

<Win, Lose or Draw>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.1 | 10/15/2023 | Coltin Meyer | Revised each section to better represent the desired outcome |
| 1.0 | 09/14/23 | Coltin Meyer | Finalized and approved each line paragraph |

**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](#_heading=h.35nkun2)

Currenly the application is not web-based and needs to be. The code needs to be re-created to correctly work with all web applications such as google, bing, microsoft edge. It must also work with MacOS, Windows, and Linux.

## Requirements

Win, Lose or Draw needs to have a web-based verson of the app with a unique environment. The app will need to have one or more teams (no team limit specified), multiple players able to be assigned to each team (no player limit specified), the ability to name each separate team (no character limit or prohibited phrases), and only one instance of each game can exist in memory at any given time.

## [Design Constraints](#_heading=h.1ksv4uv)

Needs one or more teams to play, One or more people need to be on each team, Game can only run in a single instance, must run on various platforms, Team and game names need to be unique. The app currently is only on android and needs to be able to work on other mobile platforms as well as windows, linux, and apple.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

The ProgramDriver is associated with the SingletonTester to use its attributes. Entity is a generalized superclass for each subclass below. Game, Team, and Player all share the attributes from Entity and can all use 0 to many attributes of the other classes. Game service is associated with game so they share attributes, however GameService is not a subclass of Entity and is only associated with game which is associated with Entity.

**"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](#_heading=h.z337ya)

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 is fairly flexible with its platform having terminal commands available to configure or make changes but it also preferred for web-hosting although it is not popular. | Linux is an affordable and secure platform while also being very popular for web-hosting because of its security. The main downside is that the software is complex for webhosting. | Windows is the most comfortable and fast system. It has great resource needs, however is susceptible to viruses and has low tech support. | A server on a mobile devise is not a great Idea but its popular and portable. It makes the game cost efficient and has a wider range however its not very secure and every mobile device platform is different. |
| **Client Side** | Costs about as much as windows and requires some time to learn. Needs to run as smoothly as the other platforms and web browsers including mobile devices. | Requires a lot of time and education to learn linux but its cheap. Needs to run as smoothly as the other platforms and web browsers including mobile devices. | Doesnt requires much time or education but costs about as much as mac.  Needs to run as smoothly as the other platforms and web browsers including mobile devices. | Cheapest option by far, allows clients to view updates on the go however is not as easy to use as other devices due to processing power on mobile phones. |
| **Development Tools** | A well known tool to use on mac is Swift to run languages such as HTML, CSS, and javascript. More are included such as python and java. | Linux makes use best of Notepad++ and eclipse. There is most programing languages and additional tools such as HTML, CSS JavaScript, Python, Java… | Windows uses Eclips and visual stuidio, can use notepad++, By far the easiest to use but can also us HTML, CSS, JavaScript, Python, Java… | Android and swift allow the ability to make many apps for mobile. With them you can run HTML, CSS, JavaScript, Python, 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 suggest The Gaming Room to utilize the windows operating system because of its wide range of applications and ease of use. Its also wont need much training and wont need much money. It would be nice to utilized a cloud-based server platform for flexibility, the ability to grow, and its a low cost option.
2. **Operating Systems Architectures**: Windows has many of its own applications and allows you to view its processes with the GUI to see its performance. It has many resources for communication and multimedia. It also utilizes a hybrid architecture which will help prevent system crashes. The best option would be a multi-tier architecture (presentation layer, application layer, and a data layer) which allows separation for data management, user interface, and the application logic into different layers. It allows for more security, flexibility, and room to grow.
3. **Storage Management**: Windows 11 has its own storage options to assist cleaning up the storage with storage sense however also offers cleanup recommendations to assists with keeping memory usage low. In addition the best storage management would be RDBMS (relational database management system). They allow the ability to store and retrieve data in a secure efficient way with options for data backup and recovery.
4. **Memory Management**: As described in the clients requests the game needs to have a large picture database. Windows RAM allocation allows ease with saving photos outside of the photos gallery that way not to harm pictures on the system. It would be recommended to utilize a virtual memory management to allow more memory than physically available to the user. It improves the applications performance and helps to optimize memory usage.
5. **Distributed Systems and Networks**: It is recommended to communicate between the different platforms using a microservices architecture with an API gateway. This allows for the application to be broken down into independent services that can work together. The API Gateway works and a point of entry for all requests. It makes it easier for authentication, security, and load balancing. Its also recommended that the network that connects the devices needs to be secure an reliable with backup options in case of outages.
6. **Security**: Windows has built-in security software but if its not up to par then there are many downloadable options to utilize. Personal user data is usually protected due to the frequent virus and security threat scans. It is recommended to utilize data encryption, secure coding practices, and input validation.