.

Draw It or Lose It

# **CS 230 Project Software Design Template**

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 | 09/18/21 | Corey Nance | Development of game application environment |

**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 has an app game called Draw It or Lose It and wants to develop a web-based version of this gaming app. The staff at The Gaming Room do not know how to set up the environment needed to develop this game. The requirements that are to be met is, 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 and only one instance of the game can exist in memory at any given time which can be accomplished by creating unique identifiers for each instance of a game, a team, or the player. The task is to advance and streamline the development of the web-based gaming app while keeping these requirements in mind.

## [Design Constraints](#_2et92p0)

The first design constraint developed without knowing the hardware requirements. Making the program adaptable or portable would help with adaptation when hardware requirements arise. Software wise, this application is to run in the browser so another constraint could be using a language that easily interacts with the DOM. The program would also have to be adaptable to be able to work in a variety of browsers like Google Chrome, Firefox etc. Also, because this is a game application that will have multiple users, utilizing a web server to run the application. On the procedure constraints, one would be using a data base to store all user data and could store the images needed as well. Another constraint could be using certain API’s to pull images or data from. With algorithm constraints could be using algorithms like lazy loading techniques to help quickly render needed images or using sources like JSON web tokens or Passport for user authentication and security. With security, one constraint would be setting up the code to have a client side and a sever side where all data that needs to be secure would be in the server-side code. This can also keep unwanted access to the data base or API as well as user information.

## [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 Gaming Room UML Diagram first starts out with the Entity class as the Parent class where the child classes Game, Team, and Player inherit their name and ID from. Classes Game, Team, and Player extend the Entity class showing inheritance. With Entity having variables ID and Name as private and using public methods and constructors in order to access these variables, and child classes, Game, Team, Player having their Lists also set to private but having methods and constructors also set to public makes the class variables encapsulated or hidden which is a display of encapsulation. This makes the glass very portable to where the Game service class is attached to set up the singleton. The singleton method is achieved by the default game service constructor as well as variables are all private which allows for only one instance of the game service to be in memory. This Is also done with the default constructor of the Entity class. This forces any call to the Class to be forced to go through the public constructor and methods that can have the ability to have checks to make sure every instance of the classes are unique. The program driver is where the main class is held to invoke all actions and can use the singleton tester as well to test out the game service class.

**"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** | Mac is a operating system that has features from UNIX. These features help make MacOS very stable and high performing just like the Linux Operating system. What sets it apart from Linux is the ease of use with its GUI style desktop as well as the support system behind Mac products. With all that comes the downside to Mac. Mac tends to be on the higher side of cost because of the proprietary software and high costs of licenses. (ESDS, 2021) This makes it the most expensive of the operating systems. Mac is stable, easy to use, and has loads of support but is also comes at the highest cost. | Linux is a operating system that has features from UNIX. These features help it to be a very stable and high performing server. Linux is the choice based on price. This server tends to have a lower cost due to having a free open-source operating system. The biggest drawback to this operating system is that it is very difficult to use. The low-cost factor is a great advantage for enterprise level companies. (ESDS, 2021). This is a great choice for developers that can configure an Apache or NGINX server which is open-source and lower cost. (Dobran, 2018) | As a server, windows is almost set in the middle of Mac and Linux. Its a lot pricier than Linux due to proprietary software and licensing which drives up price over Linux. (ESDS, 2021) Though not as expensive or restrictive as MacOS. MacOS as a server has either 10 clients or unlimited clients whereas windows do not put such restraints which can account for a cost savings. (Faas, 2007) A huge drawback to windows servers is that they are not as stable as a Linux server or a MacOS server. It can be a lot less expensive to maintain than a MacOS. Linux has a steep learning curve but with Windows you don’t need to be an expert at programming. Windows also has a nice server package for support and regular system updates and security fixes. (Dobran, 2018) | From an enterprise level, using a Mobile Device as a web server isn’t the best choice. This will be a good choice for someone looking for a low-powered device to run a website and still save space on a local machine. One can utilize hosting apps like I-Jetty which is an open-source web container serving up Java-based apps. (Wikipedia, 2021) On android devices, one can use app called Tiny Web Server which allows the device to run low traffic websites. (Cawley, 2019) The major downside or constraint is the low number of resources or memory which makes this only limited to no enterprise level or even just very low traffic hosting. Main advantage is saving memory and being mobile. |
| **Client Side** | | Apple products are known for caring a premium price tag. With that price tag come loads of support and R&D. Mac OS products tend to have their own guidelines for development along with associated fees. Any application developed for the client-side Mac should be straightforward for a browser like Safari, Apples own browser but added cost and developer expertise as well development time and the purchase of additional extensions or IDEs will have to take place to adapt the application to run on a Windows browser like Microsoft Edge or third-party browsers like Google’s Chrome and the open-source browser like Firefox. Developers tend to use this but enterprise companies go someplace else. | Linux is a very low-cost option due to its open source and wide range of available free to use software. (Gewirtz, 2019)  Linux is the more difficult of the three viable platforms which would make the training expense and time associated with it the focus. With this platform which may or may not overshadow the licensing cost of Windows and MacOS. Due to the open source, one might find very little licensing cost as well as extensions to help adapt the application for multiple browsers. This one here seems to be the most common of the three when it comes to big business or enterprise services. | Windows most outstanding feature is choice. No matter what computer, configuration, or form-factor one wants, windows has it. (Gewirtz, 2019) The training cost should be relatively low due the amount of Windows machines out there and with its most common feature of choice. Windows is readily compatible with almost every browser so licensing cost should be lower and only an extension to run safari or MacOS on a virtual machine would be the main factor of driving up cost. Overall windows give a system that is almost as easy to use as MacOS, but have licensing cost closer to Linux and may have to download an extension to run MacOS with safari. These extensions would add cost which makes Windows fall under MacOS for price but is not as expensive a Linux. | Mobile client side is almost nonexistent. It has insufficient processing and cant support in extensive programs for developing like Java. This relies more on an app-based structure where most of the computing would be done by cloud or internet frameworks. (Cawley, 2019) |
| **Development Tools** | | The top 12 coding programs for mac is FlexiHub, Homebrew, Xcode, iTerm2, Sublime Text, NetBeans, Tower, SourceTree, Atom, Visual Studio Code, Dash, and Go2Shell. (Weis, 2021) | The top 8 development tools for Linux is Seamonkey, Quanta, Bluegriffon, Sublime Text, Visual Studio Code, Pinegrow, Pingendo, Aptana Studio. (Tolstoy, 2019) | This one I can base of personal knowledge, I use, Eclipse for Java and C++, Visual Studio for C++, Visual Studio Code for HTML, CSS, JavaScript and more. Git Bash for command line, PyCharm for Python, MySQL Workbench to help with the database, Git desktop for easier access to my repository. The list can just go on from here. | This section is tough, most mobile development happens via a computer using something like React Native or a process called mobile first development. I did not find anything to do with software development on a mobile device but rather frameworks that make mobile development easier. Some are BuildFire.js, Mobile Angular UI, Xamarin, Appcelerator, React Native, and Dropsource to name a few. All seem to rely on other languages that are other platforms like Mac, Windows, or Linux. (buildfire, 2021) Basically trying to use a mobile device on its own does not work as a development environment. Its just not suitable and most, if not all mobile development is done using a computer. |

## 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**:

The Operating platform I would most recommend would the Windows Operating system. The main reason for this recommendation is because it seems to give the best of both worlds without taking on the highest price hardware. With windows, its most outstanding feature is choice. No matter what computer, configuration, or form-factor one wants, windows has it. (Gewirtz, 2019) Windows also has a nice server package for support and regular system updates and security fixes. (Dobran, 2018) Overall with the Gaming Rooms requirements to be hosted on a server and run in the browsers, the windows operating system seems to be compatible with a wider verity of development environments as well as browser support. With all the latest updates as well, windows seem to have been able to make their servers a lot more stable and reliable.

1. **Operating Systems Architectures**:

The Windows operating system is set up using a GUI style setup to interact with the kernel. Windows most outstanding feature is choice. No matter what computer, configuration, or form-factor one wants, windows has it. (Gewirtz, 2019) This choice comes from the many different ways you can use windows. One can use the GUI to interact with the files but can also use the command line by use of Command Prompt, PowerShell or many other compatible programs like Git Bash. With the type of application that Draw It or Lose It is, I would recommend a Space-based architecture/pattern The space-based architecture is designed to avoid functional collapse under high load by splitting up both the processing and the storage between multiple servers. (Wayner, 2015) This style is best for high volume types of data which will help Draw It or Lose It be easily scalable.

1. **Storage Management**:

Storage would be used by first using a database like MongoDB to house user and game data. Next the application would utilize the computers hard drive to store all its inactive executable files like launch instructions as well as actionable files to give instructions as to what files should be sent to memory. Draw It or Lose It’s storage management would be in the form of initial data would be saved to a database. When the game is downloaded, the main application files will be saved to the hard drive which these files can contain, launch files or executable files, browser support configurations, and application instructions. Then the application will utilize the RAM for data like the images, active users, active game sessions and relevant instructions.

1. **Memory Management**:

The game Draw It or Lose It must utilize memory to house images, users, as well as the games executable files. Starting with the images, which will be 200 images at about 8 megabytes in size and totals out to about 1.6 gigabytes of storage needed to house the pictures. To maintain performance the application should load all images into RAM. RAM space will also be utilized to load in the executable that are needed like instructions for the application. Having quick access to ready data located in memory is a must to help reduce loading times which in turn enhances user experience. The first action would be to use a database setup to house all the images and user information. The executable files would be housed in the hard drive and moved into memory when needed. Within these files would contain the instructions on how to launch the application as well as how much memory needs to be allocated for the number of active players as well as active games and instructions as to which active games get which images.

1. **Distributed Systems and Networks**:

Here we would have the client communicate with the server by way action calls via the URL path. Because Draw It or Lose It is run in the browser, we can set up the server side functions to look for URL endpoints to do access data. These functions would use GET methods to pull data from the database when a certain endpoint is reached. The PUT, POST, or DELETE methods maybe be accessed by a button or link within the website. Within these functions would have the middleware to check for user sessions, authentication, and user roles before allowing access to each method. Doing it this way allows the application to communicate between various platforms without changes to the code.

1. **Security**:

Let’s start with the client side, the initial security would be having a username and password to verify the person is who they are. Then use JSON web tokens and user sessions to help with securing the client side with time outs for inactivity. Once the username and password are entered its either matched to what is stored in a database already hashed or if it’s a new signup, the password is then hashed and saved into the database. The JSON web token is created along with a user session which is also checked in the server just in case it is timed out due to inactivity or user sign out. These measures are recommended for handling client side authorization needs which again isn't the most secure but this with the severs defenses should work pretty well.

On the server side which will house and protect access to the database is where most of the defense would be. First layer will be creating an encapsulated class where access to class variables can be controlled but not manipulated. The next layer would use a type of authentication like the RolesAllowed annotation that only allows access based on the type of role. Initial access to these roles would also have other types of authentications like checking username and password and/or user session via the JSON web token for logging in which is created or checked fist on the client side, but also checking that on the database side is another layer. The fourth layer would be hashing and encrypting passwords before saving to a database. Programs like Bcrypt can be used for hashing the passwords and Passport for authentication. Gaining access to the data based by use of functions that can each check for validation before allowing access will also help to keep the database secure and adds an additional layer.

Final measure to help make this application secure is by separating the client side and the server side files. Creating a session with a time out for the user as well as a user object. Passing that user object with the password for validation, if its valid then access is granted but if not, then access is denied. That user object is validated by checking user password against saved hashed password using something like Bcrypt and checking user session. If everything clears, then now the role of that user is checked to see what level of access is granted. Depending on that access is where the server allows for read, write, delete, or update privileges with the database.

References

Buildfire. (2021). Mobile App Development Tools: A Detailed Comparison. <https://buildfire.com/mobile-app-development-tools/>

Cawley, Christian. (2019). How to Turn an Android Device Into a Web Server. <https://www.makeuseof.com/tag/turn-android-device-web-server/>

Dorbran, Bojana. (2018). Windows Server vs Linux: The Ultimate Comparison. <https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers>

ESDS. (2021). Battle of Web Hosting Platforms – Mac Vs Linux Vs Windows. <https://www.esds.co.in/blog/battle-of-web-hosting-platforms-mac-vs-linux-vs-windows/>

Faas, Ryan. (2007). Computerworld: Leopard Server vs. Windows Server. <https://www.macworld.com/article/188036/servers-4.html>

Gewirtz, David. (2019). Windows, Mac, or Linux? We compare the pros and cons of these computing platforms. <https://www.zdnet.com/article/windows-mac-or-linux-we-compare-the-pros-and-cons-of-these-computing-platforms/>

Make Me A Programmer. (2021). Do Programmers Use Mac or PC?. <https://makemeaprogrammer.com/do-programmers-use-mac-or-pc/>

Tolstoy, Alexander. (2019r). 8 best dev tools for Linux. <https://www.creativebloq.com/features/tools-for-linux>

Wayner, Peter. (2015). The top 5 software architecture patterns: How to make the right choice <https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice>

Weis, Olga. (2021). List of 12 best Mac apps for developers. <https://www.flexihub.com/best-mac-developer-tools/>

Wikipedia. (2021). Mobile Web Server. <https://en.wikipedia.org/wiki/Mobile_Web_Server>