

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 | 07/15/2022 | Daniel Tarulis | Revised and added information to entire document. |
| 1.1 | 08/11/2022 | Daniel Tarulis | Revision of Recommendations |

**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 develop a web-based game called “Draw It or Lose it”. The game is currently only available on android and The Game Room wants it to run on multiple platforms. The game has multiple teams consisting of several people. There are four rounds at one minute each. Pictures are pulled from a set library of images and used as clues for 30 seconds while one team guesses until time runs out. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
* Game must run on multiple platforms
  + Find a way to re-write code for other platforms
  + Find a way to run existing code on other platforms

## [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 Class all inherit from the Entity class, making it a superclass. These classes will all share their attributes. Looking at the GameService class, it references the Game class. The Game class references the Team class, and the Team class references the Player class. ProgramDriver contains the main() method and uses the SingletonTester class to only have one instance of the game at a time.

**"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** | * Easy creation of features * Flexible terminal commands * Good web hosting services | * More cost friendly * Secure * Web hosting applications not as easy to find | * Large amount of software availability * Widely used compared to other platforms * Not as secure as other platforms | * Very popular platform * Cost effective |
| **Client Side** | * Average expertise and time needed * Cost on par with windows | * Great expertise and time needed * Low cost | * Low Expertise and time needed * Cost on par with Mac | * Average expertise and time needed * Slightly lower cost |
| **Development Tools** | * Swift * Notepad++ * HTML * CSS * JavaScript * Java * Python * PHP * Ruby | * Visual Studio * Eclipse * HTML * CSS * JavaScript * Java * Python * PHP * Ruby | * Visual Studio * Eclipse * HTML * CSS * JavaScript * Java * Python * PHP * Ruby | * Android * Swift * Notepad++ * HTML * CSS * JavaScript * Java * Python * PHP * Ruby |

## 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 be a good starting recommendation for The Gaming Room’s “Draw It or Lose It”. There is a wide variety of available software as well as minimal expertise and cost needed to create projects. There are also a wide variety of IDE’s and tool available on Windows.
2. **Operating Systems Architectures**: Windows uses Windows-based applications while showing a GUI for easy use. A Windows account will also be used.
3. **Storage Management**: Cloud availability with Windows makes it easy to save and retrieve data. Windows also has a built-in storage management system allowing you to manage files on hard drives easily.
4. **Memory Management**: Windows has two types of memory architectures, 32-bit and 64-bit. You can store the project information like the library of images or even the IDE in the desired location you choose. This helps that data become more secure
5. **Distributed Systems and Networks**: Outages and connectivity issues can be prevented by making sure there are enough reliable servers to support the player base as well as prevent outages.
6. **Security**: Windows has its own security software. However, A third party software would be recommended for added protection.