Draw It or Lose It

CS 230 Project Software Design Template

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	01/22/2025	Elena Van	Initial version of the software design document with the executive summary, design constraints, domain model, platform evaluation, and recommendations for The Gaming Room client.

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 expand their game Draw it or lose it which is an Android app and wants to turn it into a web-based game that works for multiple platforms. This document talks about how to design the game to meet their needs. The solution includes using software patterns like Singleton to make sure only one instance of the game exists and an iterator to make sure that it has a unique name for games, teams, and platters. These steps will help make the game scalable, secure, and able to work across

Requirements

Business Requirements:

1. The game must support multiple teams, with each team consisting of multiple players.
2. Unique names must be used for games, teams, and players to avoid confusion.
3. The game must be accessible across different platforms to expand the audience beyond Android users.

Technical Requirements:

1. Ensure that only one instance of the game exists in memory at any given time.
2. Implement a system to check for duplicate names for games, teams, and players.
3. Implement design patterns, such as the Singleton Pattern to ensure only one instance of the game is active and the Iterator Pattern to efficiently navigate collections, supporting an effective software design.

Design Constraints

Scalability
The system must support many users who access the game from different platforms. This requires optimized resource management and a cloud-based system to handle increased leads without having any performance issues.

Data Consistency
To maintain unique names for games, teams, and players across the different platforms, mechanisms must be implemented to have the data synchronized updates and avoid problems.

Single Instance
The requirement is that only one instance of the game exists in memory and will be addressed by implementing the Singleton Pattern. This would ensure that the game is managed from a single point, avoiding duplicate or problems.

Security
Data shared between users and the system must be protected from unauthorized access. This will be achieved by using encryption and secure login methods to keep the user's information safe.

Cross-Platform

The application needs to work on both web and mobile platforms. To ensure this, tools like Java, HTML5, and JavaScript will be used to create an easy-to-use experience.

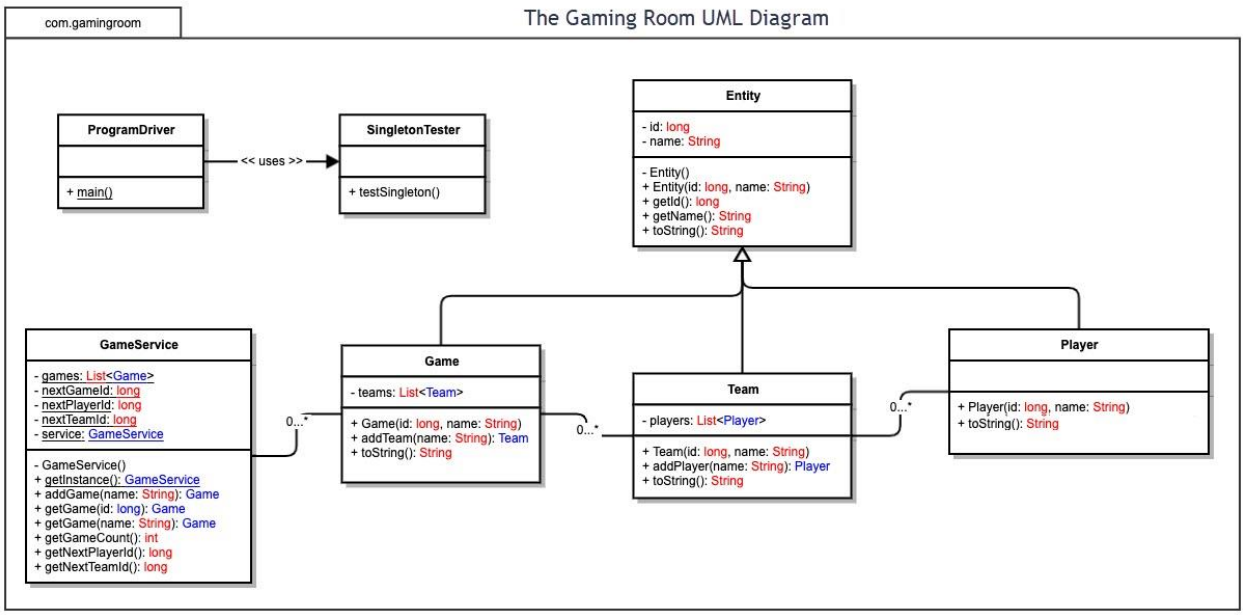
Latency and Real-Time Communcation
The game's real-time features will include timed rounds and interactions between teams, introduce challenges related to latency. Efficient communication protocols, such as RESTful APIs or WebSockets, are necessary to minimize delays and ensure seamless gameplay.

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 class diagram that is provided demonstrates the structure and relationships between classes in the gaming application, using object-oriented programming principles for an efficient design. The Entity class serves as the base class, providing common attributes (id and name) and methods (getId(), getName(), toString()), which are inherited by Game, Team, and Player, reducing redundancy and making sure that it is consistent. The GameService class implements the Singleton Pattern to ensure only one instance exists, making the management of games condense, teams, and players. It contains a List<Game> to manage many game instances, while each Game contains a List<Team>, and each Team contains a List<Player>, this creates a logical hierarchy. Encapsulation is done by keeping attributes private and using public getters for controlled access, while polymorphism allows methods like toString() to be customized in derived classes. The ProgramDriver and SingletonTester classes interact with GameService to initiate and validate its behavior. This design ensures that there will be scalability, reusability, and maintenance. While meeting requirements such as unique identifiers for games, teams, and players, and concise management in a distributed environment.



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 must work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

Software Design – Evaluation

Server-Side

Platforms	Characteristics	Advantages	Weaknesses
Linux	It is open source, free to use, and the source code is open and available online for many people to use (open source). Many people are also able to access the system resources at the same time and will allow more than one	It is free of cost to use, flexible, works on command line but has a GUI and can be converted to be used like Windows, allows end-to-end encryption, and it is available in all languages making it	Requires knowledge prior to using it (learning curve) not as easy as using Windows and software and hardware compatibility.

	application to run at the same time.	easier for many people to use.	
Mac	Fewer viruses and security issues which make it more reliable and stable to use, user interface is simplified and does not lose its features, and users can multitask.	Performance is known to be good, user interface is simple and clean, security features are good, helps protect users' data and privacy, and comes with several free and preinstalled applications (includes Pages, Numbers, and Keynote).	The cost of using MAC is expensive compared to Windows, compatibility issues, limited number of apps in the app store or other sources (app support), and few customization options.
Windows	Efficient speed, interface, program execution, and memory management.	It is user friendly and easy to use, compatible with all hardware, has a large range of software to use, and allows devices to connect to a computer and be used instantly (plug and play features).	Software must be paid for if you want to do something specific like graphic designing, expensive (license from Microsoft), and specialized help is not the best.
Mobile Platform	User-friendly interface, faster loading times, push notifications,	Work great for users to access and use the application.	Limits the use of storage and power (not practical to use for making applications)

Client-Side

Platforms	Compatibility	Development Process
Linux	All major browsers include Chrome, Firefox, and Opera. Many mobile devices run either iOS (Apple) or Android (Google), which is not Linux based. Can run a wide range of	High level of customization, strong security features, supports a wide range of programming languages. Responsive design to look food on different screen sizes. No extra development costs are needed. Does it

	hardware: desktops, servers, routers and more.	require to ensure compatibility across the different distributions.
Mac	All major browsers include Chrome, Firefox, and Safari. Safari is one of the default browsers on MAC and is used for Apple devices and cannot be used for non-apple devices. Can also be used with any mobile device including Android phones.	Mac needs a responsive design to work well on various devices and there is no extra development costs needed. Provides a smooth user experience and good web compatibility.
Windows	Not compatible with every web browser and mobile device. Chrome, Firefox, Opera and Edge on Windows can be used. It cannot access features on the mobile operating systems like how Android/iOS would without additional software. Microsoft Edge is the default for Windows.	Needs responsive design to adapt to different screen sizes, there are also no extra development costs needed. Works smoothly with most applications.
iOS	Will need to be a native app or web-based app.	Developing iOS will be expensive because it requires Swift programming and Apple Developer accounts can be pricy, depending on how long you want to buy the subscription. IOS also needs Apple hardware for development.
Android	Will need to be a native app or web-based app.	Developing for Android is cheaper than iOS. For programming language, they use Java or Kotlin and requires a Google Play Developer account, it is a one-time fee.

Development Tools

Platforms	Development Tools	Licensing Cost	Team Impact
Linux	GIT (version control), Bash (command line interaction), Visual Studio	Red Hat Enterprise Linux Server Self-Support plan costs $349 per year, while the Standard plan costs $799 per year	Using this will require expertise in the Linux environment.
Mac	XCODE, Visual Studio Code, GitHub	The licensing cost can vary depending on the type of software that is being used.	Improves security, fewer virus issues and smoother update process. Requires MacOS and Apple hardware.
Windows	Visual Studio Code, GIT, GitHub, DebugView, Microsoft Azure	Licenses can vary depending on the edition, can range from $100-200 for a standard induvial Windows licenses.	This is easier for teams who are familiar with Microsoft tools.
iOS	XCODE, Swift, SwiftUI, CodeRunner, AppCode	The annual fee for the Apple Developer Program is $99. But there is the Apple Developer Enterprise Program that costs $299 per year.	Requires macOS and Apple hardware.
Android	Android Studio, Kotlin, Firebase, Android SDK	Standard Windows license can cost around $100-200 but it can vary.	Requires expertise in Android Development.

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**: Linux is recommended to use for operating platforms for expanding "Draw It or Lose it." It is free to use and works well for hosting web-based applications. It is also

flexible and has scalable options due to it being open-source and has strong community support. It does require technical knowledge to use it, but you can It is widely used for game applications because it can handle many users and can grow as the game grows.

2. **Operating Systems Architectures**: Linux has a system structure where everything, including the core (called the kernel), device drivers, file management, and network handling, works together as one large process. The kernel is the main part of the system, controlling the hardware, managing resources, and running processes. The shell acts as the interface between the user and the kernel, taking commands from the user and making sure the right functions are carried out. System libraries help applications access important system features, like handling files or managing memory. User applications rely on libraries to perform tasks, ensuring that the system runs smoothly and safely. It also will allow multiple users to access systems and handle many tasks at once. It is built to be flexible, so new features or updates can be added without breaking the system.

3. **Storage Management**: A cloud-based storage system, such as AWS S3 or Google Cloud Storage, is recommended for Linux. These systems can store a large amount of data and grow as needed. For managing game data like user accounts, scores, and teams, a database like MySQL and PostgreSQL is suggested. Databases are reliable and will help ensure that the game information will stay consistent.

4. **Memory Management**: Linux uses memory management techniques to optimize how memory is used, ensuring the system does not run out of resources, even under heavy load. One key technique is paging, which breaks down data into smaller, manageable chunks that can be swapped between RAM and disk storage as needed. This allows Linux to run efficiently even when there is not enough physical RAM available. Virtual memory plays a crucial role here, enabling the system to use disk space (called swap space) as an extension of RAM, just in case there is not enough RAM. Used data can be stored temporarily in memory through caching, allowing the game to retrieve it quickly and improve performance. The kernel's memory management unit (MMU) tracks and controls these memory processes, ensuring data is accessed and managed efficiently, which is especially important when handling many players. These techniques help ensure that "Draw It or Lose It" can scale without slowing down, keeping the game smooth and responsive.

5. **Distributed Systems and Networks**: To allow "Draw It or Lose It" to communicate across multiple platforms, RESTful APIs can be used to share data between the game on the client and the server, such as player actions, game states, and profiles. These APIs help the different devices talk to each other smoothly. Cloud services, along with tools like load balancers and content delivery networks (CDNs), can handle many players at once by spreading out the traffic and reducing delays. This keeps the game running well even when there are a lot of players or small connection problems. To prevent the game from going down during outages, extra servers and backup systems can be used to keep the game available. The system can also keep game data synchronized and consistent across players using methods like data replication, so everyone sees the same game state, even if there are network issues.

1. **Security**: Linux offers several tools to help secure user data. Firewalls can be set up to block unauthorized access to the system, while encryption ensures that sensitive data like passwords and personal information is protected when stored or transmitted. For data being sent between platforms, using secure connections like HTTPS ensures that communication is encrypted and safe from interception. Implementing secure login systems like multi-factor authentication (MFA) can further prevent unauthorized access. Additionally, regular system updates and security patches are crucial to protect against newly discovered vulnerabilities. Linux also supports SELinux (Security-Enhanced Linux), which adds another layer of security by controlling access permissions more strictly. Together, these tools make Linux a reliable platform for user data across various platforms.