

Draw It or Lose It Web-based Solution

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/22/2023 | Rob Marlatt | Initial Design Document |
| 2.0 | 02/05/2023 | Rob Marlatt | Updated Server Side, Client Side, and Development Tools sections |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is looking to develop a web-based version of their game Draw It or Lose It, which is currently available as an Android application only. We will provide the necessary software to facilitate The Gaming Room in replacing their current Android application with a distributed, web-based software application.

## Requirements

* 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. We will create unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

## Application must be written in a software language that allows it to be accessed from any OS and Web Browser.

* Software must be written using industry standard best practices.
* Software must have the same look and feel of the original Android application.
* Project must be completed on time and within budget.

## [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)

Per the UML diagram below, our software application will have an Entity superclass, containing all the basic information needed in the other classes. The Entity superclass and the Game, Team, and GameService classes all show the principle of Encapsulation, where they have variables that are private and can only be accessed within the class. The Game, Team, and Player classes all show Inheritance from the Entity superclass. The Game, Team, and Player classes all show Abstraction, meaning that the implementation of the class can be inferred from the name of the class without needing to know how the class actually works. The GameService class shows Polymorphism, in that the getGame() method can have different arguments for it.

The Game, Team, and Player classes all have references to each other, and the GameService class has a reference to the Game class. There is also a ProgramDriver class, which includes our main() method, and it uses the SingletonTester class, which contains the testSingleton() method, used to ensure that there is one and only one instance of each Entity. The way that the Package com.gamingroom is set up is so that the classes allow for one game to be ongoing with multiple teams with multiple players, each with a unique 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)

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 has Mac OS Server that has been deprecated. Some features are inbuilt into Mac OS, but other features would require additional software; some free, some requiring licensing. | Linux has distributions of its software that are specifically designed for server applications, such as CentOS, Ubuntu Server, Gentoo, Debian, or Slackware. These have all the functionality inbuilt but would require someone proficient in Linux to operate it. | Microsoft offers Windows server, which has all the software necessary to operate and maintain a server, but it comes at a cost, ranging from $500 to $6200 for licensing, depending on server types and size. This would be good since it is designed for Windows but would be costly to implement. | Cost-wise, running a server on a mobile device would provide the best option, but since these are low powered devices, the server would not be able to scale to meet the needs of the client. |
| **Client Side** | Setting up a client on Mac would be possible, at a cost. The software is closed source and would require someone who is proficient with the OS to program and maintain it. | Linux provides many open-source software solutions to provide the client-side software, and the cost would be free or nearly free, but Linux is not the friendliest OS for programming and would require someone to be able to write the project. | Setting up the client side for Windows would be possible, but cost would be a factor. There is no open-source software on Windows, and someone would have to be proficient in Windows to program and maintain it. | Mobile apps would be easier and cheaper to write than any of the desktop OSs like Windows, Linux, and Mac; but there would have to be multiple versions of the app programmed to work on multiple hardware configurations. |
| **Development Tools** | Swift is the programming language used to write software for Mac OS. It can be written in an IDE such as Atom. There would be no licensing costs associated with becoming a Swift developer. | Eclipse and Atom are the most commonly used IDEs for writing software for Linux. Eclipse is mostly used for Java but can also be used to write code for C++. There is no licensing cost associated with Linux development. | Eclipse and Visual Studio are the most common IDEs for writing code in Windows. Visual Studios also can be used to write code for HTML, C#, JavaScript, and others. There is no licensing cost associated with Windows development. | iPhone apps would be written in Swift with Atom IDE, just like for Mac. Android apps would be written in Android Studio. These apps would require separate code to be written. There is a one-time licensing fee of $25 to obtain a Google developer license. There are no costs associated with becoming a Swift developer. |

## 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**: Considering all the above, I would recommend that we use Windows as the operating platform. It is the most widely used operating system, so most people will have a good knowledge of the platform. It has several IDEs that are compatible with Windows, and there is a low cost to implement it.
2. **Operating Systems Architectures**: Windows allows for running applications to utilize core kernel processes without making any changes to those processes, so an application can have a GUI with access to memory and other processes to make it work without causing problems with the operating system.
3. **Storage Management**: Windows provides Storage Sense, Disk Management, and Disk Cleanup directly in the Operating System. This allows the team to manage the disk storage on the system by controlling the processes accessing storage and deleting and optimizing files taking up space on the storage.
4. **Memory Management**: Windows has inbuilt Memory Management as a system utility, allowing the team to manage the processes accessing memory. We would also have a database containing the game’s image files to make them easier to access by the application.
5. **Distributed Systems and Networks**: Since the team would be implementing a client-server system for the game application, this will allow each client to access the server without affecting the other client applications that are accessing the server. This will provide a robust network that will be best for the client’s needs. This method will require a strong server network to handle all the transactions required for multiple clients connecting to the server to play the game together.
6. **Security**: Windows Defender is a robust security suite provided directly in Windows. Windows Defender provides anti-virus protection and a firewall to help prevent intrusion into the system. The team would also have to encrypt all the data being sent through the client-server system with a strong encryption system.