

Draw it or Loose it

# **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 |
| --- | --- | --- | --- |
| 1.0 | 09/18/2023 | Devin Bashaw | Initial proposal |

**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 Gaming Room is a mobile software company wanting expansion through the development of a web-based gaming application. They want multi-player ability with multiple teams that can collaborate with one another, but only permitting one instance of the game to take place in the memory at a time. Team and player names will also be unique.

## [Design Constraints](#_2et92p0)

* Each web browser has limits that will control how the game is imaged
* To support as many browsers as possible, code will need to be changed
* All game versions generated for each browser have to be the same
* Connecting mobile and web users
* The code needs evaluated to ensure that mobile and web clients can collaborate
* Current back-end may need modified to support all platforms
* Future changes have to consider all platforms across the board

## [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 program driver runs the program, starts and manages the games, as well as manages the functions made available by all classes. The Singleton test initiated by the Program driver tests the functionality of the GameService so that only one instance of the GameService exists at a time. Following OOP principals, Game comes from Game service and represents an instance of a game. It can have one or multiple teams with players in these teams to follow OOP principles concerning reusable models. These three classes: Game, Team and Player can use Entity to create sub-objects underneath them.

**"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** | All types of application will use the same back-end server communicating through http/https requests. Since the same data will be required for all versions of the application, the server can respond to the same types of requests and allow for cross-platform collaboration. This will save time, money and resources since only one server-side instance is needed to support all platforms. | | | |
| **Client Side** | Mac typically uses Safari browser. When accessing the webapp from safari the client-side CSS and HTML will need to identify the browser format then format content appropriately | Linus distros typically use Firefox, but this varies and could be complicated to support more browser styles. When accessing the webapp from the client-side CSS and HTML will need to identify the browser format then format content appropriately | Windows users will typically use Google Chrome or Microsoft Edge. We will need to code for both in the HTML and CSS application. | Android, Apple and Windows need to be considered for mobile platforms. All three need to be accommodated for. |
| **Development Tools** | Server side is dependent on the platform it runs on. A simple text editor such as Notepad++ is good for developing the front-end application design for the webapp and should provide the tools needed to script the app. For back-end, Java is a good choice since it is easily transferrable. The application itself can be coded in C++ or C# with the appropriate IDE to support the chosen operating system platform for the server-side application to run. | | | Mobile apps have their own IDE. The server-side will be dependent on the rest of the application framework. |

## 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**:
   * Ubuntu Server
2. **Operating Systems Architectures**:
   * Ubuntu Server is extremely low weight and good for a server-side application to run idle forever. This server is better at storing memory, typically faster and less resources are needed.
3. **Storage Management**: SQL Server, running on the Ubuntu Server storing data onto a RAID 1 storage array. If storage fills up then upgrade to a RAID 5 system with more storage. 120Gb volume is good to start with until the server gains popularity.
4. **Memory Management**: Ubuntu Server allows for raw memory management which can be good and bad. It will allow the application to run faster since it is not operating in a virtual block of memory space, but if an application with errors comes by the system will present stack overflow. Extensive testing is needed to make sure this does not happen.
5. **Distributed Systems and Networks**: Following this model, since the back-end server is designed to respond to requests of all applications, there needs to be room for collaboration between all platforms. The SQL server and application server will have to run on the same machine, if needed the SQL server and application server can be separated but will require a strong network connection. If the network goes down, the software will not work correctly or may fail. Fail-safe mechanisms need to be considered as well due to this possibility.
6. **Security**: Password and user information will be encrypted during client-server transport using HTTPS protocol, and the data stored will be encrypted in the SQL database. SQL to client server communication will be encrypted and any sensitive password information will be hashed to protect it during the event of some sort of breech.