

<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 | 11/13/22 | Chris Fant | Filling in the parts that needed to be filled in |

**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 is wanting help to streamline the design process for a drawing game. They are wanting to adapt their game in a web-based format. The company wants the ability to have one or more teams in a single game, each team to have multiple players, game, and team names to be unique and having only one instance of a game allowed to exist. To create this software, we will be creating various classes to work together to form the game. To make sure there is only one instance of a game we will use a singleton pattern and use validation to make sure there is no repeat names.

## [Design Constraints](#_2et92p0)

When developing a web-based application on any kind, that limits the languages you can use. You need a language that can be used on all platforms. Something else to think about is creating something that will be able to combat slow load times. Also, since this is on the web you want to consider the security measures being used.

## [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)

There are various classes in this diagram that work together to make a good program.

First there is a parent class of Entity that utilizes inheritance to share its functions and objects with the child classes. Also, you could have no GameService but have many Games and have no Game but have many Teams and the same with Team and Player. This shows the association between the different classes. Finally, you have a ProgramDriver where your main function resides to allow the program to run, and it uses a SingletonTester classes to ensure that you only have one instance of GameService at a time.

**"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 works really well with other Mac clients.  It has an ease of use for feature creation. Mac is only able to run on select computers. | Linux is safe and secure. It is able to be run for years without having any issues. | Windows uses user files on the server to identify people. This makes it easier to put their data together. | There are a lot of hoops to jump through to develop on this end. Finding someone to support your mobile app is just one thing. |
| **Client Side** | Macs usually cost the consumer more and the only machines that run Mac are Apple products. | Linux is the most common of these operating systems. There is not much cost to it as the software is open source for the most part. The only cost would be buying the machine that is compatible. | Windows is great for web site development. Using Microsoft’s own developing app Visual Studio helps streamline a lot of the work. This is true especially when developing windows apps. | Typically, with designing a mobile app you are working with limited space. You will usually have to rearrange the controls to fit the smaller screen for the user. |
| **Development Tools** | Mac uses Objective C mostly but uses C and C++ in various levels. Apple also created Swift to allow for more creative freedom form their developers. | Linux is pretty much an open book. It supports every language there is. If you want to use it, just start writing it. | Windows has the ability to support most languages. It mostly uses C++ for all of its coding and development of apps. | Depending on the type of phone you’re developing for determines the language. For iPhone’s Swift is used and for android it is typically 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**: I would recommend Linux as it is safe and secure. Linux is also accepted with almost any computer you see. This would make the app adaptable and able to be put on any computer.
2. **Operating Systems Architectures**: The Linux system has 5 parts to its architecture. The Kernel, that takes care of the primary duties of the operation system. The System Library, which is a collection of functions that are used by the operating system. The System, which utilizes utility programs to perform tasks throughout the system. The hardware, which is self-explanatory, but it is the components of the computer. Finally, we have the shell, which is typically an interface between the user and the kernel.
3. **Storage Management**: Using SSD storage would be the best as it has quicker access to the things stored in them. This would allow for quicker processing of the images and user interaction.
4. **Memory Management**: Linux will use paging to allow for seamless play of the game. This will keep all of the memory from fragmenting and keeps the program from moving slow due to compacting.
5. **Distributed Systems and Networks**: Having a network database to store all of the user information for future uses keeps them from having to reenter any information. This also allows them to continue any progress made without taking too much room on their system.
6. **Security**: To keep information secure, we can use anonymity and pseudonymization for maximum safety of users personal data.