

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 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1/22/2024 | Casey Doyle | Initial build and commit |

## [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. 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. 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, Game has a 0 to many relationship with the Team, Team has a 0 to many relationship with Player, 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 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, a variant of Unix, is the best option in serving web-based 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 windows licenses, a further factored savings in comparison). The associated administration costs are roughly in the range of competitors, though slightly less regarding the realm of service web-based environments. | <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. | N/A - as there is no generally sane or desirable demand associated with porting commercial grade webhosting software solutions to mobile devices in any commercial capacity. |
| **Client Side** | Development efforts will include testing the web-based version of the game across the mainstream internet browsers through which the software is to be accessed. A functionality test and verify plan will be defined and attested to upon each rolling major 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 warrant a user-base worth of the development effort). | Development efforts will include testing the web-based version of the game across the mainstream internet browsers through which the software is to be accessed. A functionality test and verify plan will be defined and attested to upon each rolling 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 warrant a user-base worth of the development effort). | Development efforts will include testing the web-based version of the game across the mainstream internet browsers through which the software is to be accessed. A functionality test and verify plan will be defined and attested to upon each rolling 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 warrant a user-base worth of the development effort). | The app has been designed for Android, so for mobile web access from a mobile device, the developers should take in consideration mobile optimization for the design layout and interfaces through which the game is to be accessed. |
| **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 on 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 through 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 on Windows. | N/A - Development environments for mobile devices have not matured enough to warrant recommendation in any professional capacity. |

## Recommendations

1. **Operating Platform**: The Amazon Cloud offering is a popular and stable platform we recommend for its global reach, high availability, scale of resources, and demand-based cost approach. Expanding infrastructure resources is generally performed remotely and through manageable scheduled maintenance windows, and in many scenarios can be scaled on-demand. Linux is supported, costs are respectable, and multiple technical and business constraints are met in this environment.

1. **Operating Systems Architectures**: Linux x64 architecture has been recommended to host the Draw It or Lose It environment due to its availability, proven history in webhosting, access to support, optimization of costs, and low hardware overhead. All the necessary language requirements are met, and each server supports the necessary software solutions and protocols to communicate with each necessary sibling server and edge network device. There are generally no further software costs involved, only development, deployment, and maintenance, while virtual hardware appliances and resources are allocated and quantified at the responsibility of the Amazon Cloud.
2. **Storage Management**: Linux highly supports LUN technologies, a networked storage solutions 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 individual role across the platform. Backup servers have long-term storage maximized, with a medium-range requirement for throughput, while networking and edge servers require less capacity, and a respectable throughput. Application servers need a general middle-ground level of capacity but require a high degree of throughput. Redundancy is maximized at each server in the fleet.
3. **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. 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 amount of memory available to the application and hosting stack. Scaling of these hardware resources is to be considered when resource utilizations start to approach designated thresholds, typically observed, monitored, and alerted through platform metrics and monitoring solutions, where further decision can be made by either automated or manual administrative effort.
4. **Distributed Systems and Networks**: The Amazon Cloud has software defined networking solutions available, lowering the cost of hardware networking appliances, where many of the global availability, load balancing, and failover standards are supported. Many of these technologies are freely available options which are proven by their history and use. There is the separation of effort for members of the platform, where servers are configured to maximize their performance and utility based on varying factors. Servers in a hosting environment generally include Database, Application, Web Proxy, Logging, Access, Backup, and many other roles on the network, some servers hosting multiple roles, while many roles have their own dedicated resources (virtual, or other). None-the-less, each of these different components need to speak with each other, spanning across networks, with the complexity of these multiple often isolated, external, internal, and layered segments, typically managed with dedicated hardware devices, now mostly are implemented by Software Defined Networking (available with Amazon Cloud). Servers will be paired together to provide redundancy, availability, and performance; servers will be geographically spread out across data centers, configured to work together, in serving customers who are closest, while providing services to other geo markets where maintenance or other outages might impact operation, further ensuring an always available service across the globe (all factored into a scaled pricing model, forgoing the costs associated directly with datacenter ownership).
5. **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. Platforms are generally connected via private networking, virtual, abstracted, or other transit mediums.