"The Gaming Room"

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	01/24/2025	Henoc Mudibu	This revision refines the document structure, adds clear sections for the Executive Summary, Design Constraints, and Domain Model, and aligns the content with client requirements for transitioning to a web-based platform.
1.1	02/09/2025	Henoc Mudibu	This revision refines the document, enhances the evaluation section with platform comparisons, and aligns recommendations with client requirements for expanding to a web-based platform.
1.1.1	02/21/2025	Henoc Mudibu	This revision finalizes the recommendations section by integrating platform-specific architecture details, storage and memory management techniques, distributed systems considerations, and security enhancements. It ensures alignment with client needs for cross-platform expansion and secure, scalable deployment.

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 wants to take "Draw It or Lose It" from being just an Android app to a full-fledged web-based game. The goal is to make the game accessible across multiple platforms while keeping things smooth and engaging for players. To make this work, we need to ensure multiplayer functionality, unique team and game names, and that only one instance of the game runs at a time. This document lays out the technical plan to make that happen.

To achieve this, we'll be using smart design choices like the Singleton pattern to manage game instances and the Iterator pattern to handle game entities efficiently. Our approach will focus on keeping the system easy to maintain, and secure while delivering great performance across different devices.

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

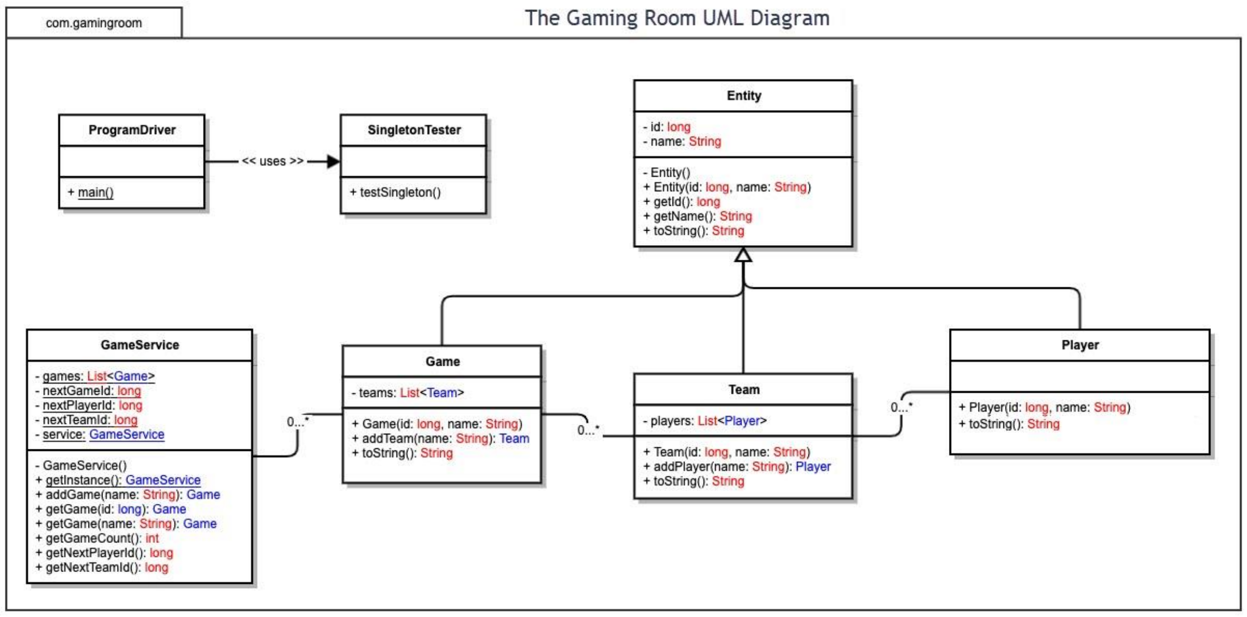
Since "Draw It or Lose It" will be web-based, we have to make sure it works well on all major platforms Windows, macOS, Linux, and mobile devices. The user experience needs to be seamless, meaning the design has to be responsive and adaptable. A big challenge is managing game state efficiently so that only one instance exists at a time, which we'll handle using the Singleton pattern.
Performance is another key factor, as multiple players will be interacting simultaneously. The system should be optimized to handle these interactions without slowing down. Security is also a top priority, ensuring safe user authentication and data protection.

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 outlines the major components of the system and how they interact. At the core, we have the Entity class, which acts as a base for other entities like Game, Team, and Player, ensuring consistency and reducing code duplication. The GameService class follows the Singleton pattern to ensure there's only one instance managing the game lifecycle. To efficiently loop through game entities, we use the Iterator pattern. These object-oriented design choices inheritance, encapsulation, and polymorphism help keep the code organized, maintainable, and flexible.



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	Mac is Unix-based, which means it's stable and secure for hosting web apps. It supports Apache, Nginx, and Docker, so deployment isn't an issue. However, the downside is that Mac servers aren't as common in enterprise environments, and Apple's hardware is expensive.	Linux is the go-to choice for web hosting. It's free, open-source, and powers most of the internet. It works well with Apache, Nginx, and Docker, making it super flexible. The only catch is that it requires more technical knowledge to set up and maintain.	Windows Server integrates well with Microsoft tools like .NET, which is great for teams already using Microsoft's ecosystem. However, licensing costs can add up, and it's generally considered less secure than Linux unless properly configured.	Mobile devices don't host web apps directly, but cloud services like AWS, Firebase, and Azure make it easy to deploy scalable backends. The main challenge is ensuring smooth performance across different devices and network conditions.
Client Side	Developing for Mac is straightforward since it supports all major browsers like Safari, Chrome, and Firefox. However, Apple has strict guidelines, and building Mac apps requires knowledge of Swift or cross-platform frameworks like React.	Linux users mainly rely on web browsers like Chrome and Firefox, so as long as the app is browser-friendly, it'll work. The challenge is that different Linux distributions can behave slightly differently, which means extra testing.	Windows supports most browsers, and web apps generally run well on it. If we want a Windows specific app, using Electron would be a solid choice. The main challenge is making sure the app works across different Windows versions.	Mobile development means ensuring the app looks and works great on both iOS and Android. A responsive design is key, and using frameworks like React or Flutter can help speed up development. The tricky part is keeping a consistent experience across different screen sizes and OS versions.
Development Tools	Development for Mac typically involves Swift, Objective-C, and JavaScript (React, Next.js). Xcode is the go-to IDE, and while it's free, you need a Mac to use it, which isn't cheap.	Linux development is very open-ended Python, JavaScript, Java, Node.js, and Docker are all common choices. Most tools are free and open source.	Windows development often involves .NET (C#), JavaScript (React, Next.js) for web hosting. Visual Studio is the preferred IDE, but the paid versions can be pricey.	For mobile, React, Flutter, Swift (iOS), and Kotlin (Android) are popular options. Apple requires a $99/year developer account, while Google's Play Store only has a one-time $25 fee.

<u>Recommendations</u>

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 ensure scalability, security, and cost-effectiveness, Linux is the recommended server operating platform for *Draw It or Lose It*. Linux is widely used in web hosting due to its stability, open-source nature, and strong security features. It supports various web technologies such as Apache, Nginx, Docker, and Kubernetes, making deployment efficient. Additionally, Linux servers offer high performance and flexibility, allowing seamless integration with cloud services like AWS, Google Cloud, and Azure. Since Linux is open source, it also helps reduce licensing costs compared to proprietary solutions like Windows Server.

2. **Operating Systems Architectures**: The Linux-based client-server architecture follows a RESTful API model, where the game's front-end interacts with the server using standard HTTP methods (GET, POST, PUT, DELETE). This stateless architecture ensures quick processing of authentication, game state management, and real-time updates. A microservices approach can further enhance performance by modularizing different game components, such as authentication, game logic, and media storage. Additionally, containerization will be used to isolate services, improving reliability and scalability across different environments.

3. **Storage Management**: With 200 high-resolution images requiring approximately 1.6GB of storage, efficient storage management is crucial. The recommended approach is to store media assets in a cloud storage solution such as Amazon S3. A Content Delivery Network (CDN) will be used to ensure faster access to images globally, reducing latency for players. Additionally, compression techniques will be applied to minimize file sizes without reducing quality, while a structured database (such as MongoDB) will manage game metadata, leaderboards, and user data.

4. **Memory Management**: To maintain smooth gameplay, effective memory management techniques must be applied. Memory caching will store frequently accessed game data to reduce load times. Texture compression and mipmapping will optimize image rendering, lowering the RAM footprint while maintaining high visual quality. Additionally, adaptive quality scaling will adjust image resolution dynamically based on the device's hardware and memory availability. These optimizations will ensure the game runs efficiently across platforms, preventing performance issues like lag or crashes.

5. **Distributed Systems and Networks**: Since *Draw It or Lose It* needs to support cross-platform multiplayer functionality, a distributed system architecture is necessary. The game clients will communicate with a centralized Linux-based backend using a WebSocket-based real-time communication system, ensuring low-latency interactions for multiplayer gameplay. Cloud services like AWS Lambda will be used to manage real-time synchronization. A load balancer will distribute traffic across multiple server instances to prevent bottlenecks, while automatic failover mechanisms will ensure high availability even during network outages.

6. **Security**: Security is a top priority for The Gaming Room. To protect user data, the application will implement role-based authentication and OAuth for secure user login. End-to-end encryption will safeguard communication between clients and servers. On the backend, security measures such as firewalls, intrusion detection systems (IDS), and access control lists (ACLs) will be implemented to mitigate cyber threats. Additionally, regular security audits and penetration testing will ensure compliance with best practices, protecting user information across all platforms.