

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 | 8/25/2024 | David Balogun | Refining design constraints, executive summary, and domain model. |

**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 website should be hosted on windows as it easier and more efficient to run a website on windows. Mac/iOS would now have as many users but should retain most of the features of creating the website on windows. Linux might have a small problem, but overall should retain the features as Windows and Mac.

## Requirements

This website should be able to run on all operating systems.

## [Design Constraints](#_2et92p0)

A website is limited to only being able to run an html interface making it hard to customize widget

without needing help from other applications. However making it a application allows for a lot

customization with widget and other things, such as databases, other applications, etc.

## [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)

When the application starts it starts the game service. The game service houses multiple games. Then, game houses teams and an entity. The team then houses player as and entity.

**"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 will be able to use most of the resources that the others can use. However it will be able to use extra features as it is good for development. | Linux will have a little bit of trouble when it comes to some particular resources, but it can use specialized applications like Jupyter notebook application. | Windows will be able to use most resources as long as some particularly useful resources such as Windows PowerShell. | Mobile development will be the most difficult as you won't be able to use most resources that you are able to use with windows Linux or Mac. However, you will be able to use some resources like Android Giraffe. |
| **Client Side** | It will take a decent amount of time for a Mac to be able to complete their application. | It will take Linux a little more time because they won't have some of the development resources that Mac and Windows have. | Windows will take a little less time due to windows PowerShell which is a very good tool for creating websites and applications. They would also have Express which is a good tool for creating websites as well.. | Despite the fact that Andrew would be the most difficult it wouldn't take that much time as Android giraffe will be able to cut down the amount of time. |
| **Development Tools** | Mac would have most of the resources as every other application, so it will be easy to use most ID's and programming languages. | Due to the abilities of jupyter notebook it should be fairly compatible with most programming languages. However, they will be stuck to Jupyter IDE. | Windows will be able to use most IDEs and programming languages however IDE such as Visual Studio will come in most handy. | Mobile will be stuck to a very select amount of IDEs and programming languages. |

## 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**: The Operating Platform should be either Mac or Windows.
2. **Operating Systems Architectures**: For Windows, It would be a Express framework using windows PowerShell.
3. **Storage Management**: A database like SQL or a NoSQL System like mongo DB would work.
4. **Memory Management**: The System will consistently run a check for duplicate data and delete it.
5. **Distributed Systems and Networks**: This would require something like a SQL Server. And code for adaptation.
6. **Security**: This system would use https and would require encoding.