

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 | 05/30/2021 | Alice Norris | <Brief description of changes in this revision> |

**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)

Our client, The Gaming Room, wants to develop a game called Draw It or Lose It. The application will run inside of a browser, resulting in a need for an application to manage the game instance, as only one instance can exist in memory at any given time. This application will manage the individual instances of each game, team, and player. Games will have each have a unique name and unique ID. Duplicate names and IDs for games should not be permitted. A team will have multiple players in it, and every team must have a unique name and unique ID. Players will have unique IDs, with duplicate names being possible for players.

## [Design Constraints](#_2et92p0)

1. Web browser compatibility

As the Draw It or Lose It will be run inside of a browser, the game must be compatible with multiple browsers. As of May 2021, ensuring support for Chrome, Safari, and Opera will capture about 90% of web browser by user share, with Chrome and Safari taking highest priority at about 63% and 24%, respectively (*StatCounter, n.d.*). Since Opera uses the same web engine as Chrome, if the game is compatible with Chrome, it should be compatible with Opera as well. The game should be written in HTML, CSS, and Javascript with Angular as a framework, due to its cross browser capabilities, built-in testing, and its lack of proprietary types.

1. Hosting platform

Being a web-based application, a centralized location for game data must be created and maintained. Amazon Web Services (AWS) presents an option for “Single Page Web App Hosting”, for apps that require a single load in a web browser. Backend data is accessed with REST APIs. This provides a complete hosting solution.

## [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 houses the main function, which is the starting point of the program, first creating an instance of GameService. It depends on the SingletonTester class to ensure that only one instance of GameService exists via SingletonTester’s testSingleton() function, preventing games from being assigned to different instances of GameService.

The Entity class serves as the base class for the Game, Team, and Player classes. All of those subclasses inherit its id and name attributes, as well as a constructor and getters and setters for id and name, which are private variables. This provides encapsulation by providing access to the id and name attributes via functions instead of being directly mutable from outside of the class. Polymorphism is implemented here by inheritance, as the objects are both Entities and their own type. Polymorphism is also implemented in the constructor for Entity, as the constructor is overloaded, providing two methods with the same name.

There is a zero-to-many relationship between GameService and Game, Game and Team, and Team and Player. That is, the GameService does not depend on a Game existing, and can have zero or many Games. Game does not depend on having any Teams, and can have zero or many games. Finally, Teams can exist without Player instances, but can also have an arbitrary amount of Player instances. These relationships satisfy the requirement that GameService has Games, a Game can have Teams, and Teams can have Players. Each have an attribute for a unique ID and a name, as inherited from the Entity class.

****

## [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** | macOS includes an Apache web server on every installation with an administration suite, macOS Server, available via the app store. It also has the advantage of a unified hardware and software platform, as macOS is designed specifically to run on Apple hardware. However, the features of macOS Server appear to be geared more toward providing a server to local networks than hosting a website or application. Many of the webhosting features of macOS Server have been removed as of version 5.7.1. MacOS Server also a very small market share for webhosting at <0.1% (Q-Success, n.d.). | Linux, by default has a simpler file structure than Windows, defined by the Filesystem Hierarchy Standard. Linux is the most widely used operating system for serving websites (Q-Success, n.d.). Due to this, modern web technologies are often well-supported for Linux, and a wide variety of hosting software for different web technologies are available. Linux is also free, as is most of the hosting and server administration software available for it. However, each piece of software generally needs to be configured independently. Security, configuration, and maintenance often require specialized knowledge to do well, and graphical interfaces are often separate from the software itself. | Hosting the website and application on Windows will require Windows Server. Windows Server is a more unified software stack as opposed to Linux, with built-in tools for almost every aspect of web hosting. It is also easier to use than Linux, as graphical user interfaces are part of the software. Management can be done via PowerShell, but this is often not needed due to the administration interface. However, Windows is not free. While still requiring some level of specialized knowledge, it does not require as much as Linux generally does. The biggest downside of Windows Server is its expense, as a license is required for each instance of Windows Server. | Mobile devices do not lend themselves to hosting web applications. They have limited hardware capabilities due to their mobile nature, generally lack hardware interfaces such as ethernet ports, and are not well-known for their uptime. While technically, Linux can be installed on mobile devices, and Android itself is a type of Linux, neither the mobile Linux versions or Android are easily configurable to host web applications. Similarly, iOS devices can technically install macOS Server, but again the hardware and software limitations rule this out entirely as an option. |
| **Client Side** | Mac OS X provides the Safari web browser by default. Safari supports a limited array of audio and video formats, and has specific requirements for optimizing media delivery. Javascript is enabled by default on Safari, but newer versions of Safari disallow the use of plug-ins for media, so care should be taken that no media requires a plug-in before being used. For mobile web on iOS as well as Mac OS X, Apple publishes guidelines for optimal delivery to its devices, and those must be kept in mind to ensure proper operation on Mac OS. | Linux distributions are fairly consistent in capability and available software whether deployed on a server or an end-user device. Due to the high variability in configuration, it is hard to gauge what special requirements will be necessary to deploy to Linux end-users. As it is much less limited than Mac OS X for media purposes, as long as the web application is compatible with standard HTML5, CSS3, and Javascript, the application should work on Linux. | Microsoft Windows default web browser, Microsoft Edge, uses the same engine as Google Chrome (Blink, V8) as of 2020. As long as the application is compatible with standard HTML5, CSS3, and Javascript, and works in Google Chrome, it should likewise be compatible with Microsoft Edge. Between Microsoft Edge and Google Chrome, this captures most of the market for Windows users. | Deploying a web application to mobile devices requires special care, as the mobile versions of web browsers have more limited capabilities than desktop versions. They also almost universally have smaller displays, and testing must be done to ensure that what is presented to the user displays as desired, be it in a mobile application installed through an app store or accessed via the web. Another idiosyncrasy of development for mobile devices is that it is not uncommon for them to have a much higher resolution display than desktop platforms, so graphics must be designed with a high resolution display in mind. For example, Apple’s Retina display has a pixel density of 326 pixels per inch, where as an average 1080p monitor has a pixel density of about 81 ppi. |
| **Development Tools** | The terms and conditions for Mac OS X forbid its installation on platforms other than Apple hardware. Testing can be achieved by either acquiring Apple hardware for testing, paying a testing service, or paying for a cloud server running Mac OS X to test. Otherwise, Safari supports HTML, CSS, and Javascript, so no specialized software other than Safari is needed to develop for the mac platform. For development of the application for Mac OS X, a Mac machine would be ideal, as this allows Xcode to be used as an IDE. Xcode is the only way to write applications using Apple’s framework, Cocoa, and Swift, Apple’s programming language for applications. SwiftUI provides the user interface. Fortunately, this development toolchain works on both iOS and Mac OS X, so writing for one platform means it will run on the other. | Due to the same variability in configuration as cited earlier for Linux, there appear to be no special tools or languages required for deployment of a web application to Linux other than HTML, CSS, and Javascript. For development, an IDE that supports HTML, CSS, and Javascript is needed. Visual Studio Code is a newer, highly customizable IDE that can be used, with a wide array of extensions and support for Windows, Mac OS X, and Linux. NetBeans is an older IDE with a built-in Javascript editor, HTML5 plugin, and native support for Angular, as well as Apache Cordova, a mobile application framework. | As stated in client side requirements for Windows, Microsoft Edge, the default browser, runs under the same engine as Google Chrome. As long as the application is compatible with these browsers, the IDES mentioned for Linux will work. | For development of mobile applications for iOS, a Mac computer running Xcode, Apple’s IDE for Mac and iOS apps, is ideal. As mentioned under the entry for Mac OS X, this would allow development of the app using Cocoa, Swift, and SwiftUI, meaning the application will run on both Mac OS X and iOS devices. For development of the web application on Android, Google publishes recommendations and guidelines specific to Chrome on Android that must be followed to ensure proper operation of the web application. As an Android version of the application already exists, the toolchain for native app development is assumed to already be in place. |

## 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**: A cloud solution to hosting is recommended through Amazon Web Services. Specifically, I recommend an EC2 server, which allows the creation of a variable number of virtual servers. This allows redundancy by cloning server instances, as well as scalability. Using a cloud service also ensures greater uptime and resilience by hosting the server on a platform designed specifically for that purpose. It also alleviates the costs and special knowledge necessary to maintain an in-house server.
2. **Operating Systems Architectures**: The recommended operating system would be Linux, due to its lack of cost, configurability, and availability of a wide array of software. The versatility of a Linux server for hosting a web application makes up for the requirement of specialized knowledge. Linux servers also have a reputation for stability and uptime, a necessity for a web application.
3. **Storage Management**: The data will physically be stored on solid state drives (SSDs), as specified by Amazon (*Amazon EBS features*). Each instance of the server will have its own storage, called Elastic Block Storage provided by the AWS platform. This storage will grow as necessary, and supports relational databases by default, allowing data to be accessed by SQL.
4. **Memory Management**: Amazon m5 instances are considered “general purpose” and have a wide array of options as far as physical memory for the servers. The actual amount needed will vary dependent on the number of users, but the scalability afforded by the platform allows quick reconfiguration if needed. For data that is repeatedly requested, caching can be configured on the server to reply to repeated identical requests in order to prevent the server itself from being overwhelmed by repeated requests.
5. **Distributed Systems and Networks**: With the recommended platform, AWS EC2, as many virtual servers as necessary can be easily created. Initially, a server specifically for the database should be used, as well as a server to act as the backend for the application. This results in a distribution of work between the SQL server, which can retrieve stored data, and the backend server, which can then process the data sent to it by the SQL server. The backend server can simultaneously be handling requests from clients. Caching servers can be set up between the clients and the backend server to allow for delivery of data that has been requested repeatedly, further reducing the load on the backend server. Multiple instances of the SQL servers, backend servers, and caching servers can be added as necessary to maintain proper operation.
6. **Security**: There are several steps at which user data can be compromised. Web traffic between the clients and the servers should be encrypted using HTTPS, and special care should be taken that the servers’ SSL certificates are always maintained. Role-based authentication should be carefully implemented by the backend and caching servers to make sure that individual users cannot access other users’ data. Firewalls should be carefully configured between the servers to ensure only data related to the application itself is permitted. Configuration of the backend and SQL servers should be hardened as much as possible, using Access Control Lists to enable only what is necessary. The principle of least privilege should be applied heavily. The root account should be disabled entirely, with the use of sudo when administrator permissions are necessary. The members of the sudoers group should be as small as possible to limit who can use sudo. Similarly, the database server’s root account name should be changed, the default ports changed, and only allow the backend servers to connect to it, as the servers are connected to virtually through AWS, which is itself an encrypted connection. To the server, it appears as if it is being connected to from the local network. Similar steps should be taken with the backend servers. Lastly, all data the servers receive should be sanitized to prevent cross-site scripting or malicious instructions sent to the MySQL database or web server.

# References

Amazon. (n.d.). *Amazon EBS features*. aws. https://aws.amazon.com/ebs/features/.

Q-Success. (n.d.). *Usage statistics of operating systems for websites*. W3Techs. https://w3techs.com/technologies/overview/operating\_system.

StatCounter. (n.d.). *Mobile Browser Market Share Worldwide*. StatCounter Global Stats. https://gs.statcounter.com/browser-market-share/mobile/worldwide.