

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/21/2022 | David D. Scott | Implementation of team instances in order to have multiple players on multiple different teams. We will have the code formatted so that only one instance can exist at a time. |

**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)

We have been asked to help with the development of the application Draw It or Lose It. The environment created for this game will require the ability to have one or more teams who can have multiple players within a specific team. Every name, including both the teams and players, must be unique and there needs to be a way ensure that each name isn’t already being used. All of these needs will also be written in a way that is profitable and functional.

## [Design Constraints](#_2et92p0)

One of the biggest constraints to this project is the Budget. It was made very clear in the beginning that the client wants to make this application within a set budget. This set budget can also make us stay on some tight time constraints as well. Further, there is only one instance of the game that can exist in memory at a time. Finally, we need to ensure that we stay to the original model as well. It has been very profitable for the company, and they want to expand on the same strengths that has already made them successful in the first place.

## [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 UML diagram shows us a lot about what the application is going to look like and how it is going to function. The Entity class will be used to build the base for the child classes. That base will consist of getters for the id and the name, both of which will be used by the child classes. This will save us effort and time because it will provide the variables for Game, Team, and Player. Game will create a list that will be used to hold all of the individual teams. Also, it will be used to add teams to the list as well. Team will be used to create a list of players and also add players to that list. Player will give us the ability to make an player instance. GameService will use a singleton pattern to ensure that only one instance of the game can run at a time. SingletonTester is used in conjunction with the ProgramDriver to be tested.

**"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** | Not very low budget friendly,  Server configuration is on the friendly side  You do have the ability to make your own server through Apple directly which can bring extra security. It may not be as well know but can always be an option for a relatively low cost.  The terminal command function is still very existing on this platform as well by being used to configure multiple things with ease if you are familiar with it. This can be quite the handicap however if you do not have the know how on your team. | Multiple different versions for different uses,  Has PowerShell compatibility  Linux has a vast array of option when it comes to server creation and management. This can also be on the lowest cost options out of the group. The software used for creation is open-source but can lead to bottle necks due to low capabilities. Linux is also not a very well know platform to the general audience. So this option as well will more than likely require a specialist. While you don’t have restraints in options, what you want to accomplish would have to be put into place before hand. | A good middle ground of both expensive and reasonable,  Third Party software options are almost endless  Windows offers some of the most functional server creation of them all with Windows Server. Also, windows has plenty of third party options. This is the most likely chose because most people are familiar with Windows. Since it is used more than the others, it will be far easier to find help for problems that others have experienced themselves. | Cheaper to make, but more variations to think about,  Cross platform development is easier than ever  Mobile platforms are far underpowered when considered next to a full blown server setup. This aspect would cause you to run into many technical contrasts due to hardware. The only pro to this server is the cost can be lower. Moreover, the process would require addition steps to create and maintain the server that the other options. |
| **Client Side** | Still expensive,  Easier GUI layout for most  Moderate skill will be needed to use  The cost across the board is very similar here. Time, just as with the server, will be dictated by team capability. Also, the other aspect is the same as well when considering how easy the maintenance of the client side will be. The greater downside is that the number of coding and file structures are limited and can lead to bottle necks. | The least cost as most Linux versions is free,  Higher skill needed to use. Least common of the Oss  This is truly the same case as the Apple option. You must have someone who is well experienced and taught about Linux. However, the client side can easily be managed without the use of extra programming due to built in capabilities. If this can be used in conjunction with meeting the server requirements, this is a very good option. It is also important to note that over half of all web servers have been ran on Linux. | Depending on needs, can be far less expensive compared to Mac,  Most common mean more individuals need less time to know how to use something new  When using Windows Server, you do not gain the advantage of it being open sourced. Because of this, while it may be the most well known OS, you would still need someone who is experienced and very knowledgeable. Not doing so would still lead to longer development times. However, in the case that someone is not, the resource material will be far more than the other options in consideration. | This is the most rapid changing platform of the bunch,  With rapid changes come rapid support with the knowledge of multiple different types of devices and what they can handle  A pro is the portability of the mobile device. However, the need of a constant internet connection to do anything can be a huge downside. Mobile devices do have frameworks that are specifically designed to make the phone experience great, but lacks the same confidence on desktop application. |
| **Development Tools** | Swift,  Python,  CSS,  HTML,  Visual Studio,  Git Hub,  Eclipse,  Etc.  With Mac, Swift is the most commonly used programing language. Further, there are a lot of IDEs that can utilize the coding language. Apple also offers Xcode for free for Swift development and has a very friendly user interface. | Java  CSS,  Python,  Ruby,  Visual Studio,  Git Hub,  Terminal,  Eclipse,  Etc.  Two of the most common IDEs for Linux, Eclipse and Atom, are very well known. Further, they both can support a decent amount of coding languages.  Both of them can be used for cross platform development to meet the client’s need. | Java,  C++,  C#,  Python,  Visual Studio,  Git Hub,  Terminal,  Eclipse,  Etc.  Eclipse and Visual Studio are extremely popular IDEs in the Windows development world. HTML, JavaScript, C +, and much more can be developed on the Windows platform with cross platform capabilities. | CSS,  HTML,  C++,  Python,  Java,  Visual Studio,  Git Hub,  Etc.  For starters, it is hard to develop an app for all platforms through one platform being Apple or Android. Apple still utilizes Swift and Android utilizes Java and others. This is a huge consideration because Apple does not support Java. If not planned for accordingly, this can lead to tremendous restraints for different features fixed through out the application. |

## 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 would be the best option for the game. It is the easiest and most widely known OS to work with. It also offers many resources online for research and development. This creates the least number of risks overall to get started, finish, and maintain the game.
2. **Operating Systems Architectures**: Windows is one of the most cost-efficient options. It also offers the expertise needed to make the game easier to come across when assigning the team. Windows also offers GUI for its applications that are web based.
3. **Storage Management**: Windows once again shows incredible options for storage management. They have tools that can make managing the digital space easy and stable while giving the option to be able to back up data in case of failures. There are also plenty of third-party supporters for physical storage for any need at reasonable costs towards the budget.
4. **Memory Management**: This is one of the downsides of Windows due to the number of third-party supporters. These applications can be poorly optimized to work in conjunction with other programs. However, Windows uses has memory management built in as a function of the system utility. This is vital for the management of the files and images that will constantly need to be managed to make the game run smoothly.
5. **Distributed Systems and Networks**: We can utilize a client-server system with Windows. Each game will need its own host and instance. We can use Windows to connect to the best host for the most optimal gameplay. A strong server network would also be needed to efficiently switch between these hosts. The success of this game greatly depends on multiple clients being able to connect to a single server and play together.
6. **Security**: Windows Defender come built into Windows out of the box. It may not be the best, but this is where those third-party options come into play again offering versatility and compatibility. We would also need to be able to encrypt the data that is being sent back and forth. Windows utilizes a numerous amount of encryption options for different types of data. When configured and managed correctly, it is very easy and cost-effective to use Windows.