Draw It or Lose It Web-Based Application

CS 230 Project Software Design

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	07/19/25	Shadab Chowdhury	Initial completion for project submission based on client requirements and 2025 best practices.

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 seeks to evolve their Android-only game, Draw It or Lose It—a multiplayer drawing-guessing game with timed rounds—into a web-based application compatible across multiple platforms, including Windows, macOS, Linux, and mobile devices. The design problem centers on managing unique game instances, teams, and players in a distributed environment while ensuring efficiency, scalability, and security. Our solution employs Java for the backend with design patterns like Singleton for single-instance management and Iterator for collections, paired with web technologies such as HTML5 and WebSockets for real-time interactions. Critical information for proceeding: Prioritize Linux for server hosting due to its stability and cost-effectiveness; integrate a database like MySQL for persistence; and conduct thorough cross-browser testing to handle platform variances. This approach will facilitate seamless expansion, minimize development costs, and enhance user engagement across devices.

Requirements

The client's requirements include: Enabling games with one or more teams, each supporting multiple players; Ensuring unique names for games and teams to avoid conflicts during user selection; Restricting the game service to a single instance in memory via unique identifiers; Supporting cross-platform access through web browsers on diverse OS; Implementing timed gameplay mechanics (four 1-minute rounds with 30-second reveals and 15-second steals); Securing user data in distributed sessions; Scaling for concurrent multi-user sessions; and Integrating stock drawing libraries for puzzles.

Design Constraints

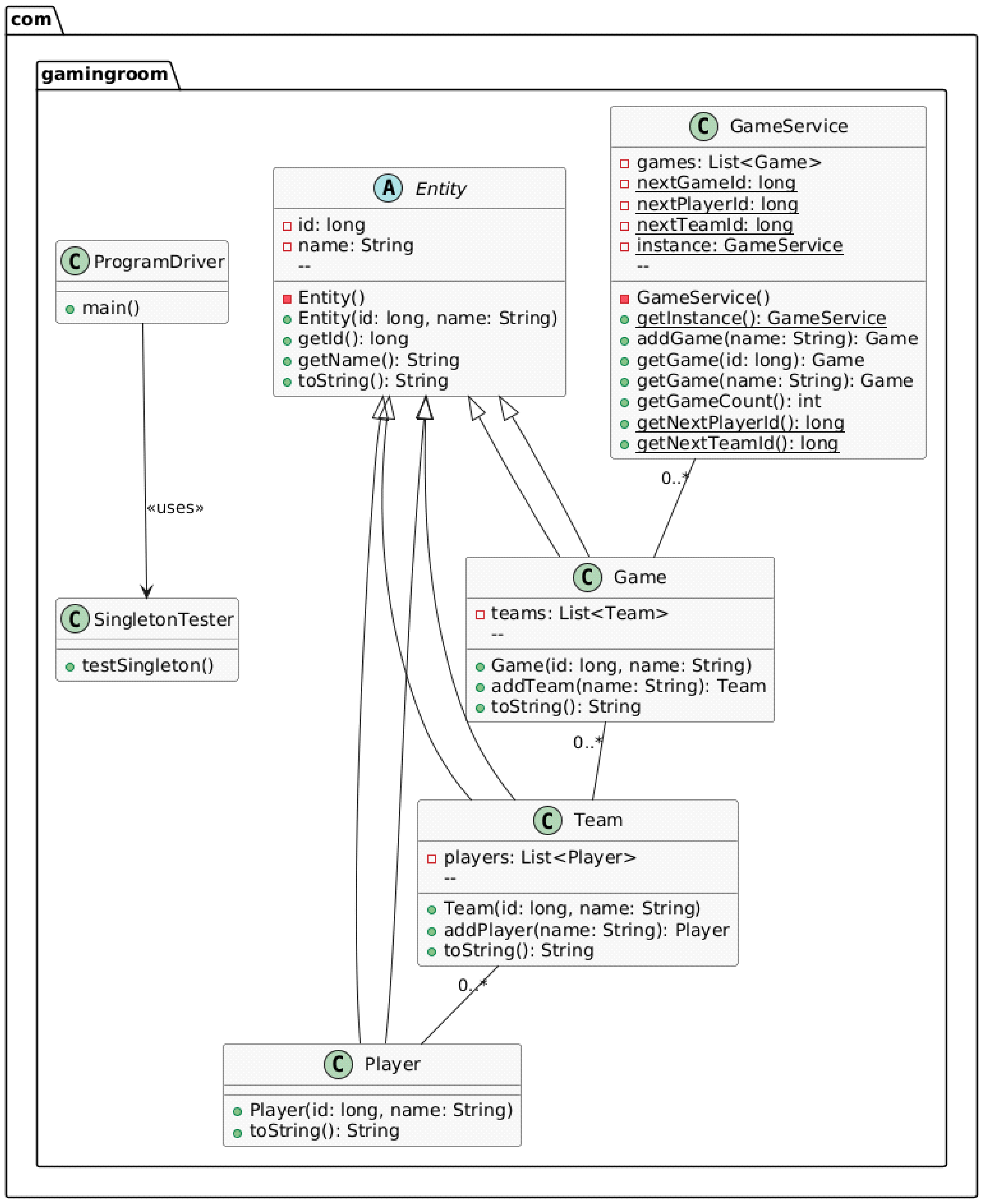
In a web-based distributed environment, key constraints include cross-platform browser compatibility, mandating use of standards like HTML5 and JavaScript to ensure consistent rendering across devices, but this implies extensive testing for inconsistencies (e.g., Safari vs. Chrome), potentially delaying deployment.pixemweb.com Network latency for real-time features like drawing updates requires low-delay protocols (e.g., <50ms via WebSockets), impacting scalability and necessitating cloud infrastructure, which raises costs and complexity in handling distributed loads.deftsoft.com Memory limits for single-instance services constrain in-memory storage, requiring efficient patterns like Singleton and optimized data structures, limiting concurrent games without external databases.lwn.net Security mandates encryption and validation in distributed setups, adding development overhead but crucial for compliance, while unique naming enforces validation checks that could bottleneck performance in high-traffic scenarios. These constraints promote a modular architecture but extend timelines for integration and optimization.

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

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.



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 offers robust security and Unix-based stability for web hosting, with advantages in ecosystem integration for Apple hardware, but weaknesses include high costs and limited data center adoption, making it less scalable for high-traffic games.	Linux provides free, customizable stability and efficiency for servers, excelling in scalability for web games, though it requires more expertise and has potential tool compatibility issues.	Windows integrates well with Microsoft tools and offers ease of use, but is resource-heavy, more vulnerable to attacks, and incurs licensing costs, limiting its appeal for cost-sensitive hosting.	Mobile devices lack server capabilities due to resource constraints and intermittency, with advantages in edge computing but weaknesses in power and security for hosting.
Client Side	Mac client support involves higher costs for Safari testing and expertise in Apple APIs, but moderate time due to consistent hardware, enabling smooth development for premium users.	Mac client support involves higher costs for Safari testing and expertise in Apple APIs, but moderate time due to consistent hardware, enabling smooth development for premium users.	Windows requires moderate cost/time with expertise in .NET, benefiting from widespread adoption.	Mobile needs high expertise for responsive design, varying costs for iOS/Android, and time for device optimization.
Development Tools	Mac deploys with Swift/JavaScript, using Xcode, VS Code, and Unity for cross-platform games.	Linux uses C++/Python with Godot, Unreal, VS Code for efficient development.	Windows employs C#/.NET via Visual Studio, Unity.	Mobile leverages HTML5/React Native with Android Studio, Xcode.

<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:

- **Operating Platform**: Recommend Linux for server-side with cross-platform tools like Unity, as it offers cost-free scalability and stability, enabling expansion to Windows/macOS/mobile via web deployment.

- **Operating Systems Architectures:** Linux features a monolithic kernel with modular drivers, supporting containerization (Docker) for microservices, efficient for distributed game loads with low overhead.

- **Storage Management**: Employ LVM with ext4 for dynamic volume handling and snapshots, ideal for game data redundancy.

- **Memory Management**: Linux utilizes paging/swapping with page tables and OOM killer, optimizing for Draw It or Lose It by overcommitting RAM for multiple sessions.

- **Distributed Systems and Networks**: Use REST/WebSockets for cross-platform comms, with cloud load balancers; dependencies like connectivity are mitigated via retries and offline modes.

- **Security**: Implement SELinux, firewalls, and MFA; encrypt data in transit, leveraging Linux's robust access controls for user protection across platforms.