

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 | <11/12/23> | <Justin Farquhar> | Changes made to the cover page, document revision history, the executive summary, design constraints, the system architecture view, the domain model, and recommendations. |
| 1.1 | <12/07/2023> | <Justin Farquhar> | Changes to Operating Platform, Operating Systems Architectures, Storage Management, Memory Management, Distributed Systems and Networks, and Security. |

**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 project’s goal is to develop a game application serving multiple platforms based on the game Draw it or Lose it. The application will render images as clues, consisting of four rounds of play each lasting one minute each.

## [Design Constraints](#_2et92p0)

* Must be able to run on multiple platforms
* Only one instance of a game can exist at a time
* Each team has multiple players
* Game and team names must be unique

## [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)

Entity has a relationship between the Game, Team, and Player classes, allowing each to inherit information from the Entity class. GameService, Game, Team, and Player are all related in the sense of each has a multiple of the other. GameService has multiple Games, Game has multiple Teams, and Teams has multiple Players. The ProgramDrive uses the SingletonTester to test the instances to make sure there are no duplicate instances.

**"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** | Easy accessibility and server configuration. Easy graphical UI. | Cheap. Has a command shell for simple server configuration and accessibility. Can be difficult to navigate. | Expensive. Has good GUI. Ability to user command prompt. | Wide variety of mobile devices. Limited processing power along with adjusting for older/newer devices. |
| **Client Side** | Can be expensive for users. Can be difficult to use for newer users. Extra skills needed to navigate. | Extensive expertise and time required to learn. Can be cheap if the knowledge is there. | Very intuitive to learn. Relatively cheap. Mostly simple implementation with wider spread OS updates. | Wide flexibility for clients. Difficult to implement due to multitude of devices. |
| **Development Tools** | Frontend of HTML, CSS, and JavaScript.  Other tools of PyCharm, Visual Studio. | Frontend of HTML, CSS, and JavaScript. Other tools of JavaScript, Ruby, Python. | Frontend of HTML, CSS, and JavaScript. Other tools of Eclipse, PyCharm, Visual Studio. | Frontend of HTML, CSS, and JavaScript.  Most development is done on other devices. |

## 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 will allow for the best expansion to other devices. It allows for a good starting point and can adjust from there. Along with the simplicity of the user interface for users. I believe Mac was mentioned previously, and can be implemented just as easily, will just have a larger upfront cost.
2. **Operating Systems Architectures**: The architecture is very straightforward and easy to use and learn. Allows for an intuitive ability to run just about anything you would want to do. Worst case there are plenty of tutorials and step by step guides to getting things going.
3. **Storage Management**: Decisions will need to be made on where the game is to be stored. Some of the files will be better to be located locally while some could be stored within the cloud. Windows allows for efficient management of storage through the file explorer. There has also been plenty of updates to allow for efficient cleaning of unnecessary files or other data.
4. **Memory Management**: Decisions will need to be made to determine which information is pertinent to keep game to game. Whether there are high scores or additional information that is worth keeping, not everything can remain to avoid unnecessarily overcrowding a user’s device. The Windows kernel-mode memory manager component manages physical memory for the operating system. It has access to RAM, random access memory, along with other drives installed.
5. **Distributed Systems and Networks**: Network based multi user interaction systems typically include a database that is shared among each player that interacts with each other and the network. Implementation of this shared database and player communications will need to be created from scratch.
6. **Security**: Consideration should be made for potential hacks/exploits along with players seeking to gain personal information on other players. Windows has built-in security protection software along with a built in learning anti-virus. In addition, encryption could be considered. Having players sign up for an account will be necessary, along with only needing the bare minimum of required information to avoid bloat and have more for potential data breaches to lose.