

<Name-of-Software-Application>

# **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 | 3/31/24 | Austin Niwa | Filled in data on the cover page, summary, requirements, constraints, architecture, model, evaluation, and recommendation |

**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 client wants an app that works on multiple platforms based on their mobile app draw it or lose it. It is loosely based on the 1980’s television game win lose or draw where teams have to compete and guess what the user is drawing on the board.

## [Design Constraints](#_2et92p0)

The constraints given are as follows:

Game has one or more teams involved

Multiple players on one team

Game and team names must be unique so users can check whether a name is in use

Only one instance of the game can be up at a time

## [Domain Model](#_8h2ehzxfam4o)

From the diagram I can see there are 7 classes where the team game and player classes are referenced by the entity class that is inheriting functions from each of them. The gameService class is associated with these 3 classes as they are linked together by the players making up the teams which are added to the game and all have names. The game service class has our singleton method and iterators for the getGame getId functions and references the game class due to the line connecting them then the program driver and singleton tester test the program to make sure it works right.

**"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** | Macs are relatively easy to use compared to other options, flexible, have access to mac applications, and easy to configure, though they are not great if you’re looking for something cheap | Linux is cheap used by many big name companies like google for example, it has command shell for server configuration as well going for it. It is also the most secure of the options as companies can change their security as they see fit. | Windows is very user friendly, expensive due to licensing, and also has a command prompt like linux. It has the best out of the box support OS applications though and patching it is easy. | Parts are not as powerful as a standard computer and specifications vary wildly for each user. I haven’t seen this utilized but it can be done. |
| **Client Side** | Has a higher upfront cost, has access to the mac ecosystem by default and easy to use. | Theres a lot of choices for distros as linux is open source so many people make their own modifications, it’s the most secure, and there are many cheap options for a machine. Unfortunately, it might not be compatible with every app. | Easy to learn and get set up but, more expensive than a linux system, don’t need to know a whole lot to user windows on the client side. Less secure a lot of variety in systems. | Everyone has a phone but their hardware is so varied that it may be difficult to account for all of them |
| **Development Tools** | Typical development tools include CSS, HTML, and JavaScript and for IDEs PyCharm, Visual Studio, and Github are often used. | HTML, CSS, and JavaScript are also used here as are Ruby PHP and Python C, C++, with the IDES spacemacs, brackets, codelite, visual studio, and VIM | HTML, CSS, and JavaScript are also used. C++, C, Java as well with the IDES Eclipse, PyCharm, Jetbrains, and visual studio. | Xamarin, xcode, and android studio are good mobile IDES. They support swift, C#,C++, objC,Kotlin, and Java. |

## 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 is the best operating system to use for this project as it is the most used operating system in the world so it is more likely that employees will have knowledge on how to use the operating system. Using windows for the project also allows us to natively test in windows since the players of the game will likely be using it gives us the ability to test for the biggest pc audience.
2. **Operating Systems Architectures**: the Windows operating systems architecture allows for the the program to integrate with the os services allowing the program to make use of virtual memory and other components in the pc. Programs can also use that integration to interact with the system kernel or the drivers of the operating system as needed.
3. **Storage Management**: Amazon web services are incredibly secure and fast. I would recommend them for this application as we can expand our storage based on popularity of the game if it gets popular. We would also not have to worry to much about the storage of the user in this case as they would only be downloading cookies at most which take up next to no space. The Windows operating system also has disk management built into it with the disk clean up tool.
4. **Memory Management**: We are using Java as the main language for this app. Java manages memory on its own so it is not necessary to include any type of memory management. We would also be making use of virtual memory which would allow us to help users with less ram and also cache files on the hard drive to be used in the near future in the game. As for our operating system Windows already has memory management covered in the operating system itself so we do not need to be concerned with that either
5. **Distributed Systems and Networks**: We will have to use a client-server system here since every client will be a different game application and will have multiple people playing on one game. We will have a great server for the game as were will be using Amazon Web Services and can upgrade or downgrade our servers as needed.
6. **Security**: Security is great on windows nowadays with windows defender being installed on every windows device. It’s gotten much better than it used to be and is great for everyday usage. Amazon web services will also keep our data secure as they have the most secure cloud infrastructure on the market and are constantly innovating new ways to keep themselves at the top of the game as far as security goes.

REFERENCES:

<https://www.redswitches.com/blog/best-linux-ide/>

<https://www.igeeksblog.com/best-mac-ide/>

<https://www.codecademy.com/resources/blog/popular-ides-and-code-editors/>

<https://www.zdnet.com/article/windows-mac-or-linux-we-compare-the-pros-and-cons-of-these-computing-platforms/>

<https://www.geeksforgeeks.org/architecture-of-linux-operating-system/>

<https://aws.amazon.com/products/storage/>

<https://finance.yahoo.com/news/15-biggest-companies-aws-011011152.html>

<https://learn.microsoft.com/en-us/training/modules/explore-windows-architecture/3-examine-windows-client-architecture>

<https://www.codecademy.com/resources/blog/virtual-memory/>

https://aws.amazon.com/security/