

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 08/13/23 | Nicole Swanson | Updated Recommendations |

**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 to develop a web-based game that can run on multiple platforms. The game they want to work with is Draw It or Lose It which is currently only on Android. The game is loosely based on the television game Win, Lose or Draw from the 1980’s. It will have four rounds each lasting one minute. It is also a team game and can include one to four teams. If one team doesn’t guess after the timer ends the other teams can have 15 seconds to guess the drawing. Another important aspect of this game is that it will have a large library of stock drawings to use that will take 30 seconds to be rendered at a steady rate.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* Must be able to run on multiple platforms.
* Web-based capabilities are required.
* Needs space for stock images and drawing program.
* Must have the ability to include multiple teams.
* Teams will have multiple players assigned.
* Unique game and team names as well as ability to check if a name is already in use prior to choosing is required.
* The memory can only hold one instance of the game at any given time so unique identifiers for each instance of game, team or player must be created.

## [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 class diagram represents the relationship between the classes in the program. The Entity class is a super class and holds the game, player, and team instances. They have an “is a” relationship with the Entity class. These 3 classes also share common attributes including id and name. The Team and Player class has a “has a” relationship with each other. This is because Team class references the Player class. This is true of the Game class also as it references the Team class. The GameService class has instances and organizes the Game class.

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | * Has a web-based server to connect all Apple devices * Must use a mac to develop game for their devices * Servers only host mac users * Has good security * Has cloud, ADP, and LDAP support | * Has an open-source server for deployment * Is good for web-based developing for multiple platforms but not commonly used for game developing * Requires minimum recourses * Has good security * Has cloud, ADP, and LDAP support | * Has multiple server deployment options * Azure is available as a good option for a web-based game that needs scaling up capabilities * Has more potential security threats * Has cloud, ADP, and LDAP support | * Web-based servers are available for deployment * Limited processing power * Planning for mobile first can be beneficial * Security is moderate, risks but not as many as windows * Has cloud, ADP, and LDAP support |
| **Client Side** | * Can cost anywhere from $5 to $150 a month a game for a server * Can take getting used to for the system, but if moving from windows or Linux can be easy * May take a large amount of time * Can communicate to server via REST API | * Can cost between $250-$1300 a year for a server * More difficult to get used to system * Fast overall system * Can communicate to server via REST API | * Can cost an average of $119 a month * User friendly, easy to learn system * Similar time as mac for developing * Can communicate to server via REST API | * Comparably priced to an Apple server * Easy to use system * Can be more time consuming * Can communicate to server via REST API |
| **Development Tools** | * Swift is a popular for this system * Can use notepad ++ * Licensing costs are about the same for all * May need multiple teams | * Most common languages for system C and C++ * Can use visual studio and notepad * Licensing costs are about the same for all * May need multiple teams | * C++ and Java are most commonly used * Can use Visual studio * Licensing costs are about the same for all * May need multiple teams | * Java is most used for mobile app developing * Cordova can be used to develop on android * Licensing costs are about the same for all * May need multiple teams |

## Recommendations

1. **Operating Platform**: Linux is an excellent choice for the Gaming Room’s ‘Draw It or Lose It’. It is an open-source operating system. This system is constantly maintained and improved. It is customizable to the needs of the user. The base operating system is also free. Also, as a server it is a common choice for its features. A big selling point with regards to this game is it can work with the target platforms for gameplay, Mac, Windows, Linux itself, and mobile devices. Also, Linux is a leader in cloud computing and the cloud is a necessary feature for a game that has the scaling needs of ‘Draw It or Lose It’.
2. **Operating Systems Architectures**: A multi-tier architecture would be beneficial for this. This architecture would separate business data on a data server, business rules on a middle tier server, and presentation logic for client. Having three tiers makes it easier to modify or replace one tier without affecting the other two, having them separated will improve load balance, and security policies can be implemented within the individual tiers without hindering clients.
3. **Storage Management**: Linux comes with a swapping system for file management. It is a little different from what is traditionally considered to be swapping for file management. With Linux it can swap out specific files not just whole processes. This would be good for the game as it can swap out image files so that only one is there at a time speeding up game loading times. This would also be good for keeping files that are only needed at start up out of the way to keep the speed up for the game. This works well with the swapping method in Linux.
4. **Memory Management**: Linux has multiple ways of managing memory that can be customized to best suit the game’s needs. They each work well with the cloud. Virtual memory is heavily used. It shares memory with competing processes as its needed. It works with the swapping system of storage management to also keep memory space free. There is also shared virtual memory so if multiple processes need the same part of code, they can use it together rather than take up more space with having repeat processes running at the same time.
5. **Distributed Systems and Networks**: With Linux a REST API would work well for communicating between the various platforms. It uses HTTP requests to get information from or put information on the server which makes it better for development and management. When REST API is implemented, it allows the application or in this case more specifically the game to use the same logic on all platforms. So, it will work the same on Mac, Linux, Windows or a mobile device.
6. **Security**: Out of the available operating systems Linux is considered to be the most secure. It has a smaller attack surface than other operating platforms and as a result is less susceptible viruses and malware. Being open source means that many people have access to the code so when a security flaw is found it is typically short lived. It is also configurable to the specific needs of the users. It would be easy to create separate security measures for the Gaming Room’s employees who need access, the future game players, and any other areas as necessary.