

<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 | 5/23/2024 | Oscar Medina | Update Executive Summary, Requirements, Constrains, System, Domains, and Recommendations |

**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)

Project background: The Gaming Room seeks to port “Draw It or Lose It,” their Android quiz game to multiple platforms. The application works by rendering images from a stock library, running a four-round game with hard timers in every function such as duration and quizzing limits. The application generates team tied score boards and tracks additions to it, turns are based on hard timers too. Currently the staff at The Gaming Room require assistance with development to set up cross-platform environments.

## Requirements

The Gaming Room for gameplay requires their game to generate, assign, track, and modify the scores of team(s) over four rounds lasting one minute each. To do that a unique id in the form of a name will be assigned to each individual team. The application must be able to open and close a picture library while having a thirty second slowed render in which the team(s) must input an answer. If no correct answer is input, then the team’s turn is forfeit and the second team is given 15 seconds to input an answer. The game must be limited to one instance at any time in the memory.

## [Design Constraints](#_2et92p0)

The main constraint is that the current game build is specific to the Android OS. The development team must familiarize with other platforms to port working builds at the standard of the “common denominator” and make sure each port maintains the requirements. This can be difficult as native tools in each OS may not respond to Android specific applications either because of hardware architecture or different programing languages.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

* Entity Class: Parent class to game class, team class, and player class. Those classes inherit attributes from Entity.
* GameService Class: Main attribute container, indexes the attributes required by The GamingRoom, example ids.
* Game Class: Contains a team’s list, and addTeam function which adds a new team to the list.
* Team Class: Contains a player's list, and addPlayer function which adds a new player to the list.
* Player Class: Contains the player’s unique id which is required to be assigned to teams and games.
* ProgramDrive Class: main statement function container, it is uses the SingletonTester class.
* SingletonTester Class: Test singleton’s behavior.

**"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 strengths are on its strong first party support provided by Apple license. Apps created for MacOS run smoothly across different machines. MacOS applications can run on WindowsOS, and Linux. Mac is also a very user center platform with ease of customization and a friendly learning curve. It is also very maintainable and consistent. Mac platform had sustained consistency across different OS generations. Cons for Mac are that as an Apple product, it is under inflexible policies. Apple has high standards to protect system integrity and security at the cost of flexibility. This limits the ability to edit or manipulate both software and generally Mac hardware is very streamlined. Lastly any additional tool or security service will tag a significant cost to the project, that is already including Apple business licenses. | Linux is one of the best options by attributes, but as an operating system it is more complex. Basic services are free or cheap and are very customizable with integrated remote administration features. It supports shared workshops with good core program protection. Better security features. Optimized usage of hardware. The disadvantages lie in the complexity that Linux has, it an OS that requires more experience from the team. Linux is less popular in the casual market so the team will have to deal with file format compatibility issues. Also, Linux host websites cannot use Windows applications. Apps must be used to translate data between Linux and Windows machines. There would need be a extra interfaces between the Linux server and Windows client Servers. | Windows is the most advantageous for simple and powerful options. It is one of the most popular corporate OS. It has as default optimal automated system updates. The Active Directory server sets up a great environment for secure, maintainable and manageable servers. Good third-party support for many tools such as Azure cloud Directory which does the same as Active Directory. A good system recovery option. Access to Sharepoint and Exchange. Simplified graphic user interface for ease of use. The main disadvantages are high licensing cost as default security protection tends to be weak. Windows servers are resource intensive. Finaly Windows is very furtive about their proprietary system architecture. | Mobile device servers are good at hosting simpler applications with well curated environments. Cost tends to be lower; updates are fast to roll out, and the tools are simple. Android devices specially benefit from many tools and licenses for server creation and maintenance. The disadvantages lie in the limited mobile OS and local hardware. Games are highly dependent on Cloud live services, so a constant and stable environment is a priority. The constant need for data transmission leaves Android apps vulnerable to third party trackers and breaches. Security customization is limited. And mobile servers in general are very limited, unstable, and hosting options have issues with stability. |
| **Client Side** | MacOS is very user friendly, it is simple to learn and retain knowledge on it. Apple has also maintained consistency over generations on their products. The main limitations are that Apple OS is centered on Apple products. Expertise on MacOS is limited on other platforms. This will limit the practicality of concentrating resources and developer time just on Apple OS. | Linux is an opensource and free OS, simple to maintain as the specs for the OS architecture are open to learn. It is very affordable when compared to Windows and Mac. Expertise on Linux will allow fast maintenance, better control on the game. As a con, lack of expertise will present a higher wall to any team as security and features would be mostly user-created. For every great feature on Linux a lack of experience will limit the ability for the team to run maintenance, secure data, and administer the server host for the game until the team undergoes serious learning periods. Also, Linux lacks the first party technical support corporate Apple and Microsoft licenses give access. | Windows OS is a well balance choice as for the right cost Microsoft will provide the appropriate services to the client. Being a widely available OS the bar for adaptation and learning curve will be lower for the team. Windows offers technical support to license holders, and upgraded security packages, there is also plenty of third-party support. The main disadvantages are that Windows extra features need to be paid for on a monthly service, and Windows source code and architecture is obscure and there is need for acclimation period for the team to manage the OS | Mobile devices can take advantage of many tools, and work done on PC is easily translatable to it. Now as a development platform it is the least ideal one as mobile OS do not support complex features when compared to desktops and laptops. The strengths of mobile are in hosting completed apps at low price, with a low entry bar, and good access to marketplaces. Something to note is that mobile devices on the market are diverse with Android and Apple IOS being essentially incompatible. That would mean that at the minimum two different teams to develop and support both mobile OS. |
| **Development Tools** | MacOS uses almost exclusively Swift as its programing language. Mac development relies on first party tools. In this case Xcode IDE which is geared for Apple development. Having a first-party, Apple backed tool kit will streamline development for the game even though this might most back-end different built. This is because Android OS does not have similar tools. Alternative the game can still be compiled on Java as it is a non-platform dependent language, but there are a few issues. Mostly a third-party app must interface on IOS and IPad, which is currently irrelevant as the game is ported to Mac only, as Apple does not support JVM on those devices, but backend development can be done on C and C++. | Linux is complicated to talk about as the tool catalog is rather massive. In terms of this project, it is important to stay consistent with the Android build so that does allow to narrow down preferences. Godot is a good cross-platform for C++ development, and jMonkeyEngine is a good Java based-engine to use on Linux and for cross platform development. | While a C based code would benefit the most from the Windows OS is a better idea to develop the port on Java to save cost. Java can be used both for the PC and Mobile OS. Many IDEs can run the JVM some examples are Visual Studio, Eclipse, and IntelliJ all come with community versions, but for security sakes and tool usage it better to corporate versions. A good game development tool for gaming 2D and some 3D would be the jMonkeyEngine. A free and open-source Java based engine with Android development support. | Java applications will be good base for mobile game development. C++ is usually a better option for gaming as it is a low-level it allows powerful applications like memory intensive games to run optimized, and it also object-oriented. Java also shares some features, but it is easier on the team. Also as identify, jMonkeyEngine is a good cross-platform tool for dual development. Eclipse and Visual Studio are good IDEAs to manage the project. For IOS development Swift and Xcode would be better tools. |

## 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**: Objectively the benchmark for development would be Windows 11. The skill ceiling to port an Android game like “Draw It or Lose It” to Windows is technically lower. Windows also provides the team with tools that will allow seamless integration while maintaining most of the work from the original build (Xsolla 2021). Windows is also one of the OS with the largest user base, so it is more likely that the developer team is both familiar with the tools in use and originally develop the game from a Windows platform. Also, it is not difficult to create servers that can host cross-play between Android client servers and Windows client servers. Windows popularity will also mean that more users can potentially reach the game too.
2. **Operating Systems Architectures**: One of the difficulty in commenting on the Windows architecture is not as open as Linux is. Windows is a layered design which consists of two main components, user, and kernel mode. The are 32-bits and 64-bits versions available, but 64-bits is the standard environment for game development and the game loop process. User mode limits the user’s ability to interact with system while kernel mode is the low-level core of operations, it manages the mainly operations between the hardware and software (GeeksforGeeks 2023). “Draw it or lose It” is a quiz game with animations and image rendering as its feature and not a game with complicated physics. If the game is well coded it should not be CPU heavy; the GPU would carry the game loop in 2D and basic 3D designs. Something to consider for Windows is Layered architecture, it can be used in a monolithic, micro-kernel or hybrid manner (). Hybrid architecture has advantages on stability and management over singular architecture choices. Nvidia GPUs deliver great performance on Windows OS. As far as hardware goes most of it works well with Windows as it is the leading OS in the market (GZ Software house 2015). A notable feature of Windows is the UWP app which helps run other Apps in a semi-universal manner across both Windows 10 and 11 (). The main downside is that the current version is not programable in Java which can limit programing to C++, Unity is one the cross-platform game engines to consider in this case.
3. **Storage Management**. For development purposes using local storage during active development is necessary as it helps prevent leaks and places higher protection on the core code of the game. A server-based storage management system can be used to transfer carryover work and task amongst devs. Central storage access allows for maintaining functionality which is important in a long-term game service, automated data backups, and optimized performance. Cloud storage can be used to save backups for data preservation in case of a security or functionality compromise. Since Windows is a Microsoft product a good option is to use Azure Cloud as the main tool for storage management (Team, C. A. 2023). It has a good range of storage available for the needs of the server. It has redundant storage so the game’s builds will be protected. It offers file sharing options for multiple users.
4. **Memory Management**: When it comes to Windows there are many memory allocation options. Local RAM and cloud storage services offer memory allocation support to Windows games. Alternative, as “Draw It or Lose It” is a simple enough game that it can be hosted as a web-browser or Microsoft app memory usage should be low. As a live service game, a hybrid between dynamic and static memory allocation will be required to keep reliable performance (Xsolla 2021). While “Draw It or Lose It” might not be a RAM heavy game it is important to note that badly contained memory allocations can hamper performance. For example, the creation of multiple instances will eventually flood the Ram and cause a crash.
5. **Distributed Systems and Networks**: Cross-platform games were at one point a new trend in gaming, but currently it is industry standard. It is important to begin with a dev environment that will meet the requirements of both platforms, this saves the team from constantly having to patch the game post launch. Androids have simpler environments and can run modern, but not CPU intensive software. To prevent connectivity and outage issues, the servers must be tailored to run at the base capacity of the lesser devices, Android OS. Dedicated client servers will have to be created to host Windows and mobile devices to connect to a joint game server. Azure Cloud also supports automated cloud-based email alert management, and monitoring services (). This will take some burdens form the team when running maintenance.
6. **Security**: Windows is a platform a host to data brokers, hackers and doxxers. A good service game will protect the data of their user base as much as possible. Hackers often target games and their builds have been released online for coders and modders to play with. When dealing with everything from server protection to third party tools it is better to have premium paid services that secure the user information. It is also recommended that additional layers of protection such as data encryption and tracking of all company computers, it is standard for work to only be done in those machines. Azure storage, Aura, and Github are usually tools used by code developers, but it is a must set aside premium or corporate accounts. Likewise, IDEs like Microsoft’s Visual Code Studio and Apple’s Xcode offer corporate licenses with stronger encryptions and security priority. Azure also offers a cloud Active Directory service to help strengthen logging and monitoring (Khdownie. 2024, May 9). Using a VPN to encrypt IP will help protect outgoing data. Isolating the client servers from the base-functions of the game server will keep the foundation of the game protected from non-admin roles. The rest is up to good practices like not linking personal accounts to company machines, using unnotarized third-party tools and environments, using open wifi, personal devices near the workbenches, and misuse of company accounts such as Emails (Pleasant Solutions n.d.).

Rahul Sehgal. (2023, November 21). *Best PC SPECS for your Game Project | Gamedev tips* [Video].

YouTube. <https://www.youtube.com/watch?v=JJLXOZEihUg>

Pleasant Solutions (n.d.). *Checklist for Securing and Hardening your Server Environment - Pleasant Solutions*. (<https://pleasantpasswords.com/info/pleasant-password-server/f-best-practices/secure-and-harden-your-server-environment/checklist-for-securing-and-hardening-your-server-environment>

Quest Software (n.d.). *What is Active Directory? How does it work? | Quest*. Quest Software. <https://www.quest.com/solutions/active-directory/what-is-active-directory.aspx#:~:text=Active%20Directory%20(AD)%20is%20a,who's%20allowed%20to%20do%20what>.

GeeksforGeeks. (2023, April 21). *Kernel in operating system*. GeeksforGeeks. <https://www.geeksforgeeks.org/kernel-in-operating-system/>

IGZ Software house. (2015, March 16). *Window architecture* [Slide show]. SlideShare. <https://www.slideshare.net/slideshow/windw-artitecture-eve/45872347>

Team, C. A. (2023, September 5). 10 Benefits of cloud Storage. *Cloud Academy*. <https://cloudacademy.com/blog/10-benefits-of-using-cloud-storage/>

Khdownie. (2024, May 9). *Enable AD DS authentication for Azure Files*. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/storage/files/storage-files-identity-ad-ds-enable>

Xsolla (2021). *Porting mobile games to the web, consoles, & pc*. Xsolla. https://xsolla.com/blog/porting-mobile-games-to-the-web-consoles-pc