

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 4**](#_Toc115077323)

[**Domain Model 5**](#_Toc115077324)

[**Evaluation 5**](#_Toc115077325)

[**Recommendations 9**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/22/2024 | Casey Doyle | Initial build and commit |
| 1.1 | 2/7/2024 | Casey Doyle | M5 Project 2 Environment updates |
| 1.2 | 2/24/2024 | Casey Doyle | M7 Project 3 Platform Recommendation updates (^stable release$) |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game that serves multiple platforms based on their existing game, Draw It or Lose It, only available as an Android app. Creative Technology Solutions (CTS), an accomplished Technology Consultant offering software solutions to businesses across the globe, has been tasked in coordinating with The Gaming Room to achieve their goals. CTS will dedicate a team of developers to produce the viable product, while also recommending that The Gaming Room have their internal team coordinate for the requirements to establish an environment for the Webpage version of ‘Draw It or Lose It’ to be homed.

## Requirements

*~~<~~* ~~Please note: While this section is not being assessed, it will support your outline of the design constraints below.~~ *~~In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>~~*

## [Design Constraints](#_2et92p0)

The Gaming Room wants to develop a web-based version of their game Draw It or Lose It to serve users of the internet. This objective is solvable, as the flexibility and accessibility provided by web development tools are a proven technology, where the chosen language Java has a strong lineage in the industry, providing almost universal compatibility for all sides: Server, Mobile, Client, and Developer. CTS recommends utilizing a private cloud provider rather than establishing their own datacenter presence, in efforts to meet the budget constraints over the intended lifespan, as greater geographic availability is desired. Some of the requirements and constraints, as highlighted below, are to be expanded upon in the recommendations section of this design plan. Constraints to consider include:

* Virtual or physical Server and networking hardware infrastructure required to house the project.
* Platform (Game) Testing for each individual use case (users of Mac, Windows, Linux can play?).
* Cloud Services utilized to provide redundancy, failover, and general high availability to users globally.
* Storage, Memory, Bandwidth, Backup, and Security plans for the hardware and data.
* Type of software to be used, with any licensing the business must abide.

## [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 UML class diagram provided below shows the relationship between the different components implemented for the Draw It or Lose It System and how they work together to present the overall player experience. Each of the different components represent a different class, where different attributes and functions are provided to accomplish necessary interactions, and required associations, for and between themselves. Though the same type of data container is being programmed here, the Iterator pattern is being used, allowing simplified access to the data elements of a sibling class, encapsulating the data while providing the necessary information when needed. We also see presented in the UML the Singleton pattern, where in development the singleton pattern is tested to be functional. The data is designed through singleton to be accessible to other classes, while also ensuring that only one instance of the class be established, validating that each unique object is indeed instantiated only once, and used only where necessary. Polymorphism through the Inheritance principal is exemplified in that we see the Entity class is the base superclass housing the common attributes and behaviors, where directly Game, Team, and Player classes all inherit and utilize those attributes and methods. The Game Service has a 0-to-many relationship with Game objects, Game has a 0-to-many relationship with the Team objects, Team has a 0-to-many relationship with Player objects, allowing many unique instances of each to be represented within. Public attributes and methods are represented with a + sign before, while – attributes and methods are objects private to that class, where the Types are shows after semicolon.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | OSX is a Darwin based operating system, a variant of Unix, is perfectly suitable in serving web-based (MAMP stack) applications to users of the internet, where the software packages required in doing so are generally open-source and freely available, in addition to Apple’s branded software packages geared toward hosting environments. However, the cost associated with Apple hardware, as well as the associated administration labor costs, are typically higher than market offerings. | Linux, is the best option in serving web-based (LAMP stack) applications to users of the internet, where in addition to the Linux OS distribution itself, the software packages required are generally open-source and freely available. Hardware costs are comparable to Windows, while requiring slightly less performance, equating to further cost savings (in addition to the licensing, a further factored savings in comparison to others). The associated administration costs are roughly in the range of competitors, though slightly less regarding the realm of service web-based developments demand. | <Evaluate Windows for its characteristics, advantages, and weaknesses for hosting a web-based software application.>  Windows has many of the same available open-source offerings required for hosting, though they also have proprietary Microsoft based internet services. Licensing costs are associated in running the OS, available in many different forms. Many vendors provide purchase options the different versions of OS licensing at time of the hardware purchase. Administration costs are also near the market rate. | ***FROM***: N/A – Demand cannot sanely or desirably be associated with porting commercial grade webhosting software solutions to mobile devices at any commercial capacity. This reality may be capitalized in the future in observation a relatively trending increase of attention on arm since Apple’s M series emerged.  ***TO***: However, when considering the server-side requirements for serving TO mobile clients, REST API calls will be made from the mobile app side, therefor utilizing the existing Server-side stack implementation. |
| **Client Side** | Development efforts will include testing the web-based version of the game across the mainstream internet browsers for MacOS. A functionality test and verify plan will be defined and attested to upon each rolling public major/minor release of the game as they become available to the end-users. As of today, Safari, Firefox, and Chrome are the main browsers worth testing across (others don’t have a user-base to warrant development effort). | Development efforts will include testing the web-based version of the game across the mainstream internet browsers for Linux. A functionality test and verify plan will be defined and attested to upon each rolling public major/minor major release of the game as they become available to the end-users. As of today, Firefox and Chrome are the two main browsers worth testing across (others don’t have a user-base to warrant development effort). | Development efforts will include testing the web-based version of the game across the mainstream internet browsers for Windows. A functionality test and verify plan will be defined and attested to upon each rolling public major/minor major release of the game as they become available to the end-users. As of today, Firefox and Chrome are the two main browsers worth testing across (others don’t have a user-base to warrant development effort). | Because the server-side follows the REST API standard, Android and IOS will be designed as feature complete replicas of each other. The IOS app requires a dedicated development team to coordinating system design and verify functionality is consistent with the Android version, where Apple AppStore licensing costs are to be included in their maintenance contract for the life cycle of the app. Mobile Browsers: since the ‘Web Based’ client development team will verify cross platform browser compatibility across the board, mobile optimization for the mainstream mobile browsers of Android and IOS will also be considered in the design, layout, and functionality testing efforts. |
| **Development Tools** | Java has been designated as the language of choice in delivering the game application to the end-user. IDEs for Java have been ported and available, where necessary tools to implement the code remotely to infrastructure hardware, while also having access to mainstream revision management solutions are available for Developers using Mac. | Java has been designated as the language of choice in delivering the game application to the end-user. IDEs for Java have been ported and available, where necessary tools to implement the code remotely to infrastructure hardware, while also having access to mainstream revision management solutions are available for Developers using Linux. | Java has been designated as the language of choice in delivering the game application to the end-user. IDEs for Java have been ported and available, where necessary tools to implement the code remotely to infrastructure hardware, while also having access to mainstream revision management solutions are available for Developing from Windows. | ***Develop FROM mobile***: N/A - Development environments for mobile devices have not matured enough to warrant recommendation in any professional capacity.  ***Develop TO mobile***:  Android and IOS each have their own environments, with access to developer documentation, resources, and support community. The IOS app store has maintenance fees associated with making the app publicly available through their market. High OS versioning and testing requirements due to frequent releases. |

## Recommendations

1. **Operating Platform**: The Amazon Cloud offering is a popular and stable platform we recommend to the benefit of its global reach, high availability, scale of resources (physical, financial, and corporate/entrepreneurial), and demand-based cost approach. Each of the main players, Microsoft, Apple, IBM, Amazon, and Google, amongst the variety of smaller webhosting companies and private datacenter and LEC operators, showcase a perceived benefit by utilizing their own product. Access to the full catalogue of different architectures is shared amongst vendor offerings. Regarding vendor lock-in, choosing any one platform over the other, employing a popular automation framework like Ansible or Puppet will minimize any associated impact to expand and/or migrate components of Draw it or lose it to other computing environments.
2. **Operating Systems Architectures**: Amazon cloud provides all possible variety of the mainstream Operating System Architectures; Microsoft ‘Windows Server’, Apple ‘macOS’, and GNU ‘Linux’ ‘distributions’ (just as important not to forget Unix, BSD, and other edge-case variants like RtOS), where the system’s platform components are satisfied through Linux to run Draw It or Lose It workloads. In comparison to Windows and Apple, Linux is removed mostly of expenditure burdens, planned and unforeseen, associated with hardware, software, and respective licensing purchase, and upkeep. Linux provides all the necessary programming languages, utilities, and packages, while each unique variant of Linux complies with RFC’s, frameworks, and standards.
3. **Storage Management**: Linux readily supports LUN technologies, a networked storage solution providing high-performance and capacity, where servers of the platform associated with hosting the web-based application will be configured to make regular backups of their local critical operating and access data. Each server plan considers redundancy, capacity, and throughput, regarding each as an individual responsibility, as well as coherently, a role shared across the infrastructure; Many of the file system descriptors, mediums, caches, and databases are sprawled across components, internal to each ‘server’, and external across the collection. Backup servers have long-term storage maximized, with a medium-high requirement for throughput capacity, while internal and edge servers require less storage capacity, with, ideally, the lowest response time and highest throughput. The application servers need a general middle-ground level of capacity but require a high throughput. Storage Redundancy is maximized initially local to each server of the fleet, with long-term storage for backup and databases, concerning all application, operations, and administration, to be housed on-site amongst different member servers of a unique regional group of datacenters and failover groups, where off-site backup-and-redundancy requirements are satisfied further.
4. **Memory Management**: The built-in memory management within Java handles most of the memory work necessary at the programming level. Garbage Collection along with other enhancements of the JVM allocates memory to threads and classes upon instantiation of the necessary code. Proper patterns are coded at the application level to satisfy design constraints, while also efficiently deduplicating object data in RAM, ensuring unique instances of objects exist while sharing reference to the collection of images. Data is assigned to the heap, where we can see a relationship to active user-base (gamers) and memory usage load of the system. The Linux servers have a very low footprint for operation, leaving a predictable capacity of memory available to the application, database, hosting, and network stacks. Memory management is placed on high priority in considering the design requirements and configurations of a region’s application-server group. Anticipating load increase, there will be a utilization threshold for when it makes more sense not to transact, if and when possible, the image data across typically-slower local and remote, network and storage, media, using DMA, interrupts, and registers, to schedule, access, and transmit data through different queues, tables, chains, buffers, and logging systems, across local-BUS and network-linked pathways amongst remote and peer controllers and servers. The main takeaway here is to have that sweet spot of ~1.6GB of memory available at the application edge, (in addition to operationally required allotment of memory, considered at a scale proportionate to the load) to alleviate congestion at hot spots in each server, across the platform, and its various layers.
5. **Distributed Systems and Networks**: Given that Amazon scatters datacenters to cover the globe, implementing Geo-Redundancy through the Amazon cloud satisfies the application’s global availability requirement. The client and server model expands beyond that of Draw it or Lose it Players and The Gaming Room, where components of the different regional platform datacenters and servers are all linked. GSLB provide datacenters the ability to share common domain spaces from different geolocations, to make balance, availability, and failover decisions for services of different regional platforms. BGP provides availability and condition-based internet routing between different peering ASN’s across the networks. Firewalls divvy up incoming requests from the clients’ requesting services from the outside world, where connections are tracked, and traffic is forwarded off to the haproxy servers. From there, load is determined based on the existing connections being tracked, packets are mangled, and sent on down the APP-vLan where a member server to the quorum will volunteer to service a request, the traffic traveling of some balanced fashion, or other round-robin-type logic over channel bonded fiber or ethernet connections redundantly communicating over all interfaces while simultaneously maintaining VRRP heartbeats at interval across an aggregate lane. The application server who volunteered to handle that client’s layer 4 REST request will now divvy up what is left of the packet and hand that layer 7 off to the JAVA code running on metal(^virtual). From there, a traditionally unloaded and relatively quiet server that just finished rebooting after a maintenance window early that morning will need to request the images from a peering database server to freshen it’s memory, which it does so from across the DB-vLan to that same haproxy server who handled the clients request, which passes the packet along to the Memcached server group (doesn’t have to mangle again, as a localized private origin communication), which just so happens to have that image loaded up in memory and ready to go, no lookup steps necessary, so it ships the reference to the first image group back in a hurry. App server handles a small penalty in breaking up that image group to store in an instance of the Game() object’s memory register for this next four rounds of game-play… and the first image is PUT right back outbound to the client, via port 443!
6. **Security**: The platform itself contains security controls, for user management, authentication, authorization, and accounting, logging, change and update management, in addition to the myriad of ever-growing security requirements of today. Logging is the heart of an information security and management posture, to which all components of the overall system rely upon; networks, databases, application and web servers, the outlying platform management, everything. Data encryption is to be deployed for any data at rest within any contained system, segment, or environment, where encryption is applied by layer for anything in transit. VPN with IKE across maybe a GRE tunnel could provide that transit security. GSLB firewalls will communicate with each other over unique multiple public internet connections, with redundant channels of redundantly balanced ethernet-bonds securely and simultaneously communicating across unique TLS-v2-WKc(Wrapped client key) encrypted links. There are many combined modalities employed to achieve uptime in respecting SLA. Encryption, AAA, Patterns, Lease-privilege are merely a portion of capability.