

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 02/05/2022 | Janece Gates | <Brief description of changes in this revision> |

**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 client, The Gaming Room, is a social media gaming company that wants to build a multi-player game called Draw It or Lose it. The game currently runs exclusively in an Android app, however the client would like to create a web-based version. This means that any device with Internet connection will need to be considered, this includes Mac, Linux, Windows, and mobile devices. Each of them interact with their respective internet applications differently.

## [Design Constraints](#_2et92p0)

Here is a list of Design Constraints that were discovered:

* Game needs to run on all major Operating Systems and internet accessible devices.
* Game needs to support multiples players on multiple teams
* Only one instance of each game at any given time
* The game will need to keep track of and synchronize the passage of time for each user
* Needs to manage and utilize a database of stock images
* Needs to be built on iOS and Android platforms
* Need for different development kits
* Test various mobile devices for compatability

## [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 GameService class would be on the server side. There will only be one instance of it, and this is supported by the fact that in the UML diagram, all of the member variables are static variables
* The Game class uses 0 or more Team objects
  + This addresses the need for the Game to support including multiple teams
* The Team class uses 0 or more Player objects
  + This addresses the need for a team to support having multiple players
* The Game, Team, and Player classes all inherit from the Entity class
  + This demonstrates the Object Oriented Programming concepts of encapsulation and abstraction.

**"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.**

* The Driver uses the SingletonTester class
  + This facilitates testing
* The GameService class uses 0 or more Game objects

## [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 is much more secure however is very costly.  OSx licenses is less costly than Windows. OS X server costs $499 for 10 client licenses, while the unlimited client server license is only $999. Compared to Windows unlimited license of $6195 | Much cheaper option compared to Mac and highly customizable with top-tier security.  Enterprise version of Red Hat is the only license version that costs. The current cost is $349 All other licenses are free. | Less secure, less customizable, and slower than the Mac or Linux, but support a much wider array of applications than the other two.  The Windows Server 2022 Standard with 16 Core Licenses is (at this time) $594.97 | Web hosting on a mobile device, regardless of Operating System, has the advantage of being much more space efficient and cost efficient, however has the disadvantage of being restricted to simple web apps.  The KS Web Pro app allows you host websites from your Android mobile devices. The cost appears to be $3.99 for the download.  Despite the advantages, mobile devices, although possible, are not at all suitable for hosting a web-based application. |
| **Client Side** | On mac, the most popular supported browsers are: Safari, Google Chrome, Firefox  Google chrome, Firefox, and safari all include a feature called “developer tools” which facilitates both web development and web application testing/debugging.  Safari is automatically installed on Macs so it can be more costly to purchase a Mac to run tests. | On Linux, the most popular supported browsers are: Google chrome, Firefox  Google chrome and Firefox both include a feature called “developer tools” which facilitates both web development and web application testing/debugging  Linux OS is free, the upgrades are free, and it is less likely to have virus attacks. The cost to test on a Linux OS is really low, because it be installed on Mac or Windows machines | On Windows, the most popular supported browsers are: Microsoft Edge, Google Chrome, Firefox  Google chrome, Firefox, and Microsoft Edge all include a feature called “developer tools” which facilitates both web development and web application testing/debugging  Testing on a Windows PC isn’t as costly as a Mac because all of the browsers listed have free downloads.  However, the cost to continue to upgrade the machine can be costly in the long run. | On mobile devices, the most popular supported browsers are: Safari, Google Chrome    Google chrome and safari both include a feature called “developer tools” which facilitates both web development and web application testing/debugging. Furthermore, there are 3rd party software providers, such as smartbear.com that provide technology to facilitate cross-platform testing for less popular web browsers.  Testing on different mobile devices can be costly because of the varies devices on the market, or still in use (older models). The browsers listed are free to install. |
| **Development Tools** | We can use Python and Javascript in order to develop the game endpoint on Mac Systems. The IDE to use for Mac would be XCode.  The cost for XCode is included in the price of the computer.  Sublime, and Pycharm has a free install as well. | We can use Python and Javascript in order to develop the game endpoint on Linux systems. The best IDE to use would be Sublime or VS Code.  Sublime, Pycharm and VS Code all have free installations. | We can use Python with the .NET framework in order to develop the game endpoints on Windows systems. The best IDE for this would be Visual Studio.  Sublime, Pycharm and Visual Studio all have free installations. | If coding on a mobile device both platforms have mobile IDEs that can be used. There are downloads for coding Java, Python, Swift and XCode. Out of all the languages listed, Python would be the easiest to use. On Android one might use Pydroid as the mobile IDE. On iOS one might use Pyto as the mobile IDE. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform:** I would recommend Linux as it is largely the best of both worlds between Mac and Windows.





**Operating Systems Architectures**:

Like Macs, Linux systems are very secure, not many viruses are tailored to them, and is very high performance. Like Windows, Linux is very customizable (in fact, virtually every single aspect is completely customizable), low cost (most versions are free), and supports most web hosting software.

**Storage Management**:



Linux can use the secondary storage available to the machine to store the pictures and user

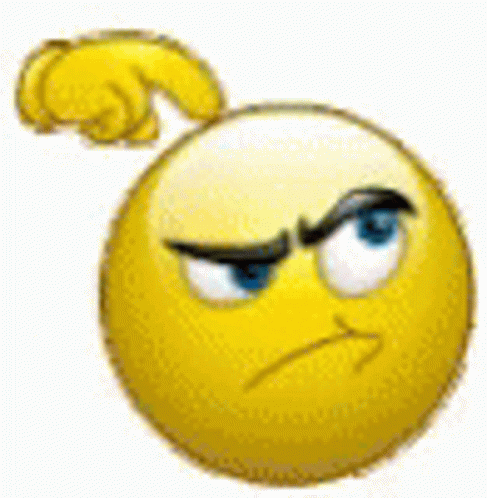
information and maintain it with its free-space management, storage allocation, and disk scheduling

features. We should use a database with indexing to make lookups of pictures fast as possible,

however this isn’t a huge concern given the fact that the number of pictures available to the game is

so small (only 200).

**Memory Management**:



The system will want to service as many clients as possible and therefore, will want to be as fast as

possible. This entails efficient cache and memory usage/management. Cache and RAM are faster,

more expensive, and in lower supply than secondary storage options. Therefore, only the most

frequently needed game data will be stored in the cache and memory. We will also be storing game

info such as game scores, user names, picture data, and the amount of time left in a round.

**Distributed Systems and Networks**:



[Download Networking Transparent HQ PNG Image | FreePNGImg](https://freepngimg.com/png/16460-networking-transparent) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

Assuming that the clients of Draw It or Lose It are expected to be spread out over large areas, the

gaming company will want to make use of both Wide Area Networks (WANs) and Local Area

Networks (LANs). One user in City A will be able to communicate with another user in City B with

the data being routed first through the WAN and then the LAN. Since we are running the app

within web browsers, it is safe to assume that as long as the web browser and device are functioning

correctly, each user will have access to the internet and therefore will have an IP address. Messages

can be sent back and forth efficiently by simply targeting the appropriate IP address.

**Security**:



In order for user data to be protected and kept secure between platforms, the data that is sent

between devices must be encrypted. Since the application will be running in web browsers, all that is really necessary to ensure secure information is to make sure we are using the https protocol. In order for the data to be protected while on a given device/platform, the use of usernames and passwords will be necessary. We can also make use of authenticators to filter what *kind* of user can access specific data. Possibly even the use of the secure id token may be necessary for extra security of highly sensitive user data.