

Draw It or Lose It

# **CS 230 Project Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 10/10/23 | Veronica Guzman | Software Design Prototype |

## [Executive Summary](#_sbfa50wo7nsh)

The current software design challenge revolves around crafting a web-based game application known as "Draw It or Lose It" for our client, The Gaming Room. The client's goal is to extend the accessibility of their existing Android app to a range of different platforms. Our proposed solution hinges on the efficient utilization of software design templates and patterns to effectively meet the client's software requirements. This project's central aim is to develop a web-based variant of the game, accommodating multiple teams with multiple players, all while ensuring the uniqueness of both game and team names. Additionally, we will put in place a distinctive identification system for games, teams, and players to enhance the management of instances.

## Requirements

The client has presented a comprehensive set of business and technical requirements for this project. These encompass the need to support multiple teams, each composed of several players, fostering collaboration within the game. Additionally, there is a significant emphasis on ensuring the uniqueness of both game and team names, preventing naming conflicts and enhancing user clarity. The project also involves establishing a distinct identification system that encompasses games, teams, and individual players, enabling efficient management. Finally, there is a strong focus on providing users with the ability to verify name availability when selecting a team name, simplifying the user experience during the setup process.

## [Design Constraints](#_2et92p0)

Developing the web-based game application within a distributed environment comes with several crucial design constraints. Firstly, it necessitates ensuring cross-platform compatibility, ensuring that the game operates smoothly across a variety of operating systems and devices. Scalability is equally critical, as the application must be primed to accommodate a growing user base and an increasing number of concurrent game instances. The implementation of robust data security measures remains paramount to consistently protect user data across diverse platforms, reinforcing trust and privacy. Effective resource management takes center stage, demanding the optimal allocation and utilization of memory and resources to enhance overall performance. Lastly, it's vital to establish reliable network communication between different platforms within the distributed environment, with the goal of minimizing interruptions and delivering users a seamless gaming experience.

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

Starting with the ProgramDriver class, which houses the application's entry point within its main() method, there exists a direct connection with the SingletonTester class. The SingletonTester employs the Singleton design pattern to validate the existence of an instance of GameService. Within the GameService class, a collection of Games is maintained, offering the capability to create new games, complete with their corresponding team and player unique identifiers. This is achieved while utilizing the iterator pattern to ensure the uniqueness of game instances. Furthermore, the GameService class demonstrates multiplicity, indicating a logical association encompassing zero-to-many relationships with the Game class. The Entity class acts as the parent class for Game, Team, and Player, thereby inheriting and sharing attributes among these child classes. Additionally, the Game class extends from Entity through an inheritance relationship, managing teams for individual games and ensuring the uniqueness of newly established teams through the iterator pattern. Similarly, the Game class also exhibits multiplicity, signifying its logical association with the Team class, allowing for zero-to-many relationships. The Team class, following the inheritance relationship from Entity, maintains a list of players and offers functionality to create new players. It employs the iterator pattern to verify the uniqueness of player instances, and like the Game class, showcases multiplicity, indicating its logical association with the Player class in the form of zero-to-many relationships. Lastly, the Player class extends Entity through an inheritance relationship, where it is responsible for constructing each player's unique identifier and name.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** | |
| --- | --- | --- | --- | --- | --- |
| **Server Side** | **Characteristics:**   * **MacOS can be used for server hosting** * **Unix-based OS, making it suitable for web application deployment**   **Advantages:**   * **Stable & secure platform** * **Can easily integrate with other Apple products (iPad, iPhone, etc.)**   **Weaknesses:**   * **Limited hardware options for server hosting** * **Fewer server specific software options (compared to Linux)** | **Characteristics:**   * **Linux is widely used for server hosting** * **Various distributions available, each with different features**   **Advantages:**   * **Excellent for web server hosting due to its stability, security, and scalability** * **Open source reducing licensing cost**   **Weaknesses:**   * **Requires more technical expertise for setup and maintenance** * **Limited compatibility with certain proprietary software** | **Characteristics:**   * **Windows server editions are designed for server hosting** * **Familiar interface for those that are used to windows**   **Advantages:**   * **Good support for .Net and Windows specific tech** * **User friendly for administrators**   **Weaknesses:**   * Licensing costs can be high * May not be as stable as Linux for certain server applications | **iOS & Android**  **Characteristics:**   * **iOS: iCloud integration, Apple specific technologies, strong ecosystem integration** * **Android: cloud platform agnostic, wide tech stack compatibility, and open ecosystem**   **Advantages:**   * **iOS: seamless Apple ecosystem, robust security, and optimized for Apple hardware** * **Android: flexible hosting options, tailored solutions, and scalability with diverse devices**   **Weaknesses:**   * **iOS: Limited compatibility, vendor lock-in, and higher development costs** * **Android: Fragmentation challenges, security concerns, and testing complexity.** |
| **Client Side** | **Characteristics:**   * **MacOS provides user friendly interface** * **Commonly used for creative and design tasks** * **Integrates well with other Apple devices**   **Advantages:**   * **Smooth user experience with well-designed interfaces** * **Seamless synchronization with other Apple products** * **Access to the app store for a wide range of applications**   **Weaknesses:**   * **Limited gaming options (compared to windows)** * **Products are on the higher scale of pricing** * **Limited software compatibility outside of the Apple ecosystem** | **Characteristics:**   * **Offers various distributions with different desktop environments** * **Highly customizable and open source** * **Known for stability and security**   **Advantages:**   * **Excellent for development and server related tasks** * **Wide range of open-source software available**   **Weaknesses:**   * **Steeper learning curve for beginners** * **Limited commercial software support** * **Hardware compatibility issues with certain devices** | **Characteristics:**   * **Dominant OS in the desktop market** * **Known for a wide range of software compatibility** * **Offers different editions for various user needs**   **Advantages:**   * **Vast software library including popular applications and games** * **Extensive hardware combability** * **User friendly interface**   **Weaknesses:**   * **Frequent updates can disrupt user’s workflow** * **Vulnerable to malware and security threats** * **Licensing costs for some editions** | **iOS & Android:**  **Characteristics:**   * **iOS: Exclusive mobile OS developed by Apple** * **Android: Open-source mobile OS developed by Google** * **Both offer a wide range of apps through their own app stores**   **Advantages:**   * **iOS: smooth and consistent user experience across Apple devices** * **Android: customizability and wide range of device options** * **Access to a vast ecosystem of mobile apps**   **Weaknesses:**   * **iOS: Limited customization and closed ecosystem** * **Android: Fragmentation can complicate development** |
| **Development Tools** | **Characteristics:**   * **XCode is the primary IDE for macOS** * **Supports development for iOS, macOS, watchOS, and tvOS**   **Advantages:**   * **XCode offers a comprehensive development environment for Apple platforms** * **Strong integration with Apples developer ecosystem** * **access to the latest features and APIs**   **Weaknesses:**   * **Limited cross platform development support** * **Mac hardware can be expensive for some developers** | **Characteristics:**   * **Various text editors and IDEs available (Visual Studio code, Eclipse, etc.)** * **Supports a wide range of programming languages**   **Advantages:**   * **Open-source development tools** * **Highly customizable development environment** * **Ideal for web and server-side development**   **Weaknesses:**   * **Limited official support from some software vendors** * **May require additional setup for specific development environments** | **Characteristics:**   * **Visual studio is a prominent IDE for windows development** * **Supports development in various programming languages**   **Advantages:**   * **Strong support for windows application development** * **Extensive debugging and profiling tools** * **Access to the Microsoft developer ecosystem**   **Weaknesses:**   * Limited support for non-windows platforms * Licensing costs for some editions of visual studio | **iOS & Android**  **Characteristics:**   * **iOS: Development primarily done in XCode using Swift or Objective-C** * **Android: Development in Android Studio using Java or Kotlin**   **Advantages:**   * **iOS: Integrated development environment tailored for Apple platforms** * **Android: Strong support for building apps for diverse Android devices** * **Access to platform-specific features and APIs**   **Weaknesses:**   * **iOS: Limited to Apple’s ecosystem** * **Android: Fragmentation can lead to compatibility challenges** * **Learning curve for platform specific 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**: I recommend utilizing the Linux operating platform for Draw It or Lose It. Linux aligns well with the client's objectives due to its open-source nature, stability, and security. It is highly customizable and provides an excellent foundation for cross-platform compatibility. Additionally, Linux supports various hardware architectures, making it versatile for different computing environments.
2. **Operating Systems Architectures**: Linux is built on a Unix-like architecture, which offers stability and security. It follows a multi-user, multitasking model that allows multiple users to run programs simultaneously. Linux supports both 32-bit and 64-bit architectures, ensuring compatibility with a wide range of hardware. Its modular design allows for easy customization and optimization based on specific requirements.
3. **Storage Management**: For storage management, Linux offers a plethora of options. A combination of cloud-based storage solutions like Amazon S3 or Azure Blob Storage can be used for scalable and reliable storage. These cloud services provide redundancy, backup, and disaster recovery options. Relational databases like MySQL or PostgreSQL can be deployed on Linux servers to manage structured data efficiently. These databases offer features such as data integrity, transactions, and ACID compliance, ensuring the security and reliability of the stored information.
4. **Memory Management**: Linux employs advanced memory management techniques to optimize resource usage for applications like Draw It or Lose It. It utilizes virtual memory, allowing processes to use more memory than physically available by transferring data to disk storage. Linux also incorporates process management and memory protection mechanisms to prevent unauthorized access and ensure data integrity. Regular monitoring and optimization are essential to identify and fix memory leaks and resource usage issues, ensuring efficient performance of the software.
5. **Distributed Systems and Networks**: To enable cross-platform communication, a service-oriented architecture (SOA) with RESTful APIs can be implemented. RESTful APIs allow different platforms to communicate via HTTP protocols, ensuring interoperability. Load balancing techniques distribute incoming network traffic across multiple servers, enhancing scalability and responsiveness. Message queues facilitate asynchronous communication, allowing tasks to be executed independently, enhancing efficiency. Failover and redundancy mechanisms ensure system reliability, minimizing downtime during outages. Effective monitoring tools, security protocols, and compatibility measures should be in place to address connectivity issues and ensure the seamless operation of Draw It or Lose It across various platforms.
6. **Security**: Linux systems can be hardened to protect against vulnerabilities and unauthorized access. Utilizing HTTPS for encrypted data transmission ensures secure communication between platforms. Robust authentication and authorization systems should be implemented to control user access and actions within the system. Regular security audits and updates are essential to identify and patch potential security threats. Linux's security capabilities, coupled with secure coding practices and regular security assessments, will ensure the protection of user information on and between various platforms.