

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 | 03/24/24 | Parker McKay | First Revision |
| 1.1 | 04/07/24 | Parker McKay | Incorporated Evaluation Section |
| 1.2 | 4/21/24 | Parker McKay | Added 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 Gaming Room wants to develop a web-based game that can serve multiple platforms and is based on the current game “draw it or lose it”. The current game is only available on android devices. The game is a multi-player concept with multiple teams using four, one-minute rounds. The team in play must guess the picture from a library of images until either getting the correct answer or running out of 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)

This game must be able to run across multiple platforms. A system will need to be created for each individual platform. Teams must consist of multiple players. This will need to have a number cap to ensure teams have enough players. Since each team name needs to be unique a database must be used to save team names and cross reference to ensure a name is not used multiple times.

## [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)

A relationship is created between Game, team and Player class’s with Entity. They all inherit from Entity which is a superclass. With UML we can see this by inheritance. Team and Player is a “has a” type while Game has a Team and gameService has Games and this is called aggregation.

**"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 is easy to access and configure servers. The graphical user interface is easy to use | Linux is Difficult to navigate. Uses command shell for simple server config and access. It is very inexpensive. | Expensive Server side. Uses a command prompt | Not as good specs as other devices. Each mobile device varies. |
| **Client Side** | Expensive. Skills needed to navigate OS | A lot of expertise required. Time needed. Linux data is required to use OS | Easy to understand and learn to support. Minimum expertise necessary. More costly than Linux | Provides flexibility but more difficult to implement. |
| **Development Tools** | Languages consist of HTML, CSS and JavaScript. Tools include Visual Studio, GitHub, PyCharm. | Languages that consist of HTML, CSS and JavaScript | Languages consist of HTML, CSS and JavaScript. Tools include Eclipse, Command Prompt, PyCharm. | HTML, CSS and JavaScript will all be used. Needs Libraries to support languages. IDE’s for programming consist of HTML, C+ + 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**: For expanding Draw It or Lose It to other computing environments, I recommend using a cloud-based operating platform. Cloud platforms offer scalability, flexibility, and ease of deployment across various environments.
2. **Operating Systems Architectures**: The chosen operating platform architectures should support both Windows and Linux environments to accommodate a wide range of users. This flexibility ensures compatibility with different operating systems and allows for seamless expansion to various computing environments.
3. **Storage Management**: For storage management, I recommend leveraging a cloud-based storage solution provided by the chosen operating platform (e.g., AWS S3, Azure Blob Storage). These services offer scalable, reliable, and secure storage options with features such as data redundancy, encryption, and access control.
4. **Memory Management**: The recommended operating platform should incorporate efficient memory management techniques to optimize the performance of Draw It or Lose It software. This includes memory allocation, garbage collection, and virtual memory management to ensure optimal resource utilization and responsiveness.
5. **Distributed Systems and Networks**: To enable communication between various platforms, a distributed systems architecture with a robust network infrastructure is essential. Implementing a microservices architecture facilitates communication between different components of Draw It or Lose It across distributed environments.
6. **Security**: Security measures should be integrated into every aspect of Draw It or Lose It to protect user information across various platforms. This includes implementing encryption protocols for data transmission, access control mechanisms to authenticate and authorize users, and regular security audits and updates to address potential vulnerabilities.