

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 11/11/2022 | Sequoia Sutherland | Initial summary draft. |
| 1.1 | 11/27/2022 | Sequoia Sutherland | Evaluation completed. |
| 1.2 | 12/08/2022 | Sequoia Sutherland | Recommendations completed. |

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lose It will be a web-based game application which is derived from a television game from the 1980s. The Gaming Room has an existing android mobile game which will be used as the model for the web application. Creating a web-based version of the game will enable The Gaming Room to reach a larger audience, as android users make up only a portion of all those who are interested in playing games in a computer or mobile environment. The application will take the model of the existing mobile game and recreate it as a web-based application so that users who are familiar with the mobile application can easily play online without learning new rules or features.

## [Design Constraints](#_2et92p0)

Keeping in mind the importance of maintaining continuity between the existing mobile application and the web-based application which is to be built, there will be little creative freedom in this project. CTS will endeavor to replicate the game which mobile users are familiar with. The web application must be compatible with all popular browsers so as to reach the largest potential audience 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)

The UML diagram for The Gaming Room simplifies the code of this game as several object oriented programming principles are used. Primarily, the classes Game, Team, and Player are all derived from the Entity class, which eliminates the need to rewrite code which is used in the same way across the child classes. The Game, Team, and Player classes are related in this model because one instance of a game will require there to be instances of teams and players as well. The GameService class is not derived from the Entity class but it relates to the three child classes because it has accessors and mutators which interact with the Game, Team, and Player classes. Polymorphism is used to overload the getGame() method to allow the user to search for a game either by the game id or by name.

**"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.**

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | macOS Server may not be the best choice for The Gaming Room to host this application because of it’s limited functionality compared to other operating systems. macOS is an affordable server option that aside from requiring Mac hardware has no other hardware requirements. There is not much scalability for the macOS and thus it is better suited for personal use or that of a small business with no plans to expand. Because The Gaming Room may want to further expand this game’s reach in the future, using a macOS Server may place limitations on the app’s growth. | Linux is a strong choice as the OS for The Gaming Room’s servers. Because Linux uses open-source software, there is more flexibility in how it’s hosting environment can be configured. Additionally, the security of Linux OS is generally considered to be one of the best. This is because Linux is designed with multiple levels of privileges, so it is very difficult to compromise the entire system. As a web- based application, Draw It or Lose It will benefit from additional security measures. The Linux OS does require the development team to be skilled in utilizing this OS both to obtain the most benefit as well as to avoid misconfigurations which could result in a breach. Because The Gaming Room will not be processing sensitive PII, a shared hosting environment may be appropriate, which would reduce costs and the Linux OS server could host the application for a low monthly cost. | Windows is possibly the most popular OS, and The Gaming Room would find it easy to manage because of its compatibility with third parties and ease of obtaining support. The cost of hosting from a Windows server is slightly higher than Linux, as it is not open-source. If The Game Room would like to rely on third parties to develop or manage parts of the application, a Windows server may be the best choice because of its compatibility. Windows is less efficient than Linux for high volume handling, so if The Game Room anticipates high usership of the application Windows may prove to be a more faulty choice. | Mobile devices cannot act as servers, so applications which are run on mobile OS must be hosted either through a traditional hosting environment or as a native app directly through the app store of each mobile OS. If The Game Room would like to host the application as a native app, it need only build an app in the Apple app store as the Android app already exists. The developers could replicate the existing application to fit iOS standards and launch it in the Apple app store easily and with a low cost. This approach could be the most cost reductive method but would require three separate environments for the mobile apps and web app. |
| **Client Side** | The Game Room has an advantage in managing client side requirements because Draw It or Lose It is a web-based application and therefore will not require interaction with the client’s OS beyond their web browser. To ensure full compatibility with Mac OS, Safari compatibility should be built into the application’s architecture. Setting HTML Doctypes is one step in ensuring this compatibility. | Because Linux OS offers users the ability to run so many web browsers, The Gaming Room may have to choose to limit it’s compatibility to the most popular browsers in order to reduce costs. Because of different browser requirements, choosing just the most common browsers such as Google Chrome in addition to Mozilla Firefox which is pre-installed on Linux machines will enable The Gaming Room to reach a large number of users while keeping costs minimal. | As with Mac OS and Linux OS, catering to Windows OS client users will largely depend on the developers building an application which has a cloud-based platform and has been tested across browsers. This application testing will assist The Gaming Room in ensuring that as many users as possible can play the game. Windows OS has Microsoft Edge pre-installed, so this should be one of the browsers which is tested for compatibility. | The addition of mobile applications will require the developers of Draw It or Lose It to create separate mobile compatible applications to function on the mobile OS of Apple and Android. Because the Android application exists, the iOS application is the only mobile application which needs to be developed. This will produce an additional cost burden, but because a large number of game player’s can be anticipated to play on iOS devices, The Game Room should see this as a worthy investment. |
| **Development Tools** | When considering what programming language or languages to use, The Gaming Room’s developers will need to consider what languages offer compatibility to all of the browsers which have been identified in the discovery phase. JavaScript is the most common web application language because of its portability and functionality. Additionally, many developers are fluent in JavaScript and thus it would be likely that the existing developers on The Gaming Room’s team will know how to build an application using this language. Python would be another powerful choice for the application because of it’s built in functions. | The application will be made to have compatibility with a set of major browsers, so specific development tools should be chosen that meet the requirements of all identified browsers. In addition to the languages previously mentioned, The Gaming Room’s developers will need to utilize an IDE to create and manage the code. Visual Studio and Visual Studio Code may be a good choice because they are free and offer the ability to work with most programming languages. | In addition to considerations in the previous two columns, The Gaming Room may consider setting up Github repositories so that the code can be internally tracked and shared. | When developing the iOS application, if The Gaming Room utilized a native app development software for the Android application, then they will need to invest in software for a native iOS application. Alternatively, the development team could choose to migrate the Android application’s code to a mobile development software such as Kobiton that supports app development for both iOS and Android. This may simplify changes to the application in the future. |

## 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**: Taking into consideration the current size of The Gaming Room as well as future expansion needs, it would be most appropriate for Draw It or Lose It to utilize Windows OS servers. While Linux and Windows both offer strong capabilities to scale the game in the future, Windows’ ubiquity makes it a better choice both in the short and long term. Because The Gaming Room may not have the budget to hire a large team of developers, operating a Windows server allows more flexibility in integrating third party tools and software into the application. Additionally, most developers are familiar with Windows OS, which means that The Gaming Room is less likely to have to provide additional training to their developers, or pay a costly specialized contractor. Windows OS servers can be managed at a reasonable price, which should be within the budget of The Gaming Room especially if cost saving measures are implemented at various levels of the project.
2. **Operating Systems Architectures**: Draw It or Lose It should be implemented with a layered pattern architecture. This is the most reasonable architecture because it is a relatively simple game application which will be hosted and accessed as a web-based application. The layered pattern architecture is one of the most popular application architectures because of its ease of creation and intuitive design. This architectural style would be advantageous to the application because each piece of the application could be separated from one another while working together to form the end application. This is helpful when working with One of the only and major drawbacks of this architecture is its scalability, but because Draw It or Lose It would see future growth in number of users rather than exponential increases in game functionality, this limitation would not be of great concern.
3. **Storage Management**: The amount of storage for game photo files need not be unlimited because there is a set number of photos which will be used for all game play. The creation of photo files within the game will not occur, so the game does not need to have this functionality. The files will merely be accessed in a read only format. In order to increase the speed of retrieval of the game files, current-file position pointers should be implemented as this saves steps in retrieving the file. The application should be designed to maintain an accurate file-open count, as one of the main functions of this application will involve opening and closing files. Storage of game files should be handled by the server’s disk. This will ensure that the photo files needed during gameplay can be easily accessed. The file-organization module will be an important piece of Draw It or Lose It because of the game’s reliance on retrieving, opening, and closing files. This layer of the architecture will be responsible for managing the file’s locations as well as free space.

**Memory Management**: It is recommended that gameplay data is stored in the server’s memory, as it is short term data that aside from the final score only needs to be retained during the current game. User data held beyond gameplay should be stored in a separate database. Depending on how the leadership team decides to weigh the expense of maintaining user data against the value that users would get from having access to this archive of gameplay data, a decision will need to be made of how much memory space to purchase and maintain for Draw It or Lose It. The more memory that is maintained, the higher the expense, but having the ability to view old scores may be enough of a draw to users that it is a worthwhile cost for The Gaming Room.

1. **Distributed Systems and Networks**: As a smaller company with limited resources, The Gaming Room should consider utilizing a cloud web service platform to host Draw It or Lose It. For example, utilizing a service such as AWS would enable The Gaming Room to keep their operating costs minimal, and scale the infrastructure services they rely on up or down depending on how much traffic the game is attracting. Utilizing IaaS also would enable The Gaming Room to have redundancy in their program network architecture, which would greatly decrease the likelihood of an incident affecting the game’s availability.
2. **Security**: As a web-based application Draw It or Lose It needs to be developed with security in mind at each step. Utilizing secure coding best practices needs to be part of the design process, and should be included in the SDLC to ensure secure coding is maintained for future updates. Additionally, the security measures utilized to protect the network hosting the game will play an important role in protecting user data. At a minimum, The Gaming Room should utilize encryption via https / TLS to the most current version when exchanging communications with users over the web. If any user PI is collected, this should also be encrypted in storage, and a key management system could be used to further protect this data. The Gaming Room should also implement network safeguards such as firewalls, load balancers, and intrusion detection systems. AV/AM software should be utilized on any device supporting the application, to further protect the network environment.