

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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)4

[**Domain Model**](#_8h2ehzxfam4o)4

[**Evaluation**](#_2o15spng8stw)6

[**Recommendations**](#_m8aleynsvzvc)10

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/19/21 | Sorosh Khalili | The initial version of the software design template |
| 2.0 | 10/03/21 | Sorosh Khalili | Update of Operating Platforms Evaluation Section |
| 3.0 | 10/17/21 | Sorosh Khalili | Present the Solution 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)

The Gaming Room wants to develop a web-based game that serves multiple platforms, based on their current Android-only game application, Draw It or Lose It. A game consists of four one-minute rounds of play. The game application renders images from a large library of stock drawings as clues, and teams compete to solve the puzzle by guessing what is being drawn. Fully rendered drawings completion at a steady rate happens at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have a chance to offer one guess each to solve the puzzle with a 15-second time limit.

**Requirements**

Technical requirements:

* A web-based version of the current game with the ability to serve multiple platforms.
* The web-based version of the game needs to have all the functionalities of the current Android game application.
* 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)

* To create a web-based version of the current game that can serve multiple platforms, the developing team needs to inspect the current Android game application’s codes to find the best approach to effectively implement all the features and functionalities of the game.
* The web-based version of the game needs to be compatible with all browsers and operating platforms without any compromises and provides a seamless and streamlined user experience.
* Since Draw It or Lose It, is a multiplayer game with one or more teams involved, this game needs to be capable of handling multiple teams with multiple players. Therefore, it must distribute the system’s resources evenly between the teams and players and keep track of who is playing at each moment to assign them the appropriate resources.
* The web-based and multiplayer nature of the game makes it essential to always maintain the network and internet connection. Therefore, it needs to have the ability to preserve and save the game status and alert the user in the event of disconnection.
* The web-based version of the game application must have the ability to detect the type of device that the user is using to access the game and adjust the game display and performance based on the device’s display size and its hardware specifications.
* To maintain the game and team names’ uniqueness, the application must have the ability to store all the previous and current game and team names and constantly check to ensure that inputted names are not previously picked by any other users.
* To ensure the full functionality of the web-based version of the game on all the operating platforms, the quality assurance team requires to perform a substantial amount of testing to assess the game’s compatibility. Consequently, extensive testing can negatively impact both the budget and the release date of the game.
* The web-based version of the game needs to check for any other instance of the game in the memory when a team or player joins the game for the first time to play and only allow the teams and players to create an instance of the game if there are no other instances already existed. If there is an instance of the game in the memory, it should prompt the team and player.
* The unique characteristics of web-based game development require multiple developing teams familiar with the client-server game development in different operating platforms as well as their development kits. As a result, having multiple developing teams can potentially increase the project’s budget.
* Since the game, Draw It or Lose It, has an application already available on the Android operating platform, the web-based version of the game on the Android operating platform should have the ability to present either of these two options to the user and proceed to either web-based or application version of the game based on the user’s selection.

## [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)

* A class can have private and public properties in object-oriented programming.
* The private property is depicted by the negative sign (-) before the name of the property and is hidden from other classes; therefore, the private property is not accessible by other classes.
* The public property is depicted by the positive sign (+) before the name of the property and is visible to other classes; therefore, the public property is accessible by other classes.
* In object-oriented programming, inheritance is a relation between the superclass and the subclass which enables the subclass to utilize its superclass’s public properties.
* To prevent other classes from having access to some of the class’s properties, those properties need to be hidden. Therefore, the class defines those properties as private. This concept is being called, encapsulation. The class then provides some public properties for other classes called getters to retrieve and setters to set the value of its private attributes.
* Polymorphism happens when a subclass changes the way that an inherited method from its superclass behaves and implements it to act in a certain way.
* With abstraction, any class that utilizes the abstract class is being provided by sets of guidelines and rules. Then the class can use those guidelines for its implementation.

Some examples of the previously explained concepts present in the following UML diagram would be as follow:

* The Entity class is the superclass (parent) for its three subclasses (children), the Game class, the Team class, and the Player class. The Game class, the Team class, and the Player class inherit the Entity’s public properties. (There is a hollow arrow from the subclass to the superclass.)
* The Entity class has private attributes (id, name) and one private method (Entity()). Because there is a negative sign before the name of those properties.
* The Entity class has public methods (getId(), getName(), toString()). Because there is a positive sign before the name of those methods. Methods that start with word get, are getters for the Entity class.
* An example of the polymorphism is the method toString() from the player class because this class overrides it to behave differently.
* There is a zero to many (0...\*) relation between the Team and the Player classes because a team can have zero to many players.
* There is a zero to many (0...\*) relation between the Game and the Team classes because a game can have zero to many teams.
* There is a zero to many (0...\*) relation between the GameService and the Game classes because a game service can have zero to many games.
* The default constructor of the GameService class (-GameService()) is defined as private (negative sign before the method) to ensure only one instance of the GameService is in the memory at any given time.
* To ensure that only one instance of the GameService is present at any time and prevent more than one instantiation of the GameService class, the singleton design pattern has been utilized. The singleton design pattern restricts the instantiation of a class to only one instance at any time. To achieve this goal, the default constructor of the GameService class (-GameService()) is defined as private (encapsulation), and the accessor(getter) for this class (getInstance()) has been provided to retrieve the only instantiated instance of the GameService class. The getInstance() method creates an instance of the GameService class only if there is no instance of the class already existed.
* To ensure the uniqueness of game, team, and player name and identification, the following steps have been used and implemented:
  1. The Entity class has been defined as a base class that ensures the uniqueness of the name and the identification attributes for Game, Team, and Player classes. This is an inheritance type relation in object-oriented programming that prevents the redundancy and waste of memory.
  2. The iterator design pattern has been utilized to check for the existence of entered name and identification in Game, Team, and Player classes. Using the iterator design pattern implementation ensures that each entered name and identification are unique to prevent conflict in the program.

**"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** | Apple’s proprietary operating system has a high licensing cost, and the operating system’s use agreement prevents its installation on any hardware other than apple’s provided ones. Building upon the UNIX platform makes macOS a stable operating system to use as a server. Since macOS is fundamentally built upon UNIX structure, it also inherited UNIX’s high-security level as well. Graphical user interface (GUI) makes using the operating system and setting up a server easy. Although macOS is highly scalable but adding new servers to scale up the hosts will increase the cost and make the solution to be very expensive due to the high licensing cost.  Modular design architecture. | Free licensing since it is an open-source operating system. It can be used on any hardware which at least meets its minimum requirements. Building upon the UNIX platform makes Linux a stable operating system to use as a server. Security has been one of the most important pillars of the Linux operating system creation since its beginning, therefore, Essential security features such as least privilege access are enabled by default. Free licensing and excellent security make Linux an extremely common choice as a server. Some distributions have a graphical user interface, but the terminal (the command line) must be used for most of the server setup process which makes it hard to use. | Microsoft’s proprietary operating system has a high licensing cost. It can be used on any hardware which at least meets its minimum requirements. Graphical user interface (GUI) makes using the operating system and setting up a server to be easy. It is very vulnerable to security threats. It is not as stable as macOS or Linux. Although some security features are enabled by default but other essential security features such as least privilege access is not enabled by default. Although Windows is highly scalable but adding new servers to scale up the hosts will increase the cost and make the solution to be very expensive due to the high licensing cost. | An open-source software based on jetty called i-jetty is a web container for hosting a mobile web server. A mobile web server consists of a gateway application running on a computer with internet access to provide DNS configuration and a connector application running on the mobile device. It has been discontinued. |
| **Client Side** | Developers’ familiarity with popular web browsers keeps the developing cost and time moderately low since the game is already available on the Android platform and it needs to be converted to the web-based version with REST API capability. A Developer with an Intermediate expertise level can develop this project. Popular web browsers are Safari, Chrome, and Firefox. | Developers’ familiarity with popular web browsers keeps the developing cost and time moderately low since the game is already available on the Android platform and it needs to be converted to the web-based version with REST API capability. A Developer with an Intermediate expertise level can develop this project. Popular web browsers are Chrome, Firefox, and Edge. | Developers’ familiarity with popular web browsers keeps the developing cost and time moderately low since the game is already available on the Android platform and it needs to be converted to the web-based version with REST API capability. A Developer with an Intermediate expertise level can develop this project. Popular web browsers are Edge, Chrome, and Firefox. | Developers’ familiarity with popular web browsers keeps the developing cost and time moderately low since the game is already available on the Android platform and it needs to be converted to the web-based version with REST API capability. A Developer with an Intermediate expertise level can develop this project. Popular web browsers are Safari, Chrome, and Firefox. |
| **Development Tools** | For developing the client-side (frontend) part of the application, developers need to utilize HTML, CSS, and JavaScript. For developing the server-side (backend) part of the application, developers need to utilize PHP and Java. Database implementation requires SQL language. IDEs such as Eclipse, IntelliJ IDEA, and Visual Studio code are just a few of the very popular IDEs | For developing the client-side (frontend) part of the application, developers need to utilize HTML, CSS, and JavaScript. For developing the server-side (backend) part of the application, developers need to utilize PHP and Java. Database implementation requires SQL language. IDEs such as Eclipse, IntelliJ IDEA, and Visual Studio code are just a few of the very popular IDEs | For developing the client-side (frontend) part of the application, developers need to utilize HTML, CSS, and JavaScript. For developing the server-side (backend) part of the application, developers need to utilize PHP and Java. Database implementation requires SQL language. IDEs such as Eclipse, IntelliJ IDEA, and Visual Studio code are just a few of the very popular IDEs | Application development requires personal computers due to their flexibility and ease of use than mobile devices. XCode is the development tool available only on Apple computers for developing iOS applications. Popular languages are Objective-C and JavaScript. Developing iOS applications requires Swift language. IDEs such as XCode for iOS development and eclipse and Visual Studio Code and IntelliJ IDEA |

## Recommendations

Depending on a company, there are several criteria involved in picking the right computational solution for their needs. Capital is one of the companies’ major deciding factors which substantially impact the implementation of their seeking solutions. Their goal is to always maximize the return value of their investment. However, capital should not negatively affect the other important aspects, specifically the security of the solution. Security is an essential part of any computational solution. It determines the reliability of the solution and whether or not it is the right one for their needs. Having these important factors in mind, the recommended solution for The Gaming Room’s Draw It or Lose It game application has been presented here.

1. **Operating Platform**:

Linux is the recommended operating platform for the Draw It or Lose It game application. Because this operating platform provides high security and relatively low acquisition cost simultaneously. Although there are some benefits in utilizing an in-house (on-premises) Linux server for the Draw It or Lose It game application, the possible lack of required resources and perhaps experienced staff to set up and maintain such a server makes it difficult to suggest an in-house (on-premises) server. Therefore, the cloud-based Linux operating platform is the recommended operating platform for the Draw It or Lose It game application. Utilizing the cloud-based Linux operating platform streamlines scalability and increases efficiency without adding extra complexity. It also results in maintenance cost reduction since the cloud service provider owns the infrastructure and consequently, maintaining the system is its responsibility. In addition to those aforementioned benefits, the cloud-based platform offers and promotes per need basis service and hardware selection options which leads to a performance-optimized and cost-efficient solution for The Gaming Room’s game application.

1. **Operating Systems Architectures**:

Linux is an open-source operating system quite similar to the UNIX operating system. Linux operating system consists of three main components which are the kernel, system library, and system utility. The kernel is the core component which the Linux operating system does not function in its absence. It is responsible for managing and interacting with the entire system resources and has direct unrestricted access to the system’s hardware. Kernel sits right above the hardware layer. System libraries facilitate kernel’s feature access to other programs. These special functions do not need code access rights of the kernel module to implement the majority of the operating system functionalities. System utilities are responsible to perform specialized tasks. The system space is the environment for administrative-level tasks such as software configuration and installation. The system libraries and utilities facilitate administrative-level tasks within the system space. The Linux operating system also contains the shell (usually as a form of a command line), processes running (dormant when not required) in the background responsible for handling service requests, and a desktop environment for user interaction. The server version of the Linux operating system has been designed toward advanced and experienced users having performance and productivity in mind. This fact leads to the omission of the graphical user interface and the desktop environment. Consequently, a high level of expertise and vast knowledge of the Linux operating system’s command line is required to set up and maintain the server environment of the Linux operating system. Utilizing a cloud-based Linux operating system facilitates and greatly simplifies the process. The ideal solution is platform-as-a-service (PaaS) which is a specific form of cloud-based computing. The cloud service provider provides and maintains the hardware and the operating platform in this model and the user only manages application and data. The Paas model is a suitable option for programmers and developers which allows them to focus on application development and relieve them from designing, building, and maintaining the required infrastructure.

1. **Storage Management**:

Storage management refers to the processes and software which improve the performance of data storage resources. In Linux, almost everything is represented by a file including system storge that is represented in the system as a file in the dev (device) directory. The benefits of accessing storage directly are better support for virtualization and quick load time. To prevent the need for inessential input/output sequential data reads and to solve fragmentation stemming from the contiguous allocation that creates storage waste, Indexed allocation schemes can be utilized. Linux uses a unique identifier for every file.

1. **Memory Management**:

The memory management unit (MMU) is responsible for managing the memory in the system including implementation of virtual memory and demand paging, memory allocation for both kernel and user space programs, and mapping of files into processes address space. Linux operating system uses techniques such as demand paging and virtual memory to be able to execute partial memory utilization which helps to run many applications simultaneously. These aforementioned techniques solve the memory size problem and improve the application performance and responsiveness. But page fault can happen as a result of demand paging execution, and it must be addressed accordingly to prevent application performance issues. Linux splits resident processes on the storage unit into pages (small logical segments) and brings them into the memory when it is time for their execution. To be able to find application processes, Linux uses a table called page table which is located between the physical and virtual space for mapping memory addresses and following available pages in actual memory. Linux also uses swapping to increase the amount of available memory but swapping impacts the system performance. The Least Recently Used (LRU) algorithm is Linux’s solution to the page swap performance issue.

1. **Distributed Systems and Networks**:

Utilizing PaaS cloud-based platform relieves the developers from system maintenance and lets them focus on application development and data management which can result in performance improvement of the game. Providing a high-quality product increases the users and consequently the success of the company. Therefore, the company can scale its cloud-based system resources to answer the new demands. The Gaming Room company can utilize more servers to balance the new demand and distribute the load evenly across extra servers to ensure a satisfying user experience. Repetition and load balancing increase system reliability. It also reduces extra pressure on servers that can cause a system failure in the event of overloading. A firewall is the added security feature of the cloud-based operating platform that can reinforce an extra level of security. One of the many benefits of having a Paas cloud computing platform is that its service provider (administrator) ensures proper optimization of the load balancing mechanism to maintain the responsiveness and performance of the servers. Since the application’s database concurrently runs on all servers, these instances need to be updated constantly across all servers to maintain the most up-to-date information of the application. Client and server communicate with each other with HTTP protocol on the internet. Using unique identifiers for each session can help developers to leverage stateless characteristics of the HTTP protocol. The key to preventing the connectivity issue is to implement redundancy in the distributed system. Redundancy helps the system to balance the load and direct customer traffics to other servers in the event of a server failure. This solution applies to the database as well since the presence of multiple instances of identical databases can ensure the continuous performance of the application in the event of a database failure.

1. **Security**:

To ensure the security of a software solution and provide a secure product regardless of the operating platform, multiple security measures such as authentication and authorization must be put in place. Although the basic authentication method for this solution is to request for username and password, this basic method can be further advanced by adding options such as two-factor authentication or single-use password as an extra layer of security. To handle the authorization for each user utilizing least privilege access is a suitable solution since it is enabled by default in Linux operating system and ensures that access is granted based on the user’s defined role in the system. Utilizing REST API can provide a connection between client and server which is stateless and only needs the identifier to be cached on the client-side. Application identifier must be unique for each session and must be removed from client and server after the game session is concluded. Highly sensitive information such as user credentials must be encrypted or hashed before storing in the database for protection and safekeeping which provides an extra level of security in case of unauthorized access to the database. Enabling data encryption before transmitting from the client’s browser increases the safety of data and prevents any damage in the event of hijacking the session.

Processes in Linux have virtual address space independent from one another which increases the security of the process. Discretionary access control (DAC) is a Linux security feature that safeguards files and directories by allowing the owner to grant appropriate file access to the other users.

**References**

A. (2020, September 10). *Tutorial: Beginners guide on linux memory management*. GoLinuxCloud. https://www.golinuxcloud.com/tutorial-linux-memory-management-overview/

Abraham, S., Galvin, P. B., & Gagne, G. (2009). *Operating System Concepts, 8th Edition*. John Wiley & Sons, Inc. https://www.oreilly.com/library/view/operating-system-concepts/9780470128725/

*Architecture of Linux - javatpoint*. (n.d.). JavaTpoint. https://www.javatpoint.com/architecture-of-linux

Ellingwood, J. (2021, April 25). *An Introduction to Storage Terminology and Concepts in Linux*. DigitalOcean. https://www.digitalocean.com/community/tutorials/an-introduction-to-storage-terminology-and-concepts-in-linux

*What is Linux?* (2019, February 27). RedHat. https://www.redhat.com/en/topics/linux/what-is-linux

Wikipedia contributors. (2021, October 23). *Cloud computing*. Wikipedia. https://en.wikipedia.org/wiki/Cloud\_computing

Wikipedia contributors. (2021, October 15). *Serverless computing*. Wikipedia. https://en.wikipedia.org/wiki/Serverless\_computing