p

Draw It or Lose It>

# **CS 230 Project Software Design**

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 | 1/20/2023 | Drew Heard | Edited Title Page, Executive Summary, Requirements, and Design Constraints. |
| 1.0 | 2/5/2023 | Drew Heard | Completed Development requirements |
| 1.0 | 2/18/2023 | Drew Heard | Completed 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 desires to release a web-based game on multiple platforms. The client wishes CTS to design and develop the initial environment of the game state. This new design will be accessible on multiple platforms and meet the client’s requirements for the game.

## Requirements

Business Requirements

-Reach a wider audience.

-Deliver a game based on the current version.

Technical Requirements

-The game will be available on multiple platforms.

-Design will be based on client defined rules.

-A game will have the ability to have one or more teams involved.

-Each team will have multiple players assigned to it.

-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 any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

1. Will the wide range of hardware from all platforms be problematic? Are there constraints from certain devices? Hardware from all platforms will have to be considered during design.
2. Will Java be the most efficient to develop such a software? Any platform that is not Java friendly will need to be identified.
3. With so many unique objects, will memory be an issue? The best programming practices will need to be exercised to ensure no memory leaks or garbage pile up.

## [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 Program Driver class uses the Singleton Tester class. The Game, Team, and Player classes all inherit from the Entity class. Team also associates with both Player and Game. Game can also associate with Game Service.

The classes Game, Game Service, Team, and Entity all use abstraction and encapsulation with their use of private variables, getters, and setters. The way the Entity class is used as a base object for all the other objects is a good example of polymorphism. The classes also use inheritance as mentioned earlier.

**"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 as a server is expensive and harder to use but secure. Mac servers might also limit your development to also using Mac. | Linux is usually free, somewhat easy to use, naturally adept at web based hosting, but probably the least secure of the big three. | Windows is secure and the easiest to implement, but also expensive. | Hardware isn’t really equipped to always run a server. Also, the least secure of all choices at least regarding Android. |
| **Client Side** | High performance and straight forward to use, but again, it is expensive and limits your development. Make sure development passes Apple’s rigorous store implementation process. | Time consuming and slightly difficult to use, but it’s cheap and adaptive. Mid-level expertise required. Keep security and application run efficiency in mind during development. | Windows would probably be the fastest development but would require some expertise to be done securely. Also, it can be expensive. Keep security and the wide range of possible client devices in mind. | The expertise required would be niche to mobile development. Cheap and probably fast for your experts to use. Keep device variety in mind and Android’s vulnerabilities(jail breaking). |
| **Development Tools** | A Mac device using Xcode in swift. Limited to development with Mac servers. A single team most likely needed. | IntelliJ iDE or PyCharm using python on a standard install of Linux. Multiple teams for multiple platforms needed. | Windows would be using Visual Studio using any language you desired on a mid-level device that can handle VS. A single team most likely needed, unless for some reason you’re using multiple languages. | Android’s Android Studio, iOS’s Xcode using swift, or Unity on either device using C++. At least two teams for each company’s devices. |

## 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**:

**For the client’s proposed application, I would recommend development and maintenance be done using Microsoft Windows**. Windows is secure, easy to implement, and versatile. Servers may be more expensive, but no niche experts will need to be hired for development. Windows has many built-in services and features will also assist in some of the following recommendations.

1. **Operating Systems Architectures**:

**Windows can be used with many different languages in many different environments**. I recommend using Java with Eclipse as it is easy to use and will be able to perform all of the client’s requirements. User interaction like the GUI, distribution, and development can all be achieved through these recommendations.

1. **Storage Management**:

Cloud storage would be the recommended storage system. Not only does cloud storage help deter physical failures that can come from common drive storage, cloud storage can be dynamic and the volume can be increased based on your needs. If any physical storage was needed, Windows has built in smart storage features, so managing the drives would not be an issue.

1. **Memory Management**:

**Keeping physical memory in mind during development**, we will make sure that the pictures allocate enough memory to be pulled efficiently, but not to allocate so much memory as to hurt performance.

1. **Distributed Systems and Networks**:

**Using a library like drop wizard, publishing the ap in mobile stores like IOS Appstore, and also offering the game through the client website should handle distribution.**

1. **Security**:

Server side, Windows has many built in security features such as Microsoft Defender. These features will proactively perform in the background with little to no guidance, so they are easy to use. Windows is also adaptable enough so that if extra security were required, it could be easily added from third parties. We will also ensure that proper encryption practices are followed.

On the client side, we will enforce the basic security practices. We will require unique account IDs, strict passwords, and offer two factor authentication. We will also use staggered system back ups so that any damage or tampering can be remedied.