

<Name-of-Software-Application>

# **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 | 09/29/24 | Jennifer Geurian | Changes to executive summary, design constraints, domain model, development requirements |

**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)

The Gaming Room’s goal is to expand their popular Android game, "Draw It or Lose It," into a web-based application that can be used across platforms. They want to target a larger audience and allow the game to be more accessible to customers

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Web-Based Distributed Environment: The main constraint in developing a web-based version of "Draw It or Lose It" lies in ensuring real-time interaction among multiple players across different platforms. This requires:

Efficient use of server resources to handle multiple, simultaneous game sessions.

Low-latency data transmission for real-time gameplay experience.

Cross-platform compatibility to reach a wider audience.

Implications: These constraints necessitate a carefully architected backend that can manage game state efficiently, a frontend that adjusts seamlessly to different screen sizes and inputs, and the use of web sockets or similar technologies for real-time communication.

## [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 UML class diagram for "Draw It or Lose It" includes classes for the game application, such as Entity, Game, Team, and Player, with Entity serving as a base class for common attributes like id and name.

Class Relationships:

Game inherits from Entity and aggregates Team, indicating that each game can have multiple teams.

Team also inherits from Entity and aggregates Player, showing that each team consists of multiple players.

This hierarchy ensures that games, teams, and players all have unique identifiers and names.

**"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** | Characteristics: User friendly and reliable  Advantages: Security and users like  Weaknesses:  Less overall users than Windows | Characteristics: Stable and scalable  Advantages:  Open-source and cost effective  Weaknesses:  Limited software support | Characteristics: user friendly and huge library of software compatibility  Advantages:  Largest user base for pc, software support  Weaknesses:  Cost of licensing and possibly security | Characteristics: user friendly and huge market  Advantages:  Largest user base overall  Weaknesses: security concerns |
| **Client Side** | Supporting multiple types of clients on Mac may require additional development efforts and costs due to platform-specific considerations | Linux client-side development may involve specific programming languages and tools, depending on the chosen Linux distributions. | Windows client-side development may require consideration of the Windows user interface and libraries. | Mobile devices are not considered as client-side platforms in this context. |
| **Development Tools** | Development tools for Mac may include Xcode and various programming languages like Swift and Objective-C. | Development tools for Linux may involve languages like Python, C/C++, and IDEs like Visual Studio Code. | Windows development tools may include Visual Studio and support for languages like C#. | Development for mobile devices typically involves platform-specific IDEs such as Android Studio and Xcode. |

## 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**: Considering the need for flexibility and scalability, it is recommended to opt for a multi-cloud approach, utilizing platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) to host and manage the data. This approach allows for redundancy and flexibility across different cloud providers.
2. **Operating Systems Architectures**: The chosen operating platform architectures should be containerized, utilizing technologies like Docker and Kubernetes. Containers provide isolation, scalability, and ease of deployment.
3. **Storage Management**: Implement a distributed file system for efficient storage management, such as AWS S3, Azure Blob Storage, or GCP Cloud Storage. This will ensure scalability and high availability of data.
4. **Memory Management**: Employ in-memory data processing frameworks like Apache Spark for handling large datasets efficiently. Distributed caching mechanisms can also be used for optimizing memory management.
5. **Distributed Systems and Networks**: Utilize cloud-based networking solutions to ensure high availability and low latency. Implement load balancing, content delivery networks (CDNs), and virtual private networks (VPNs) for secure and efficient data transfer.
6. **Security**: Implement robust security measures such as encryption (both in transit and at rest), access controls, identity and access management (IAM), and continuous monitoring to protect sensitive consumer data.

By adopting these recommendations, The Gaming Room can effectively manage and analyze the large volumes of consumer data acquired from the recent acquisition while ensuring scalability, security, and cost-efficiency.