

<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 | 01/22/22 | Paul Schwartz | Creating Entity class, creating iterators, completing various code |

**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.

## [Design Constraints](#_2et92p0)

Major design constraints are having the game run on multiple platforms, having the web-based game being playable on all of them while allowing the players to play with each other on other platforms. This has been mitigated with it being a web-based game but there are multiple web-browsers to consider and lots of testing will be needed to make sure everything plays nice with each other. Having only one instance of the game existing in memory at a time, this will require additional coding and testing to make sure this happens, this would require use of the database and how the memory structure works.

## [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 Game, Team and Player class all have inheritances from the entity class, this helps with not having the same code being written in each of those classes and just having one instance in the Entity class.

The ProgramDriver and SingletonTester classes each use each other or have an Association with each other. This allows the SingletonTester class to test things by taking them from the ProgramDriver.

The Game, Team, Player and GameService classes are all associated with each other from the 0…\*, this means they use 0 or more things from each other. They make use of each other for information.

**"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** | * Requires expertise and training * Familiar terminal commands like Linux * Can be expensive | * Easy to learn Terminal command * Has the most researchable resources. * Cost Effective | * Most Popular OS * Built in cloud functions * Lots of tools and applications | * Higher latency * Congested mobile networks * Can be used on the move * Not a lot of tools and resources |
| **Client Side** | * Highest learning curve * Cost on Par with Windows * Some users will need trained if not familiar | * Popular * Open Sourced * May require training to use * Lots of tools and applications | * Comparable cost * Easy to use * Can be a resource hog * Most Popular OS | * Used on many Different Devices * Connect anywhere with mobile network * No connection to network issues |
| **Development Tools** | * Notepad++ * Github * Git * Visual Studio, Eclipse * FlexiHub * Tower * Netbeans. | * Notepad+ * Github * Git * Visual Studio, Eclipse * FlexiHub * Tower | * Notepad+ * Github * Git * Visual Studio, Eclipse * FlexiHub * Tower | * Notepad++ * Github * Git * Visual Studio, Eclipse * FlexiHub * Tower |

## 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 recommend starting with the Windows Operating system. It has many tools, resources, and a lower initial cost than most of the others. Windows will support majority of the development tools that would be used as well.
2. **Operating Systems Architectures**: Windows has many applications and services that can be used. Those will allow access to resources of the system and allow creations of GUI’s. Can be used as a server or a client as well.
3. **Storage Management**: Windows supports many different storage applications. If not needing speed can use a regular HD or even needing fast read and write speeds a m.2 would be beneficial. Also, will have cloud integration for better backups of data.
4. **Memory Management**: With the game being web-based, it will not require fast amounts of system resources. Especially when this game will be used on mobile applications memory usage should be keep small. Windows handles memory usage from web browser and will allow efficient usage of what needed to the game. It will limit memory usage when it comes close to being maximized to not have system crashed but game performance may slow.
5. **Distributed Systems and Networks**: With the game being web-based this allows focus on building the game and letting their operating systems deal with communication between devices and networks. Mobile and slower connection may have lag and latency issues. This is the advantage of cloud-based server tech, it can ramp up and down based on usage and the number of players. As long as the software is designed to communicate and exchange data with the server players from different platforms can play with each other.
6. **Security**: First is having a secure server and network, and not exchanging personal information with clients just keep them server side. The less vital information sent to and from the OS and servers the better chance of keeping it secure. Also, maybe purging information after each game, as windows tends to create their own security issues with updates. Encrypting the network connections and information is a must.