

**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 | 07/21/22 | Austin Palmer | Initial creation and editing |

**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 to create a web-based version of their android gaming app: “Draw It or Lose It.” This app renders images from a library of stock drawings and consists of four rounds lasting one minute each. Drawings are rendered at a steady rate for 30 seconds. After 30 seconds, if the drawing is not guessed other teams have 15 seconds to guess the picture for points.

## [Design Constraints](#_2et92p0)

* One or more teams must be involved. This implies that there must be multiple objects of the team class.
* Each team has multiple players. This implies that there must be multiple objects of the player class.
* Game and team names must be unique. This implies that there must be a form of name checker that verifies the name is not used already.
* Only one instance of the game can exist in memory at a given time. This implies that the game must be developed in one instance of Game.

## [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 Entity class contains methods that relate to getting/setting an Id or name. The Game, Team, and Player classes all inherit from the Entity class and are associated with one another. The Player class adds a constructor for Player and a string converter method. The Team class contains a list of Player objects with a constructor for creating a Team, an addPlayer method which adds a player to a list and a string converter method. The GameService class and Game class are associated with each other and GameService is used to manage instances of a game and service the game.

**0"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 hosting allows for a wide range of tools for developing the server side and hosting it provides a fast/secure experience. However, MacOS can be difficult to integrate into larger server environments and it has a higher learning curve. | Linux systems are stable and reliable which allows for stable web application hosting. This would help implement changes to the server without failures. Due to its open source nature, Linux can be difficult to integrate. | Windows is great for developing web sites and hosting Windows apps due to its speed and file type hosting. However, Windows does not have a lot of built-in security, but a library may be used to enhance protection. | Mobile devices do not have as much support for web application hosting. Mobile devices allow for a wide range of customization with hosting; however, it is not a great idea to only host a server with a mobile device due to the limited functionality. It varies too much system to system. |
| **Client Side** | Macs are more expensive compared to its counterparts but are fast and efficient when supporting clients. Macs also provide the client with an easy-to-use interface. | Linux is free to download and may be the best option for programmers. Linux has a learning curve however so having a technical knowledge is necessary for clients. | Windows is the most used operating system for desktops and generally comes at a cheaper price point. Additionally, Windows is generally easy to use/learn and has a lot of compatibility. One disadvantage is that Windows does not have as much built-in security. | Mobile devices are harder to develop for due to their limited hardware capabilities (compared to a PC). There is a wide variety in types of mobile products which may cause differences in functionality between, for example, a tablet and phone. Mobile devices have a wide range for pricing but are generally more expensive but nearly everyone has one. |
| **Development Tools** | Objective C is the most common programming language. Visual Studio and Xcode are two IDEs that are well integrated with MacOS and have free versions but with paid versions that includes more features. | Most programming languages are supported and well developed for Linux. Atom and visual studio code are two supported IDEs. Atom and visual studio both have free versions. | Most programming languages are supported and well developed for Windows however C++ was used to build most Windows Applications and has the best functionality. Windows has the most options for software development tools and IDEs. | Java is the most common programming language. Swift is also used a lot for IOS development. Android Studio, XCode, and Visual Studio are common IDEs for mobile app development. XCode and visual studio have free options, but a $25 license is required for Android Studio. |

## 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 is the best choice for operating platform due to its wide range of development tools on the back end and fast/easy use on the client side. Additionally, C++ and C# are versatile languages to use for game development and have great functionality with
2. **Operating Systems Architectures**: Windows consists of two layers: a kernel mode, which can access memory and execute instructions, and a user mode, which gives a user a GUI and limits a user’s access to these operations.
3. **Storage Management**: Windows has a built-in storage management provider which allows can be used to manage storage on SSD or HDD. A user can create folders, use cloud features such as OneDrive, and move files easily in the Windows GUI.
4. **Memory Management**: The 32-bit windows OS allows up to 4 GB that is divided between the kernel and user layers. In Draw It or Lose It the kernel will have to locate and retrieve pictures which naturally are large files. The kernel manages the resource allocation while the user layer is dedicated to running smaller user processes and I/O.
5. **Distributed Systems and Networks**: An easy way to create a multiplatform game is to use a engine like Unity, which allows for cross-platform support between Windows, Linux, Mac, IOS/Android, and consoles. Unity gives developers tools for efficient game operations such as multiplayer services. In the context of Draw It or Lose It, Unity can be used to reach its multiplatform goal and create a smooth link game maintenance.

Source: https://unity.com/

1. **Security**: Windows has a built-in security program and the ability to add player/identity authentication to any program. This will help moderators of Draw it or Lose It to control its player base and ensure that there is no fraud or cheating while the game is active. Windows Security includes system scans for malware and viruses which will also help protect game data.