

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 | 05/24/2024 | Cory Graham | Executive Summary, Design Constraints, and Domain Model. |
| 1.0 | 06/06/2024 | Cory Graham | Evaluation section updated. |
| 1.0 | 06/18/2024 | Cory Graham | Recommendation section updated. |

**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)

Draw It or Lose It is a popular game, but is currently only limited to android devices. We know we can garner more activity by producing a version of the game by putting it on a web-based platform, giving more devices and in turn more people access to the game. Our team can re-build your game and make it available online. This online based game will have all the features that make the mobile app successful and popular.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Software: The application currently runs exclusively on android devices. While some of the code is potentially usable, we are going to need to build it ourselves. Its not as simple as copy and paste.

Time: Because we are writing our own software for the game we will need to set aside time for a meeting to make sure we can agree on the working project. Our team will be working in 4 week sprints, during our meeting we will determine how many sprints are required. Please note that at the end of each sprint you will receive working updates to the code.

System: There is not one single operating system. We will need to code the game so that it will work on Windows, Linux, and MacOS.

## [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 a UML or a Unified Modeling Language. This acts as our roadmap to building your game. By following the directions of the arrows, we can see what information is being pulled from where. The Entity is the main body, and it receives information about who is playing from Game, Team, and Player. You will notice that Game Service, Game, Team, and Player are connected to each other using a line with a symbol “0…\* ”. This means that these blocks also receive information from each other. The symbol simply means “one to many”. For example, the Game Service connects to Game with the “0…\*” you can read it as “one Game Service to many Games” and the next block “one Game to many Teams’.

Within the block you will also notice each block has two smaller blocks below the title. This gives information to the developers as to what variable to expect and what functions should be contained in each block function.

**"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 must 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** | MacOS offers server services. These servers work well with apple computers making them easier for apple computer owners to get full supports for what they are using. The cons of MacOS are the price to maintain servers as well as their narrow ability to offer customization. Most companies end up using other programs and customizations. | LinuxOS is a popular choice for many companies, most notable is Google. Linux offers free and open-source material great for customizations and maintenance. Due to the customizability of how to organize Linux, it has a high rate of security. Linux also offers different coding compatibility with Python and others. Cons are the steep learning curve and the lack of availability of certain programs and cross compatibility with Windows. | Windows is the most used operating system available. Windows does require users to pay for licensing which can be expensive. Windows comes with high support for third-party and has a high rate of support from its developers. | Mobile devices can act as a server base. Quality and server capacity depends on the specific operating system such as iOS or Android. |
| **Client Side** | Cost: Highest cost out of the selection.  Time: Moderate time as most people will have at least some experience with MacOS. MacOS is also claimed to be easier for first time users.  Expertise: Mid-level. Although base use is easier to grasp, any advanced use requires skill and practice. | Cost: Lowest cost, being free.  Time: High, more time will be allocated to correctly learning the system. Although the initial time would be high, after usage the time would decrease to a moderate level.  Expertise: Mid-level with a high learning curve. | Cost: Moderate, most computers already have the system.  Time: Minimal, due to familiarity the time needed will be the lowest.  Expertise: Mid-level, despite the familiarity aspect, there is still a need to practice and master the usage of the software. | Cost: Mid- high, due to niche platform for the apps.  Time: Long, as it would take longer to develop apps for each OS.  Expertise: High, due to the need to create multiple versions in different platforms. |
| **Development Tools** | Languages:  CSS  JavaScript  Java  HTML  IDEs:  Visual Studio  Atom  WebStorm\*  IntelliJ\* | Languages:  CSS  JavaScript  Java  HTML  IDEs:  Visual Studio  Atom  WebStorm\*  IntelliJ\* | Languages:  CSS  JavaScript  Java  HTML  IDEs:  Visual Studio  Atom  WebStorm\*  IntelliJ\* | Languages:  JavaScript  Swift  Objective-C  IDEs:  Other Tools:  Xcode  VSCode  Atom  WebStorm\* |

\*Denotes paid licensing

## 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**: For the operating platform I recommend using WindowsOS. The reason being that it’s used by most computers reducing the need to educate developers on its server. Also having a wide range of third-party tools makes the price to use the system worth it.
2. **Operating Systems Architectures**: Architectures include 64-bit memory allowing for more RAM and providing improved performance over their 32-bit system. The server also includes optimized features for server environments.
3. **Storage Management**: For the storage of the web-based game a cloud server would serve best. Service providers like Google would allow for upgraded storage as the game increases in popularity without having to pre allocate the storage in advance. This would also eliminate the need for personal servers managed by the Gaming Room.
4. **Memory Management**: Windows servers are designed to handle the needs of the memory through Memory Allocation, Memory Protection, NUMA Support (Non-Uniform Memory Access), and others. These techniques ensure reliability with high performance.
5. **Distributed Systems and Networks**: Getting the game to communicate between various platforms requires planning for the game to work smoothly on different systems. By using Google for the server we already have access to different platforms as its widely available.
6. **Security**: By utilizing the storage provided by Google the security is taken care of for the cloud servers. Our responsibility is to make sure that the code for the game is running smoothly and all errors taken care of with patches or updates.