

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <01/25/22> | Patrick Chu | First Draft, executive summary, design constraints, domain model |
| 2.0 | 02/06/22 | Patrick Chu | Evaluations |
| 3.0 | 02/20/22 | Patrick Chu | Recommendations |

**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 Gaming Room seeks to develop a web application capable of running on a multitude of platforms. The application will be called “Draw it or Lose it”. The app will simulate a game called Win, Lose or Draw, in which teams compete against each other to guess what is being drawn. The app will render the images from a library of drawings, and each game will consist of four rounds of one minute. The teams will be guess at a drawing being rendered during a thirty second time period, with an opportunity for remaining teams to guess at the drawing in a 15 second period if the original team does not guess the drawing.

## [Design Constraints](#_2et92p0)

-Needs at least 1 team to play, multiple teams may play

-Each team may consist of multiple players

-Game, team, and player names must be unique, and users must be allowed to check if a name is in use

-Only a single instance of a game can exist at one time

-Compatible with multiple platforms

Because the client wishes for the application to be able to run on multiple platforms, the game must be developed in such a way that players on different platforms are all able to join teams and games together. It is likely that an initial application and API will need to be written first, then ported over to different platforms that communicate with the API in order to take in account the various operating systems that will be playing the game.

## [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 is the Superclass of Game, Team, and Player classes, and they all inherit from Entity. Each of the classes will have attributes such as name and id. Game has a “has a” relationship with team as multiple teams can be referenced a single game, and team has a “has a” relationship with player as a team can have a single or multiple players referenced by it. GameService also references Game and therefore also has a “has a” relationship with Game.

**"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** | Strong GUI, good ease of use. Very accessible and easy to configure | Difficult to navigate and configure without expertise, but shell navigation is a good tool for configuration and accessibility, cheap | Good GUI, lots of available server software, well established command prompt configurations, but expensive | Mobile devices are highly variable in hardware, so creating server software is difficult as there is more to account for. Also moving physical location of server may present communication errors between server and client |
| **Client Side** | Average expertise and time needed, relatively high cost. Mac is relatively easy to navigate and has a lot of developers already | Linux requires a lot of time and expertise to develop on, fewer established tools for developing directly on linux. Very cheap however | Windows is easy to navigate, requiring less time and expertise. There are also many windows developers and tools to work on this platform. Average cost | High amount of time and expertise to develop as there are many different mobile devices and versions. Extra time would also incur higher costs in both development and support/testing |
| **Development Tools** | Programming languages:  Swift, python, javascript, java  Tools:  Visual studio, eclipse, atom, notepad | Programming languages:  C, C++, C#, python, javascript, java  Tools:  Visual studio, eclipse, atom, notepad | Programming languages:  C, C++, C#, python, javascript, java  Tools:  Visual studio, eclipse, atom, notepad | Programming languages:  C#, python, javascript, java  Tools:  Visual studio, eclipse, atom, notepad |

## 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 would recommend Windows server operating system as it well suited for expansion and can has many features to ensure success. The ability to run in protected and supervisor mode allows for the server to reliably serve requests and ensure strong performance of computers on the network with support from other features such as acting as a web server, file server, app server, database server, and more. It is also designed specifically to run on server hardware.
2. **Operating Systems Architectures**: Windows server allows for very strong storage and file management and allows for the user to control the computer memory for a chosen purpose, such as allocating memory to a specific program and freeing memory when it is no longer needed. It also allows for multi-processor scheduling which can optimize performance and allocate additional resources where they might be needed on a busy game night. The scheduling would allow parts of the processor performance to be dedicated to a specific purpose ahead of time. Powershell also allows for easy access to the machines connected to the server for regular maintenance.
3. **Storage Management**: Windows server allows for schedule backups and data relocations to attached storage to free storage on the server hardware. This will ensure that the server does not run into storage issues and also have copies of data on external drives.
4. **Memory Management**: Windows server OS comes with the ability to utilize random access memory as well as use of a pagefile to expand the available memory by writing data to a virtual access space on the hard drive rather than the available RAM.
5. **Distributed Systems and Networks**: In order to effectively have a multi-platform game, we will need to use a client-server distributed system so that a client can communicate with a geographically close server for best performance, or with the ability to choose a server in order to play with others that would otherwise not be on the same server or serve as a backup during planned downtimes for maintenance.
6. **Security**: Windows server by default comes with security layers to protect data. It also has the shielded virtual machine feature, which allows access to the virtual machine to be restricted via the network administrator. The default windows defender application control also restricts control of which applications can run on the host machine, while also detecting and blocking known malware.