

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

**Recommendations 6**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/22/24 | Joseph Caron | Initial software design documentation |
| 1.1 | 06/05/24 | Joseph Caron | Added Evaluation field information |
| 1.2 | 06/18/244 | Joseph Caron | Added Recommendation information |

**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 goal is to create a web-based application game designed on a pre-existing Android game by The Gaming Room. It needs to be able to be run on multiple platforms and will need to be able to pull from an extensive library of stock drawings to run. It also needs the ability to have multiple teams made up of multiple players all with unique names, and only one instance of a game can be in memory at any point.

## Requirements

*The Gaming Room wants the game to be based off of their existing Android game. It must be a web-based game usable on multiple platforms. It must be able to connect to and pull from a large library of stock drawings. It must be able to host multiple teams made up of multiple players, all with unique names. Only one instance of a game can be in memory at any point.*

## [Design Constraints](#_2et92p0)

* Game design must be streamlined so it will visually and functionally work on different devices such as PCs, tablets and smartphones.
* Cross functionality needs to be included so that multiple players from a variety of devices can play together.
* The game must provide feedback to the team leader if the chosen team name is already taken so they may choose another.
* Unique ids for each game, team and player to ensure each game has only one instance in memory at a time.

## [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 ProgramDriver class has the main() function that drives the entire program. It has directional association with the SingletonTester class to ensure that there is only a single instance of any game at a time. These two classes are separate from the other classes. The Game, Team and Player classes are child classes of the Entity class and inherit attributes and methods from it. The three child classes and the GameService class are all associated classes. The relationship between GameService and Game is 0 to many, meaning a Game cannot have a GameService, but a GameService can have many different Games. Likewise, the relationship between Game and Team is 0 to many and between Team and Player is also 0 to many.

**"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** | Pros: Clear and user-friendly GUI. Ready made to access, build and control web server.  Cons: Expensive to maintain, doesn’t work well with many third-party applications. | Pros: Robust built in command system and server. Inexpensive to maintain.  Cons: Steep learning curve means possible slow development. | Pros: Well-rounded, built-in command prompt, highly distributed and well-integrated with third-party applications.  Cons: Expensive to obtain licenses, not as much built-in as other systems. | Pros: Potential to cross develop between various mobile OS with third party database applications.  Cons: Impractical, device hardware limitations, too much variation between devices. |
| **Client Side** | Pros: Clean user-friendly GUI. As a popular user OS, highly distributed and easy for users to navigate.  Cons: Expensive for users | Pros: Low cost, open source.  Cons: Lower user distribution, very steep learning curve for unfamiliar users. | Pros: High user distribution and familiarity. Mid range cost to users.  Cons: More expensive than Linux, potential to be buggy. | Pros: Maximum user distribution and familiarity. Flexible in mobility.  Cons: Low screen real estate means harder to create a user-friendly interface. |
| **Development Tools** | Visual Studio Code, Homebrew, PyCharm, Xcode, iTerm2, Flask, Swift. | Python, Visual Studio Code, Git, MySQL. | Visual Studio Code, PyCharm, Eclipse, MySQL, Git. | C++, Javascript, HTML, fs, CSS, Python. |

## 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 using Linux servers as the operating platform for this application. Its versatility will be an asset in creating the application. The low cost and high security will also make running the program easier in the long run.
2. **Operating Systems Architectures**: The Linux operating platform is a UNIX based system, built on the Linux Kernel. Kernel acts as the system’s interface between the shell and the hardware. This means the application will “talk” with the Kernel to interface with the hardware, but it cannot be accessed directly from the application itself, adding to the overall security of the system.
3. **Storage Management**: I highly recommend using AWS serverless storage solution for Storage Management. The price point is much lower than purchasing or renting servers, and the maintenance costs are rolled into the usage costs. On top of that, the auto scaling on AWS means whether the game is running 100 instances or 10,000, it isn’t an issue on The Gaming Room’s end.
4. **Memory Management**: Linux has fantastic memory management built right into the platform. It uses virtual memory and demand paging to allow different instances of the program to run in parallel, which when coupled with the AWS auto-scaling, will almost completely remove any physical memory restraints and allow for fast loading and smoother running.
5. **Distributed Systems and Networks**: As this application is a web-based game, cross communication between platforms is relatively simple. Many web browsers are available on all platforms, such as Google Chrome. Chrome can be downloaded and run on any PC, Mac, tablet or smartphone, allowing any user with any of these devices to access the application and play with people using any of the other devices as well.
6. **Security**: On server side, AWS will be responsible for security, as that is part of the serverless architecture package. On the client side, as the program will be accessed only through web browsers, the built in security of the browser will play a large role, as well as the security of the device used to access the game. All operating systems have built in security such as data encryption and external firewalls, while many browsers also block attempts for outside parties to access the open connections.