

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 | 11/12/23 | Dylan Bishop | First Draft |

Version Date Author Comments

1.0 11/26/23 Dylan Bishop Second Draft

**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)

CTS was asked by The Gaming Room to design a brand-new web-based version of their Android based game Draw It or Lose It. The game should allow teams with multiple players on each team. Each game, team, or player should only occur one time.

## Requirements

The requirements that the client has requested include that the game will have the ability to have one or more teams involved. Each team should have multiple players that will be assigned to it. The game and the team names must be unique to allow players to check whether a name is already used when choosing the team’s name and only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

Draw It or Lose It already exists on Android smartphones. CTS is tasked with extending Draw It or Lose It to the web and the tech needs to be compatible with web deployment. Java has been selected for completing this project because it is the primary Android language so this should make it easier on the new deployment effort.

## [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 Diagram for the proposed project design is shown below.

**"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.**

The web-based application will have a main driver class that will be used to start the creation of the games, teams, and players. The class that will create the games is the GameService class and it will follow a singleton design pattern so only one single GameService class will exist at a time.

GameService will block its creation of any instances by being a private constructor. The only way that the GameService can be initiated is through getInstance(). This method will check if GameService has already been started and will only start if a game is not already in memory.

Once the GameService class is already running, the driver class will be able to call the method addGame(). This method will use an iterator pattern to stop games with the same name from being created. The new game will then be added to the list of named games.

As soon as a game is created a team will be added to the game with the method addTeam() which also uses an iterator pattern to stop teams with the same name from being added to the game. The new team will then be added to the list of teams.

When the team is created, players will then be added to the team with the method addPlayer() which will use the iterator pattern in order to prevent players with the same name from being added to the team. The new player with then be added to the list players.

The Game, Team, and Player classes are all subclasses of Entity, which has two protected attributes, id, and name. Also protected are the default constructors so null objects will be blocked at the creation of a game and the overloaded constructors will be used.

Polymorphism and inheritance, extending the Entity class and overloading constructors, are used in the UML diagram which are two of the many Object-oriented program techniques that are used throughout the program. Encapsulation and Abstraction are also used in the program because they are used to add teams. A team may not be created directly since the constructor cannot be accessed because it is blocked, but a user is still able to access it with the addTeam() method without the user having any knowledge of how exactly the team was added.

## [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** | Benefits of Mac include the ability to easily integrate into IOS apps and Mac apps, but it will not scale well and will require Apple to develop the server. | Linux would be the best option when wanting to host a web-based server. They remove a lot of unnecessary bloating. The downside would be the expertise of the development team. | The advantage of Windows is that it is a common way of launching servers and has a lot of options to do it. The downside is the setup and cost. | Mobile Devices could be efficient because you don’t need much hardware power for the app but if the battery died on the mobile device, it could possibly shut down the entire server. Also, mobile storage isn’t as high as it would be on a computer’s operating system. |
| **Client Side** | It is not great for developing other operating systems but can work well for web-based apps. Setting up a java environment on Mac can be more of a hassle because it is expensive to scale the software and Mac hardware is expensive as well. | It will take more time and more experience to develop Linux but because of the low cost it would be the best option. | The time would be much lower for development and the cost would be average. A lot of developers are much more comfortable on Windows. | To develop an app with a mobile device you would have to be an expert. The time wouldn’t take very long, and the cost would be low. |
| **Development Tools** | Swift and CodeX are going to be used to develop web-based applications. You can download Java onto the machine, but it will require a much more advanced setup because it is not native to Mac. | There are plenty of options for Linux development tools because of it being an open-source operating system. It will take more knowledge to set it up, but it will give a lot of options. | Using Java with VS would be the best option for Draw it or Lose It. | Whether you use iOS or Android’s operating system there will be a different IDE for both and different apps to aid in the development which would not be ideal because these are usually meant to develop mobile apps and not web based apps. |

## 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 personally would recommend developing a web-based application through Windows.
2. **Operating Systems Architectures**: Windows has a modular structure that is made of modules, hardware abstraction, kernel, executive services, an environment subsystem, and the integral subsystem. There are modes that can be either kernel or user in this system and kernel modes will allow direct access to the CPU and memory. The user mode will have many things that are locked because of security reasons which help prevent crashes that can be fatal to the system.
3. **Storage Management**: For storage management I would recommend an SQL server.
4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.> Windows uses memory management techniques to block a section of the memory for the server and use physical and virtual memory to keep steady and limit the ram usage. This will help because it would prevent the application from crashing by allocating only what is needed at the time and it would limit other program’s memory usage if the application needed more at any time.
5. **Distributed Systems and Networks**: A Spring boot application would be best because any platform with many browsers can access the application. The application would use HTTP protocols because they are extremely customizable and are good with access for the user. A system like Amazon Workspaces would also work because of the reliable network and the systems that are built in so outages would not harm the game.
6. **Security**: With security, Windows has you covered. There are many different built-in protocols within the operating system from firewalls to virus protection software. If the program was developed using a RESTful Spring boot app using the Oauth2 security protocols, that would help prevent a lot of security risks for private client information or any business information.