

Draw It or Lose It

# **Project Software Design**

Dallas Weber

Version 2.0 (10/03/2021)

## 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 | 09/15/21 | Dallas Weber | Initial Project Document Submission |
| 2.0 | 10/3/2021 | Dallas Weber | Evaluation |
| 3.0 | 10/16/2021 | Dallas Weber | Recommendation |

[PROJECT 1: 3-1]  
[**Executive Summary**](#_sbfa50wo7nsh)

Creative Technology Solutions has partnered with a new client, The Gaming Room. The Gaming Room has challenges with their popular Android app, Draw It or Lose It, in developing it as a web application. This document provides the business and operating constraints. The constraints are addressed to ensure product delivery to the maximum iamount of users.

## [Design Constraints](#_2et92p0)

**Operating/Business Constraints**   
  
*Project Schedule, Budget & Milestones*

All phases of the app development will follow the project schedule. Development and the delivery of the application will be within the budget expectations. Change requests, when submitted, will be compared to the initilal scope to understand any impact to the timeline, effort and budget.

*Team Knowledge of Web-Based App Development and Client Domain*

The project team should be skilled web developers with knowledge of main web-based application development principles. Those principles are a focus on the user, quality, scalibility, and maintainability.

Additionally, strong engineers and those with subject matter expertise will be utilized.

*Software Licensing*

There are no known licensing requirements or restrictions.

*Legal, Regulatory and Compliance*

There are no known legal, regulatory, or compliance requirements or restrictions.

*Vendor*

Vendor products will be purchased ad-hoc and be off-the-shelf solutions. App hosting will be remotely hosted by a cloud service provider.

**Technical Constraints**

*Programming Language – Java*

The app will be built using the Java programming language. The existing mobile app runs on the Android platform. Java is the official language for Android app development (hakiran78, n.d.). The existing code will be leveraged for the web-based solution.

*Operating System(s) and Web Browser Support*

Operating systems (OS(s)) execute apps through individual jobs broken down into instructions known as

“threads”. OSs execute threads in varying fashion, which can impact performance (Silberschatz, 2009).

A web-based solution involves multiple parties. The end user playing the game is referred to as the

client. The computer running the app is the server.

The client has no OS constraints. Its key constraint is the web browser employed. The app should be

compatible with leading web browsers. Google Chrome carries roughly 60% of world market share,

while Microsoft’s Internet Explorer and Edge, Apple Safari, Mozilla Firefox, and Opera combine for

roughly 20% (Web Browser…, 2020). Browsers are similar in functionality and support, but carry enough

varying nuances that consideration for each is warranted. If design efforts for all major browsers cannot

be employed, Chrome is the priority browser to support.

Web apps are built around hypertext transfer protocol (HTTP), a set of rules that dictate communication between machines (Parikh, 2020). Web apps require multiple communication requests. The app must be designed to work with a server OS capable of handling a large volume of communications through efficient thread execution. The app must be designed to optimize multi-threaded architecture. Single instruction execution (i.e. single-threaded) will be insufficient.

Request volume can be managed by a thread pool that limits the number of simultaneous concurrent relays on the server. The Java programming language carries a built-in framework design to manage and optimize thread pool architecture (Abhishek, n.d.). Thread design is not OS independent, as each OS employs a varied implementation (Oaks, 2004). Microsoft Window’s Win32 application programming interface (API) provides intuitive support for thread integration with Java (Silberschatz, 2009).

*Multi-Platform Design and Distribution*

Multi-platform is this context represents running the app across multiple systems. To leverage distribution through a web-based solution, the app should expose a lightweight API callable through standard network protocols (e.g. HTTPS) backed by functionality that executes quickly. This is best served by modular app development using object-oriented (OOP) design principles. RESTful API design supports lightweight functionality with faster processing. It forces the client to provide the necessary information needed for the server to fulfill the request and is callable through HTTP (Fielding, 2000).

*Cloud/Remote Hosting*

Web-based apps run on a remote server called the “cloud”. Four key topics to consider when hosting a cloud solution include scalability, availability, manageability, and feasibility (Lydford, 2015). Meeting these requirements internally can be costly and inefficient- particularly scalability and availability. It is more cost effective and manageable to outsource hosting to a third party that leverages server scale (Pros and Cons…, 2017). Scalability is supported through the ease at which server providers can add additional resources (Pros and Cons…, 2017). Availability and manageability are supported through heightened security efforts, more sophisticated disaster recovery, and hardware economies of scale (Pros and Cons…, 2017).

## 

## [Domain Model](#_8h2ehzxfam4o)

## The Gaming Room UML diagram below utilizes seven classes, ProgramDriver, SingletonTester, Entity, GameService, Game, Team, and Player. The Entity class has a parent relationship to the four child classes. Game, Team, Player and GameService classes are associated where each entity relies on the other. The SingletonTester class inherits from the ProgramDriver. The Program Driver class includes the main method to drive the application. The Object-oriented principle utilized in this diagram is inheritance. This has enable the SingletonTester class to perform functions and responsibility from the ProgramDriver.

## "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.

## [PROJECT 2: 5-2]

## [Evaluation](#_2o15spng8stw)

Integrating the game with other platforms besides Android, requires an evaluation of various platforms' characteristics, advantages, and weaknesses. This is done on the server-side, client-side, and the tools used for the development are identified.

**Server Side**

This part of the evaluation looks at the characteristics, advantages, and weaknesses for hosing the game on Windows, Linux, or MacOS. Up to thousands of players will be involved in the app. The ability to host the software is then based on the following evaluation.

## 

|  |  |  |  |
| --- | --- | --- | --- |
| **Windows** | **MacOS** | **Linux** | **Mobile Devices** |
| Expensive platform with both OS and product licensing costs (Horne, 2019). Targets Microsoft web framework technologies, imposing technical constraints (Heng, 2020). Intuitive GUI for administrative functions (Battle of Web…, 2013). Most restrictive user access controls and lock screen sequences that restrict malicious scope. Least privilege access tends to require additional configuration vs. Linux (Silberschatz, 2009). Process execution is more granular vs. other OSs and tends to get blocked less. Supports a robust service API that integrates easily with Java (Silberschatz, 2009). Layered architecture that is heavyweight and subject to performance and availability issues. Costly scalability. | An expensive platform due to reduced demand and licensing costs. Software packages have comparable open source Linux counterparts (Cudmore, 2020). Intuitive graphical user interface (GUI) like Windows. Architecture is a hybrid of layered and modular, enabling typical functionality to be scaled with less performance impact than fully layered architecture (Silberschatz, 2009). Implementation technologies are more diverse, imposing fewer technical constraints than Windows. Expensive scalability (as with Windows), as the OS carries more features than needed to host the application and licensing costs increase with additional servers. | An expensive platform due to reduced demand and licensing costs. Software packages have comparable open source Linux counterparts (Cudmore, 2020). Intuitive graphical user interface (GUI) like Windows. Architecture is a hybrid of layered and modular, enabling typical functionality to be scaled with less performance impact than fully layered architecture (Silberschatz, 2009). Implementation technologies are more diverse, imposing fewer technical constraints than Windows. Expensive scalability (as with Windows), as the OS carries more features than needed to host the application and licensing costs increase with additional servers. | Frameworks, such as I-Jetty promote scalability and performance for web hosting (Mobile Web Server, 2020). Requires connectivity to an internet-enabled computer to support DNS conversion, adding extra relay steps (Mobile Web Server, 2020). Popular solutions no longer actively maintained (Mobile Web Server, 2020). Many deployment tools (i.e. Jenkins, Maven, etc.) support all major OSs. Deployment solutions can be leveraged that are independent of the OS (Top Software…, 2017). |

## From this evaluation, every operating platform offers a deployment method for the servers from where the website will be hosted. The licensing cost is also different depending on the platform.

**Client Side**

The client-side evaluation focuses on the software development considerations that can support multiple customers. That is expertise, time, and the cost of running the gaming application beyond the current android-only platform.

|  |  |  |  |
| --- | --- | --- | --- |
| **Windows** | **MacOS** | **Linux** | **Mobile Devices** |
| Windows requires minimum time and expertise to have the game running. This is because it is a common OS that most people have interacted with, and hence the client’s employees will find it easy to operate.  It is cost-effective for the client, Gaming Room.  The requirement that Gaming Room needs to consider is that developing the application should be compatible with mobile devices and web browsers. | The client will require moderate expertise due to the uncommon nature of the platform.  It is also a cost-effective platform. Similar to windows, the client should ensure that the developed app is compatible with mobile devices and different web browsers. | This platform needs maximum expertise and adequate time to develop the gaming app. Similar to other platforms, Linux also requires that the app developed is compatible with various web browsers and mobile devices. | The client already uses it. The platform offers high flexibility to clients and allows the developers to see updates at different locations. Thus, it requires less time and cost for development. It is slightly more challenging to implement compared to other devices. |

## The Relevant Tools and Languages used in Different Platforms

This section of the evaluation seeks to inform the client of the relevant programing tools and languages in the different platforms. This is to inform the client of the technical requirements' impact on the program development team.

|  |  |  |  |
| --- | --- | --- | --- |
| **Windows** | **MacOS** | **Linux** | **Mobile Devices** |
| This is easier to use compared to Linux but can run the application the same way as Linux.  The tool and programming language needed include eclipse and visual studio.  Notepad ++ can be a used tool.  Languages consist of Javascript, CSS, and HTML. Specifically, the tools can be Java, Ruby, PHP, and Python. Licensing costs may be required.  Multiple development teams may also be considered for smooth operations. | Mac can run all languages. In relation to the Gaming Room requirement, languages consist of HTML/CSS/JavaScript. It also supports libraries for general-purpose and fronted languages.  Specifically, the tools can be Java, Ruby, PHP, and Python. Licensing costs may be required.  Multiple development teams may also be considered for smooth operations. | The platform works with eclipse, visual studio, together with notepad ++ for easy-to-use tool. Languages consist of HTML/CSS/JavaScript. It also supports libraries for general-purpose and fronted languages.  Specifically, the tools can be Java, Ruby, PHP, and Python. Licensing costs may be required.  Multiple development teams may also be considered for smooth operations. | These platforms allow for creating numerous apps using swift and android. Both the software and languages can be run on the other three platforms.  Languages consist of HTML/CSS/JavaScript. It also supports libraries for general-purpose and fronted languages.  Specifically, the tools can be Java, Ruby, PHP, and Python. Licensing costs may be required.  Multiple development teams may also be considered for smooth operations. |

## Recommendations

**Operating Platform**

The operating platform for the Draw It or Lose It web application that should be utilized is a cloud-based architecture running Linux. Cloud-based platforms hand over the maintenance, ownership, and acquisition costs to a provider that can accommodate scaling and do so in a fair pricing structure (altexsoft.com, 2018). Cloud providers alleviate complexities and inefficiencies by providing frameworks and automated services. These scalable services require less maintenance and overhead and can be accessed through a variety of different methods such as APIs and http (G. McGrath and P. R. Brenner, 2017). Utilizing these different methods allows for the web-based application to support operations distributed across many, varied computing environments.

**Operation System Architectures**

A 3-tier architecture is an architecture pattern used in applications as a specific type of client-server system. It divides the architecture into three tiers: data layer, application layer, and presentation layer. The application (middle) tier processes the business logic for the application. By doing so, it builds a bridge between the presentation layer and the data layer to maximize communication between the two. Serverless architecture utilizes modularization to build on this layer. Middle layer solutions are utilized to provide business logic, security, and content for rich, web-based applications (www.finereport.com, 2020). This modular approach allows for The Gaming Room to scale and pay for the services that are needed.

Linux OS follows this modular approach. The kernel of the operating system contains the core components, additional modules may be utilized dynamically to provide for the expansion of services (Fox, 2015). This provides for a lightweight solution which requires less disk space. Modular OS solutions delivers better application performance because of the few interfaces between the application program and the hardware (geeksforgeeks.org, 2021). Linux executes additional functions using system libraries which do not have direct access to the kernel (Fox, 2015)**.** Additional security is gained by limiting access to the device’s hardware.

**Storage Management**

Serverless solutions utilize many servers and distributed services to access different types of storage media using the interface in the storage management layer. The storage management layer simplifies the basic storage structure by integrating all the storage spaces into a storage resources pool, as well as realizes the mapping from physical device to logical view, called virtualization. At this layer, it also makes redundancy management, device condition monitoring, device malfunction maintenance and other functions possible (Zehua, 2013). This virtualization provides flexibility for program design without consideration of memory capacity (Zehua, 2013) and should be utilized with indexed, direct access storage. Indexed allocation has no external fragmentation issues and supports direct access. Better performance and uptime can be achieved by the removal of fragmentation issues.

The Linux file system architecture, using a common set of API functions, can support many different file systems on a variety of storage devices. The Linux **read** function is unaware of file system types as well as the storage medium being accessed. The Linux file system interface is implemented as a layered architecture, separating the user interface layer from the file system implementation from the drivers that manipulate the storage devices (Jones, 2007).

**Memory Management**

The Draw It or Lose It web application can benefit from Linux’s handling of memory. The Linux memory management subsystem manages the memory in the system, including implementing virtual memory, demand paging, memory allocation for internal kernel structures and user programs, and many others. Virtual memory abstracts the detail of physical memory from the application software, allowing to keep only the needed information in physical memory (demand paging). This provides for protection and controlled sharing of data between processes (The Kernel Development Community, 2021). Linux provides for two mechanisms for addressing huge pages. First, is the HugeTLB filesystem. This is a pseudo filesystem that uses RAM as its backing store. Files that are created in this filesystem resides in memory and mapped utilizing huge pages. Secondly, a mechanism that enables huge pages is called transparent HugePages. Unlike HugeTLB Pages, which requires users or system administrators to configure what parts of the system should be mapped by the huge pages, THP manages the mappings transparently (The Kernel Development Community, 2021).

Page caching is utilized whenever a file is read, data is placed into the page cache to reduce expensive disk access on subsequent reads. Data is put in page cache and eventually to the storage device when one writes to a file. Written pages are then marked *dirty and* when Linux needs to reuse them, it synchronizes the file contents on the device with the updated data (The Kernel Development Community, 2021).

**Distributed System and Networks**

Serverless architectures, allows for scaling to accommodate growth. As demand increases, the Draw It or Lose It application will need to run from additional servers. Using replication for load balancing is important to support the increased usage growth of the application across distributed systems. You can reduce the access load on a single machine by replicating directory data across servers. This improves server response time and provides for read scalability. Replicating directory entries to a location close to users can improve directory response time. The application database in a distributed system usually runs across multiple servers simultaneously. These instances must be synchronized continually and provides a single instance. Primary-Replica Replication can increase the read performance. New database servers can be created which sync up with the main one. Inserting or modifying talks to the primary database, then asynchronously notifies the replicas of the change and the need to save it as well. Another approach for scaling in a distributed system is sharding, also called partitioning. Sharding splits the server into multiple smaller servers called shards. Shards contain different records which are designated by rules. The rule dictates what types of records are located on which shard. Rules need to ensure that the data is spread in a uniform way (Kozlovski, 2018).

Load balancing on a distributed system effectively manages connection issues by redundancy. A load balancer can direct traffic to another app server instance when a server is unavailable to prevent downtime. A distributed database can be used to accomplish the same result. When database instances are not available, a new instance can be made available.

Client-server communication occurs utilizing the HTTP protocol. Multiple clients are served through an API. With Rest APIs, communication over HTTP is stateless and supports unique session data. A client request made via a RESTful API sends a representation of the state of the resource to an endpoint or requestor. This information is sent uses one of several formats via HTTP (JSON, HTML, XLT, Python, PHP or plain text). GET, PUT, POST and DELETE carry out these specific functions (Red Hat, Inc. , 2020). In a serverless architecture solutions are provided on top of individual functions.

**Security**

Security for the Draw It or Lose It application should use login credentials, username and password. Role-based access control (RBAC) is an access control method that assigns permissions to users based upon their role. RBAC is a simple, manageable approach to access management (Tunggal, 2021). Player roles should be differentiated between players and administrators. Players can play the game and manage teams whereas administrators are granted additional functions, managing the library image files, editing of the puzzles, or changing the underlying game mechanics. Persistent data should be hashed and stored to the database. The client-side browser should use TLS for the encryption of data to prevent a high jacking of a session.

A Linux user needs to authenticate to use the operating system with least privilege enforcement. This limits the scope of access to the user. Linux also protects directories and files using discretionary access control (DAC). File owners can provide additional access to others to read or writes of a file using DAC. DAC is enforced by using access control lists.

# References

altexsoft.com. (2018, March 16). The Good and the Bad of Serverless Architecture. *.altexsoft.com*, p. 3.

community, T. k. (n.d.). *The Linux Kernel documentation, 5.15.0-rc5.*

Fox, R. (2015). *LINUX with Operating System Concepts.* Boca Raton: Taylor and Francis Group.

geeksforgeeks.org. (2021, June 1). Different approaches or Structures of Operating Systems. *geeksforgeeks*, p. 1.

G. McGrath and P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance," 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017, pp. 405-410, doi: 10.1109/ICDCSW.2017.36.

Hoopes, J. (2009). *Virtualization for Security.* Elsevier Inc.

Jones, M. T. (2007, October 30). Anatomy of the Linux file system. *IT Infrastructure*.

www.finereport.com. (2020, May 9). 3-Tier Architecture: evrything You Should Know. *www.finereport.com*, pp. 1-4.

Stecher, G., Tamura, K., & Kumar, S. (2020). Molecular Evolutionary Genetics Analysis (MEGA) for macOS. *Molecular Biology and Evolution*, *37*(4), 1237–1239. https://doi.org/10.1093/molbev/msz312

Torres-Carazo, M. I., Rodriguez-Fortiz, M. J., & Hurtado, M. V. (2016). Analysis and review of apps and serious games on mobile devices intended for people with visual impairment. *2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH)*, 1–8. https://doi.org/10.1109/SeGAH.2016.7586263

Van Ullen, M. K., & Kessler, J. (2016). Citation apps for mobile devices. *Reference Services Review*, *44*(1), 48–60. https://doi.org/10.1108/RSR-09-2015-0041

Zehua, C. &. (2013). A Data Storage and Management Scheme in Cloud Storage Model. *2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, 1-4.

References

Stecher, G., Tamura, K., & Kumar, S. (2020). Molecular Evolutionary Genetics Analysis (MEGA) for macOS. *Molecular Biology and Evolution*, *37*(4), 1237–1239. https://doi.org/10.1093/molbev/msz312

Torres-Carazo, M. I., Rodriguez-Fortiz, M. J., & Hurtado, M. V. (2016). Analysis and review of apps and serious games on mobile devices intended for people with visual impairment. *2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH)*, 1–8. https://doi.org/10.1109/SeGAH.2016.7586263

Van Ullen, M. K., & Kessler, J. (2016). Citation apps for mobile devices. *Reference Services Review*, *44*(1), 48–60. https://doi.org/10.1108/RSR-09-2015-0041

Wang, H., Chen, Z., Xiao, G., & Zheng, Z. (2016). Network of networks in Linux operating system. *Physica A: Statistical Mechanics and Its Applications*, *447*, 520–526. https://doi.org/10.1016/j.physa.2015.12.084