

<Name-of-Software-Application>

# **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 | <03/21/2021> | Alexis Fuerte | Creation |
| 1.1 | <04/04/2021> | Alexis Fuerte | Evaluation |
| 1.2 | <4/18/2021> | Alexis Fuerte | Recommendation |

**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)

Those at the gaming room would like to develop a gaming application that can be used on multiple operating systems. They are unaware on how to create the software for it, which is where we come in to streamline the creation. We will create an application that meets the parameters set by the company, as described in the design constraints.

## [Design Constraints](#_2et92p0)

Must create unique identifiers for names of game, team, and players

Must be able to check if a name is already in use

Must allow multiple people to play on a team

Must allow multiple teams to compete

## [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 Entity class is the parent class to Game, Team, and Player, meaning that those three child classes can inherit things from the entity class, such as id and name. Also, the Game class receives information from the Team class, and the Team class receives information from the player class in order to make the lists. The GameService class is what helps list the information on the screen, and receives information from input. Meanwhile, the Program driver is simply the main driver that runs the entire program, and the singleton testor works off of it, ensuring that the classes created are being used for one instance only.

****

## [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 maintains fantastic compatibility rates between its own systems, but their servers are costly, and not wonderful for connecting outside of their own bubble of products | It is open source, and is instantly compatible with other open source programs. It is also less likely to crash, and can be modified without rebooting | Windows comes with technical support and is very user friendly, just at a higher price for it, as well as constant updates. | Finding someone to host your server is very difficult on here, and it takes strenuous amounts of work and knowledge of coding to maintian |
| **Client Side** | The user needs to have something that operates macOS, which is only systems made by apple (which are extremely expensive). | Linux is well known as a free operating system that rarely deals with viruses and malware, keeping the client happy and worry free. It is also compatible on a vast amount of computers. | Windows can operate on many things, so it isn’t as limited as mac, but it still does not compete with Linux in terms of simplicity and price. | Mobile devices come in way more shapes and sizes compared to the other three OS’s and it could make it very difficult to please the customers as it becomes prone to many issues. |
| **Development Tools** | Mac mainly uses Objective-C as their primary programming language, and they use Xcode IDE | Linux is written mostly in C, and mainly uses eclipse and NetBeans as their IDE. C is a broad language that many programmers are familiar with. | C is used to program many things on the windows platform, but they are also known to use c++. Their main IDE is the Microsoft Visual Studio, created to run perfectly with their program. | Java is the main language used to write applications on mobile platforms, and there are a variety of different IDE’s available to developers. |

## 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**: My recommendation is that you use Windows as your operating platform. I recommend them because although it comes with a bunch of other downloaded things that take up data storage, most games you’ll find are licensed through windows, and automatically made to run on windows. This means that a large portion of those who would likely want to play your game would already have windows, and then you could adjust to other operating platforms in the future. MacOS was simply not made for gameplay, and you’ll find that linux is lacking in the licensed gaming department as well.
2. **Operating Systems Architectures**: There are many different aspects of the OS architecture for Windows, so I will just touch on the main ones that stand out. First, It has HAL, or Hardware Abstraction Layer. In simple terms, HAL allows multiple applications to access portions of the hardware. Then there is the microkernel which aids with address space management, thread management, and inter-process communication (IPC). Last, there are three main environment subsystems: the Win32 subsystem the os/2 subsystem, and the POSIX subsystem.
3. **Storage Management**: For this I highly recommend an SSD rather than an HDD, only because with SSD’s, you’ll find the system runs much more smooth and fast. Naturally, there are many different kinds of SSD’s these days, but to run a simple program like this game, I’d say you wouldn’t need anything to big/fancy/expensive.
4. **Memory Management**: Object pooling is something that will be used for this gameplay, as the game has a bunch of photos that will be pulled out of memory, so having them on hand makes it easy to quickly grab the necessary object for the round and return it when done.
5. **Distributed Systems and Networks**: The nodes in network operating systems are able to have their own OS, as opposed to distributed systems requiring nodes to have the same OS, so the network OS would be good to allow various platforms to communicate through files and messages.
6. **Security**: So the great thing about Windows is that they have security measures already built into their OS, so you are already at an advantage, but also by using the windows OS, we get to use kernel mode. I touched on this a little earlier in the architecture portion, but kernel mode separates what regular users are able to do vs. those who need access into the actual operating platform, keeping users safe.