

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 | <07/16/24> | Keith Shark | Outlining project requirements and constraints of Draw It or Lose it |

**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)

To make the web application more streamlined the game system through using the Entity class to abstract the common elements that Game, Team, and Player have. The unique identifier and name. Below are the requirements and in-depth solutions addressing each requirement.

## Requirements

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

I created an addTeam method that handles the creation of teams.

* Each team will have multiple players assigned to it.

The addPlayer method handles the creation of a player for a given team.

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

To ensure that each game and team is unique. For the process of creating teams and games I used iterative patterns to check if a team or game with the same name exists. If not, I used the amount teams plus 1 as my unique identifier.

* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

We need to ensure that the application is responsive with all types of display resolutions. It’s also good to be mindful of which dependencies we use to ensure that the application functions as intended on all browsers. We also need to consider the hardware limitations on smartphones and other devices.

## [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 the base class using the singleton pattern in milestone one. This class also stores the next unique identifiers for players and teams. The Game, Team, Player class all inherit from Entity. Entity class holds the unique identifiers and name for all three classes. Each class Game, Player, Team handle the creation and array to keep track of teams and players. All relationships between GameService, Game, Team, and Player are zero to many relationships. The GameService class can have multiple games or no games at all, the same can be said about players and teams. For Player, Team, and Games they all extend the functionality of Entity.

**"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** | Macs are not as customizable hardware as other operating systems. It is more useful for interfacing with servers. | Linux is the most popular choice and versatile. Linux is extremely customizable for commercial and individual use. | Windows is another popular choice amongst enterprise users. Microsoft provides tools and services that streamline the process of running servers. | Mobile devices lack the performance and capacity to be an effective server. It is ideal for hobbyists. |
| **Client Side** | Apple products in general exist in a closed ecosystem, which makes Macs very secure. However, the downside is the cost of using products, and there are not as many tools for the client side as other options. | Linux costs less money compared to other operating systems. However, the learning curve for the operating system is significantly higher. You also might run into some issues using creative software like Adobe illustrator. | Windows is a popular option for many people. It is easy to use compared to Linux. | Main concern with mobile devices is making the user experience seamless with smaller screen sizes. There might be some learning curve when developing any app for phones. |
| **Development Tools** | For Macs the I would the main tools for developing within the apple ecosystem is Xcode. | Linux is written in a low-level programming language like c and c++. Tools like Vim and emac are popular terminal applications that developers use or vscode. | This can be built in c# or c++. Both have tools and libraries for web applications. | For smartphone Android Studio with Java and Swift for IOS. |

## 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**:

After analyzing all operating systems I’ve concluded that Linux would be the best operating system of choice.

1. **Operating Systems Architectures**:

Linux operating system utilizes a monolithic kernel architecture. Monolithic kernels are performant and simplistic as they have control over the entire memory space. There are no communication overhead.

1. **Storage Management**: For simplistic applications partitioning a disk space low-level operating system can work. However, for more scalable storage using a cloud computing service would be preferable.
2. **Memory Management**: Our server should provide tools and libraries allowing us to manipulate data safely. Linux provides the fundamental tools for encrypting and compressing data for efficient transfer.
3. **Distributed Systems and Networks**:

Linux is developer friendly; it comes with standard libraries that allow low-level network communications. It is customizable, writing personalized scripts for execution.

1. **Security**: Linux uses role-based access control to restrict data access. Extra layer of security measures can be used like private networks and encryption. Creating a secure barrier against would be attackers.