

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.3

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3-4**](#_Toc115077324)

[**Evaluation 5-7**](#_Toc115077325)

[**Recommendations 8**](#_Toc115077326)**-11**

[**References 12**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/20/2024 | Daniel Gorelkin | Initial documentation |
| 1.1 | 05/21/2024 | Daniel Gorelkin | Documentation revised |
| 1.2 | 06/8/2024 | Daniel Gorelkin | Expanding the app to multiple platforms |
| 1.3 | 06/18/2024 | Daniel Gorelkin | Conclusions and Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

We are tasked with developing game-based software for the web platform, a domain currently exclusive to the Android operating system. This software should facilitate multiple user interactions, support diverse teams, and offer a range of game modes. It must be written in a programming language universally supported by all operating systems, and it should be responsive enough to adapt to various screen sizes. Our development approach must adhere to object-oriented programming principles. Each object instance will be constructed from multiple classes, each storing information separately but communicating with each other. Given that the operating system can only run one instance of the game, we will implement the Singleton pattern to meet this requirement.

## Requirements

*Business requirements:*

* *Run remotely and be accessible online from any location.*
* *Support all operating systems that have access to web services.*
* *Support multiple teams that can participate in specific game instances.*
* *Support multiple players that can play at the same game or team.*
* *The game, team, and player’s names must be unique.*
* *Only one variation of the app can be served by the OS.*

*Technical requirements:*

* *Support internet connectivity.*
* *Ability to create multiple instances of the game, team, and player class.*
* *Instance and input duplication validation to prevent repetitive naming.*
* *The singleton design pattern must be implemented to prevent multiple instantiations of the app.*
* *Written in a major programming language using best practices and OOP principles.*
* *No data should be reachable to other users or other program classes.*

## [Design Constraints](#_2et92p0)

Because it is a web-based application, the software will greatly depend on a fast and stable internet service provider; therefore, a fast and secure server host is needed.

Our application, being globally accessible and presenting graphical content, must adhere to cultural and political regulations. It will also consume significant CPU resources due to the requirement of delivering large (0.5-3MB) images over the air. Therefore, we need to consider minimal system requirements for running the application. The transition from an Android OS to a web-based compatible version will require a complete rewrite of the software, involving the use of different programming languages such as Java, JS, HTML, CSS, and more, as well as various interfaces to run the software on different OS. This process, along with the deployment, is estimated to take one month.

## [System Architecture View](#_ilbxbyevv6b6)

* To be revised after the initial design is approved.

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram provided below shows the app's structural model. This diagram is a visual representation of the app's classes, their attributes, and their relationships. The software will follow the OOP principles by communicating between four classes. The Singleton design will allow only one GameService class instantiation, guaranteeing that only one game instance can exist in memory. The ProgramDriver class will contain the main() method and will control the app's execution. The app will follow the inheritance principle by extending the Entity superclass into Player, Team, and Game classes. Doing so will make sure to encapsulate every instance of it in the appropriate private Lists to keep data safe and invisible to the other classes. In addition, we can reuse existing code by implementing polymorphism in the entity’s child classes.

**"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)

Various operation system comparison table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |

| **Server Side** | - Powerful hardware  - Operating is usually done through GUI.  - Expensive  - Outstanding communication with other Mac OS environments.  Deployment pros:  Easy-to-use, fast, reliable web hosting and an extensible architecture for deploying Java applications based on the high-performance Apache web server with seamless integration with other Apple products.  Provides integrated OpenSSL for implementing the Secure Sockets Layer architecture with 128-bit strong cryptography, regular updates, high performance, and great scalability.  Deployment cons:  Limited availability of server software compared to Linux with fewer customization options.  Deployment costs:  Higher cost of hardware compared to Windows platform. It might be significant when serving thousands of users at once.  Licensing fees will apply. | - Stable and efficient.  - Cost-friendly to operate.  - Open source  - Secure, Fast, and customizable.  - Operating is usually done through CLI  Deployment pros:  Widely used open-source with diverse distributions and strong community support for troubleshooting and development.  Highly customizable to fit specific or custom needs with excellent performance and security.  It can be easily deployed on the cloud to meet great scalability and performance needs.  Easy integration with most APIs and security protocols. Less vulnerable to attacks and breaches.  More stable and robust compared to Windows servers.  Deployment cons:  Requires significant technical expertise to set up and maintain.  Does not support several programming languages, such as ASP and NET.  Unable to use Windows applications  Deployment costs:  No licensing fees.  Hosting is more cost-effective than Windows- or MAC-based hosting.  More expensive and time-consuming when migrating from Windows-based servers. | - More expensive than open source.  - Better support if needed.  - Can run Linux as an upper Layer.  - Operating is usually done through GUI.  Deployment pros:  Wide compatibility with various software applications with easy integration with other Microsoft products and services.  Easy Azure cloud deployment and great scalability to handle sudden spikes or massive user traffic.  Easy to use and maintain due to widely extensive documentation.  Deployment cons:  Less secure than MAC and more vulnerable to malware and viruses. It can be resource-intensive, requiring more powerful hardware when serving thousands of clients.  Deployment costs:  Higher licensing costs compared to Linux.  Cheaper hardware maintenance and upgrades. Compared to MAC or Linux. | - Slower performance  - Higher portability.  - Lower flexibility  - Cost-effective  Deployment pros:  Portable and battery-powered, therefore, can provide temporary service during power outages.  Built-in sensors and mobility features can be leveraged.  Deployment cons:  Limited processing power and storage compared to server computers.  It is not suitable for hosting complex applications with multiple users serving at once. Battery power dependent.  Deployment costs:  Virtually free as long as you have a mobile device that is connected to the internet. |
| --- | --- | --- | --- | --- |
| **Client Side** | - More secure OS than Linux or Windows  - Better hardware performance  - Intuitive for inexperienced users.  - Better compatibility with mobile devices. | - More complex to operate for inexperienced users.  - Less support.  - More freedoms offered by the administration system. | - High licensing costs.  - Supports a large number of applications.  - More intuitive to operate for Beginner / intermediate users.  - More vulnerable to cyber criminals. | - Smaller screen resolution  - Limited physical memory  - Less functional for typing  - More mobile than any other OS  - Lower performance compared to bigger devices. |
| **Client Side for multi-platform** | Because The Gaming Rooms’ request is a web-based game that will be accessible through the machine’s browser, there is no significant difference in the client’s side development of the app between the various platforms.  Costs:  The use of IDEs, JavaScript library, React, Angular, or Vue frameworks is free.  Monthly expenses for using Cloud services such as AWS, Azure, or Google Cloud to host the back end will be around $200. However, this may depend on traffic and scalability. Development estimated expenses for developing the front-end side are expected to be around $60,000 for the 6-7 months project completion ETA.  Time:  Development ETA 5-7 months. Consists of one month for planning, five months for developing, and one more month for testing and bug fixes.  In addition, regular testing throughout the development process will be needed to ensure proper functionality across all platforms and reduced testing time later on.  Expertise:  - To run a server-based front-end game, skills in HTML, CSS3, and JavaScript will be required to ensure the UI/UX and graphical functionality.  - To run a server-based rear-end game, knowledge of cross-platform frameworks like React, Angular, or Vue.js will also be needed to address the technical side of the app’s functionality.  - To ensure proper operation on various screen sizes, the app must be responsive.  - Experience with web browsers such as Safari, Chrome, Firefox, Opera, and Edge.  - Knowledge of specific macOS security features. | | | |
| Experience with macOS security features. | Experience with Linux permissions | Experience with Windows registry. | Experience with iOS and Linux security features. |
| Development Tools | Preferable languages:  C, Java, PHP, Python, Ruby, Objective-C, Swift.  Preferable lDE:  Xcode, Preface, Apple's Terminal. | Preferable languages:  PHP, Perl, Python, Ruby, or MySQL.  Preferable lDE:  PyCharm, Visual Studio, PhpStorm, Atom. | Preferable languages:  VBScript, ASP.NET, C++, HTML, Angular, CSS.  Preferable lDE:  Microsoft Visual Studio, Azure, PyCharm. | Preferable languages:  C, Java, Python, React Native, Kotlin, Objective-C, HTML, JavaScript, and CSS.  Preferable lDE:  Android Studio, Xcode, Visual Studio. |
| **Development Tools for multi-platform** | Frameworks, Databases, and additional tools for web-based, multi-platform development:  For the back end, knowledge of the Node.js, Django, or Ruby on Rails frameworks ensures compatibility across the different required platforms.  For the front end, Knowledge of React, Angular, or Vue.js  Databases should be implemented in the game to store data information; therefore, experience with relational (MySQL, Postgres) and non-relational (MongoDB, Casandra) databases could be needed.  Despite the use of the various IDEs, they are free and have no direct costs as a function of the programming language that will be used, the same as the use of the different open-source libraries and frameworks.  However, Hosting, infrastructure, or licensing for the privately owned server could range from $50 to $500 a month, depending on the back-end platform, the type of deployment, and scalability. The same price estimation will be made for the server costs of hosting and serving the data.  To secure the communication, an annual SSL certificate purchase will be required ($100).  In addition, because of the complexity of the request, two separate teams for the front and the rear end to work on the project will be needed, as well as a project manager who will coordinate between the two. | | | |

## Recommendations

1. **Operating Platform**: Because the Gaming Room already has an Android OS running app, developing the software for the Mac OS server-side environment will provide great device coverage and will be easy. By using the appropriate interfaces, it will run efficiently on the iOS system, whereas Windows can run Mac applications on its upper layer.

However, running the game from the Mac OS environment might not be the optimal solution for the Gaming Room’s needs. As described in the evaluation table above, buying and maintaining iOS equipment will be more expensive compared to the other OSs (mobile deployment is not considerable enough to meet our needs.), whereas the end result will not justify the financial investment compared to the Linux and the traditional Windows server options. Comparing the last two, the Windows OS will be the easiest to operate on the server side, although it is compromised to the Linux OS in terms of security and stability (more about it later). The Linux OS, on the other hand, will require some technical expertise to set up and get used to it, though there are no licensing fees compared to Windows due to its being an open source, and hosting is more cost-effective than Windows or MAC-based. In addition, It can be easily deployed on the cloud to meet great scalability and performance needs as our game will expand to thousands of users and will require integration with various APIs and security protocols. With this in mind, server-side hosting the game on the Linux OS will be the best solution for your needs.

1. **Linux Operating Systems Architectures:**

The architecture of the Linux OS might seem overwhelming at first glance for an unfamiliar user due to its command line interface (CLI), although it is a highly organized structure that balances performance, security, and flexibility. Its modularity and similarity to Unix principles make it suitable for a wide range of applications, particularly for web applications such as Draw It or Lose It for multiple platforms. The game will run as a web application and communicate with the Kernel layer through a series of APIs and system components.

A screenshot of a computer program

Description automatically generated

What makes Linux such a robust and versatile OS is the range of concepts, components, and design principles it offers. It’s monolithic kernel core component responsible for managing hardware resources and providing essential services to applications, yet supporting loadable modules, allowing drivers and features to be dynamically loaded and unloaded as needed without rebooting. The versatility achieved through system calls, and unique space management by segregating the memory area where the kernel executes and provides its services between the system and the user tasks where user applications run. These system calls provide the interface between user applications and the kernel. In addition, Linux employs sophisticated memory management techniques such as virtual and shared Memory, paging, and swapping to save space on disk and make the applications run faster.

1. **Storage Management**:

Having in mind that the game will be web-based and rely heavily on delivering and communicating with the user in the form of graphical files, storage management is a critical aspect of healthy app performance. Therefore, to allow future expansion and scalability, storing the app’s media on a cloud database such as Oracle will complement the app’s performance and will be perfectly integrated with the Linux OS. The data will be fetched by the server and delivered to the user’s machine’s front-end app. Considering the heavy dependency of the app on internet access, the data will be transferred in batches of 10-15 images and stored on the machine's local memory to be easily and fast accessible to the memory management system. As the turn of the specific image to be displayed arrives, the data will be mounted on the main memory and displayed to the user by the graphical layer or the direct-memory-access (DMA) controller if the main memory is overloaded. Having the data stored in a cloud database ensures its security, easy accessibility, and delivery to the user.

1. **Memory Management**:

Linux employs sophisticated memory management techniques such as shared memory, which allows multiple processes to access the same physical memory or mount information directly from the storage memory through virtual memory. Because APFS allocates disk space within a partition, all application files should be stored in one container. In contrast, the media files should be stored in a separate sub-container to provide fast performance versus constant online data streaming that could save device space but reduce software performance and decrease user experience. As the Draw it or Lose it game will be instantiated from the user end, the OS will grab all the required code and the graphical files from the SSD and mount them on the processor’s heap memory for super-fast data processing. Once the operation is completed, it will be removed from the heap memory and stored back in its container on the SSD.

A diagram of a diagram

Description automatically generated

All that operation is managed by sophisticated process creation and management mechanisms such as process scheduling, which is based on priority scheduling algorithms and controlled via signals, inter-process communication mechanisms, and process groups. To handle this scenario, the memory manager will create a massive MAP table with the addresses of all the image's locations on the HDD (secondary storage). In addition to the regular app’s execution interrupts to handle asynchronous events, the controller will be constantly busy running and transferring the data that spread in fragments all over the memory to the main memory to address the app’s execution.

1. **Distributed Systems and Networks**:

Communication between distributed systems and networks involves several key components and techniques to ensure reliable, efficient, and secure data exchange. To achieve this, the app must ensure compatibility with the web browsers and devices the clients use. The Draw It Or Lose It will use the commonly used HTTP/HTTPS approach for web-based communication protocol to send requests to the server, which processes and responds to these requests by sending back hyperlinks containing the addresses of other linked resources, data, and images. This approach will use the client-server model. For example, the client requests services from the server via web applications and RESTful APIs by sending a GET request through the OS browser and asking for data such as the following image and text file from the server. The server, which might run a different OS, receives the request, computes the results, and sends a unified structured response, such as JSON, to the client. This pattern allows horizontal scalability as the workload grows while providing access to multiple users (clients) with different operating systems. On the server side, the versatility is provided by the Representational State Transfer (REST) pattern of data, an architectural style designed for building distributed hypermedia systems. In this interoperable communication pattern, the server does not differentiate between the OS that sent the request because it was received and replied to in a standardized, uniform Interface format that any OS can interpret.

A diagram of a computer network

Description automatically generated

1. **Security**:

As previously mentioned, Linux provides a robust yet flexible environment for running secure multilayered web-based applications, including patching and automatically installing security updates. On top of using the firewall settings, Linux enforces strict user and group permissions. Role-based access control (RBAC) is used at both the operating system and application level to ensure users have the minimum necessary access. For example, each file or directory has its own permissions settings defining who can read, write, or execute files. Not less important, it runs web applications with the least user privileges necessary to limit the impact of a security breach. To grant access to its users and applications, it uses authentication and authorization protocols such as key-based SSH authentication and multi-factor authentication (MFA). Draw It or Lose It will use TLS/SSL OpenSSL encryption to implement the secure sockets layer architecture with 128-bit strong cryptography to enhance secure communication between the client and server. In addition, Creative Technology Solutions (CTS) will ensure that Draw It Or Lose It will meet the best secure coding practices by educating developers to prevent common vulnerabilities.

**References:**

Introduction to Memory Management. (n.d.). https://developer.apple.com/library/archive. Retrieved May 21, 2024, from <https://developer.apple.com/library/archive/documentation/mac/pdf/Memory/Intro_to_Mem_Mgmt.pdf>

File system formats available in Disk Utility on Mac. (n.d.). Apple Support. <https://support.apple.com/en-ca/guide/disk-utility/dsku19ed921c/mac#:~:text=Apple%20File%20System%20(APFS)>

Lavieri, D. E. (27-04-2019). Hands-On Design Patterns with Java. [[VitalSource Bookshelf version]]. Retrieved from vbk://9781789809954