

CS230 Project One Game App

# **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 | 03/24/2024 | Bee Best | Added Exec. Summary, Requirements, Constraints, System Architecture View, and Domain Model. |
| 1.1 | 04/07/2024 | Bee Best | Added Server Side, Client Side, Dev. Tools in Evaluation. |
| 1.2 | 4/21/2024 | Bee Best | Added Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to use our services to streamline the development of their web based game “Win, Lose or Draw” and allow for the game to be available across multiple platforms. The game is currently available as an Android app.

## Requirements

The client has requirements as follows, the game must have the ability to have one or more teams involved, each team must have multiple players assigned to it, the game and team names must be unique to as to allow users to check if a name is in use while they are picking their name, only one instance of the game can exist in memory at any one time (this is expected to be accomplished by creating unique identifies for each instance of a game, team, or player).

## [Design Constraints](#_2et92p0)

There are a few design constraints to consider, these being how scalable the game should be, the security aspects of it, and most importantly to the user, cross platform compatibility. Most of these only require monitoring our development and ensuring they are implemented along the way (security and scalability), cross-play will require us to ensure every system that can run this game can display it properly and communicate with every other player accordingly (usually the latter is managed by the browser being used but mobile device support may need additional effort). Another constraint could be that their current app is written for android but we do not know if it was made with future expansion in mind, as such it could cause delays as we would need to give them the necessary requirements for changes they would have to make on their side so that it all works seamlessly. All issues presented are technical in nature although a business issue may appear in the future should it take longer than expected to make support for all the types of devices that our client wishes.

## [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 UML model shows that the ProgramDriver uses the SingletonTester to do just that as seen in the function testSingleton(), these functions are separate from the rest of the class structure and only serve to run the main method which in turn runs a test function. The UML shows that Game, Team, and Player all inherit from Entity and that GameService is associated with Game via a one to many connection, Game is associated with Team with one to many, and Team is associated to Player with one to many. Lastly it shows each class and their methods these showing if they are public or private as well as the methods return types, inputs, and if anything is static or not.

**"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** | Advantages:  Mac has a high amount of documentation and technical support. Mac is also relatively secure.  Weaknesses:  Entirely controlled by Apple, Apple tends to be quite expensive which would impact the cost of a license. Further, the entire server system must be running on Mac due to how closed off of an OS it is. | Advantages:  Open source, easy to update, extremely strong security due to its open source nature.  Weaknesses:  Not many applications support Linux | Advantages:  Almost every application supports Windows. Server management is relatively simple. Windows also has a high amount of documentation to support developers.  Weaknesses:  Users (and developers) are not as free to do as they please, entirely controlled by Microsoft. | Advantages:  On Apple devices, there is nearly no way for users to cause problems. |
| **Client Side** | May require hiring those with expertise to make sure it runs well. A medium amount of time necessary to make it run optimally. | Linux is much more complicated than the other two presented here as such it will require more expertise and more time to develop. | Most people know how to do stuff on Windows, as such, the least time is needed for this (and the least amount of additional expertise is needed). | Expertise is required although the design would be easy to implement due to the fact that an android app already exists. |
| **Development Tools** | Xcode and Swift are used frequently with Mac. | Python and Javascript are commonly used for Linux as well as VSCode and Eclipse. | C#, HTML, and Javascript are commonly used for Windows (personally I like Javascript and HTML). Visual Studio can be used here. | As with Mac, because we are looking at an IOS app (Apple still), it would use Xcode and Swift as well. |

## 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**: The platform I’d recommend such that it can be further expanded in the future is Windows, it is by far the easiest to develop on due to the large amount of documentation and tools available for use with it. Also generally, the order in which expansion should be done when it comes to platforms is heavily based on how many users are on said platforms, with Windows having such an expansive userbase it’s obvious what decision should be made.
2. **Operating Systems Architectures**: Windows is generally a very user friendly OS due to its simplifications, mainly that there is the split between what the user has access to and what the computer itself is doing (with the kernel doing many background tasks that the user doesn’t need or shouldn’t need to worry about). Windows hardware can also be customized greatly when compared to other systems.
3. **Storage Management**: When it comes to storage I would recommend cloud storage, primarily because cloud storage would allow for much simple scalability and because the potential security risks and inconsistencies are outweighed by how much faster and simpler it can be deployed and managed.
4. **Memory Management**: Windows is generally efficient with its memory management (like every other major OS), primarily because of VM(virtual memory) and memory paging, these essentially allow the OS to not as easily run out of space by allowing some of the actual storage to be used as memory. On a larger scale server system this added memory could ensure that the system will always run smoothly baring unexpected circumstances.
5. **Distributed Systems and Networks**: Because the game itself will be browser based instead of being based on an external program, ensuring that it can run on at least a few of the major browsers will ensure that the majority of the population can play it. These browsers being Chrome, Opera, Firefox, and Microsoft Edge (but let’s be real, I doubt anyone playing this would play it on Edge).
6. **Security**: Depending on which specific provider for cloud storage we use, the security risks could be lower than if we were to handle the data ourselves. Google for example is known for having an extremely secure system (If I remember right don’t they hire people that can breach their systems? Or was that like a one time thing). Regardless, cloud storage is safe and secure enough such that we do not need to worry about it through the right provider.