

The Gaming Room

# **CS 230 Project Software Design Template**

Version 3.0

## 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) **6**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/30/2022 | Liam Nunes | Initial draft |

| 2.0 | 02/13/2022 | Liam Nunes | evaluation |
| --- | --- | --- | --- |

| 2.0 | 02/13/2022 | Liam Nunes | 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 client, The Gaming rooms, wishes to expand their android game by adding a web-based version. They wish for our company to provide and environment to host their games in this new expansion.

## [Design Constraints](#_2et92p0)

* Games have to have the ability to host multiple teams. To be usable for multiple games, we must be able to implement any number of teams in any game.
* Teams must be able to contain multiple players. Teams may need to be different sizes for different games so we must include the ability to have any number of members on a team.
* Game and Teams must have unique names. To avoid confusion we must make any user who tries to choose a taken name choose again.
* Only one instance of the game may be active at a time. We don’t want memory bloat so we must limit the number of games open at a time to 1.

## [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 Service contains many instances of the Game class which contains multiple instances of the Team class which contains multiple instances of the Player class. This is an example of Encapsulation as it is limiting assess of things from the class being implemented by the class implementing it. The Game, Player, and Team classes are all inheriting from Entity, this means that all of those classes has every method and variable that Entity has. This is also polymorphism the code in entity is designed to be used by many different types of classes.

"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 has a lot of flexibility when it comes to the terminal and the user interface is also pretty friendly to use. | Linux is probably the most difficult to use out of the three but has by far the best console. The other advantage is the low operating cost. | Windows has a great user interface but is also the most expensive on the server side. The cost is offset by the fact that there is more software for developers. | This is going to be highly dependent on what brand of phone you are looking at. |
| **Client Side** | Mac is high in cost like Windows. It also requires a fair bit of time to learn as it tends to be more difficult to learn then windows. | Linux is by far the cheapest option for users. That said it does tend to be the most technical and will require more time for users to learn the expertise necessary to build and maintain applications on Linus. | Windows is high in cost like Mac. It is however the easiest for users, it requires little time to learn how to build and maintain applications on windows. | Given the variation in mobile OS’s it can be difficult to adapt applications made for other OS’s and to maintain them thus mobile applications tend to take high expertise And time. |
| **Development Tools** | Languages-  HTML, Python, JavaScript,CSS,Ruby  Tools-  Swift, Notepad | Languages-  HTML, Python, JavaScript, Java  Tools-  Visual Studio, Eclipse, Notepad | Languages-  HTML, Python, C++, Java  Tools-  Visual Studio, Eclipse, Notepad | Languages-  HTML, Python, C++  Tools-  Visual Studio, Eclipse, Notepad, command prompt |

## 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 Microsoft Windows for the operating platform for this application. The first reason is this game is meant to be played by as many people as possible and in order to play it they will need to have assess to it. Windows has the largest customer base so more potential clients will be able to be reached. Windows may be a higher cost for the consumer but more consumers already have windows so I don’t see that as an issue. Additionally it requires minimal experience to use. Another factor is it has a large availability for software to aid in development.
2. **Operating Systems Architectures**: Windows applications can run in either kernel mode or user mode. Kernel mode will allow the application to access hardware directly and gives the application direct ability to tell the CPU what to do. User mode does not have access to the hardware and has to go through system APIs to communicate with the hardware. Given that this application should have no real need to communicate with the Hardware or CPU I think that a user level application is best, as a crash in the user level is far less severe as it is not communicating with the CPU directly.
3. **Storage Management**: Windows Server gives many tools for running and operating a server to store the data for the application on. Since this is going to be a primarily over the internet multiplayer game, I would recommend using a server rather then local storage. For one thing it will help to insure image integrity for the game, it will also not take up as much space on the end users device. The downside is there will be no offline play, but given the parameters I do not believe that to be a problem.
4. **Memory Management**: As far as memory management goes I think it is pretty straight forward. The user doesn’t need to keep every image for the game on their local memory, I would recommend downloading the first image, and while the user is playing download the next 2 of 3 images into memory, then as soon as a round is done delete the image that was being used and download the next. This should keep the memory free and clear while ensuring smooth game play.
5. **Distributed Systems and Networks**: First thing to consider if you want the application to run on all platforms I would recommend using Java as the programming language it is built in, Java has large support and comparability on Windows, Mac, and Linux. By using distributed systems you can slip the components that need to be run on all the different machines between them and have them share information, this reduces the load on any one machine and ensures that the various applications are synced up better.
6. **Security**: using windows server means that the client will automatically have access to build tin security measures such as Windows Defender Advances threat Protection, which will provide security to the server. On top of that I would recommend account locking all user data, so you need to log into your account with an encrypted password in order to access your data. I would also limit the information that admin accounts can access to only what they need to do their job.