

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 2.0

## 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 | 03/27/22 | Spencer Gatten | First Revision |
| 2.0 | 04/9/2022 | Spencer Gatten | Added Recommendations Section |

**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 application will be a four round, one minute guessing game. The player will look at a picture being sketched and guess what it is. The client has requested some specific properties/capabilities in the game “Draw it or Lose it.” The client wishes for there to be unique player names and Game ID’s. The gameplay type is multiteam. There can be only one instance of the game in memory at any given time. In order to accomplish this, there must be distinct identifiers linked to each instance of a game, player, or team.

## [Design Constraints](#_2et92p0)

The application must have the ability to have one or more teams. Each team consists of multiple players. Both the game ID and team name must be unique; the player will have the ability to check if a team name is in use. Only one instance of a game can exist in memory at any given time. In other words, there can not be two games being played at once.

## [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 UML class diagram includes some classes for The Gaming Room. The UML diagram demonstrates many effective principles of OOP (object oriented programming.) The three classes, “Game,” “Team,” and “Player” exhibit the principle of inheritance. Notice that the same attributes also can be found in the “Entity” class. This helps for the related classes to inherit from the entity class as there is an indirect relationship between, they themselves and the entity class. The “Player”, “Team,” and “Game” classes exhibit an “is a” relationship to the Entity class. The “Team” and “Player” classes exhibit a “has a” relationship to the entity class. Another OOP principle which can be found in this diagram is that of encapsulation. “Entity” protects data input into the program as it limits access to the public methods in the application. The end result: The privacy of Player information is protected.

**"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** | Costly. Has a very effective GUI. Good ease of use for developers. The workstation allows for server configuration and accessibility. | Inexpensive. There is difficulty navigating the platform. The command shell exists for accessibility and simple server con. | Expensive. Very user friendly not unlike Mac. There are multiple available software options. The command prompt can be used. | Very cost effective. Device specifications vary between users. There would be challenges creating a game that is compatible with all |
| **Client Side** | The time to support Mac users is about average. Some skill is required. Windows and Mac are both more expensive than Linux. | The time to support Linux users is above maximum. Linux requires some mastering. The cost is the most compelling feat. | The time to support Windows users is at a minimum and proficiency is not required. The cost is the least compelling feat. It is similar to Mac. | Maximum skill/time is needed to support diverse mobile devices. Mobile operating systems can be difficult to run on other devices. |
| **Development Tools** | Like many other OSs, the most common IDEs and programming languages that can be found on Mac are CSS, JavaScript, HTML, Python, and many others. Some tools include libraries for different languages, Visual Studio, Eclipse, etc. | Some languages include Java, Ruby on Rails, CSS, Python, HTML, JavaScript, etc. Tools for development include Visual Studio, nodejs, Github, command prompt, and Repl.it. | Languages on Windows include Ruby on Rails, Java, C++, Python, C#, HTML, and JavaScript. The development tools include Eclipse, Visual Studio, Repl.it, and Command Prompt. | Mobile devices support diverse languages such as Python, CSS, Java, HTML, JavaScript, Ruby on Rails, php, C++ etc. The development tools for mobile devices include Github, Repl.it, nodejs, Visual Studio and power shell command prompt. |

## 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 would suggest the client choose Windows as a first operating platform. There is very minimal expertise required and cost is not an issue. There is no shortage of Integrated Development Environments (IDEs) on Windows.
2. **Operating Systems Architectures**: All windows-based applications use Windows services. Each application is consequently able to use a Graphical User Interface (GUI). Accessing system resources using the GUI is helpful. The system resources include messaging, graphics and multimedia, and web services. These system resources can be accessed using a server or a user account. The GUI can be used for more than simply accessing system resources.
3. **Storage Management**: Windows 10 comes with storage sense. Storage sense is a feature that allows a user to manage files and applications on the hard drive. It also allows one to see how much hard drive space is unused and how much is used. For apps you can choose which save locations to save to which makes them easier to find. Windows 10, like many other modern operating platforms, allows you to save to the cloud (storage via internet). Projects won’t get lost or accidentally deleted thanks to the built in storage system.
4. **Memory Management**: The memory management options offered by Windows boast both a virtual and physical address space. This allows for up to 4 gigabytes. This truly means applications will run smoothly. We will need to create a database full of pictures. The trick here is we can store them outside of the default “Pictures” folder for enhanced security.
5. **Distributed Systems and Networks**: Because each operating system is different I thought it necessary to seek out a way to publish the game on all of them. I found a nifty little system called Develop 4. This is a cross-platform game creation system. This IDE can be run from any device. Creating the game is obviously the difficult part, but once it has been created you simply export the game file onto the Web, Android, or iOS. Develop 4 will help the game greatly with dependencies. Obviously, the company will need to ensure their servers are robust enough to support large player volumes. Backup power will be necessary in the event of power outages.
6. **Security**: Windows Oss give clients the user account control settings which secure data going into and away from the system. Operating platform modifications can only be made by users with administrative status. Windows has a built-in anti-spyware program which keeps unwanted or malicious software from engaging the system. Windows also has VPN options. This helps to protect client information/data and history from being used for malicious purposes. It also employs the RDP stream cipher which encrypts small amounts of data such as social security numbers, passwords, and credit card numbers. Windows also offers DirectAccess for access to and from remote locations and work sites. This feature uses auto-encrypted ESP and authentication when a user connects to a business network.