

Draw it or Lose It

# **CS 230 Project Software Design**

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 | 01/20/2021 | Kyle Gaskill | Added more iteration methods that checks for existing players and teams with the option to load them onto a team. Duplicate game name checks were also implemented to reduce redundancy. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has previously developed an Android App based on the popular 1980s television game that has show success. Wanting to expand to a larger market, they are wanting to develop the game into a multiple web-based platform. Some critical aspects involve the multiplayer platform from the Android App and the future web-based version. The specific requirements of having unique names and team names across all platforms. User “BOB” on Android should be the only user named Bob and no other platform should be able to select “BOB”. Teams will have multiple players assigned to them before the session starts. We need to implement a unique checker by iterating through the names in the user database for letters and numbers combination. We can create this process in a way that ignores case sensitive letters and will see if it exists when a user tries and creates a player name. This same system can be done for the team. Each team and player can be assigned a unique ID number that will allow our backend server iterate check for pre-existing names. This will satisfy the requirement of unique team and player name. We will also need to create a way for the user’s client to queue for a game and be automatically assigned to a team. There can be options for a person to invite people to their team implemented later. The client would create a game instance query to our server and if no instance of the game is found, can be assigned a team and instance ID.

## [Design Constraints](#_2et92p0)

Since the web application will be on multiple platforms, it is best to have methods for the user to login to their account across multiple devices. Apple likes to use their Game Center or Apple ID to store game login information on accounts. The app is currently on Android and synced with a Gmail account. We should have a method for a user to create an account/login method by utilizing a Gmail account, Facebook account on all platforms. This is to allow an iOS user creates an account on his iOS device but bind the account to another sign-in method, like Facebook or Gmail. This is way a user that may have more than one type of device or no longer has an iOS account does not lose access to their game account.

With the expansion on multiple web platforms, so must our expansion of storage to hold all this information. We must make sure we have enough storage space to account for continuous updates and additions to library of drawings based on the expected increase in player base. We want to ensure duplicated drawings are extremely low in frequency.

With also the expected increase in player base, so does the expectation of an increased in active concurrent player sessions. We will want to develop an application that monitors spikes in client-server inquiries to make sure game quality does not deteriorate by an increase in requests.

Configure an SSL certificate to be assigned to the account and all login methods bound to the account, i.e. Gmail, Facebook, etc. This will help the server authenticate each user separately.

## [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 is the parent class for Player, Team, and Game. These three classes inherit the methods and their attributes that allow the program to fetch variable information such as names, ID, and output the parameters. The Game Service shares association with the Game class, which also has association with the Team Class and then the Player Class. The data variables from the Parent Class Entity are private and is being accessed across other classes through the mutators and accessors. The “super” keyword is used in the constructors that pull the data from the parent class. The Game Service class shows polymorphism in the form of method overloading. getGame method can be called with different variables to receive different information, i.e. string name, and long id. The variables used are shared by association with the Game, Team and Player class regarding the names of players, the team name, and the ID numbers. The Team Class will create a ArrayList object from Player class which will be fed to the Game Class who creates an Array List from the Team class by accessing their Player Array List. The Program Driver (main) uses the SingletonTester Class to authenticate if a Singleton Service is being used or not.

****

## [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** | Pros: Mac offers a macOS Server which is a Unix-like server OS. Handles administration and requests of the server in a similar format to Linux.  Supports JVM Applications which will host the REST API.  Cons: Not realistically scalable on an enterprise level. MacOS only configurable on Apple products. Expensive for development and maintenance. Must train or hire staff that is knowledgeable on MacOS servers. Hardware limitation can lead to excessive computer licenses due to scaling limitations. | Pro: Linux server’s offer paramount security and arguably the most secure kernel. Can be accessed across all platforms and managed remotely. Enterprise scalability. Cost of use is lower due to no licenses and free O.S. Ton’s of built-in software to Linux for web design. Really low hardware demands. Remote functions built in.  Cons: Downsides they are harder to set up and require an understanding of how Linux works since most operations are done through a terminal. Some programs and changes have to be installed by root (administrator). Some business professional programs are not compatible with Linux. | Pros: Windows offers servers on an easy to use operating system. More support on Windows OS than other platforms. Enterprise capable scaling. Allows remote administration. Utilizes proprietary software like ASP.net that helps optimize web applications. Has GUI server management tools. Utilizes Active Directory security protocols. Can configure credentials for specific users. Domains, forests and trees can add users assigned to set privileges.  Cons: Security is a risk due to malware threats. More prone to cyberattacks. Cost higher due to licenses. Remote server client needs to be installed and configured. | Pros: Mobile devices offer low-cost web server. No licensing fees for OS.  Cons: Performance is limited based on hardware capabilities. 3rd party software may not all be compatible. Not realistically scalable due to hardware constraints and hardware costs. |
| **Client Side** | Pros: Mac uses a functional and stable programming language called Objective-C. They offer great functionality with their SDK’s.  Safari is secure and functional browser that comes on machine by default. Secure operating system with low security risks.  Cons: Operating system takes getting used to. Not user friendly. | Free operating system with no licensing fees. Open Source. Light weight can be run lower end hardware specifications. Can be run through virtual machine on your operating system. Firefox comes preloaded depending on version/type of Linux OS you install. Very good security.  Cons: Difficult to use. Takes some time before getting used to how to navigate the client. | Pros: User friendly interface. Most common Operating System, therefore universal understanding is more common. Microsoft Edge is light weight and uses less resources than Chrome. Better Privacy settings in client.  Cons: Frequent updates to ensure security vulnerabilities are patched. Expensive cd key purchase if building a machine. | Pros: store client capable of installing application. Low cost on some hardware. Ease of use operating system with simplicity features. Low security risks. Mobile, on the go. Fits in your pocket or bag.  Cons: Can have hardware constraints based on device. Operating system version varies on device, may have compatibility issues. |
| **Development Tools** | Pros: SDK will use Objective-C. Xcode, Apple’s SDK is low cost for the license at $99 a year. Objective-C is a objective oriented programing language that is easy to follow with shared variations from other OOP languages.  Cons: Downside is cost, macOS SDK’s are costly due to the constraint of only being usable on macOS via Xcode. | Pros: Linux can use a variety of IDE’s due to the support of almost all programming languages.  Offers great tools to develop web applications preloaded. Lower costs. Hardware capabilities are lower since operating system is lighter. You will be able too set up Java, C/C++ rather quickly and start the process on development with no downtime.  Cons: Setup and configuration can be tricky. Installing software is unconventional compared to other operating systems. IT support may cost more money. | Pros: Windows supports almost all programming languages with their respective IDE’s. Simple IDE installation. Ease of use with GUI operating system. Free IDE’s.  Cons: Must install JVM. IDE’s do not come preloaded. Must be installed and configured. | Pros: Can use popular Ide’s like Arduino, eclipse, Java. Can start development on the project rather quickly. Low-cost equipment.  Cons: Software can be more demanding on certain devices. Applications code is not easily backed up. JVM does not run-on android devices. JVM cannot be installed on 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**: I recommend Linux for the Operating Platform as it offers a very secure server. Linux offers great versatility when it comes to running as the operating platform of a server. Is accessible remotely and offers great security control. Linux offers a flexible operating system that allows for enterprise scaling at ease. Linux also offers great support for various operating system architectures. Linux is a free operating system, which makes it cost efficient when implementing. Linux comes with readily available web development applications on board.
2. **Operating Systems Architectures**: As previously mentioned, Linux has the most secure kernel of any operating platform. The kernel is the core part of the operating system. Beyond that, there is different modules that moderate the interactions of the hardware. I recommend using the Exokernel for the system architecture. This kernel makes the process of programs communicating with the hardware much more direct, which would increase optimization of the server. This removes the abstraction method behind file systems, memory addresses, task management schedulers, and sockets for network communication. Instead, there will be implemented applications with library operating systems that the kernel will allow the application to access hardware via memory addresses. The ability to go serverless can be implemented in the event of outages.
3. **Storage Management**: I recommend Google Cloud Storage, to store the back end data for the client and libraries of the program. This will reduce bloat of the program being accessed by memory on a client’s machine. This will also allow the server and client to scale at a potential unlimited amount. This will allow the library of images to be stored and can be changed on demand. This will allow the client to bypass updates and storage space to accommodate images and the updated images.
4. **Memory Management**: Linux will be using an Idle Page Tracking memory management that tracks the workload of memory pages being accessed and which ones are being left idle. This can help increase memory allocation on server side. Client side will have objects loaded into memory. Objects like the images that need to load quickly will be stored in memory until GC discards the old objects from memory. We would utilize -Xms setting to set the JVM memory heap to 8 GB. This will allow more memory storage in the heap that will allow more objects to be stored in memory.
5. **Distributed Systems and Networks**:

Multiple platforms communicating with each other through the distributed software will he done through REST API via client-server. With the client being installed on any given platform. The client will access information through the server by the cloud from storage. This will be loaded into memory on the local client by access through HTTP resource methods. These methods will allow the communicating between platforms and their ability to connect to games. Outages from the Linux server hub can be rerouted through serverless cloud application and allow continuous gameplay while the interruption persists.

1. **Security**: Linux offers an ease-of-use control system that allows the root (Super User) to set permissions on what files/folders can be accessed. This will prevent information being accessed in an unauthorized manner. Which will in turn help keep privacy risks with user data. User information will be stored on a secured folder. Best practices of 2FA access to server from remote location, as well as the practice of principle of least privilege. This will insure only users that need access to files will have it. Which will help keep down compromised accounts. In addition to 2FA, in the event an account is compromised, 2FA verification will be required to gain access. This will significantly reduce access to user accounts with compromised passwords.