

Draw It or Lose It

# **CS 230 Project 3**

Version 1.3

## 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 4**](#_Toc115077324)

[**Evaluation 5**](#_Toc115077325)

**6Recommendations 6**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0  1.1  1.2  1.3 | 05/16/23  05/17/23  05/31/23  06/13/23 | Devin Perry  Devin Perry  Devin Perry  Devin Perry | Everything up to the UML diagram description was completed.  The rest of the document was completed.  Some edits and changes were made to the Server Side, Client Side, and Developmental Tools sections.  Final edits were made that complete the document and made it ready for final submission. |

## [Executive Summary](#_sbfa50wo7nsh)

Gaming Room wants to take their already existing application called Draw It or Lose It and port it to be a web-based game. Draw It or Lose It is currently only available as an Android app. The solution is to bring this Android app to the web to make it easier to access. This game has some pre-built structure that comes from Draw It or Lose It, which is seen in the UML diagram in the Domain Model section below.

## Requirements

The game needs to be compatible with most major web browsers so the majority of people can access it easily. It also needs to run well, which means fast speeds, minimal input lag, and an easy to understand interface. The game should have the ability to have one or more teams involved. Each team should be able to have multiple players assigned to it. The game and team names must be unique, and a function to check if each name is unique should be implemented. Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The game needs to be compatible with most major web browsers.

The game needs to run fast and efficiently.

The game needs to have an easy to understand interface for ease of play.

The game needs to allow one or more teams to be involved.

The game needs to allow multiple players on each team.

The game needs to make sure that each game and team name is unique.

The game needs to only allow one instance of the game to exist.

## [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 shown below shows the relationships between the classes of the given application. There are seven classes in total, with them being Entity, Player, Team, Game, GameService, SingletonTester, and ProgramDriver. The Entity class is used to hold many common variables, behaviors, and attributes that will be used in other classes. The Player class represents the users that are playing the game, that has a one-to-many relationship with the Team class that groups together the players. The Game class represents the game itself and has a one-to-many relationship with the Team Class. The GameService class holds numerous variables that help the Game class run. Finally, the ProgramDriver and the SingletonTester class work together to get the game running.

**"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** | Mac is great when the user is using a Mac device. It has great advantages for this case, but if another platform tried accessing it, it would prove difficult. Mac also runs on a small | Linux is Unix based which was originally designed to be reliable and stable. Linux servers are well known for being stable and not having any issues. | Windows offers Windows hosting which can be very useful for this kind of project. The other option for hosting is Linux, and Linux is currently more popular for hosting. However, Windows does support hypertext mark-up language files. | When coding a mobile app, it is difficult to find a host for the app. Also, when creating a mobile app, you create the whole thing by yourself. Creating a mobile app gives good customizability though. |
| **Client Side** | Mac hardware is more expensive than Windows. Also, users are forced to buy a Mac product to use Mac systems. Mac often has less technical problems because of their relatively easy to use interface. | Linux is more user friendly than some others. It is accessible from most computers and has no up-front costs. Most software for Linux is also open source. | Windows is great for developing websites and applications. There are tools such as Visual Studio that can prove useful. Some options for Windows that would be helpful is that it can run other platforms as emulators. | Mobile devices come in two main forms: tablets and phones. Mobile UI is tailored to these two devices with bigger buttons and visuals because the user is using their fingers to tap on things instead of clicking with a mouse. |
| **Development Tools** | Mac suggests using Swift which is a language that anyone can use to make their own apps that will work on Apple devices. | Linux supports numerous languages. These include C++, Java, Python, and tons more. | C++ is what mainly runs Microsoft. C++ is extremely popular, so it is good that they use it and not an outdated language. | Swift would be a useful tool to develop a mobile app. Java is used for Android development. While Mac uses their languages for their mobile devices. |

## Recommendations

1. **Operating Platform**: I think the best choice for an operating platform would be to use JavaScript. JavaScript would be great to use since it can be used across all kinds of different websites. It is also quite popular, easy to use, and has high execution speeds.
2. **Operating Systems Architectures**: When considering the architecture of the game, we need to consider that it is being used across numerous platforms. The code should be able to be reused multiple times. Client-server architecture would be preferred for this game since it is being played in real time with other players. Using this kind or architecture is preferred because it allows the user to submit a request to the server, the server replies, and the user gets want they want back. An example would be the user requesting to login and begin the game, and the game allows them to enter.
3. **Storage Management**: Since the game is web based, cloud storage would be necessary. The users shouldn’t have to download anything to their device. Cloud storage is easily accessible, which is useful. This game will be relatively small, so the game files won’t take up much space. What will take up most of the space is the database of user information like scores and login info.
4. **Memory Management**: Having good architecture when it comes to memory management will be key. It will help keep the game running smoothly and make sure there are no leaks either. These leaks would cause performance issues and slow down the game. Memory is a short-term place to hold files, so they are ready to be used quickly. JavaScript has a feature that checks whether something is required to remain in memory or if it can be let go. This is a way that it manages its own memory.
5. **Distributed Systems and Networks**: A strong server will be required to handle the players from different device types. The system and network should be able to distribute the workload across itself to prevent slowdowns or poor connection for the users. In general, any game will have a harder time when there are more players online versus when there are not as many, so getting a strong server will help this issue.
6. **Security**: Ensuring that the game is protected from malicious people is important. Having a good defense and anti-cheat system will keep the game friendly and keep out any cyber attackers. Using secure coding practices and encrypting the data will help. Security checks should be regularly conducted to make sure no one has gotten in and changed anything.