

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 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/17/2023 | Christopher King | Updated to the following sections: cover page, Executive Summary, Design Constraints, and Domain Model |
| 1.1 | 04/1/2023 | Christopher King | Updated Evaluation section. |
| 1.2 | 04/15/2023 | Christopher King | Updated 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)

Development of a web-based game “Draw It or Lose It” will be created for The Gaming Room by Creative Technology Solutions (CST). Currently the game is only available as an Android only app and will be expanded to cover multiple platforms. This document will capture the essential components that will directly affect the outcome of this project. The game will have one or more team with multiple players assigned to each team. Each game, team, and player will be unique and will only have one instance of the game exist at any given time.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Concerning the clients requirements, the following constraints have been identified:

* **Cross-Platform:** The game is currently available on Android but will need to be adapted to work across all platforms. Utilizing a web-based application will help with cross-platform availability.
* **Multiple team/players:** Each team and player name will need to be unique and system will need to have the ability to check the system to see if either exist on the system already prior to accepting the name.
* **Unique ID’s:** There can only be one instance of each game, team name, and player at any given time. To accomplish this, each item will be given unique ID’s to ensure that only one instance of the game can run at any given time.
* **User Interface:** Since the game is already available for Android, the user interface will need to be adapted to other platforms using the same key features and layouts to have a consistent view.
* **Images and Copyrights:** Due to the need to utilize images for this game, all images will either need to be in the public domain or will need to have copyrights obtained to legally us the images.

## [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 following Unified Modeling Language (UML) diagram depicts the way the system will be designed to accomplish the goals of this document.

* The Entity class is a super class and extends to the Game, Team, and Player classes. Entity class contains the common attributes ID, name, and associated setters and getters.
* The Team class inherits the attributes from the Entity class and also has a private list along with attributes to assign players to the team.
* The Player class inherits the attributes from the Entity class and also has the attributes to assign a name to the player. The player class is associated with the Team class allowing the player name to be assigned in the Team class.
* The Game class also inherits the attributes from the Entity class and also has a private list of the teams along with attributes to assign game ID’s to the player and team. This has an association with the Team class to allow teams to be assigned to the ID.
* The GameService class inherits the attributes of those classes above it. The GameService contains private attributes utilized to get the next game, next player, and next team. GameService class is a singleton class to ensure that only one instance of a game to exist in memory at any given time.
* The ProgramDriver class contains the main which initiates the program and also utilizes its association with the SingletonTester to ensure that only one instance of the game exists in memory at any given time.

**"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** | Costs can be more expensive with less Mac OS users and reduced demand. Licensing could be on the expensive side. Tends to be more secure then other operating systems. Due to Mac OS server being discontinued, will need a third party client to host the server. | Linux utilizes open-source software. Would have the lowest hardware costs and be the most cost effective option. With the community constantly updating against potential threats, would be a more secure system. Linux has proven stability. There is not as much software available as Windows. | Windows has a simple server based deployment with a large range of software support. There is a wide range of hardware options that can be utilized. Windows is less secure than Mac and Linux. Updates are solely provided through Microsoft. | Although mobile devices could be utilized on the server side, due to the lack of power and memory compared to computers, it would not be an advisable solution. The cost would be the lowest, but the trade off would be power and memory. |
| **Client Side** | Cost would be similar to Windows due to both not being open-source. The time would depend on expertise with Mac products. Less time would be required for those that have previous experience with Mac. | Cost would be low due to Linux being open-source. Time would depend on experience with Linux. Linux is not as widely used as Windows or Mac OS, so support is limited. Linux could be difficult even with veteran Linus users. | Cost would be similar to Mac due to both not being open-source. The time would depend on expertise with Windows products. Less time would be required for those that have previous experience with Windows. | Cost would be relatively low due to the flexibility of mobile devices. Majority of the population has previous experience with mobile devices so set up would be minimal. One thing that will need to be taken into account would be the multiple OS options for mobile devices. |
| **Development Tools** | Swift would be the most common language used to write programs for Mac. Multiple IDE’s are available to develop with Swift such as: Atom, Sublime Text, Visual Studio Coad, AppCode, and Code Runner. | Eclipse and Atom are the most commonly used IDE’s on Linux. Languages would primarily be Java, C++, and Python. | Visual Studio would be the most common IDE utilized on Windows. VS allows development in most languages, though C++ is the most widely used. | Depending on mobile phone OS type, different tools would be needed. For Android, the most widely used platform would be Android Studio. For iOS, iCode would be the most widely used platform utilizing Swift. Unity is also widely utilized which can then be converted to an Android app or iOS app. |

## 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**: Based on our evaluations of Draw it or Lose it, we recommend utilizing a Linux operating system deployed on a cloud-based (serverless) architecture. This will reduce complexity and inefficiencies while promoting scalable and efficient solutions that require less maintenance. The services will be accessed through many methods like HTTP, API, etc. which will allow better support distributed across varying computing environments.
2. **Operating Systems Architectures**: Utilizing serverless architecture is built through modularization which will allow middle layer solutions to be utilized to deliver a robust app including business logic functions, content delivery, and security services. Linux on a serverless architecture will allow a lightweight solution that utilizes less disk space to be provided. This will improve efficiency and processing for improved UX. Utilizing Linux also executes functions in form of libraries that do not have access to the kernel allowing for a more secure system.
3. **Storage Management**: Storage Resource Management will be implemented in managing the server data. Utilizing SRM will reduce capital/operational expenses, makes management easier, enhances performance, provides increased speed and flexibility, and higher availability. This will also allow for better scalability options when there is an increase or decrease in users playing the game.
4. **Memory Management**: Memory management will be addressed within Linux operating system itself. Partial memory execution will be employed through virtual memory and demand pinging. This will allow multiple sessions/programs to run in parallel which eliminating the physical memory size constraints. This will allow the game to load and run faster which will in turn improve the UX.
5. **Distributed Systems and Networks**: Since Draw it or Lose it will be a web-based game, it will be run through web browsers which will allow for users with different operating systems to have the ability to play the game. Currently Google Chrome is available on all platforms and can be a recommended browser to play the game on. Being that this game will be hosted on a cloud, anytime there is an issue with a specific node, the game session would be able to be moved to another node automatically to reduce any downtime for the players.
6. **Security**: Since this will be a cloud-based system, the provider will be responsible for the security of the data on the server. To increase security, a role based security option will be utilized. Accessed will be split between various account options such as: admin, team, player, and user which will have predefined access abilities.