

<Draw It or Lose It>

# **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 | <03/20/2024> | <Aspen Koreny-Crawford> | * 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. * Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player. |
| 1.0 | <4/04/2024> | <Aspen Koreny-Crawford> | Evaluate various platforms' characteristics, advantages, and weaknesses for hosting a web-based software application.  Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients.  Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on each operating platform. |
| 1.0 | <4/17/2024> | <Aspen Koreny-Crawford> | Addressed recommendations for Operating Platform, Operating System Architectures, Storage Management, Memory Managememt, Distributed Systems and Networks, as well as Security. |

**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 task at hand is to design software for a game application called "Draw It or Lose It" for The Gaming Room. The objective is to develop a web-based distributed game application that allows users to play a drawing game where they must guess the word based on the drawing. To efficiently manage games, teams, and players, we propose to design a system architecture using object-oriented programming principles. To ensure only one instance of the GameService exists and allow for efficient searching and management of game-related entities, we plan to implement the singleton pattern for the GameService class and the iterator pattern for game and team management.

## Requirements

The client has specified their business and technical requirements for the "Draw It or Lose It" game application. The application must be a web-based distributed game that allows users to play and manage games, teams, and players. Some critical features include unique games and team names, efficient game and team management, and secure user interaction.

## [Design Constraints](#_2et92p0)

When creating a game application for a web-based distributed environment, it is essential to consider certain design constraints. These constraints include ensuring compatibility across various operating platforms like Mac, Linux, Windows, and mobile devices. Moreover, only one instance of the game must be running at a time. To achieve this, the game name and team names should be unique, which allows the application to check if a particular game name is currently in use. Additionally, each game should have the ability to enable one or more teams to play.

## [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 is an essential tool in the development of Draw It or Lose It software. It outlines seven classes that demonstrate some of the most important object-oriented programming principles such as inheritance, encapsulation, association, and aggregation. The Entity class is the superclass, and Game, Team, and Player classes inherit attributes and methods from it. The diagram illustrates the associations between the classes, which shows how objects collaborate and interact. Additionally, it depicts aggregation relationships, which emphasizes the ownership or containment of instances. Overall, the diagram provides a clear visualization of the software's architecture, which helps in the efficient implementation and understanding of class relationships.

**"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** | When using a Mac for server hosting, its stability, security, and ease of use make it a dependable choice for web applications. Mac servers support popular development tools and technologies like Apache, PHP, and MySQL, but its limited scalability and higher hardware cost may not be suitable for organizations with budget constraints. | Linux is a great platform for hosting web-based software applications due to its robustness, flexibility, and cost-effectiveness. It provides a wide range of features and tools for web development, including Apache, Nginx, PHP, and MySQL. Its scalability allows applications to handle high traffic loads with ease. However, Linux requires more technical expertise to set up and maintain, which may not be suitable for organizations with limited technical resources. | Windows servers are user-friendly for web-based software applications and support Microsoft technologies like IIS, ASP.NET, and SQL Server. They integrate well with Microsoft products but are more expensive than Linux servers and require regular updates for security. | Mobile devices are mainly used as clients to access web-based applications, but they can also function as servers to host lightweight web services or APIs. However, due to their limited processing power, storage capacity, and battery life, mobile devices are less suitable for hosting complex web applications compared to dedicated server platforms such as Mac, Linux, or Windows. While mobile devices can be used for development and testing purposes, they are not commonly used as primary servers for hosting web-based software applications in production environments. |
| **Client Side** | Developing software for multiple types of clients on Mac involves considering cost, time, and expertise. This may require investment in Apple hardware and software development kits, and proficiency in programming languages and frameworks commonly used for macOS development. Time-to-market may also be influenced by the availability of skilled Mac developers and the complexity of the application being developed. | Developing software for multiple clients on Linux requires expertise in Linux-based development tools such as C, C++, Python, and GTK. The availability of skilled Linux developers and the complexity of the application being developed may affect the time-to-market. Linux is an open-source operating system, making it a cost-effective solution for software development. | Developing software for multiple types of Windows clients involves considering cost, time, and expertise. This process may require investment in Microsoft development tools and technologies like Visual Studio and .NET framework. Proficiency in programming languages like C# and XAML is also necessary. Compatibility testing across different Windows versions and application complexity may impact time-to-market. | Developing mobile apps for multiple types of clients requires considering the diversity of platforms and form factors, and taking into account factors such as cost, time, and expertise. The development costs can vary depending on the target platform(s) and the complexity of the app. Time-to-market can be influenced by different factors, including platform-specific development requirements, app store approval processes, and compatibility testing. |
| **Development Tools** | Developers have various programming languages and development tools when creating web-based software applications for Mac computers. JavaScript is a popular choice for front-end development, Node.js is commonly used for server-side scripting, and Swift is the go-to option for creating macOS and iOS applications. Integrated Development Environments (IDEs) like Xcode, JetBrains IntelliJ IDEA, and Visual Studio Code are widely used for writing, debugging, and testing software. Developers often use frameworks and libraries such as React.js, Angular, Express.js, and Flask to simplify development and increase productivity. | Developing web-based software applications for Linux involves choosing from a wide range of programming languages and development tools. Common programming languages include JavaScript, Python, PHP, and Ruby for server-side scripting, and HTML, CSS, and JavaScript frameworks for front-end development. Popular development tools include text editors, such as Vim and Emacs and IDEs like Visual Studio Code, PyCharm, and Sublime Text. Additionally, containerization technologies like Docker and orchestration tools like Kubernetes are widely used for deployment and management of applications on Linux servers. | To develop web-based software applications for Windows, you need to know a variety of programming languages and tools. C#, ASP.NET, and JavaScript are used for server-side scripting, while HTML, CSS, React.js, and Angular are used for front-end development. You can use Integrated Development Environments (IDEs) like Visual Studio and Visual Studio Code for coding, debugging, and testing. Additionally, IIS and Microsoft Azure offer hosting and deployment solutions for web-based applications on Windows servers. | To develop web-based applications for mobile devices, developers must consider the unique features and constraints of these platforms. Common programming languages used for mobile web development are HTML, CSS, and JavaScript, while frameworks like React Native and Flutter allow cross-platform app development. IDEs like Visual Studio Code, Android Studio, and Xcode support mobile web development with coding, debugging, and testing tools. Responsive design and PWA principles ensure a seamless user experience across different screens and devices.Top of Form |

## 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:** For expanding Draw It or Lose It, a cloud-based platform like Microsoft’s Azure in Windows is an appropriate choice due to its scalability, compatibility, and ease of deployment across different environments. Azure provides a robust infrastructure for hosting web-based applications, offering flexible scaling options to accommodate varying levels of traffic. Its compatibility with Windows operating systems ensures seamless integration with existing technologies and tools, streamlining the deployment process for the game application.
2. **Operating Systems Architectures:** Windows operating systems follow a modular architecture, with layers including the kernel, hardware abstraction layer, executive services, and user-mode components. It provides a stable and secure environment for hosting Draw It or Lose It, with support for various hardware configurations and system architectures. Windows architecture allows for efficient resource allocation and management, ensuring optimal performance and reliability for the game application.
3. **Storage Management:** Microsoft offers robust storage management solutions for Windows deployments, providing scalable, durable, and highly available storage options. Azure Storage, integrated with Azure services, offers a variety of storage services such as Blob storage, File storage, and Table storage, catering to diverse storage requirements of Draw It or Lose It. These solutions ensure efficient data management across different platforms and devices, enhancing accessibility and reliability of game assets.
4. **Memory Management:** Memory management in Windows is handled by the operating system's memory manager, employing techniques such as virtual memory, paging, and memory allocation algorithms. The modular architecture of Windows optimizes memory usage for Draw It or Lose It, ensuring efficient resource utilization and performance. Windows memory management system dynamically adjusts memory allocation based on application demands, providing adequate memory resources to support real-time rendering and display of high-definition images during gameplay.
5. **Distributed Systems and Networks:** Windows-based deployments can leverage Windows’ networking services to facilitate communication between various platforms and devices. Draw It or Lose It can utilize messaging protocols for seamless integration across distributed systems, enabling efficient data exchange and collaboration between game instances. Windows networking infrastructure ensures reliable connectivity and communication between clients and servers, mitigating risks associated with network connectivity issues and ensuring an uninterrupted gameplay experience.
6. **Security:** Windows environments offer a range of security features, including Windows Defender for endpoint protection, BitLocker for data encryption, and Windows Firewall for network security. By following recommended best practices for identity management, Draw It or Lose It can ensure the protection of user information across Windows platforms. Implementing encryption mechanisms and access control measures will further enhance the security posture of the game application, safeguarding sensitive data from unauthorized access and cyber threats.