

The Gaming Room

# **CS 230 Project Software Design Template**

Version 3.0

## 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)3

[**Domain Model**](#_8h2ehzxfam4o)3-4

[**Evaluation**](#_2o15spng8stw)4-10

[**Recommendations**](#_m8aleynsvzvc)11

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 06/14/2022 | Travis Barbur | Third version. The evaluation section has been implemented. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room, which is a web-based game called “Draw It or Lose” is from a popular game show from the 1980’s and is available on Android as of right now. Furthermore, it’s an application that is designed to run efficiently on different platforms. The game will consist of four rounds of play lasting one minute each. The application will render images from a library stock for clues. The user will guess a phrase, title, or thing. This is a design template that will explain and highlight the application of the game. The application must incorporate multiple teams and players within the game. Various languages should be implemented separately so that the game can run on different operating services. The idea is to guess to solve the puzzle within 15 seconds before the time runs out.

## [Design Constraints](#_2et92p0)

In order for the game to run on efficiently on different operating services there are certain requirements as shown below that will need to be considered before deploying the game. This is an Android base game only so we may need developers to inherit languages to multiple planforms. Such languages for mobile devices are Android, IOS, Windows OS, and Symbian. The goal is to align the application so that it’s able to run on all mobile devices. The Name and ID identifiers should be unique, so they won’t cause any issues within the program with several alike identifiers for a single game instance memory at any given time.

* Game application platform availability.
* Multiple teams and multiple players.
* Game and team name identifiers.
* One instance of the game can exist at any given time.

## [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)

As seen below from the UML, the Entity class was created as part of this project. This class is called the “super class” because it was derived from classes Game, Team, and Player. The Entity file stores the identifiers such as “Id and “name.” This file has attributes, constructors, and methods. The GameService references to the Game, then the Game referenced to the Team and then to Player which all have a relationship. The ProgramDriver.java boards the main () file that executes all files involved. The ProgramDriver and the SingletonTester have a relationship in that only one game is run at any given time.

**"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)

Server Side

Apple (Mac):

<https://appleinsider.com/articles/17/10/14/why-you-want-a-macos-home-server-and-how-to-get-one-going> (How to create a home server with your old mac desktop.)

Linux Advantages and Disadvantages:

<https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/01%3A_Introduction_to_Linux/1.08%3A_Advantages_Disadvantages_of_Linux>

Windows:

<https://www.hitechwhizz.com/2021/04/5-advantages-and-disadvantages-drawbacks-benefits-of-web-application.html>

Mobile Devices:

<http://paw-android.fun2code.de/>

(Building your very own Android server from a mobile device)

<https://serversman.com/en/device/sm_iphone.html>

(Building a Web Server on an iPhone)

Client Side

Apple (Mac) Cost:

<https://spdload.com/blog/website-development-cost/>

Linux cost:

<https://linuxhint.com/redhat_linux_pricing/>

Windows Cost:

<https://www.cleveroad.com/blog/web-app-development-cost>

Mobile Devices Cost:

<https://spdload.com/blog/app-development-cost/>

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The Apple (Mac) is a UNIX 03 – compliant operating system. Certified by “The Open Group” since 2007. The OS offers control, security, virtual memory efficiency, a large set of commands, unified file system, portable operating system, optimized for program development and is customizable. The downside is learning how to use cryptic commands, multiple utilities, understanding the OS design features and cost. CSS Edit package for CSS editing is smooth. Adobe services are well versed with Mac OS. Time machine back up is a nice feature and the colors that Mac comes with. The biggest issue with hosting a Mac web-based software application is the cost. Every time a new software version (update) is released, the web host must pay to get the latest version updated. The cost to develop a web based application can cost from $3000 to $80,000 dollars. | Linux is a free open-source OS. Unlike Apple (Mac) it has a large community that’s helpful in getting started with web development and building software applications. This OS is the preferred choice for building a web-based software application. It’s much easier to maintain with less hours spent on upkeep. The OS works on any older computers and less vulnerabilities. The downside of Linux are some applications used from other OS systems might not work, steep learning curve with Linux commands and majority of the server-side programs run under a Windows OS since they’re design for that platform. Docker is natively run-on Linux but some applications such as Adobe and Microsoft is not widely supported. It will take a lot of time to configure the system to operate the way you want it to which can take a lot of time. | Windows which most of know is made by Microsoft. A wide variety of popular applications that are available for web-based software applications such as Excel, OneNote, Outlook, PowerPoint, sway, Word, OneDrive, Microsoft Teams, Skype, and the list continues. 95% of clients utilize Windows OS because it’s easier to utilize windows for creating software applications. Windows hosting can run ASP, .NET, Microsoft Access and MSSQL databases. Windows proprietary directory service AD (Active Directory) runs on the Windows Server. This is good for managing permissions and creating groups, users and much more. The downside of Windows are the vulnerabilities within programming development, mandatory windows updates are too often and can cause problems with other applications or within the system. | Mobile devices are not typically known to host web-based software applications due to their size of storage and memory but there are options. An Android user can create a web server application by using a PAW Server. There’s a feature for iPhone as well called “ServersMan@IPhone that allows the phone to turn into a Web server. Native mobile apps are typically built for a specific platform such as Apple, Android, or a Windows device. |
| **Client Side** | Costs are similar to Windows. To develop an Apple app, you’ll need a membership with Apple Developer Program that cost around $99 annually. Typical browsers used in an Apple (Mac) client environment are Vivaldi, Brave, Safari and Chrome. Recently, Google has mentioned to stay away from Chrome due to its security vulnerabilities. Mac runs quiet well with Microsoft applications. Costs overall are very similar to Windows. Expertise level is moderate, and time is required. | Linux comes with a default browser Mozilla Firefox once installed but other browsers can be installed such as Google Chrome, Opera, Chromium, Vivaldi, Konqueror, QupZilla and more. Linux executes quickly, usability to Web hosting environments that support scripts and have more control over Web hosting applications. Auto play blocking might be a problem when developing a software application on the browser. A cost of $299 will get a Linux user a Red Hat Enterprise Linux Workstation to meet software developers’ expectations. Linux needs much more expertise, and the costs are in-between Mac and Windows. | Microsoft Windows by default comes with Microsoft Edge as a browser. It used to be IE (Internet Explorer). The preferred browsers used are Chrome with (67%), Edge (7.75%), Firefox (7.48%) and Internet Explorer (5.21%). Cost to create a web app can range from $5,000 to $500,000 dollars. With Windows you need the bare minimum expertise level because of the usability and features. The cost lines up similarly to Mac. | The popularity in mobile devices have been Google Chrome or Safari. Other browsers such as Mozilla Firefox, Internet Explorer (IE), Microsoft Edge and Samsung Internet. The big issue is being able to develop touch controls vs the input/output devices that are widely available. A user not able to cache the phone as easily, but user still can cache the phone. There’s a process to caching the phone but it’s not done as easily. It is easier for investors and developers to view the updates in a timely manner. Since it’s a handheld device it would be much more difficult to implement a web- based application on a mobile device |
| **Development Tools** | Apple (Mac) language consist of swift for our primary language. XCode runs on the Apple environment but other IDE’s can be deployed such as Eclipse, IntelliJ (Python, C++ and Java), VSCode is another good with that works well with Apple. Other IDE’s such as Atom, Cloud 9 that support cross platform, JS, Ruby, PHP< C and Python that’s used for multiple teams. The preferred development tool to use in the Apple environment would be XCode and the cost is $99 per year to use all of the features. | Linux relevant programming languages are Python, C++, C, Perl, Java and Go. IDE tools used are Atom, Delphi, Cloud 9 Embold, Azure, Eclipse, and Jira. Most of these applications supported are open source and some are just apps with plug-ins. | Windows languages is done mostly in C with some assembly. Visual Studio, Visual Studio Code, Eclipse is the most used IDE’s in a Windows environment. These IDE’s can be tested on any web browser that I have listed previously. | Currently there are no developmental tools are available or used in today’s web development. |

## Recommendations

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

1. **Operating Platform**:

In doing the necessary research I’ve concluded that the Linux OS platform may be the best recommended choice in terms of server usage. It doesn’t matter if someone on the other end is using an Apple (Mac) or Microsoft (Windows) machine because the server is separate from the client-side. There’s less upkeep in maintaining a Linux server. As I’ve mentioned previously, the Linux will require some knowledge in understanding the commands but in time anyone can achieve this. In the long run you will save a lot of time, headaches, and money. The Linux operating platform only needs a reboot when a kernel update is needed unlike Apple (Mac) and Microsoft (Windows). The Linux machine provides more security in terms of admin access and rights to root privileges.

1. **Operating Systems Architectures**: The Linux is comprised of 4 layers.

* Hardware: All physical components that are attached to the system such as the HDD (Hard disk drive), RAM, Motherboard, CPU etc. All these physical items represent the hardware for the Linux OS.
* Kernel: The kernel is the core if the Linux OS. The kernel interacts with the hardware that provides low-level services. Without the kernel there is no OS. It’s responsible for memory, processes, and file management of the game Draw It or Lose It.
* Shell: Picture this as a skeleton. It’s an interface between the user and kernel that accepts commands from the user and performs the action as requested to the kernel.
* Applications: Typically known as a software that allows a user to perform a task such as desktop tools, text editor, media players, web browser etc. They’re utility programs that run on shell.

The kernel space and the user space are two separate areas within the virtual memory. By separating these spaces, it helps to keep the system more stable. It prevents applications from directly accessing the hardware which means that if a process fails in the user space the damage is limited and can be recovered by the kernel. This is where the shell comes in to play. When a user types of commands in the shell it performs an action that is calling on the kernel. Another way to interact with the kernel is by the desktop GUI (Graphical User Interface). By using the mouse or track pad to open, read, write, and close software applications it’s still making a call to the kernel. The programs that are being used within the user space are taking commands by using system libraries and translating them into system calls that the kernel runs. If we run a simple command uname -r this will display the name of the kernel. Mine reads 5.10.16.3-microsoft-standard-WSL2. First digit is the kernel version, then the major release, and the next two digits go into more detail about the release level. I’ve downloaded the WSL Linux command terminal (Windows Subsystem for Linux) to demonstrate some capabilities.

1. **Storage Management**: There are two partitions called MBR (Master Boot Record) which is limited to 2 TB and is being replaced by GPT (GUID Partition table) with 128 partitions and 9.4 ZataByte which is the 10th to the 21st power or A mount point is used to access data on a partition. At least one partition will be mounted on slash. ‘/’ is a notation for slash which is the topmost level directory. This allocates space for the file system, meaning that it acts as a door. The game application is relatively simple so there wouldn’t be many complications in terms of storage to run this in a disk management perspective. The best option to mitigate 8 mb each of images plus the game code would be safe to store it in a basic cloud solution or alternatively in an external HD (Hard Drive). If we were to store 125,000 images that would be 1 TB and a game application would be around 16 GB. So, there’s plenty of room on an external HD but to be safe the cloud would be a good solution in case we want to scale up. There’s plenty of DevOps tools in the Azure cloud environment to implement the Draw It or Lose It software game application. The cloud system would be customized to the game application to backup user data and ensure the profile data would be intact for backup. The game application can be accessed anytime by the user once the request has been made to the server and then to the cloud database.
2. **Memory Management**: We can start by showing the Linux memory information “cat.” The subsystem is ultimately responsible for managing virtual memory, demand paging, memory allocation and user space applications like Draw It or Lose It. The RAM will be most important since applications are stored temporarily and expelled as soon as they’re no longer needed. The 200 images are of high-resolution will need to be allocated and dispersed of in the RAM as soon as the game application closes. When deploying a game application, we can use Kubernetes that will allow our game application for continuous integration and deployment from the cloud. Kubernetes creates multiple instances where the user can access the application through a load balancer. The load balancer will distribute the application evenly, so the number of users does not fail. This would be the most viable solution to deploying our web-based game application over the network.
3. **Distributed Systems and Networks**: We’ll need to optimize Azure cloud platform to distribute the game application via web browser. We need to make the game application resources available, distribute transparency and scalability. To do this we need a solid transaction from the client > web server >Azure Cloud Platform > cloud database > Azure Cloud Platform > web server > client. This is a full cycle at a high-level operational process. We need web caches in the browsers and proxies. File caching at the server and client side. We can grid the computing systems by placing nodes throughout so we can span a wide-area network. The transaction takes place from the client side requesting authorization and then the transaction processing monitors the data being executed then the request to the server and so on. The next thing we need to do is create a virtual network and name it like frontend or backend game application. We should enable toe DDoS (Distributed Denial of Service) to mitigate attacks on our network service. This will allow a smooth operation without someone attacking our nodes in the network. The network infrastructure will need to be tested several times to make sure the distribution of the gaming application and network stability is sound from end-to-end.
4. **Security**: As stated previously the system architecture will need to be tested from end-to-end. We need to make sure we have backups in place, a redundant backup solution, multi-factor authentication password enabled, having software denied access in place and test. Built-in native web application firewall (WAF) is helpful to secure the network. As long as we enable the web application firewall feature, we can detect attackers by blocking them and reporting them. Making sure our access is limited by creating roles for privileges to the database and VMs (virtual machines). We need to encrypt the data at rest to ensure that our data isn’t compromised by an outside attacker. If the network is driven over ethernet, we need to make sure TLS (Transport Layer Security) by using HTTPS, upload a TLS certificate and make sure the application proxy is enabled. In Azure, there’s an Azure Key-Vault that we can store our game application secrets such as API keys and connection strings. Again, this is from a cloud based operation but this can be hybrid where the physical server can be secured in your own vicinity and the cloud services can be used for the data storage and protection of the game application.