

<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 | <11/12/23> | <Michael-Abell> | <Added a superclass and additional iteration> |

**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’s Draw It or Lose It is currently a web based software that allows users to be a part of teams and compete in a Win Lose or Draw style competition. It is very important that players and teams are unique and that any combination of team quantities are allowed. Finally there may only be one instance of the game running at one time.>

## Requirements

*<* A game will have the ability to have one or more teams involved. Each team will have multiple players assigned to it. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. 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)

<Anticipated design constraints are that multiple Operating systems will have to be accounted for, as well as server and networking compatibility. Having only one instance of the game will not come from hardware memory>

## [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 new class Entity has been introduced and contains the most important attributes to fulfill the clients requirements. Only one game may run at a time and teams and player names may not be duplicated. By have these stored in the Entoty parent class, the other subclasses will inherit the attributes, store the data in list, and iterate over them to ensure no duplicates are made.>

**"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 OS is very similar to Linux in it’s Unix core. Beyond that it’s a more tightly controlled ecosystem that has a small barrier of competence to overcome. Mac OS systems are typically more costly with hardware and hosting. They amount or virtualization or emulation is similar to Windows but natively much more restricted. Testing out implementation to other OS’s would require high powered CPU and RAM, extra storage space and knowledge of yet another software to run test. > | <Linux and it’s various OS’s enjoy a longstanding reputation for being an elite platform for advanced users. For anyone not in the industry and without and computer enthusiast hobbies, it’s quite possible even in 2023 that they haven’t heard of Linux, much less it's charcuterie of distros. Some of these distros with the Linux kernal are so barebones that security is simple to implement. Linux seems to draw out the more technically minded user and client so the entry point of competence is the biggest weakness. Which is like saying the most modern tank or jet fighter has a negative of needing competence…> | <Windows sets itself apart by having the most options for adaptability. There are the most amount of users, the most options for emulating other environments and the most amount of set libraries and compatibility in coding software. If it runs in another environment it can be run in Windows. One of the strengths however it’s its double sided blade. Because it is the most popular it can be more easily compromised. Just using rules of statistics the OS that attracts the most users will have the most vulnerabilities. New or uneducated coders will be quick and sloppy and pay the consequences. > | <The two more popular OS’s within mobile are Android and iOS. Both harken to Linux roots but are too independent of each other for simple adaptation. It’s essentially the Windows and Mac of the mobile world. Many business models reflect this by releasing to one environment and then once development is complete and success is attained then developing for the next environment. All platforms are constantly evolving, but the mobile OS has unique challenges in that hardware is so specialized and constantly updating. PC’s are typically motherboards and cpus, ram, storage and power. Mobile device are not as uniform and drivers and technical knowledge is needed to keep up.> |
| **Client Side** | <MAC OS will be the most costly for software development. Any software developed has minimum and recommended hardware requirements. So the machine used to make the program must supersede that of what is being programmed. Otherwise it cannot be tested. Keep in mind that the newest models of Mac OS devices will be needed as they come out so that compatibility for all users stays intact. Time should be comprible to the other non mobile OS’s, and this would be the somewhere in the middle OS to program for as it is less utilized than Windows | <Linux can run on a toaster. In fact I guarantee there are toasters, “Smart” toasters that use some kind of linux out there. The hardware barrier of cost is not a limiting factor of Linux. It takes pride on being able to use old hardware while also being able to use brand new as well. The tiume and expertise are the main considerations for Linux. En masse, software and drivers are produces for Mac OS and Windows. SO Linux typically has to adapt the code to work in their environment. And there is no Linux fairy, it’s the software developers that have to adapt the drivers and code for their own software. > | <Windows is the most prevalent OS on the market. There are so many libraries of code online that work within the Windows OS natively that none of the three issues are a problem compared to the other OS’s. Everyone has a Windows computer or works on one. None of these three a barriers to software development.> | <The main cost of keeping up with mobile OS’s are the sheer variety of device that need to be coded for. There are two main OS’s within mobile, but each one has current OS’s and previous that they must maintain for a set amount of time. It's hard to develop for the current or future dated devices when so many resources have to be devoted to maintaining older versions. This falls under more of a social contract in some places, but actual law in some parts of the world that maintenance has to continue for a set amount of time.> |
| **Development Tools** | <Xcode with developer license, Visual Studio, Emacs.> | <Visual Studio, Eclipse, Atom, KDevelop, IDLE, Emacs.> | < Visual Studio, Eclipse, PyCharm, IDLE, Notepad++, > | <Android studio, Visual Studio, Xcode, Eclipse, QT IDE |

## 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 recommend that our client pursue the iOS platform. The amount of active devices across the entire world that are active multiple times a day makes this the prime candidate. To further the recommendation, it could be argued that there are more Windows devices used a day, but my argument would be that a sizeable percentage of that would be business-use-only hardware that would not be able to download a program. Couple that with he fact that there is a small market for downloading programs that aren’t AAA rated software from major companies>
2. **Operating Systems Architectures**: <The program would be made using X-code and featuring Cocoa Touch. Cocoa provides all the framework needed to communicate to the user from the phone or iPad>
3. **Storage Management**: <For storage management we will go with the top of line: Amazon Web Services AWS. The maintenance and support is second to none and down time will never have to do with our webhost. All media can be stored and expanded for some of the best prices per reliability. With phones and iPads always connected to the internet it’s the best way to manage storage.>
4. **Memory Management**: <We’ve explained in the past how using the singleton method there can only be one instance of the software running. Adding on to the iOS utilizes ARC, Automatic Reference Counting. This paired with strong object references will keep the iOS device well within acceptable memory functionality.>
5. **Distributed Systems and Networks**: <The App Store is one of the most popular distribution programs in the world. It is maintained by Apple, just about the largest company in the world. If they can’t keep as close to 100% uptime as humanly possible, then no one else can.>
6. **Security**: <Apple is a closed system. Once the software goes out it will not be altered. AWS hosting the files is also secure. Internal security protocols are the biggest vulnerability the company faces. Invest in a good cybersecurity hire.>