

Draw It or List 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 | 05/20/24 | James Lightner | Made OS and platform recommendations to client |

**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 client, The Gaming Room, wants to develop a web-based game called Draw It or Lose It. This is a multi-team and multi-player game that will render images from a library of drawings for players to guess from. Like its in-person counterpart, each team will have a time limit in their guess and if the team cannot guess correctly the other team will have time to respond and steal points. For this to be accomplished, the web-based game will need to work across multiple platforms.

Our new client The Gaming Room, wants to develop a web-based game that can be available on multiple

platforms. The current game they want to base on is Draw It or Lose It. This game only works on Android

currently and the company wants the game to be able to run on other Operating Systems such as

MacOS, Linux, Windows, etc. The game consists of teams competing to guess what images are being

drawn. There are time limits for guessing as well for showing the drawing mark time. The game consists

of four rounds of play lasting one minute each.

Our new client The Gaming Room, wants to develop a web-based game that can be available on multiple

platforms. The current game they want to base on is Draw It or Lose It. This game only works on Android

currently and the company wants the game to be able to run on other Operating Systems such as

MacOS, Linux, Windows, etc. The game consists of teams competing to guess what images are being

drawn. There are time limits for guessing as well for showing the drawing mark time. The game consists

of four rounds of play lasting one minute each.

Our new client The Gaming Room, wants to develop a web-based game that can be available on multiple

platforms. The current game they want to base on is Draw It or Lose It. This game only works on Android

currently and the company wants the game to be able to run on other Operating Systems such as

MacOS, Linux, Windows, etc. The game consists of teams competing to guess what images are being

drawn. There are time limits for guessing as well for showing the drawing mark time. The game consists

of four rounds of play lasting one minute each.

## Requirements

The requirements for the Draw It or Lose It are:

* Web-based game application
* Multi-team game
* Multiple players per team

## [Design Constraints](#_2et92p0)

Entity creates a relationship between Game, Team, and Player class. This means they all inherit or get

information from Entity. With UML we can show this with inheritance. Making Entity a superclass. When

we look at their relationship, we see Team and Player is a “has a” type. While Game has a Team and

GameService has Games. When we use UML, we call it aggregation (HAS-A). When a user “has a” I mean

it's an instance of one class and has a reference to an instance to another class. When we look at this

diagram, we see GameService has a reference of Games, Games a reference of Teams, and Team a

reference of Players

Entity creates a relationship between Game, Team, and Player class. This means they all inherit or get

information from Entity. With UML we can show this with inheritance. Making Entity a superclass. When

we look at their relationship, we see Team and Player is a “has a” type. While Game has a Team and

GameService has Games. When we use UML, we call it aggregation (HAS-A). When a user “has a” I mean

it's an instance of one class and has a reference to an instance to another class. When we look at this

diagram, we see GameService has a reference of Games, Games a reference of Teams, and Team a

reference of Players

The Constraints for the game are:

* Must utilize a platform agnostic development environment or bring on multiple teams.
* Only one instance of the game can be in the memory at any time.
* The Game and team names must be unique to allow users to check if a name is in use or not.

## [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)

On the UML diagram we can see the entity class creates a relationship between the game, team, and player classes. The arrow in the diagram indicates that all the classes will inherit from an attribute from a super class. We can see the classes, variables and methods that will be used during development. The programDriver Class is pointing to the singletonTester. This shows us that the programDriver will use the SingletonTester to test the code. This is developed so we can test the restriction of having one instance where the game can exist in memory. The Gameservice class will hold all the methods that make up the framework of the game. One of the requirements of the program is to have unique game, teams, and players. This why each uniqueness has its own class. The numbers between the lines tell us the number of associations within each class. For example, the GameService can have zero or more associated with the Game class and the same goes with the other classes in the diagram. **"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 has accessibility and server configurability. Easy to use graphical user interface. Flexible terminal commands. | Cost friendly. Can be difficult to navigate the platform. Command shell for simple server configuration and accessibility | Server side is expensive. User friendly GUI. Has a command prompt. | Specifications are better in other devices. Mobile device specifications vary from user to user. |
| **Client Side** | Expensive for users. Moderate time and expertise required. Accurate skills and needed to navigate OS. | There is a lot of expertise and time required. Linux data is required to use the operating system. Maximum cost for Linux users | More expensive than Linux systems. Easy to learn and understand how to support a Windows setup. Minimum expertise needed | Provide flexibility to clients or even developers to see updates at any place. Slightly more difficult to implement than other devices. |
| **Development Tools** | Languages like HTML, CSS and JavaScript. Libraries to support frontend development. Other development tools include PyCharm, GitHub, Visual Studios etc. | Languages like HTML, CSS and JavaScript. Libraries to support frontend development. Other. Linux systems include JavaScript, Ruby, PHP and Python. | Languages like HTML, CSS & JavaScript. Libraries to support frontend. Developer tools include Eclipse, command prompt, PyCharm, Eclipse, etc. | Languages like HTML, CSS and JavaScript. Libraries to support frontend development. IDE’s for programming languages consist of HTML, php, 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**: Based on our evaluations and we recommend the Linux operating system. Linux is open source reducing costs, also it supports various development tools, and it is flexible and secure.
2. **Operating Systems Architectures**: Operating systems act as an interface between a user of a computer and the computer resources. The architectures are hardware, kernel, and shell. The hardware architecture consists of the memory, CPU, I/O, an any other physical devices. The Kernel is a bridge between applications and the actual data processing done at the hardware level. Finally, the Shell is a piece of software that provides an interface for users to an operating system.
3. **Storage Management**: the game Draw it or Lose it is will be a web-based application. We recommend a Cloud storage unit like google cloud. With the cloud we can minimize the storage space needed to keep the game running. As the game increases in popularity, we can upgrade and expand the storage with relative ease.
4. **Memory Management**: Linux is a versatile and easy to customize operating system. For this game we will use Java as the main language for the backend development. Java does memory management automatically. Java uses a garbage collector for its automatic memory management system.
5. **Distributed Systems and Networks**: We are recommending the Google Chrome browser because it is available on all existing platforms. The game is web-based, and Google Chrome is one of the most widely used browsers. It only makes sense to use this browser to launch the game.
6. **Security**: Google cloud services offers certain security features already built into the service. We will need to be aware of what is covered under it and what we are required to provide as the client. Even though there are certain protections in place, the team needs to remain vigilant that what we are developing has security concepts baked into it from day one.