

<Name-of-Software-Application>

# **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 | <03/17/2023> | <Ethan-Menchaca> | <Disclosed information on design constraints, created an executive summary and included a domain model.> |

**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)

A new client, The Gaming Room, has a video game called Draw it or lose it. The game is currently only available on Android; however, the Gaming Room would like this game to be available on the web. The end goal is to have the web-based version be available to multiple different platforms. In order to help design the web based version of this game, the Gaming Room has reached out to us. One solution to their problem would be to use a cross platform framework. Many games have already successfully integrated it, such as Fortnite and Dead by daylight. The only downside to this approach is time. It takes much longer to develop a game for multiple platforms. The Gaming Room has specified a series of technical requirements to coincide with his project. The first is that the game must have the ability to include one or more teams and a way to assign multiple players to each team. The second is that there must be a way to check to see if each team name is unique by seeing if their current team or game name is in use. Finally, only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

1. **Multiple Platforms:** The main objective for this project is to make Draw it or lose it web based and design it in a way which will allow the game to be played across multiple platforms. This will entail creating a cross platform framework which fits within the Gaming Rooms budget, which could also be a business constraint when price is considered. As for the technical constraint, this means the game must be coded in a way in which it can be accessed by multiple platforms.
2. **Only one instance:** One important design constraint is to ensure that only one instance of the game is available in memory. This can be accomplished by using a singleton pattern within the code of the Draw it or Lose it, which in turn will allow only one instance of each game, player, and team to exist within storage. Each instance will have to have a unique identifier, which will have to be checked to make sure only that instance is in use. This is to ensure that when a team name is selected it isn’t already being used.
3. **Allowing for multiple players and teams:**  One feature of Draw it or Lose it is that it allows for multiple players to exist within a team. This must be coded into the game to allow each player to be designated to a specfic team and to assign each player and team a unique name which can be checked by an iterator to see if it is in use. If the name of the team or player is in use, the code must return a message saying “unavailable.” The information for each team name and player name will be stored in a list.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram for the Draw it or Lose it game is shown below. It shows the base Entity class, which is the parent class for the Game, Player, and Team classes. This means that the child classes (Game, Player, and Team) can inherit and use the attributes and methods listed in the Entity class. Attributes are listed in red lettering while the methods are written in blue. The lines connecting GameService class and each child class to one another shows a symbol of 0…\* which means there can zero to many cases of each class. Many games, players, and teams can be made. The ProgramDriver class shows an arrow which points to the SingletonTester class, labeled as “uses”. This is because the ProgramDriver class will use the SingletonTester to ensure that only one instance of the game existing in memory at any given time.

\**"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 has many advantages, such as having an efficient and simple user interface which can be easily understood. However, Mac comes with some disadvantages, mainly being price and the inability to be cross platform. | Linux is renown for its efficient environments which is one of its biggest strengths. It supports cross platform programming. However, Linux isn’t as popular as mac or Windows, so supporting applications for web hosting are much harder to find. | The main strength for Windows is that is incredibly popular and user friendly. Majority of users are familiar with windows operating system, because most likely it is the system they started on. However, due to its popularity windows is subjected to more security threats than its counter parts. | Mobile devices can bring about increased mobility due to its small frame. It also has a wider consumer base, as almost everyone carries a smart phone. However, like windows, mobile devices offer very little security. |
| **Client Side** | Mac is much more expensive than its counterparts. Also, it isn’t as common as windows, so it may take time to garner a certain level of competency. | In terms of cost, Linux is very cheap and affordable. However, like Mac it is relatively rare and therefore expertise in this system would be less common as well. | Windows is cheaper than Mac and in terms of experience almost everyone that has worked with computers has knowledge on windows. It is very common and probably the most frequently used OS, therefore many users will already know how to efficiently interact with it. | Mobile devices offer more mobility as the name suggest, however, it would be less effective and harder to achieve the same goals with a mobile device. |
| **Development Tools** | Swift and objective c are popular languages with Mac os. | Eclipse, visual studio, python, and many others can be run and used on Linux. It is very diverse. | Almost every programming language can be run on windows, due to it being arguably the most popular operating system on the planet. | Switic is the best language for app development. It is compatible with Android and 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**: Linux would be the best choice for Gaming Room in order to expand its software to differing operating platforms. This can be done by setting up a Linux server which can allow alternate operating platforms to access the software through Linux Distro, such as Ubuntu and openSUSE. Setting up a Linux server comes with many pros, such as increased flexibility, stability, and security. By using a Linux server we can use the client/server architecture which in turn will support multi-platform access.
2. **Operating Systems Architectures**: The Linux operating system has four elements that constitute the Linux architecture. They are, in sequential order, hardware, kernel, shell and applications. The heart of the Linux operating system, as with every computing system, is the hardware. The hardware is made up of things such as the CPU and RAM. It is the lowest level of the Linux system architecture. Moving up from hardware, the next layer of the Linux architecture is the kernel. The kernel is arguably the most important and complex as it is responsible for every major action of the Linux operating system and communicates directly with the hardware of the operating system. The kernel is responsible for things such as memory isolation, file management, handle system calls, etc. The kernel could be considered monolithic, since it groups all of the basic services into the kernel. The next layer in the Linux architecture is the shell. This is the layer that provides an interface for the user to interact with in order to launch a program. The shell can be two types, a GUI or a command line. The final layer of the Linux architecture is the application layer. This layer represents the software that is being run on the Linux operating system, such as sublime text editor, eclipse, etc. Below is a graphical representation of the architecture of the Linux operating system.

**Chart, sunburst chart

Description automatically generated**

1. **Storage Management**: For the draw it or lose it application I would recommend using cloud-based storage. This will allow the Gaming Room to upgrade storage as their player base and application grows with minimum difficulty. The cost of the monthly subscription would have to be factored into the Gaming Room’s budget, but it would offer more flexibility and in the long term be more cost-effective.
2. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

For most computing systems, physical memory is often very limited. In order to overcome this problem, Linux uses something called virtual memory. Virtual memory is a memory management technique which transfers important data from different storage systems to support running applications, such as the RAM and hard drives, and leaves dormant data in the virtual memory until it is needed. This allows the computer to minimize the amount of memory used to run any given application which in turn allows multiple different applications to run at the same time. This also avoids the issue of having differing applications competing for shared memory space. The result is a faster and more efficient memory management system. The virtual memory does this by moving any files that aren’t currently in use in the RAM to the hard disk. This frees up space in the RAM(short term memory) for applications currently running.

1. **Security** **Distributed Systems and Networks**:

Using a cloud service could circumvent the damage an outage would cause to the gaming rooms end users. If an outage were to occur, most cloud services can be rerouted to different locations and servers which will still allow user connectivity.

1. **Security:**

There are many different considerations when dealing with security for Draw it or Lose it. When it comes to the end user, the most important way the gaming room can minimize security risks is by in turn minimizing the amount of control the end user has over the game. The user should be able to edit team name and color, and username, but privileges shouldn’t extend very far beyond that. When it comes to network security, a firewall should be implemented as a part of the server to prevent malicious attacks to the end user, and the APIs should be encrypted.