

Draw It or Lose It

# **CS 230 Project Software Design Document**

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 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

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

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/21/24 | Zachariah Spencer | Initial distribution. |
| 1.1 | 08/03/24 | Zachariah Spencer | Populated *Evaluation* section. |
| 1.2 | 8/15/24 | Zachariah Spencer | Populated *Recommendations* section. |

## [Executive Summary](#_heading=h.35nkun2)

The client, The Gaming Room, has a successful game called Draw It or Lose It that is exclusive to the Android platform. The Gaming Room wants to develop a web-based game that is based heavily on Draw It or Lose It but serves multiple platforms. The proposed solution Creative Technology Solutions (CTS) has presented is to develop a Java application because Java applications support play on a wide variety of platforms. This application will play very similar to the Draw It or Lose It application for Android, with slight alterations to make the user experience more universally enjoyable independent of what platform they choose to play from. In this game, multiple teams will compete to guess what is being drawn. The game will select a number of images from a large library of stock drawings to be rendered to all players as clues. A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
* The game must be playable from a wide range of device platforms.
* The game must have an easy-to-use user interface that is fast and responsive.

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

* The game must be developed to function as closely to the gameplay of the original Draw It or Lose It game as possible. This means the development process must take into account how the original game flowed, its win conditions, and its mechanics.
* The game must be developed in Java so it can utilize the Java Virtual Machine (JVM) to allow playability on a wide variety of device platforms. This means utilizing Java’s libraries and frameworks as well as common Java coding best practices.
* The game needs to be developed in a web-based environment. This means the game needs to be accessible over internet and able to be played by multiple people at once. This also means the game needs to be designed to handle network lag and basic network security.

## [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 UML class diagram below depicts the proposed class relations and domain model of the game. This model includes several classes: GameService, Game, Team, and Player, which all inherit from the base class Entity. The diagram also depicts the ProgramDriver class which contains our main method, indicating where the program should begin. There is also a SingletonTester class that exists for the development team to help us test new iterations of our code as we work on the project. More specifically, the SingletonTester class uses the ProgramDriver class to test the behavior of the Singleton design pattern implementation for the GameService class. The following sections will explain how each of the other interconnected classes work to fulfill the technical requirements of this project.

**Entity**

The Entity class is the base class for all entities in the game. It provides shared attributes such as id and name. Using the inheritance principle, Entity is able to allow other classes to inherit these basic attributes and behaviors. This inheritance allows us to avoid redundant code and to scale our game without accidentally creating inconsistencies from one entity to another.

**GameService**

The GameService class behaves like a central service to manage individual instances of games. It tracks a list of current games and provides methods for adding a game, retrieving a game via its id or name, and getting the number of games. It utilizes the Singleton pattern to ensure only one instance of itself is instantiated.

**Game**

The Game class represents an instance of a game within the app. It keeps track of all of the teams in a given game and also contains methods for adding teams and getting the game’s info in the form of a string.

**Team**

The Team class represents a team within an instance of a game as well as tracking the list of all the players on a given team. It contains methods that allow the application to add players to the team as well as get the team’s info in the form of a string.

**Player**

The Player class represents a single player in the game. It provides a method for getting the player’s info in the form of a string.

All of these classes work together to utilize the principle of composition by combining simpler objects to form more complex ones. The Game class is made up of several teams and each Team is made up of players. This design will allow for efficient game management.

**"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** | Powerful terminal that allows for easy access, configuration, and customization of servers.  Pros: It is modular and capable of being upgraded. There are many options to tailor fit for different web hosting requirements.  Cons:  It is not typically used for web hosting services. | Similar terminal functionality as mac but more cost effective and highly secure.  Pros:  Security vulnerabilities can be found before runtime issues which makes it one of the most favored platforms for web hosting.  Cons:  There are limited options for applications to meet the web hosting requirements. | Huge variety of software available, dominating market share. It is a closed platform.  Pros:  Shorter load times and user-friendly. A polished experience all around.  Cons:  Vulnerable to security exploits(viruses in particular), little in the way of technical support. | The server being in a static location would be ideal. Mobile devices are popular for their portability.  Pros:  Very accessible, almost everyone has a smartphone. Better compatibility with other platforms. Cost-effective for the development process.  Cons:  Highly specific development to various smart mobile devices. Vulnerable to security exploits. |
| **Client Side** | Some technically expertise specific to mac is required. Moderate time will be spent to work with Mac. Windows and Mac are comparable in cost. | Requires a lot of technical expertise to work with and may require more time as well but costs little to nothing. | Requires very little technical expertise to work with and is easy to achieve results quickly with. Costs are comparable to Mac. | Allows clients and developers to see updates promptly from anywhere. May be more challenging to work with than other devices. |
| **Development Tools** | When using programming languages on Mac, Swift is a popular choice. You can also integrate tools like Notepad++. Although Mac supports a wide range of languages, some commonly used ones include HTML, CSS, and JavaScript, often utilized for frontend development. Additionally, Macs can run general-purpose languages like Java, Python, PHP, and Ruby, along with their respective libraries. | Linux is compatible with tools like Visual Studio, Eclipse, and Notepad++, offering a user-friendly development environment. It also supports a wide range of languages and tools, including HTML, CSS, and JavaScript for frontend development, as well as general-purpose languages like Java, Python, PHP, and Ruby, along with their associated libraries. | While easier to use than Linux, it can run the same tools, such as Visual Studio and Eclipse, among others. With various tools available, Notepad++ stands out as a simple and user-friendly option. Supported languages include, but are not limited to, HTML, CSS, and JavaScript for frontend development, as well as general-purpose languages like Java, Python, PHP, and Ruby, along with their respective libraries. | You can develop a wide variety of apps using Android and Swift, both of which can be run on all three platforms. You can also work in Objective C for iOS applications. Supported languages include, but are not limited to, HTML, CSS, and JavaScript for frontend development, as well as general-purpose languages like Java, Python, PHP, and Ruby, along with their respective libraries. |

## Recommendations

1. **Operating Platform**: Considering the client's requirements, I recommend using Linux as the operating system for the server. Linux is a popular choice for server applications because of its stability, security, and performance. It offers strong support for web applications and is highly customizable, enabling developers to adapt the system to meet specific needs. Furthermore, as an open-source platform, Linux benefits from a large community of developers who are constantly improving and maintaining it. Given that Draw It or Lose It is a web-based application, Linux would be an ideal platform for hosting the game.
2. **Operating Systems Architectures**: For the proposed distributed system, I recommend a multi-tier architecture. This architecture divides the application into separate layers: presentation, application, and data tiers. This separation allows for more efficient scaling and enhances fault tolerance. The presentation tier manages the user interface, the application tier processes game logic, and the data tier handles storage.
3. **Storage Management**: Since the game requires rendering images from an extensive library of stock drawings, a high-performance storage solution is necessary to ensure smooth gameplay. I recommend using a solid-state drive (SSD) for the server's storage. SSDs offer faster read and write speeds compared to traditional hard drives, making them ideal for quickly loading images.
4. **Memory Management**: Linux, the recommended operating system, offers robust memory management capabilities. It utilizes a paging system that efficiently manages physical memory by swapping unused data to disk. This feature would benefit the Draw It or Lose It software by optimizing memory usage, and enhancing overall performance.
5. **Distributed Systems and Networks**: I recommend a RESTful API be implemented to facilitate communication between different platforms. RESTful APIs are commonly used in web applications and enable interaction over the internet via HTTP. Linux, the recommended operating system, offers strong support for RESTful APIs and simplifies their implementation.
6. **Security**: To safeguard user information across different platforms, I recommend implementing SSL/TLS encryption for all communications. This would ensure that data transmitted between the server and clients is secure and protected from unauthorized access. Security is a crucial element for any web-based application. Linux, the recommended operating system, offers robust support for security features like firewalls and access control. Additionally, Linux is recognized for its strong security and is generally less susceptible to breaches compared to other operating systems.