

The Gaming Room

# **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 | 09/16/23 | James Ferchak | Created the first version of the gaming room. |
| 2.0 | 09/30/23 | James Ferchak | Evaluation |

**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 client wants to have a web-based game that is usable on multiple platforms. The game is based on their current game, Draw It or Lose It, which as of right now, is only available on Android devices. Draw It or Lose It uses a large library of stock drawings. The game will be made using the java language and Eclipse as the IDE. The game will host multiple games happening at the same time. Each game hosts teams of multiple players. Each game and team name needs to be unique for the game to function properly.

## 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)

IOS, Android, and web development require different development software.

The API needs to be able to allow one or more teams from any of the platforms.

Game and team names need to be unique.

The team captains need to be alerted when their team name is not unique, and to allow the team to change their name if needed.

Each game, team, and player instance requires a unique ID.

## [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)

<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.>

ProgramDriver class has the main method. ProgramDriver uses directed association with the SingletonTester to test if a GameService instance is already made. Entity class is inherited by Game, Team, and Player classes. A player cannot have a team, but a team can have a player. The same rule applies to a team and game. A game cannot have a GameService, but the GameService can have a game. The GameService can only have one instance of each game running. Games can only have one unique team at any point. Teams can only have one of each unique player.

**"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 could be used as a server. The licensing is expensive and mac books have to be used for development. | Linux has the right tools to be used for web-based hosting. Licensing is free. Widely used. | Windows has servers and they are secure and easy to use. Licensing can be expensive. | Mobile devices are not well equipped to be good as high end server hosting. As they lack the power the other devices have. |
| **Client Side** | Must have a mac book to develop, this will increase the cost of development and requires someone who is knowledgeable in mac development. | Would have the longest development time. Also requires someone that is efficient in python. | Expertise is the highest requirement, but not as much as the other platforms. | Mobile devices are different from the other platforms when it comes to development. This will require employees who are efficient in mobile development. |
| **Development Tools** | iCode and swift will be used for development. As for the device used to use the products; a macbook would be needed. | Python comes preinstalled on most linux devices. IntelliJ could be used for the IDE. | Visual Studio or Visual Studio Code is the most used IDE for developing windows programs. Any language could be used with Visual Studio. C# or C++ are mainly used on windows. | Android Studiod would be the IDE used for the Android platform. A macbook and swift/iCode would be needed for IOS/iPhone development. |

## 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**:

I would recommend using a Linux/ubuntu server for hosting. Hosting a cloud service would be useful as well.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

The Linux kernel is secure and there isn’t much risk of errors. A cloud service would allow for dealing with software and hardware needs easily due to the two types of needs being separated.

1. **Storage Management**:

An HDD or SSD storage would be good. I would recommend the later option. SSDs allow for the program or server to have quicker access to assets such as the stock drawings.

1. **Memory Management**:

Setting up a watcher would help assist with cost as well the load on the system. When usage is low you can lower the required memory, and during busy times the required amount of memory can be increased.

1. **Distributed Systems and Networks**:

As the system is running on a cloud service, the servers will have to keep running. If the system crashes, another server will boot up automatically. This route of server setup will allow you to host everything needed for the game except for the client end. All the versions of the game will have to access the server information.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

A role based security system would be recommended. It will serve your needs optimally and allows separation from user, player, team, game, and administrator. These different classes of users will not be able to access information that is not for them.