

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 03/08/2023 | Tammy Hartline | Updated Summary, Requirements, Constraints, and other pertinent information needed for this project. |
| 2.0 | 03/28/2023 | Tammy Hartline | Updated recommendations and system architecture view. |
| 3.0 | 04/16/2023 | Tammy Hartline | Updated information to explain the architecture needs for the different operating platforms, and how to implement the architectures based on the client’s needs. Updated each section to meet rubric requirements. |

**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 client, *Gaming Room*, currently has a well-developed and established Android only based game called *Draw It or Lose It*. Due to the game’s success and popularity on the Android OS, they are now seeking to expand their user base to include Windows, Linux, and MacOS. The updated and integrated web-based, multi-platform game will be based on the existing Android game. They are seeking guidance with server, environment, development, and ways to streamline the process of deploying the updated multi-platform game.Each game and team name needs to be unique, and the game should only allow one instance of the game to exist in memory at a time. Each team will have more than one player assigned per team, and the game will need the ability to have multiple or single teams playing per game.

It is also imperative that each team follows industry standard best practices to ensure scalability and high quality, bug free product that compiles and meets or exceeds the client’s requirements and expectations.

## Requirements

The client requires the game to have the ability to have one or more teams involved, and each team to be assigned multiple players. Only one instance of the game should exist at a time. They request this be accomplished by creating unique identifiers for each game instance, team, or player. The user needs the ability to check whether a name is in use when choosing a team name, therefore, each game and team name should be unique. The game also needs expandable storage and memory allocation that will take run-time into account.

## [Design Constraints](#_2et92p0)

There are several design constraints that the client has made us aware of, and others that were interpreted based on conversations with the client and the development teams.

1. ***Security***
2. ***Multiple Programming Languages***
3. ***Handling of Concurrent Requests***
4. ***Player Login***
5. ***Storage and Memory Allocation***
6. ***Game Operations Management***

***Constraint 1: Security***

Because the client would like to be able to differentiate players, using multi-factor authentication should be considered to maintain the systems security and handling of login and identification.

***Constraint 2: Multiple Programming Languages***

The current application is only programmed using Java, and Android based. This needs to be adapted to handle multiple languages, in order to integrate and compile on multiple operating systems.

***Constraint 3: Handling of Concurrent Requests***

Given the new application will be available on multiple platforms, the server must have the ability and capacity to handle concurrent requests from each platform’s user-base.

***Constraint 4: Player Login***

Application needs to be able to verify whether a player’s login is accurate, and already in the system, or if user is new, the system should let the new player know whether their username is available or already taken.

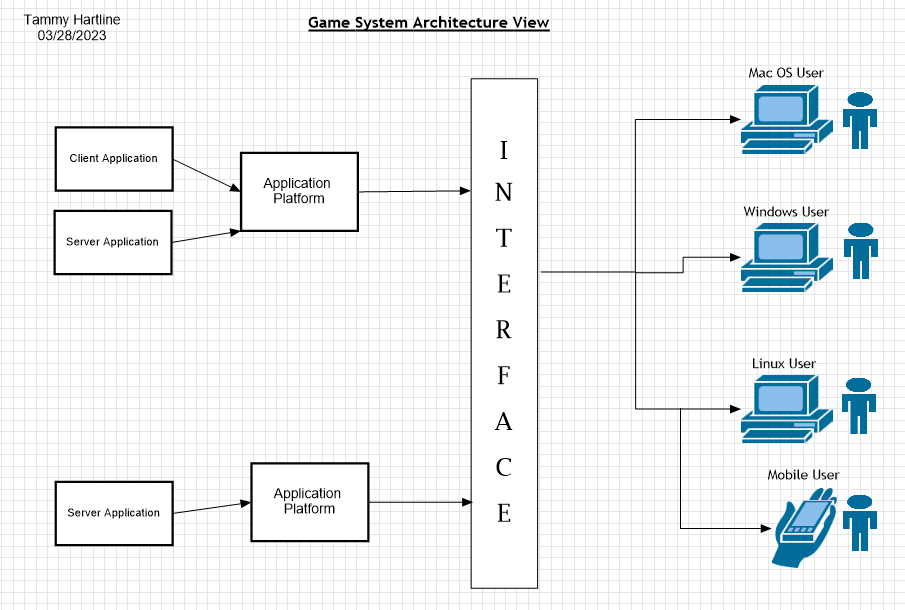
***Constraint 5: Storage and Memory Allocation***

The server side needs to account for enough storage to hold the game’s current collection of stock photos. It should also be expandable to ensure that storage is handled efficiently and to avoid continuous manual maintenance from having to performed. Automating memory allocation should also be considered.

***Constraint 6: Game Operations Management***

Developers need to consider how to handle managing the system, in order to differentiate teams, games, and players.

## [System Architecture View](#_ilbxbyevv6b6)



## [Domain Model](#_8h2ehzxfam4o)

The Entity Class is the parent (super) class of the Game, Team, and Player classes. This means that Game, Team, and Player class, as Entity’s child classes, will inherit Entity’s attributes, while each being assigned attributes of their own, that are separate to the parent class. The Game Service Class is used to ensure the client’s requirements are met, providing a single game instance at a time, unique team name (id), unique game name (id), and unique player name (id). Program Driver contains the main statement and uses the Singleton Tester class. The Game class contains a team list, Team class contains a Player list. The Player class does not contain a list, as it ensures that each player has a unique id, that can be assigned to a team. While a player can be on a team, and a team does have players, the player class does not contain or have a team or a game.

**"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)

***04/16/2023***

The majority of this game should be coded in C++, as it appears to be the most popular language for most game developers due to its compatibility on multi-platforms. For apple devices, there language, Swift, will likely be necessary for at least a portion of the program code. I would, however, suggest further research into already developed libraries that can be utilized in C++ to avoid having to code in multiple languages. I was able to find information regarding servers, and cost-effective options, which include renting servers or using a community-based peer to peer network system server. There are multiple games that already implement the rented servers, or P2P, such as Minecraft and RUST, making it a considerable recommendation. I am not familiar enough with them, however, to fully recommend this as option, given I am not certain of the security risks they impose. With that being said, if a secure and affordable option were found, this seems like an excellent solution for this project. Otherwise, there are still a plethora of options using cloud servers, as I would not recommend on-premises servers, due to their size and limitations. For security, I do not see multi-factor authentication as a necessary step on the user side. Instead, use of password restrictions and required password changes every three months should provide the user side of the game adequate security. However, it should be used for all other sides to keep the data secure.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | ***Advantages:***   * Ability to run MacOS, Windows, and Linux apps side-by-side. * Provides extensive documentation and excellent tech support. * The MacOS system has remained consistent, making it easy to operate for those used to the OS.   ***Disadvantages***:   * Cost and limitations to hardware options when compared with Windows, Android, or Linux. * New and updated software versions often require payment or a new device. | ***Advantages:***   * Wide range of distribution choices. * Open sourced with several free or cheap options. * Established reputation for its versatility in servers and embedded solutions due to ease of customization. * Superior security protocols, as compared to Windows or MacOS.   ***Disadvantages***:   * Lack of choices for pre-built machines. * File format compatibility issues. * Less research and information available about security shortcomings. | ***Advantages:***   * Offers wide range of customer and software support. * Offers ample hardware options. * Updates are easily available and process quickly due to the extensive size of userbase. * Offers corporate users, Windows based authentication integrated with Active Directory based corporate servers, right out of the box, with no add-ons or additional expense needed.   ***Disadvantages***:   * Lack of mobility development. Security protocols are lacking which can lead to malware, spyware, and ransomware. | ***Advantages:***   * Easily manages server-side calls for optimized database queries. * Data storage is persistent. * Offers users and clients mobility and access from virtually anywhere. * Users are unable to see the code, therefore unable to manipulate the program.   ***Disadvantages***:   * Often less secure and more open to corruption. * Less storage capacity than other options. * Can easily overheat, resulting in slower run-time. |
| **Client Side** | ***Advantages:***   * Ease of use once skills and understanding are developed. * Development and deployment time is average. * Offers easy cross-browsing testing.   ***Disadvantages:***   * No generic version that comes at better price point. * Only accessible on Apple devices, which can limit developers, whom may have the knowledge and training to develop MacOS programs, but do not operate and develop from a Mac device. | ***Advantages:***   * Affordability. * Control over your developments offered by Linux. * Free and open-sourced OS. * Simpler to maintain upkeep and thus takes less time for maintenance. * Offers accommodation for tree file structures.   ***Disadvantages:***   * Security could be an issue that is faced alone. * Due to it being free and open sourced, no technical support readily available. | ***Advantages:***   * Availability. * Affordability with price ranges that can be customized based on project usage and need. * Technical support and advanced security options when compared with Linux.   ***Disadvantages:***   * Development requires Windows OS expertise. * Additional cost for specific features you may want or need to include. | ***Advantages:***   * Multitude of tools that are easily available. * Affordability at several price-points. * Accessibility   ***Disadvantages:***   * Full features offered on PC’s are generally unavailable. * Run-time can be slow when compared to others. * Each device runs on its own OS, which can be specific to the device, making it inoperable or incompatible with other operating systems and applications. |
| **Development Tools** | * Swift programming language. * Xcode and Xcode Cloud which is a service that was developed specifically for Apple developers, to allow teams a faster and simpler way to build, test and deploy apps more efficiently. * Offers compatibility with Windows and Linux when ran through a virtual machine. * Eclipse for Java | * Docker, which is used to offer a consistent dev environment, build cross-platform applications and easy deployment. * Docker Hub, used to help skip the dev environment setup and allows users to dive right into development. * Shell and terminal * Eclipse for Java * Offers compatibility with Windows and MacOS when ran through a virtual machine. | * Primary language is well known C, with some parts in assembly language. * IDE Visual Studio. Visual Studio can also be used as a code editor and in some instances, a source/version control. * Offers extensive documentation for deployment. * Eclipse for Java | * Primary programming language is Java. Python and C++, or other variants of C are also utilized for mobile development projects, particularly gaming. * Extensive IDE’s , with the most popular being VSCode, IntelliJ IDEA, and Eclipse. * Xcode 12 offers deployment to IOS * SwiftUI * $99 annual Apple development program for 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**: Windows is the recommended operating system for this project. Windows is the optimal choice due to its ability to integrate with their current Android application of “Draw It or Lose It.” Windows large userbase, making up around 90% of OS for users worldwide, will provide the most options to a range of developers, skillsets, and the tools that are available to them, ensuring easier and more efficient development for a cross-platform application at the best price point.
2. Windows can easily integrate with the current Android build.
3. Offers an array of solutions, such as Xamarin, Cordova, and React to develop the new multi-platform game, with minimal redundancies.
4. Offers a wide range of Microsoft products to ease the burden of design, development, and deployment.
5. Several emulator options for command prompt, power shell, and Ubuntu compatible testing.
6. Has excellent reputation with the largest userbase.
7. **Operating Systems Architectures**: For the system architecture, I suggest using a hybrid architecture. This will pull in all the best features from the monolithic, layered, and micro-kernel architectures. One operating system made with a hybrid architecture is Microsoft Windows NT kernel. The advantage of choosing hybrid over one of the other singular architectures is the allowance this provides to their services respectively. Because of its layered approach, it is also easy to manage. The number of layers is lesser as well. The hybrid architecture will provide improved security and protection versus using just monolithic, layered, or micro-kernel. It also allows the client more customization of their system, based on their needs.
8. **Storage Management**: Given Microsoft works seamlessly with Windows, Microsoft Azure is highly recommended for this project. With competitive pricing, expert and readily available customer support, and consistent updates, this storage management system is the prime choice. There are many reasons why Azure offers the superior storage management system for this project, such as:
9. Azure Cloud – allows use of Docker containers, leveraging cloud storage instances that can be deployed to Azure Cloud.
10. Offers multiple scaling options for storage that are easy to apply.
11. Offers cost efficiency while maintaining scalability.
12. Given we know the client already has 200 8MB images that need to be stored for their gaming application, Azure also offers a file system, storage containers, and blob storage, which gives the client added access to an array of additional storage options.
13. Offers file share, which consists of 1.6GB storage per user.

**Citations/References:** https://azure.microsoft.com/en-us/services/storage/files/#features

1. **Memory Management**: Windows offers a variety of storage and memory management options. Such as Azure Storage. The OS includes virtual and physical address space for memory allocation. There is also the consideration of OneDrive, Visual Studio or even Azure Cloud services that can be used to store and manage versions. Using industry standard best practices when developing the program will help with runtime and memory allocation and management, especially when considering storage container types, such as those that do not allow memory expansion, versus those that do. Windows has proven with its newer releases that they plan to continue to improve their memory management capabilities. Taking advantage of disc paging and demand paging is one of the ways they have kept up with the demand to have more memory. Newer versions of Windows, like Windows 10, allows each process to utilize virtual memory address space in its entirety, providing amble memory for this project.
2. **Distributed Systems and Networks**: As I recommended Azure for storage management earlier, that is not the only benefit or area it can be utilized in. The Azure Cloud service offers easy solutions that provide ease of access when working with distributed systems and networks.
3. Offers maximum uptime using cloud-based email alerts, App Insight Logging and Monitoring Service. Services like those offered by Azure will enable the client to keep up with the demand of their users more easily, through automation versus manual tasks. This will ultimately take time and effort spent on network loads down to a manageable level and enable them to spend more time focused on the application itself and its features.
4. **Security**: With ways to steal user data being a constant and ongoing threat, I would recommend special focus and attention to security protocols. One service that offers protection for PC, Mac, Linux, Android and IOS devices is Aura. While this will come at a cost, using standard security that comes with an OS is not recommended, as an additional layer of protection is needed to keep systems safe and operating optimally. Aura also offers 24/7 U.S. based customer support, which is a huge pro for this service. Another consideration is ensuring company employees understand the importance of password security protocols and implementing an authentication process in tandem with authorization. I also recommend standardizing authorization protocols, that include limiting access to information on an *as needed* basis only, versus allowing the same level of access to everyone or allowing everyone access to everything. With the above mentioned in mind and given my earlier recommendations of Azure, I recommend the client utilize Azure’s App Service, through their App Service Plan using Active Directory for login. This service includes additional features as well that can be utilized with an active internet connection, such as:
   * 1. Allows limitations of access per user on an as needed or *need to know* basis using IP configurations for more specialization.
     2. Available options for storage within a VPN could offers enhanced and optimized security.
     3. Offers optimal options for database security, such as requiring SSL connectivity and whitelist only IP configurations.
     4. Encryption options for additional protection against security breaches, to protect personal data from being compromised.

**Citations/References:** <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-database-security/?ef_id=037eecb058b31669a99b888c6d034e7a:G:s&OCID=AIDcmme9zx2qiz_SEM_037eecb058b31669a99b888c6d034e7a:G:s&msclkid=037eecb058b31669a99b888c6d034e7a>