

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.gjdgxs)1

[**Table of Contents**](#_heading=h.30j0zll)2

[**Document Revision History**](#_heading=h.3znysh7)2

[**Executive Summary**](#_heading=h.2et92p0)3

[**Design Constraints**](#_heading=h.tyjcwt)3

[**System Architecture View**](#_heading=h.3dy6vkm)3

[**Domain Model**](#_heading=h.1t3h5sf)3

[**Evaluation**](#_heading=h.2s8eyo1)3

[**Recommendations**](#_heading=h.3rdcrjn)5

## [Document Revision History](#_heading=h.3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/26/22 | Clayton Wyman | Filled in the details for Executive Summary, Design Constraints, and the Domain Model |

**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](#_heading=h.2et92p0)

The goal of this project is to create a multiplayer environment for The Gaming Room’s game, Draw It or Lose It. The goal of this project is to make an environment for a list of teams, and a list of players on each team, to facilitate a connection between the clients. Our goal is to create a first prototype for the infrastructure. In addition, this ‘server’ has to be designed for a web based client.

## [Design Constraints](#_heading=h.tyjcwt)

The app must be designed to work within a web based environment, rather than as an independent application.

The app must be designed to support more than one team in a single game.

The GameService must be able to include multiple games running at once.

Game names must be unique within the GameService, and team names must be unique within the Game.

Each game, team, and player has a unique ID number to ensure only one instance of each exists at a time.

## [System Architecture View](#_heading=h.3dy6vkm)

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](#_heading=h.1t3h5sf)

The Game, Team, and Player classes all inherit from the Entity class, demonstrating Inheritance and Polymorphism. This also means each class inherits the name and ID properties, ensuring the uniqueness of all generated objects. The classes all use Polymorphism because they all alter the toString() function as well.

In addition, the model shows Abstraction, by the GameService being responsible for handling Games, the Game being responsible for handling Teams, and the Team being responsible for handling the Players.

The GameService has to 0 to many relationship with Game, Game has to 0 to many relationship with Team, and Team has to 0 to many relationship with Player. This is because each has a list of contained instances of the lower class, and can add to that list.

The ProgramDriver contains main(), so the program starts from there, and SingletonTester exists only so that ProgramDriver can test whether Singletons function correctly.

**"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](#_heading=h.2s8eyo1)

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** | Advantages  More compatible with Mac clients  Administration is more convenient than on Linux systems  Server would be user friendly for administrators, especially those not used to Linux  Weaknesses  May have issues scaling to a larger environment  If the system has more Win/Linux clients, it would be better to host with Linux  Win/Linux has more package compatibility than Mac | Advantages  Has advanced features that work well for experienced developers  Less resource intensive than other OS for hosting  Rarely faces cybersecurity threats compared to other major OS  Weaknesses  Too technically intensive for less experienced users  Many versions lack long term support  Lack of compatibility with some third party programs | Advantages  Most users, especially those who work outside of technical jobs, already have Win experience  Server would be more compatible with Win clients  Would have compatibility with most server software  Has Microsoft support available  Weaknesses  OS has an associated licensing fee  Prone to malware/security issues  Designed with less technically deep aspects than Linux | Mobile Devices are not suitable for hosting a server. Mobile devices do not exist which have the necessary processing power for hosting the server, in addition to all three major OS having existing server software and methods that can be used. |
| **Client Side** | Cheaper to develop for a Mac server  Less developers are experienced in developing for Mac, increasing time and costs  Development on Mac can be somewhat applied to development for mobile devices | Cheaper to develop for a Linux server  Easier for experienced developers to work with, less time intensive to work with  Linux provides a better development environment in many ways, and is the most secure popular OS | Cheaper to develop for a Windows server  Platform most developers are familiar with, numerous tools and resources exist to develop for it  Windows software needs more intensive security because of its inherent vulnerability | Most developers previously hired are unlikely to be experienced with Mobile development  Mobile software is likely to have less processing power than a user’s home computer |
| **Development Tools** | Visual Studio Code is one of the most popular IDEs for Mac  Ideal IDE for this project is Eclipse, which has Java compatibility, a core part of this project  MacOS tools can be used to develop apps for iOS mobile devices | Linux has access to numerous IDEs, such as Eclipse  Java can be used to host servers within Linux | Microsoft provides the proprietary IDEs, Visual Studio and Visual Studio Code for use with most languages, and a variety of plugins are available to enhance the capability of this software  Windows also has access to Eclipse, a popular IDE compatible with most languages  Easily compatible with Java and C#, some of the most popular server languages | Swift is one of the most popular languages for developing mobile apps  In addition, MacOS systems are popular as tools used for developing mobile apps  Java is still a popular language for web based apps and software applications |

## 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**: Windows is the most reasonable platform for an inexperienced company to handle. The platform is user friendly, has good software options, and is compatible with other platforms with some ease, making it the best for a company that does not have an experienced third party operating their servers.
2. **Operating Systems Architectures**: The architecture for this project should be focused largely on placing as little responsibility on the client as possible. Not only will this increase the user base by reducing the system requirements for users, but it improves security by ensuring that the minimum amount of data can be sent or altered by users, reducing the likelihood that they can send impactful altered data.
3. **Storage Management**: For storage management, a Storage-Area Network would be the best option on the server side. A SAN allows the server to do storage operations without taxing the server network, therefore reducing latency in the server overall, and for the clients. On the user side, the best way to manage storage is to keep as many operations as possible on the server side. This way, the client only needs to hold a minimal amount of assets and operations, increasing accessibility.
4. **Memory Management**: The best system for memory management, both for the server and the client, is dynamic memory. By only loading only the necessary processes and assets, both the server and client reduce their load at any time using the program. In order to do this, it’s also best that the necessary assets all be stored locally, so as to not burden either side with needing as much communication to function.
5. **Distributed Systems and Networks**: The project in a minor way, takes the form of a distributed system, with the necessity for user input driving the program. While technically the server exists independent of the users, the users do not exist independent of the system. In addition, the system has to function on a Wide-Area Network, as a Local-Area Network does not encompass the necessary online functionality for this program/
6. **Security**: The priority with security should be ensuring the security of user accounts, and ensuring that any input given is not harmful to the system. For the purpose of account security, encryption should handle the work of ensuring that the user’s password is sufficiently protected, along with common password requirements, such as requiring a certain length, and the presence of certain types of characters. Input validation should be incredibly stringent, to ensure that data being injected to the server is not possible. By preventing this vulnerability, we ensure that our software cannot be used as a vector for further security risks.