

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

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 | <11/16/24> | <Jian Wang> | < Present software design problems, design constraints, and present solutions to meet customer requirements for software. > |

**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 wanted us to design a web-based multi-platform version of their existing Android-only game Draw It or Lose It. The game is based on the 1980s TV game Win, Lose or Draw. Our goal was to help the client list software design problems, design constraints, and propose solutions.>

## Requirements

1. The software must be available on multiple platforms.

2. A game will have the ability to have one or more teams involved.

3. Each team will have multiple players assigned to it.

4. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.

5. Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

1. This web-based, multi-platform software that creates compatibility issues. This can be solved with multiple tests.
2. Multiple teams with multiple players participating in the game will create naming duplication issues. This can be solved by setting name uniqueness.
3. Only one instance of the game can exist in memory for a specified period. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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)

< Creating a game is done through the Game Service class. The Game class is introduced through the “add game” method, and the uniqueness of the game is achieved through get game id and name. Ensure that there is only one game in memory at the same time. In the Game class, the Team class is introduced through “add team” method, and then the player class is introduced through “add player” method. Similarly, the uniqueness of the team and player is achieved through id and name.

The entity class is the parent class of the game, team, and player classes. It defines the properties of other classes.>

**"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 operating system is based on Unix and has high stability and reliability> | <Linux is a very stable operating system that can run developed software efficiently and maintain long-term stability.> | <Although windows is not an open source system, but there are a lot of options and good compatibility.> | <Portability allows users to work anytime, anywhere. But usually good equipment requires a higher price> |
| **Client Side** | <Although the Mac operating system focuses on the user experience, but the hardware is expensive and the number of customers is small, and very few people use macs to play games.> | <Linux is an open-source operating system, which means you can modify, customize, and distribute the software freely.  But Linux has very little support for gaming, and at the same time, almost no home users use Linux  > | <There are a lot of users using it, which means that many problems can be found to be solved. It is easy to use and the hardware is not expensive, which is why almost all home users are using windows.> | <People can choose different operating systems according to their needs.> |
| **Development Tools** | <Mac operating system enables cross-platform development.> | <Because Linux is an open-source system, there are a lot of development tools available, as well as cross-platform development> | <Almost all software can be used on windows, and there are many tools to use. Also support cross-platform development> | <Different operating systems can choose different development tools, supporting cross-platform 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**: < Windows is recommended because it has the best compatibility and can easily expand the game to other environments. Also, windows provides rich management functions.>
2. Operating Systems Architectures: <

Kerne, the kernel of the Windows system is the core part of the entire system, which provides various basic system services and resource management functions.

Subsystem, the subsystem of the Windows system is used to handle different types of applications. Each subsystem has its own PI interface to handle different types of applications.

Driver, the driver of the Windows system is used to control hardware devices, such as graphics card drivers, sound card drivers, network card drivers. Each driver has its own PI interface to control hardware devices.

Application, the application of the Windows system is used to handle various tasks, which includes multiple applications, such as Office applications, browser applications, game applications.>

1. **Storage Management**: < Windows Storage Management can manage storage space and optimize storage by releasing space on the hard drive. Disk cleanup function can be performed by deleting unnecessary files. In addition, the Windows Storage Management Console provides a graphical interface for managing computer storage resources.>
2. **Memory Management**: < Memory management keeps track of all memory locations. Specifies how much memory each process should be given. It decides which processes will be remembered and when. It tracks when memory is released or when it is shared and changes the status accordingly.>
3. **Distributed Systems and Networks**: < To enable communication between different platforms, games can be implemented by implementing centralized servers that can handle game synchronization, real-time updates, and message exchanges between players on different platforms. The server side has the right to make changes, while the client side only has the right to execute. This prevents the client from being modified, resulting in an imbalance in the game. The server side has the right to deny access when there is an error message on the client side, such as a duplicate name.>
4. **Security**: <Access to the server side can be controlled through user authentication and authorization mechanisms. In addition, data encryption technology can be used to protect customer information.>