

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 | 06/17/23 | Matthew Pokorny | Proposal for game software design |

**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)

There is already an Android app called "Draw It or Lose It" from the developers, The Gaming Room. Based on the present game, they want to adapt their web-based game to support various platforms. The program is made up of numerous games that each include multiple teams with multiple players. Each game and team name must be unique for the game to operate correctly. Draw It or Lose It has a vast stock of drawings to call upon for the facilitation of its gameplay. The Gaming Room's staff has no idea how to set up an environment to facilitate gameplay on non-android platforms.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* Android, iOS, and web platforms each have different tools and capabilities/requirements.
* The program should work between each platform for seamless cross-play.
* The capacity to inform the team captain that a team name is already taken and give them the option of selecting another.
* A game will have the ability to have one or more teams involved. Regardless of the platform.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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 ProgramDriver Class contains the main method. The ProgramDriver Class contains the main method. ProgramDriver checks whether GameService already has an instance by using Directed Association with SingletonTester.

Game, Team, and Player classes all descend from the Entity class. The necessary attributes of Entity are inherited by Game, Team, and Player. A Team can have a Player but not the other way around. A Game can have a Team but not the other way around. A GameService can have a Game, but not the other way around. Each game that runs on Game Service must only have one instance active at any given time. At any one moment, each Game can only have one distinct Team. Each team can only have one of each player at any given moment.

**"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 servers, but licensing is expensive, and development requires Apple approved and sold hardware. | Since Linux is open source, development and usage would be free. Linux is well equipped for web-based hosting situations and environments. | Windows servers are easy to set up and use, but they can be expensive due to licensing costs | Mobile devices could be used as servers, but their hardware restrictions would make them unusable for practical back-end hosting. |
| **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. A person with quick development skills is needed for this, which will increase the cost. | The cost for this would be highest in development time, you also must have someone that is used to using Python. | Expertise is probably the highest requirement for Windows. I would strongly suggest utilizing the.NET framework for its capability and security. | You want to find developers that have experience developing mobile apps. Things like user interaction and display need to be handled differently than they would be on the web. |
| **Development Tools** | A Mac comes with iCode, and tools can easily be downloaded from their app store. All coding should be done using Swift. | Python is already installed on most Linux distributions. You could use IntelliJ’s Ultimate IDE to code for this. | The industry standard and ideal method for creating Windows applications is Visual Studio Code. You could use almost any language, but many Windows programs are written in C++ or C#. | You will require an Android Studio expert to create the app for Android. For iPhones, you require a developer who is proficient in iCode and has a MacBook.  Or you can use a multi-platform game engine to have true parity |

## 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:** For Draw It or Lose It's Kubernetes cloud hosting, I suggest using a Linux Ubuntu Server or equivalent.
2. **Operating Systems Architectures:** The Linux kernel has a solid security record.The Kubernetes clusters make it simple to divide up hardware and system requirements.
3. **Storage Management**: You may utilize either an HDD or an SSD storage device, although I suggest the latter. The SSD will enhance the customer experience since users would not have to load photographs onto personal computers as rapidly. I propose establishing a Kubernetes node for data storage and a NoSQL node for gameplay content and user management to optimize storage. Link the URL to the picture's position in NoSQL.
4. **Memory Management**: I recommend adding a load monitor on the system to save money. When usage is low, you may minimize the required RAM, and when usage is high, you can add all that is needed, so you only pay for what is needed to deliver the greatest experience.
5. **Distributed Systems and Networks**: Since this system is in the cloud, if there is a minor outage on the servers, the game would never be forced to halt. You may simply relocate the node towards another server, or if the system fails, another server will immediately start. Going this way will permit you to manage everything except the client end of the game. This will enable every operating system to create a client that can access that data. The separation of functions provided by Kubernetes makes system administration and structure easier.
6. **Security**: I propose that you use a role-based security mechanism. It will best fit your purposes and allow you to separate admin, game, team, player, and user. You may use this to prevent a person from obtaining data that they are not intended to access.