

<Name-of-Software-Application>

# **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 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this revision> |

**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 software design problem that is presented would be the need to manage a list of active game in the game engine. The GameService class must be able to add and retrieve existing games by both id and name and then return the number of games active. The solutions involve implementing a singleton in the GameService class while maintaining a private constructor so that only one instance is active. In the GameService class there are three public methods available for managing the games, that include addGame. getGame(long id) and getGame(String name) the solution proposed meets the requirements of managing a list of active games. It is important for the client to understand the importance of the GameService class being a singleton.

## Requirements

*Technical requirements*

*-able to accommodate multiple users*

*-compatible with multiple platforms and devices*

*-ensure reliability and maintainability by using best coding practices*

*Business requirements  
- allow multiple users to play games simultaneously*

* *Store and retrieve game data*
* *Track game scores and statistics for each player*

## [Design Constraints](#_2et92p0)

design constraints for developing the game application in a web-based distributed environment would include scalability and the ability to handle multiple users, as well as taking into consideration network latency and designing the game application to minimize the impact, and ensuring the compatibility of the application to be designed and work on multiple platforms to ensure functionality for a broad user base.

## [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 UML class diagram provided below the game, team and player class have a many to one relationship with the entity class that they inherit so that the methods defined in the entity class can be used without having to redefine them in the separate classes. There is encapsulation found in the Game class as it has private fields used for both its ID and Name to hide the internal workings from the other classes. Yet has public getter methods accessible to the other classes. There is abstraction found in both the Game and entity class only exposing methods used to interact with the game such as adding players or getting the games name.

**"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** | Being popular with developers there are a lot of resources and support available, as well as having many built in tools for web development making it easier to set up and configure. However they can struggle with compatibility requiring additional configuration to properly run | Linux has the reputation of its security over other options and is highly customizable in comparison. A downside of linux being that it may need a higher level to technical expertise and slightly more complicated than other avenues however it is highly customizable in comparison. | With it being so widely supported is backed by Microsoft and many third part providers and can be helpful for resolving issues relating to hosting a web based software application while having a user friendly interface. While also having a reputation for being secure it is less of a target for hackers, but could prove to be more costly than other avenues as Windows licenses have been shown to be expensive. | Due to them having smaller screens it can be challenging to properly display complex web applications. The limited processing power also can prove to be an issue for Mobile devices, however them being always connected to the internet could provide users quick access to the application from anywhere they have internet access. These devices have also been proven to be more vulnerable to security threats which could compromise web based applications hosted on them. |
| **Client Side** | Supporting clients on mac could prove to be costly including software licenses and development tools, however its been a longstanding platform and previous iterations designed for Unix systems can be ported with ease. Mac having a strong focus on user interfaces being used with ease could prove to take longer to design with proper usability which could increase development time and cost but would provide a better user experience | The cost of using linux to support clients would presumably be less than other avenues being as it is open sourced so there would be no licensing fees. However there are still the cost for development tools, there is a lot of customization regarding linux so that could prove to be more time consuming but provide a seamless user experience. | Software development considerations for using windows would include the need to ensure compatibility across the different versions which would take longer than other options to ensure proper usage. Being as popular of a platform as it is security may be good however it tends to have more than other options so it may be more costly to take these considerations into the software. | Because of the variety of mobile options development time may take longer as the application needs to be optimized for the different options. Similarly this would be costly however it would vastly increase the user base with its ease of access. |
| **Development Tools** | C, Swift and C++ are the languages used on Mac. The popular IDEs that are used in conjunction would be Xcode, Visual studio code and eclipse many of these options support multiple programming languages. | Programming languages used with linux include PHP Python, Java, Javascript, HTML, and Ruby with a wide variety of IDEs used for the large number of available languages including Eclipse Visual Studio Code, Atom, Sublime Text and NetBeans. Other tools that are used for deploying software on linux include MySQL, or PostgreSQL regarding databases. | Relevant programming languages used when building this type of software for windows devices includes C#. C++, Java, JavaScript, and python. With popular IDEs including Visual studio, eclipse and Intellij IDEA that provide a range of features beyond code editing including version control and debugging. Other tools include Microsoft .NET framework which includes a library of prebuilt code for common tasks and GIT which is a popular version control system to maintain and manage code. | Relevant programming languages used when building this type of software for mobile devices includes HTML CSS and JavaScript, using popular IDEs such as Xcode, Visual Studio and Android Studio. There are a variety of other tools that can be used such as Mobile emulators to be able to test mobile applications for multiple devices on one device. As well as many different Mobile App frameworks that can be taken into consideration such as React Native or Ionic when building mobile applications. |

## 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 recommend the use of a cloud-based platform on Linux on the server side as they provide reliability and flexibility moving forward to other computing environments these would include Amazon Web Services, or Microsoft Azure. This could cut costs regarding maintaining physical infrastructure while providing the ability to deploy to Windows, mac, Linux, and mobile devices. With the easy implementation and use of the application on a variety of platforms it will provide the user with a variety of options to use the application on as the customer requested and make the overall experience more accessible. Because of the use of a cloud-based platform it makes it significantly easier to further scale the application and its capabilities in the future if needed, easily adding, or removing the resources based on what is required at the time. It is rather easy to find support and resources for Linux operating systems and the software packages and tools required for the application. Because Linux is free and open source it has a large community that is known for being secure and stable regarding hosting applications. The flexibility provided by being on Linux is second to none, it is highly customizable allowing easier optimization and performance of the server and can be custom tailored to the requirements provided for the application. Linux is known for its command-line interface which is often found to be more powerful than its competitors as well as more efficient than other graphical user interfaces such as those by Windows and Mac which would allow for a quicker execution of commands and scripts. On top of being a cost-effective option to use Linux to host this program it provides a strong support system based on the popularity of Linux and number of resources at hand, while also being a reliable and flexible choice.
2. **Operating Systems Architectures**: Using the previously mentioned options could be a microservices architecture so that it can be fragmented and deployed separately ensuring flexibility and scalability going forward. Microservice architecture allows each service to be developed and deployed separately which would enable faster development cycles as well as encouraging a more efficient utilization of the resources needed for the application. Using the recommended Operating platform of Linux, this microservice architecture could be implemented using containerization technology such as Kubernetes or Docker, which are the most popular. Using Docker moving forward would allow the development team to package the application and the dependencies in these containers which as stated before are separated allowing a variety of uses independently including testing, development and deployment while being isolated from underlying infrastructure which would enable easier management of the services independent of one another. While Docker allows the containment, the other option can be used in conjunction and is a container orchestration platform. The uses of a container orchestration platform include the automation of deployment and even scaling and management of these containers that are customized based on the requirements. These two options will provide an application that is easily scalable and flexible to changing requirements as needed. Because of how Micro service architecture separates the services, it simplifies the maintenance required by minimizing the risk of introducing errors and containing the work done as well as providing a faster time to market by speeding up the development process and in turn saving money for the customer and providing a smoother development process for the team. Another invaluable thing to consider regarding microservice architecture is the ability for each of these contained services to have different frameworks as well as programming languages which could provide more flexible options as well as the use of specialized technologies for each service which could provide a better performance and smoother use of the application for the user.
3. **Storage Management**: An appropriate storage management system to be used with the previous recommendations would include using a distributed file management system such as Network File System, which like previous recommendations would encourage and ensure easy scalability and availability while able to store large amounts of data while being supported by most operating systems. NFS is also known for its high-performance allowing fast and reliable data access going forward, taking these things into account this would be my recommendation for the application on Linux. There are a variety of options available for Storage management however Network file systems would make an appropriate choice because of its scalability as well as its high availability, the importance of having the data accessible in the event of server failure could not be understated where maintaining this data is important for a smoother user experience.
4. **Memory Management**: The recommended operating platform uses memory management techniques including using virtual memory which can allow for a more efficient use of the memory resources and minimize memory limitations. While also including kernel level memory allocation that ensures all the processes can be run efficiently without causing issues for other processes. This technique minimizes memory limitations while preventing out of memory issues when using the application. Because we chose Linux as the operating system for the server side there are a few perks, as mentioned the kernel level memory allocation is important but there is also a technique unique to Linux called the “oom killer” which would prove to be useful as its name implies it terminates processes that are consuming a large amount of memory and in turn freeing that memory for other process which can prevent systems from crashing or becoming unresponsive. Another unique feature to Linux is a kernel parameter that helps manage virtual memory allocation called “Swappiness” which would help optimize the memory usage while preventing out of memory errors that could happen otherwise. Linux is not alone with the kernel level memory allocation but because of these features that can be implemented it would be an optimal choice to improve the overall performance and reliability.
5. **Distributed Systems and Networks**: Knowing that the client would like to communicate between various platforms this can be accomplished with distributed software and using techniques like microservices architecture where each component provides a unique service and can communicate with other components over the network. It is important to consider the network connectivity is reliable and secure however the system should be able to detect failures and be able to handle issues by switching to a backup service or using a load balancer to prevent applications overloading any one instance depending on the issue. An important thing to consider would be the introduction of redundancy and other fail over mechanisms which would prevent disruption in the event of service failure as well as ensuring high availability. Using the previously recommended microservices architecture would work well to communicate with various platforms as it is both flexible and scalable while being composed of smaller services that can communicate using a standard communication protocol. Another asset that should be considered is the effectiveness and ease of use regarding the monitoring and management of these services, which can be done with a variety of tools to monitor the network traffic, server performance, and as well as the individual performance of these microservices to ensure reliability and availability for the application and its users. By implementing the right tools, system administrators can ensure a smooth and efficient experience while minimizing downtime and other issues that could arise. Management tools would also prove useful in managing the containers previously mentioned in the microservice architecture, that would include deploying updates, managing the configurations, and even the troubleshooting.
6. **Security**: Security is a must and there is a multitude of techniques to protect user information on and between the various platforms, this would include regular updates and patches to the software, as well as including encryption to the data between the platforms. Authentication for the user can also be an option to ensure only the designated user can access the system, or the implementation of access control to achieve the same, which Linux offers a range of tools for. Linux tools for authentication and access control include Pluggable Authentication Module and SELinux which are unique to this operating system. Network security should also be taken into consideration and firewalls and intrusion detection systems should be in place to ensure the safety of the user data. Security implementations can be done in a variety of ways and it’s important to use a multitude of options to ensure there is a fail safe in case of an issue. Regarding the Distributed Systems and networks, it should be noted that in addition to the monitoring and management tools there are logging and reporting tools which would be invaluable in the safety of user data and general security, the use of these can record the events and the actions of the system not only would this be useful in troubleshooting issues but could also optimize the performance of the system. There should be clear policies and procedures taken when managing the system to ensure consistent and effective use, minimizing issues such as downtime. Encryption is an important technique that should be considered by the client, the implementation of this would protect user information while in transit between the different platforms as well as the data on the server which can be done using technologies like HTTPS, or SSL/TLS. By incorporating a variety of these tools and techniques the data of the user will be secure and ensure the protection of the information held by the system.