

The Game 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 | 08/10/24 | Caleb Ehrich | Recommendations for new application |

**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 Game Room wants to build a web-based gaming experience that can be used on multiple platforms. Currently, The Game Room has the game “Draw it or Lose it” available on Android. In “Draw it or Lost it” there are multiple teams with multiple players on each team. There are four rounds in the game. In each round the teams have a chance to guess the image and if they fail the other team is able to guess to steal the round.

## Requirements

1. Game needs to support multiple teams.
2. Game needs to support multiple players on each team.
3. Game and Team names must be unique.
4. Only one instance of each game can exist in memory at any given time.
5. Must be compatible with multiple platforms

## [Design Constraints](#_2et92p0)

While designing and developing the new application we must adhere to the requirements listed above. When making decisions we must make choices that will allow the application to function and look the same on each platform. We will either need to create the application in multiple ways for each platform or we will need to explore options that we can reuse across all platforms.

## [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 creates the relationship between Team, Player and Game. This means we get all the data from Entity. We can see this in through inheritance on the UML diagram. When we look at the relationships between GameService, Game, Team and Player, we see a “has a“ type. This means, GameService has a Game, Game has a Team(s), and Team has a Player(s).

**"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** | Flexible terminal commands to help with configuring the server. Need access to make needed updates/changes. | Same as Mac, but would be more cost effective | Has the ability to run more software compared to other systems. | Compared to other devices, some specifications are better, but having an immobile server that can be tracked would be best. |
| **Client Side** | Similar cost to setting up Windows but would take more time and technical expertise to be able to set up properly. | Set up cost would be lower but would take extensive time and expertise to set up. | Similar cost to Mac, but level of time and expertise to set up would be minimal. | Would be the most flexible. Updates can be sent/seen anywhere. Would take a moderate amount of time and expertise. |
| **Development Tools** | Mac can run most popular development options. Mac supports the programs and tools needed to develop and run using languages like JavaScript, HTML, CSS, Ruby, Python, Java etc. | Linux does not work well with many of the common development tools like visual studio, eclipse, etc. There are other tools or programs developers could use but may not be as familiar with them. | Windows is like Mac. It supports all the popular development tools and programs and support the same languages as Mac. | There are many mobile options that can support the same things as Windows or Mac and could be supported in the other three platforms. |

## 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**: After comparing each platform, I believe the best option would be Windows. Windows supports all the needed software tools and languages. It requires less expertise, making the learning curve much easier. Windows would be cost effective.
2. **Operating Systems Architectures**: Windows is a robust platform that will be able to support the Graphical User Interface. The user interface will have access to all systems resources. With the Windows based apps and built-in functionality, the application will be able to support messaging, web services, graphics and other multimedia. Windows is flexible enough that we can offer services based on specific servers or user accounts.
3. **Storage Management**: Windows offers many nice features to help manage storage. Using features like storage sense, we can manage the hard drive and control how much space is consumed. Using cloud features, we can also throttle up the memory, storage, performance specs based on the amount of usage at any given time.
4. **Memory Management**: Considering the game will use and need quick access to pictures and other multimedia items we will need to have a database to store everything in. From that database, we can use caches and other Windows tools to allocate the memory usage to allow users to have quick access to the needed files.
5. **Distributed Systems and Networks**: I did some research and I found multiple platforms that can help support cross-platform game publishing. With some of these platforms we can also allow cross-play to allow players of different platforms to play together. Windows supports many of these tools.
6. **Security**: Windows offers built-in security protection software. We can use Windows services to authenticate and authorize users. The built-in security protection will scan and monitor the server for malware, security threats, and viruses.