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  1.1  1.2 | 09/16/2023  09/28/2023  10/14/2023 | Mark Daley  Mark Daley  Mark Daley | Editing technical and business constraints. Evaluation of operating platform characteristics. Linux specific recommendations.  **Revision(1.1):** Updated evaluation characteristics of the different operating platforms  **Revision(1.2):** Updated recommendations to the Gaming Room |

**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**

Our client The Gaming Room is looking to develop a web-based game (Draw it or Lose it), which is currently only available on android. The fact that its only available on android at this point, I'm assuming the code for this android-based app is currently coded in Java. We can re-use Java in order to develop the web-based version of this application but some adjustments may have to be made in order to accommodate for the fact that this version will be web-based or a web-based language such as Javascript will be used. The game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit. The foundation/core of the game doesn't seem to be too complex.

**Requirements**

One of the requirements of the application is that a game will be able to have multiple teams involved with multiple players per team. Team name and player names must be unique within a game, preventing multiple players or teams to have the same name. The application will also be limited to having one game instance in memory at a time.

**Design Constraints**

Previously mentioned, this application will be web-based. So we are limited to programming languages such as Java or Javascript for optimal functionality for the back-end. With an addition of HTML and CSS in order to produce the front-end.

**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**

The diagram below shows us 5 classes which is the foundation of our game. The lines with open arrows indicate that whatever the class is pointing to is what that class inherits from the other class (inheritance). So in this UML Diagram the class Team inherits from Entity or Team "extends" Entity, making Entity the superclass and Team the subclass. The same goes for the classes Game and Player. The numbers on the line (0..\*) indicate multiplicity. Meaning that a class may have one or more items. In this case the class Team may have multiple Player objects within the class. We also have direct association within the UML Diagram with the Program Driver and the Singleton Tester. The Program Driver houses the Singleton Tester forcing them to communicate with each other. Our UML Diagram also shows how each class houses private instances and public methods (encapsulation).



**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** | Using MacOS server can offer the advantage of feature creation if on the other side a Mac client is being used. And can be done in an easier manner than compared to Windows server working with Windows clients. A disadvantage would be the cost to support a MacOS server and its limited software compatibility.  **Revision:**  MacOS for server-based deployment isn't a very popular options amongst the others. The expensive licensing costs would also be another factor to consider. | Linux servers tend to be very open-source in nature and can be very flexible in terms of customization, so you aren't limited to what features can be implemented within the server. Although this is great Linux seems to have a learning curve to it and isn't very intuitive to many.  **Revision:**  Linux servers are amongst the popular servers to use when deploying a server. A lot of Linux server distributions are free which leads to a more cost-effective server. | Windows offers great flexibility in terms of hardware support since it can handle many cores and processors. The security within the server is also another advantage. One disadvantage is that the user has all control over hardware resources which can lead to user error if settings aren’t monitored correctly.  **Revision:**  Windows is another popular OS for hosting web-based servers. Not as cost-effective as Linux but still very popular due to its wide use. Windows allows for impressive and dynamic web applications with the use of .NET and ASP.NET frameworks. Windows also offers Microsoft SQL which can be use to to manage large databases (players). | Mobile device servers can be used to control many devices through remote management (MDM). It is also a cost-effective way to host a server compared to other platforms.  **Revision:**  Some mobile devices support Mobile Device Management to remotely manage applications and deploy them |
| **Client Side** | MacOS client side would be cheap and cost-effective.  **Revision:**  Applications typically coded in Objective-C or Swift for native apps. Will have to employ a singular codebase for application to be cross-platform. | On the client side of Linux since it is open-source it would also be cost-effective and cheap, but Linux does have a learning curve making people unfamiliar with the client.  **Revision:**  .NET framework can be used in order to deploy a cross-platform web-based application for MacOS, Windows and Linux. | On the client side of Windows the cost wouldn't be as cheap as something like Linux, but due its wide use and popularity, learning Windows shouldn't be an issue.  **Revision:**  For cross-platform applications React Native can be used in order to allow application to run on multiple OS. Supports multiple languages like Java, Swift and C. | Mobile devices would also be cheap on the client side  **Revision:**  React Native can be used ensure that the code is re-usable. React Native is also coded in Javascript making it one of the most popular languages. A team can be easily assembled to develop this due to Javascripts popularity. |
| **Development Tools** | MacOS supoorts frameworks and IDEs such as Eclipse and Visual Studio code. MacOS also has a built in terminal allowing code to written inside. Code can also be compiled and ran within MacOS.  **Revision:**  One of the more popular IDE's for MacOS is "Sublime Text". Its contains many features that are common in a lot of other IDE's, making it a good standard. There is a free version of "Sublime Text" but a license is required to be purchased after a free trial. | Linux supports a web-based tool for managing servers remotely with just the use of a web browser. It can be used to monitor performance and configure services. Linux also supports the use of Text Editors and Emulator Terminals to access the systems command line.  **Revision:**  A solid IDE for Linux would be Visual Studio Code. VSC offers many extensions that make coding and developing software simple. It has a very lightweight configuration making it good for beginners. This makes assembling a developer team easy. VSC is a free to use IDE. | Windows has a wide variety of IDEs and development tools and frameworks to be used. Such as VS Code and Microsoft.net framework.  **Revision:**  Popular IDE for Windows would also be VSC. VSC supports many languages giving flexibility to a team. | Mobile devices that run AndroidOS greatly support Java, making it a widely used language when developing applications for AndroidOS. For iOS devices Swift would most likely be the language of chose, due to the fact that it was made by apple and iOS devices are optimized to use Swift.  **Revision:**  Android Studio is a popular IDE for deploying applications on mobile platforms. Also has a built in emulator for testing applications before their actual release. |

**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**: For the most flexible platform that will allow Draw it or Lose it to be expanded to multiple computing environments I think Linux would be the best operating platform to accomplish this task. Linux allows for great flexibility and is also reliable and secure. In terms of cross-platform applications, Linux wouldn't hinder Draw it or Lose it from expanding to Windows or MacOS.
* **Operating Systems Architectures**: Linux is reliable and extremely flexible. Linux is also hardware flexible allowing it to be used on any hardware that runs Windows or MacOS. It also provides good security with extensive configuration possibilities. Linux also has built in security features which make it a viable option.
* **Storage Management**: Linux generally uses the concept of logical volumes to provide storage to users. Logical volumes abstract the storage that is available to a user from the actual physical disks. Logical volumes on Linux are managed by system software called Logical Volume Manager (LVM). So games will be created with this method of taking readily available storage and using it to create games of Draw it or Lose it.
* **Memory Management**: Linus utilizes virtual memory, allowing more memory to be used than what’s actual available on the physical hard drive space. This is done by using paging, which breaks up memory into smaller memory. These pages have unique addresses so that individual memory processes can take place.
* **Distributed Systems and Networks**: Linux allows communication between systems and device with the use of inter-process communication systems. It uses three main methods which are message queues, semaphores and shared memory. A server less architecture can also be put to use, although it may be more pricey than a conventional server-centric architecture. With the application, connection and communication between devices will be accomplished by the use of a client-server architecture.
* **Security**: Linux systems employ certain methods to keep things secure. One being low automatic access rights requiring the user to have permission open files, attachments or to alter kernel options. This makes the spread of viruses and malware very minimal. The Linux system can employ firewalls also, which controls and filters incoming and outgoing traffic within the network. Encryption is also implemented within Linux systems, so even if there would be a data leak, the information would be encrypted, protecting the information.