

The Gaming Room

# **CS 230 Project Software Design**

Version 1.2

## 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 | 3/13/2023 | Steven Klotz | Changes made to cover page and all following sections, except Evaluation and Recommendation. |
| 1.1 | 3/27/2023 | Steven Klotz | Added details to the Evaluation section. |
| 1.2 | 4/11/2023 | Steven Klotz | Added details to the Recommendation section. |

## [Executive Summary](#_sbfa50wo7nsh)

This project is to develop a multi-platform web-based game. It’s based on *Draw It or Lose It*, but that game is only available for Android devices, so the company’s goal is to bring it to other devices. The Gaming Room’s purpose is to have several teams competing for 4 rounds with each round having 1 minute. They must guess what images are being drawn before time runs out.

## [Design Constraints](#_2et92p0)

* Game must work on multiple platforms, or **cross-platform**, as it is currently only available on Android devices.
* Identifiers must be unique and only one game can be happening at any time. This would require **unique names** to be given to each game, teams, and players.
* The **user interface** can either be made similar to the existing Android game, or an upgrade from it.
* Game must be able to support more than one team with each team containing **multiple players**, which would require a more server friendly approach to the code of the game.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram below gives us a visual design of the game system. The lines between each class show they’re associated. The numbers between each class shows us the number of associations in each class. The Entity class connects the Game, Team, and Player classes. We can see the programDriver will use the SingletonTester to test the code. This tester lets us test having one game existing in memory at a time. GameService will hold all the code that make the game’s basic functionality. The UML class diagram will help in developing the game and assist in approaching the final product.

**"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** | MacOS has simple and easy to follow accessibility and with configuring servers. GUI is easy to operate along with the terminal. | Difficult to use. Simple server configuration and accessibility due to the command shell. | Expensive for servers. GUI is easy to operate along with the command prompt. | Not the best specifications compared to other devices, which also vary quite a lot. |
| **Client Side** | Very expensive. Some experience required and time to put into it. Mild skills for using the OS itself. | Lots of expertise and time needed. This data is required to use the OS. Close to being the most expensive. | Very, very expensive. Easy to learn and comprehend for the user, therefore minimum expertise required. | Flexible for updates anytime, anyplace. A little trickier to implement than other devices. |
| **Development Tools** | Libraries supporting frontend development. Dev. tools also include PyCharm, GitHub, Visual Studios, etc.  Languages consisting of HTML, CSS, and JavaScript. | Libraries supporting frontend and languages. Languages consisting of HTML, CSS, and JavaScript. Linux contains JavaScript, Ruby, PHP and Python. | Libraries supporting frontend and languages.  Languages consisting of HTML, CSS, and JavaScript. Dev. tools also include Eclipse, PyCharm, etc. | Libraries supporting frontend and languages. IDE’s include HTML, PHP, C++, and Python.  Languages consisting of HTML, CSS, and JavaScript. |

## 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 believe the most appropriate operating platform that will allow The Gaming Room to expand *Draw It or Lose It* to other computing environments would be Windows. This is due to it having the largest amount of compatible IDEs to use.
2. **Operating Systems Architectures**: Microsoft’s Windows operating platform is a graphical operating system that is, and always has been, developed and published by Microsoft. It can be used to watch entertainment, connect to the internet, use its own browser or a 3rd party browser, play video games, and store and manage files.
3. **Storage Management**: Microsoft’s Windows, and especially their current version, Windows 11, contains its own full-fledged storage management system that is advanced by their *Storage Sense* system. This system is a silent assistant that works together with OneDrive to automatically free up space. It does this by making files that are normally locally available become online-only again if it detects you’re no longer accessing those files, or more specifically if you haven’t accessed them for a certain period or more.
4. **Memory Management**: Microsoft’s Windows uses memory management techniques for the *Draw It or Lose It* software by, again, utilizing its *Storage Sense* system to allow for intricate storage and management of the game’s photos and players, while also keeping them together in a secure memory space.
5. **Distributed Systems and Networks**: Network games are multiuser interaction systems that usually contain a database that is shared with other players being physically distributed and interacting with each other via the network. Game developers do have to implement this shared database and inter player communication system from scratch, but this could change in the near future.
6. **Security**: Microsoft’s Windows does contain its own security protection software “out of the box”, meaning just having Windows installed on your device will already implement this security system. Although, it’s still good to use another party’s security system due to all of the user data and information that comes with a game like *Draw It or Lose It*, or any game or application.