

<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 | 03/14/2022 | Sean Little | Updates to executive summary, design constraints and 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)

The Gaming Room has a game currently titled “Draw It or Lose It”. The purpose of the game is for teams to be able to play against each other, a picture is pulled, and the first team has 30 seconds to guess it, if they are unsuccessful the second team will have 15 seconds to steal. At the present time this is only available through the Android App store. The Gaming Room doesn’t know how to go about getting a web-based version of the game set up. They would like the game to have the ability to have one or more teams playing, and each of those teams will have multiple players. They also have the desire for each team name to be unique so players can check what names are in use. Unique identifiers will also need to be set up so only one game exists in the memory at any given time.

## [Design Constraints](#_2et92p0)

Which operating system is going to be used is going to be a constraint, because when developing you must take into account that the company wants this to be run on different web-based platforms. Also, because there are many platforms wanted, we must make sure that we have a team comprised of people who know the multiple platforms. We would also need to consider if they are on a schedule and when they need this by and if there is any kind of specific budget for everything that they are wanting. The other thing we need to consider is that since they are not familiar with the needs to complete this are they going to be doing any of the work or is it just going to be through us.

## [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 classes of Game, Team, and Player are all subclasses of Entity, which means the inherit information from Entity. They all share common references. These references are things like name and id. Also, while they all inherit from Entity, they are also al connected. Staring from the GameService which moves over to the Game which leads us to Teams which ends us with players.

**"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** | The easiest to use out of the operating systems. While it is looked at for web hosting it is not always the most popular because of the flexibility of the system, in servers and terminal configurations. It is also upgradeable so it can offer various options. | It is popular and preferred when it comes to security. Because it is more secure most of the security issues that before it launches. It is also very handing when choosing a web-based hosting service. Although popular it can be difficult to find applications to support it on the web. | When it comes to the server side, Windows can be nice because it does use a closed platform. Next to MAC it is one of the more popular operating systems. Because it is closed platform the support you can get for it can be limited. You would need someone who handles or knows that very well to fulfill requests from the client. | From the server side you must be able to keep security high. As well as have people who understand the aspects of all the different mobile devices. This will be a biggest player because it is probably the most popular option out of the operating systems. It also gives the biggest reach because it is in the hands of almost everyone on the planet. However, because you do have so many devices it can lead to poor security because of the constant movement of networks. |
| **Client Side** | It is basically plug and play so the expertise needed on the client side is minimal to moderate. As far as cost is concerned because ti is easy to use, it can be less than some of the others, because if you don’t know it you can learn it fairly quickly. | It is not as widely known for most clients so it can be costly because you must have a high level of expertise for whoever is working on the client’s side of things. | The cost is like the other operating systems, you need someone who knows them and can make the needed requests. It is simpler because it is a closed platform, so the expertise needed is minimal. | When it comes to Mobile Devices you must consider what you are going to use and the cost because of the different operating systems. This can take multiple people on the client side because you will need to have someone that understands all the mobile devices. This could be a good thing though because it is mobile, they don’t have to be plugged in to receive updates or to even play the game. |
| **Development Tools** | Swift is the most popular when running languages, but it is not the only one that can be used. Because of its flexibility it can use all the languages like the other platforms. So, like the others you have access to the different libraries depending on what language they decide to use. | Like Windows it can run the many languages and IDEs for programming, like Eclipse and Visual Studio. Languages like JAVA, Python, C, C++ and others can be used depending on what the team decides to use and cost for the client. | Like Linux when it comes to tools but little bit simpler. Many tools available for its visual studio, eclipse. Yu can even use the notepad to write and then drop the code where you need it. You have access to the same libraries with the languages like JAVA, Python, C, C++ and so on depending on what the most cost effective one to use for the project would be, | You have lots of different options when it comes to programming tools and languages. With android using open sources. Because you have access to the different languages you have access to multiple libraries as well, including those in JAVA, Python, C, C++. Those are just some examples. |

## 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**: For ease of use, I would recommend that the Game Room start with MAC and then move to the mobile devices to get everything moving toward widespread customer base. The ease of use for both desktop and mobile devices with MAC males it the better choice for grabbing attention of new users. Also, since it does use similar languages, it would be easier to expand to the more complicated platforms.
2. **Operating Systems Architectures**: With MAC you will have access to the operating system and messaging services to be able to use the graphics and everything from it. Then you have the flexibility when you move it over to mobile to use Graphics and multimedia platforms for the different types of devices.
3. **Storage Management**: with Mac you have the cloud-based storage services to keep space on the device itself free. Then with the mobile devices you have the local storage and SD storage, and each come with its own cloud-based service as well. This is important when you consider the number of images the game is going to have in game play. MAC has that built in cloud storage that the users can keep their devices clear while
4. **Memory Management**: This will all come down to what language you decide to use, and the memory needed for the pictures that will come with the game since that is its main objective. This will be done when working with the IDE you chose to use and will allow the programs to store on the computer or device for what it needs. When it comes to the user I would suggest going with the MAC OS and iOS for development with both memory and storage. I suggest this because when it comes to game play the Mac systems have their own built in cloud server that you and the end users can use.
5. **Distributed Systems and Networks**: With starting with MAC and moving to mobile devices and platforms this makes it easy to transition, because you have MAC OS and iOS running on the same network, and then you can move to other mobile platforms. Moving cross platform on networks that share operating similarities make it easy to move dependencies around.
6. **Security**: With the MAC and Mobile Device securities they offer standard practices to secure data on the servers. Mobile Devices now come with the biometric systems that allow for secure practice and ease of access at the same time. MAC uses the MFA to get in so your data while programming can be secured. They are also ever evolving because the threatens are ever changing. With both systems you can automatically push updates for security when needed as well to make sure you are always secure.