

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 8**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/20/24 | Katie Baca Dempsey | Creation |
| 2.0 | 10/6/24 | Katie Baca Dempsey | Evaluation updated |
| 3.0 | 10/20/24 | Katie Baca Dempsey | Recommendations updated |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room, developers of Draw It or Lose It, would like to develop a web-based version of their game. Currently, Draw It or Lose It is only available for Android devices. By creating a web-based version of the game The Gaming Room hopes to serve multiple platforms.

The Gaming Room has requested that a game of Draw It or Lose It be able to be played with one or more teams with each team consisting of multiple players. Each game and team name must be unique. Users will be informed if a name is available when choosing a name.

## Requirements

*A game will have the ability to have one or more teams involved.*

*Each team will have multiple players assigned to it.*

*Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*

## [Design Constraints](#_2et92p0)

Technical Constraints

We will need hosting for the web-based application and library of images.

Games, teams, and players will all have unique names and IDs.

We will ensure users will be able to add multiple players to a team and multiple teams to a game.

We will account for a high amount of traffic because of the multi-user nature of this game.

Business Constraints

We will utilize developers that have experience with web-based applications.

## [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)

Inheritance

The classes Game, Team, and Player inherit from Entity. Rather than rewrite the same functions for each class, getId() and getName() are written once and inherited by the children classes from the parent class.

Polymorphism

Polymorphism is utilized making customization of inherited methods more efficient. Each child class overrides the toString() method inherited from Entity, printing information specific to their own class.

Encapsulation

GameService, Entity and Entity’s children are utilizing private variables and lists. This demonstrates encapsulation and ensures that other classes are unable to access these private variables and lists keeping them safe from interference.

Abstraction

Abstraction extends from encapsulation in that details may be kept hidden while some functionality is exposed to other classes allowing certain elements to be utilized without needing the full picture keeping the interaction simple. For example, when adding a game, team, or player, the process is simple while behind the curtain the code first checks for an existing game, team, or player by the name provided before making the addition.

**"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** | As of 2022 MacOS no longer supports web hosting | Linux is more affordable as it is open-source. However, more experience is required when it comes to utilizing Linux.  Developed with security in mind  Known for stability  Stable without reboot for several days  High processing power | Proprietary software makes this a more expensive option than Linux. Windows is more intuitive than Linux due to its high adoption rate, making it more accessible. Also, there are many tools already included with the software.  Less secure than Linux but improving  Less stable than Linux  Environment includes GUI | The hardware, storage, processing, and memory limitations are too great to consider hosting on a mobile device |
| **Client Side** | User authentication and encryption of passwords and user info  Security protecting communication between the front-end and back-end  Responsiveness can be achieved through the use of responsive design principles  Testing on browsers supported by Mac | User authentication and encryption of passwords and user info  Security protecting communication between the front-end and back-end  Responsiveness can be achieved through the use of responsive design principles  Testing on browsers supported by Linux | User authentication and encryption of passwords and user info  Security protecting communication between the front-end and back-end  Responsiveness can be achieved through the use of responsive design principles  Testing on browsers supported by Windows | User authentication and encryption of passwords and user info  Considerations need to be taken for a range memory, storage, and that the types of connectivity used to access the app will vary in strength and speed.  Security protecting communication between the front-end and back-end  Responsiveness can be achieved through the use of responsive design principles  Testing on browsers supported by iOS and Android |
| **Development Tools** | Programming Language Options  C++  Python  Swift  HTML  CSS  JavaScript  IDE Options  Xcode  VS Code  Eclipse  PyCharm  Framework Options  Node.JS  Flutter  Sencha Touch  Corona SDK  Framework 7  SpriteKit  SceneKit | Programming Language Options  C++  Python  HTML  CSS  JavaScript  IDE Options  VS Code  Eclipse  PyCharm  Framework Options  Node.JS  Flutter  Sencha Touch  Corona SDK  Framework 7 | Programming Language Options  C++  Python  HTML  CSS  JavaScript  IDE Options  VS Code  Eclipse  PyCharm  Framework Options  Node.JS  Flutter  Sencha Touch  Corona SDK  Framework 7 | Programming Language Options  C++  Python  Swift  Java  HTML  CSS  JavaScript  IDE Options  Xcode  VS Code  Eclipse  PyCharm  Framework Options  Node.JS  Flutter  Sencha Touch  Corona SDK  Framework 7 |

## Recommendations

1. **Operating Platform**: The Linux operating platform will allow The Gaming Room to expand Draw It or Lose It to Android and iOS platforms.
2. **Operating Systems Architectures**: The Client Server Architectural Pattern should be utilized. This architectural pattern will allow a client, devices operating on Windows, Android, or iOS, to interact with the server, our network, and ultimately play Draw It or Lose It.
3. **Storage Management**: Cloud base storage should be employed. This will be cost effective, and the amount of storage can easily be scaled as needed. This will avoid any under or overinvestment and the associated complications that may arise.
4. **Memory Management**: Stored data will be optimized for a range of device sizes. For instance, the image library can be resized and compressed allowing the server to respond to a client request for an image with the version of the image file that best fits the user’s device window size. By providing the website with an appropriately sized image file for the user’s screen and associated operating system, resource use is efficient, and the network is not bogged down sending oversized files.
5. **Distributed Systems and Networks**: Many hands make light work. By spreading out the different components that come together to create Draw It or Lose It, storage, servers, and the devices used to interact with the software, we ensure efficiency. Data that will not be changed or changed infrequently, can be saved in storage where a server can call upon it when needed, freeing memory both on the server itself and users’ devices. Outsourcing server management gives us the advantage of employing experts to ensure we are able to offer uninterrupted service, and that we are able to scale to accommodate for increased traffic. Most likely, several servers will be dedicated to Draw It or Lose It to ensure the load is balanced and that if one were to fail, another server is available to take its place.
6. **Security**: Linux offers security features that will keep user information and the system itself protected. Unauthorized access will be prevented by requiring user authentication, unique user login credentials and a reasonably complex password. This can be made more secure by requiring a two-factor authentication and including biometric information as an option. Associated with the login should be role-based access control, only giving a user the minimum access that they need to interact with the application.

To be prepared for potential security attacks, consideration should be taken for known bugs that could be exploited across the various platforms being programmed for along with providing software updates adding new protections as needed. Additionally, utilizing data encryption will keep users’ accounts and information secure. Lastly, a firewall should be utilized to secure the server.