

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 | 11/10/2023 | Justin St.Clair | The cover page history and summary and more has been changed to reflect the recommendations that were submitted. |
| 1.1 | 11/20/2023 | Justin St.Clair | Updated the client, server, and recommendations based on the updated pros and cons for each OS that I have gained. |
|  |  |  |  |

**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 project is to turn the game Draw it or lose it to run on all types of platforms besides Android. This game is about teams playing against each other trying to figure out what the picture they were assigned to guess within a time limit and it consists of four rounds. If the team that is trying to guess the picture does not succeed the opposing team members each get 15 seconds to guess to gain the points.

## Requirements

The requirements for this program is for it to run on other platforms compared to just being out on Android. Both teams will have multiple players but still remain evened out. Only one game can be taking place at a time for the teams. The game and names of the teams must be unique in its own way that allows others to see if the name they wanted is available or not.

## [Design Constraints](#_2et92p0)

They require it to be web-based for others to have access to the game. It would consists of four rounds one minute a piece. It is to revolve around multiple teams trying to guess the picture was pulled from a library that they must guess right to earn points if not the other team gets 15 seconds to guess. The client wants the picture to rendered at a 30-second mark. Only one game can exist in the memory at any time. The way this can be accomplished is by having special identifiers for the game, team, or player.

## [Domain Model](#_8h2ehzxfam4o)

The entity class is basically our handler for game, team and player class. They inherit the characteristics of that class. Of the four classes that were created for this they all reference each other. The programdriver class is how it was created or built the project to meet the expectations of the client. From the programdriver class we are able to access everything and run the program. With the singletontester it allows us to run it to the design constraints.

**"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** | Pros: Using the apple site, it will allow you to use IOS if you pay $20. This also includes software to help you create a webpage.  Cons: hardware options are limited. Requires Mac system using this. Updates are only through apple. The updates require payment. Highest Cost | Pros: The license to use Linux is free but the price depends on what the server must run for the program. Open-Source software available. Lowest requirements on Hardware. Cost-effective option. The stability is great. Its always updated and guarding against any potential security threats. The open source is not centrally managed.  Cons: Not many people are comfortable using this. Not much software available. | Pros: For windows you must purchase the license to use it. They have multiple packages for the server, and they vary in price depending on what you need to run. It is a simple server-based deployment. It offers a large range of software support. It also offers a wide range of hardware options as well. It also receives quicker updates due to a large userbase.  Cons: The downside is that it is expensive to run but everyone is more comfortable using it. It is less secure than Mac and Linux. It has a closed source. You can only update through Microsoft. It also offers limited customizations. | Pros: This ranges from zero dollars to $320 dollars a month depending on the CPU and the memory that you have to use for the program that you are wished to run. If you have a good reputation for it then you can easily cover the cost and many people use this on a daily basis. Can manage calls to the server side to optimize questions on the database. Stores persistent data. The code cannot be seen by the users.  Cons: Server side would need to integrate with cloud servers or a physical server. |
| **Client Side** | Pros: Since apple is not an open source the cost is quite low but not hardly ever used. Wide range of well supported web browsers. It features many tools for creators to use. Easy cross-browser testing software. It has medium development deployment and time efficiency.  Cons: It requires Apple product with Mac. | Pros: This one is an open source and has low costs. This offers a wide range of well supported browsers that come with many tools that the creators can use. It has a tree file structure accommodation.  Cons: But it requires the user to be educated in how to use it. | Pros: it also offers a wide range of browsers with tools for the creators. Easy cross platform testing is available. Quick development and deployment is available as well.  Cons: More difficult to test for MacOS browsers. | Pros: you have expertise with android app development.  Cons: Difficulty testing in other environments and browsers. It takes longer to develop it. |
| **Development Tools** | Documentation for deployment on Mac. You can easily run this and Linux through a virtual machine. The application must be approved by apple. It is only 99 dollars a year for the apple dev program. It has Eclipse for java and Mac OS X. | You can deploy it at any time. It can run easily on windows and Mac through virtual machines. It has a shell prompt and terminal. It is an open-source community. No license cost. It also has Eclipse for Java. | You can deploy at any time. The documentation is extensive for deployment on windows. Can run Linux through VM. No license cost. It offers visual studio code for JavaScript and it has eclipse as well. | Xcode 12 is deployed to IOS. Its only 99 dollars a year for the apple dev program for IOS. It also has the SwiftUI that you can use. |

## 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**: After doing more research I would still recommend using Linux.
2. **Operating Systems Architectures**: The reason being is that the license is free despite the having to train the ones to work on it. It offers many great software and programming languages to create the program you want to create. It is also not centrally managed. It requires lower hardware requirements.
3. **Storage Management**: The xfs filesystem is the default one it offers and it works for many things. It groups each project in a great location to access in a timely manner. You have access to the things that you need to create what you want too.
4. **Memory Management**: It divides the systems accessible physical memory into pages, with their own physical address. Then it is mapped to a virtual address allowing each process to have its own area that corresponds to the physical memory. The manager of the memory management this is in charge of the allocation.
5. **Distributed Systems and Networks**: Linux uses multiple software components but they run it as a single system. They are also connected by a local network and a wide area network.
6. **Security**: Linux is well known for their security parameters. They are known for rarely getting infected by a virus or malware. So that helps in it protecting their clients information from being taken and used for malicious purposes. It has been proven that their security is stable and it is always being updated to guard against any potential security incidents that may take place.