

Draw It or Lose it Web-Based

# **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 | 7/22/22 | Gary Jenks | Initial Design documentation |
| 1.1 | 8/1/22 | Gary Jenks | Added to and edited the evaluation portion |
| 1.2 | 8/14/22 | Gary Jenks | Final recommendations section |

**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)

Currently The Gaming Room want to translate their android based app Draw It or Lose it into a web based one. One that will work across a multitude of browsers. Being already made for Android it has existing Java code. The client might want to also look into other options but remaking the game in a different language could prove more costly than modifying the existing code.

## [Design Constraints](#_2et92p0)

The biggest design constraint would be keeping to the same framework set in place by the android app. To change up to much of the app would prove detrimental to the user experience given the familiarity of the existing android version.

## [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 parent entity class is a shared inheritance target for Game, Team and Player. While GameService, Game, Team, and Player all have an association that is zero to many from left to right on the diagram and as listed in order here The class are used as iterators shown in blue on the uml. ProgramDriver has a directed association with SingletonTester.

**"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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Might want to consider other options before committing to mac Servers options seem limited in 2022 and official final release of macOS sever was at the end of 2021 it seems further updates will no longer support it. | Linux is often free depending on the package and is also highly secure like windows.  Because of the open source nature of the OS the amount of options to choose from can be high while cost remains relatively low. | Windows servers Known for security and File management systems.  With one of the most familiar GUI interfaces it is also quite user friendly.  However, prices are on average much higher than linux options | For mobile devices we could consider using a cloud based servers. The hardware themselves are generally not made for hosting large memory sinks like databases |
| **Client Side** | Being web based the focus would be getting the app to run on supported browser types. Javascript/HTML/css works across the board here instead of choosing a different framework or language depending on the OS.  If development is to be exclusively for mac than finding a dev team fluent with Swift should be the goal. | As long as the Linux system is not just command line based users will often have access to the same web browsers as the other OS.  Being a web based allows for a possible easier development using some type of web focused framework.  If developing for use on this OS sticking with C++, java, or python would be a good choice. | Same as the previous two when it comes to a web based app development. Being Windows OS It has access to the most development tool options and an extensive list of variations to develop applications for windows. | Smaller screen sizes play a big role in how web based apps work and how the user experiences the app. Using reactive and responsive web design will help cater the user experience depending on the device. If developing exclusively for mobile than using react native to be able to deploy to both app stores using one dev team might be the best option. |
| **Development Tools** | Languages/Tools:  Swift  java  FullStacks(MERN, MEAN)  Ruby on rails  Eclipse  Visual studio code  xcode  MongoDB last checked the Designated server options was around $60 per month  Eclipse and Visual studio code are both free to developers  Xcode is $99 per developer | Languages/Tools:  C /C++  java  FullStacks(MERN, MEAN)  Python/django  Eclipse  Visual studio code  pycharm  MongoDB last checked the Designated server options was around $60 per month  pycharm professional starts $199 per year then drops in price  Eclipse and Visual studio code are both free to developers | Languages/Tools:  C /C++  C#  java  FullStacks(MERN, MEAN)  Python/django  Eclipse  pycharm  Visual studio/ code  MongoDB last checked the Designated server options was around $60 per month  pycharm professional starts $199 per year then drops in price  Eclipse and Visual studio code are both free to developers  Visual studio for business can range from $45 to $250 per developer per year. | Languages/Tools:  Swift for ios  C# unity games  Java for android  FullStacks(MERN, MEAN)  Python/ Django  Eclipse  Visual studio code  MongoDB last checked the Designated server options was around $60 per month  React is a free to use javascript library and React Native is a free to use software framework  Eclipse and Visual studio code are both free to developers |

## 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**: Since the goal is to make a web app that will work in browser the operating system recommendation is more for development purposes. Specifically, while it might be a bit more costly I’d recommend windows since we are looking to make an app that works everywhere. It has one of the largest user bases and most compatibility/options with dev tools.
2. **Operating Systems Architectures**: Windows allows for good interfacing with a very friendly user experience that many are already familiar with. While security privileges are not as straight forward as linux for development files can still be managed in a similar manner locking and restricting access by file and folder.
3. **Storage Management**: Storage will be on a serverless database. This will allow for the company to scale as is needed. Also removes the need to physically host a server reducing cost. I suggest building a react native app using AWS s3 for the data the app needs to store.
4. **Memory Management**: The App we are building is for browsers, so local browsers will host the fetched information as needed. Operating systems just need to have browser that is capable of displaying the web page properly which since the app will be made with react native, we can make it a responsive app design allowing it to adjust the users screen size and preferred browser.
5. **Distributed Systems and Networks**: The amazon web service S3 system is highly reliable when it comes to preventing outages. then the front end will could also be hosted on another reliable source such as Heroku another scalable service. After that since the game is web based if the end user doesn’t have a reliable connection their experience could be limited. As there is already an android version of the app if that was built to work offline the user could still use the app if all the information was locally stored.
6. **Security**: Nice part about using AWS S3 the system also has built in user information encryption and as web app we can build in other security features that will secure login information as well. Another option to consider is that this is a simple game app, from an ethical stand point, what sensitive information do we really need to store, we shouldn’t be collecting user data for something like this not even names are really needed the less information we request the less we need to handle as well. If building the app with user security is truly priority, security should focus on not allowing exploits to user’s system from our app i.e. input protection to stop users from running malicious code while connected to others.