

<The Gaming Room>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

**[CS 230 Project Software Design Template](#_Toc115077317)** [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 | <11/12/23> | <Jacob Burchett> | <Changes made to cover page, history, executive summary, requirements, design constraints, domain model, evaluations, and recommendations> |
| 1.1 | <11/26/23> | <Jacob Burchett> | <Updated evaluations> |
| 1.2 | <12/10/23> | <Jacob Burchett> | <Updated 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 a web based game based around the existing "Draw It or Lose It" game. The game should have multiple teams with multiple players each. The game will consist of 4 rounds at 1 minute each where an image is pulled from a large stock of images and will gradually render while allowing guesses until the image renders in it's entirety or the image is guessed.

## Requirements

## The Gaming Room has asked that the new game be web-based. Because of this, we will need to make the game multi platform. In addition, the game must support multiple teams and players, each with unique names. Only one instance of the game can run at any given time.

## [Design Constraints](#_2et92p0)

* Multiple platforms
* One of more teams
* Multiple players per team
* Unique team/game names
* Only one instance of the game can exist at a 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)

Below is the UML diagram for the project. Looking at the diagram, we can see that Entity has become a super class, connecting to Game, Team, and Player via inheritance. We can also see aggregation present within the UML diagram as Team references Player, Game references Team, and GameService references Game.

"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** | Easily configurable. Flexible terminal commands. | Cost effective but difficult. Command shell allows for easier server configuration. | Expensive on server side. Command prompt allows for speedier configuration. | Varies device to device. Potential for lower specs than the other platforms may cause limitations. |
| **Client Side** | High cost. Low time and expertise requirements. Mac is user friendly and easy to learn to navigate. Mac development is straightforward. | Low cost. High time and expertise requirements. Linux can be difficult to navigate for new users. Open-source design may be hard to manage. | High cost. Easy to work with, low expertise needed. Windows development is easy to navigate and quick to learn. | Moderate cost. Easily updated but requires a learning curve for developers not used to mobile accessibility. |
| **Development Tools** | IDE for development (Pycharm, Eclipse, Visual Studio, Komodo). All major languages acceptable. Eclipse IDE for Java programming sufficient. Pycharm for Python programming recommended for ease of future development. HTML libraries required. | IDE for development (Pycharm, Ecliple, Visual Studio, Komodo). While Linux will allow for development in most major languages, C++, Java, and Python are recommended. Eclipse IDE for Java programming sufficient. Pycharm for Python programming recommended for ease of future development. | IDE for development (Pycharm, Eclipse, Visual Studio, Komodo). All major languages acceptable. Eclipse IDE for Java programming sufficient. HTML libraries required. Pycharm for Python programming recommended for ease of future development. | IDE for development (Pycharm, Eclipse, Visual Studio, Komodo). All major languages acceptable. Eclipse IDE for Java programming sufficient. HTML libraries required. Pycharm for Python programming recommended for ease of future development. |

## 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 current recommendation is Windows. While Mac and Windows both would allow for ease in development and maintenance, the lower cost of entry for users with Windows generally makes it a more appealing platform. Mac development would require the acquisition of a developer key with an annual fee that must be paid for the continued development of Draw It or Lose It.
2. **Operating Systems Architectures**: Windows is a layered OS consisting of two major parts, the user mode and the kernel mode. Available in both 32 and 64-bit variants, Windows is available to users with a wide range of system operating capacity. The Gaming Room should focus on cloud based Windows architecture so that the acquisition of servers is not required. While there is a monthly fee associated with cloud based software, ultimately the cost would be less than that of the acquisition of servers and the employment of individuals for server maintenance and upkeep.
3. **Storage Management**: Windows allows the user to easily manage storage. A PC utilizing windows can easily accept new storage options and the management of software on the allotted storage can be maintained easily by navigating to the storage panel within windows by simply typing "storage" into the search bar. Cloud based storage would simplify the process as information would be stored offsite and accessible from anywhere with an internet connection. Due to the online nature of the game, cloud based storage would require no additional hardware for users and may increase appeal due to low personal storage requirements. The use of cloud storage would also allow for developers to easily make changes to the game for future updates without the need to push said updates to users for additional downloads.
4. **Memory Management**: Like storage, Windows systems can easily upgrade RAM. The more common bios are easy to understand and follow along with. Most come with a "wizard" that will assist with anything necessary should a user opt to upgrade their own PC. The "storage sense" of Windows 10 and 11 will also allow for improved management of memory. The utilization of virtual memory would also be of benefit. Though the expected memory usage of Draw It or Lose It is relatively low, utilizing virtual memory may make the game more accessible for users with lower performing computers and increase speeds for all users.
5. **Distributed Systems and Networks**: In regards to systems designed for multiple users across a network, databases are distributed to the player via their internet connection which will communicate with one another, allowing for play online. Internet connection to a cloud based game would reduce downtime before startup as there would be little to no initial download required by the users before interaction. This could lead to a more appealing game to the general public and increased user base.
6. **Security**: Windows has built in security. This will serve as a base level of security. Windows Defender is much more reliable now than in past years. Keeping Defender updated is the best way to ensure security currently. Additional security software may be installed to serve as a layer system against malware. With cloud based architecture, various providers of cloud services have additional layers of security via firewalls to help ensure the security of the system. The implementation of these various security options will lead to a much safer experience for users and The Gaming Room.