

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 |
| --- | --- | --- | --- |
| 3.0 | 2/19/2024 | Logan Riedell | Updated Recommendations |

**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 client would like a game called Draw It or Lose It inspired by another game by the name of Win, Lose, or Draw. We propose a web-based game application.

## Requirements

Needs to be web-based. Hardware will need to be discussed. Needs to have large supply of drawings.

## [Design Constraints](#_2et92p0)

Games have 4 rounds. Each round lasts 1 minute. Games should support multiple teams with multiple players on each team. Each team, player, and game should have a unique ID to identify them. Only one game at a time.

## [System Architecture View](#_ilbxbyevv6b6)

Client will need a server to run the program and support multiple users and server files. Client should have high upload speeds so as to have a sufficient connection to its users. Server should have backup power to take care of interruptions in power supply.

## [Domain Model](#_8h2ehzxfam4o)

The Entity class is the base class that game, team, and player extend or inherit from. This is so the id and name parameters can be used with these child classes as well. Each class has its own set of variables or properties to describe them. Each class has their own methods or actions to facilitate different operations specifically pertaining to their instances.

**"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** | A lot of people enjoy the look of the Mac user interfaces which makes it easier to navigate and set up. Mac’s would use low power but may not have enough for large scale applications. Mac OS comes with a fee. | Linux is used in many servers and functions. Linux is very reliable and has many developer tools. Linux is also very customizable. Linux is open-source and free. Navigating Linux can be challenging for beginners. | Windows is a popular OS that has plenty of developer tools and is compatible with most 3rd party applications. It is used for servers as well as personal computers. Some people may not like the file system navigation. | Almost everyone has a mobile phone nowadays to make calls receive and send data. Developing products for mobile applications is a must in today’s world as there are many people who do not even own a personal PC but own several mobile devices such as a tablet and phone. Mobile devices tend to have less power and memory. |
| **Client Side** | Mac is very reliable and has many features that make interacting with the OS enjoyable. Mac may not have the compatibility and freedom Linux and Windows do but has several 1st party applications for many needs. | Linux is free. It may not have all the compatibility of Windows or Mac. Linux is very customizable has an intuitive interface. | Windows will use a browser like Edge but users can download others they prefer depending on that application the web page will run based on internet connection and the clients own set up. Developing for a Windows system should be comparable to Linux or Mac. | Mobile devices usually have different screen sizes that need be considered while developing applications. They also have less capabilities due to their small size and applications and user interfaces have to develop with this in mind. |
| **Development Tools** | Many of the same tools and IDEs available for Mac are also available for other operating platforms. Specifically for Mac users, a proprietary option is called Xcode. Xcode offers numerous capabilities, allowing users to easily and efficiently create graphical user interfaces and websites. | A good IDE for the Linux platform is Eclipse. Due to its features and resources, which are highly compatible with Java, it is very popular among Java developers on any platform. Its debugging tools and WAR file manipulation capabilities make it an excellent fit for Java development. | For Windows, Microsoft's Visual Studio and Visual Studio Code are suitable IDEs for almost all programming languages. They offer numerous features to assist with various types of development and are continuously updated with more. For C# developers, there is no better IDE than Visual Studio 2022. | Most languages have specific frameworks and modules that assist in developing for mobile devices, whether it be for Android or iOS web-based applications. Most IDEs have integrated tools to help visualize applications on mobile screens. Browser software can also be a powerful tool for this purpose. Browsers enable a developer mode to preview what sites will look like on various devices. |

## 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 recommend a cloud-based game to optimize scalability making adding new features, memory, and storage to allow for changes to the game, reliability, increasing user count, and cross compatibility between platforms**. Platforms like Microsoft’s Azure or Google’s Cloud Platform can host the game and provide extra tools for managing resources.

1. **Operating Systems Architectures**:

Using a cloud-based model the provider handles most of the miscellaneous processes while the owner’s development team can focus on the game’s applicability and functionality.

1. **Storage Management**:

**Cloud**-based systems are good for scalability meaning as the game’s functionality grows and new ideas users or information is needed to be stored cloud memory can be increased or decreased based on demand. Cloud providers use top of the line processing servers and have ample resources to handle many users. Redundancy storage can ensure that connectivity issues, hardware failure, and outages do not lead to a loss of data.

1. **Memory Management**:

Cloud based systems have greater processing capacity and memory capacity than local machines they specialize in application hosting so memory load can be scaled to make sure the application runs smoothly on any system.

1. **Distributed Systems and Networks**:

**Cloud based systems can balance the demand of the applications to multiple servers so that no one server gets overloaded**. Content delivery networks can help keep the resources closer to the end users which means faster delivery of data. Storing the game on the cloud also ensures that everyone can access the game no matter what platform using a portal such as API’s or a browser since any system with a connection can access the information allowing more user’s to play the game.

1. **Security**:

**Cloud providers offer many types of tools for data encryption. Encrypting data can ensure any bad actors cannot see the information unless given access to the decryption methods**. User authentication can ensure that whoever is trying to access resources in storage is supposed to have access. Using a multi-factor authentication user will have to go through many different methods of authentication to access their profiles and data. Regularly updating security software and doing regular vulnerability testing can ensure that the application is safe from bad actors.