

The Gaming Room

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 07/26/2024 | Leigh Ann Stack | Changes were made to the document revision history, executive design constraints, system architecture view and domain model. |
| 2.0 | 7/30/2024 | Leigh Ann Stack | Changes made to executive summary |
| 3.0 | 08/13/2024 | Leigh Ann Stack | Added recommendations. |

**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 Gaming Room project is to develop a web based game that works on multiple platforms based on the current game Draw it, or lose it. This game is only available on android right now. This game is based on the classic TV game show. With this transition to a web-based platform, the game will be available to more people, across devices.

## Requirements

* Web based Accessibility
* Game round and time limits
* Post-time guesses
* Single instance of game and unique identifiers

## [Design Constraints](#_2et92p0)

* Cross-Browser Compatibility
* Only one instance of the game can be running at any time
* Game and team names are unique, the game will allow users to check to see if name is available
* Each team can have multiple players
* Data security
* Resource Management

## [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 diagram shows the unique class structure within the application system. Attributes like id and name help create unique identifiers throughout the game. The super class of Entity creates a relationship between game, team and player class. This means they all get information from the Entity superclass. GameService has a composition relationship of games, which hold and manage game instances. The program class driver is the entry point of the game. This is where the main function lives. Program driver is responsible for adding teams and players to the GameService instance.

This UML based diagram shows key object oriented programming principles. These principles include inheritance ,encapsulation and abstraction.

**"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** | -Easy accessibility  -Easy server config  -Easy to use GUI  -Flexible terminal  Commands  -Stable platform  -Limited server software | -Budget friendly  -Platform is difficult to navigate  -Command shell for simple server config.  -Open source OS | -Expensive server  -GUI is user friendly  -Command prompt | -Mobile specs vary from user to user.  -Portability |
| **Client Side** | Expensive  Moderate expertise  Skills needed to navigate OS | Linux data required for OS. | Pricier than Linux  Easy to learn  Minimum expertise required. | Responsive design  Uses native features such as GPS and camera |
| **Development Tools** | Languages: HTML, CSS and JavaScript. | Languages: HTML, CSS and JavaScript. | Languages: HTML, CSS and JavaScript. | Languages: 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**: The Gaming Room should use a cloud based architecture. When you don’t have a server to maintain you can make sure your application runs smoothly at all times. Windows would be my recommendation for an OS. They offer the most user friendly GUIs on the market. They have a large library of powerful suite programs that will work across platforms.
2. **Operating Systems Architectures**: Windows is a OS developed and published by Microsoft. Windows uses a hybrid architecture that’s divided into two main components, user mode and kernel mode. There is no direct access to hardware which helps protect the system from crashing. The kernel mode is where the core of the OS runs. This includes kernel, device drivers and devices. It has direct access to system data and hardware.
3. **Storage Management**: Windows has many built in utilities to help with storage management. There are tools like disk cleanup that are designed to help maintain the systems storage by removing unused files. Storage Sense from Microsoft allows for calculation of how much storage is being used and will alert when almost full.
4. **Memory Management**: Windows has paged virtual memory. There are physical, hidden files on the hard disk which frees up RAM for other active programs.
5. **Distributed Systems and Networks**: Need a database to share data among players so they can interact over the network. A client server distributed system will work best with this game. This allows the game to have multiple clients that work on its specific system.
6. **Security**: A role based security system would be best. This allows for separation of admin, game, team, player and user roles. This will help users not be able to access information they shouldn’t have access to.