

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 5/20/2023 | Winnie Kwong | Updated Executive Summary, Requirements, Design Constraints, Domain Model, Evaluation, and Recommendations to fulfill the client’s request. |

**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)

Client’s name: The Gaming Room

The Gaming Room is currently developing a new game called “Draw it or Lose it” that it currently only available on the Android app. The Gaming Room has requested to streamline the development by preparing a software design document and to begin developing the game application and software requirements.

The purpose of the game is to allow multiple teams to play four rounds per minute. A picture is pulled from the company’s rendered library database and one team will continue to guess until the time runs out. If the team does not solve the puzzle before the time expires, the other team has the chance to solve the puzzle within the 15 second time limit.

## Requirements

**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 and can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

* The game application needs to be secure to protect user data from breaches. If the application does not have proper security measures, users are at risk of being exploited by hackers.
* Another design constraint is having scalability issues to handle large numbers of users. The application needs to have enough capacity to handle all requests, including the player’s needs, or it can cause the application to crash or cause server overload.
* The game application must be accessible to run on multiple platforms. If not, there are high risks of application malfunctions, such as slowing down or freezing due to different devices’ hardware or software miscommunication.
* The game needs to be compatible with all types of platforms. Since the game application is only on the Android app, the current code must be rewritten or converted from the existing code to other relevant programming languages and tools to meet all software requirements. Being incompatible with all platforms is a design constraint because it will require more people to manage the code and its functionality.

## [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 UML class diagram includes vital information that structures the outlines the project’s functionality. In the diagram, Game, Team, and Player all are inherited from Entity meaning all three classes have a relationship with Entity and will inherit Entity’s attributes of ID and name. By inheriting the Entity class, the properties are acquired and passed along to the child classes which increases re-usability.

Since there is a relationship between the classes, it shows aggregation and is utilized in the diagram to show that three classes can exist independently without its parent class Entity. Utilizing aggregation means the Entity contains the three classes, but the three classes can exist without the Entity.

In the diagram, there is a composition connector that states that the ProgramDriver cannot exist without the SingletonTester. The ProgramDriver uses the SingletonTester, to show that only one instance of the GameService class can exist in memory at any given time.

Also, in the diagram, it references that there is a cross-relationship with GameService, Game, Team, and Player. The diagram represents that a game service can have multiple games that includes multiple teams of multiple players during each round of the game in the instance.

**"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** | Mac has a flexible terminal that can configure servers or make other changes and has great security however it is very expensive. | Linux is affordable compared to Mac and Windows and stable because of its processing power. However, Linux is difficult to navigate. | Windows offers more software availability but crashes a lot and lacks system stability and performance. | Highly popular, offers portability, and better compatibility but is costly and has a poor security system. |
| **Client Side** | Software development is expensive for clients and requires a moderate amount of time and expertise. | Linux is very low in cost, basically free but requires a lot of time and expertise to understand the operating system. | Windows is average cost but is very user friendly, so it requires less time and expertise to utilize. | For software development considerations, cost varies but ranges between a Mac and Windows. It is convenient for mobility but required moderate time and expertise. |
| **Development Tools** | Programming languages and tools used on Mac are C, C++, Swift, HTML, CSS, Javascript, Java and Python. | Programming languages and tools used on Linux are Visual Studios, Eclipse, HTML, CSS, Javascript, Java and Python. | Programming languages and tools used on Windows are Visual Studios, Eclipse, NetBeams, HTML, CSS, Javascript, Java and Python. | Programming languages and tools used on mobile devices are HTML, Switft, C++, Java, and 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**:

An appropriate operating platform allowing The Gaming Room to expand Draw It or Lose It to other computing environments would utilize Microsoft Windows. Windows offers a wider variety of software and is compatible with all hardware while supporting plug-and-play features. Furthermore, the operating platform is user-friendly and is compatible with most programming languages.

1. **Operating Systems Architectures**:

Windows would be the best recommendation for an operating system architecture. Windows is user-friendly that allows maintainability and expendability for development. Also, because of the smooth communication between the different components, it enhances the performance of the operating system. Since Windows is a monolithic architecture, all the various components are in a single kernel that communicates by function calls.

1. **Storage Management**:

Storage management helps to organize a balance between performance and storage capacity. An appropriate storage management system would be using Windows storage space. Windows storage space can also group multiple drives into a storage pool, making it easier to locate file creation and placements of large projects. Windows also offers automated defragmentation, allowing one’s computer to run at peak performance.

1. **Memory Management**:

Memory management controls and maintains the main memory and how it transfers the primary memory to the disk. Windows memory management helps allocate and deallocate virtual memory while supporting memory-mapped files, shared memory, and copy-on-write, enhancing performance and ensuring that applications have the space to run and process the data as efficiently as possible.

1. **Distributed Systems and Networks**:

It can be accomplished to communicate between various platforms by designing a cross-platform application corresponding to a central server. A cross-platform application ensures that all players can play on a mobile device or a MacBook. The application will communicate with a Windows server to do the heavy lifting of data processing, which can prevent connectivity issues. Therefore, the application uses the minimum required bandwidth for all devices preventing possible outages.

1. **Security**:

Windows security has built-in security software that scans for malicious software, and User Account Control (UAC) prevents unauthorized changes. The Windows security system is updated regularly and can help to detect any of the latest threats. UAC ensures that only the changes or installations intended to happen are allowed, and all others are prevented. Also, Window’s security system makes it easy to navigate and customize the settings to fit the customer’s preferences.