

Joshua Millan

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 1/22/2022 | Joshua Millan | First draft of project |
| 2.0 | 2/5/2022 | Joshua Millan | Reviewed all work and expanded on evaluation section |
| 3.0 | 2/24/2022 | Joshua Millan | Final review and expanded architecture recommendations |

**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 game company wants a game that times players to solve a puzzle as well as tracks teams, players, and the game. They need 4 things a game will have the ability to have one or more teams involved. Each team will have multiple players assigned to it. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. lastly, only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

A constraint is having multiple teams and games going on but no repeats. It will also be hard to tie all classes into one to get the game working properly. Staying organized will also be a constraint since there will be a lot going on.

## [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)

To begin we have 3 classes that extend from Entity. Then you need to start with Entity to capture things that will be needed across all 3. Then you need to go through Game, Team and Player and build those for what is needed past the over all entity. All 3 are needed because they all take information from one another to function properly. Game services comes after that to form the game more and ensure that things only happen once and there are no issues as well as organizes everything. Once these main things are done to test everything the driver will use the tester to ensure no errors so far.

**"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 must work together.

|  | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Apple has a huge iCloud network and many servers to back up all their devices across the board  Mac OS server is a program that may be used on a Mac. According to Apple's website, Mac OS Server is merely $20, making it a low-cost solution. However, for these duties, Mac is not as popular as, instance, Linux or Windows. | Linux has a Linux server that is built specifically for the Linux operating system.  Linux is noteworthy since it has a variety of server-capable distributions. Linux Server is a low-cost, open-source operating system. But you'll need someone with Linux experience to run the server. | Windows has many servers and many of third parties that have server on windows, so you have options.  Windows Server is a service provided by Microsoft. According to Microsoft's website, implementation may be pricey, but it is completely functional. | Each mobile company has their own servers, so you must use all if you want the software to be on all.  In terms of cost, running servers on mobile devices is the most cost-effective option, as it requires little to none to get started. |
| **Client Side** | Easiest to use, best ecosystem, and simplest looking as well. For the client this is a good “daily” that still can do a good amount.  Use would be dependent on competence, with someone with more experience with Mac requiring less time and someone with less experience with Mac requiring more time. | Advanced for the user’s aspect. Very flexible and customizable for those that are well versed. But a more Advanced complex system.  You'll need a lot of time and knowledge because Linux isn't widely used, and you'll need someone who knows their way around it. You'll also need to give them plenty of time to work because Linux can be challenging even for experienced users. | Windows the happy medium between complexity and functionality. Very well known and understood by many users.  Time would be dependent on skill, with someone with more Windows experience requiring less time and someone with less Windows experience requiring more time. | Mobile devices are simple to use when looking at simple things like games or typing. But more complicated software’s like photo editing is hard due to the size of the display and the phones “power”.  Because mobile devices are simpler to operate with, experience may not be a problem. There would be more time required because there are numerous operating systems and mobile devices to be worked on. |
| **Development Tools** | FlexiHub  Homebrew  Xcode  iTerm2  Sublime Text  NetBeans  Tower  SourceTree  Atom  Visual Studio Code  Dash  Go2Shell  Swift is the most popular programming language for Mac apps. Swift may be utilized with a variety of IDEs, including Atom and Visual Studio. | Gedit  Vim  Netbeans  Bluefish  Geany  Aptana  Quanta Plus  On Linux, the IDEs Eclipse and Atom are widely used. Eclipse is typically used for Java, although it may also be used for other languages such as C+. | Rad Studio  Embold  Collaborator  Studio 3T  Linx  Genexus  Kite  NTFS Permissions  DbSchema  NetBeans  For Windows, Eclipse and Visual Studio are popular IDEs. Visual Studio may be used to create in HTML, C#, and JavaScript, among other languages. | Ionic  Neptume DX  FileMaker  Alpha Anywhere  Zoho Creator  Mobile Impact  Modo Campus  The development tools for iPhones are like those for Macs, and iOS apps are often built in Swift, however iOS and macOS have different interfaces. |

## 

## 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**: Windows would be the best so that you could expand. This is because there are more developers on Windows and more projects on it as well. So, you have a greater community with more knowledgeable developers. Windows also boasts plenty of useful IDEs to make the process easier.
2. **Operating Systems Architectures**: Windows provides services that enable all Windows-based applications to display a Graphical User Interface (GUI), access system resources, and much more. Graphics and Multimedia, messaging, and web services are all examples of these applications. These services can be accessed by a user account or a server.
3. **Storage Management**: Windows allows us to add more storage to our system, especially if we have space for extra disks to merge into a storage pool, which will improve performance and data protection with Storage Spaces. It could also assist us in making the most of the storage you already have by removing clutter and utilizing the cloud.
4. **Memory Management**: We will generate and require many files when we begin the project. We could create those files from scratch, but there's a strong possibility that using one of these game engines, which is a software development environment for game creators, will make things lot quicker and faster. Instead of hard coding everything, the game engine may supply us with various libraries and effects that are already constructed, which would save us a lot of time. It will be easier to arrange these files if we have our own library database on the computer we are using.
5. **Distributed Systems and Networks**: Even though I am presenting and advocating that we begin with window-based programming, I recognize that we must eventually support all platforms, independent of operating system or device. This project may benefit greatly from the use of a cross-platform game creation tool. I discovered that 'Unity,' one of the most widely used and cost-effective cross platform game engines, can handle Windows, Linux, Android, iOS, and other platforms. It's also crucial to pick the correct server. Because many people may be playing the game at the same time, it's critical to choose a strong network and server that can always manage the high demand.
6. **Security**: Mac OS would be best for this since in my opinion all the malware is written on Windows or Linux. They also have very good security protocols tat less hackers try to get through compared to the others. Windows is widely used and is well known for hacking and having poor security. Linux is a harder operating system so the people that use it are good enough to know the loopholes and exploit the systems. But Mac OS doesn’t have those issues thanks to not being mass adopted by the whole community.