

<Trevor Enos>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <1/26/2025> | <Trevor Enos> | <In this version I summarized the design requirements and any kind of ways that we can accomplish these designs. I also talked about constraints that could come up and cause issues for the program. I also talked about the UML and how each part of the program relates to others and also how each class works. Lastly, I talked about the OSs and the strengths and weaknesses of each one and which one I can recommend based on certain uses.> |
| 1.1 | 2/14/2025 | <Trevor Enos> | <Filled out the evaluation table and described the characteristics for each platform for all three of the main topics. |

**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 problem that our client is having is that they need a basic application setup for them that has the functionality needed for their game. This application also needs to be streamlined and efficient. The client wants the game to have the ability to have one or more teams involved. They also want each team to be able to have multiple players that can be assigned to them. Our client would like that each name of the game and teams to be unique and there is a function that checks if a name is already being used. Our client would like only one instance of a game to be stored within the memory at a time, this is done by creating a unique instance identifier for each type, so one for game, team, and player. As far as the hardware environment the application needs to be compatible with all platforms specific hardware. For PCs we use normal cpus and gpus and etc, but for other platforms, we might use mobile chips for each respective platform. For instance Apple now uses its own processors especially in their macbooks and ipads, because of this we need to make sure that our application runs well on their processing platform. For android it depends on the phone but they also use different chips than what we would see on a desktop platform. >

## Requirements

*<*For this client we need the ability to add multiple teams with multiple players in each team. This can be done with our addTeam and addPlayer methods that will also include a while loop with the functionality that checks to see if a team or player with the name given already exists within our list of data. Lastly, we only want one instance, we can achieve this by using an if loop to see if an instance already exists, if it does then it returns it, and if it doesn’t then creates a new instance.>

## [Design Constraints](#_2et92p0)

<Constraints that we could encounter are as follows. The first is that because we are creating an entity class we need to make sure that the player, team, and game classes, are properly accessing the entity for the output to display as desired. The next constraint is that we need to make sure our check functions work properly, this is because if they don’t we will have duplicated names and could possibly cause errors. It could also cause problems with our instances which in turn could cause the entire process to not work as intended. Lastly, or add and get functions needed to work as intended and access the respective lists for each value type of game, player, and team. If this does not happen when a check function is called it will not know if a name is already taken, this could also cause problems with our output making it so that either nothing displays or the values of the display are incorrect. The game will need some kind of complex management system that oversees and can manage the teams and their users. The application will need adequate storage and memory values so that the desktop application can store and render images at a level that is similar or even better than that of the mobile platforms specifications/statistics.>

## [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)

< For our UML diagram we have an Entity Class this class is a superclass that consists of three other classes. These three classes are Game, Team, and Player. These classes host our data for our players and teams this ensures that we are able to have multiple teams within our game and those teams have multiple players. These classes are the main entities of the game application. The GameService class has a relationship with game class. The purpose of this relationship is that it manages our game instance and also grabs data from the game class. The ProgramDriver class is our main method and establishes that we have a singleton tester and a GameService instance. It also makes sure that the instance is the only instance that occurs. The singleton tester class is dependent on ProgramDriver as is shown in the UML. This application uses multiple programming principles, these include inheritance, encapsulation, and abstraction. Inheritance is shown through the relationship between Entity and Game/Team/Player. These three classes all inherit from Entity. Within GameService we see encapsulation, this is shown by its limitations of data so that it can only be used in certain ways, thus maintaining a level of built in security. Abstraction is shown by our application only showing what needs to be shown and limiting what needs to be seen.

**"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 is considered to be more user-friendly. It is also great for high-power use cases and offers great performance on large-scale deployments. Works well across Apple products as they all use the same general OS. The main weakness of Mac is that the OS is different from other options and people that are not used to it could have problems with it. Mac does have Servers that can be used, however these servers are mostly for business and or production services. As far as licensing costs you do need to Purchase MacOS.> | <Linux offers a wide variety of pros. Some being that it offer less exposure to malware and can run with low system demand. Another benefit is that it is open-source and doesn’t need a license. Because of it being open source it is a great option for aspiring or even proficient developers it even has development tools built in. The disadvantages of it is that it can be harder to use for users, it does not offer a lot of professional support. The use of it is also not as common so applications are often not as well kept for Linux as other OSs. Linux has a ton of server options but one example would be ubuntu. In terms of ubuntu it is free to download.> | <Windows is probably the most common OS that most users use, because of this the tech support and overall compatibility of the OS is above all others at least in my opinion. It offers more options in terms of software and even hardware. However, it also has more known vulnerabilities but because they are known they also usually have fixes our at least ways to avoid them. Windows has its own Server options and these options do need to be paid for and need a license.> | <Mobile devices offer more portability and a lot of times also offer a lot of variability in terms of controls as some devices offer touch and even touchless controls. Because these devices are portable their uses offer a lot of options. The downside of them is that they tend to be small and also because of the size sometimes overall power of the systems can be reduced because of them. Mobile devices do not have a server option as they are not typically powerful enough for hosting servers, however, they can be great for users to use the service while one the go. |
| **Client Side** | <In terms of cost, the cost of Mac based systems tend to be higher but the skill level of using them most of the times is a lot lower than other OSs. Therefore the time needed to gain experience in the OS is lower. When making this platform we need to ensure that there are different versions made for each platform so that they can function similarly but maintain its compatibility with each platform. As far as browsers go Apple can use any of the chromium based browsers aswell as safari. As far as popularity the two most popularity used to be firefox and safari, but since its creation chrome has become really popular as well.> | <For Linux based systems the cost of it can be lower because of the lack of a need for a license. However, the time and skill of it tend to be higher than a lot of other OSs. For Linux, some kind of open source files or even a forum document could be helpful so that Linux users can setup and run the application as its intended. For linux just like windows can use most browsers that other OSs can with the exception of Apples Safari browser. For popularity, it would probably be firefox as it is the default for most linux distros.> | <Windows probably offers the best middle ground in terms of price to skill level requirements. It also probably would not take as much time to learn it as there are a lot of resources out there that can off a large amount of help to people. Windows would need the application to run with whatever version of windows is the latest for instance we are currently on windows 11 so the application needs to updated and ready for windows 11. While it needs this it also needs to be able to run on other platforms. Now for browsers on windows windows can use most browsers that the other can, with the exception of safari again. The most popular would be edge which is the default , then followed by chrome and firefox.> | <Mobile device are probably the lowest out of all of them in terms of price. That being said sometimes because of the size of the devices the user experience and therefore the skill level of using it might be higher than average. As far as browsers go for mobile devices a lot of them are available, it mostly depends on what phone you are using. For apple phones you have safari available and most people are probably ok with that. However, chrome is still a popular option. For android phones you can use a lot of browsers but the most popular are probably chrome and edge because of the familiarity with those platforms on other devices. |
| **Development Tools** | <IDEs such as VSCode and Xcode are probably the most used platforms for Mac Users. A lot of developers may use built in code to make their code work this makes things quicker and more efficient. Developers may also use the run function to make sure that code works as they are working on their code rather than having to fix a bunch of stuff. | <Linux offers more range in terms of what IDEs you can use. One example of them is VSCode. It is also open-source adding a lot of coding freedom. Multiple teams could be needed when using Linux this is because Linux has a wide range of freedom meaning that a lot of languages could potentially be used and not all programmers are going to skilled in all languages. So having teams who are specialized in certain languages could be beneficial. As far as coding goes Linux is and has a lot of support for open-source coding which is typically free. This means that built in functionality is probably limited to what IDE is in use rather than the actual system as well.> | <Java, and Python are common languages and usually can be used in IDEs like Visual Studios and Jetbrains. For windows we could again potentially benefit from having multiple teams this is mostly because just like with linux there are a lot of options in terms of IDES. So, having a team that is skilled in one specific platform could boost productivity and lead to a faster release. As far as costs go there are different options that could range from free to having a charge. Just like MAC, developers will likely use built in functionality and running to test and run code with these built in functions coding could become more efficient and coding could happen quicker.> | <Coding work for mobile platforms typically are not done on the mobile devices they are typically done on other platforms then adapted to work on mobile devices. For these programs all platforms have access to Visual studio and or Java. Common languages that are typically used are java and python. > |

## 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 best recommendation would Windows for Operating platform quality. This is because it offers the most variability while also offering a lot of support and can often give users a more natural experience. Especially for applications that would likely benefit from a web-based application.>
2. **Operating Systems Architectures**: <The client is trying to make a game and because of this often uses JavaScript/HTML/CSS. These three can be used on all of our current OS options but because the OP recommendation is already Windows the recommendation would again be windows since it benefits from both of the requirements we want so far.>
3. **Storage Management**: <A lot of the times you can benefit from different types of storage types. Depending on the use case. For instance for servers a NAS is often used because it can be setup in ways that even if a drive fails most of your storage would be saved in a another drive. However, for personal use a regular hard drive or a more modern m.2 drive are of better use.>
4. **Memory Management**: <As far as memory management it could be said that at least for desktop/laptop computers they use RAM which come in different forms but function mainly the same. For most use cases having more available RAM is typically better. So for the servers that would be needed to run the games online functionality it would be recommended that those Systems/servers have a good amount of RAM in them.>
5. **Distributed Systems and Networks**: <Distributed systems can be used to house your main servers this allows for failure protection in the event that one of the servers fails you still have other servers running that can maintain the game and keep it live. This can also allow for different servers to run specific things so that each server can maintain a low utilization and run at its absolute peak performance.>
6. **Security**: <In terms of security windows is a good option because of the known issues, because of the known issues there are also known fixes. That being said if you’d like the overall best protection it is often regarded that Linux has the lowest threat level. So if you have people that are experienced in linux and can run it efficiently it is a great option.>