

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)5

[**Recommendations**](#_m8aleynsvzvc)7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <11/12/21> | <Mariam Haji> | Summary of Gaming Room “Draw It or Lose It” software on several platforms |

**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 Gaming Room. The Gaming Room wants 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 purpose of this game is multiple teams consisting of several people going four rounds at per minute each. When a picture is pulled from a library of images a team guesses till the time runs out. If not answered each of the other opposing team members gets 15 seconds to answer.

## [Design Constraints](#_2et92p0)

The game needs to run on multiple platforms, Usernames have to be unique as well as the game can only run on one instance at a time. The game will also need more than one team and a team will need to have more than one person in order to play.

The gaming room would like the application to run on all devices so the development team needs to work on an iOS app, Windows app, Linux app as well as MacOS app and any other device however there is already an android device version currently functioning. In order to do this the team can either use the attributes from the android app and use inheritance or re-write the code in other languages.

## 

## [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)

Entity creates a relationship between Game, Team, and Player class. This means they all inherit or get information from Entity. With UML we can show this with inheritance. So, each class will share common references like “name” and “id”. Making Entity a superclass. When we look at their relationship, we see Team and Player is a “has a” type. While Game has a Team and GameService has Games. When we use UML, we call it aggregation (HAS-A). When a user “has a” I mean it's an instance of one class and has a reference to an instance to another class. When we look at this diagram, we see GameService has a reference of Games, Games a reference of Tea, and Team a reference of Player.

**"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) | | | | |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| **Server Side** | Mac web hosting is the best amongst the lot since the software is secure and almost impenetrable as well as the hardware is durable and can perform multiple tasks.  Mac hosts run Apache servers similar to Linux which allows for basic web code plus software like WordPress and cPanel to run smoothly.  One of the disadvantages of web hosting on a Mac would be the high prices hosting companies charge | Since Linux is an open platform, it is more popular than Mac and Windows, code can be used, modified and redistributed for free of cost in both commercial and non-commercial abilities.  The number of resources needed to run Linux is often much less than Windows or Mac  Linux web hosting may not be the most secured compared to the others but if there are plenty of options for reliable hosts in the market. | Windows work best with databases mostly one of the best qualities of a windows machine is the capacity to incorporate data bases well.  Due to popular usage of windows user’s web hosting on windows is reasonable compares to Mac in terms of pricing.  One disadvantage of using windows would be response time may lag and applications and processes may either fail to run or run slowly.  Another disadvantage of windows would be security issues since windows are very common and easy to hack. | Mobile applications require shared hosting and they cannot be standalone. Ideally for the web-based game the mobile app can be hosted on a shared hosting or a virtual private server.  Advantages would be is the application is hosted on a shared hosting the cost may be cheaper however the security would not be competent since it’s a shared server with hundreds of other websites.  Alternatively hosting the application on a virtual machine may be expensive since most VPS plans are mainly based on the amount of dedicated RAM available to your virtual machine. |
| **Client Side** | The application development process on a Mac may be costly than windows or Linux since ideally, we could need to deploy the app on cloud in order to achieve cross-platform portability. | Since Linux is open source the development process is easier and cheaper than a Mac as well as more secure than windows. Cross development of the applications won’t cost as much as it would on a Mac | Windows being the most common operating system for developers there will be minimal development expertise as any programmer can use a windows machine another advantage is IDE’s such as Visual Studio Code are easier and compatible to run on a windows machine since they are Microsoft owned. | For mobile platforms these run different unlike Mac, windows or Linux OS and will require expertise in mobile development. Since Most iOS apps can't run on Android without significant changes and vice versa.  The mobile app development process is quite different depending on which OS you're developing for attracting more cost to cover on resources required such as license, training or hiring of new developers. |
| **Development Tools** | Recommended development tools would be to use Java, C or Python to write the app and the app can run on Visual studio code which will attract licensing cost on a Mac but not necessarily on windows or Linux | Recommended development tools can be IDEs such as eclipse, Visual Studio code and or Xcode and recommended languages would be  Java, C or Python | To run Java on a windows machine IntelliJ would be ideal however I highly recommend Visual studio code as it has a vast library for Java, C and Python which are ideal for cross-platform programming | Recommended IDE, would be Android Studio, or a more generic, nonoptimized IDE, such as Eclipse for the android app and Xcode for the iOS app.  Recommended languages would be Java for android and C for iOS. |

## 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**: Now that we have established that Draw It or Lose It will be a cross platform application. I highly recommend Linux as the operating platform. Some of the advantages a Linux machine would have over windows or Mac-OS would be versatility. Due to its open-source nature, Linux is highly customizable, with a wide range of features that make it easy to incorporate additional features and functionality into the Draw It or Lose application across all devices. In addition to this the operating system is also cost effective due to the license free nature.
2. **Operating Systems Architectures**: The Linux operating system's architecture mainly contains the Kernel, System Library, Hardware layer, System, and Shell utility. While using Kernel mode during development the developer gets complete access to every computer resource. Kernel executes all the processes and facilitates various services of a system to the processes. Also, it facilitates secured access to processes to hardware hence it is very fast and efficient.
3. **Storage Management**: Although most machines come with internal storage drives that are either SSD or HDD. I would recommend cloud storage such as amazon S3 cloud storage since it’s easily accessible, scalable, has great security features and high-performance services.
4. **Memory Management**: Linux Memory management services are built on a programming foundation that include outlying device called Memory Management Unit (MMU).

MMU translates physical memory addresses to linear addresses used by the operating system, and requests a page fault interrupt, when the CPU tries to access memory that it’s not entitled to.

A basic unit of memory under Linux is “page”. All available physical memory is organized into pages towards the end of the kernel’s boot process. Page size used by Linux is about 4096MB

1. **Distributed Systems and Networks**: Creating Draw It or Lose for cross-platform I would suggest having one codebase for the application since this will make the application more stable and easier to maintain even during upgrades. Hosting the game files and database on cloud with reliable security to user data, development teams can make changes on one codebase and deploy it to all platforms at once saving both time and money. In addition to this I also recommend the use of a version control system such as Github or Bitbucket as this will assist to track and mange changes to files, plus manage releases and patches for each platform.
2. **Security**: Linux is an open-source operating system that allows highly modular, modifiable server infrastructure, with customizable user privacy and security elements. If Gaming servicing opts to use the S3 cloud storage, the service has encryption features and access management tools that enhance numerous security optimization modules to add privacy and security functionality to the system.