

<Name-of-Software-Application>

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 03/16/2023 | Eric Lewis | Executive Summary and Design Constraints |

**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 would like to develop a web-based game available on multiple platforms. The current game they want to base on is Draw It or Lose It. Currently, this game only works on Android, and our client would like the game to run on other Operating Systems. The game consists of teams competing to guess what images are being drawn. There are time limits for thinking and showing the drawing mark time. The game consists of four rounds of play lasting one minute each.

**Requirements**

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

**Design Constraints**

1. Making it cross-platform we will need to transfer this game from what it currently is to a webapp.

2. Bugs, whenever porting something to a different platform it comes with bugs. We will have to run more in-depth test runs.

3. Make sure that the game can handle multiple teams. We need to make sure that the server can handle this task.

4. Make sure that every user has their own ID for the game that is unique to them, making sure that the ID is only used one time.

5. Make sure that all the images are Userfriendly so we can reach the family users.

**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 UML diagram uses mostly inheritance. The player class starts by inheriting from the team class then the team class inheritance from the game class and the game class inherits from the GameService class. The Player, Team, and Game class inherit characteristics from the Entity class. The ProgramDriver class helps build the game for the user and the SinglotonTester class allows only one player and one team to play at the same time but allows more than one game session to go on at the same time.



**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** | Apple charges $20 per server that runs on platforms like IOS. Apple also provides a wealth of support materials to help you optimize the functionality of your website. MacOS is a rapidly expanding operating system and will keep expanding. | Running a server on Linux is free. However, Linux is a server that is frequently attacked, so you must adhere to best practices for server security to keep user data safe. The fact that so few people are familiar with Linux is a major drawback. | The price of a Windows server is much higher than that of other OSs. It might cost between $100 and $200 per month. Starting can become very challenging as a result of this. However, Windows is the OS that is most frequently used globally | A server for hosting apps typically costs between $70 and $320 per month, depending on the amount of data it uses, the anticipated number of users, and overall scalability. But if your app has a larger user base, it might be able to pay for itself through in-app purchases or ads. |
| **Client Side** | MacOS requires some time and moderate  expertise. Similar to windows in price. Use Mac-specific development tools and test the application on Mac browsers like Safari to make sure it is compatible with all web  browsers. You must also confirm that you are running the most recent version of MacOS. | Out of all the OSs, Linux requires the most knowledge and time. It is inexpensive or free. You must make sure that the application runs on browsers like Epiphany, Konqueror, and many others while attempting to make it compatible with all Linux web browsers. To guarantee that the application will run, make sure to use Linux technologies. | Windows OS requires a minimum of expertise and time. Windows is comparable in price to Macs. You must make sure the application can run on web browsers designed specifically for Windows, like Microsoft Edge, in order for it to function on Windows. Using Windows development tools to make sure the application functions as intended. The application will function best on all browsers if you use the most recent version of the software. | The application would need to be developed with more knowledge and time than it would for Windows or Mac. The price would be marginally higher than that of Windows and Mac. But compared to Windows, Mac, and Linux, mobile has a larger audience. To make sure the application works with all mobile devices, there are some things you would need to do. Use the appropriate iOS development tools first, of course. Ensuring that you comprehend Apple's conditions. Last but not least, confirm that the application can be downloaded from the App Store. Since 98% of OS used are Android or IOS I chose to stay with these two in my answer. |
| **Development Tools** | Swift is the most widely used platform on Mac. Swift is open-source and simple to use. Mac users can use Notepad++ as well. MacOS also has its own IDE called Xcode made for Mac by Apple. Mac makes the development process simpler because there are no language restrictions. | The most popular IDEs for Linux include Visual Studio, Eclipse, and others. On Linux, you have the option of using any IDE, making the process simple. Linux imposes very few restrictions on developers. | Compared to Linux, Windows is much simpler to use. It is compatible with Linux's IDEs, such as Visual Studio, Eclipse, and others. The primary programming languages used on Windows are C++ and C#. The user-friendly Windows OS makes for a smoother experience, which makes everything work together even better. | You should use Swift for Apple apps and the Android NDK for Android apps. You could then use all mobile Apple and Android devices. The use of Flutter, a framework that enables developers to write code once and use it on both iOS and Android devices, can simplify the process of creating apps. |

**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**: Draw It or Lose It may be expanded by The Gaming Room to other computing environments using one of the available operating platform options. The selection of a platform will be influenced by a number of variables, including the available budget, compatibility, scalability, and performance standards. Linux is what I advise The Gaming Room to use. Linux is an affordable and open-source platform, which can make it a very good option. Linux is a great option for the server platform of The Gaming Rooms because it provides a high level of flexibility, stability, and security. Additionally, it supports a wide variety of programming languages, which can help with the creation of Draw it or lose it. This makes it simple to scale the application and allows it to be installed on a variety of platforms and hardware configurations.
* **Operating Systems Architectures**: Using the Unix architecture as its foundation, Linux is an open-source operating system. It is a popular option for server platforms because it is highly flexible, adaptable, and secure. Linux is compatible with a wide range of hardware platforms, including x86, ARM, and MIPS. The most popular architecture for Linux servers is the x86 one. It works with a variety of processors, such as Intel and AMD. The x86 architecture can address large amounts of memory and support cutting-edge features like virtualization because it supports both 32-bit and 64-bit modes. All things considered, Linux provides a high level of adaptability and compatibility, enabling it to function on different hardware architectures. Due to its adaptability, Linux applications are simple to scale and deploy across various computing environments. Developers can select the architecture that best satisfies the demands of their application, whether those demands are related to performance, energy efficiency, or compatibility with particular platforms.

*Architecture of linux - javatpoint* (no date) *www.javatpoint.com*. Available at: <https://www.javatpoint.com/architecture-of-linux> (Accessed: April 11, 2023).

* **Storage Management**: Depending on the unique requirements and use case, a variety of storage management systems are available for the Linux operating system. Here are a few of the most well-liked and frequently used choices. Most Linux distributions come with Logical Volume Manager (LVM), a scalable and adaptable storage management system. Administrators can easily resize, move, and allocate storage space as needed because it gives them the ability to manage disks, partitions, and volumes dynamically. In addition, LVM supports sophisticated features like mirroring, striping, and snapshots, making it perfect for high-availability and disaster recovery scenarios. A user-space file system interface for Linux called Filesystem in Users pace (FUSE) enables users to create and use file systems without needing root access. Custom file systems that can be mounted and accessed like common file systems can be made using FUSE. FUSE is perfect for distributed and cloud-based applications because it supports a variety of file system types, such as network file systems and virtual file systems. A distributed file system protocol called Network File System (NFS) enables remote systems to access files over a network. In Linux environments, NFS is frequently used to provide shared storage resources that many clients can access. NFS is secure and dependable for enterprise applications because it supports a variety of authentication and encryption mechanisms. Servers and storage devices are linked together by a fast network known as a storage area network (SAN). Block-level storage access is made possible by SAN, and administrators can centrally manage storage resources. Enterprise environments frequently use SAN to supply scalable and dependable storage resources for crucial applications. Overall, a variety of factors, including performance, scalability, availability, and security requirements, will influence the selection of a storage management system. After carefully weighing these factors and assessing the unique requirements of The Gaming Room, the right system should be selected.

Wikipedia contributors. (2022, December 21). Logical Volume Manager (Linux). In *Wikipedia, The Free Encyclopedia*. Retrieved 14:36, April 11, 2023, from <https://en.wikipedia.org/w/index.php?title=Logical_Volume_Manager_(Linux)&oldid=1128657117>

Wikipedia contributors. (2023, March 3). Filesystem in Userspace. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:38, April 13, 2023, from <https://en.wikipedia.org/w/index.php?title=Filesystem_in_Userspace&oldid=1142589804>

* **Memory Management**: The Draw It or Lose It software is guaranteed to operate effectively and dependably on the Linux operating system thanks to a variety of memory management strategies. In order to give each process its own address space, independent of the amount of physical memory that is available in the system, Linux first uses virtual memory management. Using this method, the Draw It or Lose It software can use more memory than the system can support without crashing or encountering other memory-related problems. Second, Linux uses page cache management to keep data that is frequently accessed in memory, minimizing disk access and enhancing performance. By using this method, the Draw It or Lose It program can cache frequently used data, including game assets, textures, and other resources, which speeds up loading times and improves gameplay. Linux also employs memory compression and swapping techniques to move unused memory pages to disk, freeing up physical memory for other processes and minimizing memory usage by idle processes. These methods aid in memory usage optimization, decrease disk access, and enhance the Draw It or Lose It software's dependability and performance.
* **Distributed Systems and Networks**: A distributed software architecture can be used to enable communication between various platforms in Draw It or Lose It. With this architecture, the application is divided into more manageable, independent parts that can communicate with one another over a network. These components can be installed on various operating systems, including Linux, Windows, and macOS, allowing the software to function on a range of gadgets. Depending on the particular needs of the software, communication between the components can be accomplished using standardized network protocols like TCP/IP, HTTP, and MQTT. However, using distributed software and networks brings with it new difficulties like connectivity problems and outages. It is crucial to take into account the dependencies between the components of the distributed system and the network connecting the devices in order to mitigate these difficulties. For instance, if one component fails, it might have an impact on the entire system, so fault-tolerance and redundancy mechanisms should be used to guarantee high availability. Additionally, network connectivity problems may result in delays or lost data; as a result, the software should be developed to gracefully handle these circumstances using methods like caching, queuing, and retrying. Draw It or Lose It can communicate with different platforms using distributed software and networks, but reliability and performance of the software must be carefully considered when evaluating the dependencies between components and the network.
* **Security**: To safeguard user data on and between various platforms, a multi-layered security strategy should be used. This strategy should include security precautions like access control, encryption, authentication, and auditing. User data should be protected with encryption both in transit and at rest, and only authorized users should be able to access it. The user's identity should be verified through authentication, and all access and data modifications should be documented through auditing. The Linux operating system has a number of built-in security features that can help safeguard user data. Linux, for example, provides access control features like user/group management and file permissions that can restrict who has access to user data. Linux also has disk-based data encryption tools like LUKS and dm-crypt. SSH, PAM, and LDAP are just a few of the authentication techniques that Linux supports in order to verify a user's identity. Last but not least, Linux provides data access and modification tracking auditing tools like auditd and syslog. By utilizing these security features and implementing additional security measures, Draw It or Lose It can ensure the security of user information on and between various platforms.