

<Draw it or Lose it>

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | <11/10/22> | <Alexis Yarbrough> | <Preparing document foe the software design> |

**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 desires to succeed in the making of a web-based game that is found in their game now called Draw it or Lose it. The game they have is accessible in only the android app but they want their new game to be available on multiple different programs. This new game will include four one-minute rounds where an image will be taken from a library and one team will have to estimate the answer before the time ends up. The opposite team will have fifteen seconds to guess the answer if the first team answered incorrectly.

## [Design Constraints](#_2et92p0)

* One or more teams to be involved
* More than 1 player is assigned to each team
* A unique name is essential for the game and team names so it can allow people to make sure it is not already in use
* Only one instance of the game can exist in the memory at any given time

Based on this information this game should have one or more teams involved, for team names they need to be unique so users don’t use the same name. Each team will have more than one player assigned to it and once the game is played one instance of the game will exist in the 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)

Starting with the Entity class a connection is between the player, team, and game and this allows them to take information from it. The entity class has similar features with the player, team, and game and they all have a is-a association. Player and team, team and game, gameservice and game have a “has-a relationship. This is showed by inheritance. The game has teams and the team has players, the gameservice has games and this lets them have more than one. That is showed in the UML with aggregation.

**"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 OS can have great features and be on the pretty side. The features are easier to use than other Ops and can be easier to understand. The drawback of this is it doesn’t have as many options or flexibility for upgrading. | Linux has low infrastructure requirements, so this helps it show good performance. The server is not demanding for the client hardware, and it has a lower resource utilization. | Compared to Macs, windows can offer a software adaptability. For windows they have bad virus protection and tech support | More mobile, and you can use wherever and it not be a hassle. There are more customers with it. Drawbacks for it can be the cyber crime that happens so often with them. |
| **Client Side** | First thing that clients notice about macs are their price, they are more expensive, and it is only provided by apple. | You can purchase linux for a lower rate although you have to know what you’re using when you have one. | Windows are easiest to use and don’t have to use a lot of time but they can get just as expensive as a mac. | Depending on the brand these devices can be really expensive. |
| **Development Tools** | There are a lot of languages, most all of them that can run with using Mac OS. | Swift, python, and java are some of the best programming languages for linux. | Similar to linux, windows can use the same IDEs and mostly all languages can run with windows. | For android and ios systems swiftic is probably the best and more suitable. |

## 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 would probably be the best to recommend since it is easier to work with and learn, and the software availability is better than other OS, you can use a multitude of IDEs and languages.
2. **Operating Systems Architectures**: Windows OS has a kernel which is the most trusted part of the OS and is built upon a 7-layer networking model
3. **Storage Management**: Windows storage management manages a big range of storage configurations. They can use single-disk desktops for external storage arrays. Firsect management of files to your hard drive can be made aswell and doesn’t take up all the space and it would be easier to find your applications.
4. **Memory Management**: windows can assign your RAM for a single application so this allows for more optimized ram for gaming. Being that it takes a lot of data to create and store games in windows that would be the best option.
5. **Distributed Systems and Networks**: Since the main point of building this game is so it can be used on different platforms I would use Python. Although this would require a python interpreter when used with widows.
6. **Security**: Windows has system updates that can be set automatically to keep the user and the system information safe. It also has a built in security protection.