

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 2.0 | 10/15/2022 | Rushil Patel | Added information related to the software design |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions' technical consultants will design a web-based game called "Draw It or Lose It" for The Gaming Room, the customer (CTS). The game is presently available solely on Android, but it will be expanded to serve additional platforms in the future. CTS will offer the technical capabilities and environment, as well as ensuring that the scope of this project allows for the creation of a web-based version that fulfills five critical software criteria. The game will support one or more teams, each team can have several players affiliated with it, the game and team names will be unique, and only one instance of the game will exist at any one moment.

## [Design Constraints](#_2et92p0)

1. Constraint 1: Storage

2. Constraint 2: Player/Game/Team Management

3. Constraint 3: Concurrent Requests

4. Constraint 4: Security

5. Constraint 5: Player Login

6. Constraint 6: Programming Language

Constraint 1: Storage – Must account for enough server storage to host the client's stock photo library.

Constraint 2: Player/Game/Team Management - To supervise players, games, and teams, a player management system must be developed.

Constraint 3: Concurrent Requests - For a new, multi-platform player base, the server must be able to manage many concurrent requests.

Constraint 4: Player Login - A mechanism that allows players to log in and identify themselves in the game is required.

Constraint 5: Security Must have system to maintain security/login/identification to differentiate players with login authentication.

Constraint 6: Programming Language: Existing code for client’s android app is Java.

## [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 figure below demonstrates how the game "Draw It or Lose It" will be implemented in the technological environment. The graphic depicts seven classes that contribute to the environment desired by The Gaming Room.

The Entity base class is designed to store the shared characteristics (id and name) and actions (getters and toString) of its three subclasses: Game, Team, and Player.

Player extends Entity to store information about each player in the game and output that information via an override toString() function. The Player class is also related to the Team class, thus the customer's demand that "each team will have several players allocated to it" may be met. Each Team has a private list of Players as well as a public means for adding another person (addPlayer).

A Game-Team relationship exists, similar to the Player-Team relationship, in which the Game class maintains a private list of Teams and has the option to create a team (addTeam). The addTeam function uses an iterator technique to ensure that each team name is unique, as well as an override toString method to notify the user if their preferred name is already in use. This satisfies the customer requirement that "a game will be able to have one or more teams participate" and that "team names must be unique to allow users to verify whether a name is in use."

The relational pattern continues with the GameService-Game relationship, in which GameService maintains a private list of Games and can add games (addGame) as well as an iterator to determine if the Game name is unique. The GameService class is a singleton, which means that only one instance of the game can exist in memory at any given moment. To validate this, a SingletonTester class with the testSingleton() function was created and used in the ProgramDriver class. With the tests passing, all of the user software requirements for the game application have been met in an efficient and maintainable manner, while utilizing standard object oriented principles such as inheritance, overriding relations, and common design patterns such as the Singleton and Iteration patterns.

**"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)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Pros:  Simple server-side deployment.  Proven security. Extensive documentation for macOS apps. Available technical support plans and packages  **Cons:**  Limited hardware options.  Requires Mac system using macOS.  Updates solely provided through apple.  New software versions require payment.  Highest cost. | Pros:  Open-source software available.  Lowest hardware requirement.  Most cost-effective option.  Proven stability.  Community constantly updating and guarding against potential security vulnerability.  Open Source is not centrally managed.  Cons:  Not as much software available when compared to windows. | Pros:  Simple server-based deployment. Large range of software support. Wide range of hardware options. Quicker to get updates due to large userbase.  **Cons:**  Less secure than both Linux and Mac. Closed Sourced. Updates solely provided through Microsoft. Limited customization | Pros:  Can manage calls to the server side to optimize the queries on the database. Stores persistent data. Code cannot be seen by the user.  Cons:  Server side would need to integrate with cloud services or physical servers |
| **Client Side** | Pros:  Wide range of well supported web browsers featuring many tools for devs.  Easy cross browser testing software. Medium development time and deployment.  Cons:  Requires Apple product with MacOS. | Pros:  Wide range of well supported web browsers featuring many tools for devs.  Tree file structure accommodations.  Works with every web browser due to plethora of open-source software. Quick development and deployment | Pros:  Wide range of well supported web browsers featuring many tools for devs. Easy cross platform testing besides MacOS. Quick development and deployment  Cons:  More difficult to test for MacOS browser | Pros:  Expertise with android app development  Cons:  Difficulty testing other environments and browsers.  Longer development time |
| **Development Tools** | • documentation for deployment on MacOS  • Can easily run Windows and Linux through local Virtual Machines  • Application must be reviewed and approved by Apple • 99$/year for Apple Dev program • Mac OS X  • Eclipse for Java | • Deployment at any time  • Can easily run Windows and MacOS through local Virtual Machines  • Shell prompt and terminal  • Open-source community  • No license cost • Eclipse for Java | • Deployment at any time  • Extensive documentation for deployment on Windows  • Can run Linux through VM  • No license cost  • Visual Studio Code for Javascript/HTM L • Eclipse for Java | • Xcode 12 for deployment to iOS  • 99$/year Apple dev program for iOS • SwiftUI |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to We prefer Windows above all other operating systems for expanding Draw It or Lose It to other computer environments. This platform is a good competitor because of the following factors:

- Works with the latest Android build. "Install Android Studio or a cross-platform solution like as Xamarin, React, or Cordova, then configure your development environment on Windows."

- Using a set of Microsoft tools, developers can simply design, build, and deploy apps and solutions for Windows PCs.

- A plethora of emulators for cmd/powershell/ubuntu capabilities are available on Windows, allowing for testing across multiple platforms.

- The most popular and largest user base for game creation.

1. **Operating Systems Architectures**: Windows divides its operating system into two modes: user mode and kernel mode. User mode processes are directed at the user and influence much of what the user interacts with. Kernel mode is more low-level and beneath the hood, handling inputs and outputs, memory management, networking, hardware management, and routines. To store data, Windows use a directory structure. Windows also enables multiprocessing and hardware flexibility, allowing for system customization.
2. **Storage Management**: We strongly suggest Microsoft Azure for storage due to their low pricing, excellent customer service, and constant updates/support. Azure also provides the following additional features:

- Docker containers may be used to exploit cloud storage instances in the Azure Cloud computing environment.

- Cloud storage enables for easy scaling up and down dependent on user base. Storage might be raised during the first launch, when user influx is projected to be substantial - Storage choices in Azure include: Azure File system, Azure Storage Containers, and Azure Blob Storage.

- “Draw It or Lose it” has 200 8MB Base Game Images to be delivered to the application would be stored in Azure File Share, which is 1.6GB of storage per user. i. <https://azure.microsoft.com/en-us/services/storage/files/#features>

- Cloud storage is quickly becoming a widely used storage solution due to its extreme cost efficiency and scalability.

1. **Memory Management**: Windows 10 is the most recent version of the Windows operating system, and it has once again enhanced memory management to enable quicker and more effective memory loading.

This is accomplished largely by disc paging and demand paging, which operate as extensions of the computer's physical memory or RAM. Disc paging does this by allocating a portion of the hard drive as additional RAM. Demand paging works by breaking down operations into smaller tasks that are only put into memory when they are needed for immediate processing. In Windows 10, each process may also use the whole virtual memory address space, which is more than adequate for the program.

1. **Distributed Systems and Networks**: Another reason we suggest Azure as your cloud service provider is its simplicity of use when dealing with dispersed systems and networks.

- With cloud-based email alerts, Azure App analytics Logging, and monitoring technologies, Azure provides optimum uptime. When scaling up to handle 1000 concurrent games with four participants in each game, this will be important.

- By offloading the network strain to Azure, you can concentrate on the application and its features.

1. **Security**: Azure simplifies the logistics of user information and personal data protection. The devices will connect to the "Azure App Service" operating on an App Service Plan and will log in using Azure Active Directory. Azure provides the following added features: - IP settings for whitelisting access to resources (such as player or personal information) or the whole app.

- The possibility of storing data in a VPN within the cloud for increased security.

- The database might be IP Whitelist only, password protected, and require SSL communication to secure user data.

- Options for obscuring user data to secure personal data and information in the event of a breach.