

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 | 05/29/2024 | Kyle Gilbert | Initial submission of design template |
| 1.1 | 6/8/2024 | Kyle Gilbert | Revised submission |
| 1.2 | 6/23/2024 | Kyle Gilbert | Added detail to the Recommendations section. |

## [Executive Summary](#_sbfa50wo7nsh)

The Game Room would like to create a game where players compete in trying to guess a picture being drawn of something. We are to facilitate a web based version of this game.

## Requirements

* A created game will allow one or more teams to be active in play
* Each team may have more than one player
* Team names must not be equal to one another
* Only one instance of each game can be allowed in each case

## [Design Constraints](#_2et92p0)

* The first design constraint is having the ability to ensure that only one instance of each game is created. This could be done using the singleton method.
* Next we need to ensure that more than one player can join each game. We need to treat each game as a new class to hold the data.
* No 2 teams can share a same name, so a check of the strings that make up the names need to be coded.

## [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)

This UML diagram shows the “Game Service” module is responsible for creating games. Each game is responsible for creating teams to be used in the games and the teams need to be comprised of players. This UML diagram also contains the ability of the program driver to use the SingletonTester class.

**"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** | Mac is known for its reliability and Security, but can have a high cost. | Cost will be low but more work is required on the development side of things. | Works in the windows environment and will need lots of maintenance to match the windows updates. | Building for mobile devices requires change in interface and needs to work for android and Apple |
| **Client Side** | Typically more expensive, but safer. | Getting programs to run with linux can be more of a chore, most likely will need to emulate windows to make this work | Clients should have an easy time on Windows as things are primarily plug and play | Development should take more time as we need to develop interface that works with a touchscreen. |
| **Development Tools** | Tools tend to be user friendly and can generate some good looking ui’s but can be costly | Developing in Linux takes more work but generally low cost. | A lot of free support is out there for development on the windows side. | Development tools exsist at low cost, but need to be different to allow touch screen functionality. |

## 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 best operating platform I can think of for this project is Java. Java will work with most any OS and finding developers that work with this language should be easy. The Java Virtual Machine ecosystem. More information found here:

https://docs.oracle.com/en/java/javase/22/vm/java-virtual-machine-technology-overview.html

1. **Operating Systems Architectures**: To utilize Java across multiple platforms, it should be abstracted and run as an executable to be used in other OS. It could be further helped by the use of service like Docker. This allows the use of containers that will allow consistent deployment across many different OS environments.
2. **Storage Management**: For the storage of our information this game, I would recommend using Amazon could service. It would require payment, but in the long run, should be worth it. Amazon is also capable of providing database solutions in the form of RDS (Relational Database Service). This provides a lot of services that are done automatically as your clusters are monitored and they are equipped with self -healing storage and automated scaling.
3. **Memory Management**: Memory management should be built in to the program we are utilizing. Java is capable of ensuring that the data is in ram and no leaks are taking place. We could also use a program like Visual VM. This program can display local and remote java processes. This can be used during development and testing of the game. It allows you to profile performance and memory usage, visualize threads, monitor the memory management of the game and display thread dumps. This tools can help the streamline of the testing and ensuring that we are releasing a stable product.
4. **Distributed Systems and Networks**: If the gaming room were to host its own servers we would be capable of managing the ability to span multiple platforms. We may need to do this to ensure they would be capable of that. This would add up-front cost and added responsibility for ensuring safety. We would also need load balancing and auto scaling because we’re not sure how many players will be on a game at any specific time. AWS is capable of elastic load balancing
5. **Security**: Security can be insure by the use of HTTPS for communication. This protocol is encrypted so users data is kept private. This can prevent user data from being stolen and make phishing attempts less likely for individuals. Dependencies will need be regulated and updated to ensure the safety of the data we are keeping and managing. We can also implement logging of any anomalies to ensure that we can investigate these separately to expose areas that may need more work.