

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template 1**](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 4**](#_heading=h.2s8eyo1)

[**Evaluation 5**](#_heading=h.17dp8vu)

[**Recommendations 6**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/26/24 | Tyler Lacroix | Added information for Initial Design Document. |
| 1.1 | 2/07/2024 | Tyler Lacroix | Added and Updated Server/Client/Development information in the ‘Evaluation’ and ‘Recommendations’ sections. |
| 1.2 | 2/20/2024 | Tyler Lacroix | Updated ‘Recommendations’ section. |

**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](#_heading=h.35nkun2)

The client, The Gaming Room, wishes to create a multi-platform competitive guessing game called, *Draw It or Lose It*. There currently exists a version of the game on Android, and The Gaming Room would like a web-based port of the game for modern operating systems. The game consists of multiple teams of players guessing at what the image being drawn is. The drawings to be used are taken from a library of stock images. Additionally there is a timer to make sure rounds don’t go on too long, and there are four rounds total before a winner is chosen.

## Requirements

* The game must be accessible through a web-based client.
* Each individual game must be able to support multiple teams of multiple players each.
* Player, Team and Game IDs must be unique to avoid conflict with other running games.
* Application must consist of only a single running instance of the game.
* There should be a hard 45 second limit for each round. 30 seconds for the first team and an additional 15 seconds if the initial team fails the guess.
* There should be 4 rounds per game, and the team that wins the most rounds wins the game.

## [Design Constraints](#_heading=h.1ksv4uv)

* **Cross-Platform support**

The wish of The Gaming Room is to expand their audience from currently being limited to Android devices to other platforms. As the wish is to have the game be web-based, the only real requirement for possible platforms is that it MUST be able to communicate with HTTP.

* **User interface**

As the current version of the game is exclusive to android devices, the UI of the game must be overhauled for the web version to have support for multiple common PC resolutions.

* **Support for multiple teams/players per team.**

The biggest requirement for the game overall is that it must be team based. This means that there must be a way for players to be identified as being part of a team in the backend. This can be achieved by having unique IDs for both players and teams, and having player IDs being associated with a respective team ID.

* **Images and licenses to use them.**

As the game relies heavily on stock images, The Gaming Room must have the proper licenses to use them. The android version of the app currently uses these images so it’s safe to assume they currently have the rights to their current library.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

## 

The following UML diagram was used to layout the desired structure of the application. It outlines everything needed from the game driver service to player and team handlers. The base superclass the game driver uses is the Entity superclass. This class contains a lot of the basics that the Player, Team and Game objects require to work, such as a long var to store the unique ID and a few basic setters and getters for the ID and name. Separately from these there is a program driver, which is where Main() is located and the SingletonTester class which can be used to make sure that only one instance of the game exists at any 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](#_heading=h.z337ya)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | + Easy to monitor  - Not designed for large scale distribution  - Not open source  - | + Fully open source  + Lots of specialized distros that are made specifically for web servers  - Not many programs are compatible with Linux, which can make sourcing software challenging  - Difficult to use if you aren’t used to it. | + Stable choice  + Built in server protection  + Easy for development and administration  - Not open source  - Proprietary software can be difficult to use if you aren’t knowledgeable about it. | + Allows easy development of both mobile and desktop websites.  = Some open source options  - Can’t handle large amounts of user traffic |
| **Client Side** | - Unenjoyable experience for the average user  - Expensive  - Not great for gaming  - Limited selection of web browsers  - Knowledge of Mac is almost required | + Free and fully open source  + Variety of specialized distros means there's a version of linux for almost every user.  + Very popular choice for game development.  - Difficult to on-board as very few computers come with Linux pre-installed  - Lack of linux ports for popular software | + The most common and versatile proprietary Operating system for desktops.  + Strong user security  + Very popular choice for gaming and development.  + Large variety of powerful browsers for the user to choose from. | + Easy to get hands on and use  - Extremely device specific  - Mobile ports of web browsers are significantly more limited than desktop browsers. |
| **Development Tools** | + One of the best platforms for Java, Python and web design  - Extremely difficult to use most other popular languages. | + Lots of popular open source options for development tools.  + Variety of specialized distros  - Very few industry standard development tools have linux ports | + Most popular industry standard tools are developed for Windows.  - Windows often slows down significantly depending on the size of the project. | + Apple devices are powerful enough to support development on them  + Android has a decent variety of development tools |

## 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 would recommend Windows for development, as it’s by far the most popular choice and has the most specialized and standardized software for the tasks at hand.
2. **Operating Systems Architectures**: As Windows is such a widely used operating system, its internal architecture is widely documented by sources both first and third party. It’s GUI is easy to use and navigate and is designed for ease of use.
3. **Storage Management**: Microsoft offers OneDrive as a possible storage option.
4. **Memory Management**: Windows offers functionality to control the amount of memory allocated to specific tasks, giving the development team full control over how much of the resources the game uses.
5. **Distributed Systems and Networks**: There are a huge variety of browsers available for Windows devices, from Microsoft's own Edge, to the popular Google Chrome for end users to access the game from.
6. **Security**: Microsoft has an extensive security team to aid in development as well as a versatile firewall that can be implemented and customized to fit within the server structure.