

<Draw it or Lose it>

# **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 | 05/20/2022 | Seth Hamrick | Executive Summary, Design Constraints, and Domain Model updated with initial request from The Gaming Room and current information on project. |

**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 Gaming Room has an existing game application available on the Android platform. They are now seeking to develop a web-based game application to serve multiple platforms. The Gaming Room requires a very specific environment to develop the web-based application as desired. This environment requires that only one instance of memory be in use at a given time, each game and team name are unique, and that each team and game will allow for multiple players and teams, respectively. Java programming language should facilitate many of the environment requirements.

## [Design Constraints](#_2et92p0)

There are two large scale design constraints with this project. The first of these constraints is the application must be accessible on multiple platforms. The second of these constraints is that any development will need to be compatible with any existing or future developments from The Gaming Room. These constraints may affect the proposed solution to the environment constraints, as well as the development process. The remaining major constraints around the project include the technical constraints surrounding the hosting environment. The first of these constraints is that only one instance of memory may exist at a given point in time. This is likely the largest constraint surrounding the environment and may require a specific design pattern. The next technical constraint associated with the environment is that each game and team name must be unique. The final constraint surrounding the environment is that each game should allow for multiple teams to participate as well as each team should allow for multiple players to join. These environment constraints should be met with the use of the OOP Java.

## [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 ProgramDriver class contains the main() method, which is the first method that will be ran within the program. This class uses direct association with the SingltonTester class which contains the method testSingleton(). This method will serve as a test for the program. The Entity class is the parent of the Player, Team, and Game classes. That is to say that each Player, Team, and Game may also be considered an Entity. The Entity class contains protected variables id and name which will be an attribute for an existing Entity. The Entity class contains functions that return the current Entity’s id and name. The Entity class also contains a function to print the id and name as a string. The Player class is a subclass of Entity and it contains no additional attributes. However, the Player class does contain its own constructor and override method to print information as a string, which we see throughout the remaining subclasses. There are 0 to many associations between the Player and Team class indicating that there may be 0 or multiple players associated with a team. The Team class is also a subclass of Entity but contains an additional attribute that is a list of players. The Team class contains a function that will add a player to its list of players. This function will require that the player’s name is unique before adding. The Team class also has 0 to many relationships with the Game class indicating that 0 or multiple teams may be associated with a game. The Game class is a subclass of Entity and contains the additional attribute that is a list of teams. The Game class is very similar to the Team class containing similar methods, except in the Game class a unique team will be added with the addteam() function. Finally, the Game class has 0 to many associations with the GameService class, again indicating 0 or many games may be associated with a GameService. The GameService class is a singleton class which ensures that there will only be one instance of GamerService ever created. This may be noted by its static service variable, private constructor, and static public method getInstance(). The GameService class also contains private variables including a list of games along with variables for the next player, team, and game ids. The GameService class contains methods that return the game by a given id or name, the number of games active, and the next player and team id.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac offers options for server-side hosting but are not known for their ease of use and integration across multiple platforms. There are some expensive costs associated with this option, which includes the need for Apple hardware for development. | Linux offers itself as a free open-sourced platform that would allow for use on the server side. Linux is one of the most used website hosting platforms. The most concerning aspects of Linux would be the need for a high degree of technical knowledge in addition to its more error prone open-source format. | With multiple versions for both personal and professional use there is likely a version that will fit our needs. However, Windows does not offer the most secure option available and has some licensing costs associated with it. | Mobile devices are capable as acting as a server, but are likely not the best option for serving multiple clients at a time. |
| **Client Side** | Mac is often a top selection for web development due to its ease of use and versatility. The cost associated with development on Mac is significant and requires Apple hardware. | Linux a free very powerful and capable option in the right developer’s hands. However, development in Linux would require a development team well versed within the system. Additionally due to the open-source nature of Linux errors may arise and there is no technical support available. | With costs between that if the free Linux and more expensive Mac, Windows offers a more affordable closed-source option. This may come with some trade offs at times though, for example Windows is likely one of the slower options available. | Mobile development may offer a faster and cheaper process; however, this may come at a cost of raw processing power. Additionally, this development method would likely require the most frequent maintenance and upgrades. |
| **Development Tools** | JavaScript and Swift would be the recommended languages for development. Xcode is available for free download on Apple hardware or and IDE such as Eclipse could be utilized. | Python and C++ have been the preferred languages for development on Linux systems. Python is a very powerful language for web applications. There are several IDEs that could be selected and depending on the language the preference may shift. Intellj IDEA would be an example that could be used and is compatible with JVM languages. | Visual Studio is the staple IDE used on the Windows systems. Visual Studio supports several language options including C#, C++, JavaScript, just to name a few. | AgXcode or Eclipse may be used as IDEs. I would recommend the use of Java for its versatile cross-platform use. |

## 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 use of Linux OS for the server side of the web application. This is because Linux is a very stable and secure OS that has been proven as a reliable server throughout its’ use. Linux servers are less commonly targeted by hackers as compared to its more mainstream counterparts. Additionally, Linux is an open source and free to access server option as compared to the other options on the market which have more cost associated with them. Linux servers also provide methods of scalability as the needs of the application grow. Due to the free and open-source nature of Linux the one major pitfall it possesses is the need for user’s technical skills and lack of technical support. However, errors are rare occurrences within the Linux OS as it is very simple in design and improved through its open-source nature. Additionally, the technical requirements may be combated during development with the use of a containerization platform such as Docker for easy development, integration, and implementation.
2. **Operating Systems Architectures**: The Linux OS has a monolithic kernel design. This means that all the OS services reside within the same memory area, or more easily said all the OS services reside in the kernel space. This is a powerful design for ease of implementation and hardware access. Being located within one space creates a heavy dependency though and if there is a problem with one system component it may render the rest as useless. In addition to Linux using a monolithic kernel design it also utilizes a modular design. This modular design helps limit the amount of memory and usage of the CPU by only loading what is necessary for the current functionality request of the system. A modular design contains modules with the code necessary for some system functionality. These modules may then be unloaded and loaded as needed by the OS to match the functionality requests. In addition to this “only take what is needed” approach these loaded modules may then be easily loaded into new instances when needed. Overall, this modular approach helps the management of memory, CPU usage, and speed of the system. The only aspect that we haven’t touched on within this design is the security benefits. With the modular design this also increases security as it limits the access to certain hardware at given times within the system.
3. **Storage Management**: The storage requirements for this project are currently not very taxing. The current request image library is approximately 1.562 GB in total. Including the library, application code, and perhaps user and game information if the client requests. 8 GB is likely enough to begin running the application but also allow for slight expansion of the application if required. This storage amount could easily be achieved through the local server hardware. This would be the recommended option to begin with if cost were a driving concern. However, the speed of access and utilization of the server may become overwhelmed as traffic increases or as the game library expands. These server demands may need to be considered heavily and will be further discussed in the memory management section. The use of a cloud-based storage would ease many of the demands on the server, increase security, and improve recoverability at a slight cost. If a cloud-based storage option were to be desired it could be implemented from the start of the project, the initial startup cost would reflect accordingly but the avoidance of future headaches may be well worth it.
4. **Memory Management**: Linux memory management is rather complicated and has very configurable settings. However, one simple concept with the Linux OS memory management is that everything goes through the system’s RAM. Linux utilizes the RAM and cache of the system and CPU to allow easy access to recently utilized items of information. This design places a large emphasis on the hardware specs required for the RAM on the server side of the application.

The use of cloud storage would help combat this RAM demand of the server. This due to the cloud-based storage being accessed directly using an API which could be implemented within the code of the application. This would by-pass the need for the server to utilize large amounts of RAM to serve multiple games and users across the network. Additionally, the cloud-based storage would increase the access of the storage contents as cloud storage often provides concurrent servers so that if one is to go down another is often available to mitigate the downtime. Security would also be improved with the use of a cloud-based storage option along with the ability to recover lost data. I believe that the benefits of both the storage and memory management in a cloud-based design fully justify the additional cost for the requirements of this application.

1. **Distributed Systems and Networks**: The distribution aspect of this application is extremely important as the client seeks to serve multiple platforms simultaneously including mobile and web. This challenge will be most easily achieved using an API such as the RESTful API. This API is one I have selected to point out as it contains many of the features we desire from an API for this application. The RESTful API as many other APIs places most of the workload on the server as opposed to the client overall improving the portability of the application. The RESTful API uses HTTPS protocol which is encrypted HTTP to send and receive requests and responses between the server and client. Utilizing HTTPS allows this API to communicate between a server and all the major platforms the client is looking to target including, all mobile users, Windows users, Linux users, and Mac users. The network with this method would solely rely on the internet connection of the user and the server. The server’s internet connection we obviously have some relative control over, so the remaining half simply places the responsibility of the connection in the hands of the user’s internet provider.
2. **Security**: Linux OS has features that increase security and access to specific portions of the system that will be useful on the server side of the application. These features include password authentication, discretionary file access, and the ability to perform security audits. Password authentication is often the first line of defense in many systems. To increase security, I suggest the use of a username and password authentication for users within the web application themselves. This will add a layer of security for anyone attempting to access the server through the application code. To ease the mind even further I would highly recommend some form of dual authentication such as a one-time code sent via text or email to add yet one more layer of protection. Linux additionally allows discretionary access to the file system depending on the user’s rights to limit unauthorized access to the system. Finally, Linux OS provides the ability to perform security audits which may prove useful as scans, vulnerability identifiers, or compliance testing. In addition to the Linux OS security features, I highly recommend the use of a firewall to add security at the network level to help mitigate the need for security at the OS level. A firewall would facilitate the monitoring, flagging, and prevention of threats located within the network traffic. That covers much of the server-side security but now we must consider the client-side security. First and foremost, the client-side security will begin with the users respective operating system and its’ intrinsic security features such as the ones described for Linux. Additionally, if the design of the application is using an API specifically the RESTful API, the user will connect to the server via an internet connection through HTTPS. HTTPS is an encrypted form of HTTP to increase security. From here additional security features the user may have such as antivirus or antimalware software applications or even a firewall may further protect the user from threats. Security is an important aspect of this application for the clients, and for this reason I would advocate security inclusions outside the OS, third party adjunctive tools such as antivirus software or firewalls, and dual authentication for application login. Beyond those discussed I would also recommend a logout timer for inactivity and application user education. These two additional factors will help to mitigate unauthorized access and misuse of information that may lead to a breach in the system. Security needs are ever growing, and future security monitoring would be required such as the Linux provided audits previously discussed.