

The Gaming Room

# **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 | <01/28/2024> | <Jonathon Adkins> | <Updated content for Executive summary, design constraints, and requirements> |
| 2.0 | <02/11/2024> | <Jonathon Adkins> | <Updated cells within development table> |

**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)

A new client, “The Gaming Room” needs a web-based game that also serves multiple platforms based on their current game, “Draw it or Lose it.” This is currently available only in a Android app only.

## Requirements

* *The game needs to run on multiple platforms.*
* *The game will check for unique team and the game names must be present.*
* *Should have multiple teams of multiple people.*
* *Only one instance of the game should be allowed at any given time.*

## [Design Constraints](#_2et92p0)

The game being developed will have multiple design constraints which include producing the game to run on multiple platforms, currently this game code is only written to work on Android platforms. The current code must be able to run across multiple platforms, this process may take new developers to be hired to work on updates and working on the code across other platforms.

## [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 Entity class becomes a Super class that allows team and player classes to all inherit from Entity class. The GameService class references to the Game class and it has reference to the Team class, which references to the Player class. Having all these classes referencing to one another is called aggregation. There is also the ProgramDriver where executions happen and uses the SingletonTester which also has a use relationship with the ProgramDriver class. The classes are set up to only allow for one game at a time with multiple teams and players each being pulled from a list.

**"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** | **Advantages: -**Good UX and GUI  Easily configurable server setup  ***Weaknesses****: Offers fewest software options for deployment*  *Most expensive platform for hosting* | **Advantages:** -Open source Most secure option -Most cost effective  Most control over server configuration Cheapest hosting available  **Weaknesses:** -Least platform level support -most obtuse UX | ***Advantages:*** *-Open Source Large amount of platform support and adoption Most software options*  ***Weaknesses:*** *-Proprietary Software as a service model for platform Somewhat inflated hosting cost* | -Existing application in this platform  -Different OS specifications  -Moderate to Low cost of hosting |
| **Client Side** | Most expensive commercial user platform ceiling -Moderate Development expertise required -Least flexible browser configuration. | -Requires most experienced Development team  Cheap for users  Requires the highest level of expertise for the user.  -Most flexibility in the browser environment | Most user-friendly platform, relatively low barrier for entry  Good UX and GUI  Relatively low expertise required of development team, common browsers with the mobile platform. | - Intuitive GUI and UX  Least development expertise required.  platform specific browsers must be considered in the implementation. |
| **Development Tools** | Recommended IDE: Eclipse, Xcode implementations  Highest cost of licensing  Languages: Swift, Javascript, Python, CSS | Recommended IDE:  Eclipse, Visual Studio, Repl.it  Some free alternatives to licensed IDE  Languages: C, Java, Python, Ruby, HTML | Recommended IDE: Visual Studio, Eclipse, Repl.it  Some free alternatives to licensed IDE  Languages: C++, Java, Python, HTML | Recommended IDE: Android Studio, Xcode, etc, (platform specific)  Moderate cost of licensing IDE  Languages: Swift, Objective C, Java |

## 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 would choose the Windows platform for the Game Room, mainly because the overall lower cost and market penetration is already provided. Which can see future collaborations on Windows and could easily be ported for mobile device platforms. There are also many IDEs, emulators, and databases that are on windows platforms that can be utilized.
2. **Operating Systems Architectures**: Windows can provide services used by all Windows based applications that enable them to show Graphical User Interface which is GPU. These can be accessed by using a user account or a server. There are two modes kernel mode and user mode, kernel can be looked at like the master mode or privileged mode. User mode processes graphical user interface and affects systems that the user will interact with. The user wouldn’t have access to kernel mode and would be more in the background of the platform.
3. **Storage Management**: While using windows for storage management I would recommend using cloud-based storage to allow for easy scale up and down as the user base continues to grow. If we used disk or hard drive storage it would hinder the growth potential and would cost much more to scale up or down. With cloud based it comes with 24/7 support and continuous updates as technology evolves at a rapid pace. This is just a few of the advantages of cloud-based storage.
4. **Memory Management**: The Gaming room will need to create a database and/or library with many pictures. The memory allocation allows us to easily store all this picture data. Windows has built in memory management by using disk paging to act as an extension of the physical memory. Disk paging reserves a partition of the hard drive disk to store temporary files. They are separated into smaller files and uploaded only as needed during the game.
5. **Distributed Systems and Networks**: I recommend using a client-server distributing system to provide gaming services to each player. Each client application would have its own connection to the game Draw It or Lose It and would rely on that connection. If connection was lost the game would be saved in temporary memory for restart. Offer a Windows application with web-based connection to be accessed on Mac-OS and Linux operating systems.
6. **Security**: Windows has built in virus protections already installed within the environment, The database and its safety features would extend across all platforms. Any data could be uploaded into the cloud which has built in security and customer support. This would take one less worry out of the equation compared to hard drive/RAM based storage security.