

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/22/2022 | Steve Blevins | Updated Executive Summary, Design Constraints, and Domain Model. |
| 1.1 | 06/05/2022 | Steve Blevins | Completed evaluation for multiple platforms considering client, and server side, and available development tools. |
| 1.2 | 06/19/2022 | Steve Blevins | Completed recommendations addressing operating platform and architecture, memory and storage management, distributed systems and networks, and Security. |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room, has a mobile game on the market named Draw It or Lose It. Currently this game is only available for the Android platform which drastically limits the user-base, and exposure of the game. To increase market exposure and attract more users for other platforms such as IOS, MAC OS, and Windows users, we will develop a web-based environment for the game. The web-based environment will support multiple platforms with only one development project which will be much more efficient and cost-effective both short, and long term as opposed to multiple applications for various platforms.

## [Design Constraints](#_2et92p0)

**Overall**

* Support for a distributed environment including a host, and multiple sites to run the service and host the environment to support a growing number of users, and reduce latency.
* A web development team with knowledge of the Android system. Required for code reuse from the existing Android app in development of the new, web-based game.

**Software**

* The game may have exactly one instance in memory at any given time (Singleton).
* A game must allow for one or more teams to be playing.
* Teams must allow for multiple players
* Game names, and team names must be unique, and must be checked on creation to avoid duplicates.

## [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)

Below is as class diagram illustrating the required classes for the game, and their relationships. The ProgramDriver class contains the public main function and uses the SingletonTester class at times to confirm whether a class is a singleton or not. The GameService class is associated with the Game class having a zero-to-many multiplicative association. The Game class has the same relationship with the Team class, and the Team class has the same with the Player class. The Game, Team, and Player classes all inherit from the Entity class.

Four OOP principals are evident; Encapsulation, Abstraction, Polymorphism, and Inheritance.Polymorphism is shown by the overriding constructors, and methods of the Entity base-class in the Game, Team, and Player classes. The three of which inherit from the Entity class which clearly shows inheritance. Encapsulation is indicated by private access modifiers on all class attributes, as well as the GameService, and Entity constructors, and Although Entity is not an abstract class, it is somewhat abstract due to its private constructor.

**"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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Apple operating systems have deployment methods and web-hosting built-in with robust management software, as well as it's versatility in supporting a multi-platform network.  Licensing options are either limited based on the number of clients, or an unlimited client option with a range from $500 - $1000. In addition, a package add-on will cover 3 years of licenses for an additional upfront cost which saves on long term cost of licensing. | With Linux being open source, and having so many distributions, plenty of options are available with additional software for deployment, and server management.  Linux servers have an encompassing array of licensing options based on requirements of the server, and level of support. 1 year with self-support has a cost of $350, or standard, and premium support options for $800, and $1200 respectively. There are also licensing add-ons such as High availability, Smart Management, etc. | Windows Deployment Services allows network-based deployment with a snap-in tool, command line, and through Windows PowerShell. Windows server licensing options are comparable in price / client support to apple. A small number of clients (50) has a cost of $501, however, other license options are core based which may be more appealing due to the change in cost based on growth / use. The cost in these options is $1069 for 16 core licenses. | Although possible for smaller, simpler websites, server-side / hosting from a mobile device would be undesirable in this application. Although cost of maintenance and management would be cheap, only a small number of clients would be supported and would ultimately lead to slower speeds and crashes. Because most of the mobile platform server applications are open source, licensing would not be an issue. |
| **Client Side** | Mac has a lot of high performance, and high-quality hardware along with robust systems, and high-level security. However, this comes with a lot of restrictions, and Mac tends to put more focus on compatibility between other Apple products, and less focus on outside compatibility which could be a setback. Apple hardware and software tends to be higher in price compared to others. | Linux will be extremely versatile in development, and cost will be kept low due to open-source, and the number of distributions. Linux supports all of the major languages and has a lot of similarities to the Android platform that will allow development teams to overlap. | Although Windows, and Mac are equally as popular, Windows has a much lower cost and may be customized easier to support the needs of the development team. There are also many tools available for windows that may not be for other operating systems. | Mobile development may be feasible with tablets, and many IDEs have mobile application versions. Scaling the development may be difficult however due to hardware limitations on mobile devices, as well as wireless dependencies. Larger programs may be difficult to develop on mobile platforms without additional peripherals such as a keyboard and mouse, however and computer will be able to support the same peripherals, as well as have better performing hardware. |
| **Development Tools** | Mac OS supports all of the most common programming languages including C, C++, Java, and Python. Additional Languages supported by Mac are Objective-C, and Ruby. Supported IDEs are Xcode, Eclipse, Visual Studio Code, CLion, and Appcode. | Linux may be useful with Android mobile app development as well since the platforms are so similar. IntelliJ is the most popular IDE for Linux; however, others are available such as Bluefish, and NetBeans by Apache. Relevant languages are Java, C, C++, and Python. | Some of the most popular Windows programming languages include C, C++, Java, and Python. For web development, relevant languages are HTML, CSS, and PHP. Numerous IDEs are available for various languages including Eclipse for Java, and Visual Studio for C++ (also supports other languages). | Android-specific applications may be written in Java, or Kotlin, while iOS applications may be written in Objective-c, and swift. Special requirements are necessary for iOS development including an Apple development account, and a Mac computer running Xcode IDE. Android is open source and minimal restrictions are in place for app 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**: A cloud-based Linux server is recommended as an appropriate operating platform. Using a cloud based, leased Linux server will add a cost for leasing, however this cost will be offset by using significantly less resources to maintain and operate the server. This also eliminates a hardware purchase cost, and alleviates the concern of hardware obsoletion.
2. **Operating Systems Architectures**: Being open source, Linux is extremely versatile and contains support for TCP/IP communication, and GUI display which are both critical for game operation, and communication to clients. Further, Linux is desirable for is multi-user / multi-programming abilities. Due to the requirement of numerous games running at once with a large volume of users, the multi-user / multi-programming quality is a necessity. Hardware and peripherals communicate through the kernel. The shell and the hardware work in conjunction to connect the user with the operating system, and applications / utilities provide pertinent functionality and management abilities.
3. **Storage Management**: Linux supports a direct access method as identified during evaluation. Direct access will allow faster and easier data retrieval and modification to reduce latency when transferring image data, user data, and game data. In addition, separate partitions, including a dedicated database for the images, will more efficiently organize and separate the aforementioned data.
4. **Memory Management**: Virtual memory is used in Linux to map addresses to physical memory. This helps ensure that only data which is necessary be sent to physical memory through a request and reservation. As the game is run and the image files are read, they will be loaded into the cache for quick, and efficient access without the constant need to access the disk which is resource-demanding. As games end, image files will be deleted, memory is freed, and the next image will be loaded.
5. **Distributed Systems and Networks**: Using distributed software in a client-server type architecture will facilitate communication between the server and various other platforms through HTTP / TCP IP. A distributed system will further reduce the load on the server, and distribute this to the clients through use of storage, and memory for game data. Requests will be made to the server for necessary data as required. It is recommended to have multiple servers for the game to provide some redundancy in case of outages, and to distribute the demand on the network as well.
6. **Security**: By using a cloud server as recommended, maintenance such as updates and physical security will be managed by the cloud host and will not be a concern although worth noting is the importance of keeping the server up to date, and maintaining security software such as anti-virus, and a firewall with only essential ports open. Permissions for the applications and data on the server should be the minimum required, and data transfer to and from clients must be encrypted.