

Draw It Or Lose It!

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | <11/25/23> | Brendan Clarke | Creation |
| 1.2 | <12/10/23> | Brendan Clarke | Additional Chapters |

## [Executive Summary](#_sbfa50wo7nsh)

Our latest game Draw It or Lose It will be beginning development soon, with the latest challenge being selecting the best platform for developing this game on. Thus, together we will explore all options for identifying the benefits and drawbacks of each of our options to ensure the game can be developed efficiently and securely.

## Requirements

Some requirements for this game to be as imagined, will be for the application to render images from a large library of stock drawings to give as “Clues”, additionally for the game to run on several different operating systems each with their own corresponding device.

We need to identify the best operating platform, Stack, and hardware management solutions, as well as how we can distribute the game. Then we can take a look at how we can maintain a good user experience and keep users data private after the game is released.

## [Design Constraints](#_2et92p0)

Some issues we might run into is enabling ease of use across the board from device to device. We can look at how some similar games are run, and the quality of which they run at to best grasp at what possibilities this game truly holds, vs what we want it to do.

This might be in how high of resolution users and draw, and the speed of the rounds.

## 

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | High costs for use,  Use Servers for deploying,  Not extremely wide ranging in options but can be scaled. | Low cost for use,  A lot of open-source support, for many ways to learn how to use deployment and hosting services. | Server can be used for deployment,  Similar costs for deployment to Mac,  Easily scaled and with a wide variety of options. | No available solution |
| **Client Side** | Can use the latest version of HTML and the latest version of any web browser. | Can \*also use the latest versions of HTML, CSS, and JavaScript for the most advanced UI available. | Almost the same as Mac and Linux systems with several options for modern browsers and use of the latest web UI technology. | Different screen sizes in almost every device on the market means more effort into User experience design. |
| **Development Tools** | Has good API and developer tools with great software applications.  Will need developers to understand MacOS UI. Some software costs money for use and publishing. | No licensing cost,  Similar tools have been made available such as used on Mac.  Also has ease of use to create virtual machines to run alternative OS’s | Great with C# and the commonly used Visual studio,  Cost for licensing on some software,  May need employees to be educated in the UI. | Uses well known languages such as java, and python.  A lot of documentation and templates for coding basic applications. |

## Recommendations

Referring to Hardware and Software for development and hosting solutions:

1. **Server Hosting**: For server side, I would recommend Linux, as it is super cost-effective while being easily scaled with several opportunities for customization along the way. Linux is also easy to set up and run, meaning that the hardware required may be cheaper at first to help get the app running. With Linux being cheap it will allow for more funds to be invested into the cloud infrastructure usage on a PAAS platform.
2. **User Interface/ App development:** I would recommend using web-based languages to be accessible on almost every device, this however will mean more energy necessary for User experience to ensure most used devices can be supported.
3. **Teamwork**: Using services such as Git or Teams can help keep the team organized and on the same page. This will keep us all efficient and on task.
4. **Development tools and Hardware**: For the best way to keep our software cross platform, we will need to use Python or PHP because they have huge support for these types of applications, as well as keeping the development costs and licensing costs low.
5. **Storage and Memory Management:** To keep the user’s best interest for game play, we need to prioritize efficiency when developing the game. It will ensure that the game can run on older and less powerful devices and ensure that even users who have the latest and greatest can still play at max performance. We will utilize compression algorithms, and file formats that keep file size low. We will also use a service like Microsoft’s Azure for hosting the game. We will keep most of the game data held in the cloud and only hold core necessary code and data on the user’s device itself. We can also ensure that despite low size and easily “streamable” assets are made, that it does not affect the intended quality of the overall game, and gameplay experience. There is a perfect balance between the two that will take fine tuning once the game is completed.

Referring to Distribution of the game and extending the life of the game :

**1. Multiple Platforms:** Since our game will be running on several different devices, we will be running our code through out PaaS infrastructure and written in code that can be run on every device. We will need to investigate which platforms are still popular, potentially release a beta or early access on just one platform and distribute the game further once we can assure the game play experience and display sizes are accounted for in each release and version of the game.

**2. Network**: As mentioned above, a PaaS will be the best route for hosting and being able to stream the game and all its assets to each user accordingly. This will require a well distributed cloud infrastructure service. Azure and AWS offer both great uptime and have several locations to create a centralized application from where each user can be on different sides of the planet but still connected with low to no latency. When outages occur on a large cloud-based service such as these, the users will still be sent to backup servers. This will be the best route for ensuring the game is as accessible as possible.

**3. Security**: Microsoft, and Amazon’s platforms for cloud hosting offer great flexibility for utilizing the latest security frameworks, and high encryption. This will be a huge priority since due to the nature of how we will set up, distribute, and keep memory usage low, most of the user data, user input, and user information will need to be located on the cloud. Many government agencies operate their data through these cloud infrastructures with ease and little worry of intrusions. However, as with anything in security, even if the ham is held in a safe, in a nuclear fallout shelter 10 miles underground, it is still guarded by a human. Humans will always make mistakes; we cannot prevent this from happening totally. But what we can do is ensure employees who need access to regions of our servers only have access to the material they require. As well as maintaining a high level of education on the latest human targeted attacks. Cyber security is ever going battle, and having a dedicated team focused on data protection will be of great use.

**4. Updates**: Our model of hosting will allow for easy over the air updates, as well as being able to hold a production game, and a dev version of the game where bugs and potential security faults can be tested and updated frequently.

Conclusion,

Creating an app that is accessible, fun, and secure is extremely difficult, but not impossible. Using

the methods listed above, it can be achieved and held to its original design. This alongside following.

industry standard techniques on how we develop, how we manage user hardware, and how we

keep user data safe will ensure that if our game is fun, it will also be functional for everyone who

wishes to enjoy it.