

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

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 | <mm/dd/yy> | <Your-Name> | <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)

To solve the problem of having the ability to have one or more teams involved we are creating a list of teams inside the game object.

To solve the problem of having multiple players assigned to a team we are creating a list of players inside a team object.

To ensure uniqueness of a player and or team we are iterating through the list before a team or player is created and returning the player and or team if they are already found in the list

To ensure that only one instance of a game can exist in memory at a time we create a unique id for each game instance. Not with a UUID like you would expect but just an increment operation.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The design constraints for a web based are

1. The code is written in java so we will need a controller to send the information to the presentation layer (CSS/HTML).
2. There is no persistence of data so when the application is closed the players, games and teams will be lost
3. There is no limit set to the amount of games teams can join or teams players can join
4. The entity element violates the Liskov substitution principle making the cohesion not as good ( I can never remember if its high or low cohesion)

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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 very limited support for hosting a production server. | Linux is the industry standard for server side hosting, Apache is free and open source and widely supported | Windows has windows server and requires a license. | You cannot host an application on a mobile device. |
| **Client Side** | As a web application the app will use HTML/CSS/ and JavaScript, as long as a browser is installed the app will run fine | As a web application the app will use HTML/CSS/ and JavaScript, as long as a browser is installed the app will run fine | As a web application the app will use HTML/CSS/ and JavaScript, as long as a browser is installed the app will run fine | For IOS the app will be developed with Swift, for Android the app will be developed in Kotlin. These are specific languages for their respective devices. |
| **Development Tools** | I would use VScode for the front end, I would use PhaserJS as it’s a powerful JavaScript library for making games, and Eclipse for the backend (Written in Java) both are free | I would use VScode for the front end, I would use PhaserJS as it’s a powerful JavaScript library for making games, and Eclipse for the backend (Written in Java) both are free | I would use VScode for the front end, I would use PhaserJS as it’s a powerful JavaScript library for making games, and Eclipse for the backend (Written in Java) both are free | For IOS you need a mac with Xcode, for android you need android studio and will use Kotlin. It will use the same API calls to the Java Backend |

## 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**: The operating platform we will choose to host our game, will be Ubuntu Server. We will choose an LTS (long term support) version so we can make sure we get the next 4 years of updates and bug fixes. Ubuntu has been a leader and most well-known Linux distribution since Mark Shuttleworth started the ubuntu project. Ubuntu is optimized to run Java, and integrates well with Apache.
2. **Operating Systems Architectures**: The architecture for Ubuntu Linux is a 64 bit architecture with multicore/multithreading support to take full advantage of Java threads. The operating system is container ready for ease of devops integration.
3. **Storage Management**: Storage Management will be a raid backup and NVMe SSD main storage. This will provide the fastest read/writes virtual memory and database access.
4. **Memory Management**: We will use virtual memory with paging, so each process will have a physical address space which will be swapped out as needed to a logical physical space when necessary
5. **Distributed Systems and Networks**: Java will be running in the backend as a RESTful service providing API calls through HTTPS. Images will be streamed from Blob files in the database to the user using Java Streams.
6. **Security**: Security will consist of Least privileges principles, sudo will be required for admin privileges. We are using the JVM which is one of the most secure sandbox platforms on the planet. Every piece of data will be considered untrusted and will be checked and cleaned. Ubuntu firewalls will be in use.