

The Gaming Room

# **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 | 04/14/2022 | Brennan Rodgers | Updated the recommendations. |

**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 staff at The Gaming Room do not know how to set up an environment to develop a web-based version of their Android game application “Draw It or Lose It”. They want the app to serve multiple platforms. The solution is to help streamline development of the web-based application and create something that follows and exceeds their software requirements. We will also need to convert the Android program to others as a reference for how the programs will work.

## [Design Constraints](#_2et92p0)

* Ability to have one or more teams involved where each team will have multiple players assigned.
* Unique team names (Team names cannot be the same)
* One instance of the game can exist in memory at any given time
* Will need unique Identifiers for each instance
* Will need application to run on multiple platforms

## [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 Game, Team and Player classes all inherit from the Entity class. All these classes will share certain references such as name and id. The Game and Team and Player classes are also aggregated because the Game class references the Team class, and the Team class references the Player class. This helps in reusing code as it allows us to provide a reference to one class from another. The ProgramDriver class and the SingletonTester class also use each other , which allows for one game to go at once.

**"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** | Apple offers a server-based deployment method. The interface is easy to use and configuring it is also easy. There is only so much it can do as there are restrictions compared to Linux or Windows. | Wide variety of deployment methods. Provides resources, open source which is why it is free to use. Not many restrictions, with that there are not many support options as Linux is very broad. | Windows has very secure servers. It provides a simple interface, so it is easy to use. Support is widely available because windows is a very common OS. It is a proprietary software, so it does not have an open source and provide as many resources as Linux. | Not strong enough to be a fully functioning server when running a game like this, on multiple platforms. Needs to function on many kinds of mobile devices. Very cost effective |
| **Client Side** | Will take the user some time to get used to. Since it is not open source, someone with experience would need to navigate the system. This is an expensive option but will take quicker to develop with easier use of the interface on the server side. | Will need someone with lots of experience to navigate. Since it is open source, there is lots of resources, which means there are lots of options on what you want the software to do. Though it is a cheap option, it will take longer to complete as Linux is very complex. | For someone with experience with windows, it would not take very long. It is on the expensive side, but there are a lot more support options as it is a well-known operating platform. | Considering mobile devices are easier to handle, you would not need much experience. It would be a cheaper option but would need to develop for multiple platforms as there are multiple kinds of mobile devices. That would mean a longer wait time. |
| **Development Tools** | IDEs that are used for Mac systems are Swift, Javascript, HTML, Python and C. Tools would include Eclipse, Visual Studio, Xcode or Atom. | Linux mainly uses Java or C. Eclipse is a very popular tool for those two languages. | There is a wide variety of languages for Windows. Those include Python, C languages, HTML and Javascript. Visual studio is a widely used Windows tool, and eclipse can also be used. | Depending on the device. For androids, a language like Java would be helpful. For IOS you could use Xcode and swift to create an app. Can also use languages like Python or HTML |

## 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**: After reviewing all the different operating platforms, I believe that Windows would be the best option for “The Game Room” to use. Though it is a bit pricy, it is worth it in the end with the amount of production you can get out of it. It offers a lot of development environments (IDEs) and great support options. Because Windows is such a common operating platform, there is no shortage of experts who can work easily on this platform.
2. **Operating Systems Architectures**: The Windows architecture is made up of all windows made apps that connect directly with the graphical user interface, which allows access to memory, storage and other processes that will have no effect directly with the program.
3. **Storage Management**: Windows comes with great storage management options already built into the platform to where you can manage any hard drive that is hooked up to the computer and currently running Windows. With something like Disk Management, which is a tool built into Windows, you can maintain and keep track of how storage is utilized and see what tasks the drive is performing.
4. **Memory Management**: When creating the game, there will need to be an included library with pictures that will appear for the game. With memory management, those pictures will be stored in a folder and referenced in the software when it is time for those pictures to pop up.
5. **Distributed Systems and Networks**: With this game, a client-server system will have to be implemented for it to work to the best of its ability. When the client and user interact with the game, it will be constant communication to the server which will then provide feedback to the user, so on and so forth. With the necessity of multiple players playing at once, there will need to be strong and secure connections to and from the server.
6. **Security**: Providing multiple layers to the security, things such as two factor authorization for anyone accessing the software from the server side and encrypting data that is sent to and from the server. This would allow for safely distributing data of the users to the server so it can be stored.