

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 03/18/2023 | Tatiana Case | Initial documentation for new game, Draw It or Lose It, with unique factors. |
| 2.0 | 04/2/2023 | Tatiana Case | Updated evaluation and recommendation sections with detailed information |
| 3.0 | 4/16/2023 | Tatiana Case | Finalized recommendation details |

## [Executive Summary](#_sbfa50wo7nsh)

Our client “The Gaming Room” would like to create a web-based game named “Draw It or Lost It.” In each instance of the game, team, and player will have unique names. According to the client’s specifications, the game will only allow one instance of the game in memory at a time and allow multiple teams with multiple players.

## Requirements

* A game will have the ability to have one or more teams involved.
* Only one instance of the game can exist in memory at any given time.
* Each team must be able to have multiple players assigned to it.
* Game and team names should be unique and must allow users to see if a name has already been used.

## [Design Constraints](#_2et92p0)

* Must be written in a web-based language, like Java or Python, to be fully functional on any web-based environments.
* Must check game and team names in to ensure each game is unique by using a singleton pattern when creating games and teams.
* Must allow multiple teams with multiple players on each team by implementing team and player objects.
* Must allow only one instance of the game to exist in memory by creating unique IDs for each instance of the game, team, or player through the iterator pattern.

## [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 class, Team class, and Player class each inherit from the Entity superclass. This allows the attributes and operations in the Entity class to be written only once and used by each class inheriting from it. The GameService class, Game class, Team class, and Player class are all associated. They also use a zero to many associations. This is so they can use as many instances of the others as needed or none. The ProgramDriver class drives the package while inheriting the SingletonTester class.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac OS boast its flexible terminal commands to configure the server, allow access, and alterations. OS X Server is available for Mac, but unless the client purchases their own hardware, finding hosts will be expensive. OS X Server is $499 USD for up to 10-clients or $999 for unlimited clients. | Linux is the most popular webhosting OS. Linux is opensource. This means maintenance and license costs tend to be cheaper than closed OSes, like Windows. Also, typical cloud providers, like Google and Amazon, offer Linux over Windows. | Windows servers are beneficial because they are familiar Also, because it is GUI based and their office apps run on the same server. License costs, typically per user, tend to be high. Licenses range from $6,200, for up 16 core licenses, to $500, for up to 50 clients. Hosting platforms are also more limited compared to Linux. | Mobile devices can be used as a personal webserver or file server but not multiuser serving. The hardware is limited. Also, they are not scalable. Cost would be variable because the hosting tools would need to be built in house. |
| **Client Side** | To develop for Macs, you must have an apple computer running the latest version of XCode. Also, the macOS SDK is in Objective-C or SWIFT. Those are lesser-known languages.  Moderate expertise and time would be required.  Ability to integrate with Apple Watch and other Apple devices would be beneficial. Automatic software updates is also a plus. | Development in Linux would be written in Java, C/C++, or Python. These are all commonly used programming languages.  Maximum expertise and time required. Linux operating systems are more difficult to navigate compared to other OS’s. It is still the most cost-effective because it is open source | Windows is developed using C# or .NET, which are commonly used. There would be no barrier to entry to development since Windows is so well known.  Minimum expertise and time required. | Mobile devices are not designed to be multi-user. However, design a client application for Android or iOS is straightforward.  Android SDK is Java based. So, code developed for Windows and Linux might be able to act as a starting point. |
| **Development Tools** | Can use PyCharm or Eclipse as IDE may require Apple approval for App launch. | Can use Eclipse for Java for IDE, open source and free licensing. | Simple to develop with many IDE options and wide variety of language integrations. | Different IDE and developer paths between amongst different OS (iOS and Android) |

## 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**: Operating System recommendation is Windows because it is the most widely used. It also has many development options and a large support community. Using Windows as the operating platform will allow us to engage with a larger customer base because of its familiarity.

1. **Operating Systems Architectures**: Windows NT, the base of Windows Operating Systems, is a two-layer hybrid platform of physical (user) and virtual (kernel) architectures. The user layer has access to the necessary resources to run the applications and interact with the I/O stream. The kernel layer can access full hardware and can pass resources to the user layer.
2. **Storage Management**: We can recommend OneDrive to users with low storage capacities. Microsoft Account OneDrive offers users a capacity of 5GB of cloud storage for personal data. If the user is an Office 365 subscriber, they get 1TB of OneDrive storage to save data and possible offset the required space for the game.
3. **Memory Management**: Windows utilizes an internal feature called Windows Memory Management. This feature takes a portion of storage space addresses it for virtual RAM. The system will move processes that are not currently being accessed to this virtual memory to protect the live processes on the physical RAM.
4. **Distributed Systems and Networks**: Even though a release on gaming consoles is not yet planned, it would be beneficial to release and distribute “Draw It or Lost It” on the Xbox app that is in Windows. This can allow for safe distribution and sales from a trusted platform that is already successful within Windows.
5. **Security**: Features like Windows Hello and Microsoft account to login requirements protect the system from unauthorized users. Windows Defender is a standard when it comes to antivirus software, and it comes installed on all Windows devices.