

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 | 09/19/2024 | Chris Bridges | Add players and Teams |
| 1.1 | 10/02/2024 | Chris Bridges | Evaluation |
| 1.2 | 10/14/2024 | Chris Bridges | 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)

Draw it or Lose it is transitioning to a web based platform that would allow people to access it more easily. The client wishes to have the web app available on IOS and Android.

## [Design Constraints](#_2et92p0)

One instance of the game can run at a time

Game must run on all platforms

Input validation

Multiple teams and players

The Game Room wants their app to be released on all platforms, so the code must be adjusted for each release. If the client wishes for it to be released as an app on the stores, that will require additional architecture to facilitate. If the client wishes to publish on a website, additional debugging would be needed for different browsers. These restrictions will require additional time, money, and resources to accomplish.

## [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 entity class is a superclass or parent class from which the game team and player classes inherit. Gameservice references game class while the game class references the team and player classes.**

**"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 isn’t commonly used for server sided operations. Scaling and price may be a factor. | Inexpensive, open source and stable. Requires knowledge of CLI. Can integrate with AWS, GCP, and other cloud easily. | .net framework makes it easy to host web-based apps. More expensive than Linux, but easier to manage. | Mobile hosting is not an option. Mobile devices do not have the hardware required. |
| **Client Side** | Cost would be the highest in terms of licensing fees. Development and deployment would be easy. | Would require builds for different environments. Low cost, but Linux isn’t a popular platform for consumer use. | Largest client base. Easy to develop for and costs would be moderate. | Costs would require license for both Ios and Android. Would require frameworks that would work between the two operating systems. |
| **Development Tools** | Development in Swift with Xcode would facilitate the development of this web based program. | Visual Studio code can be used to create this web based program and a variety of languages. | Visual Studio can be used to create the web app with .net framework and possibly C#, my preferred language. | React would be a good choice for programming language. |

## 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**: Linux is great for its stability, flexibility and cost. It’s a widely used server OS due to its scalability, and the open-source nature of the platform. Linux also supports a numerous amount of programming languages and frameworks which would allow *Draw it or Lose* to be used on multiple platforms.
2. **Operating Systems Architectures**: deploying *Draw it or Lose it* on a Linux server would allow the game to be accessed through a web browser. This would reduce the need for client platform specific applications and make it easier to connect with different devices. Using a browser-based application would allow the game to be accessed on multiple operating systems and allow for updates and maintenance to be simpler.
3. **Storage Management**: Using Azure Database PostgreSQL would allow for storage expansion and ease of use. This allows for reliability, scalability, and the addition of many features. This would integrate well with other Azure services and allow for storage expansion as needed. PostgreSQL’s compatibility with Linux and multiple programming languages makes it an excellent option for handling player and session data
4. **Memory Management**: Caching would reduce the need to access the database frequently. This would improve performance by storing frequently used information in memory. Paging would allow the system to handle larger amounts of data by storing them in blocks, allowing quicker access to the game’s components as needed. Virtual memory would help the server allocate memory dynamically, improving the system’s ability to manage multiple users at once. This flexibility is important for supporting a multiplayer game where the player base can fluctuate at different times of day.
5. **Distributed Systems and Networks**: Websocket allows for low-latency communication which would allow real time interaction between players in multiplayer mode. By using Azure’s scalability, the system can dynamically adjust to web traffic and minimize downtime. This would improve the user’s experience. Managing and optimizing traffic based on peak player times would be necessary to manage the costs.
6. **Security**: To protect the user’s information, SSL/TLS will be used to ensure all data transmi9tted between the client and server is encrypted. OAuth 2.0 provides secure and scalable authentication that will mitigate the risk of unauthorized access. Azure’s security features provide an extra layer of protection to the server infrastructure including DDoS protection, firewalls, and regular updates to security.