

Draw It or Lose 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 | 01/20/22 | Emily Wood | Initial Design Document |
| 2.0 | 02/06/22 | Emily Wood | Development Requirements |
| 3.0 | 02/20/22 | Emily Wood | Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose it, which is currently available in an Android app only. The game will have more than one team with multiple players per team. A drawing will be pulled from a library and rendered at a steady rate for thirty seconds while a team tries to guess the drawing. If the team does not guess correctly after the thirty seconds, the other teams will have fifteen seconds to offer one guess each. This will last for four rounds.

## [Design Constraints](#_2et92p0)

* *Game must be a web application.* Since the game will be web-based, it will be accessible to any operating system if there is a web browser. Since it won’t be native to a device, we must not include any features that require hardware such as a camera or GPS.
* *Only one instance of the game can exist in memory at any given time.* Unique identifiers must be created for each instance of a game, team, or player.
* *A game will have the ability to have one or more teams involved.* Each team name must be unique.
* *Each team will have multiple players assigned to it.* Players must stored to each team.
* *Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.* Must be able to iterate through team names to determine if it is already in use.

## [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 classes Game, Team, and Player are all child classes that inherit from the parent class Entity. They will all inherit the public functions from Entity. The GameService class can have zero or more games associated with it. The Game class can have zero or more teams associated with it. The team class can have zero or more players associated with it.

**"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** | Its origins are in Unix, so it has many of the same advantages of stability as Linux. However, it does suffer from being proprietary and requiring licenses. | Less user friendly. Open source means more cost effective. Much more reliable and stable and easier to modify. | More user friendly and can integrate with other Microsoft products. Usually has more features available. | Must be device specific. Since there are multiple devices available, it may require more work and more cost involved. |
| **Client Side** | Users are used to paying for apps on the App Store, so potential for profit is greater. Approval process is long and drawn out. | While powerful, is less familiar to clients and may require more work. Costs may be down, but time and expertise go up. | Most home desktops run on Windows, and therefore most people are familiar with it. The costs of licensing may increase, but the overall work should not. | Native apps run faster and can access the system resources (camera). Tend to be more secure and can work without an internet connection. Can be more expensive to develop and maintain and may not be approved by an app store. |
| **Development Tools** | Java, JavaScript, Swift, Objective C. Mac specific IDEs include Xcode, Espresso, and Coda | While it supports most languages, the preferred language is C. Because it supports most languages, most IDEs are available to use. | Java, JavaScript, Python, C#, .NET, C, C++. I’m partial to Visual Studio, however Eclipse can be used for Java. | Objective-C, Swift, Java, Kotlin, C#, Xamarin. IDEs such as Visual Studio, VSCode, and Eclipse. |

## 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 that we go with Linux. Linux can be the most cost-effective and much of Linux is now supported by and integrated into Windows. Scalability is innate to Linux and added by default. It is a more widely used platform as of late and standard UNIX development tools work on Windows.
2. **Operating Systems Architectures**: Linux is based on the Linux kernel. The source code can be cross compiled for many computer architectures. By using an IDE, debugging and deployment can become a single step. This makes adding another architecture to application development easy.
3. **Storage Management**: Storage in Linux acts as a filesystem tree. Disk partitions are a way of breaking up storage into smaller, useable units.
4. **Memory Management**: Memory management in Linux has two components. It deals with allocating and freeing up physical memory. It then deals with virtual memory. Virtual memory is applied to the running process.
5. **Distributed Systems and Networks**: A distributed system is a collection of processors that do not share memory. Linux can be used as a network operating system. This can provide file sharing and be helpful when multiple users are trying to access the same game and guess the same drawing. Each computer acts autonomously, but it is aware of the network and able to communicate to others. Developers commonly use Manjaro because it can be customized and comes with all the tools required to compile and develop.
6. **Security**: Authentication and access control are two pillars of Linux’s security model. Unique user identifiers and group identifiers help provide a barrier to user information to only those who require access.