

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 |
| --- | --- | --- | --- |
| 1.0 | 11/11/2022 | Ryan Bermel | Revised information relating to the software design. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants a web-based version developed of their popular Android game Draw It or Lose It. The game is to have the ability to have multiple players on multiple teams, allow unique team names, and allow only one game to exist in the memory at a time.

## [Design Constraints](#_2et92p0)

* The game must be able to run on multiple types of platforms
* Have the ability to have multiple teams and multiple players per team
* Checks to ensure unique team and player names
* Allows only one game to exist in the memory at one time

## [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 Game, Team, and the Player classes inherit from the Entity superclass having the attributes and the operations contained to only be written once and have each class inheriting from it. Each of the classes GameService, Game, Team, and Player are all associated with one another. They use zero to any number of associations meaning the can have and use as many instances as needed. The ProgramDriver class delivers the package and will inherit the SingleTester class while doing so.

**"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** | Can be costly. Has flexible easy to use terminal commands to configure the server. Easy to access and make fixes or change. | Cost effective, difficult system to navigate. Has a command shell for easy access and server configuration. | Can be costly. Easy to use, has the ability to run on multiple systems. | Not too expensive. Specifications differ depending on the device. |
| **Client Side** | The cost is similar to fellow platforms, but requires a little more knowledge than competitors. | Even with the cost being more economic than Mac or Windows, the time and experience necessary can be too difficult for some. | User friendly, easy to use and understand. The knowledge needed to use the software is minimum. | Does not take long to master. Has supportive system. Not as expensive. Easy to maintain and update. |
| **Development Tools** | Works with most common programming languages such as: Python, HTML, and C++. Supports Eclipse, visual Studio, etc. | Uses ordinary languages: Ruby on Rails, Java, Python, etc. Supports Eclipse, Visual Studio, GitHub. | Easiest to use than competitors. Uses ordinary languages: Ruby, Rails, Java, Python, etc. Supports Eclipse, Visual Studio, GitHub. | Uses frequent languages of Python, Java, HTML, Ruby on Rails, C++. Supports GitHub, Visual Studio and command prompts. |

## Recommendations

1. **Operating Platform**: Even though can be difficult to grasp, the platform I would recommend would be Linux. Linux is open source and supports various tools used to development. It is also a secure and flexible platform to use.
2. **Operating Systems Architectures**: The architectures are kernel and shell. The Kernel connects applications and data processing done at the hardware level. The Shell is a software piece that shows and interface for the users to the operation system.
3. **Storage Management**: An appropriate storage would be a cloud system. The more the game becomes popular, the storage will need to expand to cope and with Google Cloud the cost to do so will be kept at a minimum.
4. **Memory Management**: For the development of this game Java will be used. Java utilizes memory management automatically through a system called Garbage Collector. Therefore, implementing a memory management system is not required.
5. **Distributed Systems and Networks**: If we utilize the browser Google Chrome that is available on all platforms for this web-based game, it should be easy to access to all users regardless of user OS.
6. **Security**: Google provides the hardware and infrastructure with Google Cloud so data center security is no issue. By using Google apps and having the game automatically download patches, the software will continue to be secure.