

The Gaming Room

# **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 | 01/23/2022 | Jacob Mueller | Initial Application, limited comments |

**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)

Citing a lack of developer resources, The Gaming Room is looking to outsource software development efforts to a separate entity. Aside from that issue, problems needing a solution are an ability to play the game either as a single multi-player team or as several multi-player teams, game and team names need to be checked for uniqueness, and only one instance of the game can exist in memory at a time. First off, we can handle development efforts for as long as they’ll have us, I can add some code in to only allow the game to run if there exists at least one team and at least two players, there are options available to allow for the game to reference existing team names and only allow unique game and team names, and finally the unique operators that will be used after the previous steps are accomplished will help to have only one instance of the game in the memory at one time.

## [Design Constraints](#_2et92p0)

The concept makes sense on paper, see UML diagram, the service itself will need a way to track games, games will need a way to track at least one team, and teams will need a way to track at least two players. Troubleshooting and debugging the system in early development will be time consuming if any issues are present. The development team isn’t the most experienced and will need some time to produce the services, but with no time or budget constraint evident, that may not be a problem.

Putting the game up for use on different devices through a web-based format may require some testing before release to make sure that the system works with multiple devices and operating systems.

## [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)

As stated above the game service class will have to check with the game class, the game class with the team class, and the team class with the player class. At least one game needs to exist for the service to have something to do, and likewise at least two players (“each team will have multiple players assigned to it”) to a team and at least one team to a game. The Entity class is the parent class that the three builder classes (so named because they make up the necessary parts of the game service class) will derive their properties from. Each of the builder classes (game, team and player) will have unique id’s and names which will provide a unique quality to each instance of the game service. Furthermore, as you can see in the upper left corner of the diagram the program driver will use the user input to populate the games, teams and players and then for any games instantiated, the singleton tester is used to restrict the instantiation to a single object. So, only one game exists in memory at a time, working as advertised and per document requirements.

**"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.

Requirements

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.

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.

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** | Everything I’ve seen online says that out of the choices available a good mac host will be hard to find and will not offer the best experience for hosting or visiting. That being said, if the macOS is what a person is used to it may be the best fit, the community behind the mac is very loyal and pleased with the system.  This may be a less price-efficient course in the short run but could pay off in the future for handling other apple products. | Being free use and open source, linux is designed to be a more accessible operating systeem. Many users will already have experience from setting up the system already and will provide helpful resources on optimizing server hosting.  Cost concerns are minimal, upkeep and purchasing are very manageable since Linux is free and open source, with no mandatory updates like other os’s. | Windows is THE gaming OS, more people are familiar with and use this system for gaming than other systems. Everything is put in place initially for the user so there is little to no confusion.  However, there are issues with how inelastic the system is with updates, and the cost for hosting is dependent on Microsoft, so it may not be the cheapest means for server hosting. | Obviously due to size the mobile device will not have the ability to host the servers for the game that the other operating systems possess. Some outsourcing of server hosting will need to be developed alongside the software itself. Which may bring you back around to one of the umbrella corporations behind windows, mac, etc. |
| **Client Side** | Gaming is not the mac OS’s strong suit, and it isn’t advertised as such either. For a large release of a web-based game, macOS offers little outside of a dedicated user base. Once again, this could be a better option if looking at how easy it will be to pivot from the mac onto other apple products like the mobile device line. | Tauted as the most efficient operating system amongst many developers, this may be the way to cut down on the issues of development time and cost. But, because of the setup that is required on the part of the user, this may not be the most user-friendly option for gaming. | Windows will be the easiest system to work with as far as marketing, outreach, and development are concerned. It is a pillar of the community, it is often either the initial system or most utilized system for users, especially for gaming. Having the mandatory updates that come with the system could get in the way of the games success, and there is no way around that issue since it is a decision made by microsoft. | With the available resources the production cost and time could be minimal, but that requires a team comfortable with the languages and tools available. Depending on the nature of the development team there could be issues when transitioning from one device to another during production. |
| **Development Tools** | Sublime, NetBeans and Visual studio code are all well-known and frequently used development tools used across other operating systems, I’ve seen many stories about transitioning from apple to Linux seamlessly. So, it will be easy to pivot from mac development work to another system, especially those in the apple product line. Python has been a recent rising star which hasn’t been over looked by developers using the mac os, python has a ton of support and is a very intuitive language. Going through line by line makes it easy to find and fix errors, but it can lead to a lengthy process of trial and error if the code is not efficiently written. Swift is another language, very similar to python, that apple products have been using and it is lauded as a safe and effective language focused on scalability and outreach to other devices and platforms. | Linux can also use sublime and visual studio code, but due to its uniqueness, there could be some issues in pivoting from a more linux specific IDE to a windows or mac IDE. Although it is very user friendly for customization, there isn’t a strong support team to help users or developers out in a pinch. As stated previously it is a free and open source platform that will make it a lot cheaper to work with and it can be easy to pivot from linux to mac. LInux uses C as a base programming language, which means that it has no problem handling simple tasks and can easily blend with other offshoot languages popular right now. | Visual studio code, all editions, have made it a great choice for developing code in a variety of different languages. Huge licensing price tag, if I wasn’t in school and getting ide’s and discounts I would definitely not be able to afford running everything on windows. But you do get what you pay for, lots of support, focused updates, easily accessed user interface. Its programming language being primarily C makes it easy to blend with java or python depending on developer preference and eludes to the fact that there is a ton of support that has built up over time, so no problem will seem too big. | Many mobile devices are derived from a parent OS (iOS and macOS) and there are tools available to develop the software on a mac and then port that over to the iOS. Also, there are a myriad of IDE’s that will work with specific languages and allow for quick production of mobile applications. Important to remember that those parent os’s aren’t always cheap when licensing products, and the mobile device focused tools may not be the most developer friendly depending on the team. To me, java is the mobile device android language. It is widely used among the android app developers, so it could be a great place to start since the game was on android first anyway. Java is not the friendliest language and can be tough to introduce a new team to, but if the development team has experience with pointers and safe operating, there should be no issues. |

## 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**: I would recommend the windows OS and the windows suite of products, covering pc, gaming systems, and mobile devices; the OS for pc should be the best place to start since they have the longest lifespan to date and will provide a good jumping off point for development.
2. **Operating Systems Architectures**: There is a layered design to the operating sytem architecture. It is a great place to start and could be pivoted from with relative ease. Basically, since the program will be following a single instance and check for certain variables from game service through to the player class, it will be best to utilize the layered aspect of the software as seen in the UML diagram by using a layered design perspective. The layered architecture was cited in the reading as a mobile convention used prominently in ios, which means that it could be a beneficial architecture to use as the teams move forward in developing the game for other mobile devices, and then retroactively across the other apple products.
3. **Storage Management**: Microsoft windows has come a long way and has a lot to offer in terms of storage management through the hardware in single disk desktops and external storage options. Shortest seek time first scheduling will be the best way to manage how information is stored and accessed.
4. **Memory Management**: Microsoft uses paging and will take some pages from RAM, when ram is full, and apply those pages to the hard disk to free up valuable space for the RAM. Not the most optimized system, but it will help to make the process more efficient. Multilevel directories would be a great option for dividing the workload of holding the library of images used in the game and typically works well with many users in operation.
5. **Distributed Systems and Networks**: Microsoft teams is a great example of how the distributed system could be used and would operate. It has come a long way through prolonged use due to the recent covid quarantines. Likewise, the networking capabilities have been optimized since more people have had to work from home, now would be the best time to start distributing a game across multiple platforms, if there is a viable way to host servers for users.
6. **Security**: I trust the microsoft security programs, they are built in and have been built up to even include some system diagnostic reporting and has always been a great resource for detecting and protecting from malware. Users with a windows device have the security built in already when the hardware is obtained.
7. =====================================================================================
8. Bibliography for references used to complete code.
9. <https://stackoverflow.com/questions/5125107/java-class-cannot-be-resolved-to-a-type>
10. <https://stackoverflow.com/questions/29870890/java-the-method-is-undefined-for-this-type>
11. <https://stackoverflow.com/questions/24117713/editor-does-not-contain-a-main-type-in-eclipse>
12. <https://stackoverflow.com/questions/16225177/error-selection-does-not-contain-a-main-type>
13. <https://stackoverflow.com/questions/15398703/exception-in-thread-main-java-util-nosuchelementexception>
14. Table sources
15. <https://digital.com/best-web-hosting/mac/>