

Draw It or Lose It-The Game Room

# **CS 230 Project Software Design**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design** 1](#_Toc128304461)

[Table of Contents 2](#_Toc128304462)

[Document Revision History 2](#_Toc128304463)

[Executive Summary 3](#_Toc128304464)

[Requirements 3](#_Toc128304465)

[Design Constraints 3](#_Toc128304466)

[Domain Model 4](#_Toc128304467)

[Evaluation 5](#_Toc128304468)

[Recommendations 8](#_Toc128304469)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/21/2023 | Michael Almon | Added Executive Summary, Design Requirements, Design Restraints, and a description of the Domain Model. |
| 2.0 | 02/04/2023 | Michael Almon | Added Evaluation Summary |

## [Executive Summary](#_sbfa50wo7nsh)

Our new client The Gaming Room, wants to develop a web-based game that can be available on multiple platforms. They want to base it on their current game, Draw It or Lose It, which is itself loosely based on the 1980s game show *Win, Lose, or* Draw. While Draw It or Lose It is currently only available as an Android app, the company also wants the new game to be able to run on multiple platforms, which can include Windows, MacOS, Linux, ChromeOS, and mobile devices such as iOS, Android, and Windows. The game consists of teams competing to guess what images are being drawn as the application renders them. The drawings will be rendered at a steady rate and will be fully rendered after 30 seconds if not guessed correctly. The opposing teams will then have 15 seconds to submit one guess each for the rendered image.

## Requirements

For technical requirements, The Gaming Room wants the game to be able to support one or multiple teams to compete against each other. The game must also be able to render an image at steady rate and fully render the image in 30 seconds. The game must be able to take multiple guesses during this time and determine if one is correct. If one is not correct, the game must give the opposing teams 15 seconds to submit one guess each and determine if one is correct. If multiple guesses are correct, the game must be able to determine which one was submitted first. As a business requirement, The Gaming Room will need to obtain licenses and copyrights for images when appropriate. As both technical and business requirements, The Gaming Room wants the game to be web-based and to be able to run on multiple platforms.

## [Design Constraints](#_2et92p0)

* The game needs to be web-based. Subsequently, this means a hosting service and database server will need to be utilized.
* The game needs to be multi-platformed. The Gaming Room Will need to decide which platforms to include, such as Windows, Linux, MacOS, ChromeOS, Android, and iOS. Other platforms could include gaming consoles such as Xbox and Playstation. Each platform will require different criteria for being included on it.
* Knowledge of multiple programming languages will be needed. Front-end development (the user interface) might require knowledge of HTML, CSS, and JavaScript. Back-end development might require knowledge of Python, Ruby, Java, and SQL.
* The game design needs to be able to handle at least one team, and each team will have multiple players.
  + Each player will need to have a unique identity.
  + Each team playing will need to have a unique identity.
  + Users will need to be able to check whether a user name has been taken.
  + Teams will need to be able to check whether a team name has been taken.
* Only one instance of a game can be played at a time.
* The game rules are as follows:
  + There are four rounds lasting one minute each.
  + The drawings are rendered by the application at a steady rate, to be completed after 30 seconds
  + Players for the current team have 30 seconds to guess what is being rendered.
  + If the current team cannot guess the image, the opposing teams each have 15 seconds to submit one answer.
* The game will need to be able to handle multiple inputs from different sources and determine the order of submission.
* With an unknown number of players at any time, memory management must be considered.
* Licenses and copyrights of existing images must be considered.

## [Domain Model](#_8h2ehzxfam4o)

The ProgramDriver class contains the main() function, so this is the main driver of our game. Everything will start here, including obtaining the memory reference to the game to be played. It uses the SingletonTester class to ensure that only one instance of a GameService() exists at a time.

The GameService class is the main driver of our game control. The private GameService() constructor and the public getInstance(): GameService enable the creation the unique GameService(). This also prevents any additional GameService() objects from being created, allowing control over access and memory management. It stores the current Game()s being played in a list, along with functionality to add Game()s, retrieve Game()s by name and id, and get a count of total current Game()s. It also allows the application to acquire the next Team() id and Player() id.

The GameService() class can hold none to many Game() objects at a time. The Game() class keeps a list of Team() objects for that game, allows for the creation of a Game(), and the ability to add Team()s to the list. It can also display information about each game through the override toString() method.

The Game() class can hold none to many Team() objects at a time. The Team() class keeps a list of Players() object for that game, allows for the creation of a Team(), and the ability to add Players()s to the current list. It can also display information about each team through the override toString() method.

The Team() class can hold none to many Player() objects at a time. The Player() class allows for the creation of a Player(). It can also display information about each player through the override toString() method.

The Game(), Team(), and Player() classes are all child classes of the Entity() class, and will inherit its available public methods. Each of the child classes can use the getId() and getName() methods from the Entity() class to access the corresponding id and name. Game().getId() will retrieve the id of the desired Game() instance, but not the Team() or Player() id. It is similar for Game().getName().

**"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** | An expensive platform due to reduced demand and licensing costs. Software packages have comparable open-source Linux counterparts **(Cudmore, 2020)**. Intuitive graphical user interface (GUI) like Windows. Implementation technologies are more diverse, imposing fewer technical constraints than Windows. Expensive as the OS carries more features than needed to host the application and licensing costs increase with additional servers. Popular for web app hosting. | Open source with numerous libraries and utilities pre-packaged **(Horne, 2019)**. This also allows for more freedom and customization in design. Command line operation allows for lower resource consumption and performance rates. Least privilege access enforced out of the box **(Taylor, 2018)**. No additional licensing fees as the OS is free to use. Most secure OS regarding malware and cyber threats. Can have steep learning curve, so a knowledgeable administrator is necessary. Some versions may not have long term support. Very popular for web app hosting. Linux will require more managing. | Expensive platform with both OS and product licensing costs **(Horne, 2019)**. This licensing does guarantee long term support, however. Targets Microsoft web framework technologies, imposing technical constraints **(Heng, 2020)**. Intuitive GUI for administrative functionsand out-of-the-box functionality. Less monitoring is needed and is therefore very user and beginner friendly. A GUI is much more resource intensive and is mandatory for Windows. Windows is more prone to malware and cyber threats. Popular for web app hosting. | Frameworks, such as I-Jetty promote scalability and performance for web hosting **(Mobile Web Server, 2020)**. Requires connectivity to an internet-enabled computer to support DNS conversion, adding extra relay steps **(Mobile Web Server, 2020)**. Many deployment tools (i.e., Jenkins, Maven, etc.) support all major OS. Deployment solutions can be leveraged that are independent of the OS **(Top Software…, 2017)**. |
| **Client Side** | Regardless of OS, development should ensure optimized, modular processes that handle numerous requests and return quick, lightweight responses to the client (understanding…, n.d.). REST API structure supports this need. Support modern browsers, secure authentication, and transmission protocols that prevent session hijacking. Mac OS Supports a minimal number of file structures that can limit media types (see Linux discussion) | Supports a minimal number of file structures. This limits media type options that can be delivered to the client without additional programming overhead, as out-of-the-box file type support is limited and needs additional programming to handle additional types. | Optimized to run with Microsoft proprietary web browsers, imposing a potential technical constraint. However, all major web browsers are supported, and the Microsoft web browsers support the REST framework. | Insufficient processing capability and no plugin architecture to support heavyweight or executable media types (e.g., Java applets), all logic must live server-side (Bartlett, n.d.). Better suited for lightweight REST API with concise message exchange (e.g., JSON) **(Monus, 2020)**. Requires expertise in developing for optimal user experience (UX) across varying form factors **(Zmora, 2015)**. |
| **Development Tools** | macOS is built on top of UNIX, from which Linux is derived. This gives much of the same command line functionality as Linux, with the GUI accessibility of Windows. This gives macOS accessibility to tools like Adobe, which Linux doesn’t have. Open-source frameworks and IDEs, like Maven and Eclipse, eliminate licensing costs and provide robust, OS agnostic, deployment solutions. Visual Studio for Mac is also very powerful, especially for .NET developers. | As Linux has PHP and MySQL support, installing WordPress is easier on a Linux server. Also, it offers easier access to HTTP, Apache, and other site-creation tools, JavaScript and NodeJS environments, and Perl and Python programming languages. HTML, CSS, and JavaScript will also be need for client-side development. Eclipse is a great IDE for Linux and includes language support for most needed language with a powerful open source and free version. | Windows is optimized for its proprietary .NET framework or its open source ASP.NET leveraging C# for OOP. HTML, CSS, and JavaScript will also be need for client-side development. Visual Studio would be a good IDE choice for Windows. Visual Studio support for more than 36 programming languages, including C#, F#, Visual Basic .NET, C++, HTML, CSS, and JavaScript. Visual Studio is a must if developing in a Windows environment. | iOS and Android favor different programming languages. Android favors Java, which is not natively supported on iOS. Frameworks or platform-specific languages may be needed to cross platforms. Expertise in particular frameworks or polyglot programming practices are needed to support cross platform development. |

## [Recommendations](#_2o15spng8stw)

1. **Operating Platform**: Based on my evaluations, I would recommend the Linux operating system.

Linux provides excellent stability and cost benefits, as well as excellent security. The biggest drawback, I believe, is the necessity of a devoted and knowledgeable Linux developer.

1. **Operating Systems Architectures**: Linux offers many architecture aspects that make it appealing. As mentioned above, it is very stable. This is due to the modular design philosophy and its open-source nature. This has allowed thousands of programmers to thoroughly debug and optimize it. Its open-source nature also provides amazing cost savings due to no licenses needing to be bought. Linux is also very flexible and scalable.
2. **Storage Management**: Given that Draw it or Lose be a web-based application, I would

recommend a Cloud storage. With cloud storage, we can utilize only the storage space needed to

keep the game running. As the game increase in popularity or we add images, we can upgrade the storage without having to spend money on additional hardware. This allows us to pay only for what we need. It also allows us to minimize any physical and mechanical overhead and maintenance. A content delivery network would also help bring images closer to the user, allowing for more prompt responses. It would also add another layer of scalability and remote support.

1. **Memory Management**: Memory management on the development side will involve extensive testing prior to release. Any object that is allocated space will need to be deallocated once the object is not longer needed. This can include games, teams, users, or images. Memory leaks can result from objects that are not properly deallocated, causing sluggish performance and eventually crashing. Programming with Java helps alleviate some of this, with Java’s garbage process that automatically deletes any object that is no longer needed. Proper programming techniques should be followed no matter the language being used.
2. **Distributed Systems and Networks**: Needing Draw It or Lose It to be available on various platforms, we will need to use application programming interface (API) that is widely available on multiple platforms. This will allow communication between the user and application. RESTful or GraphQL would be two good choices. Each provide commands that allow for universally formatted data that is required for users on multiple platforms.
3. **Security**: Security is the most important aspect of this application. With communication constantly going between the application and users, there is a need to safeguard both ends. The APIs discussed above provide a layer of security. They should be implemented to require some kind of authentication to prevent either endpoint from being publicly available. Other layers of authentication and authorization should be added on top of that as well. User levels, strong password requirements, activity monitoring with alerts, and multi-factored authentication are basic layers that should be adopted.

**References**

Cudmore, Dale. (2020, March 2). *Mac Hosting: It Was Hard, But We Found 5 Top Hosts For Mac Web*

*Servers*. Retrieved from <https://www.whoishostingthis.com/compare/mac/>.

Google. (n.d.). *Cloud CDN: Content delivery network  |  google cloud*. Google. Retrieved February 26, 2023, from https://cloud.google.com/cdn#section-8

Heng, Christopher. (2020, February 12). *Should You Choose a Linux or a Windows Web Hosting*

*Package? And Is There Such a Thing as a Mac Web Host?* Retrieved from <https://www.thesitewizard.com/webhosting/linux-vs-windows.shtml>.

Horne, Katie. (2019, November 12). *Operating Systems for Web Hosting: Linux vs Windows*. Retrieved

from https://www.whoishostingthis.com/compare/operating-systems/.

Java Garbage Collection Basics. (n.d.). Retrieved February 26, 2023, from https://www.oracle.com/webfolder/technetwork/Tutorials/obe/java/gc01/index.html

Jensen, Z. (2019, May 11). *Graphql - an alternative to rest: End point dev*. GraphQL - An Alternative to REST | End Point Dev. Retrieved February 26, 2023, from https://www.endpointdev.com/blog/2019/05/graphql-an-alternative-to-rest/

Linux architecture - detailed explanation. InterviewBit. (2022, June 10). Retrieved February 26, 2023, from https://www.interviewbit.com/blog/linux-architecture/

Logvinenko, A. (2023, February 22). *How to secure web applications from vulnerabilities in 2023*. MobiDev. Retrieved February 26, 2023, from https://mobidev.biz/blog/best-practices-to-secure-web-applications-from-vulnerabilities

Mobile Web Server. (n.d.). In Wikipedia. Retrieved March 29, 2020, from

http://en.wikipedia.org/wiki/Mobile\_Web\_Server.

Monus, Anna. (2020, February 13). *SOAP vs REST vs JSON - a 2020 comparison*. Retrieved from

https://raygun.com/blog/soap-vs-rest-vs-json/.

Zmora, Amir. (2015, May 5). *5 Mobile Form Factors to Consider When Building Your App*. Retrieved

from https://thenewdialtone.com/mobile-form-factors/.