

The Gaming Room’s “Draw It or Lose It”.

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 3/19/2023 | Shane Beck | Designed the first version of the document. |
| 1.1 | 4/2/2023 | Shane Beck | Grammar corrections, evaluation completed, miscellaneous changes. |

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lose It, the game where teams will compete to guess what is being drawn, where they will use rendered stock drawings as clues. It was consist of four rounds of one minute each, with the drawing clues being fully complete at the 30-second mark. Each team will have a chance to offer a guess provided the first team fails to guess. Our client, The Gaming Room, requires our help to setup the environment for a web-based version of the game which is currently only available as an android application.

## Requirements

* *The game must be able to have one ore more teams involved.*
* *Each team will have multiple players assigned to it.*
* *The names of the games and teams must be unique, allowing users to check if a team name is available.*
* *Only one instance of the game can exist in memory at any time through the use of unique identifiers for each game, team, and player.*

## [Design Constraints](#_2et92p0)

**The current game is only available as an android app:**

One thing to consider is that the company only has an android version available, this will constrain our ability to transition it to a web-based game, as for one, we will need to make sure the web-based players can continue to play with those on android. It also is a must to keep the languages the game is designed in similar, in this case as android games are primarily programmed in java, using java will work.

**Only one game should run at any given time:**

This constraint is something we have to consider, we will need to make sure the web browser that the game is playing on does not have multiple games running. This is something very important to making sure the game operates as intended and to keep any (possible) cheating from occurring through running multiple players on the same device. This will put some more constraints on how we adapt the game to the web, making the tracking of the user’s IP address necessary, as well as the use of cookies.

**The varying operating systems may impact our ability to allow all users to access the web-based version:**

Something to consider is that not all operating systems have the same software or web browser, so we must keep in mind in order to get the game working for each user we will need to address each operating system and browser separately, adapting the software used as needed. This will impact the development as this will cause the process of building the web-based version much longer than if we were targeting a single operating system and browser.

## [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)

We start with the ProgramDriver and SingletonTester, these two classes are the parts that make sure the program actually runs as intended, without them each other class technically doesn’t do anything. Next, we start off with the GameService, this class while others don’t inherit from it, is where the games functionality comes into play, managing the creation and running of each game, team, and player. Here we can see some OOP coming into play as the GameService is a singleton object that is created to be used by the ProgramDriver. Following such, we have the base Entity class that includes some basic functions and variables that the Game, Team, and Player class all inherit from, which allows is to see the re-usability of code that comes with OOP. The Game class represents each running game that the GameService starts, and within has a collection of Team objects., which also contains a collection of Player objects.

"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, being a more closed and individualized ecosystem than the rest has the disadvantage of less developers using it for server-projects. It does, however, have a lot of development tools, but generally the mac is not used for running a server. | Linux is an open source operating system that has great support for running servers as the developer community is very large around the operating system. Most servers use Linux these days, as such the tools and software are already ready to use. | Windows has a lot of available memory making running a web-based application relatively easily, however as it’s mostly designed for the end-user, it does not have as many tools useful to running a server. | Mobile devices do not have the ram necessary to support the running of a web-based software application, or most applications in general. |
| **Client Side** | Running clients on a mac is a bit more costly due to the prices of a mac system as well as the tools used within. There is also the lack of freedom that the mac has, as such as the operating system is pretty tightly locked down which may affect any other tools we want to use, especially if they aren’t apple native. | Linux has the advantage of being open source as such the tools are pretty much all freely available to be used, so the cost will be minimal. However as linux is mostly used solely by those that are more tech savvy, it may not be the best idea to focus on linux for the client side. | Windows has a large user base and support for pretty much any browser or software that you will need. The cost itself shouldn’t be much at all nor the time as the tools are already available, however expertise to windows-development can be higher due to the many versions of windows to consider. | For supporting mobile clients we must consider that most mobile devices do not have much ram, as such running an application on top of the browser is normally out of reach, as such time will be a much bigger factor due to the optimization you would need. |
| **Development Tools** | Programing Languages: C++, Swift, Java, Javascript.  IDEs: Xcode, Eclipse, Visual Studio. | Programming Languages: Java, JavaScript, C++.  IDEs: Eclipse, and Visual Studio, NetBeans, IntelliJ IDEA. | Programming Languages: Java, JavaScript, C++.  DEs: Eclipse, and Visual Studio, NetBeans, IntelliJ IDEA. | Programming Languages: Java, Kotlin, Javascript, Python.  IDEs: Android Studio, Android SDK, 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**: Linux, being an open-source operating platform allows for easier portability to other operating systems and platforms, as well as a large amount of already-flushed out server-side software. As such it should be used as the operating systems architectures, as the back bone of the servers, etc.
2. **Operating Systems Architectures**: The architecture of the chosen operating system is one that allows for scalability as we see fit, as well as in-depth management of resources. Linux, unlike other operating systems, makes it easier for us to disable everything except for those that are needed so as such we can vastly reduce memory usage using this architecture allowing for easier scalability of our server. Linux also does not require much in the way of actual displays, being easy to operating using a command line via remote access as such physical requirements will be lowered as well.
3. **Storage Management**: Logical Volume Manager (LVM), is a software-based RAID system that works through allowing you to create pools of storage and add more hard drive space as needed.
4. **Memory Management**: The operating system, with it’s open source nature, allows more control over the features you wish to include, so you can disable any software that you don’t need allowing more memory to be available for the game to use.
5. **Distributed Systems and Networks**: It may be accomplished through firstly a central server that is running on one type of software that can be interacted with by each type of client, in this case a server that connects to the browsers of the users through their network and then after determining the clients operating system, modifies the used client-side code to match what their device will support. A big thing is to make sure that we are also checking for the version of the browser that the client is using as well, as some browsers are outdated and can’t use the same software that modern browsers can, in those cases we could switch to a different method to deploy the software to the user.
6. **Security**: User information will need to be encrypted so it’s only accessible by the server and the client that the information pertains to. Linux is great for this due to it’s cryptography software as well as multitudes of security software that will help keep any unwanted actors from interfering with the server.