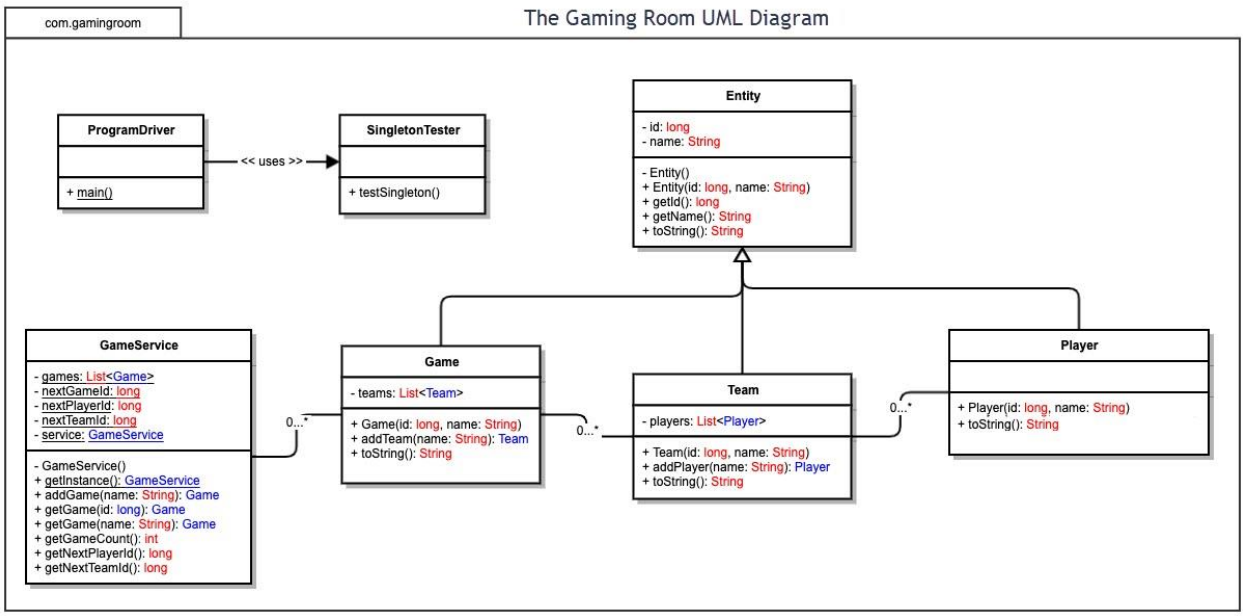
System Architecture View

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 domain model diagram shows how the different components of the game system are structured and interact with each other. The Entity class acts as the foundation for all other game related elements, such as Game, Team, and Player, by providing common attributes like "id" and "name" that every entity

shares. The Game class is responsible for managing multiple Teams, while each Team contains several Players, establishing a clear hierarchy within the system. The GameService class is central to the application, overseeing the creation and management of game instances. It ensures that multiple games can be handled effectively by holding references to them. The ProgramDriver class is the entry point of the application, responsible for initiating the system, creating a single instance of GameService, and facilitating the addition of games, teams, and players. Also, the ProgramDriver class depends on the SingletonTester class to verify the correct implementation of the Singleton pattern. The system leverages core object-oriented programming principles, including inheritance, which allows subclasses like Game, Team, and Player to inherit properties from the Entity class, reducing redundancy in code. Encapsulation is used in the GameService class to protect its data by only exposing methods to interact with it, preventing external access to its inner workings. Finally, abstraction simplifies the interface by hiding complex details, allowing users to focus on creating and managing games, teams, and players without worrying about the underlying processes. This design structure ensures the system is both efficient and easy to maintain.



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	< It's Unix-based, which means it's stable and secure for hosting websites. It's good for developers because it has lots of tools, but it can be more expensive when it comes to hardware. It also doesn't scale as easily as Linux or Windows. >	< It's open source, so you can customize it and use a lot of software and tools. It's great for scaling up and is known for being stable and secure. However, it has some limits with its graphical interface and might not work well with all hardware. >	< It works with lots of software and has a strong community for developers. It supports a wide range of hardware and has good documentation. However, it does have more known security risks.>	< It's portable, meaning it can be used on different devices. However, it has limited screen space and mostly relies on touch and gestures. Also, the hardware can vary in capabilities. >
Client Side	< It's easy to use, so you don't have to spend much time learning how to use it. But, building and maintaining different versions for different clients can get expensive, take more time, and need different skills.>	< It's free to use and share, but you still need to think about other costs like hardware and tools. It can be harder to learn, and you'll need different skills for different clients.>	< The licensing fees can be more expensive than free, open-source options.>	< When developing apps, you need to think about making them work well on different screen sizes and dealing with possible connectivity issues. Also, consider using native features like the camera, GPS, and push notifications.>
Development Tools	< Node.js and JavaScript are commonly used. IDEs such VSCode and XCode. >	<It has a lot of tools, like VSCode, Atom, and Sublime Text, for developers. Plus, it has a strong command-line system and package managers to easily install and manage software>	<C# and the .NET framework are commonly used for building web apps on Windows. Popular tools for coding in these languages include Visual Studio and JetBrains IDEs>	< In addition to Java and JavaScript, Kotlin, Swift, and Objective-C are popular for app development. Android Studio and XCode are the main tools used, along with emulators and simulators to test the apps>

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**: <To expand the game Draw It or Lose It to more devices, it's best to develop a web-based version. By building the game using modern web technologies such as HTML5, CSS, and JavaScript, players will be able to access and enjoy it on desktops, laptops, tablets, and smartphones without the need to download an app. This approach ensures a consistent user experience across different screen sizes and operating systems. Additionally, a web-based platform allows for quicker updates, easier sharing via links, and broader accessibility, making it an ideal choice for reaching a wider audience and growing the player base. >

2. **Operating Systems Architectures**: <The game will use a combination of a relational database and cloud storage to manage data efficiently. The relational database will store structured information such as player profiles, game progress, achievements, leaderboards, and gameplay statistics. This setup allows for quick querying, secure transactions, and data consistency, which are crucial for delivering a smooth gaming experience. Meanwhile, cloud storage will be utilized for managing large or unstructured media assets like images, audio files, and potentially video clips. By leveraging cloud storage, the game can efficiently handle high volumes of content, enabling players to access their data across multiple devices. This architecture supports scalability and high availability, ensuring the game can grow with its user base while maintaining fast load times and seamless synchronization.
3. >

4. **Storage Management**: <The game will use a combination of a relational database to store player data, game progress, and statistics, and cloud storage to handle media files like images. Cloud storage ensures the game can scale and be accessed across different devices.>

5. **Memory Management**: <Windows gives you a bunch of ways to store files and manage memory. For example, you can use things like Azure Storage, OneDrive, or even tools like Visual Studio to save and keep track of different versions of your work. When a program runs, Windows uses both physical memory and virtual memory to help it run smoothly. Newer versions of Windows, like Windows 10, are better at handling memory, making sure programs have enough to use without slowing down. Using good habits while building your program like picking the right kind of storage and planning for how memory is used will make everything run better. Some storage types can't grow, while others can, so it's smart to choose the right one.>

6. **Distributed Systems and Networks**: <To allow players to interact across devices, the game will use a distributed system with a central server or cloud setup. The server will keep the game in sync, send real time updates, and handle messages between players. It will also handle network problems like poor connections by having systems in place to deal with issues like delays or outages.>

7. **Security**: <Since there are always new ways people try to steal data, it’s important to take security seriously. One tool that helps protect devices like PCs, Macs, phones, and tablets is called Aura. It does cost money, but it adds an extra layer of protection, which is better than just

relying on the basic security that comes with your computer or phone. Aura also has U.S. based support, which is a big plus if something goes wrong. It's also important to teach employees how to use strong passwords and set up systems where people only get access to the data they need, not just giving everyone full access.>