

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc98678305)

[Table of Contents 2](#_Toc98678306)

[Document Revision History 2](#_Toc98678307)

[Executive Summary 3](#_Toc98678308)

[Design Constraints 3](#_Toc98678309)

[Domain Model 3](#_Toc98678310)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03-20-2022 | Apurva Shukla | Start of the project. |

## [Executive Summary](#_sbfa50wo7nsh)

The company CTS behind the product The Gaming Room wants to create a web-based version of their already established Android app. This requires the use of Java since the developers already have experience using it and is easier to maintain in this way.

## [Design Constraints](#_2et92p0)

The application must scale well. The Android app interface may not work well for the web interface, and will have to be tested and possibly changed to fit the interface of the web. The website also has to work well with mobile and desktop devices, and if possible, should scale just as well with other forms of electronics devices, namely game consoles and smart TVs.

## [Domain Model](#_8h2ehzxfam4o)

The Entity Class is the parent class of the other classes. The Entity class contains the basic required method calls and is inherited by the other classes. The Team class contains the addPlayer method, which calls the Player class methods and creates a method to add players to the respective teams. The Game class contains the addTeam method which calls the Team class and methods to add teams after the game is initialized. The GameService class keeps track of all IDs that are running and makes sure that there is only instance of each Game, Player, Team running. If there is any discrepancy, the GameService method will throw an exception.

**"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.**

## 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**: Linux should be the platform of choice for the company, as Linux provides better scaling options in terms of server management, creates smaller, less power-hungry when creating multiple instances of the game, and have the infrastructure costs be low and very easily scalable when working with higher user counts.
2. **Operating Systems Architectures**: The architecture recommended is the use of a two-tier application, where the usernames and storage of photos and database values are handled by servers, whereas the client handles the retrieval and processing of user data from the server to get the game started.
3. **Storage Management**: Storage Management in a two-tier application is managed by the server, and since the client only has to parse the images from the server and so the server can have RAID configurations so that the images are stored always, and won’t have to worry about losing the images, and the client-side application can stay a small size.
4. **Memory Management**: Since the client side has to parse images, it needs to be able to discard the fully completed image once downloaded, so if the game uses a memory buffer and manages the image from there, the buffer can always be cleared to keep memory management a low-hassle deal.
5. **Distributed Systems and Networks**: The client-server application will work on REST API implementation, and hence the calls made will never be a system-specific call, but will make REST specific calls, which are the same throughout all operating systems.
6. **Security**: Security will be handled by using common verification techniques such as SSL certificates for connecting to the sever securely, public-private key usage for protecting user information on both server and client side, and captcha usage to prevent bot attacks when user logons to play the game.