

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Tyler Tirado

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 | 09/17/23 | Tyler Tirado | Changes to the main cover page, the document revision history section, executive summary, design constraints, system architecture view, domain model and recommendation. |

**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)

Our client, The Gaming Room, wants to create a project that is a web-based game based on the game draw it or lose it and will be hosted on multiple platforms. The objective of the game is multiple teams, each containing multiple players, playing for four rounds each one minute long. In these rounds a picture is pulled and presented to the teams and one team guesses until time runs out. If not answered each of the other teams will get 15 seconds to guess.

## 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)

The design constraints are as follows:

The game must run on multiple platforms. In each game there will be multiple teams, in each of these teams there will be multiple players. There can only be ONE instance of the game running at any given time. Both the Game Name and Team Name must be unique so as to be able to check availability.

## [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)

Starting with Entity it creates a relationship between the Game, Team, and Player classes, inheriting all information that Entity has. Both Game and Team have relationships with classes GameService and Player, respectively. This relationship is known as aggregation and it is when a class has a reference to another class, in this case, GameService has a reference to Games and Games to Team and Team to Player.

**"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** | While macOS can be used for server side development, it is a less common OS compared to Linux for production environments. | Linus is a good option for server side application because of its stability and security features but it may require more expertise to set it all up correctly.  Linux offers various server-based deployment methods for hosting web applications including Ubuntu Server and CentOS | Similarly to Mac, Windows can be used for server side development it is just less common than Linux and thus may require higher licensing costs for server deployment.  Windows server offers strong supports for.NET applications, as well as being very user-friendly. | Excellent accessibility allowing user to access apps anywhere they have an internet connection. They also support touch interfaces and sensors. Some downsides could include lower processing power and varying levels of compatibility due to high volume of different devices.  Using mobile devices for service side development is less viable than the other options since it would require hosting a server as well as the application development. |
| **Client Side** | Higher cost of entry for users. macOS can be a bit trickier to get setup and require a little more time and expertise.  Using cross-platform development frameworks like React Native of Flutter can help reduce your development costs since you can write code and deploy it on multiple platforms. | There is a high cost in both time and expertise when using Linux.  Still by using a development framework like React Native, you are able to development with a single codebase, and this solution is optimal for proper development of our application on multiple platforms. | Pretty easy to learn and use, little expertise needed.  React Native would also be the choice when it comes to the windows environment, since its still multiplatform compatible as well as being open source to minimize costs and licensing fees. You will need a development team that has experience developing with React Native. | Will help provide easy accessibility to end users and developers, but may require a bit of expertise to implement.  React Native frameworks allow for cross-platform development of mobile applications for iOS and Android. They significantly reduce the need for separate codebases for each platform. |
| **Development Tools** | Relevant programming languages would be HTML, CSS, JavaScript. As well as Visual Studio Code, Sublime Text, and WebStorm for the IDEs. Libraries to support front end development. For backend development using programs such as Node.js. | Relevant programming languages would be HTML, CSS, JavaScript. As well as Visual Studio Code, Sublime Text, and Atom for the IDEs. Libraries to support front end development. For backend development using programs such as Node.js. | Relevant programming languages would be HTML, CSS, JavaScript. As well as Visual Studio Code, Visual Studio and Sublime Text for the IDEs. Libraries to support front end development. For backend development using programs such as Node.js. | Relevant programming languages would be HTML, CSS, JavaScript. For Mobile Web Development you can use React Native and Flutter. As well as IDEs such as Visual Studio Code, Android Studio and Xcode. For backend development using programs such as Node.js. |

## 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**: Windows as the primary operating platform provides several advantages, including a broad user base, a robust development ecosystem, and compatibility with various devices. This choice ensures that the Draw It or Lose It game can be developed and expanded with Windows as the foundation while maintaining compatibility with other platforms.
2. **Operating Systems Architectures**: Windows supports both 32-bit and 64-bit architectures, providing flexibility for deploying the game to different devices and systems.
3. **Storage Management**: Microsoft Azure offers scalable and secure cloud storage solutions, such as Azure Blob Storage, suitable for storing game assets, user data, and drawings. It ensures data durability, accessibility, and seamless integration with Windows-based applications.
4. **Memory Management**: Windows manages memory allocation and deallocation efficiently through its memory management techniques.
5. **Distributed Systems and Networks**: Implement a client-server architecture using Windows Server for the backend and Windows-based clients for communication between various platforms. A client-server architecture allows for centralized game logic and data storage on Windows Server, enabling seamless communication between Windows-based clients and other platforms. Dependencies can be managed by defining clear APIs and leveraging standard network protocols like HTTP or WebSocket.
6. **Security**: Implement user authentication and authorization using Windows Authentication mechanisms for Windows-based clients. For cross-platform clients, employ industry-standard authentication protocols like OAuth. Utilize Windows Firewall and network security policies to control traffic and restrict access to authorized entities. Encrypt user data and sensitive information using Windows Data Protection API (DPAPI) for Windows-based clients. Ensure that all platforms adhere to security best practices, including regular updates and patches, input validation, and secure coding practices. Employ third-party security tools and services to scan for vulnerabilities.