

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 0/18/2022 | Nicholas Kolment | Original Design |
| 1.1 | 10/2/2022 | Nicholas Kolment | Edited Exucitive Summary, Design Constraints, and Evaluation |
| 1.2 | 10/16/22 | Nicholas Kolment |  |

**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 to create a multi-platform web based game based on their current Android game titled “Draw It or Lose It”. The game needs to serve multiple clients and platforms. Each game will have one or more teams involved, each team will have multiple players assigned to it. Game and team names must be unique, and should allow users to check if a name is in use. Only one instance of the game can exist in memorrry at a time.

## [Design Constraints](#_2et92p0)

The project will require three development environments, for Android, iOS, and browser.

Memory usage will have to be at a minimum to ensure the most amount of devices will be able to use the application, using the server to process information is the best way of solving this issue.

Since security is a priority, multi-factor authorization should be added to the log in for each sign in.

## [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)

Game, Team, and Player classes inherit from Entity Class. The Team class take information from the player class, the Game class takes from the team class, and the GameService class takes from the Game 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** | -More expensive than Linux  -Flexible Terminal Commands | -Cheapest  -Open source  -Good security  -More difficult than windows  -Requires working with the command line | -Around the same price as Mac  -Easiest to use  -Most Available  -Most likely to get viruses | -A mobile server would not be practical, if the device is moved to a low service area, then the server will be slow. Less processing power |
| **Client Side** | -More than Linux  -Used less than Windows but moe than Linux  -Moderate Time  -Moderate Expertise | -Lowest cost  -Least used  -Maximum Time  -Maximum Expertise | -Cost similar to mac  -Most used  -Minimum time  -Minimum Expertise | -Max time and expertise due to mobile devices using different versions of their operating systems |
| **Development Tools** | -Swift | -Eclipse – Java  -PyCharm - Python(Django – used for web based python development.) | -Eclipse - Java  -Visual Studio – C++    -PyCharm - Python(Django for web based applications) | -Java for Android  -Swift for iPhone |

## 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**: Windows
2. **Operating Systems Architectures**: Windows API provides services for all windows based applications to provide access to a Graphical User Interface, audio systems and more.
3. **Storage Management**: Windows allows for storage expansion by adding more hard drives, also storage management and use of the cloud.
4. **Memory Management**: Windows can access up to 4GB of physical memory, it uses a data tree structure to mark each node of the tree as free, committed or reserved.
5. **Distributed Systems and Networks**: Making the game on a cross platform engine could make this cheaper and easier than developing all the code from scratch. Using an engine such as Unreal or Unity we can make the game compatible with all the required operating systems.
6. **Security**: Windows comes installed with security programs such as the firewall and a basic virus and threat protection appliocation. A third party security software will add for extra protection for the users data.