

<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 | 03/19/2023  04/08/2023 | Maddy Tritel | Modified Summary, Design Constraints, and Domain Model explanation, characteristics, advantages, and disadvantages of developing this type of application deployed on various platforms. |

**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)

Creative Technology Solution (CTS) latest client, The Gaming Room, aims to develop a web-based game based on the 1980’s television game “Draw It or Lose It”. Currently, the application is only available for Android devices. However, the client would like to expand functionality to serve all platforms, not just Android.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Design constraints for The Gaming Room application “Draw It or Lose It” are as listed below:

* The client requires the application to be web-based to allow access to more users and across various platforms.
* The app will render images from a large library of stock drawings as clues.
* Each drawing will be rendered at a steady rate which completes at the 30-second mark
* Each game will consist of 4 rounds, with each round lasting 1 minute
* Team members will compete to guess the puzzle (a phrase, title, or thing)
* If the team fails to solve the puzzle within the given time limit, remaining teams can offer one guess each to solve the puzzle within the 15-second mark.
* A game will have the ability to have multiple 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.

## [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 Entity class handles the Game class, Team class, and Player class. Each of these classes inherit various attributes and methods from the Entity class. All four of the classes, GameService, Game, Team, and Player reference each other. The ProgramDriver class is where our team will develop the project to meet the client’s requirements. In this class, we can access and execute all the created classes. The SingletonTester class ensures the program will run according to the client’s design constraints. The class will allow for there to be more than one player and multiple teams, while maintaining one active game session at a time.

**"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 permits for hosting of the type of game application our client requires. MacBooks hardware permits development of complex applications, and a high security presence. It offers many tools to locate resources and support. Some disadvantages include MacBook hardware can be more costly than other platforms. Licensing fees can also be high. Additionally, applications must be Mac compatible. | Linux is advantageous as it’s free for anyone to use; there are no licensing fees involved. Linux is open source, meaning it’s affordable for any organization for hosting server-side web-based applications. Additionally, unlike Mac, it is extremely customizable, letting users configure and use only the software they need, making it an efficient and lean platform. Disadvantages include this platform can be more difficult to learn and maintain compared to other platforms, meaning added time and resources. | Windows OS can be used for hosting the type of application we require server-side. Advantages include its compatibility with a wide array of applications and programming languages, user-friendly interface, and ease in creating and configuring a server, and built-in security measures, such as anti-virus software and firewalls to protect the application from threat actors and potential breaches. Cost is a disadvantage, as costs for licensing fees can be high for Windows, especially for small organizations. Additionally, Windows can be susceptible to crashes and performance issues, as it is resource intensive. | Mobile devices have limited hardware capabilities, with less storage, compared to traditional server setups. One advantage includes a lower cost than purchasing an in-house server, ease of use, and simplicity in developing web-based applications. Disadvantages include scalability limitations, as well as security vulnerabilities known to exist in mobile devices. |
| **Client Side** | Supporting multiple types of clients on Mac requires some software development considerations. Cost must be considered as developing applications that support multiple types of clients on Mac can have a steep price. Varying needs of clients mean the requirement of different development tools, and frameworks, adding to the overall cost. Additionally, adequate time should be allotted to ensure software is compatible with various platforms, which may need additional testing, and optimization. Supporting various types of clients also requires developers who are experts in Mac development; and understand the needs of their clients and the appropriate software to support those needs. | In developing software to support multiple types of clients, using various platforms, on Linux, there are several considerations that must be accounted for. Although licensing is free, costs may vary based on the complexity of the software at hand, size of development team required, and time to create and maintain the game software. Developing complex software on Linux takes time, especially for custom software. It’s important to plan time efficiently to ensure a polished, and functional software product which meets the needs of clients. This requires a team of software team with expertise in Linux, in addition to specific client needs. Additionally, user-experience should be prioritized, ensuring an intuitive interface across all platforms. | The software development considerations that are important when developing software that supports multiple types of clients on Windows include high cost for licensing fees, as well as tools, such as hardware, and frameworks, to accommodate a wide range of platforms and devices. Time must be allotted efficiently, as developing software to support various platforms can be time and resource intensive. For example, developing UI that are customized to various device interfaces, as well as testing across various environments to ensure optimal functionality. Additionally, developing for multiple platforms and devices requires that developers have knowledge of various programming languages, cross-platform technologies, frameworks, and specific tools for each platform and/or device. | In developing software that supports various platforms and devices on mobile devices, various considerations should be accounted for. Costs can be high. When accounting for licensing fees, as well allow as hardware that allows for cross-platform testing, the fees do add up. Developing software that supports multiple types of clients on mobile devices can be expensive, as it may require investing in tools and technologies that can accommodate different platforms and devices. This may include purchasing licensing and hardware to facilitate testing across various platforms. Building cross-platform applications can be a time-consuming process, as doing so requires more resources and work to ensure optimal compatibility and that the application is functioning as intended. A team of developers is needed with a knowledge of mobile cross-platform technologies, frameworks, etc. |
| **Development Tools** | Programming languages and tools used include knowledge of Swift, and Objective-C, two languages utilized to create applications in MacOS. Objective-C is the main language used to create MacOS and iOS applications, using the Apple developed Cocoa framework. Cocoa frameworks include frameworks for user-interface design, and data management. Interface. | Primary languages include for developing cross-platform software applications in Linux include C++, Java, and Python. Eclipse and Visual Studio IDE are both used for creating cross-platform software on Linux. Make is a great automation tool that allows developers to define a set of rules which dictate how to build an application from its source code. | Languages include HTML, and JavaScript. Xamarin allows developers to create applications across various platforms. Visual Studio is a popular IDE that can be used. | Languages include Swift, and Java. Ionic is a mobile framework that allows developers to build applications across various platforms using JavaScript, HTML, etc. Additionally, Visual Studio IDE can be utilized to create this application, as it allows for deployment of this type of application across various platforms and devices to meet the needs of our client. |

## 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**: According to my research, the operating platform that will be most ideal to satisfy the needs of this project is the Linux platform. There are many reasons why I believe this to be the case. For one, Linux is completely open-source, free for anyone to use, and more secure operating system by nature. Security was of high concern for the client, and a Linux system delivers this security, in part, by restricting root access using a user privilege model. As the superuser, the client will own all privileges, and other users will be granted only enough permissions to complete routine tasks, but will require additional permissions to open files, attachments, etc. These low automatic access rights mean that threats such as malware are far less common on Linux than on other systems. Additionally, unlike other platforms like Windows, Linux systems do not require use of Anti-Virus software, as these tools are often not needed for security when you have a Linux system (Das, 2022). The system has built-in security measures such as firewalls that utilize packet filters within the kernel. Different to other platforms, Linux distributions can run on even low-spec hardware, as their low 256 GB RAM requirement is satisfied by most hardware machines, saving money on hardware upgrades. Lastly, the open-source nature of the platform means a high savings on licensing fees which can quickly stack up.
2. **Operating Systems Architectures**: Unlike Windows and Mac OS, the Linux kernel is monolithic, meaning all hardware and driver operations are handled in the kernel. While bloat can happen for this kernel type, this should not be a major concern due to the low hardware requirements for the most common Linux distributions (Ubuntu, etc.). The Linux architecture consists of two layers, the user space, and the kernel space. Within these layers are four main components. These components include the hardware, kernel, shell, and user applications. The shell functions as an intermediary between the kernel and the user. There are approximately 380 system calls, including “read”, “write”, “close”, etc. Basically, the shell accepts commands from the user and executes the kernel’s functions (Day, 2021). These shells can be classified as either command line shells or graphical shells. User space, or the code in the OS that exists outside the kernel, is separate from kernel space, and is accessible to root users.
3. **Storage Management**: The recommended storage management system to be used with Linux is the ext4 file system. This widely used system supports storage of very large file sizes, making it a suitable choice for storing a large library of stock images to be rendered in the game. Ext4 also includes journaling, a technique which records changes in a log before being written to disk. In the event of an outage, this tool is crucial to prevent data loss and maintain data integrity of the game (Both, 2017). Storage must be easily accessed on the website, as the process is converting an Android program to a web application. Additionally, the Gaming Room may consider a cloud-based storage solution such as AWS. AWS is reliable, can store large amounts of data, accessible through the internet, though it is limited by the internet connection speed. AWS allows users to pay only for resources they need and allows simplified scalability to meet demand of a growing user base.
4. **Memory Management**: Linux employs various memory management techniques to ensure resources are utilized efficiently and effectively. One of these technique is through use of virtual memory. Virtual memory makes use of the fact that memory being used is used all the time. Programs in memory are separated into pages, and the parts that the kernel determines are not needed are moved to the hard drive. When they are needed, they can easily be returned into RAM and retrieved. This also allows large applications to run by utilizing more memory than the physical RAM on the device (Delony, 2022). Linux also uses page caching, which stores commonly used data in memory. This memory swapping frees up memory, preventing the application from crashing due to insufficient resource usage. This technique is especially useful for the game, when rendering massive number of images from the stock library, requiring fast and efficient access. Utilizing these techniques for memory management will ensure enhanced performance and efficient use of resources.
5. **Distributed Systems and Networks**: A distributed architecture will ensure communication across various platforms where various game instances are running simultaneously. This will allow players to interact with their team members (and those on the other team) in real-time and play games together. Hosting the game on a cloud-service provider such as AWS will ensure maximum uptime and scalability. Employing middleware software, such as Knack, offers the tools used by the application to achieve high availability guarantees, and fault tolerance to ensure the game can continue operating in the event of a power outage (“Online Database software made easy”, n.d.). Code optimization should also be completed to ensure game communication is not impacted by excess network traffic causing undesired latency.
6. **Security**: A major concern for The Gaming Room is to ensure data is protected across various platforms (including Linux, Windows, Mac, and mobile devices). Security measures should be taken to ensure this data remains safe and out of the wrong hands. For example, user credentials should be encrypted both at rest and while being transferred to ensure only authorized users can access this information. Additionally, multi-factor authentication should be used to ensure appropriate access control. Linux distributions offer a wide variety of security measures such as stringent user access control rights, file encryption, and more. These measures further aid in preventing unauthorized users from gaining access to the game. Lastly, audits should be performed by the IT team to identify security risks and implement solutions to mitigate these risks before an attacker can take advantage of these security holes. Using these techniques will build ensure a robust security presence and keep user/game data safe.

References

Both, D. (2017, May 25). *An introduction to linux's ext4 filesystem*. Opensource.com. Retrieved April 16, 2023, from https://opensource.com/article/17/5/introduction-ext4-filesystem

Das, A. (2022, October 29). *11 reasons why linux is better than windows*. It's FOSS. Retrieved April 16, 2023, from https://itsfoss.com/linux-better-than-windows/

Day, B. (2021, October 25). *How secure is linux?* Linux Security. Retrieved April 16, 2023, from https://linuxsecurity.com/features/how-secure-is-linux

Delony, D. (2022, August 13). *What is virtual memory on linux? how to manage it*. MUO. Retrieved April 16, 2023, from https://www.makeuseof.com/virtual-memory-on-linux/

Online database software made easy. (n.d.). Retrieved April 16, 2023, from https://www.knack.com/