

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 06/20/2022 | Corey Pennebaker | Final Additions |

**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 new client The Gaming Room has hired CTS to create a web-based game called Draw it or Lose it. The game will consist of multiple teams with multiple players. Each team will have a turn to guess the puzzle while being shown stock images from a library as clues for 30 seconds. If the current team cannot solve the puzzle by the 30 second make, the other teams will have 15 seconds to guess or solve the puzzle. Each game will consist of 4 rounds each round lasting one minute a piece. The team with most points at the end of round 4 wins the game.

## [Design Constraints](#_2et92p0)

The game should be developed using JAVA and object-oriented programming. Classes for teams and players should be developed which inherited from a super class. The client also requires for the new game be available across multiple platforms as it is only available on android. So, applications will also have to be made for macOS, Windows, iOS and Linux to encourage new players into their new game. Each team will have the ability to have multiple players assigned to it.

## [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)

The UML Diagram here shows the layout of the proposed program to be developed. It has and Entity class with two private variables Id, and Name. The Game, Team and Player classes all inherit their properties from their parent Entity class. Through the use of inheritance, we are able to have multiple classes all using the same data in our Parent Entity class for all sub classes Game, Team and Player to eliminate unneeded data in our code. The GameService class acts as our singleton class to ensure that only one of our instances can be running at one time. And the ProgramDriver class uses the SingletonTester class to test and make sure only one instance can be running during the 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)

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** | Flexible and easy to use terminal features to connect with the server  Does not work well in large scale projects | Linux is a very popular on the Server-Side of applications because it is open-sourced  Linux is also typically cheaper to maintain oppose to MacOS or Windows | On the server-side windows provides an excellent graphical interface and is easy for users to learn the platform | Mobile Devices are not a good option when considering the server side of applications. Although they can handle run the applications themselves the device lack processing power. |
| **Client Side** | Mac has a higher price tag than most but with that comes an easy-to-use graphical interface and is easily updated in the background | Even though Linux is open source it isn’t as popular with users because it does not offer a great GUI like MacOS and Windows and is much more difficult for new user to learn | There is probably some type of windows operating system sitting in every household of the US. It gives users an easy to use graphically interface, but this operating system has been known for its security flaws | Mobile devices give users a great way to access their favorite apps anywhere. They offer an easy use to lightweight interface for use on the go. |
| **Development Tools** | Mac runs with Objective-C programming and can use other common languages like java and python  Specific IDEs like Xcode are available for developing mac applications but you will need a current version of MacOS to use | Like Windows, Linux can support multiple popular IDE for Developing making it a good option if working with application development | Windows can run a wide variety of different IDE for developing applications and make it a good option for developing across multiple Operating System Requirements | Developing applications for mobile devices can be difficult because of the wide range of screen sizes between phones and tablets. And multiple programming languages are needed to develop across both android and apples Iphone/Ipad |

## 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**: When deciding what operating platform to use it’s important to keep in mind for which part of the design architecture we are planning. When it comes to our user/client interface the most likely candidate is Windows OS, but with a simple GUI switch reports to a main server makes expanding to new user interfaces much easier. And for the server-side OS Linux would be best since it is open-source and free and is really an industry standard for server operating systems.
2. **Operating Systems Architectures**: To keep The Gaming Room application easily updatable and have a good architecture structure to expand on and grow to new users/new interfaces. Everything should be built around a simple three tier architecture. By hosting the main functions of the gaming room app on a centralized server. We will be able to give the flexibility to expand onto new user interfaces in the future, by developing a simple and easy to use client-side application. With the server pushing a modularized application that the user interface only needs to connect with to receive instruction.
3. **Storage Management**: Cloud based storage has become the number one solution to anyone’s long term storage needs. With companies like Amazon AWS and Microsoft Azure making it very easy and simple for new projects to host their information needs with ease.
4. **Memory Management**: The ways in which the separate operating system in our tier is important to keep in mind. We do not want to create unwanted data or multiple variables, this will create bugs and poor performance while users are running The Gaming Room app. And to help expand the application it will be beneficial to keep all process on the server side of the application.
5. **Distributed Systems and Networks**: By choosing to build The Gaming Room app around a three-tier architecture we can have the entire network of users working through one system. Instead of a big bulky application the user only needs to install a simple user interface to communicate back to the main server with. This also allows us to keep all our code in one central location making it much easier to maintain and lets us deliver updates to the app much easier and less complicated for the user.
6. **Security**: Addressing security on the user side best option is two factor authentication. At the very least require 2FA when signing into a new device, and give users the option to enable it each time they sign in. On the application side security become a bit easier because of the architecture we have built around. The actual application wouldn’t have to be download and install the client OS. Instead, we have the user connect to a server that is hosting our application to use, this allows us to keep all sensitivity information safe from any unwanted prying eyes.