

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 | 03/18/2025 | Chris Williford | Changes made to the cover page, executive summary, design constraints, domain model, and recommendation. |

## [Executive Summary](#_sbfa50wo7nsh)

The purpose of the Gaming Room Project is to develop a web-based game that will serve multiple platforms based on the current game ‘Draw It or Lose It’. This game is currently only available on an android app. The game is played by multiple teams of varying sizes competing four rounds that last one minute each. Each round, a picture is pulled from a library of images, and one team guesses until the time runs out and 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 teams 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 app must run on multiple platforms, ensuring that it runs seamlessly on all devices.

## [Design Constraints](#_2et92p0)

* The code needs to ensure that there is only one instance of the game being able to be active at any point in time. This will ensure that the game functions as intended.
* All names must be unique and allow users to check whether a name is already in use when choosing a name to avoid overlapping.
* The game needs to be designed for web-based platforms, needs to address constraints involving network communication, security, and compatibility with all the required browsers, platforms, and devices.

## [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 Entity class creates a relationship between the GameService, Game, Team, and Player classes. They all inherit information from Entity, and the UML shows this inheritance. Each of these classes depend on one another, but the ProgramDriver is the main class and is linked to the SingletonTester class to allow it to perform inherited activities from the main class.

**"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** | **Characteristics:** Growing in popularity due to web hosting capabilities.  **Advantages:**  It has more intuitive software and design.  **Weaknesses:**  It is less commonly used and comes at a price. | **Characteristics:**  More cost-efficient and is most secure.  **Advantages:**  Most preferred of all listed due to security of web hosting.  **Weaknesses:**  It is less commonly used, and hardware drivers are not as readily available. | **Characteristics:**  More expensive. Most widely used.  **Advantages:**  Cheaper hardware and has a higher level of experience among users.  **Weaknesses:**  More vulnerable to viruses and could cost more in the long run. | **Characteristics:**  Portable, most popular, and accessible.  **Advantages:**  It Can reach a larger portion of the population all over the world. No additional cost associated.  **Weaknesses:**  Physical security on mobile devices is lacking. |
| **Client Side** | Fairly expensive for users and moderate skills needed to navigate the OS. | Not as popular as the other options and it is expensive. | It is most commonly used. It is more expensive than Linux. Easy to learn. Less training time. | Flexible for clients and developers. High level of technical support. Hardware is upgraded regularly by users. |
| **Development Tools** | Eclipse, JavaScript, PyCharm, Visual Studio, GitHub, CSS, HTML, Xcode. | HTML, CSS, JavaScript, Ruby, PHP, Python, Eclipse, GitHub, Atom, NetBeans. | Java, Python, C++, NetBeans, HTML, JavaScript, Eclipse, Command Prompt, CSS. | HTML, CSS, JavaScript, C++. |

## 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**: Windows would have to be the recommended OS for the web-based software with it being more compatible than Mac and is cheaper in terms of hardware.
2. **Operating Systems Architectures**: Windows is a graphical user interface-based OS. It will be easier to navigate without needing a lot of experience with using command prompt.
3. **Storage Management**: Windows is more compatible with a database management system for its storage. It is the most efficient overall with its adaptability.
4. **Memory Management**: Windows automatically takes care of memory management. However, users can take control of it if they wish to do so.
5. **Distributed Systems and Networks**: The best system to run this type of network would be LAN. LAN is the most reliable and allows protection if there is an outage or in the event of a crash. LAN offers the maximum connectivity through a HUB that can magnify a signal.
6. **Security**: Windows comes with a built-in security called the Windows Firewall. It is a free antivirus software for users to use. It offers basic internet safety that will catch phishing bugs on the internet. However, there are more advanced antivirus software that are out there that do come at a price, but they can offer more protection for the user.