

<CS230 Project 1>

# **CS 230 Project Software Design Template**

Version 1.0

## 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)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.1 | <11/10/21> | Raven DeCoste | Refactored and added team and player adding |

**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 wishes to add the ability to add teams to games, and to be able to add players to those teams. A UML diagram was provided as an outline. The Game, Team, and Player classes must be refactored to extend the Entity class.

## [Design Constraints](#_2et92p0)

In designing a web applications, the constraints of which things occur client-side and server-side must be considered from both a functional standpoint and a security standpoint. Ensuring proper communication between the two is another constraint.

## [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, Team, and Player classes extend the Entity class. Players may be added to teams, teams may be added to games, and games may be created. The game service class manages the set of games. A singleton tester exists to ensure that the singleton design pattern is properly implemented. There may be 0 or more of each entity.

"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 servers may be easier to set up for those who have little system administration and networking knowledge compared to Linux, however, they are both UNIX systems, so it mostly comes down to the user space stack. | Linux servers are free from the software bloat of Windows and are a great choice for a highly configurational server. | Many people are familiar with Windows. Such familiarity may make a Windows server easier to manage than another OS. | Hosting a server on a mobile platform would be inefficient compared to a stationary server. |
| **Client Side** | Mac is a Unix system and cannot run certain Windows specific libraries and Function calls found in many applications, however, avoiding these, code written for Mac can often be easily made to work on Windows. | Linux is a Unix system and is good for developing software that has good compatibility. | Windows often utilizes libraries and system calls not supported on Unix systems, however, Windows has the largest market-share, thus, programs for Windows will run on most user’s systems. | Mobile architectures are a more specific use-case than traditional desktop operating systems, thus more specific expertise is required, which may drive up cost. |
| **Development Tools** | Vim is a popular development environment. For those seeking a traditional IDE, eclipse is a good alternative. | Vim is a popular development environment, especially among Linux users. For those seeking a traditional IDE, eclipse is a good alternative. | Eclipse is a popular development environment for Windows, as is Visual Studio, however, it is not open source. | Eclipse is an IDE that can easily be used with mobile-specific languages like Kotlin. |

## 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 service should be hosted on a Linux server.
2. **Operating Systems Architectures**: The program should be made following UNIX specifications and should avoid Windows specific libraries and system calls in order to promote compatibility between systems.
3. **Storage Management**: Information stored-client side will be deleted once games are completed.
4. **Memory Management**: Caching can be implemented to speed up games given how often information will be requested from the server.
5. **Distributed Systems and Networks**: Ensuring that each platform sends the same format of data to the server will allow the server-side components to work the same for every system that is connected.
6. **Security**: User passwords and other sensitive information will not be stored in plain text, instead it will be stored using hashing. Linux is a popular OS when considering security.