

Draw it or Lose it

# **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 | <01/21/2023> | Dillon Belanger | Executive Summary completed with Design Constraints and Domain Model sections. |

**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 wants us to develop a web-based prototype for their game Draw it or Lose it. For this design, the game will have: Multiple teams, multiple players on each team, have unique team names to differ from names already in use, and have only one instance of the game to be on at all times.

## 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)

* Team and player objects will be needed for multiple teams as well as multiple players for each team
* Game will have to be written using Java or python for full functionality as it is a web-based game
* Game and team names must be reviewed and checked within the game to not have overlapping and have separate teams to function
* Only one instance of the game should exist in the memory through unique id for the instance of the game as well as team and players

## [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 Game, Team, and Player classes are inherited from the entity superclass providing all the attributes needed and operations. GameService, Game, Team, and Player classes are linked to each other with a zero to many associations allowing many instances to be used as needed. ProgramDriver class drives the package by inheriting and using the SingletonTester class.

**"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** | Mac is easy to understand and easily accessible, providing straight forward server configuration. Providing a graphical user interface for extended ease of use. | There is more difficulty when using this compared to other platforms. It has easy server configuration and accessibility with command shell and is cost effective. | Windows on the server side is costly. Has ease of accessibility with a graphical user interface and many software options, includes a command prompt. | Mobile devices can vary between different users and their phone types. Very cost effective with android and IOS, although creating a game is difficult with compatibility to other mobile devices and platforms. |
| **Client Side** | Time is needed for the user to learn and navigate with the right skills. This is not a cost-effective option for the user. | User for Linux users there is much time needed to use Linux systems. Minimum cost is needed but knowledge is a requirement. | Not as much time is needed to learn and navigate Windows systems. There is ease of access with the catch of the cost being more expensive than with Linux systems. | Much time and skill are needed to work with mobile devices with application development. Compatibility with other platforms is particularly challenging and limits mobile device systems. |
| **Development Tools** | There are multiple IDEs for mac including java, html, css, and python. Tools that mac has to offer include visual studio, eclipse, and tools for online development. | IDEs for Linux include java, python, Ruby on Rails and HTML. Development tools for this system are Github, Visual studio etc. | IDE’s within Windows include python, C++, html and other C languages. Developer tools for this platform include eclipse, command prompts, and many others supported from other OS systems. | Programming languages for mobile devices include html, php, C++, python and so on. Development tools include Github, command prompts, and so on. |

## 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 operating platform of choice based on all the research done is Windows. I believe that although Linux may be better at a technical level, Window’s familiarity and convenience in use outweighs that. The cost of using windows is relatively low and there are many IDEs that work with Windows. It is quite common and allows more user knowledge and compatibility for the game in the future.
2. **Operating Systems Architectures:** For Windows, the application can use the kernel processes without interacting directly with them**.** So, applications can use resources for a GUI and window setup without affecting the process needed for the operating system to function.
3. **Storage Management**: Windows comes with disk management as well as storage sense. These allow the storage to be managed efficiently and it comes directly with the operating platform. There is also disk cleanup which works in tandem with storage sense to remove unnecessary files that could take up storage space. Disk management can perform the advanced storage tasks that are needed for the application to effectively manage the storage needed. These are all conveniently built into Windows itself.
4. **Memory Management**: Windows also had memory management built into the system utilities. This allows a database for the game’s image files to be made and accessed by the application with ease.
5. **Distributed Systems and Networks**: For the application to work properly Windows can be utilized for a client-server distributing system. Allowing the capability of a single server application for each client, this means the clients with better or worse systems can adapt for the application to be functional at all levels. Effective server networks will also allow multiple clients to connect to the single server for multiple players per game.
6. **Security**: Windows has its own security built in being “Windows Defender.” It is a mandatory system security that all windows platforms would have. So, the data for the application can be encrypted and sent through with tried-and-true encryptions. This would strengthen the security for the application, and it is completely provided by Windows itself.