

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 | 03/19/23 | Ethan Hutchison | Brief description of changes in this revision |

**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)

Our clients The Game Room currently have an app called Draw It or Lose It that is only available on the Android app store. They would like to create a web-based game that serves multiple platforms under the same name. The game will render images from a library of stock drawings as clues and a team will have 30 seconds to guess and if the team fails to guess the drawing, then other teams will have 15 seconds to guess the answer.

## 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)

-Each game must have one or more teams involved.

-Each team must have multiple players

-Game and team names must be unique

-Only one instance of the game can exist in memory at a time

## [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 where all three inherit information from Entity. This means they will share common variables making Entity a superclass. We can also see that that player and team is a “has-a” type and Game has a Team and GameServices has a Game. This means that the class is referencing another class so Team references Player, Game references Team and GameServices references Game.

**"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** | Flexible in creating features and changes using terminal commands. | Flexible with its terminal commands as well. Very cost efficient. | Has more software that can be used than other OS. | Need a strong stable server. Other OS have better specifications. |
| **Client Side** | Need a moderate expertise in using. High cost. | Needs a large knowledge and high expertise to use. Very low cost. | Don’t need to be expert to use. Cost can be like Mac. | Provides flexibility to clients and developers but is more difficult to implement |
| **Development Tools** | HTML, CSS, JavaScript, Java, Eclipse, Python, PyCharm, php, C, C++, Visual Studios, Github | HTML, CSS, JavaScript, Java, Eclipse, Python, PyCharm, php, C, C++, Visual Studios, Github | HTML, CSS, JavaScript, Java, Eclipse, Python, PyCharm, php, C, C++, Visual Studios, Github | HTML, CSS, JavaScript, Java, Eclipse, Python, PyCharm, php, C, C++, Visual Studios, Github |

## 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 operating system that I recommend for The Gaming Room is Windows because it is flexible, easy to learn, and easy to maintain.
2. **Operating Systems Architectures**: Windows has the Windows NT system that is a layered system that has 2 different modes that are user mode and kernel mode. User mode is what basic users of windows will see, while kernel mode is more hidden and deals with most processes. With windows it also allows for customization in hardware and can be used on most machines.
3. **Storage Management**: For storage an SSD or a HDD can be used but a SDD is smaller, faster, lasts longer, and more reliable than an HDD so I would recommend an SDD.
4. **Memory Management**: To help with memory management I recommend adding a watcher because then when gameplay is low the amount of memory being used can be lower and at peak times it can be used much more.
5. **Distributed Systems and Networks**: With windows there may be issues such as network lag, queuing issues and possible server overloading that can cause multiple issues. With a strong server, regular maintenance and the development team working together to try to deal with these issues then I think that the issues can be handled.
6. **Security**: I recommend using a role-based security system with authentications. This way only people with certain roles can do different things and it ties it to the person’s account. For the general security of the servers there will need to be some sort of antivirus and scanning software. There should also be routine checks on the server security. Then finally some sort of encryption for user data.