

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 11/15/20 | Danica Hesemann | Updated Executive Summary, Design Constraints, and Domain Model |
| 1.1 | 11/29/20 | Danica Hesemann | Updated Evaluation |
| 1.2 | 12/13/20 | Danica Hesemann | Updated 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 has developed a gaming app, Draw It or Lose It, which is currently available on Android devices. They want to develop a web-based version of the app but do not know how to set up the environment.

The game will need to incorporate a library of images that act as clues for each puzzle. To begin a game, each player will be assigned to a team, then one or more teams can join a game. Each game consists of four rounds of one minute each. For each round, the application then starts a timer and renders an image from the library at a steady rate for 30 seconds until it is complete while one team guesses the puzzle. If the team fails to correctly guess the puzzle after one minute, the other teams are allowed 15 seconds to guess. Each team should consist of multiple players and have a unique name. Only one instance of each game should exist in the memory at a time. Each game can consist of one or more teams and should also have a unique name. Players should be able to check if the name of their team or game is already in use.

## [Design Constraints](#_2et92p0)

* We will be using Java for the project. We work primarily with Java because it is a universally recognized language that can be used on various browsers with the potential to be compatible with multiple platforms in the future with the use of additional development languages and tools.
* Our web site needs to be built with common web application tools (e.g. HTML5, PHP), using prevalent versions that are supported by all current browsers.
* We will maintain the same actions and functionality as the existing Android application. We will use the same basic design and will not be adding additional features to the game assuming that the Android app is not a single player game without a central server. This will minimize the cost and labor necessary to develop the web application. The web app will also use the same library of images that the Android app uses.
* We will stay within The Gaming Room’s budget to ensure that they can reasonably afford the resources and labor that are necessary for the project.
* We will need to adhere to The Game Room’s schedule. Every effort should be made to work efficiently and stay on track.
* The application developers will need to consider things like bandwidth and internet speed which could affect the performance of the web app.
* The application will need to have a database to ensure that no games, teams, or players are duplicated. Unique names and ID numbers can be stored in the database to make sure that only one instance of each game, team, and player 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)

The ProgramDriver class provides the code that starts the program. It contains the main() function which starts the game service and initializes the game data. The ProgramDriver class uses the SingletonTester class object to test the behavior of the GameService singleton object. Including the SingletonTester class ensures that the program is using a single GameService object, which is crucial for the program to function correctly.

The GameService class provides a singleton service for the game engine. It stores any number of objects from the Game class in an ArrayList called games and provides functions to retrieve or add a Game object to the games list. The Game class holds information about a single game, including the teams involved in each game. It stores any number of Team objects in an ArrayList called teams and provides functions to add a Team object to the teams list, construct a Game object, or print information about a Game. The Team class holds information about a single team, including the players involved in each game. It stores any number of Player objects in an ArrayList called players and provides functions to add a Player object to the players list, construct a Team object or print information about a Team. The Player class holds information about a single player and provides functions to construct a Player object or print information about a Player.

The Game, Team, and Player classes all inherit from the Entity class. This inheritance relationship also means that objects from the Game, Team, and Player classes are polymorphic since they can be classified as an Entity as well as a Game, Team, or Player. Using separate classes for the games, teams, and players keeps the program simple and organized in a way that helps it to run efficiently. If the game service only used an Entity class that incorporated all of the games, teams, and players, there would be a greater risk that the program would run incorrectly. For example, using a function to retrieve the name of a game using its ID could return the name of a team since they would be classified as the same object type.

If all objects are classified as part of either the Game, Team, or Player class, the Entity class is an example of abstraction. I do not see a reason why any Entity objects will be constructed, but the class holds the common attributes between the three child classes and provides more efficient common functions to retrieve an object’s name or ID. Each class has private attributes and public methods to retrieve them, which is an example of encapsulation that prevents a user from changing data values that could affect the functionality of the code. For example, once a Game object is added with a unique name and ID, the user cannot change the data to match another game’s name and ID.

****

## [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** | Characteristics of Mac OS Server include Mac and iOS device management, an included security firewall, $19.99 full access on a Mac, a simple user experience, and compatibility with PHP and Python, and management systems such as BIND, Postfix, PostgreSQL, and OpenLDAP. Some advantages are that it is easy to use and inexpensive. Its weaknesses are that it is for managing Mac and iOS only, and services have been reduced, resulting in a more limited size, limited features, and less scalability. New versions likely do not offer the necessary means to host a large scale, multi-platform game app. | Characteristics of Linux servers include an Open Source system, free licensing, low distribution costs, database management systems, memory size options, network speed options, built-in security measures, Apache or Nginx web servers, and compatibility with Perl, PHP, Python, and Ruby. Some advantages are that it is not very demanding on hardware, it is inexpensive, stable, scalable, and relatively secure. A Linux server should offer methods to host and deploy an application on a large scale with many clients. Some disadvantages are that some common programs may be incompatible with the server, and operation may be more complicated than on Windows or Mac. | Characteristics of Windows Server include built in security measures such as shielded virtual machines and anti-malware software, Linux containers, a well-established proprietary system, memory size and network speed options, licensing prices ranging from $500 to $6000+, higher distribution costs, database management systems, Microsoft IIS web server, and compatibility with ASP.NET and VBScript. Some advantages are its wide availability and compatibility with common apps and programs, it simple and easy to use GUI, scalability, and access to long term support. A Linux server should offer methods to host and deploy an application on a large scale with many clients. Some weaknesses are its complicated licensing and high cost. | Characteristics of mobile devices as servers are server capabilities through web server apps for free or up to $3, and memory size and network speed that is dependent on the device. Some advantages are that hosting on a mobile device is inexpensive and can be good for testing a small web app or using as a home server. Some weaknesses are that hosting on mobile devices is rare, often inconvenient, there is little support, and the capabilities are highly limited by the device’s limitations. While client devices may be able to communicate with the server, it will not have the means to host a multi-player web application across many platforms. |
| **Client Side** | Developing the application on Mac will require a Mac development environment or virtual machine to test the program, which may have a relatively small cost. The team will need to allocate time for adapting the app to a web app, testing on Mac and making adjustments as necessary. The team will need the skills to use and develop applications in a Mac environment, or they will need to plan for the time and cost of bringing in an expert. If they want to release an app for Mac on the Apple store, they will need an Apple developer account, which will cost $99. | Developing the application on Linux will require a Linux development environment or virtual machine to test the program, which may cost a small amount. Adapting the Android app to a web app, testing it on Linux and adjusting as necessary will take time that the team needs to account for. The team will need to be skilled enough with Linux to test and edit the program, or they will need to plan for the time and cost of bringing in an expert. | Developing the application on Windows will require a Windows development environment or a virtual machine to test the program, which may have a small cost. The team will need to account for any time it might take to create the web app from the Android app, test it on Windows, and make necessary adjustments. The team will need to be skilled enough with a Windows Environment to test and edit the program, or they will need to account for the time and cost of bringing in an expert. | The app already exists for Android devices, but time may need to be allocated to test the app with any changes and fix any issues. This may require a Windows or Linux development environment and an Android device for testing. Developing the application for iOS will require a Mac development environment or virtual machine, as well an iOS device for testing, for which the cost will be accounted for if the resources are not already available. The team will need to allocate plenty of time to develop and test the app. The team will need someone with expertise in iOS development, or they will need to plan for the time and cost of bringing in an expert. To release the app on the Apple store, they will need an Apple developer account, which will cost $99. |
| **Development Tools** | The main program will be written in Java due to its popularity and portability. Other useful languages for creating a web application include HTML for web structure and presentation, JavaScript for handling dynamic content on a website, and PHP for communicating with the server. Some tools that may be useful are a free open source IDE like Eclipse or Apache Cordova, or an Apple specific one like Xcode, which is free on Mac, especially if they want to develop an application on the Apple store for Mac. A project management tool like Maven may be useful to manage the compiler. If the development team does not have expertise working with Mac, they may need to add to the team. | Assuming the game is a web application accessed on a browser, the languages and tools for developing on Linux will be largely the same as on Windows and Mac. Useful languages are Java for the main program, HTML for web presentation, JavaScript for dynamic content, and PHP for server communication. Free open source IDEs like Eclipse, Apache Cordova, or Atom can be used for code editing. A program like Maven can be used to build and manage the project. If the development team does not have expertise working with Linux, they may need to add to the team. | The languages and tools for developing on Windows will be largely the same as on Mac and Linux. The main program will be written in Java, HTML should be incorporated web presentation, JavaScript for dynamic content, and PHP for server and network communication. Code can be written and edited on a free open source IDE like Visual Editor, Eclipse, or Apache Cordova. Maven or a similar program can be used to build and manage the project. If the development team does not have expertise working with Windows, they may need to add to the team. | Like on Windows, Mac, and Linux, the main program will be written in Java, and it will incorporate JavaScript for complex methods. HTML and Cascading Style Sheets (CSS) can work together for the presentation of the mobile app. In addition, Objective-C, C++ (Android), or Swift (iOS) may be used to develop the mobile app. The mobile app can be written on free open source IDEs like Apache Cordova, Android Studio (Android), the Android plugin for Eclipse (Android), or Xcode (iOS). If the development team does not have expertise working with iOS, they may need to add to the team. |

**App store costs, licensing costs(client), expertise or skills, time? What would be time?**

## 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**: Linux has low software licensing costs and an abundance of open source software packages that can greatly lower the cost of implementation. The Linux open source software generally has support available at a reasonable price, so that the technical risk from using the software is small. Linux is considered to be a secure operating system that is less likely to be targeted by malicious actors.
2. **Operating Systems Architectures**: The program uses a client-server architecture, where many client devices with varying types of operating platforms can connect to a server that holds a library of images, a database, and an application server containing the program and a REST API, which is essentially a set of functions that allows the client and server to communicate. A REST API requires all information necessary for the call to be included in a request to GET, PUT, POST, or DELETE. The client communicates with the server by making requests to the REST API. A web server receives requests from the client and directs it to the appropriate API call to receive a response for the client from the server.
3. **Storage Management**: The program should store data from each class, such as user information or game data, in a database on the server. The server should have plenty of disk space to store the database, a 1.6 GB library of images, and the program itself or the application server containing it. To manage storage, it is recommended to set aside around twice as much disk space than what the company anticipates will be needed after accounting for growth, and it is recommended to identify unnecessary data in the database and delete it regularly.
4. **Memory Management**: Server memory should be used to retrieve and store player, team, and game data, as well as puzzle images, to be accessed in active games. To prevent memory leaks, the program should use classes with constructors and destructors so every constructor called can correspond to a destructor call once it is no longer needed. Temporary data generated by function calls should either be deleted as part of a function or it should be stored somewhere to be deleted later by a garbage collection process. It is recommended during development and testing to conduct memory usage tests to check for memory leaks after performing numerous operations.
5. **Distributed Systems and Networks**: It is recommended to replicate the application server onto multiple servers and connect them with a load balancer to distribute the work between the servers. This ensures that if one server experiences a network outage or other issues, the load balancer can redistribute duties to the other server(s) and the program can still function. Using this strategy, it is recommended to create a database server to store the database which can also be replicated onto multiple servers in case of problems, using whatever high availability capabilities the database vendor provides. The servers containing the application server can all be at the same location, or they could be distributed in different locations, which could provide an opportunity to use an algorithm to direct each client to the appropriate server and minimize latency, particularly if one of the servers grows disproportionately large.
6. **Security**: To maximize security, the program should account for different roles, and all users should be assigned to a user role, which should be given access only to the information and functions necessary to play the game. Requests made to the server by client devices should be given access only to the necessary data for gameplay. Measures should be taken to ensure the code is as secure as possible and any errors or bugs should be corrected promptly. All classes and data should be kept on a secure server with security measures such as a firewall, anti-virus software, and the minimum default privileges for all roles. Authentication should be handled carefully with measures such as password length or character requirements for users and perhaps multifactor authentication for admins.