

# Draw It or Lose It.

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | <09/12/23> | Eric L Foster. | A detailed description of the application and its expectations. |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions(CTS) has entered into a partnership with The Gaming Room for the development of a web-based version of their game “Draw It or Lose It”. The goal of the project is to bring the experience of “Draw It or Lose It” to a larger customer base by making the program cross platform compatible while launching as a web application. The primary challenge presented here is that there is an existing android application; to which the main gameplay and user experience must remain the same when presenting the application to a larger audience across multiple platforms.

As a solution to these problems the recommendation is to use web technologies that are widely in use such as HTML5, CSS3, JavaScript and Java. Using these technologies ensures that the application will work across multiple platforms. There also needs to exist a database to store related game information such as player names, team names, game id’s etc. The database and the server side session management system needs the ability to scale based on demand; both need to exhibit some level of elasticity to reduce overhead.

## Requirements

1. Multi-Platform Compatibility: The web-based game should be accessible on various platforms.

2. Team-Based Gameplay: The game should support one or more teams, with multiple players assigned to each team.

3. Unique Game and Team Names: Game and team names must be unique. Users should be able to check name availability when choosing a team name.

4. Unique Identifiers: The system should create unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

1. In a web-based environment, latency and bandwidth can affect user experience drastically. Issues such as high latency can lead to lagging issues and most users stop using the app after experiencing high amounts of lag. One way to solve this issue is to choose game structuring and mechanics that are less sensitive to latency and bandwidth issues.

2. The game must work on a multitude of devices and web browsers, each with its own congruence issues. One way to resolve this is to develop a responsive design that adapts to different screen sizes. React would be a good frame work for creating a responsive layout.

3. Games hosted on the web are susceptible to security threats such as cheating and hacking. One way to control the impacts of nefarious actors is to implement strong authentication and authorization protocols. Always use encryption for sensitive data.

4. As the number of users increases, the game environment should be able to scale according to increased traffic and demand. A cloud-based infrastructure that exhibits elasticity and agility would be ideal to handle growth or surges during peak use times and special events.

5. The game is going to generate a large amount of data, including user profiles, game id, and leaderboards. Including centralized data storage into the cloud-based infrastructure would help to manage the large amounts of data generated and reduce the distance that data needs to travel.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The children classes Game, Team, and Player all inherit from the parent Entity class. This means that these classes have their own methods, and they also have all the public methods of the parent Entity class. The multiplicity between the child classes begins with there can be zero to many instances of the Game class instantiated from the GameService. There then can be zero to many instances of the Team class instantiated from the Game class. Lastly, there can be zero to many instances of the Player class instantiated from the Team class. We can also see examples of polymorphism with each class defining its own toString() method. Encapsulation is achieved for each class by making the class variables private. Also, the SingletonTester class is not aware of ProgramDriver class and has only one method to ensure that a single unique object exists only at one place in memory.

**"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** | Advantages include a Unix-based operating system, security features, and integration with tools commonly used for web development. Disadvantages are that Macs can be expensive and compatible software/server configurations may be limited. | An advantage of using Linux is that it is a highly customizable and very secure environment for hosting web-based applications,  A common disadvantage is the learning curve for those not familiar with its CLI and system configurations. | An advantage of Windows is a user-friendly environment for hosting web-based applications. a plethora of compatible software and development tools. The disadvantages are licensing costs and security vulnerabilities. | An advantage of Mobile devices is a highly accessible platform, allowing users to access services almost anywhere. The disadvantages are processing power, smaller screens, and varying operating systems. |
| **Client Side** | Developing software for multiple platforms on Mac requires considering the cost of Mac-specific development tools, Vast knowledge in macOS- technologies, and additional time to structure the software for the Mac environment. | Developing software for multiple platforms on Linux requires managing the cost of Linux tools, expertise in Linux technologies, and time to adapt and optimize the software to function within the Linux ecosystem. | Supporting multiple platforms with Windows requires considering the costs for Windows development tools, requiring expertise in Windows-based technologies, and allocating sufficient time to optimize the software to function within Windows OS. | Supporting multiple platforms on mobile devices involves considering development costs, expertise in mobile app development, and time for designing and testing the software to ensure a seamless user experience across various devices. |
| **Development Tools** | Relevant programming languages for development on Mac include Swift, Objective-C, and popular IDEs and tools like Xcode and AppCode. | Relevant programming languages and tools for development on Linux are languages such as C, C++, Python, and Ruby in conjunction with IDEs like Visual Studio Code or Sublime Text. Along with development tools such as Jenkins for the CI/CD pipeline | Relevant programming languages and tools for development on Windows include languages such as C#, C++, and .NET, often utilized with IDEs like Visual Studio and tools like Microsoft Azure for cloud-based deployment. | For developing software on mobile devices, Relevant programming languages and tools include Swift and Objective-C for iOS development, Kotlin and Java for Android, alongside frameworks like React Native, with IDEs such as Xcode and Android Studio, and version control tools like Git. |

## 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 infrastructure, utilizing platforms like Microsoft Azure or Amazon Web Services (AWS), would best allow The Gaming Room to expand Draw It or Lose It to various computing environments.

2. Operating Systems Architectures:

Use Amazon Web Services (AWS) as the selected cloud-based platform architecture, with an emphasis on scalability by implementing virtualization and container technologies to deploy, manage, and scale the game across various platforms, ensuring a positive user experience.

3. Storage Management:

Implement SQL storage management by integrating a RDMS, like PostgreSQL or MySQL into the cloud platform. This will enable storage and retrieval of game data, unique identifiers, game details, and team names, all while focusing on scalability.

4. Memory Management:

Optimizing memory usage and ensuring high performance for the Draw It or Lose It software on the AWS cloud platform requires the implementation of dynamic memory management techniques, including features such as garbage collection and memory pooling.

5. Distributed Systems and Networks:

Use a microservices architecture in Draw It or Lose It to facilitate communications across multiple platforms via RESTful APIs, ensuring low-latency and fault-tolerant connections. Integrate a message queue system like Amazon MQ, which is based on the Java Message Service. Amazon MQ offers scalability and a standardized communication approach for Java applications, to manage component dependencies, address connectivity challenges, and uphold data integrity within the application.

6. Security:

Utilize Amazon Web Services (AWS) for the application "Draw It or Lose It" to guarantee end-to-end encryption using HTTPS. Implement AWS Identity and Access Management (IAM) to safe guard user information. Use enhanced security measures with AWS-based Multi-Factor Authentication (MFA) and token-based authorization.