

Draw It or Lose IT

# **CS 230 Project Software Design Template**

Version 1.6

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)9

[**Domain Model**](#_8h2ehzxfam4o)9

[**Evaluation**](#_2o15spng8stw)12

[**Recommendations**](#_m8aleynsvzvc)18

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/09/2022 | Maurice Wesley | Filled in the bracketed information on page one.  Changed the footer and header |
| 1.1 | 11/11/2022 | Maurice Wesley | Summarized the client’s problems  Identified the client’s business and technical requirements |
| 1.2 | 11/11/2022 | Maurice Wesley | Translated the client’s requirements into design constraints  Explained the rationalization behind design constraints |
| 1.3 | 11/13/2022 | Maurice Wesley | Explained the object oriented principles of the UML model in the Domain Model section |
| 1.4 | 11/26/2022 | Maurice Wesley | Evaluated the various platforms for their characteristics, advantages, and weaknesses for hosting a web-based software application. |
| 1.5 | 11/26/2022 | Maurice Wesley | Determined the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients. |
| 1.6 | 11/26/2022 | Maurice Wesley | Identified the relevant programming languages and tools that are used to build this type of software for deploying on each operating platform |
| 1.7 | 12/11/2022 | Maurice Wesley | Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. |

## [Executive Summary](#_sbfa50wo7nsh)

I (Maurice Wesley) was tasked, recently, with assisting The Gaming Room with developing a web-based game that serves multiple platforms based on their current game (Draw It or Lose It). Currently, the game is only available as an Android application. More specifically, The Gaming Room wants the web-based game to have the ability to have one or more team involved and each team have multiple players assigned. Additionally, the game and team names must be unique. Functionality will involve the ability for the user to check whether a name is in use when choosing a team name and player name. Finally, The Gaming Room wants to implement only one instance of the game in memory at any given time.

**Solution**: Create a singleton and unique identifiers for each instance of a game, team, or player. Alternatively, the game modifications should be made and deployed using a .Net framework that is cross platform.

## [Design Constraints](#_2et92p0)

1. **Processes**:
   1. **Information Sharing:** The Gaming Room wants a web-based application that supports multiple games, players, and teams. Several users may want access to the same piece of information (for instance, current high score). The distributed environment must allow concurrent access to the high score while using different platforms (Silberschatz, Galvin, Gagne, 2022).
   2. **Computation speedup**: The Game wants to draw from a large library of stock drawings as clues. The images are rendered at a steady rate for a duration of 30-seconds. The processes (jobs) will have to run at the same time to retrieve the images, store them in a container, and display them successively to multiple teams and players. Also, the application could have multiples games running at the same time. This can only be achieved if the server has multiple processing units or input/output channels and the user’s platform can process the incoming information (Silberschatz, Galvin, Gagne, 2022).
   3. **Modularity**: The application should be designed in a modular fashion (multiple sub-problems). This will facilitate enhanced speed, agility, and maintenance during service interruption. Each concurrent process can access different methods (sub-problems) or the same methods depending on the required task. An implementation of a singleton class in the Game Services class creates a unique identifier for each instance of a game. Implementing Game and Team classes will regulate the unique identifiers for the teams and players respectively (Silberschatz, Galvin, Gagne, 2022).
   4. **Convenience**: A web-based application in a distributed environment will require a mechanism that allows the processes to exchange data and information. The Draw It or Lose It application will need a location to store which teams are playing a specific game, what players are on a team, and a list of players. Additionally, each game being played will need access to the large library of stack drawings (Silberschatz, Galvin, Gagne, 2022).

This will require shared memory. Shared memory provides convenient efficient communication. The processes are able to retrieve the images and display the images. Then the processes use message passing to communicate what the current team guessed and allow the other teams to know what the current teams saw, the team’s guess, and enable another team guess based on the same images (Silberschatz, Galvin, Gagne, 2022).

1. **Threads**:
   1. A single process can contain multiple threads. Most modern platforms are multi-treaded and can perform more than one job at the same time. The Draw It or Lose It application server will have the ability to run more than one thread. However, the user’s operating platform may not be able to process the incoming information and the application will need to govern the flow of information by creating a que (a list of information) or stack (Silberschatz, Galvin, Gagne, 2022).
   2. **Responsiveness**: Accessibility will impact the user’s decision to play Draw It or Lose It. The most time-consuming operation is retrieving the images from the large library. As previously mentioned, the images are displays at a steady rate for 30 seconds. The web application would have a process with a thread to display the images while another thread retrieves the next image from the large library of stock images. Multi-threading will increase the responsive to the user (Silberschatz, Galvin, Gagne, 2022).
   3. **Resource Sharing**: Shared memory, mentioned earlier, must be explicitly considered and designed by the programmer. The Draw It or Lose It application could take advantage of the inherent memory characteristic of threads. Threads share the memory and resources of the process they belong to. This allows for several different threads of activity while eliminating the need to allocate separate memory (Silberschatz, Galvin, Gagne, 2022).
   4. **Economy**: Eliminating the need to allocate separate memory for threading reduces the cost associated with process creation. Managing processes is more exhaustive than managing threads (Silberschatz, Galvin, Gagne, 2022).
   5. **Scalability**: Modern operating platforms have multiple cores on a single chip and each cores replicates a different process to the operating system. Designing the Draw It or Lose It application with multithreading provides a more efficient use of multiple cores and enhanced concurrent processing. Implementation does require the developer to design programs that are multi-threaded and write scheduling algorithms (Silberschatz, Galvin, Gagne, 2022).
   6. **Considerations**: We will need to finds ways to decompose the current application written for the Android platform into smaller concurrent task. These smaller tasks must strike a balance between efficiency and added value. Consideration must be given to the data accessed, retrieved, and/or manipulated by the threads. The data will have to be divided to accommodate running on multiple cores. Furthermore, there may be dependencies that require execution of some task to be synchronized. Finally, the testing and debugging of multi-threaded processes are more difficult. Continuous integration and concurrent testing during each sprint will be instrumental in ensuring the application meets the client’s expectation after deployment (Silberschatz, Galvin, Gagne, 2022).
2. **Memory**:
   1. **Protection**: The Draw It or Lose it web-based application will require significant memory to host concurrent games, run the operating systems, and process user requests. The developer must be conscience of the speed relative to accessing physical memory and ensuring the operating system is protected from user processes. The protection is achieved by developing a program that instructs the CPU to compare the address generated in user mode with the registers. This will prevent end game users from manipulating the game platform to achieve an unfair advantage or uploading malicious software that disrupts the application operation (Silberschatz, Galvin, Gagne, 2022).
   2. **Dynamic Loading**: A web-based application does not require the entire program to be stored on the user’s operating platform. The physical memory allocated on a server is scalable. However, the physical memory on a user’s platform is generally finite. The developer can obtain better memory utilization by writing programs with library routings that implement dynamic loading. One of the benefits is that dynamic loading does not require special configurations within the user’s operating system (Silberschatz, Galvin, Gagne, 2022).
   3. **Contiguous Memory Allocation**: I mentioned earlier that the main memory must accommodate the demands of the operating system and the user processes. A common method, known as contiguous memory allocation, divides the main memory into two partitions (operating system and user processes). More specifically, the Draw It or Lose It application would benefit from the variable petitioning scheme. The application will have multiple instances and will need to implement a table to help the operating system keep track of what parts of memory are available and which parts are unavailable. A possible example: the application could put new games into a que when the requested memory is unavailable (Silberschatz, Galvin, Gagne, 2022).
   4. **Paging**: The developer may have fragmentation (bits of memory left over that are non-contiguous) while implementing contiguous memory allocation. A possible solution is to implement paging. The developer could implement programs that allow for non-contiguous memory allocation. Paging, a form of dynamic relocation, divides the main memory into fixed blocks. We do not have to worry about external fragmentation. Any free frame can be used when it is available (Silberschatz, Galvin, Gagne, 2022).
3. **Business Constraints**:
   1. Further communication with the client is needed to determine the budget, expected time to completion, and client ability for user acceptance testing. The product owner will meet with a focus group to develop end-user stories and develop the product backlog (Silberschatz, Galvin, Gagne, 2022).
   2. The Scrum-Master will enable to product backlog and conduct Sprint planning for each sprint. The development team will implement continuous integration during upgrading the current Android application to a cross-platform application. The tester will implement concurrent testing as soon as any working code is available (Silberschatz, Galvin, Gagne, 2022).
   3. A Sprint Review and Sprint Retrospective will be held at the end of each sprint. It is important for the team not to incur any technical debt going into the following sprints. Any changes from the clients will be addressed, noted, and injected into the following sprint. If necessary, the product owner will re-prioritize user stories in the product backlog to increase agility (Silberschatz, Galvin, Gagne, 2022).

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

**Portability**: The program is modular. We know that if we have a problem with a class, then we can immediately fix that specific class. The classes can be used with other programs. The classes contain attributes that are directly associated with the class in the real world. They do not contain any ancillary attributes.

**Inheritance**: A class can implement attributes from a parent class. The Entity class is the parent class of the Game class, the Team class, and the Player class. The Game, Team and Player class all inherit the id and name from the Entity class in their constructor declaration in the UML diagram and implement super in their constructor definition. Also, the subclasses have access to the Entity class’s public member methods. The toString() methods in the Game, Team, and Player class have access to the id and name in their respective constructors by invoking the member methods in the Entity class to retrieve the object instance attributes. However, they cannot invoke the Entity constructor with an object type of the subclass (Encapsulation).

**Encapsulation**: Refers to the hidden nature of an objects data components. Another class or object cannot access the data directly. The Entity class has two attributes that are listed as private. The subclasses of the Entity class can use their constructors to create an instance of each sub-class object type with parameters to update id and string. However, this is facilitated by inheritance and utilization of super in the subclass’s constructor definition. The developer can tell that the attributes are private because of the minus sign in front of the attribute name. Additionally, developer would have to instantiate a Game object to access the teams array list and instantiate a team object to access the players array list. The program is designed to ensure that there are not in duplicate teams in the teams array list and duplicate players in the players array list.

The premise of the application is based on the program implementing a singleton object so that there is only instance in memory at a time. This is accomplished in the Game Services class. The Game Service constructor is private. Also, the singleton member variable in the Game Service class is private. A Game Service class object would call the get instance member method. This method checks if the static member variable service is null. If it is null, then the static member variable is assigned to a new Game Service object.

**Polymorphism**: Different types of objects can be accessed via a common interface. There are many different instances of Polymorphism. A Game object, Team object, and Player object are instances of the Entity class. We have run time and compile time polymorphism. The UML diagram demonstrates compile time polymorphism. The compiler knows what method to execute depending on what arguments are passed into the overloaded constructor or more specifically, the overloaded getGame() method. Game Service class has a none to many relationship to Game, Game has a none to many relationship to Team and Team has a none to many relation to Player.

The next observation includes the teams array list and the players array list. For now, we will consider the teams array list in the Game class. To add another team, the addTeam() method first creates a Team object set to null. Then it checks the current instances in the teams array list for a match to the current name passed into the method as an argument. If a match is found, then the method assigned the newly created object to the current instance in the array list. When there is not match found, then the newly created Team object is assigned to a new Team object. The new Team object constructor passes in a parameter to retrieve the current singleton instance of the next team id and the name. Finally, the new instance is added to the teams array list.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

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** | Mac provides environments that more effectively accommodate commercial software applications. An article mentions that some may prefer web-hosting development using MacOS because of the inherent nature that it shares with UNIX. Linux is a sub-category of UNIX. The terminal and features are very similar. Some developers like the minimalist style of the operating system (PixemWeb, 2022). It is rare to find a host that offers Mac hosting. The Gaming Room application is written in java. The developer would need to transform it into java byte code and install a JVM to convert it to machine code for Mac. | Requires more configurations before the system accommodates commercial software applications. Linux is the preferred system for server-side development. Linux hosting is shared, flexible, and cheap. Linux is open source and does not require a license. Linux provides support for interacting with a relational database management system where requests are made with a structured query language. Additionally, Linux is compatible with PHP which is the preferred scripting language for web development. The operating system is more secure and efficient than Windows. Linux servers seldomly need to be rebooted. The Gaming Room application would require 100% uptime and Linux would be the better choice. | Windows provides environments that more effectively accommodate commercial software applications. The Windows server’s operating system is Windows itself. It supports windows related products like ASP.Net which is Microsoft’s sever side scripting language for dynamic web pages. A developer has to use a Microsoft specific database management system such as Microsoft Access and Microsoft SQL server. Windows servers become unstable when they have to run multiple database, file, and web servers. | The biggest advantage to mobile cloud computers is that companies or developers have access to a larger target market. Developers can easily move the data stored on a mobile device and have it run on a mobile cloud. It is scalable and the developer or customer can access the data from anywhere in the world. The Cloud offers multiple platform support for different devices. There is minimal upfront cost and no hefty licensing fees like Windows. Backup and restore is easy due to the disaster recovery software in the Cloud. Some of the disadvantages include security. Choosing the most reliable service provider could enhance security. Cloud computing depends on the internet. The mobile user may have spotty coverage or be on a throttled data plan. In many cases, it is difficult to switch from one provider to the next. |
| **Client Side** | Amazon Web services provides a dedicated server for Mac known as an EC2 instance. The pricing is around six dollars per hour or $4320 per month for a general purpose m5n instance for MAC. It takes about 18 minutes to setup an EC2 instance. The average amount of time could be between a few minutes to a few days. Expertise needed will encompass employees familiar with network, security, software, and website development. If the IT team is comfortable setting up a server, then an unmanaged dedicated server might work well. Otherwise, a managed dedicated server might be better. | Dell provides a SUSE Linux Enterprise Server, 1-2 SKT with unlimited virtual machines, level 3 subscription over 3 years and digitally fulfilled for $8100. It takes about 10 minutes to setup a Linux Enterprise Server. Expertise needed will encompass employees familiar with network, security, software, and website development. If the IT team is comfortable setting up a server, then an unmanaged dedicated server might work well. Otherwise, a managed dedicated server might be better. | Windows Server 2022 Data Center provides a dedicated web hosting server for Windows. The client must purchase a Windows license to use the cores. The price is $6155 per month. The server is meant for highly virtualized machines. It takes between 10- 20 minutes to setup a Windows Server. Expertise needed will encompass employees familiar with network, security, software, and website development. If the IT team is comfortable setting up a server, then an unmanaged dedicated server might work well. Otherwise, a managed dedicated server might be better. | Amazon cloud hosting services has no upfront cost and a pay as you go pricing structure. This makes entry to cloud computing for mobile application web hosting very affordable. Amazon offers a m4.large on demand cloud compute package for $874.40 per year. Additional savings are available for reserved instances. Amazon utilizes AWS CloudFormation to automatically deploy the entire setup. They have a GitHub repository that contains all templates and code needed. Similar to an EC2 instance for Mac, the decision to purchase a managed or unmanaged EC2 instance depends on the comfort level of the IT team and/or development team. |
| **Development Tools** | Programming will be done using the cross-platform language java. Eclipse is a free open-source tool and is an adequate IDE. The Java Virtual Machine framework works in transforming the java files into machine code. The associated JVM software will need to be install in the operating system. Afterwards, the IT team will have to log into the host manager on Amazon. Set up networking, resolvers, IP address, configure nameservers, and services.  Alternatively, Amazon utilizes AWS CloudFormation to automatically deploy the entire setup. They have a GitHub repository that contains all templates and code needed. Additional tools/requirements are an AWS Account with sufficient permissions, a computer with AWS command Line interface, unix shell, SSH client supporting the ssh\_config, a remote desktop client. However, deploying this example with incur a $50 charge because Mac instances can only be released 24 hours after allocation. | Programming will be done using the cross-platform language java. Eclipse is a free open-source tool for the IDE. The Java Virtual Machine framework works in transforming the java files into machine code. The associated JVM software will need to be install in the operating system.  The Dell dedicated services has an expert team available through live chat to help with after sales service. Additionally, the IT team will have to configure the network interface, IP address, gateway, and netmask. | Programming will be done using the cross-platform framework. Visual Studio is an adequate tool for the IDE and it is free. The .NET framework works in transforming the files into machine code for deployment in a distributed environment. The associated .NET software will need to be install in the operating system. Microsoft mentions that the highly diverse scope of potential deployments makes it unrealistic to state recommended hardware requirements. If The Gaming room wants to install it on a local computer. Minimum 1.4 GHz 64-bit processor, supports NX and DEP, supports second level address translation. Minimum of 512 MB. The computer must have an available storage compacity of 32GB and 4GB for the GUI. The system partition does require additional space. The Network adapter should be capable of 1 GB per second and compliant with the PCI Express architecture specification. Additional tools: internet access, trusted platform module, Graphics device and monitor capable of Super VGA, and other peripherals. | There are two widely used frameworks for developing cross-platform mobile applications. The client or development team could choose Flutter (from Google) or React Native (uses React and Java Script). Both frameworks are free and the developers can use Visual Studio Code. The test flight app that is owned by Apple will have to test the application for deployment on IOS. Also, we will need an Apple developer account. We can invite up to ten thousand users during beta testing. After testing we can release the application for IOS by submitting it to the App Store. The application currently is written for Android. The hardware required is minimal because AWS does all of the heavy lifting. However, during deployment, the underlying hardware on which you are deploying the gateway VM needs four or eight virtual processors assigned to the VM, 12 GB of RAM assigned to the VM, and 80 GB of disk space for installation of VM image and system data. |

## 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**

Linux is a free and open-source operating system that has been around since 1991. Linux has extensive documentation, but lacks application support. It is easy to use, customizable, provides a higher level of security, and can be ran on other computing environments. As I previously mentioned, Linux servers seldomly need to be rebooted.

The Gaming Room application would require 100% uptime and Linux would be the better choice. Dell provides a SUSE Linux Enterprise Server, 1-2 SKT with unlimited virtual machines, level 3 subscription over 3 years and digitally fulfilled for $8100. Programming will be done using the cross-platform language java. Eclipse is a free open-source tool for use as an IDE. The Java Virtual Machine framework transforms the java files into machine code. However, the associated JVM software will need to be installed in the operating system and/or user’s operating system.

**Operating Systems Architectures**

The Linux architecture facilitates communication between the hardware and the software by managing the hardware resources and connected peripherals. In this section, I cover five components of the Linux Operating System Architecture.

**Kernel**: Considered the foundation of the operating system, the kernel manages network communication, file management, and hardware integrations. The kernel keeps track of memory, what processes are running, and process requests for service from processes (Red Hat, 2019). The kernel is open source and anyone can download and redistribute it.

**System Library**: A collection of non-volatile resources used by programs to perform specific tasks. The shared library is the most common system library in the Linux operating system. The shared library is a collection of libraries that are loaded when a program is executed, put in memory, and bound at run time. The Gaming Room application will need to perform math operations to calculate points accumulate, leading team, and how many teams are playing. The developer can use a built-in library to perform these tasks (Kili, 2022).

**Hardware layer**: The hardware layer contains the physical machine and the connected peripheral devices. These devices include: the central processing unit (reads memory, writes to memory, and performs computations), hard disk drive (secondary storage), and random-access memory (loading programs and memory for executing requests). This layer also has a hardware abstraction layer (HAL). The HAL is a layer of programming that allows the Linux operating system to interact with a hardware device at a high level instead of at a low level (Study Buff, 2022).

**System Utility Programs**: These programs provide file management and system administration. Additionally, the user can manage network settings and disk partitions. Linux has a central location where developers can find and download applications like Ubuntu to manage applications in a distributed network.

**Shell Functions**: Most developers interact with the operating system through the shell. The shell provides an interface between the developer and the kernel, receives user commands, and translates them into kernel calls. It is a programming language that enables shell scripts to automate common tasks.

**Storage Management**

Oracle database is a premiere provider of storage management solutions. They provide utilities for transaction processing, data warehousing, and mixed database workloads. Oracle Database supports data reduction, hybrid columnar compression, cluster file system, and cloud services (Ubuntu, 2022). Oracle has a wide range of partitioning schemes to increase performance (work with relevant data), availability (partition manageability), and decrease cost (storing data appropriately).

Oracle classifies multiple Linux server storage management services under the category of Real Application Clusters. I would like to focus on Automatic Storage Management (ASM) and Automatic Cluster File System (ACFS).

**ASM**: Enables consistent and simplified automated storage management across the server. It increases storage utilization and agility. The storage is managed with a small number of storage pools called disk groups. Each disk group contains a collection of related files. ASM governs the layout and data organization to enhance performance and protection. It consolidates storage and simplifies the administrative tasks of system administrators. Additionally, ASM provides the framework for ACFS.

**ACFS**: Provides support for multiple server platforms. It allows for file-based and filesystem-based snapshots, giving the user an ability to efficiently provision test and development environments. Furthermore, ACFS implements encryption for flexible and secure storage management. The Gaming Room can leverage ACFS replication capabilities for disaster recovery. Also, ACFS has automatic resizing, support for sparse files, file system freezing, and metadata acceleration (Oracle, 2022).

**Memory Management**

The developer must be conscience of the speed relative to accessing physical memory and ensuring the operating system is protected from user processes. The protection is achieved by developing a program that instructs the CPU to compare the address generated in user mode with the registers. This will prevent end game users from manipulating the game platform to achieve an unfair advantage or uploading malicious software that disrupts the application operation (Silberschatz, Galvin, Gagne, 2022).

The physical memory allocated on a server is scalable. However, the physical memory on a user’s platform is generally finite. The developer can obtain better memory utilization by writing programs with library routings that implement dynamic loading. One of the benefits is that dynamic loading does not require special configurations within the user’s operating system (Silberschatz, Galvin, Gagne, 2022).

More specifically, the Draw It or Lose It application would benefit from the variable petitioning scheme.

The application will have multiple instances and will need to implement a table to help the operating system keep track of what parts of memory are available and which parts are unavailable. A possible example: the application could put new games into a que when the requested memory is unavailable (Silberschatz, Galvin, Gagne, 2022). Also, the address of an image in storage will be in main memory and the images will be retrieved from secondary memory when need.

Memory management impacts the execution time of processes and ensures the accuracy and availability of data from the secondary memory (storage management). It guarantees that the right data is present in the main memory. An example would be the correct image being displayed and/or the correct teams are playing a game. The data has to move to and from main memory. More specifically, the current image is in main memory and replaced with the next image.

The operating system must not relocate the data if it is being used by concurrent processes. Memory management ensures that a process can access data it has rights to and restricted from accessing other processes. Similarly, the game application implemented a singleton class where only one instance of a game was allowed to be in main memory. Conversely, the memory management system must facilitate the ability for multiple processes to share the same resource. The game application will have multiples concurrent games. The game processes may need to access the same image/resource.

As I previously mentioned, the available space in main memory is small. There may be small memory chunks available after process execution. The memory management system has to implement defragmentation techniques for the efficient utilization of the main memory.

**Distributed Systems and Networks**

A distributed system must have a network that connects all of the components to enable the transfer of data and/or information between them. It contains multiple nodes that are physically separate but are linked by a network, cable, or antenna. Early games were developed using a peer-to-peer architecture. The game was located on many different machines and players interacted with each other. The Gaming Room application implements a Client/Server distributed systems architecture. Now, the game is located on the server and players interact with the server (Apache Booster, 2018).

The components of the system are the various nodes (operating platforms), the centralized computer system (web application), and the coordinated activities and shared resources managed by the storage management system (in this case Oracle). The distributed system software enables computers to coordinate their activities and to share the resources (hardware, software, and information) (Apache Booster, 2018).

The Linux server can provide 24-hour uptime that is very reliable and secure. The Gaming Room’s application is heavily dependent on the server. The application would not work if the operating platforms are unable to connect to the server. The operating platform sends a request to connect to the server. The server then responds by asking the computer/user to provide the proper credentials. The user/operating system sends the credentials to the server. The sever creates a connection with the user if the credentials are valid. The Gaming Room can leverage Oracle’s replication during outages. The web-application data is backed-up in another location. The Gaming Room could easily move the stored data to another location and resume services.

The data and information for the game are transferred between the user and the server through a series of request. The client side sends inputs in the form of mouse movements, sticks movements, and/or button presses. The server side processes these inputs/requests and returns the aggregate along with the requests from other players (Apache Booster, 2018).

I mentioned that the game would be developed in java using Eclipse as an IDE, and distributed using the appropriate JVM installed in the user’s operating system. The initial code downloaded from the app store will contain the associated JVM software. This code enables the user’s operating system to communicate with the server through a plug-in or connector.

The client-server architecture decreased the amount of band-width that was needed for each individual player. This enabled more players for a specific instance of a game. The client-side executes of sub-segment of the code which foretells the action of the user and reduces the latency or response. The server/application has a predetermined refresh rate that dictates the number of times, per second, the information is updated. The information updates can be sent to every player using property bags instead of individual send requests (Apache Booster, 2018).

**Security**

The user space interacts with the kernel through a system call interface. The CPU operates in kernel mode and user mode. Code running in kernel mode has unrestricted access to the hardware (Red Hat 2018). The separation prevents any process that fail or processes that try to access privileged information from causing harm to the rest of the system.

**Zero Trust**: At its core, Linux provides Security-Enhanced Linux which allows administrators to have more control when determining who has the proper credentials to access the information system. Subsequently, Linux incorporated Zero Trust IT security. Traditional network security starts inside of a firewall. Conversely, Zero Trust is based on the principle of designing network security where every interaction begins in an untrusted state. It is an attempt to prevents gaps in security architectures that implement one-time authentication.

John Kindervag challenged the prevailing consensus to design network security based on the “hard shell, soft center” principle. Under this methodology, individuals within the perimeter had a higher level of access than individuals on the outside. He mentioned that this was an insufficient architecture because cybercriminals can garner help from individuals on the inside of the shell and move laterally to launch a cyber-attack. The vulnerability increases when the network contains a vast number of endpoints.

The Zero Trust Network Access segments access and limits user permissions to specific applications and services (Red Hat, 2022). The Zero Trust model requires The Gaming Room to apply a “verify and never trust” philosophy. This will prevent any attackers that breach the hard layer of protection (firewall) from moving laterally.

**API Security**: The Gamming room must consider API security. The Draw It or Lose It used an API to build a maven application that implemented authentication and annotations. Broken, exposed, or hacked APIs are behind major data breaches (Red Hat, 2022). The Gaming Room REST API supports Transport Layer Security (TLS) encryption. TLS keeps an internet connection private and checks that the data sent between two systems (a server and a client) is encrypted and unmodified (Red Hat, 2022).

There are steps that The Gaming Room can take to increase the API’s security. It is recommended to use tokens to establish trusted identities and then control access to services. TLS will help ensures that the right individuals are updating and modifying data. Also, continuously monitor the API components and identify weak spots that can used to break the API (sniffers). I recommend using an API gateway. An effective gateway authenticates traffic and analyzes how the Draw It or Lose It application API is being used (Red Hat, 2019).

**Access Control**: It represents the user for which credentials have been supplied with a request. Similarly, it symbolizes the security context within which the code is running. The Gaming Room application implements role-based security rights associated with a principal object. In the Microsoft documentation, the principal object is bound to a call context that is created by default with each new AppDomain. The application domain-specific policy is configurable. Depending on the security policy, the runtime can create a principal object that reflects the operating system token associated with the current thread of execution. The runtime uses the object to represent unauthenticated users. When the Principal object is transmitted across application domains with the same process, the remoting infrastructure copies the principal object’s reference to the callee’s context (Microsoft, 2021).

**Flow Control**: This applies to the flow of information contained in objects and ensuring that this information does not travel through unsecured channels. At a high-level, proper training must be provided to employees for enhanced situation awareness to prevent onlookers from acquiring unauthorized system privileges. The flow of information should be sent through a secure channel that continuously monitors traffic and sends notifications during connection, transit, and completion.

**Stack Inspection**: An extension of the principle of least protection is stack inspection. The stack inspection counter-measure is contained within the java virtual machine for hosting in a distributed environment. Protection and authentication are based on privileges. Not every method is allowed to assert a privilege. Based on trust, a method executes an access request within a privileged block to perform the access to a protected resource directly or indirectly.

Subsequently, a call to the check permission method is used to invoke a stack inspection to determine if the request should be allowed. The aforementioned reasons make programing in java more secure than C++ and Python. A java program cannot directly access memory through the use of pointers. This further validates my recommendation to develop the Gaming Room application using java and the associated JVM framework.

**RBAC**: Implementing separate user accounts could fall under the umbrella of role-based access controls. The Draw It or Lose It is a maven application which implements a drop wizard authenticator to assign role names and an authorizer to retrieve them. The game rest controller java file uses annotations to assign certain privileges based on a role. A role can be given to a process with the privilege to execute a specific system call. Similarly, the privilege can be removed with varying degrees of revocation.

**Resources**

Silberschatz, Galvin, Gagne. (2022). Operating system concepts. [White Pages]. Retrieved November 11, 2022 from [Title Page | Operating System Concepts, 8th Edition (oreilly.com)](https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_tp_r1.html)

AWS. (2022). Amazon ec2 dedicated host pricing. [White Papers]. Retrieved November 27, 2022 from [Amazon EC2 Dedicated Host Pricing](https://aws.amazon.com/ec2/dedicated-hosts/pricing/)

Otey, Michael. (2022). Understand windows server 2022 licensing. [Blog Post]. Retrieved November 27, 2022 from [Understanding Windows Server 2022 Licensing | Petri](https://petri.com/understanding-windows-server-2022-licensing/#:~:text=Each%20Windows%20Server%20Datacenter%20and%20Standard%20edition%20server,physical%20cores%20of%20the%20server%20must%20be%20licensed.)

Dell. (2022). Dell price list 2022. [Web Page]. Retrieved November 27, 2022 from [Dell SUSE LINUX ENTERPRISE Price - Dell Price List 2022 (itprice.com)](https://itprice.com/dell-price-list/suse%20linux%20enterprise.html)

Chen, Phil. (2019). How long does it take to launch an amazon ec2 instance. [Blog Post] Retrieved November 27, 2022 from [How long does it take to launch an Amazon EC2 instance? | Phil Chen](https://philchen.com/2009/04/21/how-long-does-it-take-to-launch-an-amazon-ec2-instance)

GTHost. (2022). How long does it take to set up a dedicated server. [Blog Post]. Retrieved November 27, 2022 from [How Long Does It Take To Set Up A Dedicated Server? - GTHost](https://gthost.com/blog/how-long-does-it-take-to-set-up-a-dedicated-server/)

Windows. (2022). Hardware requirements for windows server. [White Pages]. Retrieved November 27, 2022 from [Hardware requirements for Windows Server | Microsoft Learn](https://learn.microsoft.com/en-us/windows-server/get-started/hardware-requirements)

Red Hat. (2022). 3 ways to configure a network interface in linux. [Blog Post]. Retrieved November 27, 2022 from [3 ways to configure a network interface in Linux | Enable Sysadmin (redhat.com)](https://www.redhat.com/sysadmin/network-interface-linux)

Armstrong, Rick. (2021). Setting up ec2 mac instances as shared remote development environments. [Blog Post]. Retrieved November 27, 2022 from [Setting up EC2 Mac instances as shared remote development environments | AWS Compute Blog (amazon.com)](https://aws.amazon.com/blogs/compute/setting-up-ec2-mac-instances-as-shared-remote-development-environments/#:~:text=Prerequisites%201%20An%20AWS%20Account%20with%20sufficient%20permissions,VSCode%20with%20Remote%20SSH%20Extension%20installed%20and%20configured)

Nego Sentro. (2016). What are the advantages and disadvantages of mobile cloud computing. [Blog Post]. Retrieved November 27, 2022 from [What Are The Advantages & Disadvantages Of Mobile Cloud Computing? (negosentro.com)](https://negosentro.com/what-are-the-advantages-and-disadvantages-of-mobile-cloud-computing/)

Amazon Web Services. (2017). Costs in the cloud: how much does aws cost? [Blog Post]. Retrieved November 27, 2022 from [How much does AWS cost? Cutting your cloud costs using AWS (cloudacademy.com)](https://cloudacademy.com/blog/how-much-does-aws-cost/)

Vidyaikin, Alexandr. (2021). What is mobile app hosting and why do you need it. [Blog Post]. Retrieved November 27, 2022 from [What Is Mobile App Hosting And Why Do You Need It? - Purrweb](https://www.purrweb.com/blog/mobile-app-hosting/)

Mueller, John. (2017). Hardware requirements for aws developers. [Blog Post]. Retrieved November 27, 2022 from [Hardware Requirements for AWS Developers - dummies](https://www.dummies.com/article/technology/information-technology/networking/cloud-computing/hardware-requirements-aws-developers-246477/)

Red Hat. (2018). Understanding IT security. [Blog Post] Retrieved December 10, 2022 from [Understanding IT security (redhat.com)](https://www.redhat.com/en/topics/security)

Red Hat. (2019). API security. [Blog Post]. Retrieved December 11, 2022 11, 2022 from [What is API security? (redhat.com)](https://www.redhat.com/en/topics/security/api-security)

Red Hat. (2022). What is zero trust. [Blog Post]. Retrieved December 10, 2022 from [What is Zero trust? (redhat.com)](https://www.redhat.com/en/topics/security/what-is-zero-trust)

Kili, Aaron. (2022). Understanding shared libraries in linux. [Bog Post]. Retrieved December 11, 2022 from [Understanding Shared Libraries in Linux (tecmint.com)](https://www.tecmint.com/understanding-shared-libraries-in-linux/)

Study Buff. (2022). What is the hardware layer linux. [Blog Post]. Retrieved December 10, 2022 from [What is the hardware layer Linux? - Studybuff](https://studybuff.com/what-is-the-hardware-layer-linux/)

Interview Bit. (2022). Linux architecture: detailed explanation. [Blog Post] Retrieved December 09, 2022 from [Linux Architecture - Detailed Explanation - InterviewBit](https://www.interviewbit.com/blog/linux-architecture/)

Ubuntu Pit. (2022). The 15 best database management systems for linux desktop. [Blog Post] Retrieved December 10, 2022 from [The 15 Best Database Management Systems for Linux Desktop (ubuntupit.com)](https://www.ubuntupit.com/best-database-management-systems-for-linux/#:~:text=Best%20Database%20Management%20Systems%201%201.%20Oracle%20Database,CUBRID%20...%208%208.%20SQLite%20...%20More%20items)

Oracle. (2022). Oracle ACFS technical overview. [White Papers]. Retrieved December 11, 2022 from [Oracle ACFS 21c - Technical Brief](https://www.oracle.com/a/tech/docs/oracle-acfs-technical-brief.pdf)

Silberschatz, Galvin, Gagne. (2022). Operating system concepts. [White Pages]. Retrieved November 11, 2022 from [Title Page | Operating System Concepts, 8th Edition (oreilly.com)](https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_tp_r1.html)

Microsoft. (2021). Principal and identity objects. [Article]. Retrieved December 08, 2022 from [Principal and Identity Objects | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/standard/security/principal-and-identity-objects)

Geeks For Geeks. (2022). What is a distributed system. [Blog Post]. Retrieved December 9, 202 from [What is a Distributed System? - GeeksforGeeks](https://www.geeksforgeeks.org/what-is-a-distributed-system/)

Apache Booster. (2018). How does a game server work to handle multiple players. [Blog Post]. Retrieved December 11, 2022 from [How does a game server work to handle multiplayers? (apachebooster.com)](https://apachebooster.com/blog/how-does-a-game-server-work-to-handle-multiplayers/)