

Draw It or Lose It

# **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 | 3/20/2022 | Daniel | <Brief description of changes in this revision> |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

With this project, the client has asked for the access of one or more teams and multiple players per team. We will use separate classes to create teams and players. We will list each team and player, and to prevent duplicate team names, we will them in array list. We will then create identifiers for the teams and players.

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

This is a web-base gaming app, allowing one instance of the game to exist. Each team name and player must be unique. There is a time limit of 30-seconds for the drawing to be guessed before the opposing team gets a chance to guess what the drawing is. The opposing team gets 15-second time limit to make one guess.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

*ProgramDriver* is where the main() method is located and is an associated with the *SingletonTester* class. The *SingletonTester* class checks to see if there is only one team name or player name exists.

The *Entity* class is the parent class to the *Player*, *Team*, *Game* classes. In the *Entity* class we will create an id in a long integer form and create a name in string form. Each id and name will be stored in the *Player* class. The *Team* class then uses the *Player* class to create teams, which is used by the *Game* class. Each class, *Player*, Team, Game, are multiplicities of each with no limits on players or teams. The *GameService* class is a multiplicity of the *Game* class, with no limit of games. The *GameService* class creates a nextGameId, nextPlayerId, and nextTeamId to allow the game to be played.

**"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 is easy to use to create software if you use Mac clients. Mac allows the use of Windows and Mac features. Lower possibility of malware. Versatile between Apple devices. Mac has fewer software and lower graphic capability. | Linux is free, open sourced, and harder to get malware. But has a low number of users compared to Mac and Windows. Due to the lack of users, there is lower amounts of applications for support. | Windows has very few software not supported. But has the possibility of malware. | Mobile Devices are mobile, meaning that you can take them almost everywhere there is a cell service. The user can have access to the game as they have service. A weakness is that the mobile device will change ip addresses and this could lead to a security breach. |
| **Client Side** | Mac is expensive to buy the Apple products. Since the MacOS is supported by all apple products, it is easy to learn to use. | Linux is free and easier to maintain. Linux may seem difficult to learn at first, but it is easy to learn. | Powerful computers are required to run Windows, which are not cheap. PCs are cheaper than Macs. Windows is easy to learn. | Mobile Devices are expensive and need a monthly cellular provider contract. Mobile devices are easy to learn and use. |
| **Development Tools** | JavaScript, SQL, Java, and C# are some relevant programming languages for Mac. Xcode is an IDE used for MacOS. This can be used to create apps for Apple mobile devices, too. | Python, C++, C, Perl, and Java are some relevant programming languages for Linux. Eclipse, Gedit, and Ubuntu are IDEs for Linux | C#, C++, Python, and Java are some relevant programming languages for Windows. Microsoft Visual Studio, PyCharm, and Eclipse are some IDE used for Windows. | Java, JavaScript, and Swift are some relevant programming languages. Android Studio is an IDE used for mobile. |

## 

## 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**: I would recommend using Windows as the operating platform, because this is the OS that I own and know the best. I like using Windows due to the amount of software that is supported for Windows. This gives us flexibility in what software we want to use.
2. **Operating Systems Architectures**: Windows is produced by Microsoft and is written in C, C++, C#, and Assembly language. My PC runs on Windows with a 64-bit operating system.
3. **Storage Management**: PCs are capable to add storage space in different forms. Solid state storage or external hard drive. With PC we can add more storage if we ever need to.
4. **Memory Management**: Windows could allocate the memory using three different forms. First fit is where the memory is allocated in the first available spot that is big enough. Best fit is where the memory is allocated to the smallest possible spot that will fit. Worst fit is when the memory is allocated to the biggest possible are. When memory is deleted, we need to defragment the remaining memory. This will help organize the memory space and allow us to efficiently use our memory space and eliminate as much dead space as possible.
5. **Distributed Systems and Networks**: Since the application is web based, we can use Java as our programming language. This will allow the application to be available cross platform. Java code gets compiled into a bytecode that can be run with any Java Virtual Machine. This will give the flexibility to use any OS that we own.
6. **Security**: Due to Windows vulnerability to malware, we will need to add in some security measures to protect our application and our end users. We can either buy or create the security. This will add to the cost or the time to the project.