

Draw It or Lose It

# **Software Design Document**

Version 1.2

## Table of Contents

[**Software Design Document** 1](#_Toc115347244)

[Table of Contents 2](#_Toc115347245)

[Document Revision History 2](#_Toc115347246)

[Executive Summary 3](#_Toc115347247)

[Design Constraints 3](#_Toc115347248)

[System Architecture View 3](#_Toc115347249)

[Domain Model 4](#_Toc115347250)

[Evaluation 5](#_Toc115347251)

[Recommendations 6](#_Toc115347252)

## [Document Revision History](#_grjogdjh5fi8)

| **Version** | | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- | --- |
| 1.0 | 9/17/2022 | | Jasmine Zeng | Updated cover page; Document Revision History, Executive Summary, Design Constraints, Domain Model, Evaluation, and Recommendations sections; and document footers. |
| 1.1 | 9/29/2022 | | Jasmine Zeng | Updated cover page, Document Revision History and Evaluation sections, and document footers. |
| 1.2 | 10/13/2022 | | Jasmine Zeng | Updated Recommendations section and document footers. |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions (CTS) have taken on a new client, The Gaming Room. The Gaming Room has an Android-based game, *Draw It or Lose It* (“game”). This game is identical to the 1980s television game, *Win, Lose or Draw*, where one or more teams compete against each other to guess what is being drawn. The Gaming Room seeks to upgrade this game into a web-based version that will serve several other platforms such as MacOS, Linux, Windows, and non-Android mobile devices. The program will also be programmed to the following specifications: only one instance of the game will exist at any time, each instance can involve one or more teams, each team can have multiple players, and game and team names will be validated for uniqueness. The Gaming Room staff do not know how to set up the environment, so CTS have prepared this software design document to describe their plans and recommendations in the game’s development phase.

## [Design Constraints](#_2et92p0)

1. **The game must be optimized for each platform.**
   1. MacOS, Linux, Windows, non-Android mobile devices, and Android mobile devices utilize varying software development kits (SDKs). The code in the Android version should be translated to these SDKs to retain functionality in all versions of the game. The Gaming Room has not specified if they would like cross-platform player features either. If so, additional programming is required so players from all platforms can play with each other.
2. **The game must be able to run multiple layers of verification.**
   1. Each game instance and team name should be assigned a unique identifier to ensure that only one instance of the game is being run at any given time. Input validation for game and team names should also be run against each other to avoid affecting the game’s functionality, such as score count.
3. **The game must have adequate memory.**
   1. The game must be capable of hosting all teams and members for each instance.
4. **The game must have a cohesive user interface.**
   1. The appearance, layout, and functionality of the game in non-Android versions should be identical to their pre-existing Android application. Alternatively, the game can be re-designed for all platforms.

## [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)

The Unified Modeling Language (UML) diagram below provides a visual overview of the technical layout and fulfilled requirements of *Draw It or Lose It*.

The Entity class is the parent class to subclasses Game, Team, and Player. Entity encapsulates these subclasses into details which belong to a single game. Conversely, each subclass inherits the features of Entity class. Each subclass also extends Entity to provide specific information about the Game, Team(s), and Player(s). Each Game can have one or more Teams. Each Team can have one or more Players. This fulfills The Gaming Room’s requirements for each instance of the game.

The GameService class holds all vital methods for game functionality, such as getting the game, team, and player name(s).

The ProgramDriver class runs the game and uses the SingletonTester class to process information in each instance of the *Draw It or Lose It* game.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS provides full support for Mac applications. However, it is not ideal if services are outsourced to third-party providers. Licensing is also expensive. Additionally, Mac devices are required to develop and maintain the server, which are costly and become outdated more quickly than the others. | Linux is a popular choice among these operating platforms in application development. Linux is open-source, free to license, compatible with all web browsers, and highly customizable. However, this operating platform is harder to learn compared to the others. | Windows supports many applications, third-party softwares, and programming languages. It is also secure and easier to set up among the other competitors on this list. However, licensing can be expensive. | Mobile devices are not ideal for server use, although feasible through open-source software. Using this platform will come with difficulties for testing environments, browsers, and applications. They also do not have enough power to host. |
| **Client Side** | MacOS can run several web browsers and development tools. It is also user-friendly. However, Mac devices are required to develop and test applications on MacOS, and this becomes expensive and quickly outdated. | Linux can run virtually all web browsers and development tools due to its free, open-source software. It is also easier to develop and deploy applications on this platform than the others. However, these benefits are balanced out by its steep learning curve. | Windows has the same competencies as Linux when running browsers and development tools, as well as the development and deployment of applications. It is easier to learn on Windows than Linux. However, it is not open-source and requires licensing fees. | Mobile devices are application-optimized. Consequently, experienced mobile developers are needed for this platform. It will require more specialized developers to manage programs here due to differences between the mobile interface and the other three platforms. |
| **Development Tools** | Mac uses Swift, the primary language for Mac’s computer and mobile device applications. It is easier to write and faster to run apps here with Swift. Xcode is the primary IDE for Mac development and is free on the Mac App Store, but also has a developer tier for $99/year which allows app upload privileges to the app store. | Python usually comes pre-installed with Linux. Atom and Visual Studio Code are two of the most ideal IDEs for Linux since it supports and runs most languages. Both IDEs are free, although Visual Studio offers professional developer tiers ranging from $500 to $1,200/year. | Most Windows applications are written in C or C++. Visual Studio Code is Window’s standard IDE since it supports these languages, and many more. See Linux’s *Development Tools* section for Visual Studio Code pricing. | Most mobile Android programs are written in C++. Most mobile Apple programs are written in Swift. AndroidStudio is best for developing Android mobile apps and Xcode is best for developing Mac mobile apps. Android Studio has a one-time fee of $25 which the client might have already paid. See Mac’s *Development Tools* section for Xcode pricing. |

## Recommendations

1. **Operating Platform**: The Linux operating system is best for all aspects of this project. Its free licensing and open-source features will reduce project costs. It also supports various languages, IDEs, and other tools needed for development and testing. It is also secure.
2. **Operating Systems Architectures**: An operating system works as an interface between computer resources and a user. The architecture of the operating system is its overall hardware and software design, and how these two operate together. The Linux architecture is composed of the Kernel, System Library, Hardware Layer, System, and Shell. The Kernel provides communication between the software and the hardware. The system library is a set of special functions which help the operating system run properly. The hardware layer contains the physical components of the computer. The system executes specific and individual tasks. The shell acts as the user interface allowing communication between the kernel and the user.
3. **Storage Management**: Because the game will be available on several platforms, and each game instance will have varying amounts of data to store and access, it would be best to store and access data on a cloud storage service. Amazon Web Service’s (AWS) Snow Family provides various storage services based on access frequency and will only bill The Gaming Room for storage used.
4. **Memory Management**: Linux is a highly customizable operating platform that provides internal memory management through demand paging and virtual memory. In demand paging, information is brought into the main memory only when it is needed. In virtual memory, programs can make a special memory reservation that no other programs can access. This allows instances of the *Draw It or Lose It* game to be securely accessed, edited, stored, and saved. AWS also offers memory options such as instance stores, which provides temporary block-level storage that has the same lifespan as the instance.
5. **Distributed Systems and Networks**: The game can run seamlessly by developing on Linux and deploying through AWS cloud computing services. As previously mentioned, Linux is a robust platform for development and testing. Once the game is launched, AWS can provide The Gaming Room with a fully cloud-based application deployment. AWS can store the code and initialize it only when requested by players. AWS then only bills the company for these requests. This pay-as-you-go model cuts down on physical server operation costs for running the game and is easily scalable. AWS is also excellent at load balancing all incoming traffic from all platforms and has plenty of data centers (termed “availability zones” by AWS) and storage sites (termed “edge locations” by AWS) throughout the globe which minimizes risks of server outages or connection issues.
6. **Security**: Linux is rarely affected by malware. This makes the game development and testing process secure. When the game is deployed through AWS, player information can be protected through the service’s encryption and key management features. AWS also provides continuous and thorough threat detections and monitors the accounts and workloads of the game software itself.