

Draw It or Lose

# **CS 230 Project Software Design Template**

Version 1.2

## 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**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/25/2024 | Anthony Bolosan | Initial draft of the software design template |
| 1.1 | 06/09/2024 | Anthony Bolosan | Information for evaluation section added |
| 1.2 | 06/23/2024 | Anthony Bolosan | Information for Recommendations added |

**Executive Summary**

Our new customer, The Gaming Room, seeks to create a web-based game that is accessible across numerous devices. The current game they intend to base on is Draw It or Lose It. The game is now only available on Android, but the business hopes to expand its compatibility to other operating systems including MacOS, Linux, and Windows. Teams compete to predict the images being drawn. There are time constraints for guessing and for displaying the drawing mark time. The game has four rounds, each lasting one minute.

## Requirements

The customer seeks a scalable web-based game application that works on multiple platforms, including Windows, Mac, Linux, and mobile devices. The application should have multiplayer features, real-time updates, and secure user authentication. Furthermore, it must efficiently manage concurrent user sessions and assure data consistency across distant contexts.

## [Design Constraints](#_2et92p0)

The fundamental design limitations are to provide cross-platform interoperability, manage network latency, and preserve data consistency among distributed servers. These limits have an influence on development by requiring the adoption of platform-agnostic technologies like HTML5, JavaScript, and cloud-based databases. Furthermore, the architecture must include robust error handling and synchronization capabilities to solve potential connectivity issues and maintain a consistent user experience.

## [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 gaming system's visual design will be created using the UML diagram (Unified Modeling Language). Let us look at how the software will be built. The UML diagram shows how the entity class connects the game, team, and player classes. The arrow shows that each class will inherit an attribute from a super class. We may view the classes, variables, and methods that will be utilized throughout development. The programDriver Class in the top left corner points to the singletonTester. This indicates that the programDriver will use the SingletonTester to test the code. The purpose of this development is to test the game's ability to exist in memory just once. The Gameservice class encapsulates the game's sophisticated methods and features. The application requires separate classes for each game, team, and player, as shown in the diagram. The ties between classes indicate their association. The numbers between the lines represent the number of relationships within each class. For instance, the GameService can have several games associated with the Game class, as well as other 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Apple, the corporation that produced MacOS, provides server-based environments.  Having a MacOS server can provide significant benefits for Mac clients on a network. Advantages include full support for Mac apps, easy management, and a fantastic graphic interface. One disadvantage of using a Mac server is its high maintenance costs. Additionally, it may not be suitable for large firms that rely heavily on third-party apps and modifications. | Linux has several advantages for web hosting.  Large organizations, like Google, have employed Linux servers. The primary benefit of using a Linux server is that it is free and open source. This means that tools may be deployed and used at a low or no cost.  Linux offers adjustable security, allowing enterprises to tailor it to their own needs. Finally, Linux web hosting supports Python, PHP, Perl, and Ruby. One drawback of using a Linux server is the steep learning curve for those without prior knowledge. Some popular apps may not be supported, therefore migrating from Windows to Linux might be challenging. | Windows is a robust operating system.  The program is proprietary and requires licensing fees.  While Windows license might be costly, there are some benefits to using a Windows web host server.  Advantages include compatibility for a wide range of applications and third-party software, as well as ease of patching and hardware upgrades. Developers who are familiar with platform scripts, such as ASP.NET and MySQL, will find complete support. | Although mobile devices for web servers are not well recognized and may seem impracticable, they may be implemented. Oracle is a company that provides mobile server-side implementation.  Oracle Database Mobile Server manages applications, users, devices, and data across large numbers of mobile or distant devices.  Oracle's benefits include support for iOS and Android development tools, as well as synchronization with NoSQL databases. |
| **Client Side** | Swift/Objective-C and Xcode skills are required while developing on the Mac. Mac customers want high-quality, optimized software, which might extend development time and expense. | Linux clients are less prevalent, therefore supporting them may require considerable effort. C++/Python is commonly used in development, as are IDEs such as Eclipse. Costs can be reduced thanks to open-source tools. | Windows clients are widespread, requiring knowledge of C#/C++ and Visual Studio. Development is generally cost-effective, but ensuring compatibility across different Windows versions can be challenging. | Mobile development involves multiple platforms (iOS, Android). This requires knowledge of Java/Kotlin (Android) and Swift (iOS), along with IDEs like Android Studio and Xcode. Costs and time are higher due to the need to support multiple platforms. |
| **Development Tools** | Swift/Objective-C and Xcode expertise are essential when creating on a Mac. Mac users prefer high-quality, optimized software, which may increase development time and costs. | Linux development uses a wide range of languages and tools, including GCC, Python, and IDEs like Eclipse and Visual Studio Code. The open-source model offers for greater flexibility and reduced expenses. | Visual Studio, which supports a variety of languages (including C# and C++), is widely used for Windows development. It is a powerful IDE with numerous features; however, the license may be pricey. | Android Studio (Java/Kotlin) and Xcode (Swift) are necessary for mobile development on both Android and iOS. Cross-platform technologies, such as Flutter or React Native, might accelerate development, however they may have downsides as compared to native development. |

**Recommendations**

**Operating Platform**: I recommend using a cloud platform like **AWS** or **Azure**. These platforms support web applications well by providing:

* **Scalable Infrastructure**: Automatically adjusts resources based on demand.
* **Cross-Platform Compatibility**: Works with various web browsers and operating systems.
* **Development Tools**: Offers tools to make development, deployment, and management easier.

**Operating Systems Architectures**: Since Draw It or Lose It runs in a browser, support for multiple browsers across different operating systems is essential. This includes:

* **Supported Browsers**: Ensure the game works with Chrome, Firefox, Safari, and Edge on Windows, macOS, Linux, iOS, and Android.
* **Testing**: Develop on any platform, but test on all target platforms to ensure compatibility. Tools like Selenium or BrowserStack can help with this.

**Storage Management**: Use cloud-based storage like **Amazon S3** or **Azure Blob Storage**. This provides:

* **Scalability**: Adapts to growing storage needs.
* **Reliability**: Ensures data is stored across multiple locations.
* **Integration**: Works well with other cloud services.
* **Backups and Versioning**: Protects data from loss and enables recovery of previous versions.

**Memory Management**: The recommended platform will use:

* **Garbage Collection**: Automates memory management to prevent leaks.
* **Memory Caching**: Services like **AWS Elasticache** or **Azure Cache for Redis** store frequently accessed data in-memory for better performance.
* **In-Memory Data Stores**: Reduces latency by keeping critical data close to the application.

**Distributed Systems and Networks**: Ensure efficient communication through:

* **RESTful APIs and WebSockets**: Enable seamless communication and real-time data transfer.
* **Content Delivery Networks (CDNs)**: Reduce latency by delivering content closer to users.
* **Microservices Architecture**: Breaks the system into independent services for

**Security**: Implement strong security measures, including:

* **Encryption Protocols (TLS/SSL)**: Protect data in transit.
* **Encryption at Rest**: Use services like AWS KMS or Azure Key Vault for stored data.
* **User Authentication**: Use secure protocols like **OAuth 2.0** and multi-factor authentication (MFA).
* **Security Audits**: Conduct regular audits to comply with regulations like **GDPR** and **CCPA**.