

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)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 8/22/2020 | Candice Rogers | Final revisions, 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)

Client *The Gaming Room* seeks a cross platform, web-based application to run an existent mobile game application product titled *Draw It or Lose It* with the following requirements:

* A game session 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.

This application is currently only available on Android devices (Google Play Store). The client has requested development for a browser application version in order to expand user base and market penetration.

## [Design Constraints](#_2et92p0)

Typically, a web-based application is ported into a mobile environment; this project presents an uncommon “reverse port” from mobile (Android) to web (browser), although this description is not completely accurate. If the Android application is native, i.e., initially developed from the Android Studio and therefore native on the Android OS, then the only way to truly “port” the application to the web would be via Chrome OS. Even then, the only hardware capable of running Android Studio application on Chrome OS is the Chromebook tablet.This constraint means a new version of the application will have to be built for web using the appropriate tools, engine and language(s).

As for concerns relating to the active planning and development of the web application, one of the major design constraints for *Draw It or Lose It* will be translating the UI (user interface) from the mobile environment to the browser environment:

* The change in resolution from lower to higher may result in unsatisfactory results if existing assets are recycled (graphical UI elements, fonts used for text)
* UI layout may need to be restructured to accommodate large resolution PC/Mac screens
* Graphical elements should be in a format that allows for vector scaling (SVG) where possible; if none currently exist, recreation or conversion of assets to this format will require extra time from the design team

The next major consideration will be UX (user experience), specifically mode of interactivity. Mobile applications rely on touchscreen capability for end-user interaction and control; this will be absent on the majority of machines used to run the web-based version. This may affect placement of elements in the application, for example user text input fields or interactable buttons, since clicking with a mouse and typing with a keyboard are quite different than using touch controls.

Since end users will be playing the application over an internet connection, issues such as downtime, traffic and publication of application updates are important considerations. A cloud-based hosting service is highly recommended for this project as the most viable solution to this constraint. While the bulk of a cloud-based service is largely managed by the provider, the server architecture and services selected by The Gaming Room will be tantamount to keeping the application accessible and secure. Specifically, having multiple server instances lowers the possibility of downtime. Also, server instances run in multiple regions for location diversity will minimize local outage events that could disproportionately affect a specific geographic location. Another suggestion is a selection of services built for dynamic scaling, meaning resources can increase as user needs change - higher traffic, a larger user base, or handling application updates. The selected cloud services should adhere to the proposed budgetary constraints while still ensuring the stability of the application.

Since the proposed version of Draw It or Lose it is a web-based application, it will be accessed and run by end users locally on internet connected machines via web browser application. Cross-browser compatibility should be considered on all levels to ensure the application runs and displays correctly on different web browser applications such as Chrome, Firefox, Edge, etc. This will require potentially extensive testing and due diligence from the development team.

## [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 GameServices class contains the main functionalities for the application. It adds, searches for and checks duplicates on team and player data using an array list.

* The GameService getInstance is the singleton that will control memory instances for games to only once per session
* addGame adds a new game with an ID name
* getGame will retrieve an existing game via name or ID
* There are also functionalities to control the turn-based system regarding which player on which team can go next during gameplay
* getGameCount retrieves the current number of game instances

This class is associated with Game, Team and Player, which contain and handle the names and IDs of these attributes. Note the multiplicity of zero to many, where one game instance may have none to many teams and players. In turn, the three main attributes are identified through inheritance with the Entity class. Finally, the main method is contained in the ProgramDriver class initializes the game data, with a directed association to the SingletonTester class, which simply tests the singleton function within GameServices.

The functionality for adding game sessions to the array and assigning each instance an ID number are held as private variables, as are those for assigning the Player ID and Team ID. This data should not be accessible or editable by outside parties.

****

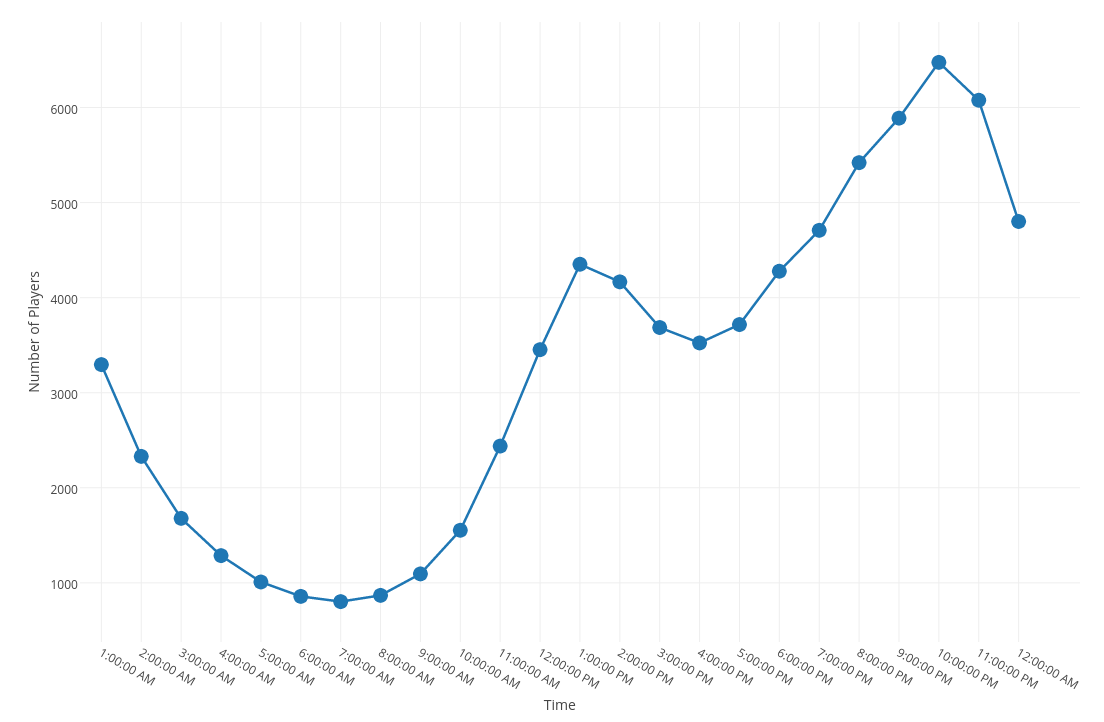
## [Evaluation](#_2o15spng8stw)

Since this version of *Draw It or Lose It* is a web-based application, greater flexibility in development is available as opposed to a game that is run locally in a desktop OS or mobile OS environment.

One of the great advantages in utilizing Java for cross-platform applications is the JVM. Java is cross platform in the sense that a compiled Java program runs on all platforms for which there exists a JVM. This holds for all major operating systems, including Windows, Mac OS and Linux.

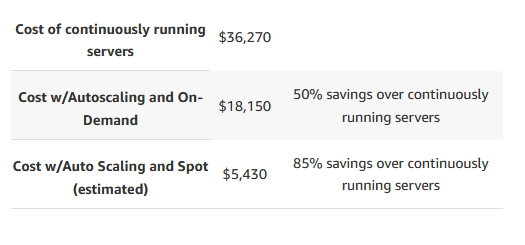
Regardless of what platform the cloud service provider utilizes to run and maintain their servers, a Java-based application should be compatible. Many of these providers offer specific SDKs for Java web applications as well. For example Amazon AWS provides the AWS Toolkit for Eclipse and a versatile AWS SDK for Java — a set of Java interfaces that simplifies the task of tying an application with the extensive list of popular AWS services such as Amazon EC2 (IaaS for utilizing multiple server instances), Amazon S3 cloud storage, DynamoDB NoSQL database and many others.

Considering the proposed budget for this project, the client should be informed that cloud hosting is an ongoing charge, usually in the form of a scheduled yearly payment. Cost can vary widely depending on how simple or complex the hosting services are, as they can be completely customized to the needs of the client and can quickly be changed as the application service grows and develops. Another consideration would be a service like Amazon GameLift. Instead of using the traditional approach of buying a fixed amount of servers that run continuously all month, with Amazon GameLift the client would only pay for the capacity used. GameLift charges for instances by duration of use and bandwidth by quantity of data transferred.



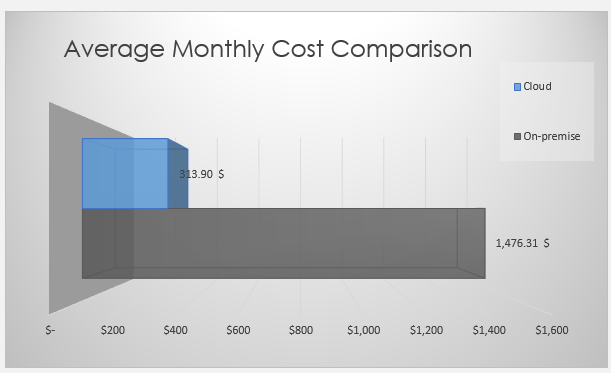
*Amazon - https://aws.amazon.com/gamelift/pricing/*

Above is an example of tracking end-user traffic for use in auto-scaling. This would be used to determine the amount paid for hosting. This can actually result in significantly saving on cloud server costs.



*Amazon https://aws.amazon.com/gamelift/pricing/*

As for local server hosting solutions, the client would be required to calculate costs of physical server equipment/machines, staffing of those knowledgeable in server-side maintenance both physical and virtual, as well as planning for appropriate space within an appropriate secure location. Linux Ubuntu is a popular choice for both smaller scale and enterprise level locally maintained server-side operating platforms, as it is flexible, open source and stable. Ubuntu servers are also renowned for their robust integrated security features and controls, something that would be essential to maintaining a database that contains personal information of the user pool. The next best option, or perhaps an option of equal value, would be a Windows Server 2019 based solution. This is a very widely used platform across the globe, able to be tailored to many different business needs. Microsoft has recently made all local server-side solutions compatible with Azure cloud-based services by default, allowing hybrid capabilities and multi-layer security features. The buzzword for this innovation is “hyperconverged infrastructure”, which aims to seamlessly converge local and offsite hardware, modernization of traditional applications through utilizing containers and microservices, virtually extend datacenters and elevate security from the ground up - starting with the operating system.



*https://www.sherweb.com/blog/cloud-server/total-cost-of-ownership-of-servers-iaas-vs-on-premise/*

Unfortunately, a macOS Server platform could be considered inconvenient and impractical for launching this particular application. It is a highly specialized and proprietary platform that has since been depreciated progressively since 2018 (<https://www.zdnet.com/article/the-death-of-macos-server/>) as Apple continues to build up and focus on direct to consumer mobile devices such as iPhone and iPad. Indeed, macOS Server no longer supports such basic services as email handling. Considering web-based applications, both DHCP and DNS handling and resolution services have also been depreciated, essentially gutting its ability to run a network. Even Apple-made server *hardware*, the Xserve, was discontinued in 2010, and outdated, used racks still sell for well over $5,000 per unit today, making them fairly cost prohibitive.

Regardless of hardware choice, the client should consider cost of upgrading infrastructure on a regular basis as well. The average lifespan of physical server hardware and equipment is currently 3-5 years. Comparatively, pricing may change on a cloud-based plan yearly, though increases are usually small.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS Server or Mac-based hardware/machines will likely not be utilized or relevant to this project, since it has been depreciated over recent years. It is also highly proprietary and cost-prohibitive, making it an impractical option. | Linux-based server-side solutions could be a good fit for this project, offering relatively inexpensive hardware options, robust security, open source flexibility and widespread utilization with Ubuntu. | Windows Server 2019 offers a hybrid cloud-to-local solution, as well as a fully cloud-based solution through Microsoft Azure. It is highly flexible and customizable, cost effective and robust in security features and scalability. | Mobile devices will not be utilized to host this web-based application, as they do not have the capacity to do so. However, while the extant mobile application version of Draw It or Lose It is currently assumed to be successful with whatever hosting solution it is deployed on, refinements or cost savings could be proposed for future updates. |
| **Client Side** | JavaScript web-based applications run on Safari browsers typically without issue as long as end-user has JS enabled in their local settings. However, considering the highly proprietary nature of Apple products and applications, special care may be needed during development to run the app on iOS and macOS hardware/devices via Safari or alternately, with the iOS versions of other browsers such as Chrome. | JavaScript web-based applications typically run on Ubuntu open-source compatible browsers without issue as long as end-user has JS enabled in their local settings. Practically any web browser can be run on a Linux machine, from Chrome to Firefox to Opera. Attention to browser specific optimization for the most popular ones will still be important. | JavaScript web-based applications typically run on Microsoft Edge, Chrome, Firefox, etc. on Windows PC systems without issue as long as end-user has JS enabled in their local settings. As with a Linux system, attention to browser specific optimization for the most popular ones will still be important. | JavaScript web-based applications run on Android and iOS devices within browsers with some rendering and usability issues as long as end-user has JS enabled in their local settings. Application will not be optimal for mobile devices; recommendation to alert end-user to download mobile app for best experience. |
| **Development Tools** | Using specialized Mac-based developer tools would be a difficult task for this project. MacOS and iOS application development is highly proprietary, requiring the use of physical Apple hardware or the use of a VM – which is not recommended due to potential security and stability compromises. The only advantage to using Mac developer tools would be to guarantee compatibility on Mac systems/OS. Therefore a Mac-based deployment plan alone is impractical for development of this web-based application. | Since the application will be based in Java and JavaScript, which can be written using the developer’s choice in IDE or native OS on their machine of choice, a Linux platform would be more than capable of handling development, testing and deployment of the application. Care should still be taken to ensure cross-platform compatibility for the end user, though the flexibility of Ubuntu’s open source should give developers an advantage in this. | Since the application will be based in Java and JavaScript, which can be written using the developer’s choice in IDE or native OS on their machine of choice, a Windows platform would be more than capable of handling development, testing and deployment of the application. Care should still be taken to ensure cross-platform compatibility for the end user. | Since a successful mobile version of the application already exists, current development practices could be improved upon for Android and iOS. For running the web-based version of the application, a physical mobile device should be used for testing to check if it runs correctly on the mobile version web browsers, both in mobile and desktop display modes. |

## 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**:

In this particular case, the Web itself is, essentially, the end-user operating platform. The game application would be hosted on a website and run on a web browser, which in turn is run on any Internet capable device, such as PCs, Macs or tablets. This will make the application largely “cross-platform” largely by default, in the sense that it will be compatible with a range of machines running different OS or browsers. However, care must be taken to ensure the application displays and functions correctly on various web browsers.

As for server-side operating platforms, the main three recommendations offer solutions for fully on-site, hybrid, and fully cloud-based systems, respectively:

* Linux Ubuntu server(s) maintained onsite by dedicated staff. Physical machines to be installed in appropriate secure and climate controlled area. Allows for complete control and maintenance of all databases, memory, etc.
* Windows Server 2019 “hyperconverged infrastructure” hybrid of limited server machines onsite, blended with Azure cloud services.
* Amazon AWS full cloud-based services maintained completely offsite, with all database and backend server-side tasks handled by Amazon.

If overhead is the client’s main concern, then a fully cloud-based system may be the best choice out of the three options.

1. **Operating Systems Architectures**:

An Ubuntu system will likely require the consultation of a professional server architect to recommend how much and what kind of equipment would be needed. For back end or server-side deployment and maintenance that is less involved and requires fewer machines, Windows Server 2019 offers a very robust hybrid cloud to local solution if the client desires to maintain some storage and hosting control on-premises, combining server hardware infrastructure with Azure cloud services. If a fully cloud-based system is selected, Amazon AWS would be the most flexible and cost-effective solution, as well as most time efficient, since Amazon sets up everything for the client through a series of consultations.

1. **Storage Management**:

With excellent advancements in security, architecture and affordability, a cloud-based storage management service is highly recommended to the client. Some reliable services include Amazon AWS and Microsoft Azure. These providers offer managed database systems and strong data security – far more than can be created and maintained by the client.

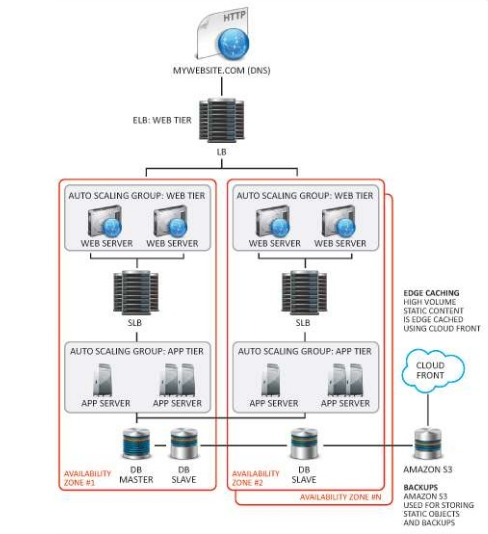
1. **Memory Management**:

JavaScript automatically allocates memory when objects are created and frees it when they are not used anymore. This is called GC or garbage collection. While this does alleviate manual memory management through code once the application is deployed and run by the end-user, there are some issues to keep in mind. One concern is the possibility of memory leak in the Microsoft Edge browser, especially in large-scale applications. Ensuring performance will require testing on a variety of browser environments and constant upkeep of best-practices by developers.

Maintenance of physical memory capacity on the server side will depend on what kind of service is utilized. With a cloud-based system, physical memory upgrades and allocation is dictated by the service provider – in this case either Microsoft or Amazon. This is included in the monthly or yearly cost of the service. With a localized system, memory upgrades would have to be paid for by the client, for both hardware and labor of installation.

1. **Distributed Systems and Networks**:

In regards to web applications, a distributed system typically comprises of any machines, hardware, containers, servers or nodes working together as to appear as a single source computer to the end-user. The machines will have a shared state but can fail independently without affecting the entire system. With storage management hosted by a cloud service, servers are maintained by that provider. The standard for Internet based gaming has been a centralized model for many years, where a typical “client-server” arrangement is utilized. While this used to mean an application developer that hosted the game had to invest in a server directly, effectively creating a three-tier system, that is no longer the case.



*Courtesy css-cloud.com*

1. **Security**:

With the utilization of a cloud-based data management and storage service, a large part of the security requirements for the application is already taken care of. However, attacks against web apps range from targeted database manipulation to large-scale network disruption. For end-user visible elements such as the application website itself, involving user login and registration, authentication, group interactivity and communication, a WAF (web-application firewall) is recommended to protect against malicious HTTP traffic, along with HTTPS certification. For a specific security service, Cloudflare is recommended, as it is easily integrable with other services mentioned above.