

# 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 | 11/13/2022 | Zachary Derepentigny | Updated template to detail the Draw It or lose It design project. |

**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 Game Room team lacks the expertise needed to set up the environment needed for the web-based version of Draw It or Lose It. The CTS Team will streamline the development of this game be preparing this design documents and begin development of the application. Development will need to follow the software requirements provided by The Game Room team. Following the software requirements will allow for the development of the ideal application to house the web-based version of Draw It or Lose it.

## [Design Constraints](#_2et92p0)

The Game Room team has provided a list of software requirements that must be followed during development of the application. The application will need to have the ability to involve one or more teams within a game. Each team will need to have the ability for multiple players to be assigned to them. The application will need to be able to check for unique team and game names, preventing duplication of existing names. Only one instance of the game will be able to exist in memory at a given time by creating unique identifiers for each instance of game, team, and player.

## [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 individual classes Game, Team, and Player all have a inheritance relationship with the Entity superclass, in that they all inherit elements from it. This can also be described as a “Is A” relationship, in that each class is an entity. The remaining classes have a “Has A” relationship between them. In other words, the GameService class has Games, which in turn has Teams, which further has Players. This relationship can be described as Aggregation in UML terms. Describing the UML diagram this way further breaks down how each class relates to the others and the shared relationships between them, even if indirect.

**"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** | Mac features flexible command terminals allowing a greater degree of configuration. For web-hosting Mac provides flexibility, but is overall less commonly used than Linux or Windows. Mac licensing costs were more difficult to research than Linux and Windows, likely as a result of it decreased popularity. | Linux offers similar flexibility to Mac, however, is less commonly used. Being open source, Linux often provides the best security as issues that arise are often resolved at great speed making it the safest option for webhosting. Due to the niche nature of Linux, it has the least flexibility and options, but also seems to have the lowest costs when compared. | Windows offers a wider range of software options vs. the other two OS options. Windows sees the most use of the options and thus often has the most support and comfortability of them making it a common choice for webhosting. Windows licensing costs varied greatly compared to the other options but seemed to be the highest on average. | Generally weaker than the other OS options. Mobile is at a disadvantage due to servers often favoring being in a single location. Mobile also provides the worst performance of these. Likely due to the increased mobility, mobile webhosting boasted astronomical license pricing compared to the stationary OS options. |
| **Client Side** | More user experience is required to become proficient with this OS when compared to Windows. Mac will often require moderate costs, but increased time and expertise to dedicate use on the client-side | Requires the most user experience to use, with this offsetting cost. Linux boasts some of the lowest cost in most aspects of its development and use, however, demands the most expertise, and likely by extension, time to develop and use properly. | The most approachable and easy to use OS when compared to Linux and Mac, while still having similar costs to Mac. Windows provides the greatest accessibility for the client side limiting time needs, however, has moderately higher costs than Linux. | Mobile provides greater flexibility and mobility to the client and is generally easier to use due to limited features. Mobile platforms benefit from their flexibility in the location of use, however, often require greater costs and expertise with the platform to benefit from this. |
| **Development Tools** | Java is the most popular programming language for Mac, commonly utilized through Eclipse IDE. Eclipse is a common IDE that has few technical requirements for a team. Eclipse also benefits from being entirely open source, limiting costs. | Linux also commonly utilizes Java, but can also extend to C++, these can be utilized through Eclipse or Microsoft Visual Studio. Visual Studio, like Eclipse is also commonly used for development, however, does feature higher costs for licensing. | Python and C++ are commonly used on Windows using IDEs such as PyCharm for Python and Visual Studio for C++. Python, like Eclipse and Visual Studio is commonly used and accessible but shares Eclipse’s advantage of being entirely open-source and free. | Java is commonly used for development on mobile devices, often using the Eclipse IDE. Mobile IDE’s have a variety of accessibility for development teams, potentially necessitating multiple teams with varying costs by IDE. Existing, open-source IDEs are, however, still available for mobile use. |

## 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**: Windows provides a similar server-side experience as Mac, while being significantly more usable than Linux or Mac. The utility of the application may make Windows more or less convenient than Mobile OS. Because of its reliability and history of usability, Windows is the best recommendation for an operating platform.
2. **Operating Systems Architectures**: Windows can utilize both 32 and 64-bit architectures. 32-bit is generally better for older systems, however when building a new application, 64-bit provides better performance and features, making it the better option.
3. **Storage Management**: Solid State Drives are typically the highest quality storage management products, offering the best access to storage, as well as performance. These will generally always be the best option for performance without losing on cost.
4. **Memory Management**: 64-Bit Windows can allocate up to 16 TB of memory between the kernel and individual user programs. 8 TB of memory is designated to each, contained in virtual addresses.
5. **Distributed Systems and Networks**: Distributed systems and networks generally utilize a database that is shared among users that shares the relevant information over the given network. Databases must be designed from the ground up, but once created, allow for the needed information to be seamlessly communicated across the systems involved.
6. **Security**: An application such as this game is relatively isolated and not a large security risk, however security should always remain a priority in applications. Limiting the collection of user data can increase the overall safety of the application, by keeping identifying information separate from the application outside of user entry. Windows provides a customizable security experience allowing as little or as much as desired, when compared to OS’s like Mac or Linux.