

Draw It or Lost 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 | <5/26/24> | Aaron Rodriguez | Filling out necessary information about the project |

**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 has reached out to us to that we can help facilitate the creation of their game, Draw It or Lose It. It is only available on Android phones, so they are having us make it playable via the web so more people can play on different devices.

## Requirements

* A game will have the ability to have one or more teams involved.
* 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.

## [Design Constraints](#_2et92p0)

The design contraints will be that the game will not be able to be very graphically intensive. Due to the way websites and browsers work, the programming languages for them are not low-level enough to effectively manage memory and interact with the CPU and GPU at the level of AAA games can.

## [System Architecture View](#_ilbxbyevv6b6)

--

## [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.**

Using this diagram, I can see that the ProgramDriver class is the class in which the main method resides, and it uses the SingletonTester class. The Entity class is the base class, in which it holds several data points such as the id, name of the entity. The Game, Team, And Player class all extend the entity class. The Game class contains the team. The team class contains the players. And the Player class containts the player ID and name. The GameService class uses the Game calss and this handles the games, nextGameId, nextPlayerId, nextTeamId, and an instance of itself. All of these classes have getters and setters for their respective fields.

## [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** | Mac may be tricky only because Mac computers tend to be slow. Not always the case but their price to performance is amongst the worst of laptops, so they may be a drag during the game depending on how old they are.  []  An advantage to running a web application on mac is that they have a flexible command terminal to help with customizing the server  A disadvantage is that the mac’s are not very easy to upgrade/swap components | Linux is the odd bunch out of the group. Getting linux to work should be fairly simply as linux can run on anything. Just have to make sure the game is compatible with the browers.  []  An advantage to Linux is that it’s the most widely used OS to run a server as it is very low level and doesnt come with bloatware  A disadvantage to Linux is that it may be difficult to find support for applications on this OS because it’s not widely used on the consumer side | Windows has the advantage of being the most used, so this one will be the easiest to develop as there are not maybe hoops you have to jump through as it is the most supported.  []  An advantage to windows is that it’s very easy to upgrade, and there is a lot of programs and software support  A disadvantage to windows is that it’s not the most secure platform and can easily get a virus(es) | Advantages of mobile is that it has the potential to have the most players as almost everyone has a phone nowadays.  []  An advantage to mobile is that it has a wider reach as most people have phones nowadays  A disavantage to running a server on a mobile device is that it’s almost impossible as the phone is nowhere near fast enough to keep up with the demand of a full server |
| **Client Side** | I think perhaps Mac may require the most expertise; It’s not as widely used as windows and will require more fine tuning to get it to work.  []  I think the time and expertise required for Mac is moderate because it’s a specialized OS that is widely used by alot of people | Much like Mac, linux isn’t used nearly as much as windows in consumer computers, so more attention to detail may be needed.  []  I think the time and expertise for Linux would be expert because Linux is not widely used and supported and thus would require expertise in it | This should be the most cost and  time effective OS to develop the game for. Expertise is minimal compared to Linux and Mac.  []  I think the time and expertise required for windows is miminal because its widely available and supported | Expertise will be of upmost importance; will the previous 3, you are working with a desktop environment and a phone environment is totally different.  []  I think the time and expertise required for mobile devices is expert level because mobile is a specific OS and cannot easily be translated between them like desktop PC’s |
| **Development Tools** | For Mac, Javascript, HTML and CSS can be used to make the game. IDE is subjective as it can just be a plain text editor. For the database, SQL.  []  Mac would most likely used Swift or Java to make the software. The impact that this has is that swift is not a widely used language as it is Apple proprietary’s language. I don’t think multiple teams are needed. There many be liscening costs | For Linux, Javascript, HTML and CSS can be used to make the game. IDE is subjective as it can just be a plain text editor. For the database, SQL.  []  Linux can use any programming language but maybe c++, or java. I don’t think multiple teams are needed. There many be liscening costs | For Windows, Javascript, HTML and CSS can be used to make the game. IDE is subjective as it can just be a plain text editor. For the database, SQL.  []  Same thing as linux. I don’t think multiple teams are needed. There many be liscening costs | For iOS, swift is the language of choice. For Android, Java is the language of choice. For the database, SQL.  []  If iPhone, Swift. If android, java. I think multiple teams may be needed. One for iPhone, one for Andoid. There many be liscening costs |

## 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**: To expand Draw It Or Lose It to other computing environments, I would use Linux for the server as it is lightweight and has extensive server support tools and is what most performance servers use.
2. **Operating Systems Architectures**: From my (admittedly limited) knowledge with Linux, is “modular”, meaning the kernel and system libraries are built independently and thus can be optimized and updated as such which makes it highly versatile and able to adapt to a wide variety of hardware platforms.
3. **Storage Management**: SQL would allow data to be synced and store across different platforms.
4. **Memory Management**: Linux manages memory by using paging, segmentation and virtual memory in some cases which can support smooth operation even under high load.
5. **Distributed Systems and Networks**: For communication between them, using RESTful APIs can enable cross platform communication. Using Redis for caching can also improve performance.
6. **Security**: 2FA to login to accounts as well as end to end encryption and password locked games. Encrypting data coming and going from different platforms is also crucial. Having password requirements will also help.