

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/16/22 | Alexis Indick | Updated the title. |
| 1.0 | 05/17/22 | Alexis Indick | Added summary, design constraints, and domain model explanation of Draw It or Lose It. |
| 1.0 | 05/18/22 | Alexis Indick | Fixed up the design constraints. |
| 1.0 | 05/30/22 | Alexis Indick | Added the chart for the evaluation. |
| 1.0 | 06/12/22 | Alexis Indick | Updated the Recommendations part. |
| 1.0 | 06/13/22 | Alexis Indick | Finished the Recommendations part. |

**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 wants a web-based game that serves multiple platforms. Draw It or Lose It is the game that want developed for other platforms since it is only available for Android devices now. Draw It or Lose it will render images from a large library of stock drawings as clues. One game consists of 4 rounds one minute each. Drawings will be rendered at a steady rate to meet the 30 second mark. If a team doesn’t solve the puzzle right in time, the other teams are able to offer one guess each to solve the puzzle with a 15 second time limit.

## [Design Constraints](#_2et92p0)

* User-friendly interface
* Multiple platforms can play the game.
* More than 1 team must be implemented.
* 1 game can only be run at a time.
* Must have unique team names.
* 1 game is 4 rounds for 1 minute each.
* Images must be rendered at each 30 second mark.
* Other teams can offer one guess each with a 15 second limit.

## [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 Game, Team, and Player class inherit from the Entity class. Each class shares a reference of the name and id variables since it comes from the Entity class. The GameService class has an association with game from zero to many. The Game class has an association with team and so does Player and Team also.

**"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 terminal is quite flexible with its commands, and it makes getting access to the server easier along with making changes to the server.  The downside is though that Macs are quite expensive to run a server. It is about $20 a month to run (which is inexpensive), but you would also want a very good Mac for it to run on so the price skyrockets.  They are also less popular to host servers on. | The terminal is also very flexible with its commands, but Linux machines are cheaper than Macs to run a server.  The downside is that Linux is a bit hard to navigate if you are new to it. | For the server side of Windows, there is more software available compared to Mac and Linux OS.  But the downside is that Windows are more susceptible to viruses and easier to hack. It is also more expensive than Linux to run a server since it is about $20 a month for a standard edition but the Data Center edition is $165 a month, which is what we’d want for running the server. | Mobile devices are quite inexpensive to run a server for. An app hosting service can range anywhere from $20 to $70 a month but this depends on the service.  The portability of mobile devices though is quite good.  If the server stays immobile, it is easier to keep track of it. It is also cheaper to run a server on mobile devices compared to computers. |
| **Client Side** | Cost is expensive and time would depend on the developer and how familiar they are with Macs. It can be quite difficult. | The cost is cheaper but is time consuming to master if you are not familiar with Linux. | The cost is like Macs. Time is also dependent on the developer and their familiarity. But is easier to figure out. | Very flexible and inexpensive, but difficult for developers to implement. Since there are many different mobile devices, it can be very time consuming to develop for each. |
| **Development Tools** | Swift is the language used on Macs. There are many IDEs that can be used with Swift like Atom, Xcode, Visual Studio Code, and AppCode. | Java is the most used language for Linux. IDEs like Eclipse, Visual Studio, and Atom can be used to develop. | Many languages can be used on Windows like Java, C#, C++, Python, etc. IDEs that can be used for development are Visual Studio and Visual Studio Code, Eclipse, IntelliJ, etc. | For mobile devices, IOS devices are in Swift, and Android is written in Java, but supports other languages. You can use many IDEs for either but for Android the most famous is Android Studio and Xcode for Apple IOS. |

## 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 think the platform to be used should be Windows. This is because it is the cheapest operating system and widely used more than Linux and Mac. There’s many IDE’s and software that works on Windows more than the other 2 platforms.
2. **Operating Systems Architectures**: I think Windows again will work here more. Windows has a simple GUI that is just more user-friendly and easier to use. The command prompt is powerful and easy to configure server stuff on. Many languages work for programming on Windows vs. the other platforms.
3. **Storage Management**: Windows wins here again for storage since it allows for us to add more storage whenever we want, allows for cloud storage like AWS, and has storage management options to optimize storage usage. Storage Sense for example, is a feature Windows can use to help the user get a feel for what are unnecessary files and programs to be deleted to save space.
4. **Memory Management**: Windows uses both physical and virtual memory. Memory management is also a utility in Windows. Virtual memory in Windows is about 4GB in space which is a plus. The more memory we have, the faster the application will run and smoother.
5. **Distributed Systems and Networks**: Since we need to accommodate for other platforms besides Windows, we can use a cross-platform tool like React Native which supports many programming languages and platforms or Unity that is more used for creating games. We also need to make sure we use a good server like Windows Server. Though it is expensive, Windows Server is a very good and reliable service, especially when it can be used with Microsoft Azure which is a cloud platform service so everything will be in the cloud and storage is saved.
6. **Security**: Windows has its own security like Windows Defender and the firewall. We can utilize this, but we can also find extra anti-virus software for cheap like AVG to keep use protected better. Encryption of the data is also important when it goes between the server and client back and forth.