

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 |
| --- | --- | --- | --- |
| 2.0 | 10/0]17/24 | Kylee Ranck | A few changes were made to the recommendations section. |

**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 was developed to serve multiple platforms, loosely guided off a current game called, Draw It or Lose It. Draw It or Lose It is only available on android. The purpose of the game is to have minute long rounds with teams consisting of multiple players through four rounds. A team will make guesses when an image is pulled from the library until the time runs out. If the current team cannot guess correctly, then the other teams have a fifteen second rebuttal chance.

## Requirements

Review Executive Summary

## [Design Constraints](#_2et92p0)

1. Needs to run over multiple platforms
2. A team consists of multiple players
3. One instance of the game can exist at any time
4. There must be unique team/game names so they are not duplicated or copied from elsewhere

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

Entity flows to Game, Team and Player classes. Those three classes receive information form Entity. This is shown as an inheritance through UML which results in Entity as a superclass. Team and Player include “has a” type while Game “has a” Team and GameService correlates with Games. This is known as aggregation (HAS-A) in UML. In other words, a “has a” has an instance of another class. GameService has a reference of Games, Games has a reference of Teams and Team has a reference of Player.

**"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  \*Easy server configuration  \*Easy graphical user interface  \*Disadvantage: less preferred for web hosting services | \*Difficult to navigate  \*Cost friendly  \*Has a command shell for simpler accessibility  \*Able to catch security flaws before they grow | \*Has a command prompt  \*Server can be more expensive  \*Higher demand over other platforms  \*Disadvantage is that sensitive to viruses | \*Specifics work better in some devices vs others  \*Each carrier has own specifics  \*Is cost effective, better compatibility overall |
| **Client Side** | \*Need to have knowledge to navigate operating system  \*Expensive for users  \*Time and semi moderate expertise needed | \*Cost heavily for Linux users  \*Needs Linux data to operate  \*Needs to be really familiar with system as only has a few applications available. | \*Easy to learn and understand  \*Cost more than Linux  \*Less loading period | \*Allows updates to be tracked openly  \*More difficult to use than other devices  \*Cost effective  \*Has a lot of technical support available |
| **Development Tools** | \*Languages: JavaScript, HTML and CSS  \*Development tools: PyCharm, Visual Studios  \*Easy frontend development | \*Languages: JavaScript, HTML and CSS  \*Development tools: Python and Ruby  \*Easy frontend development | \*Languages: JavaScript, HTML and CSS  \*Development tools: Eclipse and PyCharm  \*Easy frontend development | \*Languages: JavaScript, HTML and CSS  \*Development tools: C++ and Python  \*Easy frontend 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**: The platform I would choose for the Gaming Room is Windows. This is the option I would choose as it has enough IDEs to use, and it is easily accessible to any user and easy to comprehend. The cost is low and affordable by many.
2. **Operating Systems Architectures**: Windows is an operating system that is graphical and publishes by Microsoft. It allows users to connect to the internet, play games, watch videos and run/store software. The major benefit to this system is that a user can make changes to the memory, and it will not change the way the platform performs.
3. **Storage Management**: Storage sense is a feature of Windows 10 that is very useful in monitoring the amount of space the files are taking up. The system has a built-in disk management as well. One benefit of the disk management is that it can be broken down into different components. Two of the branches of the disk management are disk cleanup and storage sense. Storage sense can maintain the storage while the disk cleanup removes files no longer in use.
4. **Memory Management**: Windows 10 lets the user store their files together in one spot and it can aid in managing the games photos. A database would aid the application by giving space to store all the photos.
5. **Distributed Systems and Networks**: Sharing interactions through a multifaced system are aided using databases shared amongst players that are not in the same physical space. The sever that needs to be able to compute and share one game amongst a variety of players at the same, needs to be strong and developed for each system.
6. **Security**: Windows contains imbedded security protection software such as windows defender. Data encryption is impertinent to protect all data that is inputted and transferred to other outside sources.