

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 | 1/28/2024 | Destiny Mancha | A separate class, named Entity was created with ID and name parameters to ensure one instance of a game existing at a given time. This class is inherited by other classes. The ability to add players to teams was also added. |

**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 is requesting that Creative Technology Solutions develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It. This is currently on Android only. It can be costly and time consuming to make this game playable on multiple platforms. Using Windows is easy to use and modify, making it ideal for developing for the use of the game on multiple platforms.

## Requirements

The Gaming Room would like to develop the game while staying within budget and the deadline. They would like the web-based game to be converted to and playable on multiple platforms.

## [Design Constraints](#_2et92p0)

1. Only one instance of the game can be active at a time.
2. The game must be cross-platform compatible.
3. The game must allow one or more teams and multiple players within those teams.
4. Team names must be unique with their own ID.

## [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 Game, Team, and Player classes extend the Entity class. In other words, they inherit from the Entity class. For example, each class shares the attributes “id” and “name.” This makes Entity a superclass. The Team and Player classes have an aggregation relationship with Game. Game classes have teams and the GameService class manages the Games.

**"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** | Yes, Mac can be used as a server. However, it is expensive for licensing and macBooks are required for development. It is more commonly used for web applications. | Yes, Linux can be used as a server. It is more secure, reliable, and does not freeze as much. It is low cost and doesn’t require licensing fees. Quick development. | Yes, Windows can be used as a server, but the licensing can be costly. It is also not as secure as Linux and Mac. | Not recommended to be used as server. It can’t compete with the others. Resources are limited, which can result in performance issues. Does not have great security. |
| **Client Side** | Mac has good software development kits, but you have to use MacBook to develop and you need someone experienced with macOS and swift. This can be costly. | Linux is low cost, however, there is a steep learning curve, so it can be time consuming and you will need someone with experience developing in Linux. | Windows licensing can be costly. Requires an expert in .NET languages, such as C#. Easy cross-platform testing (unless for macOS). | An experienced person in developing apps is required as it is different than developing on web. Longer development time. |
| **Development Tools** | Using swift will make the software development cycles faster. Mac also provides good SKDs. | Eclipse, Visual Studios, PyCharm offer tools for development. Can run macOS and windows as well. | Supports almost all software development languages and tools. | Android, Mac, or unity. Unity can be programmed using .NET and C#. |

## 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 the recommended operating platform. Windows has a wide variety of software development tools and is compatible with numerous programming languages. This makes it easy to learn.
2. **Operating Systems Architectures**: Windows offers graph user interfaces, which would make the game very user-friendly, efficient, aesthetic, clear, and accessible.
3. **Storage Management**: Windows uses Storage Spaces, which help protect data from drive failures. It also allows for an extension of storage over time. Windows allows for drive space management using Space Sense. This feature can free space up for the user automatically by deleting items they do not need.
4. **Memory Management**: Because the game requires a library of images, the memory allocation features will allow the storage of the pictures outside the default folder. These features enable the organization and security of the files. This is especially beneficial when working in an Integrated Development Environment.
5. **Distributed Systems and Networks**: Windows provides numerous tools, such as “Unity” for cross-platform game development. Windows is a popular game development platform. There is enough server infrastructure to allow for large player volumes. There is also enough to provide backup power to make up for outages or connectivity issues.
6. **Security**: Admittedly, Windows is not the most secure operating system. However, on top of the built-in security, there are other options that can and should be taken to secure user data. The additional security can cost money, but the simplicity in compatibility with multiple platforms makes up for the extra cost. Windows also supplies scanning for any security threats, so long as the OS is updated accordingly.