

<Draw it or Lost it>

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 07/16/2023 | David Gibson | Changes have been made to the Executive summary, Design constraints, and Domain model. |
| 1.1 | 07/30/2023 | David Gibson | Updated Development Requirements |
| 1.2 | 08/10/2023 | David Gibson | Operating Server Recommendations |

**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](#_sbfa50wo7nsh)

The Gaming Room has chosen Creative Technology Solutions to bring their existing Android game, Draw It or Lose It, into the broader digital landscape. Our task is to build a web-based adaptation of the game that functions seamlessly across various platforms. Draw It or Lose It is a collaborative game where teams compete to guess drawings from an extensive stock image collection. The game includes four one-minute rounds, with each drawing reaching completion halfway through each round. If a team can't decipher the drawing within the allotted time, the remaining teams are allowed to guess.

## Requirements

* **Multiple Teams & Players:** The application must support numerous teams with multiple players each, necessitating a robust architecture for concurrent user management.
* **Unique Naming System:** Game and team names need to be unique, implying an efficient real-time checking and locking mechanism to prevent duplications.
* **Single Game Instance in Memory:** The application must maintain only one game instance in memory at any given time, stressing the need for effective memory and session management.
* **Staff User-Friendly:** The application must be designed in a manner that is easy to manage by The Gaming Room staff, who may not possess extensive technical knowledge.

## [Design Constraints](#_2et92p0)

Developing a web-based game like "Draw It or Lose It" poses several design constraints. The game must be compatible across various platforms, which requires meticulous design and testing. The system must support many users at once and ensure unique game and team names. Only one game instance should exist in memory at any given time. The interface needs to be easy for The Gaming Room's non-technical staff to navigate, which might involve creating simple documentation or even training. Lastly, the game should incorporate strong data security measures.

## [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)

The GameService class is implemented as a Singleton, ensuring a single instance controls the game state, adhering to the principle of encapsulation. Utility classes SingletonTester and ProgramDriver interact with GameService to drive the application flow. The Entity class is a base class holding common attributes (id and name), demonstrating the principle of inheritance, and improving code reusability. Game, Team, and Player are subclasses of Entity, inheriting its attributes and behaviors. The Game class contains a list of Team objects, and Team holds a list of Player objects, representing a hierarchy in the gaming domain. The principle of encapsulation is shown by keeping data attributes private and providing public getter methods. Polymorphism is shown by the toString method being overridden in subclasses to offer class-specific implementations. These principles help to fulfill software requirements efficiently by providing a well-structured, understandable, and maintainable codebase, allowing easy tracking and addition of games, teams, and players.

**"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.

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 a reliable choice for hosting web-based software applications. It offers high build quality and performance, making it suitable for resource-intensive applications. One downside is that it is more expensive, and the server management tools might not be as comprehensive as Linux and Windows. | Linux is open source allowing customization and optimization, making it popular for necessary server applications. It provides strong security and flexibility with support for various programming languages and frameworks. One major weakness is the learning curve. | Windows servers are well-known for their instinctive and user-friendly interface. The servers seamlessly integrate with the large collection of Microsoft's development tools, making it convenient for developers. Windows servers need frequent updates to maintain optimal performance and protection. Windows servers also have higher licensing costs, which could greatly impact a company’s budget considerations. | Mobile devices are not recommended for server-side hosting of web-based software applications. Server-side hosting generally requires powerful hardware, stable network connectivity, and constant operation, which are not attributes of mobile devices. |
| **Client Side** | When supporting multiple types of clients on Mac, it's important to consider the price, duration, and expertise required. The price might involve expenses for Mac hardware and software licenses. Development duration can vary based on application complexness and familiarity with macOS-specific technologies. A moderate amount of expertise is required. | Supporting multiple types of clients on Linux can be cost effective due to it being open source. Development time may be affected by compatibility testing across the various Linux distributions. Proficiency with a language like C, and familiarity with Linux distributions and tools are necessary. The expertise required is higher compared to the other OS. | When supporting multiple types of clients on Windows, you may incur costs related to licensing expenses for development tools and software required for Windows applications. Windows offers an abundance of resources that can expedite development times. A low amount of expertise is required. | When supporting multiple types of clients on mobile devices, costs may rise due to multiple development environments and physical testing devices. Development time could increase as separate codebases for Android and iOS need maintenance and thorough testing. A moderate amount of expertise is required. |
| **Development Tools** | Developers on macOS commonly use HTML, CSS, JavaScript, and server-side languages like Node.js, Python, or Ruby for web development. As for tools and IDEs, Mac can utilize a variety of code editors like Visual Studio Code, PyCharm. | Developers on Linus can use the same set as Mac HTML, CSS, JavaScript, and server-side languages like Node.js, Python, or Ruby for web development. As for tools and IDEs, Linus can utilize a variety of code editors like Visual Studio Code, PyCharm, and Eclipse. | Windows developers also rely on the same web development languages, HTML, CSS, JavaScript, and server-side languages like Node.js, Python, or Ruby for web development. As for tools and IDEs, Windows can utilize a variety of code editors like Visual Studio Code, PyCharm, and Eclipse. | Mobile web development typically involves HTML, CSS, and JavaScript for creating responsive and mobile-friendly websites. In addition to mobile web development, developers can also leverage the power of Android and Swift to create mobile applications. Both languages and software can be run on all three major platforms - Mac, Linux, and Windows, offering flexibility and accessibility for developers. |

## 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 using the newest version of Windows Server to expand Draw It or Lose It to various computing environments. It's a powerful, secure, and scalable operating system suitable for game servers. It's designed to support a wide range of applications, making it particularly suitable for gaming servers. Its adaptability ensures that Draw It or Lose It runs efficiently, providing players with a smooth and enjoyable gaming experience.
2. **Operating Systems Architectures**:Windows Servers utilize a dual-mode operational structure for efficient system management. In the Kernel Mode, it handles important, low-level tasks directly related to the system's hardware. This mode is essential for stability and speed. On the other hand, the User Mode is designed for running standard applications, guaranteeing that the entire system isn't jeopardized if an app faces issues. By splitting these responsibilities, Windows Server achieves a balance of performance and security, providing a reliable environment for gaming applications.
3. **Storage Management**: Storage Spaces Direct (S2D) is an excellent choice for storage management within the Windows Server environment. It lets you group together attached drives to create a single large storage area. Using disk types NVMe, SSDs, and advanced caching ensures fast and reliable storage, making it perfect for games like Draw It or Lose It. S2D's adaptable storage means it can handle varying demands, scaling up or down as needed.
4. **Memory Management**: Windows Server memory management is designed for performance, especially for gaming applications like Draw It or Lose It. A key feature is its use of virtual memory. The server extends available memory by assigning a portion of the hard drive space by using virtual memory, effectively augmenting the physical RAM. Meaning that even when the actual RAM starts to fill up, Draw It or Lose It can run smoothly without issues, as less frequently used memory components are moved to this virtual space. It provides consistent performance, even under heavy loads or extensive play sessions.
5. **Distributed Systems and Networks**: Both client-side and server-side tools are essential to allow Draw It or Lose It to function on any device. On the client side, APIs ensure that whether players use a phone, tablet, or computer, their device communicates correctly with the main game server. On the server side, load balancers manage incoming player data, making sure no server gets too much traffic. Lastly, if a server has a problem, failover immediately shifts tasks to backup servers, guaranteeing the game runs smoothly for all players.
6. **Security**: To keep Draw It or Lose It safe for users, we can focus on three key areas. First, Authentication & Authorization ensure that players are who they say they are, using methods like multi-factor checks. Next, Regular Updates & Patches are essential. By regularly updating the game's server, known weak spots that hackers might target are fixed. Finally, Client-Side Security is about making sure players' devices are safe. By urging players to keep their game and device software up-to-date, potential entry points for hackers are reduced. Together these practices can provide a secure gaming experience for everyone.