

# CS 230 Project Software Design Template

Version 1.3

## 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 | 09/18/2024 | Armando Laguardia | Summary, requirements, constraints, and domain model description added. |
| 1.1 | 10/05/2024 | Armando  Laguardia | Evaluation section. |
| 1.2 | 10/14/2024 | Armando Laguardia | Recommendations. |
| 1.3 | 10/15/2024 | Armando Laguardia | Reference page added. |

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 is seeking help from CTS to create a new game. This game needs to be web-based and cross platform. However, it also needs to be based on their current game, Draw it or Lose It.

## Requirements

The game needs to be cross platform, specifically web based. Meaning that it should be in major stores for different operating systems such as the apple store and google store, as well as running on the browser of a computer. The game should also have similar features to that of their previous game Draw it or Lose it. These features will be further addressed below.

Two lists (functional and non-functional) will be used to better portray the clients’ requirements:

Functional Requirements:

1. Cross platform availability, including mobile.
2. Web based
3. A game will have the ability to have one or more teams involved.
4. Each team will have multiple players assigned to it.
5. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.

Non-Functional Requirements:

1. Game should perform well.
2. Game should adhere to proper security measures to protect data.
3. 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)

Making the game accessible from a web browser.

* Solution: This is tricky, as most web accessible games have migrated to clients. However, there are some frameworks and technologies that could aid with this problem. If this isn’t feasible, a suggestion would be having a web page for the game from which you can download the client or app.

Making the application functional on different operating system including mobile ones.

* Solution: Use cross platform functional frameworks while using a test-driven approach to verify functionality.

Making sure that only one instance of the game exists in memory at any time.

* Solution: Implement singleton pattern and unique identifiers.

## [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)

UML description

"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.

## The application starts with a main class named ProgramDriver; this class is what starts the creation of the games and all other assets. This class uses the GameService class to instantiate a service from which we can access all the methods related to the service. The GameService class itself is implementing the singleton pattern so that so only one instance of the class can exist in memory at any given time. There is also a directed association relationship between the ProgramDrive class and the SingletonTester class.

## 

## When taking a closer look, we see that there is also an Entity class; this class is then inherited by the Game, Team and Player classes, which are all types of entities. This relationship is depicted by the blank arrowhead line.

## By having different classes be an entity, we are also implementing polymorphism. Additionally, the entity class won’t be directly instantiated; therefore, it could be counted as an abstract class. From left to right, there are also association relationships, starting with the GameService class and ending with the Player class. The notation in between the lines showcases that the classes can have 0 to many of the objects to the right. For example, a game service can have 0 or many games in memory, a game can have 0 or many teams in memory, and a team can have 0 to many players in memory.

## 

## Encapsulation is also present throughout the entire diagram; the easiest way to pinpoint it is by noticing all the visibility modifiers in every class. The fields are all private, and most methods are public except for the constructor (singleton pattern). Then the GameService class uses proper getters to access the information.

## 

## The way the app works is by having the logic fall under the game service class, then creating an instance of it in the main class from which we can access all the required information to build and run our game.

## [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 OS is good for development. It is not often used for server hosting.  However, it comes with Apache built in which can be used for hosting. It is based on Unix, which means is stable for development and the supports Docker. | Linux servers are stable, safe and flexible. It is Open-source and highly customizable with options to use Apache and Nginx. Additionally, it is widely supported and low cost. | It is a popular system which can be easy integrated with Microsoft tools such as SQL Server. However, licensing costs can be high and can be less efficient with high traffic. | Mobile devices are not typically used for server-side hosting. They are portable, efficient and good enough for local processing. However, limited in power and resources. The limitations make it not suitable for large-scale server hosting. |
| Client Side | The client side can be costly due to how expensive Apple devices can be. However, the high quality of apple devices also means that there is high performance. The required knowledge is web technologies such as HTML, JS and CSS as well as Apple’s development environment. | Low cost due to it’s open-source nature. Cross-platform development is fast and requires familiarity with Linux and open-source tools such as Docker. However, the overall cost can be low or free due to open-source nature. | Licensing can make it costly. Microsoft tools can help shorten development time and requires knowledge of Microsoft-specific frameworks. With that said, its broad support and familiarity for most users can help overcome some obstacles. Additionally, this is typically popular for business or enterprise environments. | Supports a wide user base. However, there is high cost for native app development. Lengthy development time and requires knowledge of platform specific languages. |
| Development Tools | Team will need familiarity with the technologies and mobile deployment could be costly.  Tools include Xcode, VS Code, Sublime and Eclipse/SpringToolSuite. Languages include Swift, Java, JS, and Python. | Team needs Linux experience and there will be no licensing costs.  Tools include Vim, Eclipse, Docker, Kubernetes. Languages include Python, Java, JavaScript and Ruby. | Team will most likely have windows experience due to it’s popularity. Licensing can be costly  Tools include Visual Studio, Docker (Windows Containers), IIS.  Languages include C#, ASP.NET, JavaScript, Python | “Mobile” development is too broad, there are two major operating systems that might require different expertise (Android & iOS). Developer tools could cost additional money.  Tools and languages vary depending on which mobile system is used (Android/iOS). |

## 

## 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: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>**

The recommended operating platform that would allow the Gaming Room to expand into other computing environments is Linux. This is because of its cross-platform development capabilities, low cost and open-source tools.  Using this platform, the team will be able to meet all the goals set by the client. However, there are a few other things to consider. For example, the mobile app should classify as a hybrid which means combining web and native app features to become accessible cross-platform (TOP 5 PROGRAMMING LANGUAGES, 2023). The recommended language should be anything C based such as C++, Java and C#. Additionally, the front end can be coded in in the React framework. This framework makes it possible to use native components and implement API’s.

With that said, server consideration is also of utmost importance. A Linux-based server is stable, safe and offers a compatible environment with the aforementioned languages and frameworks. Not only is Linux widely supported, but also preferred for server operations due to all the previous factors and low cost. When it comes to memory management, tools such as valgrind and gdb can be used to compile and test code across various environments. A Linux server is robust enough for large scale applications such as a cross-platform game and great at handling traffic with low performance loss.

Lastly, utilizing Linux implies having access to an open-source platform with an incredible and wide variety of development tools that require no fees. This reduces the cost and makes it easy to push boundaries without going over budget. Furthermore, using tools such as virtual machines through Docker will enable access to development environments from other platforms. Even more so, as far as deployment, there are many cloud services such as Google Cloud and Azure that offer support for Linux-based servers.

In summary, using Linux is a great option for the team as it meets all their requirements as well as two of the client’s requirements:

1. Low cost
2. Cross platform capability.

With this recommendation, the team should have no issues meeting all goals and maintaining a reliable infrastructure.

1. **Operating Systems Architectures: <Describe the details of the chosen operating platform architectures.>**

Linux is an open-source UNIX-based operating system which operates with Linux kernel (GeeksforGeeks, 2023). It is modular, has multi-core CPU support, uses docker from cross-platform work. Linux comes with predefined system libraries such as the GNU C library, libpthread for multithreading, dynamic linker and a math library. Linux can use Docker to create containers for cross platform development.

For API recommendations Flask and Django offer flexibility and quick development. For an environment, the app can be deployed on Node.js which is a free open-source server environment (W3Schools, 2019).

Node.js is ideal for handling asynchronous operations and efficiency when handling high traffic environments. This is perfect, given that a cross-platform game will typically have more traffic than a native application.

To ensure scalability and cost-effectiveness, using Docker in conjunction with Kubernetes is a great idea. Docker ensures the creation of small light weight containers which can be easily deployed. While Kubernetes can then be used to manage these containers. This ensures the application can scale itself and keep resource usage optimized. This combination is typically easy to manage and cost effective which makes it ideal for our Linux-based server deployments.

1. **Storage Management: <Identify an appropriate storage management system to be used with the recommended operating platform.>**

There are a wide variety of appropriate storage management systems that can be used with Linux. Some examples are Amazon S3, Google Cloud, and MySQL databases. Specifically, images should be stored in a cloud storage service. A personal recommendation would be AWS, as this would provide accurate storage calculations, keep the data safe, and increase accessibility.

1. **Memory Management: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>**

Linux uses virtual memory, paging and caching. The platform also uses several memory management related caches such as buffer cache, page cache, swap cache and hard caches (Rusling, n.d.). Client side, garbage collection is used to managed unused memory.  Specifically, Linux’s management system is what allows processes to utilize more memory than physically available. It doesn’t take two brains to realize that this process would be crucial for a game where memory demand can change depending on the number of active users. Additionally, page caching will help reduce disk access by storing frequently used data in memory. Afterwards, swap caches manage overflows when the system reaching a state of high load. For application specific memory management, there are options such as Valgrind a GPL’d system that can “automatically detect many memory management and threading bugs, avoiding hours of frustrating bug-hunting, making your programs more stable “(Valgrind: About, 2024). If that is not available, GDB can also be used during development to fix memory leaks. All of these factors ensure that memory is properly managed, and the game doesn’t suffer from performance degradation.

1. **Distributed Systems and Networks: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>**

There can be an implementation of microservices architecture using REST APIs for communication. The benefit of this, is that a REST API “defines a consistent and uniform interface for interactions between clients and servers” (Gupta, 2019). Meaning that the application will be standardized throughout all platforms. When it comes to network, a content delivery network would be good for low-latency and Kubernetes can be used for scaling, load balancing and handling outages. For user traffic, the software should be distributed to various sources such as social media and its own web page. This web page should include a web version of the game. The app not only needs to be able to run on multiple platforms, but it should be available in its respective stores. The game should be found on the Google store, Apple store and Mac, Windows and Linux packages for different versions.

In essence, the microservices built on REST will allow communication between different components of the application, regardless of platform. The CDN will then reduce latency and ensure access to game content from people all over the world. In the event of an outage, Kubernetes the “portable, extensible, open-source platform for managing containerized workloads and services” (Kubernetes, 2023) can automatically manage and recover services ensuring minimal downtime. Lastly, it will also distribute traffic to multiple containers which will ensure smooth gameplay. This is important because the game will be posted in multiple sites and available for all major platforms. Therefore, high amounts of players are expected.

1. **Security: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>**

Security is of upmost importance. There are a few considerations to keep in mind when trying to protect user information on and between various platforms. On this case, SSL/TLS encryption will be beneficial to keep all data transmitted between users and the server private. Data stored in the cloud such as in AWS servers will be encrypted with AES-265 which will provide maximum security. For authentication OAuth can be used to ensure identifiers are verified across platforms. Moreover, the recommended platform comes with security specific features such as SELinux which provides mandatory access controls to restrict processes based on security cont3exts. Adding an IDS for intrusion detection could also be beneficial as it will help prevent malicious attacks.

References:

GeeksforGeeks. (2023, September 25). Architecture of linux operating system. GeeksforGeeks. <https://www.geeksforgeeks.org/architecture-of-linux-operating-system/>

Gupta, L. (2019, June 13). *What is REST – Learn to create timeless REST APIs*. Restfulapi.net. <https://restfulapi.net/>

Kubernetes. (2023, September 19). *Overview*. Kubernetes. https://kubernetes.io/docs/concepts/overview/

Rusling, D. (n.d.). Memory Management [Review of Memory Management]. <https://tldp.org/LDP/tlk/mm/memory.html>

TOP 5 PROGRAMMING LANGUAGES. (2023, October 26). TOP 5 PROGRAMMING LANGUAGES USED IN MAKING MOBILE GAMES - Tamatem Games. Tamatem Games. <https://tamatem.co/news/top-5-programming-languages-used-in-making-mobile-games/>

*Valgrind: About*. (2024). Valgrind.org. https://valgrind.org/info/about.html

‌

W3Schools. (2019). Node.js Introduction. W3schools.com. https://www.w3schools.com/nodejs/nodejs\_intro.asp

‌

‌