

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template**](#_gjdgxs)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_3znysh7)2

[**Executive Summary**](#_2et92p0)3

[**Design Constraints**](#_tyjcwt)3

[**System Architecture View**](#_3dy6vkm)3

[**Domain Model**](#_1t3h5sf)3

[**Evaluation**](#_2s8eyo1)3

[**Recommendations**](#_3rdcrjn)5

## [Document Revision History](#_3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/9/22 | Caleb Jones | Initial declaration of design constraints and software specifications. |
| 1.1 | 11/21/22 | Caleb Jones | Updated information regarding the Evaluation (pg.4) |
| 1.2 | 12/5/22 | Caleb Jones | Updated information regarding the Recommendations (pg. 8) |

**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](#_2et92p0)

The main software design problem being addressed is the transition of the game from an Android application only to become a web based application. The solution is to begin implementation towards a virtual web based gaming environment. This will involve a single instance of the game being hosted on a server. Where players on all platforms can join uniquely named teams to compete in a game inspired by the 1980s television game Win, Lose or Draw. The key factor is the emphasis on scaling from a single android application to a web based one with platform independence.

## [Design Constraints](#_tyjcwt)

A local or cloud hosted server will have to be reserved to facilitate hosting the games. Single and unique game instances and team names will require an emphasis on validating entries. A focus on portability and functionality on most common web browsers, operating systems and mobile devices. The web based environment will have to have the necessary resources allocated to improve performance, load times, processing and general latency.

## [System Architecture View](#_3dy6vkm)

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](#_1t3h5sf)

There is a general ProgramDriver that serves as a starting point and runs the entire application. It uses SingletonTester. This validates the software requirement for there to only be one instance of the game and that names are uniquely defined. Entity serves as the parent class from which the remainder of the classes and functions inherit. Entity also encapsulates the entirety of the game features. Game, Team and Player inherit methods such as getID and getName from Entity. While Entity is the primary parent, GameService acts as a more general facilitator for the games general functionality. Its instance can be called throughout the program because it is static. GameService primarily calls Game and through this, new game lists are created as well as general game information such as name, count, and game, team and player ids. The implementation of the private constructor for GameService to ensure that only one instance of the game exists at one time. Game extends Entity and encapsulates Team by creating a list of Team objects along with making calls to Team to add a team to the list. The toString() method is also present here inherited from Entity. Through polymorphism it can be overloaded to output the desired information. Team extends Entity and encapsulates Player from which a list of Player can be created as well as adding players and an overloaded toString() method. Player extends Entity. While not abstract, it does not encapsulate any other classes. It processes a polymorphed toString() as well as general constructor features for the Player object.

**"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](#_2s8eyo1)

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** | Only Apple can release updates to their platforms. Therefore a fee must be paid to update anytime there is a new version. Mac servers can only be hosted on Mac products. Mac provides the most secure web services. Mac does allow for server-based deployment for the hosted website. Licensing costs can become rather high ranging from $60 to $499 a month. The Gaming Room will likely consider the $499 due to the scalability opportunity provided by the increased capacity. | Linux allows for an open source manipulation therefore keeping features up to date will be free. There are limited resources required to run Linux. Allows for user choice of security measures. Linux does offer server-based deployment for the hosted website. The licensing for Linux can vary due to the customizable nature of the provider. A common server hosting service is Red Hat with premiums of $1299 yearly or there are free options available but this requires additional development focus and lacks in security. | Only Microsoft can release updates to their platforms. Therefore a fee must be paid to update anytime there is a new version. Windows requires high performing hardware to host servers. Frequent updates and server downtimes. Allows for use of Microsoft proprietary web applications. Windows also allows for server-based deployment for hosted websites. The licensing for Microsoft ranges from $501 to $6155. Due to the current scale of The Gaming Room the projection will be around $501 to $1069 with further pricing negotiated with a Microsoft representative. | Utilize a number of different services to facilitate hosting. Therefore there is a selection in terms of cost, functionality and security. There are also open source options. Mobile devices are not likely to have a server-based deployment for hosting websites. This will likely be done through a cloud based service such as Amazon Web Services or Azure. Both offer a generous free trial period. This will allow the Gaming Room to navigate its scale and adjust pricing accordingly. I would indicate a mobile device server hosting is not suitable for our current needs. |
| **Client Side** | Mac web hosting services are user friendly. Time and expertise are minimal in terms of developing functionality to support multiple clients. The cost is a leading factor as the increased functionality brings along a larger price tag. There is a significant level of expertise required for an individual to navigate apple specific software since that is the only equipment able to run the hosted server. Outside of that, the general additional development will be necessary to successfully deploy a compatible application. Mac does allow for applications to be used on non-apple products. | Development could become increasingly time consuming due to the number of possible configurations that will have to be done. Features such as security require high expertise or incur a cost to outsource different features. This initial bootstrapping will impede deployment but once that foundation layer has been established development on the application itself will include general compatibility tweaks to ensure functionality on all platforms. | Development is more streamlined as services are provided and distributed by Microsoft directly. Development considerations include if there will be any embedding of proprietary Microsoft Web applications then this is the only option. Cost can scale as additional features and services are added on. The set up time for a Microsoft based server is relatively low. The benefits are well received in terms of cost if an effective scope was defined prior. Then in terms of the additional development to deploy the application to multiple platforms, nothing extra will have to be established outside of the general compatibility tweaks. | Often allows for Cloud based hosting. A wide range of hosting options to choose from. Cost can vary as well as development time and consideration to factor what service will support all the desired functionalities. While the cloud based features can be leveraged quite well in terms of scalability and general development can be effectively navigated to ensure general compatibility with all platforms. The scope of server hosting in terms of a mobile device often goes beyond the mobile device itself. Therefore other server hosting such as Microsoft, which also has cloud based features, should be considered first. |
| **Development Tools** | Mac servers are based in Apache written with Perl, Ruby and Python. Apache Tomcat and Java virtual Machines are also common. Therefore IDE’s such as Eclipse or Intellij could be utilized. One development team could effectively navigate development. The team would have to have experience working on MacOS as well as some technical knowledge of aspects such as spring and HTML/JS. The licensing for most IDE’s is free although Itellij does have an enterprise version with a fee. | Linux servers are based with Apache. Python and C++ are utilized heavily. IDE’s such as Visual studio which allows for general coding of a range of languages is preferred. Multiple development teams might be necessary. One to focus on the security and bootstrapping of the server. The other team focuses on development of the application. The technical knowledge will have to be extensive in terms of cyber security and server hosting and then in the form of Python, C++ and or HTML/JS for the web application. The IDE’s used will all have effective free versions. | Windows often uses C, C++ and C# to deploy a number of their services. Therefore applications like Visual Studio and Visual Studio Code will be utilized heavily. One development team could effectively navigate deployment. Strong technical knowledge of Microsoft based coding languages will be necessary. The cost of the IDE’s utilized will be free. | As far as mobile devices go they are written with primarily Python and C++. Many of the web application hosting services like Azure utilize many of the common languages like Python and Java, JavaScript. Therefore Visual Studio and Visual Studio Code will be common IDE choices. One strong development team would be necessary. They should have technical knowledge of cloud based applications as well as general coding knowledge. The licensing costs may vary depending on the type of service used. It is also important to highlight that the mobile device itself is not a suitable development environment. The features offered by a mobile device hosted server could be acquired more easily through another operating system and development would be more agile and cost effective. |

## 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 recommended operation platform is Mac. This will allow The Gaming Room to expand Draw It or Lose It to other computing environments. The server side shall be based on a Mac OS to support all client operating systems that a customer could ask for.
2. **Operating Systems Architectures**: Mac OS is a multi-layered structure. It begins with Darwin, the Unix core of the system. Next is the graphical system which contains Quartz, OpenGL and QuickTime. Then the application layer which consists of Classic, Carbon, Cocoa and Java. Then the user interface Aqua. The servers are hosted with Apache and are integrated well with other Mac products. All hardware will have to be approved Mac or Apple functional products to work with the chosen platform. Once this is in place the Apache server will host the Draw It or Lose It application for access across all client operating systems.
3. **Storage Management**: There will have to be physical and virtual structures in place for Draw It or Lose It to be functional. Servers will host the game instances and store player, team and game data accordingly. Personal caches can be implemented for the client specific player data. There will have to be a permanent storage for the current 200 images on file that are used during play. In terms of databases MySQL is recommended for the Mac OS. However, functionality with ODBC and JDBC connectivities are available as well.
4. **Memory Management**: Mac uses kernels to manage memory and functionality. Mac often compresses memory and keeps it in RAM. Due to Draw it or Lose it ‘s game design, much of the images, player and team data will be run through the RAM to reduce load times for frequently accessed user data. Development will focus on creating objects and instances as they are needed. Mac OS offers a built in Garbage Collection feature that should be utilized regularly to reduce memory leaks.
5. **Distributed Systems and Networks**: With Apache being the underlying framework for hosting web applications communication to various different platforms is accomplished with relative ease. While many of the hosting systems will be dependent upon a Mac OS. The client side will enjoy consistent levels of connectivity with minimal downtime from the infrequent server shutdowns. Apache allows for development to access the server through various URI based requests. Configurations regarding REST connectivity can also be leveraged on a Mac OS server with some development time and planning to avoid potential security risks due to exposed endpoints.
6. **Security**: Mac provides the highest level of web based security. From hosting to general use, Mac’s complex development language and systems create an extremely secure environment. Mac OS is also aggressive in its updates to address any potential security risks. The security on the server side is secure through the selection of the Mac OS. Development will then be able focus heavily on securing the client side through access points, tokenization and authentication. User protection can then be leveraged on both fronts. Firstly through a secure server and database. Secondly through a well developed and secure client side.