

The Gaming Room Software Implementation

# **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 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/19/2023 | Elijah Thomas | First Draft |

**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)

## Requirements

The gaming room is looking to branch out into different operating platforms and has come to us at Chada Tech to consult regarding how to do so. The company has decided they would like for a solution where each game will have more than one team, each team will have multiple players and teams must be unique. In this summary I will recommend hosting the game service on a Linux server environment.

## [Design Constraints](#_2et92p0)

1. Service must be compatible with different browsers
2. Maintaining iOS/Android applications requires a team of dedicated engineers.
3. Service must be hosted online through some choice of server administration.

## [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 UML class diagram below shows how we will implement the software solution for our client. As you can see through abstraction and inheritance we will abstract the class entity and use it to keep track of all unique names and id’s. We will implement a singleton design structure through our game service class which also serves as a good representation of polymorphism. The game service class will be able to handle different input and still return a game instance based on the need of the user/engineer.

**"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**   * multithreaded I/O, * security * server scalability * physical/digital infrastructure * LDAP/ADP * Cloud support * Cross platform capabilities | Pros:   * high quality UI * Standardized development * LDAP support * ADP for iCloud   Cons:   * Low Compatibility * Unfamiliar operating system | Pros:   * Infrastructure control * High security * Stability * Ease of access * Cross Platform * Can be hosted by LDAP using PAM * PAM (Pluggable Authentication Modules)   Cons:   * Unfamiliar operating system * High margin for error * Longer development lifecycle | Pros:   * High security * Prebuilt Resources * Accessibility * Cross platform * ADP * Can be hosted by LDAP   Cons:   * Resources window specific | Pros:   * Speed * Performance * Offline access * Marketing   Cons:   * Longer development lifecycle * Storage cost to users * Difficult updates * Shared profits with Appstore |
| **Client Side** |  |  |  |  |
| **Development Tools** | * -apple disk utility * -net install/net restore * -deploy studio * -package Maker * -swift * -macOS * -AWS * -C# * -C * -JavaScript * -HTML * -CSS * Java * Maven * Dropwizard | * -Visual Studio * -NetBeans * -eclipse * -AWS * -C * -rust * -JavaScript * -HTML * -CSS * Java * Maven * Dropwizard | * -RSAT * -Visual Studio * -AWS * -C * -C++ * -JavaScript * -HTML * -CSS * Windows server machine * Maven * Java * Dropwizard | * -react * -ionic * -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 recommend using the Linux operating platform as we are able to configure, implement, and test different server configuration to best fit the client.
2. **Operating Systems Architectures**: I recommend the Linux system architecture because it has high cross platform capabilities, auditable security features, scalability.
3. **Storage Management**: A server database with good scalability should be sufficient to save cost for this project. I am recommending AWS’s DynamoDB as the data and image files that will be stored do not need a relational database.
4. **Memory Management**: the Linux systems architecture allows for low level projects to be made that will handle server/client resources memory efficiently.
5. **Distributed Systems and Networks**: Systems and networks would be most secure on a virtual service provider such as AWS or Google cloud. For this project I also recommend using the Linux Pam module to import and use LDAP features and security preferences.
6. **Security**: By using a third-party data encryption platform to organize, secure, and protect user data we will ensure that our systems and architecture are secure and compliant with government regulations. For this project I am recommending the data encryption company Baffle as they specialize in protecting enterprise assets in a cloud environment.