

Draw it or lose it web app

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[CS 230 Project Software Design Template](#l6ti7uoag22u) 1

[Table of Contents](#j0zll) 2

[Document Revision History](#grjogdjh5fi8) 2

[Executive Summary](#sbfa50wo7nsh) 3

[Design Constraints](#et92p0) 3

[System Architecture View](#ilbxbyevv6b6) 3

[Domain Model](#h2ehzxfam4o) 3

[Evaluation](#o15spng8stw) 3

[Recommendations](#m8aleynsvzvc) 5

## [D](#grjogdjh5fi8)[ocument Revision History](#grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/14/21 | Andrew Cruz | adding new software requirements and initial reviewing |

**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.

## [E](#sbfa50wo7nsh)[xecutive Summary](#sbfa50wo7nsh)

The Game Room is currently facing a major design problem. The team wants to expand their user base since they only have an android app. In order to expand their user base, our company suggests moving to a web-based distributed environment with multiplayer functionality. The game will have the ability to have one or more unique teams involved, all of which are composed of unique multiplayers.

## [Design Constraints](#et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

* Less secure on a web app
* Application less efficient in web form
* Compatibility with systems and browsers.

## [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.

## [D](#h2ehzxfam4o)[omain Model](#h2ehzxfam4o)

Game service is a singleton pattern. This is a class that has global access but only one instance. Every game is an object made with the gam class. The game class connects the team and player classes to the entity which those players are using to play the game.

**"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.**

## [E](#o15spng8stw)[valuation](#o15spng8stw)

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 is based on the linux kernel which makes it secure, but has limited support with servers. | Linux is the most secure platform and has amazing support with servers. Linux is also optimized for servers. | Windows has great support for servers but is less secure. Better hardware will be required to run windows server. | Mobile devices are based on linux and are secure. They aren’t designed to run servers. |
| **Client Side** | Programming for Mac can be expensive because it requires a whole new client for a minimal amount of people using and the expertise is needed since its less popular. | Linux is extremely popular for development of programs that require servers. It is difficult to develop on often times and requires immense expertise. | Windows is the most popular and easiest client to develop for. Cost will be low and expertise is plentiful. | Mobile devices are extremely popular and are linux based, so there should be a good level of expertise. Cost can be high if developing for IOS and Android. |
| **Development Tools** | Java, Java Eclipse, Python, Pycharm. | Python, C++, Java.  Eclipse and visual studio are great. | Python, C++, and Java.  Eclipse, Visual studio, and Netbeans IDEs are awesome. | Python and Java are best.  Visual studio seems to be the most popular for development on Mobile. |

## 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**: IOS would be an excellent operating platform to move to since the gaming room already has an android app and has provided access to desktop users with the new web version.
2. **Operating Systems Architectures**: IOS is one of the most popular phone operating systems in the world and has major market share in developed countries. IOS users technically have access to Draw it or Lose it, but a full fledged app would take things to the next level and give a much better experience.
3. **Storage Management**: Storage would be extremely similar to android, there is very limited storage and anything that could be stored in the cloud, should be stored in the cloud. iPhones have limited storage capability and their storage ins’t expandable like many android phones.
4. **Memory Management**: IOS will also share similar memory management techniques to the android app. The core application and the current image being rendered would be the only thing we would keep in memory.
5. **Distributed Systems and Networks**: The server handles what image gets served to each client, and who wins. The client is responsible only for connecting to the server and verifying the image is correct before the start of the game. The server should’t have any trouble with multiple platforms playing together seamlessly. Outages do pose an issue for Android and IOS users due to the likelihood of them being on a mobile network. If they disconnect from the server, and someone hasn’t won yet, they should be able to continue to guess, if someone has won, they simply lose that round.
6. **Security**: Fortunately, IOS is a linux based operating system and Apple has their software heavily locked down. Every app on the Apple App Store is vetted by Apple which already makes the platform very secure. As for our app security, the only data we want to send to the server is our login credentials, and our guess time essentially.