Win, Lose, or Draw

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	02/04/24	Briana Long	Executive summary, Design Constraints, DOmain Model

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 client, The Gaming Room wants to develop a web-based game that serves multiple platforms for their game, Draw It or Lose It. The game is inspired by the game television game show Win, Lose, Draw. Which involves teams guessing phrases and titles based on drawings. The game will need to be scalable for a large player base as well as holding data for the games themselves and the drawings. The game will also need a user-friendly interface that is cross compatible between operating systems. The applications will be able to support two teams of multiple players per match. The name of the game and the teams will be unique. The system will check the names in order to achieve different identifiers. This will ensure a single instance to secure data integrity. The game will be round based with 30 seconds to reveal the drawings. There will need to be data on the drawings to keep them rendered.

Requirements

Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client's business and technical requirements in a clear and concise manner.*

Design Constraints

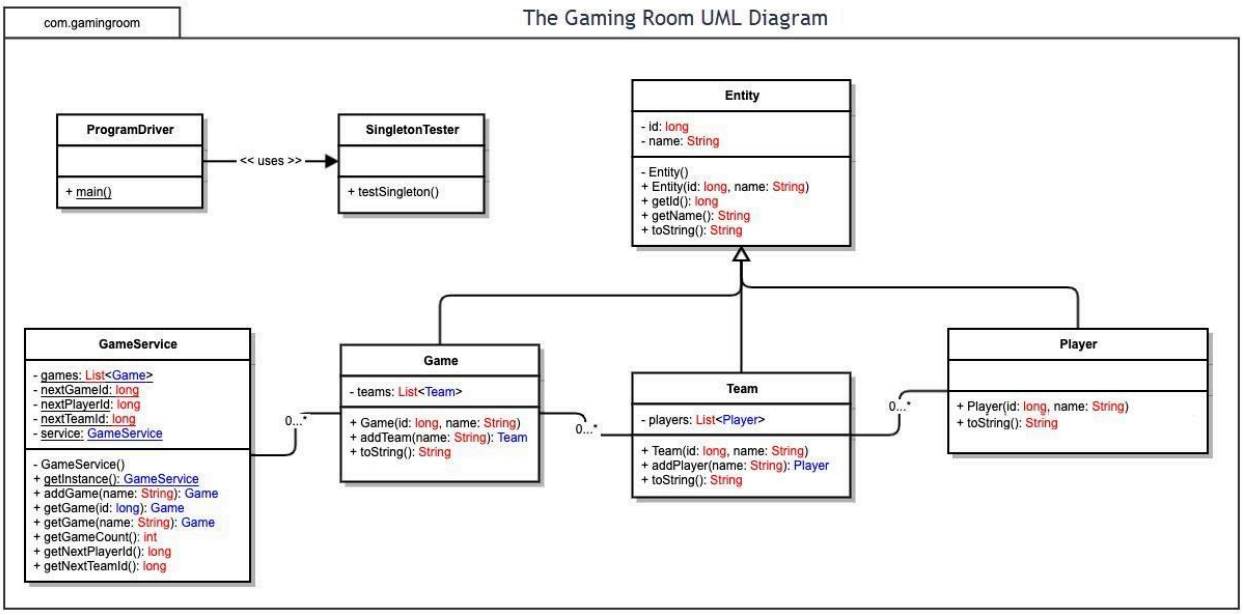
Using a web-based game application can allow for scalability of players and game instances at the same time. This would require good resource management to be necessary in the code. There will need to be security implemented into the application to all the integrity of data, input authorization and authentication for it's users. This would prevent data breeches and any other attacks. While developing a web based application resource constraints will also have to be included to optimize the program. As the game will need to be able to process and run on varying devices interfaces and operating systems.

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 program driver uses the SingletonTester. The Entity class is associated with the Game, Team, and Player classes. The GameService class contains the attributes such as the game list and ID, as well as player and team ID, and GameService. It also contains the methods of the GameService, player count, and team and game IDs. The Game service class is associated with the Game class. The Game class contains the team names. This class is associated with the Team class. The team class contains the player list of IDs. The team class is associated with the Player class, which contains their user names. These classes fit into the Enity 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	Developer friendly and is good for small to medium-sized applications. There are limitations to scalability and tend to have a greater cost.	Cost-effective with scalability. Factors of hardware compatibility, user interface preferences, and software requirements should be considered.	Good choice for web-based applications, user user-friendly, and works well with all Microsoft products. Factors to be considered are licensing cost and resource consumption.	Portability and wide audience. Factors to consider are limited resources and security concerns.
Client Side	Development tools require specific expertise of developers with knowledge of Apple guidelines. Licensing fees are higher. Mac OS has a unique interface. Cross-application requires more work with other teams.	Devs must decide on what version to use based on compatibility and user interface. Devs must know a variety of languages and tools which can be more expensive. Requires more documentation than other OS.	Licensing costs for development tools must be factored in. Must be tested across various Windows versions. Devs need to know various Windows languages and tools. Must follow security practices.	Increased development costs for multiple mobile platforms. Devices vary in UI, size, resolution, and hardware. Devs need to know platform-specific languages and frameworks.
Development Tools	Languages: Swift IDE: Xcode, Visual Studio Code Version Control: GIT	Languages: C, C++, Python, Go, Rust IDE: VSC, Eclipse, IntelliJ IDEA Version control: Git	Languages: C#, C++, JavaScript IDE: Visual Studio, Eclipse Version Control: GIT	Languages: Swift, Java IDE: XCode, Android Studio Version Control: Git

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**:
   A cloud service such as Microsoft Azure would be an optimal platform.

2. **Operating Systems Architectures**: It can run on both Linux and Windows OS. Cloud services allow for the best option of scalability as an on demand resource.

3. **Storage Management**: With the storage management Azure storage can be used. This is important as it is a scalable resource. Azure database can be used as well for management.

4. **Memory Management**: Memory management is important for the game as it can allow data to be stored and retirevied quickly. Being able to cache important and immediately needed data that is frequently used.

5. **Distributed Systems and Networks**: Azure Logic Apps can be used for workflows that can be used with other platforms and services to allow integration. Being able to securly use the cloud services can be done with Azure as well. Connectivity outages can be prevented due to outsourcing to the cloud service. There are many servers used to prevent a full outage of the game service.

6. **Security**: For security an end to end encryption can be implemented to protect data such as user information. When using cloud platform services access management is often included for security of the product. This prevent hacking and other malicious attempts. Regular maintenance of the platform and software are needed to kep up against new threats and bugs.