

*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 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/18/2023 | Andrea C. Sherry | Recommendations regarding operating platforms, operating systems architectures, storage management, memory management, distributed systems and networks, and security have been implemented in the project software design template. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is a company that has developed an app called Draw It or Lose It which is currently available for Android only. The Gaming Room would like to develop a web-based game that will serve multiple platforms. The staff working for The Gaming Room are currently struggling with setting up the environment. To help them, Creative Technology Solutions will be streamlining the development through the creation of a software design document and the development of the game application.

## [Design Constraints](#_2et92p0)

Based on the information given to us by The Gaming Room, these are the following design constraints:

* The language used for this project will be Java as this will be a web-based game.
* Since the application will be developed for a specific platform, we will be using API framework.
* The application must allow for one or more teams involved.
* Names will be created for players, games, and teams.
* Players have to be assigned to each team.
* The timeframe for this project will be 2 months.
* Testing of the game will last 2 weeks.
* Fix any errors that might arise with the web-based game – 1 week.

## [System Architecture View](#_ilbxbyevv6b6)

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)

First, the UML class diagram provided below displays several classes, these being the Player, Game, and Team classes, having inheritance or “is-a” relationship with the new Entity class. This new class allows us to keep the code written in one single class rather than having to write it multiple times inside the different classes. Next, four of those classes, these being the GameService, Game, Team, and Player class have 0 or more objects associated with each other. Finally, the ProgramDriver class and the SingletonTester class have an association relationship or “uses”. The navigability found between them allows for the main class to test.

**"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** | A characteristic is that while Mac is popular when web hosting, a weakness is that it is not the preferred operating platform for web hosting. An advantage that Mac has is a flexible terminal command that allows for server configuration, make changes to it, and access it. | A characteristic for Linux is that web hosting is preferred due to how highly secure it is. An advantage is that problems are caught quickly before they become a problem. The disadvantage is the difficulty of finding applications that support the needs of web hosting. | An advantage is that there is a large amount of software available when compared to other operating platforms. A disadvantage is that Windows is known for its security issues which makes this operating platform vulnerable to hacking. A characteristic is that Windows has a higher resource requirement than other operating platforms. | A characteristic is that it is more portable than other operating platforms. An advantage is that it produces better results at a lower cost. An advantage is that the server is immobile which allows it to be tracked in a single place. A disadvantage is that this operating platform is not as secure as other available operating platforms. |
| **Client Side** | Mac has a moderate cost due to not being open source. Moderate time and expertise required. | While the software development consideration for time and expertise is high when working with Linux, its cost is minimum. | Windows offers a moderate cost similar to Mac operating platforms. Its expertise and time requirements are minimal. | Although implementation is slightly more difficult, the cost and expertise are moderate, and it allows for clients to easily access updates. |
| **Development Tools** | Relevant programming languages are CSS, HTML, and JavaScript. Some other languages are better to support frontend such as Python, Java, or Ruby. A tool to run with Mac could be for example NotePad++. | Some relevant programming languages are CSS, HTML, and JavaScript. Some tools to use with Linux are eclipse, visual studio, and NotePad++. Languages that could be used to support frontend could be PHP, Ruby, or Java. | Some relevant tools to be used with Windows are eclipse, visual studio, or NotePad++. Programming languages to use are JavaScript, HTML, and CSS. Relevant languages to be used to support frontend are Ruby, PHP, and Java. | Some of the relevant languages to use are JavaScript, HTML, and CSS. Languages to support the frontend among other things are Java, Python, and Ruby. A relevant tool to use in mobile devices could be, for example, swift. |

**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**: To help The Gaming Room expand Draw It or Lose It to other computing environments, I would recommend a cloud-based platform. Using a cloud-based platform will not only provide flexibility, but also scalability and easy deployment across different environments such as desktop computers, mobile devices, and cloud gaming platforms among others. An example of a cloud platform that has been popular among many industries and the gaming industry specifically is Amazon Web Services or AWS for short. Amazon Web Services provides services that would benefit our client in developing his project, these services being Amazon EC2, Elastic Load Balancing, Auto Scaling, and AWS GameLift among others. The different services would provide:

• Virtual servers in the cloud with easy instance customizations

• High availability and distribution of high incoming traffic across different EC2 instances, improved game performance, and uninterrupted gameplay.

• Adjustment of EC2 instances based on the user demand would ensure that their game can handle the different user loads.

• Multiplayer game session hosting through the simplification of the process of deployment, operation, and scaling of the different game servers in the cloud.

2. **Operating Systems Architectures**: When recommending Operating Systems Architectures, we want to ensure we consider those that are supported across different platforms to reach a wide audience. Some of the Operating Systems Architectures that I would recommend to our client are Windows, Android, and iOS. First, Windows is one of the most dominant desktop and laptop computers in the market. It is also the most widely used version and, through the use of Windows as an Operating System Architecture, The Gaming Room will be able to reach a wide range of users. Next, Android is one of the most popular mobile operating systems worldwide. Choosing Android as one of the operating system architectures would allow tablet and smartphone users to play the game which would increase their audience. Finally, recommending iOS as one of the operating system architectures would help our client reach Apple mobile device users as iOS has a significant market share worldwide.

3. **Storage Management**: To accommodate the different game data across the various platforms, I would recommend The Gaming Room use cloud storage, cross-platform compatibility, file formats, and platform-specific APIs as part of a storage management approach. First, cloud storage would allow our clients to store and manage game data. Cloud storage offers our client ease of integration across different computing environments, scalability, and accessibility. Next, cross-platform compatibility offers a storage management solution compatible with various platforms and operating systems and the possibility of storing and retrieving data continuously across different computing environments. Additionally, file format is an approach that should be offered to our clients. This approach allows for the game data to be stored using file formats that are compatible with various platforms. These file formats can range from JPEG or PNG for pictures or WAV and MP3 for music. Using this approach ensures that the different game assets as well as data will be accessed across different computing environments. Finally, platform-specific APIs can help our client handle storage management on various platforms. Using platform-specific APIs would help our client interact with the underlying storage system effectively.

4. **Memory Management**: Memory management is very important when trying to expand our client’s game to other computing environments. It is important because The Gaming Room will need to have a memory management strategy that helps them accommodate the game’s memory requirements across the different platforms. To do that, I would recommend catching and preloading as well as cross-platform memory considerations. First, caching and preloading allow for them to store frequently accessed data in memory or game assets. This helps ensure that users have smooth gameplay, and it minimizes the game loading times. Finally, I would recommend cross-platform memory considerations as a practice to use during this project. This helps the various memory management practices to be compatible with the various operating systems and platforms they will be utilizing. With this practice, one has to ensure that tests and memory optimizations are performed on each platform to ensure optimal performance and stability.

5. **Distributed Systems and Networks**: To implement a distributed systems and networking approach is very important as this will enable seamless communication, coordination, and synchronization between the various components of the game across different platforms. To do that, I would recommend a client-server architecture as well as RESTful APIs. First, a client-server architecture will allow for game clients from different platforms to connect to a central server. This server would be able to handle matchmaking and game logic between players among other things. It also ensures consistent gameplay across platforms, which players will appreciate. Finally, RESTful APIs would facilitate communication between the server and the game clients. This would provide a scalable and standardized approach to exchange data over the network which would facilitate the integration of game functionality across the different platforms that they will be utilizing.

6. **Security**: It is very important to implement security practices that will prevent unauthorized access and will protect our client’s data. Some of the practices that I would recommend are secure user authentication, encryption, and secure network communication. First, authentication would ensure that players across the different platforms verify their identity. Some of how they could identify themselves could be through multi-factor authentication or strong password policies. Next, encryption would protect the data that would be transferred over the network. SSL or Secure Sockets Layer would help encrypt communication between server and player and this would ensure that data is protected and confidential. Finally, secure network communication can be implemented by including firewalls, IPS or intrusion prevention systems, and IDS or intrusion detection systems. This will protect our clients from unauthorized networks accessing our client’s data and attacking their systems.

**References**

Horne, K. (2023, June 1). *Which OS should you choose for your web hosting server?*. Digital.com. https://digital.com/best-web-hosting/operating-systems/

Long, R., & 23, I. E. on O. (2022, October 8). *Windows hosting vs Linus hosting which is better for you?*. Web Design Leads & Resources. https://www.omegaweb.com/windows-v-s-linux-v-s-mac-web-hosting/