Creative Technology Solutions (CTS)

**Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 04/10/2024 | Douglas Robinson | Initial |

**Executive Summary**

Draw it or lose it goal is to develop new ports of software for Linux, Mac, Windows, and mobile. The considerations in these goals are price, time, and projected audience size. CTS is interested specifically in a review of the merit of porting to each respective operating system.

**Requirements**

*Important*

**Design Constraints**

*Important*

**Rationale**

*Important*

**Additional Notes**

1. **Windows**
   1. Server-side
      1. All three of the traditional operating systems offer sever-style configurations for hosting this game, however I would strongly advise against putting it on a windows server. Windows is not open source so it includes a licensing fee for use, it is generally less secure than Mac and linux for two reasons; it’s a very popular choice so more resources are devoted to exploiting it, and the Operating System comes with tons of unnecessary baggage that runs in the background (assuming default settings) that are known to be targets of hackers, much of your choice of software for things like databases are limited to microsoft proprietary tools which is even more fees and licensing needed. It is not all bad as there is a reason why it’s so popular, it is easier for a beginner to use and deploy software to; even with that I don’t believe that temporary ease of labor is worth the long term licensing and security concerns. Also if windows is really needed to host, you could containerize this software or place it in a Virtual Machine.
   2. Client-side
      1. The client-side is a very different story and of all the operating systems I would recommend this one almost the most, Windows easily makes up the largest market share of desktop users and the second most of all operating systems worldwide (StatCounter 2024). This software already uses a cross platform engine (the web browser) so the primary thing that needs to change is the layout and controls of the game for Windows as most people use a mouse and keyboard for a desktop. The other thing to consider is development for the web browsers and going with the top 2 is a pretty good bet at the moment as together they make up 83.57% (StatCounter 2024). I would advise against going with any more as web browser development is notorious for being high maintenance and would require a new dedicated team per browser of developers per browser. Work on the browsers is the primary part for getting this to automatically work cross platform, so once you’ve done this for one you should probably do it for all as browsers work almost the same across desktop clients.
2. **Linux**
   1. Server-side
      1. Linux is definitely the best long term choice as it’s open-source and free, it has very good default security configurations on most distributions, and it supports tons of other free high quality software, and even though there’s no dedicated support team like in the 2 other options, there’s still a large community for Linux for help and support. The biggest benefit is that Linux has docker which allows you to do pretty much anything that the other servers can do natively.
   2. Client-side
      1. This will likely just work if developed correctly for the windows client, it may need some minor adjustments so it is definitely worth the extra minimal effort to put in to get this working on Linux, for more on the browsers, see the windows section.
3. **Mac**
   1. Server-side
      1. The Server distribution for this was discontinued in 2022 it can host servers on the client variants but the default configurations were meant for clients so A lot of work would need to be done to ensure availability, uptime and security. I would strongly advise against this as both other options at least offer an advantage over each other; this does not.
   2. Client-side
      1. This will likely just work if developed correctly for the windows client, it may need some minor adjustments so it is definitely worth the extra minimal effort to put in to get this working on Linux, for more on the browsers, see the windows section.
4. **iOS**
   1. Client-side
      1. In this case the work might already be done as the layout and controls for mobile are already developed for Android, if work is needed it will be minimal.

Development Tools

The programming languages we are going to need areType/Java-Script, HTML, Ruby, SQL and a little CSS. The Frameworks We will use for this game are Angular and Ruby-on-Rails; The database will be mySQL; The IDE will be visual studio-code; we will use vm-ware and esxi for testing. If we are on a budget this could probably be handled by a single small team of ~3-5 web-devs, but if time is of the essence I would make a Front-end team a back-end team and a team for developing automated tests. The tests will be super important as we are going to want to run these overnight on our VM’s every day a commit was made or some piece of our client’s software got updated. To my knowledge, the only thing we’ll need a license for in this plan is VMWare and the OSes we run on the VMs if we can’t find a workaround.

**Sources**

StatCounter. (March 2024). Desktop Operating System Market Share Worldwide. <https://gs.statcounter.com/os-market-share/desktop/worldwide>

StatCounter. (March 2024). Browser Market Share Worldwide. <https://gs.statcounter.com/browser-market-share>