

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/14/2023 | Win Inlaw | continuing developing game application |

**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](#_35nkun2)

The software design problem is the issue of creating a game application while yet being web-based. A current proposed solution is taking the application and making object-oriented programming. Such classes would be Game, Team, GameService, etc. These things can be combined when making different commands creating all these classes to interact with themselves.

## 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](#_1ksv4uv)

* Web-based- The game application being developed is required in a web-based distributed environment
* Java Programming Language- The project will be developed in the code of Java programming language
* Scalability and Performance- We want the game application to be able to obtain a large number of games, teams, etc.

## [Domain Model](#_2jxsxqh)

* The **GameService c**lass maintains games and provides methods to add them. This guarantees a single instance following Singleton is accessible when going through the application.
* The **Game** Class maintains a list of teams taking place in the game. This also provides ways to add teams.
* The **Team** class stands for a team in the game and its players.
* The **player** class is for the player while giving a way to retrieve a string representation.
* The **Entity** class is the starter for all classes for this application. Allowing the other classes to inherit all attributes and behaviors to all that apply.

**"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](#_z337ya)

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** | characteristics-popular in web hosting. It also allows for an extremely easy creation of features for Mac clients then Windows clients  advantages-we can further advance and upgradable as there are different web hosting. Offers major advantages if you use Mac clients in your network.    disadvantages- not as recomanded for web hosting services | characteristics- most people desire beacuse it is secured. a unixbased.  advantages-even minor details are cuaght early on. environemnt thats powerful, stable and reliable.  disadvantages-harder to find more support in web hosting applications | characteristics-it is a close platform. Supports traditional hypertext marl-up language files.  advantages- it will have less loading time and very high comfortablility. Windows hosting.  disadvantages-very much poor technical support | characteristics-this prodoct is more popular and has very high portability.  advantages-This has a higher reach and much wider reach. This is also very cost-effective.  disadvantages- very poor security. hard to code your entire backend yourself . |
| **Client Side** | cost- similar to windows  time- moderate amount of time  expertise- moderate amount of expertise. Mac is costlier than Windows and the user is forced to buy a Mac system built by apple. | cost-This will require miniual cost  time-Maxium  expertise-Maxium. Best operating systems around. | cost- very similar to mac  time-minimum  expertise-minimum  great platform for developing web sites and native windows apps. | cost-minimum cost  time- minimum but yet flexibile  expertise-Maxium difficult. falls bewteen phones and tablets |
| **Development Tools** | HTML/Javascript/C++ | HTML/C++/Javascrpit | HTML/C++/javascrpit | HTML/C++/Javascrpit |

## 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 say the gaming room should lean on windows devices. Minimum expertise and minimum cost.
2. **Operating Systems Architectures**: Windows allows applications to represent GUI while granting system resource.
3. **Storage Management**: Windows allows you to control files on your hard rive. While still allowing the cloud to save data.
4. **Memory Management**: The memory allocation is a great tool allowing for easy storage of pictures of course out side of the default picture folder.
5. **Distributed Systems and Networks**: Develop enables cross-platform game creation and with it being a IDE that can completely run on any device.
6. **Security**: Windows comes with a built-in security protection software. it will always recommend another source to secure user data and information.