

GameRoom

# **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 | 11/12/2023 | James Reid | Executive Summary; Design Constraints; Domain Model |

**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 Staff have tasked our firm with developing a web-based version of their “Draw It or Lose It” game. The program must have the capacity to have multiple players which are split into teams while preventing duplicate names.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* The game must be in a web-based language.
* Requires team and player objects.
* Must verify uniqueness of team and player names.
* The game must have only one instance active in memory.

## [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 UML Diagram below shows how the different classes in this program work together. The Program is run by the Program driver class which includes the main method. It creates objects of players and teams which both are entities and inherit the properties of entities. While the objects are generated, the class SingletonTester verifies that the team names and ids are unique and flags duplicate names or ids. Game Service contains most of the game’s functions that are called when needed.

**"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** | The mac operating systems have web Apache server commands built in as well as python. Mac is limited to apple’s supported hardware configurations. | Linux can provide very powerful servers as the operating systems have built-in command systems and webservers. Linux is also much more cost-effective as many “flavors” are free to download as compared to mac and windows which require users to purchase a license. | Their servers are seen as easy to use as they are very similar to the client versions of the operating system. Windows also allows web extensions and SQL support. | Mobile apps have less options to backend than the other operating systems. This setup would not work as well for our application. |
| **Client Side** | Mac costs more on average than windows. Ease of use is similar between the two operating systems, but users generally prefer one over the other. Mac is generally quick to adopt and easier to learn than other operating systems. | Linux is a very “light” operating system. This allows fast computing and customizability. Linux is harder than other operating systems for the average user to learn. Once learned however, the multitude of different flavors and programs makes Linux a very cost-effective and useable tool for developers. | Windows has many proprietary programs that can only be used on windows devices or virtual machines. Windows can easily virtualize other operating systems. | Mobile devices come from many different companies and have many different forms and functions. Mostly mobile devices either run on android or iOS. Mobile devices allows users to apps and programs anywhere. Mobile apps need their UI to fit many different screen sizes. |
| **Development Tools** | Apple uses many proprietary programs as it uses Unix as its main kernel. Mac’s utilities include but are not limited to Xcode, dash API, JavaScript, HTML, CSS, React, Chrome development tools, and SQL | Linux offers many programs for users as many are akin to other operating systems proprietary programs. Atom, Vim, Visual Studio, Bash, Git, Flask, and Node. | Visual Studio Code, git for Windows, Git bash, node, Javascript, HTML, CSS, React, mySQL | Firefox, Opera, Samsung browser, Chrome, Metro browser. Website should work across all mobile browsers.  (Mobile devices may have to allow access to mobile browsers) |

## 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**: I believe that a Linux based server would be the best option as it can be very “Lightweight” and customizable.
2. **Operating Systems Architectures**: The Linux kernel is very stable and can be easily applied in many different niche ways. Linux can easily separate between system and hardware needs.
3. **Storage Management**: The two options that I would recommend would be SSD’s if your goal is the fastest possible read and write speed. This would allow fast access to photos, user information, etc. If your goal is to have the most space available for the cheapest, I would use high-capacity hard drives. With either system, a file storage program would help make the best use of the available space.
4. **Memory Management**: The operating systems of Linux can have a watcher application that throttles memory based on usage. This will allow the memory to adapt to the current load. This will help if you are paying for the memory usage or trying to save on energy usage.
5. **Distributed Systems and Networks**: A cloud-based system has the benefit of switching servers when needed in order to avoid downtimes. The node can be moved to a separate server if there is maintenance or issues with the current server. Client-side devices will just need an application that can access the server. The server hardware can do all the “heavy lifting”. This can allow devices with limited processing power to be able to access the server.
6. **Security**: The separation of roles in an admin and user based system would be my first choice. It is easy to implement and manage. This role-based system can help prevent users from accessing or editing information that they should not have access to.