

Mohamed Elmarzougui

# **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 | 11/12/2023 | Mohamed Elmarzougui | Adding information related to software design |

**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 Gaming Room's project to transform an Android-exclusive game, Draw It or Lose It, into a web-based platform accessible across multiple devices. The game revolves around multiple teams, each engaging in four rounds lasting one minute. Teams try to guess phrases derived from a library of images, with the challenge of solving before the timer expires. If unanswered, the opposing teams have a chance to guess within a 15 second window.

## 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)

* Multiple Players per Team: Game logic and user interface must efficiently accommodate multiple players within each team.
* Must run on multiple platforms: Cross-platform compatibility is a priority.
* Only one instance of the game can exist at any time (to ensure streamlined memory management and prevent conflicts.)
* Unique Game and Team Names.

## [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 Entity class establishes a connection between the Game, Team, and Player classes, indicating that they all inherit information from the Entity class. In UML, this relationship is visualized through inheritance, designating Entity as the superclass. Analyzing their interconnection, we observe that Team and Player exhibit a "has a" relationship type. Specifically, Game has a Team, and GameService has a Games. In UML, this association is referred to as aggregation (HAS-A), signifying that one class instance holds a reference to another class instance. In practical terms, when a user "has a," it means they possess an instance of one class while having a reference to an instance of another class. Examining the UML diagram, we notice that GameService has a reference to Games, Games a reference to Teams, and Teams references of Players.

**"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 servers leverage a Unix-based operating system, providing a stable and secure environment for hosting web-based software applications. | Advantages:  Linux's cost-efficiency, being free and open-source, is ideal for server hosting, particularly for organizations with budget constraints.  Weaknesses:  Linux pose a challenge to learn how to use it. | Windows servers feature a user-friendly graphical interface for simplified server management.  Weaknesses:  Windows servers may entail licensing costs, making them cost more compared to open-source alternatives. | Mobile devices demand server-side architectures capable of handling high levels of user interaction.  Advantage: Server-side infrastructure enables the implementation of push notifications.  Weakness: Server-side components must address security concerns related to data transmission and storage |
| **Client Side** | development for Mac entails budget considerations for Apple hardware and licensing fees. | There is a lot of expertise and time required.  Linux data is required to use the operating system. | More expensive than Linux systems. Easy to learn and understand how to support a Windows setup. | Provide flexibility to clients or even developers to see updates at any place. Slightly more difficult to implement than other devices. |
| **Development Tools** | Key Development Tools: Swift is the primary programming language for Mac applications, and Xcode serves as the integrated development environment (IDE) for building, testing, and deploying software on macOS. | Linux software development commonly involves using C, C++, and Python, with compilers such as GCC and Clang facilitating the translation of source code into executable binaries. | Windows software development is prominently anchored in C# and Visual Basic .NET . | Xcode is the primary IDE for iOS app development, while Android Studio serves as the official IDE for Android, providing essential tools for mobile app creation. |

## 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**: The most appropriate operating platform that will allow The Gaming room to expand Draw It or Lose It to other computing environments is Windows. Windows is best as it won’t run into a shortage of IDEs to work with
2. **Operating Systems Architectures**: Microsoft Windows, developed and published by Microsoft, is a graphical operating system that facilitates file storage, software execution, gaming, video playback, and internet connectivity.
3. **Storage Management**: Windows storage sense is useful for local file management. For a cross-platform application, cloud-based storage solutions like AWS S3 or Google Cloud Storage can provide scalable and accessible storage.
4. **Memory Management**: Windows 10's storage sense feature enables the storage and administration of Draw It or Lose It photos and player data, consolidating them securely in a unified memory space.
5. **Distributed Systems and Networks** Network-based multi-user interaction systems, like network games, commonly involve a shared database accessed by physically distributed players who interact over the network. Presently, developers of network games often need to create the shared database and inter-player communication functionalities from the ground up.
6. **Security**: Implement end-to-end encryption and secure authentication for user data across Mac, Linux, Windows, and mobile devices. Regularly update security measures. Windows comes with built-in security protection software.