

The Gaming Room

# **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 | 03/17/23> | Crystal Berkhan | The Game Room would like to develop a web-based program that is available on multiple platforms. |

**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, The Gaming Room, is seeking the development of a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, that is currently only available on Android. The new game will allow for teams with multiple players assigned to it and multiple teams involved, game and team names are unique, and only one instance of the game can exist in memory at any given time.

## Requirements

The requirements of this project include: the game runs on multiple platforms based on the current game, the game will have the ability to have one or more teams involved, each team will have multiple players assigned to it, uniqueness of game and team names, and only one instance of the game can exist in memory at one time.

## [Design Constraints](#_2et92p0)

Design constraints for developing the game application include: the game runs on multiple platforms, uniqueness of game and team names, and only one instance of the game can exist in memory at one time. A major design constraint is the game running on multiple platforms. This can lead to a longer and slower development, requires more than average maintenance in the future, and a large number of resources. The idea of having unique team and game names may also extend development time for the additional code to be inputted to ensure this is the outcome of the program.

## [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)

In the UML diagram below, it provides a clear visual of the relationship between all of the classes. The Game, Team, and Player class all inherit information from the Entity class. GameService, Team, Game, and Player classes are all in reference to one another.

**"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 is not designed for gaming and provides a very limited experience for users, it Is easy to use, and has flexible terminal commands. | Linux is not recommended for beginners due to its complexity, software lacks compatibility, and gaming experience is not as high quality as Windows. | Windows can be expensive but is extremely user friendly, great for running games without issue, and games are typically developed for Windows by default. | Mobile devices are fast, efficient, and easy to transport. Different devices run games and programs differently. |
| **Client Side** | Mac is expensive for users; users need to take time to become familiar with navigating the system. | Linux is a very complicated platform to use and is not recommended to beginners due to the knowledge required to use it efficiently. | Windows can be quite costly, but very easy and straightforward to navigate. | Creating web-based games on mobile can be costly with required updates throughout all devices, specifically Android and iPhone to maintain a functional game |
| **Development Tools** | HTML, CSS, C++, and JavaScript are languages used for development. | HTML, CSS, C++, and JavaScript are languages used for development. | HTML, CSS, C++, and JavaScript are languages used for development. | HTML, CSS, and JavaScript are languages used for development. |

## 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 option for all computing environments to allow for

The Gaming Room to expand Draw It or Lose It.

1. **Operating Systems Architectures**: Windows is very easy to use, allows for the storage and efficient navigation of files, run games, and access the internet.
2. **Storage Management**: Windows already has built-in features to manage storage, such as disk cleanup and storage sense to rid the system of unnecessary files.
3. **Memory Management**: Windows also already has a feature built in for memory management to allow for the proper utilization of space. Windows has its own virtual address space for each 32-bit process and each thread does not have access to one another to prevent damage.
4. **Distributed Systems and Networks**: Users can access the same network with the use of a client-server distributing system to allow for the dependency of a single server application for the game to allow for players to create teams and interact with others and communicate.
5. **Security**: Windows has built in security management; however, to ensure complete privacy of users, another type of security is recommended.