

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc120275126)

[Table of Contents 2](#_Toc120275127)

[Document Revision History 2](#_Toc120275128)

[Executive Summary 3](#_Toc120275129)

[Design Constraints 3](#_Toc120275130)

[System Architecture View 3](#_Toc120275131)

[Domain Model 3](#_Toc120275132)

[Evaluation 4](#_Toc120275133)

[Recommendations 7](#_Toc120275134)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/11/2022 | Max MR | Initial Software information Update |
| 1.1 |  | Max MR | Added Operating Platform Information |
| 1.2 | 12/11/2022 | Max MR | Adding Recommended virtual specs |

**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 Client “The Gaming Room” has outsourced our company’s services to further build on a mobile gaming app that is currently only available on the Android App store. They are requesting us to make a web-based game that is based on this current game called “Draw It or Lose It”. This game is teams competing on guessing the main selected item, person, or place. It uses a cycle of photos to give a hint of this selected goal item. With a time-limit, and 4-round based game system. The reason for this is to bring in more users, thus raising revenue

## [Design Constraints](#_2et92p0)

This game is going to be web-based on cross-platform. There will have to be API documentation to help communicate across different platform languages.

Individual ID’s that are Unique is a must have. The Gaming Room wants the game to have a Unique gamer handle system. This creates a system where multiple players can’t have the same name. Since this game is going to be on multiple web-based platforms, this will require a database management system to manage all the ID’s and programming to check if the ID already exists in said database.

This game must also have Multiple teams and players per game. We have been told that multiple teams of multiple players are a requirement that must be achieved. Since this game is communicating across different platforms, there will need to be a server implementation to support the communication of multiple players/teams in a single game

Finally, this game MUST be based on the original, we will have to work with The Gaming Room on getting original files, to be able to create this game the form of a mirror reflection of the original app. The performance and appearance must be as similar as possible.

## [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)

For this project, a UML model has been provided. If you refer to the UML diagram, you can see that the

Entity class creates a relationship with the Game Class, Player Class and Team Class. These classes are derived from the entity class and will inherit attributes from this class. The Game Service is the instance and holds most the accessors in this case, making it the foundation of the game. The ProgramDriver is using the SingletonTester to test the instance. Scrum methods will be used for this project, and therefore this UML Diagram is subject to change throughout the course of this project.

**"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** | MacOS servers having great user-friendly interfaces, with an updated and award-winning cyber security, but can be costly to the  Customer. | Because Linux is mostly open-source tools are free or very cheap or can be self-created. This also makes it very easy for the customer-client to customize their security system. Linux is only as good as we make it to be, with that said the downside is the maintenance that would have to be applied to create such an cost and operations efficient environment. | Windows has many benefits. It can support application implementation and great security features. Most languages are able to effectively communicate on Windows OS. SQL databases are naturally efficient with Windows OS.  The downside is that Windows requires licensing to be able to use its servers. Theses licensing costs can be very expensive. | There are many servers to choose from, but it all depends on the server-side provider that we choose. Most have great structure administration when it comes to management on android and iOS devices. These come at cheaper prices then the latter. But can cost efficiency such as communications between languages. |
| **Client Side** | Mac users must be able to afford a client device, making this option the most expensive. With simplicity and user-friendly applications this requires little experience to be able to operate the OS. | Although very cost friendly, this OS isn’t widely recognized by users. This causes difficulty operating the OS and management of the system’s capabilities. | Windows is also an expensive option, isn’t very cost friendly. Windows allows for wide range of platforms such as Edge, Chrome, and Firefox. This OS contains a command prompt and is widely recognized so users don’t have trouble managing OS. | Mobile devices vary. The client side can vary from the parent company such as Apple, Android, and/or Google OS. These are all different. Android allows users more control of the device configurations, but is a bit more difficult to operate. Apple is very user friendly, but allows for less configurations to be changed. |
| **Development Tools** | The Main language for MacOS development tools are mainly Swift. Swift is an easy development tool to use and learn. It’s open source, has timely processing times and is very secure. But the downside, is that it is only limited to iOS systems. | Linux is very versatile. Because it is open source, is compatible with a lot of IDE’s, API’s and languages. Linux is very exceptional in this category. | Windows has many IDE’s that are usable when approaching development tools. Such as VSCode, Eclipse, Atom and much more. It is also compatible with Java, Python, and C classes. Which are amongst the most useful languages in development. | Mobile app development tools such as React framework, make great use of JavaScript for developing apps. It is both used with iOS and android apps. We can also use Flutter that is a development kit by Google that is can be used with Android, iOS and Fuchsia. |

## 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**: CTS recommends Linux. Linux has great cost efficiency. It can be creating to be a very secure platform with plenty of flexibility amongst languages. Linux also supports Ubuntu and Python scripting; this is preferred if the game will have cloud capabilities moving forward.
2. **Operating Systems Architectures**: Operating systems consists of top layer which is applications. Applications communicate with the Kernel. The kernel communicates with the Hardware level which is CPU, Memory and other I/O devices, and instruments.
3. **Storage Management**: Draw It or Lose It is cross-platform. Because it is cross platform, CTS recommends that you use a Cloud based platform as we said earlier. This will attach and communicate well with Linux. Using Linux, you can create the Cloud servers to communicate with the application and only use the memory that you need. This will let the game perform much better and handle more traffic and allows upscale without having to assess capital expenditure on hardware upgrades. There are plenty of Cloud options for gaming, such as Azure, AWS, Google Cloud Platform, and Oracle (for mobile applications). If you choose to use Cloud, we can provide more information on a cloud platform provider.
4. **Memory Management**: Java will be the main language for the backend of this memory management. Java is easily understood, is object oriented very easy to implement across multiple platforms and uses implicit memory management. This will leave out the explicit memory management code that would need to be provided for the game.
5. **Distributed Systems and Networks**: Since the game is going to be web-based, the best web browser to use would be google chrome. This browser is a secure, user-friendly browser that is available across most OS. This would many demographics, and different countries to connect to the game whether they would have a Windows OS or iOS.
6. **Security**: While in development, we will have to imply QA testing. This will ensure there are no logical bugs that can be exposed in our system. As for memory management, Security will be implied already from many different providers. If Cloud, the security will already be provided there. If not Cloud, there will have to be an IT team deployed to mange security. There will have to be an updated version that can be set for automated patches and updates.