

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 | 01/21/23 | Jorge Torres | Changes to the cover page, added executive summary, design constraints, domain model |
| 2.0 | 2/4/23 | Jorge Torres | Completed evaluation table |
| 3.0 | 2/13/23 | Jorge Torres | Completed 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](#_sbfa50wo7nsh)

The Client The Gaming Room wants to develop a web based game based on their current game Draw It or Lose It that is currently on the android platform. As the Gaming Room staff does not have the Necessary resources to complete the task they have been asked to reach out to CTS for the help in streamlining this project and see it to fruition.

This game will render images from a stock library of drawings and the teams must guess the answer to what it is. If the team fails, the opposing team needs to say the correct answer within a 15 second window.

## Requirements

1. Multi player platform with multiple teams’ involvement
2. Unique player Id for each team member.
3. Unique Game and Team ID’s so users can check if the names they pick are being used.
4. One instance of game can exist in the memory at any given time.

## [Design Constraints](#_2et92p0)

1. Multiple codes to account for future expansion apart from the customers requirements.
2. Cloud service and internet access for application, speed threshold? local server or cloud-based server?
3. Player communication for team player involvement.
4. No Budget Designated, need budget to understand resource allocation and availability.
5. No specific timeframe given.
6. Specific hardware needs as rendering is a heavy load process and no simple hardware can suffice.

## [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 bridge of communication between Game, Team, and the Player class. They Inherit from Entity. Team and Player are good examples of Aggregation. It’s an instance of a class that has reference to an instance of another class. If we look at this project, Game Service has a reference to Games.

**"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** | With Mac, the positives of using it are:  It deploys a simple server process.  It has Very well documented Information for its Software  Proven Secure OS  Support plans and technical support available.  The negatives are:  Its very hardware constrained.  No OS cross use  Exclusivity for Updates  High-Cost option | With Linux, the positives of using it are:  Open-Source software available and not managed by one entity.  Proven Stability  Lowest hardware requirements  The Negatives:  Not a lot of software available  Not used as commonly as other OS platforms | With Windows, the Positives of using it are:  It deploys a simple server process.  Big range of hardware options  Quicker update because of larger userbase  The negatives:  Less security  Closed source.  Updates are only provided through Microsoft. | With Mobile devices, the Positives are:  Code cannot be seen by the user  Stores only the data needed.  The Negatives:  No integration of cloud services on the server side. |
| **Client Side** | The Positives:  Good amount of supported web browsers with development options  Average time consumed to develop and deploy.  Negatives:  It requires use of a MacOS Product. | The positives:  Good amount of supported web browsers with development options  Fast, less time consumed to develop and deploy.  Open-Source Software  Negatives:  Not used as commonly | The Positives:  Good amount of supported web browsers with development options  Easy cross platform testing  Fast, less time consumed to develop and deploy.  Negatives:  Hard to test on MacOS Browser | The positives:  Experience with Android app development  The negatives:  Takes longer to develop and deploy.  Hard to test in other browsers and environments, |
| **Development Tools** | Can easily run Windows and Linux through local virtual machines.  Developer Application needs to be reviewed and approved by Apple.  Eclipse for Java | It can be deployed at any moment.  Open-source community  No license cost.  Eclipse for Java | Can Run Linux through Virtual Machine  No license cost.  It has Visual Studio Code  Eclipse for Java | 99$/year for Apple developer program  For IOS  Swift UI |

## 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 the evaluation done, I feel comfortable recommending the Windows Platform for this project because it can integrate with current state and help build further states for mobile Android devices. It’s also the largest userbase and popular for software development.

1. **Operating Systems Architectures**:

With a friendly user interface and easy customization of hardware, built-in security checks and great memory data, Windows is the clear choice.

1. **Storage Management**:

Given that I selected to use Cloud Storage, I would recommend using Azure, as this, developed by Microsoft as Windows, will seamlessly integrate with the Operating platform and provide the most efficient cloud based storage option. It is also secured and encrypted for the convenience of the user.

1. **Memory Management**:

Windows features the use or access of virtual memory. This allows the user to reserve some of the space unused in the storage disk to store some data, in the process of not overloading the RAM memory available. Likewise, it can move around in case storage is needed. This alone should be enough for the Application.

1. **Distributed Systems and Networks**:

We will be using a client-server system, providing the best experience for each client with single client server, which will adapt to each client’s system. For this we need a good network too that can handle multiple clients.

1. **Security**:

Windows Has a built-in security System called Defender. This will help with the process, but I would recommend encryption of all data that is moving back and forth. And security client to server specific internal files with Access control.