

# **Draw It or Lose It**

# **CS 230 Project Software Design Template**

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 | 01/28/2024 | Tucker Mayo | Everything from the cover page to the recommendation section as seen above has been modified. |
| 1.1 | 02/11/2024 | Tucker Mayo | Added recommendations to recommendations section. Evaluated each operating system with pros and cons in the Evaluation section. |
| 1.2 | 02/29/2024 | Tucker Mayo | Added more detail to recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

The gaming room company is looking to design a new, web-based game. They want this game to run on multiple platforms. The goal is to make this game like their current Android only game, Draw It or Lose It.

## Requirements

* Must be able to have 1 team or more.
* Multiply players per team.
* Game names and team names need to be unique.
* 1 instance running at a time only.
* Web-based
* Four 1-minute rounds
* 15 seconds for opponent to steal question if not answered.
* Pictures to be used as clues.
* Gradual fill on clue images up to 30 seconds.

## [Design Constraints](#_2et92p0)

* Must be able to have 1 team or more.
* Multiply players per team.
* Game names and team names need to be unique.
* 1 instance running at a time only.
* Web-based
* Four 1-minute rounds
* 15 seconds for opponent to steal question if not answered.
* Pictures to be used as clues.
* Gradual fill on clue images up to 30 seconds.
* As of now, no end date or time constraints.

## [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)

In the UML design below, the Entity class is the parent class to the Game, Team, and Player Class. They will inherit the information from the Entity class. All the classes below are connected and refer to each other. The only class in this design that does not connect to any other class is the Singletong Tester.

**"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** | The servers are easy to configure. Apple offers server-based environments as well. They have strong security measures as well. A big advantage is how easy to use the interface is. Unfortunately, there is a smaller pool of macOS users. An important weakness for Mac’s is the cost. Both getting the hardware and maintaining afterwards. It can also be difficult to completely customize some programs. | It has a lot of advantages for web hosting, most importantly though it is free and open-sourced. These servers would require the least hardware on our side. Another big advantage is the security, it's very customizable to what we’ll need. However, a weakness for this OS is it’s the least popular option, so it comes with a learning curve if you’re unfamiliar. The servers also don’t have a wide range of options for their software. There can also be migration issues from another operating system to Linux. | The most popular software and GUI. It is proprietary software, which means we need to pay for licensing. Windows licensing can get expensive, but it comes with good advantages. It supports many applications and third-party software. It’s easy to update hardware and software when needed. It also has many support resources for their servers. However, this is the least secure due to its popularity. | It is easily portable and only accessible by one user. It would be impractical, I believe, to run a web-based server from a phone. It is possible though. Oracle offers a service to implement a mobile server. It can manage apps, users, and devices. An advantage of an Oracle database is it has iOS and Android developer support tools. It’s optimized for data collection. The security on a mobile device is difficult to work with, as well with working with specific requirements with the OS. |
| **Client Side** | The hardware is the most expensive among the 3 OS’s. It supports all browsers and associated tools.  A hybrid app could be created over a native app, so it is portable. | It's the most diverse, but requires the most expertise. The react-native web can be implemented support iOS, Android and web platforms with a single codebase. | The easiest OS to use. It can be expensive; however, it is generally cheaper than a macOS. | The most flexible and used. They have lower hardware specs; this can make development and user usage difficult. |
| **Development Tools** | Languages used include HTML, CSS, JavaScript. Can also use PyCharm, GitHub, and visual studios. VSCode for syntax highlighting and code previews, Homebrew package manager to install Unix and Mac utilities, Xcode IDE, iTerm2 emulator, Tower git client, Dash api browser. | Languages used include HTML, CSS, JavaScript. Can also use PyCharm, GitHub. Visual Studio Code, Atom, Vim, bash command line, node, flask. | Languages used include HTML, CSS, JavaScript, C++. Can also use PyCharm, GitHub, visual studios, Eclipse, and Command prompt. Gvim, node, npm, yarn. | Languages used include HTML, CSS, JavaScript. Can use C++, python |

## 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 Gaming Room, I believe we should use a server operating with Windows. I think Windows is the safest operating system we can use, not only for security reasons, but for how easily accessible it is to users. Using the most popular OS will give us more opportunity to gain users, as well as be able to keep our software up to date.
2. **Operating Systems Architectures**: Windows architecture allows for the utilization of the platform’s kernel processes without affecting them. So, with this, our app can run better by using the internal hardware from the users OS. This will help our game run smoothly without affecting the user’s experience.
3. **Storage Management**: Using a Windows OS, we have Disk storage, with built-in Disk management and Storage Sense. Disk management will help us with vital information storage. Storage sense helps maintain the storage for us nu deleting unnecessary files, so this will help us save space, so we don’t constantly have to go in and manually delete things often.
4. **Memory Management**: Windows excels at memory management and saves files efficiently. A database would need to be created for the image files required for the game.
5. **Distributed Systems and Networks**: For multi-interaction systems that are network based, a client-server system will be utilized. Each user app is going to run depending on that user’s system. Our game will be running on a single server. Ensuring the server newtowrk strength is key, as we can only be successful with this game when multiple users are playing the game together through one server.
6. **Security**: Windows Defender is built into Windows, and for a built in security system it works well. Something we will need to do is to make sure to encrypt data being sent between our server and the user.