

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 | 09/13/2023 | Justin Brown | Executive summary, design constraints and domain model sections are filled as per the rubric. |
| 2.0 | 09/27/2023 | Justin Brown | Evaluation table filled as per the rubric. |
| 3.0 | 10/10/2023 | Justin Brown | Recommendation section filled as per the rubric. |

**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)

Our client, The Gaming Room, has asked us to develop a web-based version of one of their already existing games – Draw It or Lose It – that is only available on Android devices. Our objective is to create a web-based version of the game that is identical to the one that already exists on Android devices and to ensure only one instance of the game can exist at a time as well as checking for identical team names.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

1. Team names and game ids must be unique.
2. It must run the same on each of these platforms to ensure a quality game for players.
3. Database must be established in order to hold memory for team names and unique game ids.

## [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)

* The ProgramDriver class is used to run the game.
* GameService class is used to handle the games and teams and will also use the singleton pattern so only one instance of GameService can run at a time.
* The Game, Player, and Team classes all inherit from the Entity class and contain information pertaining to the list of games, the players, and the teams respectively.
* Both the Team and Game classes check for unique names

UML diagrams are efficient at laying out a foundational blueprint for software and to ensure all of the client’s requirements are met to help develop an efficient program. Composition and polymorphism are present to show the relationship between classes and how they work together with 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** | Both Mac and Windows server-side upkeep can be rather expensive when compared to other platforms but is easier to use as a tradeoff. | Much more cost friendly than Mac and Windows, however it is not as user friendly and more difficult to navigate for someone not experienced. | Both Mac and Windows server-side upkeep can be rather expensive when compared to other platforms but is easier to use as a tradeoff. | Mobile devices have a lot more variety and different carriers, which may lead to complications in trying to support most/all mobile devices. |
| **Client Side** | Mac is typically more expensive than its other counterparts – especially Linux – but in return is more user friendly and is more time efficient. | Cheaper option, but is more niche than its Mac and Windows counterparts, and as a result more time is needed to adjust and familiarize with Linux. | Windows is typically more expensive than its other counterparts – especially Linux – but in return is more user friendly and is more time efficient. | Mobile devices have the most variables when it comes to developing software that can be supported, but programs designed for other platforms may not run as well on mobile devices. |
| **Development Tools** | Languages that are commonly used for Mac systems are JavaScript, C++, HTML, Java, Python, and many other commonly used languages. IDEs include PyCharm, Eclipse, Visual Studio, and other commonly used IDEs. | C and C++ are the most popular languages to build software for Linux systems, as well as Python, Java, JavaScript, and HTML.  IDEs mostly used are Visual Studio and Eclipse. | Languages that are commonly used for Windows systems are also found on Mac systems, such as JavaScript, C++, HTML, Java, Python, and many other commonly used languages. IDEs include PyCharm, Eclipse, Visual Studio, and other commonly used IDEs. | JavaScript is mostly known for building mobile apps and its ease of use for doing so, but C#, Python, and Java are also capable of doing so.  Visual Studio is most likely the best IDE for this task. |

## 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**: Between Mac and Windows, the latter is slightly better in my opinion. Windows OS supports numerous server roles, and in addition has more ease of use which adds to its reliability. There are also plenty of IDES available to use on Windows and will generally have low cost for upkeep.
2. **Operating Systems Architectures**: With Windows, applications can have access to memory systems as well as other key features in order to run the applications as efficiently as possible. This would make it so this application can run as smoothly as possible without disturbing the other processes the machine may be running during that time.
3. **Storage Management**: Windows has quite a few strong features that focus on storage management. Disk Cleanup mainly would help remove unnecessary files in order to optimize storage space. Disk Management is another Windows feature that can handle more specialized tasks beyond the of simple clutter file removal.
4. **Memory Management**: A database will be required to store the images needed for the game, as well as storing user information as needed. Windows provides Memory Management as a feature, which will prove handy for this task.
5. **Distributed Systems and Networks**: A strong server network is needed for this project. Multiple clients would need to connect to a server in order for users to play together. As such, a client–server distributing system would be best here.
6. **Security**: User security is one of the most important aspects to consider when building a project for clients to use. Poor security can lead to disastrous outcomes. Leaked email addresses, sensitive information being exposed, passwords potentially becoming compromised, etc. I would recommend rolling out the option to use Two-Factor Authentication or 2FA. From the OS side of things, Windows offers many solid lines of defense that are worth consideration, and as such would provide a solid line of defense for the user.