

***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 | 11/10/2022 | Alex Roberts | <Brief description of changes in this revision> |

**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)

This client needs help setting up the environment, so they need our help to streamline the development of this game. The client has requested that a series of software requirements be met for this game application. These requirements are: The 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)

This app is being developed for android only as of right now so there is no constraint when it comes to platform but there will be a lot of changes that need to be made if this changes. We are focusing on the software at this time but we do need to keep in mind what hardware this will be applied to so we need to ensure that our program can be universal to different hardware's. We also need to ensure that we write precise code and make sure that we compress as much as possible so the game runs smooth and the images can render at a steady rate to keep up with the pace of the game. If we overload it too much it could slow the rendering down and take the fun away from the users and make the game unplayable.

## [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)

In this UML diagram it shows that this program uses a singleton style. The Entity class is a base class that holds common attributes and behaviors. The GameService, Game, Team, and Player classes all use the attributes from the Entity class. The GameService class focuses on the inner workings of the game and gives information to the Game, Team, and Player classes. This class is used to set up and start the game while ensuring there is only one game running at a time. The Game class is setup for the user to add their team to the game with their team name which is acquired through the Team class. The Team class is used to add players to the team and name their team. The player information is acquired from the Player class. The player class is the first part of the program that the user interacts with and it is designed for the player to create their player id and insert themselves into the 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** | Mac can be used as a server but it is expensive to license it and you have to use a MacBook for development. | Linux is great for web-based hosting. It is the most popular operating platform as it works great and the licensing is free. | The licensing is expensive but windows has servers that are very secure and easy to use. | Mobile devices are not very well equipped to be ran as a server as they lack the power for high-end but can still be used as a server and development. |
| **Client Side** | You must have a MacBook to develop which can be costly and requires someone who knows how to use swift. Although, they do have good SDK’s that are easy to use. | For Linux you need to know Python and it will require the most development time. | The highest requirement for Windows would be the expertise. You could use .NET framework for security. | User interaction and displays need to be taken care of differently than web-based apps so you want someone that has experience with mobile development. |
| **Development Tools** | Mac coding will be done with swift and it requires a MacBook. | Linux usually comes with Python already installed. You could use IntelliJ’s Ultimate IDE or VSC to write code for this. | Visual Studio Code is the best way to code apps for Windows. You could use just about any language to code for Windows. Most are done in C++. | For Android you need to use Android Studio. For iPhone you need a MacBook and swift. You could use Unity, which uses C++ and then convert it to the correct platform but you still need a MacBook to convert it to iPhone. |

## 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 Linux for this application to allow it to expand it to other platforms. The Linux OS can be compiled and ran on any system architecture. This would allow the application to be ran on any hardware system, even ARM based processors. This would provide the maximum possible amount of scaling to other systems and there would be no licensing fees.
2. **Operating Systems Architectures**: The Linux-based OS is a UNIX-based operating system that uses the Extended File System and Unified File System storage formats. Linux Operating Systems are open-sourced and community-developed thus producing a more refined and secure platform for production-scale applications.
4. **Storage Management**: This game application uses roughly 1.6 GB of files within the application so it should take advantage of the NFS file shares available within Linux to create a small repository of images to pull from. This can ensure that this application will be the only one to access these files which means the files will not be running anywhere else and take up unnecessary space.
5. **Memory Management**: Linux OS generally uses paging instead of segmentation within memory management. To help with this, Linux also creates memory “zones” to classify sections of memory and tell the system instead of the applications what it will have access to.
6. **Distributed Systems and Networks**: Linux OS works very well with containerization which allows container frameworks to create many server nodes that the application can run on. This can mitigate the risk of network or system outages.
7. **Security**: Linux operating systems treat everything as a file. Access to mounted drives is obtained via a special file in the filesystem. As such, the traditional security model for Linux OS’s is to assign read, write, and execute access to every file or folder within the OS. To protect the security of the game, the application can be given exclusive access to the images it needs for each game. As for network security, Linux web hosts have complete HTTPS support as well as native support for end-to-end encryption in some distributions.