

Draw it or Lose it

# **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 | 11/14/2024 | Michael Clerico | Attempted to improve project two grade |

**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 expanding on their existing Android game, draw it or lose it. They want to transform it into a web-based application that can be accessed across multiple platforms. This document will outline the design of the software and address the clients’ requirements. This includes how it will be accessible across multiple devices, enabling multiple teams to play together, having unique team names, and all this through one single game instance stored in memory. We will create technical support for continued maintenance that will ensure scalability and a user-friendly gaming environment.

## Requirements

* A game will have the ability to have one or more 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.
* 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)

* Cross-Platform compatibility – The game will be available on a multitude of operating systems and devices. These systems include MacOS, Windows, Linux, and ChromeOS. This also includes different hardware from Tablets, Chromebooks, and Desktops. Each operating system will need its own development, testing, and deployment and production environment to ensure consistent performance across platforms.
* Game data storage needs to be accessed regularly from multiple devices. The game necessitates the need for enough memory and storage to render stock images at an adequate speed.
* User authentication will have to be implemented from device authentication because of the migration to a client-server setting. User authentication will also require an administrative login to manage the users and teams. This will have to be created from scratch because it does not exist.

## [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)

Game: Manages the overall game including teams and rounds. Here is where the single instance is enforced.

Team: Each team of players are identified with a unique name

Players: Each individual player is identified with a unique name

Entity: Game, Team, and Player attributes and behaviors are held here. Behaviors include adding, updating, and deleting within the game.

Inheritance – The unique identifiers are managed by the entity class; the behaviors can be customized within the subclasses. This satisfies the requirement to have unique names for games, teams, and players.

Singleton – Singlton pattern ensures only one game can exist in memory at a time. It uses encapsulation to ensure controlled access. Getters and setters are used to access the methods.

Iterator – The game associates with teams, and the iterator allows for traversal of the team collection. The players are also associated with teams and the use of an iterator allows for transversal of the player list. This satisfies the game having one or more teams involved and each team having multiple players assigned to a team.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Known for stability and security through MacOS Server which can oversee web hosting. User friendly Interface. Mac has higher costs in hardware and licensing fees. Will run into limited hardware options. UNIX based but strict proprietary kernel. Supports ACLs and user authentication. MacOS manages injection attacks through TLS and AAS. Limited cloud support due to strict Apple licensing agreements and Apple ecosystem. Utilization of VMware or parallels is used for virtualization but is not as flexible when compared to windows or Linux. | Extremely versatile with server deployment due to it being open source and free use. Has compatibility issues with proprietary software and requires more technical experience. Open-source UNIX based kernel, so Free to use. Community driven updates can be fast. Also uses ACLs and user authentication. DoS attacks will be mitigated through IP filtering and load balancing. Linux is ideal for cloud environments due to it being open sourced and widely accepted across popular cloud platforms. Docker is used for virtualization and is ideal for isolated environments of testing. | Windows has a multitude of tech support. It easily integrates with other products and windows deployment system. Will manage web hosting. Higher licensing costs when compared to Linux. Strong centralized RBAC which implements ACLs and user authentication. Uses TLS, AES, IP Filtering, and load balancing to mitigate DOS and Injection attacks. Widely supported across multiple cloud platforms. Has dedicated support of Hyper-V which will support multiple VMs running. This ensures a robust testing and development environment. | Mobile devices are not usually used for hosting servers. Interacts through APIs using TLS to ensure secure communication. Android and IOS do not have server components. |
| **Client Side** | Knowledge of swift with the use of Xcode and ensuring compatibility with safari. Will cost more due to the specialization. Will add more time due to specialized coding for Mac. MacOS is optimized for safari. Webkit is the engine that is used with Safari. Safari is included in the Apple ecosystem. It is not officially supported on Linux, Windows, or Android. Firefox and Chrome are also supported. Safari holds 8.9% market share Sheldon and Lockhart (n.d.). | Linux is compatible with various languages including Python, JavaScript, and HTML/CSS. Compatibility with Chrome and Firefox is critical. Lower cost due to open source. Chrome and Firefox dominate in popularity. Firefox holds 7.8% of the market share. Sheldon and Lockhart (n.d.) | Windows is compatible with various languages including Python, JavaScript, and HTML/CSS. Cost will be moderate due to licensing fees for the development tools. Chrome, Firefox, and Edge compatibility critical. Edge is a windows built-in web browser but is based on the Chromium engine. Chrome, the most popular web browser is available also. Firefox on the other hand works off the Gecko engine. Edge Holds 10.6% of the market share. Chrome holds 66.9% of the market share Sheldon and Lockhart (n.d.). | The use of React Native should be used to integrate into Android Studio and Xcode Simultaneously. Compatibility with different screen sizes will be important. Higher cost and time due to multiple platforms testing and specialized software. Chrome and Firefox are readily available across both iOS and Android. Chrome is your standard browser with android. Safari again is exclusive to iOS. If we just isolate mobile market share chrome still holds 65.97% of market share vs safari’s 23% Statista (2024). |
| **Development Tools** | Swift, Objective-C, HTML/CSS, and JavaScript through Xcode, VS Code and Eclipse. Primarily uses Xcode for development environment. | Python, JavaScript, HTML/CSS can be implemented using PyCharm, Eclipse, and VS Code. The use of open-source tools is vast. | C#, Python, JavaScript, C#, HTML/CSS can all be used through PyCharm, Visual Studios, Eclipse, and VS Code. | Mobile app development happens on Windows, Linux, or MacOS, not on mobile devices or for mobile devices. |

## 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 the use of Linux for a multitude of reasons. First is its Cost-Effectiveness. Linux is an open-source platform which will eliminate the licensing costs of the other platforms. This makes scalability down the road less costly. Linux provides a varying array of options for web servers. Adaptability when it comes to multiple cloud environments and virtualization are critical when it comes to game hosting and development. The wide array of development tools available to meet the needs of different web browsers will be critical to build a high-performance web game. Linux utilizes Unix based architecture that ensures reliability and robust security features. This is critical of a user authentication gaming environment (Stallman, 2022).
2. **Operating Systems Architectures**: As stated in my initial recommendation, Linux uses the Unix based modular kernel. This allows you to customize system resources to ensure an efficient gaming environment. The game’s backend will manage the server and API requests from the clients. The clients will be using browsers or mobile apps. The server will manage the games user authentication, game state image storage, game logic, and interfacing with other systems for scalability like cloud storage. Linux Docker allows for isolated testing and deployment which is ideal for game components and in the end enables greater maintainability (Stallman, 2022).
3. **Storage Management**: For storage management I recommend Cloud storage like Amazon s3. Amazon S3 allows for unlimited storage compacity. This will allow the game to grow with no limitations on image size. Cloud will allow for data reliability even during high demand periods. The cloud storage and speed will enable photos to be quickly retrieved to ensure a smooth gaming experience. S3 integrates seamlessly with Linux servers. S3 also supports AES and TLS security during rest and transit. This ensures the data is only to be accessed by authorized users. There will be minimal upfront cost and easy cost management with pay as you go structure (Amazon Web Services, n.d.).
4. **Memory Management**: Linux uses a virtual memory technique such as combining ram and disk space to manage large datasets efficiently especially during game play using high quality picture renderings. Linux will cache frequently accessed game assets with L1 and L2 cache to reduce access to large storage solutions like cloud and enhance speed and performance (Chapel, 2023). This dynamic memory allocation allows Linux to manage memory resources efficiently. Linux will automatically reclaim memory when they terminate. This process of garbage collection ensures resources are free to be used (Silberschatz, Galvin, & Gagne, 2018).
5. **Distributed Systems and Networks**: Using a central server we will manage the game logic and user authentication with communication secured by APIs. With the implementation of Amazon S3 we can utilize AWS load balancer to distribute traffic to ensure consistent performance even during high traffic situations. The other benefit to AWS is it has built in data synchronization with RDS to keep data synchronized across multiple regions. AWS will also dynamically adjust resources to meet current user demand (AWS, 2023).
6. **Security**: For security we will use a multitude of factors to mitigate attacks and ensure data integrity. The first most fundamental security that we will implement is user authentication. We must know who is logging in and what access they will have. This leads me to the need part of user authentication and that is role-based access control. We must assign administrators and users. Everyone will have strict permissions and we will enforce them through account authorization. Then we will have to protect the data within the system with AWS using AES-256 while data is at rest and TLS while in transit. The use of AWS shield will utilize input validation to prevent DOS attacks. Having a regular backup to Amazon S3 will be essential for quick recovery in case of data loss (AWS, 2023).

Reference

Statista. (2024). Market share held by leading mobile internet browsers worldwide from January 2014 to August 2024. Retrieved from <https://www.statista.com/statistics/263517/market-share-held-by-mobile-internet-browsers-worldwide/>

Sheldon, R., & Lockhart, E. (n.d.). Web browser comparison of Chrome, Firefox, Safari, and Edge. TechTarget. Retrieved from [Web browser comparison of Chrome, Firefox, Safari and Edge | TechTarget](https://www.techtarget.com/searchenterprisedesktop/feature/Web-browser-comparison-How-Chrome-Firefox-IE-Edge-stack-up)

Stallman, R. M. (2022). *The GNU Project and Linux*. GNU Operating System. Retrieved from <https://www.gnu.org/>

Amazon Web Services. (n.d.). *Cloud computing services.* Retrieved December 10, 2024, from <https://aws.amazon.com/>

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating system concepts* (10th ed.). Wiley.

Chapel, J. (2023). Cloud storage cost comparison: AWS vs. Azure vs. Google. Retrieved from <https://jaychapel.medium.com>.