

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 | 07/16/2020 | Dallas Phelps | Initial document including Executive Summary, Design Constraints, and Domain Model. |
| 1.1 | 8/1/2020 | Dallas Phelps | Added evaluations of requirements for Sever side, Client side, and development tools to be used in project. This listed pros and cons of each and what is required to succeed using theses tools. |

**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)

Draw It or Lose it is an Android based game that is expanding to a web-based platform. The game will need to be able to run on multiple platforms. Below are a lost of constraints that must be followed to ensure a successful product delivery.

## [Design Constraints](#_2et92p0)

***Business Constraints***:

*Budget*

The team will need to adhere to the design budget. Changes and additions to project must be within the range of the budget.

*Team*

The team will need to be experienced in web design.

*Server*

The company should ensure web servers to support web servers for the game. Servers should be maintained to increase maximum up time.

***Technical Constraints:***

*Programming Language*

The programming language used will be Java. Code from android version may be used since java is used for both projects.

*Operating Systems and Web Browsers*

The game will be running through a web browser so the operating system is a minimum factor, however the OS must be able to run one of the supported browsers.

The most common browsers per OS are as follows:

* Chrome
* Firefox
* Edge
* Safari
* Opera

Supporting these browsers will allow for a border audience.

*Game Application Environment*

The game must support multiple player on one or more teams.

The games must be unique

The team names must be unique.

Only one instance of the game service can be in memory at any given time.

To achieve this the Singleton and Iterator patterns will be used to design the application.

## [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 contains the Main() function the runs the application. This class is responsible for starting the instances and running the other classes. The ProgramDriver classes share an association relationship between the SingletonTester Class. The relations title depicts that the ProgramDriver uses the SingletonTester class in order to validate that only one instance is running.

The Four classes on the bottom of the diagram (GameService, Game, Team, Player) all share an association with each other with included multiplicity. This is stating the limits of each object in the relationship. For example, the Team shares a zero to many (denoted by “0..\*”) relationship with Player. We can read this by saying that a team can have zero or many players on it.

Game, Team, and Player classes inherent from the Entity class. This allows all the sub classes to share related attributes such as the ID and Name

****

## [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 must 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** | The major downside to Mac is the cost for licensing and software. Mac is very costly when put beside a Windows or Linux environment. The downside to Mac is upgradability. With Mac you get what you get. For server hosting this may not always be a good idea as many times servers need much more RAM. The place where Macs excel the most is with security. Macs are well known for their security against malware and viruses. | Linux is a strong candidate for any use. It is free and open source with many libraries to choose from. Since this is open source platform Operating systems can be customized to suite the needs of the developer. There are different distributions that can be used for specific needs i.e. server hosting. Works on least privilege access which helps with security. Although this helps Linux is still very susceptible to trojan horse viruses. | Windows is very strong at user access controls. Although this takes extra steps to use. Windows uses Active directory which can be set up using permissions and users can be given rights based off roles. The downfall is that windows has different versions of their OS. For server hosting Windows Server is the best method but is very expense to license out. AS one of the more popular pro-choice OS, Windows has a higher risk factor than other OS due to viruses and malware being more common. | Mobile devices running the Android OS can be used as Web Servers. These are better used for small servers, or family sites. The reason is that web servers do not normally use much processing power to run a website. However, it can be difficult to configure. Need to be connected to a pc for DNS to work properly. Mobile servers can use multiple programming languages that are supported by all major OSs. |
| **Client Side** | Development should ensure modularity that can run on multiple browsers. Modern browsers should support REST API as this offers quick responses from request. Must support authentication protocols to ensure security. Running major services on the server side would reduce client-side development require the proper transferring and response packets at quicker speeds must be used to ensure reliability for users. | Limited file structure reduces cost of programming for multiple types of clients. The use of REST ASI will ensure that the game is transferring secure and responsive HTML format to the device. Running major services on the server side would reduce client-side development require the proper transferring and response packets at quicker speeds must be used to ensure reliability for users. | These browser support Rest API. Other major web browsers are supported by windows machines, which reduces the overhead development cost, as Microsoft has worked toward unifying its development cycle for its OS. Running major services on the server side would reduce client-side development require the proper transferring and response packets at quicker speeds must be used to ensure reliability for users. | Due to no plug-in availability on mobile devices, extra work would be needed to be able to run. This would mean that the sever would need to handle most of the work. This would increase cost and overhead to be able to ensure proper functionality on mobile devices. Running major services on the server side would reduce client-side development require the proper transferring and response packets at quicker speeds must be used to ensure reliability for users. |
| **Development Tools** | IDEs such as Maven and Eclipse can be utilized in order to cut cost of licenses fees. Application is built in Java, with the REST API being used to develop into a web-based application. | Eclipse for the use of Java language. Other languages are supported such as Python and C but would only increase cost as more expertise in multi-language programmers would be needed. | Windows is known for its .Net framework and C#. However, many other languages can be used. Microsoft sales development tools at expensive licenses cost, but free open source IDEs are available for download and are supported on Windows Operating Systems. Only one team would be needed to port the game to Windows as the same languages and IDEs are used. T he only viable difference will be in code that affects the OS. | Android apps use Java as its primary programming language. This helps reduce the overhead as the applicant ion is developed in this matter. Another team that focuses on IOS would be needed in order to port the game to IOS, as Apple uses its own proprietary language called Swift. This language is derived from objective C. This raises cost of development as another team would be needed to design the client side of the application. |

## 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**: The platform the game should be run on would be a serverless architecture. There are a few reasons behind this choice. Cost, management, and portability are major factors when considering a hosting platform. Using a serverless approach reduces cost due to reduced maintenance. This is covered under your service fees for using the platform. This reduces the need of additional developers and engineers needed to maintain services.

Serverless architectures are accessible using a variety of protocols and services such as HTTP and APIs. This is useful for The Gaming Room which uses the REST API for security. Focusing on code that communicates with the architecture results in better functionality while cost is reduced from less time engineering a supporting an in-house solution.

1. **Operating Systems Architectures**: With serverless architecture multiple environments can be developed for just by setting up one environment. (Bashir, 2019) This architecture compiles many services into a modular approach, allowing The Game Room to deliver a functional secure application. This type of architecture iterates on the microservices architecture and allows for saving by using only needed services as it is charged by execution.
2. **Storage Management**: Serverless Architecture can use multiple servers and pulled from multiple storage devices at once to run an application. Indexed storage should be used as it helps to eliminate fragmentation and offers fast load times. This would also be needed to operate due to the ability to pull from multiple locations. Use of the LOOK method would work here. An index is created pointing to locations of storage and files needing to be called.
3. **Memory Management**: In serverless architecture, code is uploaded to the service party where the third party takes responsibility does what is needed for the service to run, leading them to managing the processes. However, code should be developed carefully as not to waste memory. Processes should be held on disk storage until needed. This will help with waste. Linux supports this type of storage and memory allocation through virtual memory. It calls only segments that are needed into memory, which causes faster load times.

Files are developed with index pointers that help memory calls find the right place on the disk to call from. If Linux is used only partial data need be loaded in order for the application to run, thus given plenty of time in order to load indexed locations into memory thus resulting in less wasted space.

1. **Distributed Systems and Networks**: Distributed systems are systems that are located on different machines that use messaging to act as one coherent unit. In these types of systems each unit should be able to fail independently on one another. (Gibb, 2019) These systems are common for web applications. HTTP is the protocol used in Draw It Or Lose It in order for it to run on a browser. HTTP is a stateless protocol which must use an API to help server and client communication.

Since the system runs on multiple machines, fault tolerance can be increased. If one machine fails, there can be another that acts as a redundant machine. This allows for more uptime. Using serverless architecture allows for scalability in our system. As our application scales, out system will begin to run on more servers. This is also done by the use of a load balancer. The load balancer is used between the client and the servers and accepts incoming network and application traffic and distributes the traffic to multiple servers. This ensures that there is no one single point of failure in the system. (Load Balancing, n.d)

In order for this to work successfully, the system must be synced together. This will allow for the system to act as one unit as required by a distribution system.

1. **Security**: Security is an important topic in order to protect resources. Users should be able to input a username and password for access. Any stronger form of authentication would only lessen the user base and be to restrictive for our use. Use of the REST API allows for us to set RBAC (Role based Access Control). This supports the method of least privileged access. Users are assigned player roles where only few modifications can be made in game, with no access to resource or application management files.

In order for increased security, passwords should be secured in an encrypted hash table. The encryption should be up-to-date and can able to decipher encrypted data. Data should be sent over the connection encrypted to prevent loss of data if session hijacking were to occur.

**References**

Silberschatz, Abraham, Peter Baer Galvin, & Greg Gagne. (2009). *Operating System Concepts, Eighth*

*Edition*. John Wiley & Sons, Inc. https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/.

Bashir, Faizan. (2019, July 5). *What is Serverless Architecture? What are its Pros and Cons?* Retrieved

from https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/.

Arsov, Kristijan. (2017, November 10). Microservices vs. SOA – Is There Any Difference at All? Retrieved from https://medium.com/@kikchee/microservices-vs-soa-is-there-any-difference-at-all-2a1e3b66e1be

Gibb, Robert. (2019, July 26). What is a Distributed System? Retrieved from https://blog.stackpath.com/distributed-system/

Load Balancing (Not Dated). Educative. Retrieved from https://www.educative.io/courses/grokking-the-system-design-interview/3jEwl04BL7Q