

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 | 08/20/2022 | Robin Robinson | Initial revision |

**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 game “Draw it or Lose it” will comprise of one or more teams per game. Each team must have a unique name that is not currently in use in a game. The game is 4 rounds, 1 minute each giving a team 30 seconds, until drawing is done, to guess. If a guess is not made the remaining teams each have 15 seconds to make a guess.

## [Design Constraints](#_2et92p0)

Constraint #1: There can only be 2 Teams per Game

If each round is 1 minutes long with each team having 30 seconds to guess, so 2 teams would equal a whole minute for the initial guesses

Constraint #2: No Game, Team, or Player names can be in use in the respective name pool.

If someone is already using the name “Lions” then if someone else tried to use the name “Lions” it would be to confusing to the players, so each name for their respective name pool must be unique

Constraint #3: There must be at least one team per game.

There must be at least one team per game for a game to start or there would be no one to play, because if there is no teams then there are no 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)

Entity is the super class with Game, Team, and Player inheriting from Entity, particularly the id, and name for the respective classes.

When the ProgramDriver is initiated with game1 and game2 it gives each and game ID incrementing from the last, starting with 1 if the provided name is not in use. When a Team and then Players are added, the same thing is done with them as it checks the name to see if it is currently in use, if not it gives it an id.

**"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** | Does have a server specific OS, though it is not widely used | Is open source and has many options for what is needed. | Has a server specific OS that makes it easier to utilize on the server side. | Used by many people that like to be on the go, it is more varying as there is Android and Apple and others that each have there own operating system |
| **Client Side** | Can be costly for the client to use macOS and is not as compatible with some software as Microsoft | Very cost effective as it is open source as well as very adaptable to the client’s needs, less compatible with other software as the other operating systems. | Can be more costly for the client, but has high user rate and is easier to maintain and has a lot of support | Can be fairly cheap and is highly used by many people so has a high probability of being used. |
| **Development Tools** | Median ground of popularity, so many people have experience with macOS. It has many development tools available as well. It is not very adaptable to the developer though. | Highly adaptable for the developer and has many available and open-source options. It is highly adaptable as well. | Most easily identifiable and most likely to have more experience with Windows, as it is highly popular. | Can be more difficult to make compatible with many different mobile devices |

## 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 would recommend Windows as it has the best balance of support and cost efficiency as well as adaptability.
2. **Operating Systems Architectures**: I would recommend Windows because it is the most familiar to work with most people as it is the most popular at this time.
3. **Storage Management**: I would recommend Windows because it is the most adaptable for the cost of its use.
4. **Memory Management**: I would recommend Windows because it is the most familiar to most people that will utilize it.
5. **Distributed Systems and Networks**: I would recommend Windows because it has a server specific OS that is easily navigable and used by many.
6. **Security**: I would recommend Windows because it is updated and maintained regularly and has may things to help with security of a software.