

Draw It, Or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 |
| --- | --- | --- | --- |
| 1.0 | 11/7/23 | Daniel Sholes | Executive summary, requirements, constraints |

**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 seeks to launch a game called Draw It, or Lose It, on a web based environment. It exists already on Android and we have been requested to bring it to the web to reach more players. The game must only have one session open at a time, and have multiple teams of up to four. The game consists of stock images being used as clues, and the teams compete to find the phrase based on them.

We are using a singleton pattern to prevent multiple game sessions.

Java is the language in which Android (for which the existing app currently runs) so it will be used in this web application as well.

## Requirements

Requirements are:

* One or more teams may take part
* Each team may have multiple players
* Player and game must be unique
* Only one instance of the game can exist at a given time

## [Design Constraints](#_2et92p0)

We need to migrate this game from an Android mobile system to a wider web system. The fact that the original game was on Android, it stands to reason that we would keep the application in Java. We need to select a server and database system that will support this game. We also need to keep budget in mind, as well as schedule.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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.**

## The game starts in a singleton from GameService. The singleton makes sure there is only one instance of the game at any given time. GameService’s constructor is private, thus stopping any instances from being created. The GetInstance method creates an instantiation, but also checks to make sure that there are no other instances already active.

Games can be created once the GameService is running, and each game can have teams, which in turn have players.

Games, Teams, and Players are all subclasses that inherit from Entity. Entity has two private attributes: id and name, further making sure that everything is unique.

The UML diagram shows several object oriented elements. The most obvious is the inheritance of Game, Team, and Player from Entity. Polymorphism is used in the overloading of the constructors. The process of creating and adding teams demonstrates abstraction and encapsulation.

## [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 well known server option for Mac is OS X Server. OS X Server is specifically designed to be installed and run on Apple machines. It allows for the creation and use of file-sharing, email, caching, XCode, and wiki servers. | Linux is a very popular option for servers and hosting. With its open source status and low maintenance cost, Linux is, in many ways, a superior choice for servers. Popular Linux servers include Bluehost, arguably the best hosting server available. | Windows has a server system known as Windows Server. Windows Server is the platform for building an infrastructure of connected applications, networks, and web services. Microsoft has also implemented a service called Azure, which seamlessly connects Windows Server to the cloud. There are a number of services available for Azure users and one can take advantage of a 12 month free promotion. | Mobile devices lack the power and capability to host multi-user systems. That being said, there are other server options available that allow hosting of multiple mobile apps. |
| **Client Side** | Native MacOS development requires a Mac computer, typically running an IDE called Xcode. In general the code would be in either Objective-C or, more recently, Swift. These two languages are less known than some of their counterparts, and focus specifically on Apple hardware. | Development on Linux offers programmers many good options, including a wide selection of languages. These languages include Java, C, C++, and Python. All of these languages are very popular (some go so far as to call Python the language of the future) and do not require any special expertise | Windows has access to a number of options. A very popular choice is .NET Core and the .NET Framework. These systems work with languages such as C# and VB.NET to create a layered approach to development wherein the language of choice compiles to a common language compatible with multiple systems. | Mobile devices typically fall into one of two categories: Android and iOS. Android is built on Java, with support in a newer language called Kotlin. Since Java’s VM setup allows it to be developed on many different systems, it is easy to branch off here. iOS is built largely on Swift and has the same limitations as MacOS systems. |
| **Development Tools** | The two most common languages for Apple hardware are Objective-C and Swift, usually developed in an IDE called XCode. XCode is currently in its 15th version and costs $99 a year for a basic license. | Linux is open source and has a wide variety of development environments available. For languages such as C and C++, code editors like Emacs and Vi are popular. For those looking for more options there are editors like Atom. Eclipse is an excellent choice for Java programming, and PyCharm is well known among Python developers. For those looking for something flexible, VSCode works with many languages, with many plugins and configurations. | Windows has a number of development options, both for languages and development environments. The most popular may well be the aforementioned C# and .NET Core, but other languages such as C++ and Python work as well. For environments, the most popular is certainly Visual Studio, now available in the Community Edition for free. VS is a very powerful environment with many options and extensions available. | The most popular IDE for Android is definitely Android Studio, put out by Google at no charge. Android Studio allows for simple, rapid development for Android devices in either Java or Kotlin. To develop in iOS, one needs a computer running MacOS and an active license for XCode, with the same provisos as other Apple based development (MacOS). |

## 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**:

The ideal operating platform for a web app of this kind is definitely Linux. Linux is the most popular server for this kind of application, and with good reason. Its open source qualities make it easy to work with, as does its ability to work with multiple computer languages. Furthermore, it is very easy to secure, and security is a must forthis client.

1. **Operating Systems Architectures**:

The back end of this software should be on a Linux server, while the front end would work on multiple systems. The front end could work through a web browser, or an on site client to play the game

1. **Storage Management**:

There are a few options for storage management. The most common is probably simple hard drives (HDD). If we are looking for a bit more speed on the client side, a solid state drive might be appropriate (SDD). Alternatively, we could rely on cloud storage.

1. **Memory Management**:

Linux uses Paging for its memory management. In Paging, computer stores and retrieves data from the secondary storage to the main memory of the device. The secondary storage is subdivided into “pages” that are used for storing and retrieving data. Paging is an important part of virtual memory implementation. Since Linux is the system we will be using for the server, it fits that we will also use its Paging for memory management.

1. **Distributed Systems and Networks**:

The front end and back end of this software application will communicate with one another using REST APIs. This will allow the information to get from the back end to the front end, such as Android, Windows, iOS, MacOS, or other Linux systems.

1. **Security**:

We will make effective use of authorized security to keep the systems secure. Specifically Role-Based Security. We will embrace the philosophy of Least-Privilege, where users have only the bare minimum of clearance and access they need to do what they need to do, and no user will be given Admin access. We also want to include a firewall to block unfriendly users, and I would recommend a penetration test to check for other vulnerabilities.