

The Gaming Room: Draw It or Loose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 3/18/2023 | Matthew Tyson | Initial Commit |
| 2.0 | 4/2/2023 | Matthew Tyson | Evaluation |
| 3.0 | 4/16/2023 | Matthew Tyson | 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)

We have been hired by The Gaming Room to develop a web-based version of their game, Draw It or Lose It, which is currently available as an Android app only. The new game should support multiple platforms and have the ability to have one or more teams with multiple players assigned to each team. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name, and only one instance of the game can exist in memory at any given time.

To accomplish this, we will be using the Unified Modeling Language (UML) class diagram to represent the game application's domain model. We will be using object-oriented programming principles to create classes that will support the game's software requirements. Additionally, we will use design patterns to create unique identifiers for each instance of a game, team, or player.

## Requirements

The client has requested that the following software requirements be met for the game application:

1. A game will have the ability to have one or more teams involved.
2. Each team will have multiple players assigned to it.
3. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
4. 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.

## [Design Constraints](#_2et92p0)

The design constraints for developing the game application in a web-based distributed environment are as follows:

1. The game application must be developed in a way that allows it to be accessed from multiple platforms.
2. The game application must have a responsive design that can adjust to different screen sizes.
3. The game application must use a library of stock drawings for rendering clues.
4. The game application must implement software design patterns to ensure efficient and scalable development.
5. The game application must use unique identifiers for each instance of the game, team, and player.
6. The game and team names must be unique to allow users to check whether a name is in use when choosing a team name.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

Class Descriptions:

* ProgramDriver: This class contains a single method, main(), which is the entry point for the program.
* SingletonTester: This class contains a single method, testSingleton(), which tests the Singleton design pattern.
* Entity: This class represents an entity in the system and has two properties, an ID and a name.
* GameService: This class manages the creation and retrieval of Game objects. It maintains lists of Game objects, as well as the next available IDs for games, players, and teams.
* Game: This class represents a single game and has a list of Team objects.
* Team: This class represents a team in a game and has a list of Player objects.
* Player: This class represents a player on a team.

Relationships:

* ProgramDriver <<uses>> SingletonTester: This relationship indicates that the ProgramDriver class uses the SingletonTester class.
* GameService is connected to Game: This relationship indicates that the GameService class manages Game objects.
* Game is connected to Team: This relationship indicates that a Game object has a list of Team objects.
* Team is connected to Player: This relationship indicates that a Team object has a list of Player objects.
* Game, Team, and Player have a connection with Entity: This relationship indicates that Game, Team, and Player objects are all entities and have an ID and a name.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | macOS offers a built-in web server, Apache, which can be used to host the application. Alternatively, other web servers like Nginx or Node.js can be used.  As of April 21, 2022, Apple has discontinued macOS Server. Existing macOS Server customers can continue to download and use the app with macOS Monterey.  macOS is a proprietary operating system, and the license for the server version can be costly. | Linux has a wide range of web servers available, including Apache and Nginx, which can be used to host the application.  Linux is open-source software and does not require a license for the server version. However, support and maintenance costs may be necessary. | Windows Server offers Internet Information Services (IIS) as its built-in web server, which can be used to host the application.  Windows Server is a proprietary operating system, and the license for the server version can be costly. | Mobile devices do not have the capability to host a web-based software application in a server-side configuration. Instead, the application must be hosted on a server and accessed by the mobile devices via the internet. |
| **Client Side** | To ensure compatibility with all web browser platforms on macOS, the development process must include testing on popular browsers like Safari, Chrome, and Firefox. | To ensure compatibility with all web browser platforms on Linux, the development process must include testing on popular browsers like Chrome, Firefox, and Opera. | To ensure compatibility with all web browser platforms on Windows, the development process must include testing on popular browsers like Chrome, Edge, and Firefox. | To ensure compatibility with all mobile devices, the application must be developed as a responsive HTML interface that can adapt to various screen sizes and orientations. |
| **Development Tools** | Developers must have expertise in the relevant programming languages like JavaScript, HTML, and CSS, as well as frameworks like React or Angular.  IDEs like Visual Studio Code or Xcode can be used for development. | Developers must have expertise in the relevant programming languages like JavaScript, HTML, and CSS, as well as frameworks like React or Angular.  IDEs like Visual Studio Code or Eclipse can be used for development. | Developers must have expertise in the relevant programming languages like JavaScript, HTML, and CSS, as well as frameworks like React or Angular.  IDEs like Visual Studio or Visual Studio Code can be used for development. | React Native: React Native is a popular cross-platform app development framework that uses JavaScript programming language and allows developers to build native apps for both iOS and Android platforms. It can be used with a variety of IDEs, including Visual Studio Code, Atom, and WebStorm. |

## 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**: A possible operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments is the web browser. The web browser is a software application that can run on various devices, such as desktops, laptops, tablets, and smartphones. The web browser can also support different operating systems, such as Windows, Linux, MacOS, Android, and iOS. The web browser can access the game application through the internet or a local network, using standard protocols such as HTTP and WebSocket.
2. **Operating Systems Architectures**: The web browser operates on different operating system architectures, depending on the device and the operating system. For example, Windows uses a hybrid kernel architecture that combines features of monolithic and microkernel architectures. Linux uses a monolithic kernel architecture that integrates all the core functions into a single kernel. MacOS uses a hybrid XNU kernel architecture that combines features of monolithic and microkernel architectures. Android uses a modified Linux kernel architecture that adds features such as Binder and Ashmem. iOS uses a hybrid XNU kernel architecture that adds features such as Mach-O and dyld.
3. **Storage Management**: A possible storage management system to be used with the web browser operating platform is the IndexedDB API. The IndexedDB API is a web standard that provides a low-level interface for storing structured data in the web browser. The IndexedDB API allows the game application to store data such as user preferences, game settings, game progress, and game scores locally on the device. The IndexedDB API also supports transactions, indexes, queries, and concurrency control.
4. **Memory Management**: The web browser uses memory management techniques for the Draw It or Lose It software based on the JavaScript engine and the garbage collector. The JavaScript engine is a software component that executes the JavaScript code of the game application in the web browser. The JavaScript engine allocates memory for objects, variables, functions, and other data structures dynamically at runtime. The garbage collector is a software component that periodically scans the memory and frees up the memory occupied by unused or unreachable objects. The garbage collector helps to prevent memory leaks and improve performance.
5. **Distributed Systems and Networks**: The Draw It or Lose It software can communicate between various platforms using distributed software and the network that connects the devices. One possible distributed software architecture is the client-server architecture, where the game application consists of two components: a client component that runs on the web browser of each device, and a server component that runs on a remote machine. The client component is responsible for rendering the user interface, processing user input, and sending and receiving messages to and from the server component. The server component is responsible for managing the game logic, coordinating the game sessions, and broadcasting messages to all the clients. The network that connects the devices can be either wired or wireless, using protocols such as TCP/IP, UDP/IP, or HTTP.
6. **Security**: To protect user information on and between various platforms, the Draw It or Lose It software can use security techniques such as encryption, authentication, authorization, and auditing. Encryption is a technique that transforms data into an unreadable form using a secret key or algorithm. Encryption can be used to protect data at rest (such as in IndexedDB) or in transit (such as in WebSocket). Authentication is a technique that verifies the identity of a user or a device using credentials such as username and password or digital certificates. Authentication can be used to prevent unauthorized access to the game application or the server component. Authorization is a technique that determines what actions or resources a user or a device can access based on their roles or permissions. Authorization can be used to restrict access to certain features or functions of the game application or the server component. Auditing is a technique that records and monitors the activities or events of users or devices in logs or reports. Auditing can be used to detect anomalies or violations of security policies or rules.