

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.3

## 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 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this revision> |

**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 game “Draw It or Lose It” will require the ability to have one or more teams involved, with multiple players, all with unique identifiers. This means during the one and only instance of the game the teams and players can be stored in a list. When a new team or player joins the game the application will search the list to ensure that team or player name does not exist yet. If the name does not exist it will be added to the list within the application so that the next team or player cannot use it.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Since the application will be in a web-based distributed environment the application will not be available on mobile devices. This also means the users will be responsible for ensuring there device is configured to support the application within their environment. Additionally, once the environment of the application is decided the development must be tailored to the new environment to ensure proper functionality.

## [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 diagram below describes the “Gaming Room” software application. The diagram demonstrates the use of inheritance, with the Game, Team, and Player classes inheriting attributes and methods from Entity. The three classes will have access to the “id,” and “name” private attributes via the method the classes will be inheriting. There is association between “GameService,” “Game,” “Team,” and “Player” represented by a straight line. Currently, the “ProgramDriver” uses an interface of the “SingletonTester” to ensure only one instance of game can be created at a time. The entirety of the application is surrounded by the box labeled “com.gamingroom.” This shows how the application is portable as long as the entire application is located in that single package.

"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** | Included in Mac server is Profile Manager, Open Directory, and Xsan. Terminal is available for easy setup and changes. User-friendly GUI available. Apache HTTP built into Mac for websites. Powerful security features. | Stable services with advanced permissions. Like Mac server setup can be done within a terminal allowing great control. Expertise maybe limited yet this system has large community support. Powerful security implementation. | Windows allows a terminal style command-line such as Powershell. Permissions are limited. System supports a great amount of applications compared to other systems. | IOS type servers allow for many different applications based on user’s specifications. |
| **Client Side** | Option is expensive. Minimum amount of time required. Specific expertise needed for Operating System option. User friendly interface. | Minimal cost due to licensing of operating system. Specific expertise and larger amount of time required, compared to other options. | Option is on the more expensive spectrum of operating systems. User friendly, with minimum amount of time required. No specific expertise required due to user popularity. | Difficult setup compared to other operating systems. A large time requirement due to setup difficulty. Updates and maintenance can be performed anywhere. Specific expertise required. Doesn’t support applications from other platforms, efficiently. |
| **Development Tools** | Recommended programming language is Swift, though Mac also supports Python, C++, Java, Objective-C, Perl, and more. Available IDEs include Visual Studio Code, Xcode, Eclypse, and more. Other custom Mac tools include but aren’t limited to Interface Builder, and Apple Development tools. | Relevant programming languages include Bash, C, Assembly language, C++, Python, Java, and more. IDEs are similar to Mac including Eclypse, Pycharm, Visual Studio Code, and more. Development tools include debuggers, Vim, GCC and more. | Relevant programming languages are a combination of C, C#, Asm and C++. Also support is Python, Java, Javascript, CSS, and HTML and more.IDEs include Visual Studios, Visual Studio Code, Pycharm, Notepad++, and more. Other tools include Powershell, Windows package manager, Windows community toolkit and more. | Relevant programming language for mobile devices depends on the user. Some may include Java, Objective-C, Swift Mobile, HTML5, C, C++ and more. IDEs include cross platform tools such as Kivy, and Beeware (for Android). Another tool is Transpiler, used to transfer a source code language into another programming language. |

## 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**:

The recommendation, due to the size of the application not being too large, is to use a Linux server. With the community support, cost effectiveness, and control over a Linux server, this option would do the job effectively, efficiently, and without much cost. It is even capable of running windows applications.

1. **Operating Systems Architectures**:

This operating system consists of the kernal (the operating system programming), a slew of system libraries, utilities, hardware, and the shell. Linux can be used on various types of hardware and utilizes a shell to interface with the user via command line or GUI (graphical shell).

1. **Storage Management**:

For storage management, Kubernetes is a cost effective option. Also being open-source allows for easy licensing, and it is free. It is capable of being automated which would free up man hours and makes this option very customizable.

1. **Memory Management**:

Linux memory management is very complex and configurable. Since the memory management uses its own form of technical terminology, familiarization would be necessary. Although terminology may cost hours and expertise, the settings can easily be accessed via the Linux shell/terminal located in the root directory.

1. **Distributed Systems and Networks**:

The network should be setup with a backup server, in case of disruptions or outages. Due to the high customization the system can be set to switch to the backup Linux server if the primary does go down for any reason. Personnel can be ready to reboot the primary server if no automated reset is yet available. This secondary will allow games to continue without disruptions, to include any required system maintenance.

1. **Security**:

Due to the Linux community being so robust, security is priority with a Linux operating system. There are teams within the Linux community always improving it’s security features, and capabilities. Additionally, the use of Kubernetes will ensure no direct access to the application is seen by the users (client systems). This reinforces security of the game application, making it far more difficult to exploit possible existing bugs or memory leaks in the application or operating system.