

<Draw it or lose it>

# **CS 230 Project Software Design Template**

Version 1.1

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.gjdgxs)1

[**Table of Contents**](#_heading=h.30j0zll)2

[**Document Revision History**](#_heading=h.3znysh7)2

[**Executive Summary**](#_heading=h.2et92p0)3

[**Design Constraints**](#_heading=h.tyjcwt)3

[**System Architecture View**](#_heading=h.3dy6vkm)3

[**Domain Model**](#_heading=h.1t3h5sf)3

[**Evaluation**](#_heading=h.2s8eyo1)3

[**Recommendations**](#_heading=h.3rdcrjn)5

## [Document Revision History](#_heading=h.3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.1 | <09/12/2022> | Myles Pruitt | Adding information for the “The Gaming Room” based on requirements. |

**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](#_heading=h.2et92p0)

Our client The Gaming Room wants to develop a web-based game that serves multiple platforms. First, we need to ensure performance and speed are acceptable. Rendering images takes a lot of resources especially if it’s high res, it’s important to set a baseline for image resolution and it’s consistent throughout the application. Second important to know our tech stack, especially on the database end. How will we store the images, and what database is easy to use for all types of web browsers? In my opinion, using SQL is probably the best way, No-SQL database you are usually limited by paywalls and sometimes the lack of customizability. The third is the overall visual design. We need to know aesthetically, how visually appealing the game is to the user and whether or not, if we are limited by the coding language we use. For instance, REACT.JS is good at making a lot of UI elements interactive and visually pleasing compared to regular HTML and CSS.

## [Design Constraints](#_heading=h.tyjcwt)

* Performance
* Scalability
* UI and UX design
* Web-based game
* Drawings render at a steady rate and complete at a 30-second mark
* The game consists of four rounds of play, last 1 minute each
* Application will render images from a large library of stock drawings
* Game and team names 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 a given time

## [System Architecture View](#_heading=h.3dy6vkm)

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](#_heading=h.1t3h5sf)

The UML diagram represents the classes associated with the “The Gaming Room” web application “Draw it or Lose it”. The object-oriented principles that are represented in the UML diagram are accessors, mutators, and methods that inherit characteristics from the entity class. Classes also share relations, for instance, Game service can create many 0 to many games, Games can create many teams, teams create many players, and they all inherit from the entity. The program driver class has access to all necessary classes and can execute them, there is also the SingletonTester class meant for testing. How the UML is designed only allows for one game session at a time with multiple players and teams.

**"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](#_heading=h.2s8eyo1)

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 paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Purchasing apple servers are relatively cheap, but service is discontinued. There might not be much support for this product and there are not a lot of people who use this product. | Linux is typically free and there are a lot of open source resources you can use. Downside is that many people are not familiar with Linux and Linux is not user-friendly. | Windows is the most commonly used OS, but costly on the server side of things. You will most likely be paying in the thousands to get everything going. | Mobile application servers are cheap compared to every mainstream OS. The only issue is you are limited by hardware, so paying close attention to performance is a necessity. |
| **Client Side** | Client side of Macs is cheap. Mac is known for its user-friendly environment as well. The only issue is that not a lot of people are familiar with MACS. | Linux is open source, there is a lot of flexibility and a lot more freedom in its environment compared to the other OS. The cost would be cheap to free, time could be an issue since Linux is not a user-friendly environment. Depending on the distro performance could also be lackluster. | Windows would not take long to learn. Most people are familiar with windows and it is typically their first OS. Costs could be high, and terms of time could be fast since there would be essentially a lot of support. | The cost would be relatively low, a good amount of people use Mobile devices more than regular computers. People would be easily able to learn about the environment and to some degree, everyone has used a mobile phone at one point in time. |
| **Development Tools** | Microsoft Visual Studio, Atom, Webstorm, Netbeans and eclipse | Atom, Netbeans, web storm, Eclipse, IntelliJ, and IDLE | Windows has a variety of IDEs and tools since its a big part of the tech market. Visual Studio, Atom, Sublime, IntelliJ, NetBeans, and Webstorm. | Android Studio, Xcode, eclipse, Microsoft visual studio |

## 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**: Creating for the Windows platform before expanding to other platforms will be the best start for the “Draw it or lose it” web application. Linux, Mac, and Mobile devices acknowledge Windows as being the most used PC OS globally and would most likely provide support for migration to their platform.
2. **Operating Systems Architectures**: Windows OS has an easy navigation system, which other OS try to emulate since it's the most used operating system. Windows also has a lot of cross-platform utility since a lot of different IDEs support the system compared to MAC, LINUX, and Mobile.
3. **Storage Management**: Windows on a business level is used mostly for storage management because of its ease of use and support. Owning a server is expensive, but well worth it compared to alternatives. Scalability is also a big bonus when acquiring a windows server which helps with maintainability over long periods of time.
4. **Memory Management**: Windows has incredible memory management from physical to virtual memory. Understanding and modifying memory are very easy to understand compared to other operating systems which might require terminal use. Some programs can facilitate memory management processes in Windows as well giving people options.
5. **Distributed Systems and Networks**: Distributed System and networks has always been a challenge for any operating system, but in terms of issues being addressed Windows has done a fairly good job. Windows OS provides a lot of functionality when coming to network stability, lagging, and overloading. There are even dedicated windows network technicians which can support you if you are stuck which some other operating systems can lack in. Windows documentation also makes users aware of best practices to avoid certain issues or mitigate downtimes by suggesting things such as load balancing. There are also a lot of third-party tools that can assist with networking issues that are typically user-friendly.
6. **Security**: Windows also has good security. Windows defender is constantly being updated to effectively counter threats to computer systems and has a dedicated research team which other virus companies pool from. Windows ensures that its users are protected and emphasizes good practices to be used by their users, developers, and hobbyist. Encryption is also part of windows' security utility to add an extra layer of security, to avoid malicious users from ciphering data.