

Draw it or Lose it

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | <07/22/2022> | Christian Blanchette | Initiated summary |

**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**

The cliente wishes to create a a game wiht a web based platform centered around guessing what another player is attempting to draw. The primary concern is expanding the games userbase by expanding into more OS platforms, as the game is only on Android phones at the moment

**Design Constraints**

The primary concern with a web based game is that the fact that expanding on to new platforms suhc as apple phones and desktops mean new versions must be maintained, as well as compatible if the cliente wishes to make the game cross platform

**System Architecture View**

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**

Currently, Entity is a class that acts as the parent to teams, players and the game instance itself, from which all core variables are acquired. Game serveice creates the game itself. There can be as little as zero games and many teamrs and zero teams to many players, as shown in the relationship arrows.

The entity class is the superclass to Teams, Players and Game instances, so all variables are inherited. Game service governs over the game itself, and there can be zero games to many teams, and zero teams to many players, as well as zero instances of game seervice to many games, as shown by the relationship arrows in the image below.



**Evaluation**

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 has to 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 existance of one universal Web browser in the form of safari means all users will have the same web browser in use, atleast on this platform | Linux is inherantly customizable, as a result many variables may complicate communication between players | The large marketshare belonging to windowsmeans that most users will be using some form of windows architecture, and will be compatable | The varying mobile devices on the market means server compatability may not be garunteed on all phone types |
| **Client Side** | The inherant uniformity of the Mac design philosophy means all clientes using a mac will have very similar OS specs | See above, the customizability extends to the client side operations as well, complicating maintenance | The history of versions no longer supported but still in use, windows 7 being the primary example, means it would be difficult to maintain compatability | Again, the variability of the mobile market means that it will be difficult to ensure the app works for all users |
| **Development Tools** | Mac and windows both have many similare IDE's supportes, so maintaining versions between the two will requie little training to maintain | Linux Shares many IDE's with the mobile market, so development between these two would increase efficiency should we expand into both markets | Being the largest market, Winsows has access to some of the most well developed and maintained development tools | All of the largest mobile based development tools use Javascript, meaning training in JS will be carried over to this project as well as other dev tools should a switch be necessary |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**: The windows and Mac Market would be a good set of platforms to move into, cementing a userbase in nearly all desktops. uniformity within each operating system means very little maintenance for the playerbase size
* **Operating Systems Architectures**: The differences between windows and Mac are many, but there is a precedant for compatability between the two, and the rather simple nature of the app means compatability is very much a possibility
* **Storage Management**: The app size is rather inconsequential, as desktop computers have access to plenty of storage. This does not necessarily mean all that space must be used, but it would certianly allow for more room for potential features in the future
* **Memory Management**: Much like the storage situation, the memory constraints are rather inconsequential, many desktops have the memory processing to managea rather simple app. There should still be care made to ensure no memory leaks or an excessive use of memory
* **Distributed Systems and Networks**: Theres little in the way of getting an app on either platform once it is available, no licensing is necessary. networking may or may not be an issue depending on if crossplay is a desired feature
* **Security**: Ensuring that no malicous data could be uploaded into an instance of the game will make sure that one compromised user cannot compromise others.