

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 | 03/20/2022 | Jason Hall | Completed report |
| 2.0 | 04/7/2022 | Jason Hall | Revised Evaluation |
| 3.0 | 04/22/2022 | Jason Hall | Revised 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 would like develop a web-based game “Draw it or Lose it” that operates on multiple platforms. Currently the application is only available on Android. A game consists of four rounds lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. 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)

* Multiple players each game
* Only one instance of the game can exist in memory at one time
* Must run on multiple platforms
* Game and Team names must be Unique

Multiple players for this game will either require multiple users to use one device to play the game or will require an active internet connection in order to play with multiple users online. The game is required to be on multiple platforms which would require taking the code for the game that already exists and converting it to a language the MacOS supports such as swift. If the game is to be played online then cross-platform compatibility will increase time needed to complete the application.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The Entity class is a super class and the parent class to the game, team and player class which means that all three of these classes inherit information from Entity.

**"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 OS X Server  Advantages: secure and less prone to viruses/hacks.  Open Source  Cost efficient  Disadvantages: less resources available for development. | Advantages:  Cost efficient, secure.  Open Source  security risks are caught before issues arise.  Disadvantages: less applications available that support web hosting needs. | Windows Server 2022  Advantages: large amount of resources available to develop.    Can handle business critical workloads.  Cloud/on premise option.  Cost is moderate  Disadvantages: prone to viruses | Advantages:  Cheap to host on a mobile device  Able to host basic websites.  Disadvantages:  Easily overloaded |
| **Client Side** | Requires MacOs devices in order to develop MacOs apps. Development time will increase due to this. | Less Linux experts available which will increase development time. Cost will be less due to less people working on linux side. | Most common Operating system used. Development time will be less than other operating systems. Cost will be moderate due to more available developers. | Provides flexibility at little to no cost. Will require a powerful mobile device and very skilled developers. |
| **Development Tools** | IDEs:  SwiftUi  Xcode  Programming languages:  Swift  HTML  CSS  Javascript | IDEs:  Eclipse  Visual studio  Netbeans  Programming languages:  Python  C++  C  Java  Javascript  CSS  HTML and more | IDEs:  Eclipse  Visual Studio  Netbeans  Programming languages  Python  Java  C++  C  Perl  Ruby  HTML  CSS  Javascript | Android:  *IDEs*:  Eclipse  NetBeans  Visual Studio  *Programming languages:*  Kotlin  Java  C++  C#  iOS – can be handled with macOS IDEs/programming languages. |

## 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 recommend starting the application on Windows, as the game will be developed in the shortest amount of time due to the number of experts and resources available. The extra time can be used to review and test game before moving to other operating systems.
2. **Operating Systems Architectures**: The Windows application programming interface (API) provides services used by all Windows-based applications to enable applications to provide a Graphical User Interface (GUI), access system resources as well as many other features.
3. **Storage Management**: I recommend consider using cloud storage. This will be more cost effective for the game developer as creating a data center and providing the hardware necessary for the data centers can be expensive to create and maintain. Maintaining your own data center can be expensive when the hardware fails and can take some time to repair which can cause your customers to complain when this happens. Cloud storage options such as AWS can provide resources that will not only meet the needs of the game but provide more data/storage as needed. AWS will also let you know if you are not using all the data in your plan in order to cut costs for your company.
4. **Memory Management**: Memory allocation will make it easy to store pictures in a library or data base for the game during the development process. Using a stack data structure will be useful here for the storage of files, objects, data and pictures.
5. **Distributed Systems and Networks**: Since the game is going to be web-based it should be created in HTML 5 combined with Javascript and CSS, this will allow the game to be accessible on all platforms(windows, iOS, and android).
6. **Security**: Windows has built-in security protection software, but is also more prone to attacks than other operating systems. It will be important to provide team with social engineering training as well as training in secure coding.