Draw it or Lose It

CS 230 Project Software Design Template

Version 1.2

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	11/15/2024	Nicholas Justus	Added content for all sections, including requirements, constraints, evaluation, and recommendations. Incorporated design patterns as per project requirements.

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 requested a software design solution for transforming their game, *Draw It or Lose It*, into a web-based application that supports multiple platforms. The solution involves implementing design patterns such as Singleton for ensuring only one instance of the game service exists at any time and Iterator for validating unique names for games, teams, and players. This document outlines the proposed design, constraints, evaluation of operating systems, and recommendations for building a scalable, efficient, and maintainable architecture.

Requirements

The application must:

- *Ensure only one instance of the game service exists at any time (Singleton pattern).*
- *Allow multiple teams and players to be associated with a game.*
- *Ensure game, team, and player names are unique (Iterator pattern).*
- *Support deployment across multiple platforms, including Linux, Windows, macOS, and mobile devices.*
- *Provide a secure and scalable environment for hosting and operating the game.*

Design Constraints

The design constraints include:

1. **Singleton Pattern**: Ensures that only one instance of the game service exists, reducing memory overhead and preventing conflicts.
2. **Unique Names**: Names for games, teams, and players must be unique, necessitating validation logic and efficient searching (Iterator pattern).
3. **Distributed Environment**: The application must support multiple platforms, necessitating a scalable and platform-agnostic architecture.
4. **Resource Optimization**: The design should minimize resource usage and maintain high performance across 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.

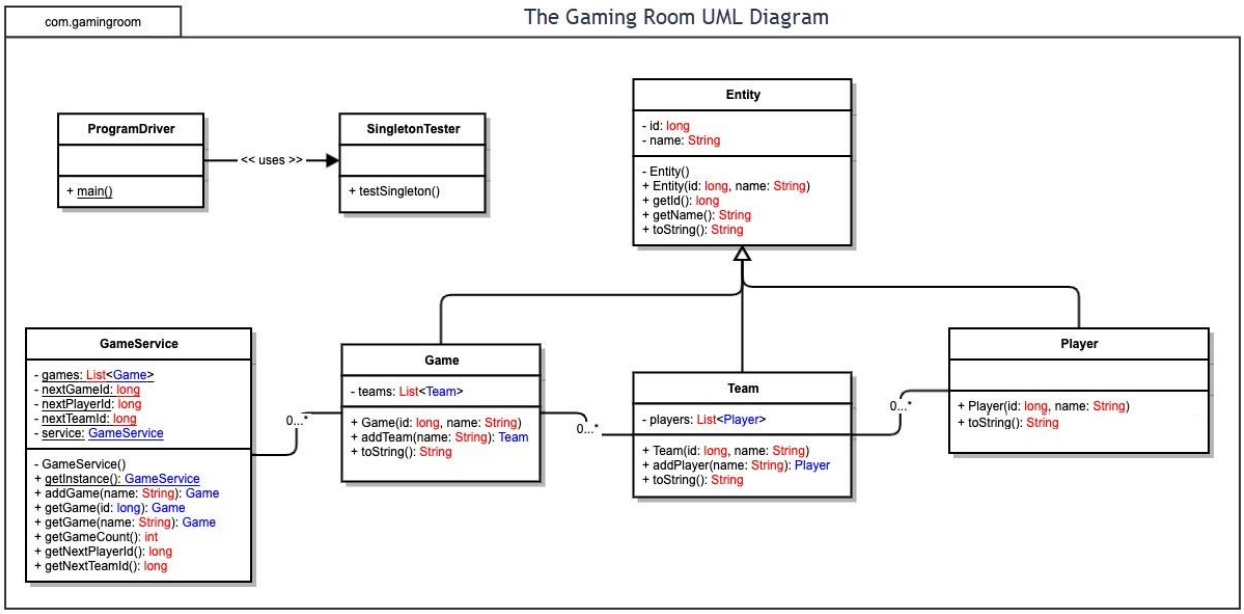
Domain Model

The UML diagram illustrates the relationships between entities:

- **Entity**: A base class containing common attributes (id and name) shared by Game, Team, and Player.

- **GameService**: Implements the Singleton pattern to manage all games and ensure a single instance is active.
- **Game, Team, and Player**: Inherit from Entity, demonstrating inheritance. Each game can have multiple teams, and each team can have multiple players.
- **Iterator Pattern**: Used in GameService for validating unique names when adding games, teams, and players.

These object-oriented principles ensure code reuse, maintainability, and efficient memory management.



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.

Platform	Server Side	Client Side	Development Tools
Linux	Linux is a widely used platform for web-based applications due to its robust server-side capabilities. It supports various web servers such as Apache and Nginx and offers cost-effective solutions as it is open-source with no licensing costs. Linux servers are highly scalable and reliable, making them suitable for hosting applications that require high availability.	Linux can support the client-side application through modern web browsers like Firefox and Chrome. However, the desktop user base for Linux is smaller compared to other platforms, which may limit its reach.	Linux provides extensive development tools, including GCC for C++, Python, and Java compilers. IDEs like Eclipse and VS Code are freely available. There are no licensing costs for these tools, but the development team may require Linux expertise.
Mac	**MacOS supports server-side deployment through tools like macOS Server, though it is less common for hosting large-scale web applications. Licensing costs may be higher as macOS hardware is required.**	**MacOS supports modern web browsers like Safari, Chrome, and Firefox, ensuring compatibility for the client-side application. Its strong presence in creative industries makes it a valuable platform.**	**MacOS provides Xcode for development, supporting Swift, Objective-C, and C++. While Xcode is free, Mac hardware requirements can increase costs. Additional tools like Homebrew enhance development capabilities.**
Windows	**Windows servers are a popular choice for web hosting, especially with enterprise clients. Tools like IIS (Internet Information Services) enable seamless hosting of web applications. However, Windows Server licensing costs can be significant.**	**Windows dominates the desktop market, supporting the client-side application on browsers like Edge, Chrome, and Firefox. Its wide adoption ensures high reach among users.**	**Windows supports development through tools like Visual Studio, which offers extensive libraries for .NET and C++ development. Licensing costs may apply for enterprise editions of Visual Studio.**
Mobile	**Mobile platforms themselves do not serve as servers but can interface seamlessly with server-side applications hosted on Linux, Windows, or Mac servers.**	**Mobile platforms, including Android and iOS, dominate user engagement. Cross-platform development tools like Flutter or React Native can ensure compatibility and performance on both platforms.**	**Mobile development requires tools like Android Studio for Android and Xcode for iOS. These tools are free, but the team must invest in specialized expertise for each platform.**

Recommendations (Project 2):

Based on the evaluation, **Linux emerges as the optimal platform for server-side deployment** due to its unparalleled cost-effectiveness, scalability, security, and extensive adoption within the web hosting industry. Its flexibility and robust support for high-traffic applications make it the most practical choice for The Gaming Room's game application expansion. While Linux minimizes licensing costs, it also offers reliable tools and frameworks to facilitate efficient development and deployment.

For the **desktop client-side application**, **Windows should be prioritized** due to its dominant market share and widespread user base, ensuring maximum reach and engagement. **Mac** also represents a valuable client platform, particularly given its strong presence in creative industries and its compatibility with modern web technologies. The **mobile platforms, Android and iOS, are indispensable** for achieving the widest audience reach and enhancing user engagement. These platforms dominate the mobile gaming market and should be a central focus for development. To reduce development complexity and maintain consistency, the client should leverage **cross-platform development tools** such as **React Native** or **Flutter**, which allow for seamless compatibility across mobile and desktop environments.

The development team will need to acquire specialized expertise in mobile and desktop platforms to ensure a modern, responsive application design. Additionally, it is important to plan for **potential licensing costs** associated with tools like **Visual Studio for Windows development** or **Xcode for macOS development**. These costs should be factored into the budget, alongside investments in training and resources for the development team.

By adopting this strategic combination of server-side, desktop, and mobile technologies, The Gaming Room can ensure a scalable, cost-efficient, and highly engaging gaming application that meets the needs of its expanding user base across multiple platforms.

Recommendations (Project 3):

<u>Operating Platform</u>

For the server-side platform, Linux is the best choice for hosting the game, *Draw It or Lose It*. Linux offers cost-effectiveness, high scalability, and robust security, making it ideal for web-based applications. It supports various web servers like Apache and Nginx and can handle the demands of a distributed, multiplayer gaming environment.

<u>Operating Systems Architectures</u>

The Linux architecture uses a modular structure, which makes it highly efficient and reliable. Its kernel supports multitasking and multiple users simultaneously, which is essential for a gaming platform that needs to manage multiple games and players at once.

The platform also provides strong support for networking and inter-process communication, which aligns with the needs of the distributed system.

Storage Management

The Ext4 file system is recommended for Linux servers. It offers efficient storage management with journaling capabilities to prevent data loss in case of power failures. Ext4 is compatible with large file sizes and directories, ensuring that data related to the game, such as player statistics and game states, is handled reliably.

Memory Management

Linux uses advanced memory management techniques like paging and swapping to efficiently manage RAM usage. It ensures that frequently accessed data is kept in memory while less critical information is swapped to disk. This helps optimize performance during high-traffic periods when many players are using the game simultaneously.

Distributed Systems and Networks

To enable communication between platforms, the game should be implemented as a distributed system using APIs to connect the client applications to the server. Linux's strong networking capabilities, including support for TCP/IP protocols, will help maintain reliable connectivity. Tools like load balancers can manage traffic efficiently, ensuring the game runs smoothly even during outages or high demand.

Security

To secure user data across platforms, the Linux server should implement SSL/TLS encryption for all data transfers. This ensures that sensitive information like player credentials and game data is protected from interception. Additionally, the use of firewalls and regular security updates will help safeguard the server. Authentication mechanisms, such as OAuth, should be used to manage user access securely.