

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## 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 &** [**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

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

[**Recommendations 8**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/22/2024 | Sumiko Mitchell | Initial document update, focusing on the executive summary, design constraints, and domain model overview. |
| 1.1 | 02/09/2024 | Sumiko Mitchell | Update to the Evaluation table**.** |
| 1.2 | 02/19/2024 | Sumiko Mitchell | Update to the **Recommendations**. |

**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)

**Draw It or Lose It** is a gaming app developed by The Gaming Room exclusively for the Android operating system. A game consists of four one-minute rounds involving teams having 30 seconds to guess a phrase, title, or thing using a gradually revealing image as a clue. If the guessing team fails to solve the puzzle before their time expires, the other teams can give one guess each within a 15-second time limit to solve that round’s puzzle.

The Gaming Room seeks to expand the reach and scope of **Draw It or Lose It** by launching a web-based version of the game. Creative Technology Solutions will assist in configuring hardware and streamlining software development to achieve this goal.

## Requirements & [Design Constraints](#_2et92p0)

* Address cross-platform and platform-specific considerations to ensure the game's compatibility across diverse platforms.
* A game must consist of one or more teams.
* A team must consist of one or more players.
* Game and team names must be unique to allow users to verify name availability.
* Each game, team, and player must have a unique identity to ensure that only one game instance exists in memory.

## [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 Room UML Diagram focuses on the game, team, and player management classes.

Classes:

* **Entity**: A base class that contains attributes and methods common to Game, Team, and Player IDs.
* **Game**: A class that includes lists of teams and manages game information associated with the GameServices class.
* **GameService**: A singleton class associated with the Game class used for game and data management.
* **Team**: A class that includes lists of players and is associated with the Game and Player classes.
* **Player**: A class containing a player ID associated with the Team class.
* Additional: The **ProgramDriver** class contains the Main method, and the **SingletonTester** class contains a method for testing game instances.

Relationships:

* The Game, Team, and Player child classes inherit from the Entity base class attributes and methods associated with IDs.
* The GameSerivce class manages zero-to-many Game class objects.
* The Game class manages zero-to-many Team class objects.
* The Team class manages zero-to-many Player class objects.

The Gaming Room UML class diagram displays object-orientated principles of inheritance and encapsulation. As described above, the Game, Team, and Player classes inherit ID attributes and methods from the Entity base class. Encapsulation is present in all game management classes, as shown by using private attributes to control access to and protect gaming data.

The Gaming Room software requirements are efficiently fulfilled in the UML class diagram using the GameService's public method getInstance, ensuring that only one game instance exists in memory. Additionally, the GameService class requires unique game, team, and player 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)

Server Side: Evaluate each operating platform's characteristics, advantages, and weaknesses for hosting a web-based software application.

Client Side: Determines the software development considerations (cost, time, expense) that are necessary for supporting multiple types of clients as they pertain to the operating platform.

Development Tools: Identifies the relevant programming languages and tools (IDE and other tools used to build this software for deployment on the operating platform).

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Apple software and products have a well-deserved reputation for being user-friendly and stable. Since Apple discontinued its macOS server software in 2022, 3rd party server software is needed to host web-based applications on Apple hardware. This project may have more robust standalone or cloud-server-based options unless the client has existing Mac server hardware or prefers only Apple products. | Linux, a free and open-source OS, is favored for web hosting due to its reliability, stability, and broad hardware compatibility. Its extensive device driver library enables flexible server hardware configurations without Mac or Windows compatibility issues. As with the MAC operating platform, web hosting requires a 3rd party software. | Microsoft offers integrated solutions for business technology needs from hardware to enterprise-level software. The Microsoft Windows server operating platform supports built-in web hosting using Internet Information Services (IIS), so there is no need to secure additional web hosting software. However, licensing costs for Windows servers can be significant, a concern for large-scale deployments. | Mobile devices like smartphones are portable, so server hosting costs should be a cloud-based solution for a lighter connectivity load. Having could-hosted server services could provide TGR with the scalability and flexibility needed for consumer demands. Licensing costs for a cloud-based server depend on the services and features TGR would utilize. |
| **Client Side** | The **Draw It or Lose It** web gaming app was initially developed exclusively for Android. Expanding its reach to include Windows and Linux platforms may entail significant costs for acquiring Apple hardware and software. Additionally, additional software purchases will likely be necessary to acquire the development tools for targeting Windows and Linux platforms. If the current TGR development staff is unfamiliar with the Apple environment, they may have a steep learning curve to overcome. | While the Linux operating platform is free, the web hosting and application development software will not be. TGR will need to consider the cost of these and other 3rd party tools for cross-platform development, especially with the client's desire for robust gaming scalability to accommodate "thousands of players." Experience with development on the Linux platform would be ideal for TGR, ensuring optimized Linux performance and knowledge of the tools necessary for dealing with cross-platform compatibility issues. | Microsoft provides comprehensive, integrated hardware and software solutions that TGR could leverage to develop a robust web-based gaming application with cross-platform compatibility. However, it's important to note that licensing expenses will be associated with the server operating system, along with costs for essential development software and tools. These ongoing expenses may escalate with enterprise-level usage or the addition of advanced features. The widespread popularity of the Windows server platform ensures that experience with the system is common, and developers can access training resources readily available from Microsoft to enhance their proficiency. | Client-side cost considerations for developing a web-based mobile game would include acquiring target platform-specific development software and tools necessary for designing and coding the game interface and mechanics optimized for mobile devices. Additionally, budgeting for server hardware and software needs to support the game's functionality, such as managing user accounts, storing game data, and facilitating multiplayer interactions, is essential. |
| **Development Tools** | Apple APIs, Swift | Docker, Apache HTTP server, NGINX | Windows Server, Visual Studio, ASP.net | iOS: Xcode Android: Android studio |

## Recommendations

1. **Operating Platform**: The Gaming Room aims to expand the audience of its Android-only mobile gaming app, "Draw It or Lose It," by making it available not only on Android devices but also on various other operating platforms, including Linux, Windows, and Apple iOS devices. To achieve this goal, I recommend utilizing a cloud-based operating platform. Leveraging An Existing Cloud-Based Platform, Such As Microsoft Azure, Will Ensure Cross-Platform compatibility, scalability, and help contain development and deployment costs.
2. **Operating Systems Architectures**: Microsoft Azure is not a traditional operating system but a cloud platform that offers various services built on top of different operating system architectures. The Gaming Room can use Azure's flexibility for cross-platform development via virtual machines or container services like Azure Kubernetes Service (AKS) and web-based access to the "Draw It or Lose It" application.
3. **Storage Management**: The Gaming Room has an image library of 200 high-definition images, each around 8 MB. Expanding the image library to enhance the "Draw It or Lose It" user experience and avoid image repetition is essential. Microsoft Azure offers solutions for this expansion with Azure Blob Storage, which provides scalable storage for images, and Azure SQL Database, a relational database service, for managing image metadata, game, and player information.
4. **Memory Management**: Microsoft Azure memory management techniques are similar to operating systems, such as memory allocation, compressing, swapping, and monitoring. For The Gaming Room, Azure Virtual Machines will be dynamically allocated and managed to ensure optimal memory usage for running "Draw It or Lose It" across the Linux and Windows environments.
5. **Distributed Systems and Networks**: Microsoft Azure offers services such as Azure App Service, which facilitates the development and deployment of microservices within a distributed architecture. This architecture enables communication between application components like microservices and allows them to exchange and synchronize data regardless of the deployment platform. Microsoft's robust network infrastructure with built-in redundancy helps minimize the impact of network outages, limiting disruption for The Gaming Room customers.
6. **Security**: The top priority is safeguarding user information and The Gaming Room data. Microsoft Azure has a multi-layered approach to address most security concerns. Entra ID, integrated within Azure, authenticates and authorizes users, applications, and resources, while Multifactor-Authentication (MFA) strengthens this security position, ensuring secure access to "Draw It or Lose It."

Microsoft Azure primarily uses a pay-as-you-go (PAYG) model, offering flexibility and scalability for The Gaming Room to adjust their usage and expenditures based on their current or predicted situation requirements. While some Azure services have subscription options, the PAYG model is an industry approach, and The Gaming Room would benefit from the dynamic resource allocation and cost control.