

Gaming Room

# **CS 230 Project Software Design Template**

Version 1.2

## 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 |
| --- | --- | --- | --- |
| 1.2 | 04/24/2022 | Kalin Mason | Included Entity base class and abstracted game, player, and team from Entity class |

**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 allows for users to connect to a single instance of a game. They can add games, teams to games, and players to teams. The application doesn’t allow for duplicates of teams, games, or players. Users can join already in progress games or teams but can choose to create their own if needed.

## [Design Constraints](#_2et92p0)

Some of the design constraints for a web-based distributed environment is the OS the game is running on. The game was originally created for Windows OS users and may require new compilation for iOS or Linux. Another constraint of this application is the case sensitivity for team names, player names, and game names. If someone doesn’t type in the name exactly (including casing) they may join the wrong team or game. A future enhancement to convert all strings to upper or lower case could correct this issue.

## [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)

Entity is the base class for Game, Team, and Player, with an association of Game to GameService. Entity has the attributes of id and name that will extend to the child classes. Entity also has methods of getId, getName, and toString while also have two constructors; one base and one that takes a long and string as parameters. GameService has attributes nextGameId, nextPlayerId, and nextTeamId that stores long values to track the next available IDs for these class objects. GameService also contains a list attribute to hold the created games, while also having one last attribute to return the GameService instance to ensure only one instance is made. GameService contains methods to get all the private attributes of it’s class as well. Game class has one attribute that isn’t inherited from Entity and that is a list for the teams of Team object. With game, there are three methods with being a constructor to the class taking a long and string as parameters, addTeam which adds a team to the list requiring a string as the parameter, and toString which will print info about the game and team. Team is similar to game in the sense that it has a list attribute, constructor with the same parameter types, and the ability to add a player to the team and player list. toString method will print info about the team and players. Lastly, we have Player object which has two public methods Player constructor and toString. toString will print info about the player. Last we have the ProgramDriver and SingletonTester which use each other and ProgramDriver is the main function that will run at execution.

**"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** | There isn’t a lot of support for server side hosting from Mac anymore. Mac does offer some nice libraries for CSS though, so this could be an advantage. | One of the cheapest options out there. A lot of support from developers and the community. Downside is that if you’re not familiar with Linux, the learning curve can be steep. | Downside could be the cost associated with programs and applications needed to assist in hosting. Advantage is the large amount of documentation available online and available. | An advantage is the ability to access the full device, including software and hardware. While a disadvantage is the limited programming languages and skills needed for mobile devices depending on the OS. |
| **Client Side** | Mac offers a rich amount of libraries for UI and UX design. Integrated ecosystem for all devices.  Lack of proper package management systems. Takes time to get used to the differing tools for Mac OS. | JavaScript  Visual Studio  NetBeans  Linux offers a rich environment of tools to development client side code in. With the ease of use of the CLI, you can leverage that to create rich sites. | JavaScript  Visual Studio  Windows offers a lot of programming advantages such as documentation for a lot of the available tools.  While a disadvantage of Windows is most of the tools are proprietary and require a license. | Objective C  Java/Kotlin  Linux is an open source OS and has many advantages such as free or cheaper tools and solutions. A downside of Linux client side development is the need to learn and adapt to Linux if not used to the OS. |
| **Development Tools** | xCode IDE  Homebrew package manager  iTerm2 terminal  Objective C | NetBeans  Ubuntu  Sublime Text  Visual Studio Code  C programming language | Mostly C  Visual Studio  .NET | Visual Studio  Android Studio  Swift  Java |

## 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 think Linux should be used to host the application. With the cheaper or free options available, it makes more sense to use Linux.
2. **Operating Systems Architectures**: Linux can host the application on a server that is maintained only with the CLI. This allows for better resource management for the application and allows the application to run without a lot of overhead or need for more equipment or tools.
3. **Storage Management**: Logical Volume Manager (LVM) is a software that can automatically allocate the appropriate disk space for an application or service on the server. This will allow for automation of the SM software without a lot of oversight from someone.
4. **Memory Management**: Linux takes user input and stores it into RAM which can read/write to and from the HDD. The HDD can store data that needs to be stored longer than what RAM will store it for or may lose. RAM will utilize the caches on the CPU in order to create faster read/write operations. If a process is needed to be executed, it’s first loaded into RAM then executed. The executed task could write data to the CPU or RAM caches as needed.
5. **Distributed Systems and Networks**: Since this is hosted on the web, each client can connect to the application via web sockets. The web sockets won’t require a specific OS to be used, but may create varying experiences which can be handled by the server. The advantage of using web sockets is the ability to allow others to use their OS of choice, but may be required to use a specific browser or collection of browsers.
6. **Security**: Linux requires each user to be walled off from each other. These users are protected behind a username/password combination and have lower automatic access rights. This makes it harder for malware to spread between clients and due to the nature of having multiple distributions available for Linux, it can be harder for malware to spread due to the various operating environments, system architectures, and components.