

draw it or loss it

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 6-16-22 | Jeffrey scroggins |  |

**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 company was asked by The Gaming Room to design a web-based version their Android-based game Draw It or Lose It. The game should allow multiple teams with multiple players per team. Each game-instance, team, or player should only occur once.

## [Design Constraints](#_2et92p0)

The Gaming Room has an existing Android-based deployment of the Draw It or Lose It. CTS has been asked to extend this to the web and thus the tech stack needs to be compatible with web-deployment - Java has been selected for this purpose. Java being the native Android SDK language should ease this new deployment effort.

## [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 designed UML shows multiple Object-oriented program techniques. Polymorphism and inheritance are used in the extension of the Entity class and the overloading of constructors. Encapsulation and Abstraction are also apparent in the techniques used to add teams. The application consists of a main driver class that will be used to initiate the creation of the games, teams, and players.

**"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** | Macs can be used as a server, although the licensing is expensive, and you must have Mac books to develop. | Linux is the most popular webhosting OS. Because Linux is opensource, maintenance and license costs tend to be cheaper | Windows servers are nice because they are GUI based and many applications used in the office will also run on the server  License costs, typically per user, tend to be very high with some going into the thousands of dollars | Mobile devices can be used as a personal webserver or file server, but they are not equipped for multi-user serving.  The downside is that hardware is limited and cost is unkown |
| **Client Side** | Mac has good and easy to use SDK’s but the thing is you must have a Mac Book to develop for Mac. This will add up cost and requires someone that has developed swift. | Development in Linux should be straightforward as Java or C/C++ or Python could be the language of choice | Windows is typically developed using C# or .NET which are both common. There would be no barrier to entry to development of a Windows client application. | Mobile devices are not designed to be multi-user. However, design a client application for Android or iOS is straightforward. |
| **Development Tools** | A Mac Book that has iCode on it. All coding will be done using swift. | Linux development may take the form of C++, Java, or Python. | Windows is primarily developed using C# | Android SDK is Java based and the most widely used Android IDE is Android Studio  iOS’s languages are almost exclusively developed in XCode |

## 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**: Regardless of the software chosen using Linux-based servers should be the solution. Linux servers reduces the costs and does not limit access to data centers the same way that Windows servers can.
2. **Operating Systems Architectures**: The suggested architecture would be for a backend server that manages the game environment and rendering. Because Draw It or Lose It gameplay is not dependent dependent on streaming so it doesn't need low latency.
3. **Storage Management**: Unless the Game Room wants to purchase their own hardware no decision on storage medium, for hard drives or solid state driver will have to be chosen. Either one should should do well.
4. **Memory Management**: Linux uses the concept of page cache for data stored in main memory virtual memory for any pages allocated. Linux uses demand paging which allows for lower memory usage.
5. **Distributed Systems and Networks**: Uptime considerations and outage prevention are the reasons so many applications are being built in cloud native architectures. Both cloud and hardware will allow transparency and prevent outages.
6. **Security**: Protecting user information is paramount. This should be done on a local server if possible, rather than on the distributed systems to prevent unauthorized access. Member access is one of the reasons I recommended Linux. Even if this information is stored on the primary server, Linux allows the administrator to build in protections. While windows offer some protections. Linux offer file protections over three layers aswell allows us to more customizations.