

The Gaming Room

# **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 | 03/19/2022 | Patrick Dowd | \*Updated the Executive Summary  \*Reviewed and updated the Design Constraints  \*Described the domain model from the provided UML. |
| 1.1 | 04/05/2022 | Patrick Dowd | \*Updated the Evaluation  \*Added Implementation 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)

The game Draw It or Lose It is loosely similar to the 1980s television game Win, Lose or Draw, where teams compete to guess what is being drawn. Rather than a player drawing images on an easel to help team members guess the puzzle (a phrase, title, or thing), the application will render images from a large library of stock drawings as clues. A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit. We have been asked to facilitate the development of the web-based version of the gaming app. The game is currently available in an Android app only.

## [Design Constraints](#_2et92p0)

**Software Requirements:**

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

**Development Constraints:**

* Active games will need to be hosted on the server side.
* When a team is created, the team name must be verified on server to confirm it does not already exist.
* When a new user is created, the username needs to be verified on server to confirm it does not exist.
* Strongly recommend we host the game as browser based for use on PC and Mac written in JavaScript, while continuing to use existing Android app and potentially develop an iOS app. This model would largely capture the population of prospective users.
  + We need to determine if we can redevelop the code from the Android app, or if we need to write a new version in Swift or comparable language for iOS.
* Web Version needs to be responsive and to run in various browsers, including but not limited to:
  + Microsoft Edge
  + Apple Safari
  + Firefox
  + Google Chrome
* We need to better understand how users sign-on to Draw It or Lost It. Users will need to be able to access their profile regardless of platform.
  + Need to inquire if the client want to use platform unique to platforms like Apple ID or GMAIL.?

## [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)

**ProgramDriver**: *ProgramDriver*runs the application, and uses the **SingletonTester** class to test that there is one instance of *GameService* running.

**GameService:** Only on instance of *GameService* can be run at any time. It can contain zero to many active *Games*.

**Entity:** *Entity* has a parent relationship (is a Superclass) to *Game, Team,* & *Player*. Each will inherit the class variables and class methods.

**Game**: *Games*is a subclass of *Entity*. It is also associated with *Team* and can contain zero to many instances of *Team* in its internal list.

**Team:** *Team* is a subclass of *Entity* and is associated with *Player,* containing zero to many instances of *Player* within its internal list.

**Player:** *Player* is a subclass of Entity.

**"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** | **Advantages:**  -Easy to maintain by non-IT users.  -Useful for small implementation as it can be implemented with $20 software to host from any Mac computer.  - macOS support Windows file system formats including both NTFS and FAT.  **Disadvantages:**  -Services are being depreciated as server renting becomes more widespread.  -Some software are not available on MacOS Environment (Note: Many services are changing to cloud based – reducing this issue).  -macOS can read NTFS and FAT Windows format but Windows cannot read the macOS file system. You need to install 3rd party software in Windows to do so. Some other software like footnotes has compatibility issues while moving files from macOS to Windows. | **Advantages:**  - More versatile and offer greatly enhanced security  - Due to the open source nature of Linux, servers running this operating system are not required to pay for a license, which lowers costs  - They’re more reliable, which allows them to experience maintenance and failure without shutting down when paired with the appropriate redundancy measures  -Most common web hosting OS.  **Disadvantages:**  -As an open source project, you have to pay for specialized support or rely on community of open source developers who may not be able to answer your questions.  -Command line based, no Remote Desktop features. | **Advantages:** -It provides compatibility with other Windows tools. - If a lot of your business’s tech relies on Windows, then Windows server hosting is the natural choice. It will work seamlessly with all the other programs you use, and won’t require you to rebuild or work on anything from scratch. And the tech professionals your company relies on to keep all your systems working properly will already know how to work with the Windows hosting OS.  **Disadvantages:** - It’s not as secure as Linux hosting-  It’s more expensive.-It can be less stable. | Mobile distribution heavily relies on Server hosting and stable bandwidth to work. Can be compatible with various hosted servers, most commonly Linux. |
| **Client Side** | **Advantages**:  -Uniform Graphical Interfaces across applications – making easier navigation for end-users.  -Efficient and quality performance.  - macOS support Windows file system formats including both NTFS and FAT.  **Disadvantages:**  -Mac hardware tends to be more expensive.  -Fewer apps are developed native to macOS. | **Advantages**:  -The software is open source and free to use with no cost.  **Disadvantages:**  -Not commonly used by end users.  -As an open source project, you have to pay for specialized support or rely on community of open source developers who may not be able to answer your questions. | **Advantages:** -Most commonly used operating system. **-** A great platform for **developing**  web sites and native **windows** apps. Especially if you use Visual Studio or Visual Studio Code.  -Easier to support with existing infrastructure.  **Disadvantages:** -Less secure. -Software more easy manipulated by end-user. | Require Software to be coded unique to each operating system OS (and sometimes unique to device – phone or tablet – to be formatted to include specific features or display sizes).  Can be costly to develop on multiple operating sytems and to maintain those products.  Different distribution models by OS – can affect distribution of updates and ROI. |
| **Development Tools** | **Swift:** Swift is a powerful and intuitive programming language for iOS, iPadOS, macOS, tvOS, and watchOS. Swift code is safe by design, yet also produces software that runs lightning-fast.  macOS supports C, C++, CSS, Java, JavaScript, HTML, PHP, Perl, Python, Ruby, or Vala. | Linux supports C, C++, CSS, Java, JavaScript, HTML, PHP, Perl, Python, Ruby, or Vala. | Windows supports C, C++, CSS, Java, JavaScript, HTML, PHP, Perl, Python, Ruby, or Vala. | Dependent on operating system.  **Swift** is one of the **best mobile app development** frameworks available in the iOS platform.  **Java** is the official language for Android App Development and consequently, it is the most used language as well. |

## 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**: I recommend hosting the game on a Linux server, distributing the sessions via the web. This would allow us to host the game in HTML on macOS, Windows & Linux, while creating unique apps for iOS and Android. App versions could also be developed for other OS, should the need for additional functionality arise. We would be able to write our program in Java to be hosted on our server. Linux is one of the most commonly used server types and would function ideally for our game distribution.
2. **Operating Systems Architectures**: We will need to host the application written in Java with ports setup for communication through the DMZ to the user device. We would host a Database on our server (perhaps in SQL) that would store the player data and game session data. Data transactions between our Server and the Client would need to be encrypted. The user interface type – HTML or app – will depend on the particular device the Client is using.
3. **Storage Management**: The OS will need to dedicate storage to the game driver.

As we design this application (and site) we need to consider the size of the application and the amount of RAM that will be used when our application is run. If we intend to implement our application in a mobile environment, we must keep our application small, as storage availability is significantly more limited.

Runtime information – such as game session data and user information – can be stored on our server, freeing permanent storage on our device. Our server will host a library of 200 high-definition image files to choose from, each one approximately 8 megabytes in size. If we were to host those files on the local user application, it would increase the size by of the application by 1.6 gigabytes. By storing those files and only using them in game instances where they are appropriate, we can significantly decrease the application file size and reduce the memory burden in an active game session on the client side.

1. **Memory Management**: The game program will need to be hosted on the client device. We will need to provide minimum RAM requirements for use of our game to ensure the product runs consistently. We need to dedicate memory to active session data being taken from system calls to our hosted server. As we design our game, we need to ensure that we only instantiate objects when they are required in the application and deallocate memory for those items when they are no longer needed.
2. **Distributed Systems and Networks**: Game sessions will require stable internet connection with a minimum required bandwidth. A database will need to be hosted on a where authenticated game users can make system calls to their game session on our server to retrieve or mutate values relevant to their active game session. This will require the use of JSON (if our application is built in Java) and SQL Database management on our hosted server. We will need to dedicate ports for data transactions for the game session, as well as any potential user chats should the feature be included/required.
3. **Security**: Data transactions between the client system and the server need to be encrypted. Calls to the server should only access the game instance, and the specific variables that interface with the game session and user interactions. The user should not be able to access other player or game data on the server. We can accomplish this by assigning access levels to each user account to dictate what elements they have access to, and assigning users to specified game sessions that they are actively participating.