

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 | 05/21/2022 | Brady Steele | Updated constraints, evaluation, recommendations |
| 2.0 | 06/05/2022 | Brady Steele | Updated evaluation |
| 3.0 | 06/18/2022 | Brady Steele | Updated 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 game called Draw It or Lose It. This game would be a web-based application that can run on many platforms. This game is currently only available as an Android app. This game involves teams competing to guess what is being drawn. There are four rounds, each lasting one minute. The drawings are expected to be complete by the thirty second mark, and if a team does not correctly guess by one minute, the opposing teams have fifteen seconds to solve the puzzle.

## [Design Constraints](#_2et92p0)

* This must run on multiple platforms and is currently an Android application.
* The game requires multiple teams that must have unique names.
* There may only be one unique instance of the game running at a time.
* Developers with unique skillsets are needed to be able to translate the code over from the Android app into the different environments.

## [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)

In the UML class diagram, we can see that Entity is a super class, and Game, Team, and Player all inherit from it. The GameService class is also connected, as it has a reference to the Game class, which is in reference to the Team and Entity classes, as well as the Player class. The ProgramDriver and SingletonTester classes are related as well. This relationship allows for the creation of a single unique game with multiple teams consisting of multiple players.

**"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 has a very user-friendly terminal and can essentially operate as any platform. Anything a Mac can’t do, can be accomplished with a dedicated VM. | Linux servers are relatively low cost, but Linux does not necessarily seem as user-friendly, as a Unix based system. | Windows has their own server capabilities available, as well as a full suite of Windows software products. This makes it easier to find what you’re looking for, but you are limited from using Mac platforms. | Server capabilities of a mobile device don’t seem to be at the same level as these other operating systems. I think a server would be better off hosted elsewhere. |
| **Client Side** | This one really comes down to experience because operating costs are seemingly similar across the board. A person really needs to be familiar with Mac processes and the time taken would be much longer for an individual with no experience here. | Linux maintains the lowest cost, but potentially the highest barrier to entry. Linux may require the highest level of experience and expertise compared to the rest of these operating systems. | The cost seems consistent across the board, including here with Windows. The same goes for the Mac, where a person needs to have experience with the processes to be effective with this operating system, or else the time is increased. | Mobile devices are easier in this category because there are many good mobile device tools available. Mobile devices are becoming increasingly more powerful and capable and could be a viable solution for portability. |
| **Development Tools** | The Mac has the most access to the full suite of development tools and frameworks. Being a Unix-based machine, Macs are very powerful, but can also access Linux environments. This makes the Mac the prime choice for development, due to the usability. Development team members with different areas of expertise might be needed to handle each different platform. | Linux can work with multiple development environments and frameworks but does not have access to any suite of Apple development products. This can be a hinderance, as multiple machines or VMs may be required to complete development across a whole suite of applications. Development team members with different areas of expertise might be needed to handle each different platform. | Windows has a lower barrier to entry than Linux but runs into the same problems with Apple products. Windows can handle a lot of things well but cannot complete the full suite of products that would be required for the product. Development team members with different areas of expertise might be needed to handle each different platform. | The mobile suite of development tools is very diverse and can be accomplished on pretty much any machine. The exception if for iPhone applications, as the primary tool for creating these is Swift, and potentially XCode as a development environment. This can only be accomplished on a Mac but is a worthwhile language to utilize. Development team members with different areas of expertise might be needed to handle each different platform. |

## 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 that The Gaming Room expand their operating platform to the Mac. I say this because of the ability of the Mac to handle any development environment, while also satisfying the need for an iPhone mobile application to complete the full suite of products translated from the original Android app.
2. **Operating Systems Architectures**: The Mac architecture contains an open-source OS X kernel called Darwin as the first layer. This is followed by a graphics layer, then an application layer, and finally the Mac user interface.
3. **Storage Management**: The Mac comes with different storage management tools to help optimize the operating system. Being an Apple product, any user also gains access to cloud storage with iCloud.
4. **Memory Management**: The Mac would use its built-in memory management tools for developers to optimize the use of RAM for application processes with Draw It or Lose It. The operating system could store the game data in the memory, to access it throughout the game, without entering any low memory situations.
5. **Distributed Systems and Networks**: To port this game out to many operating systems, and to avoid outages, this game could be hosted as an application container on a Virtual Machine. This could be done many ways, such as a Docker container, or running on a Kubernetes service. These services would keep the pods running, even if there was a master node outage, and could utilize rolling updates to ensure that there is no application downtime.
6. **Security**: There are tons of options for application security within the Mac operating system. There are many APIs available for developers to utilize to keep sensitive information private, such as cryptographic interfaces, algorithms for user data protection, among others that are included with the Apple developer suite.