

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/21/2024 | Xander O’Hara | Updated sections: Executive Summary, Design Constraints, Domain Model |
| 1.1 | 08/04/2024 | Xander O’Hara | Updated all sections under “evaluation” |
| 1.2 | 08/15/2024 | Xander O’Hara | Updated all sections under “Recommendations” |

## [Executive Summary](#_heading=h.35nkun2)

The Gaming Room has given us the task of porting their Android-only game, "Draw It Or Lose It," to a web version compatible with Windows, Mac, Linux, and iOS. For a seamless cross-platform experience.

The game must support multiple teams, each with multiple players, and ensure unique game and team names using an Iterator pattern. To maintain a single instance of the game in memory, we will implement a Singleton pattern.

## [Design Constraints](#_heading=h.1ksv4uv)

**Software Constraint: Refactoring**

The Android app, developed in Android JS (JavaScript, HTML, CSS), will be mostly refactored into React Native for web, increasing development time.

**Software Constraint: Cross-Browser Compatibility**

The application must run on all systems and browsers, including Android, iOS, Mozilla, Chrome, Firefox, and Internet Explorer, requiring extensive cross-browser testing.

**Software Constraint: User Data Management**

User data must be handled securely and privately across platforms. Secure coding practices will be employed, using the "Draw It or Lose It API" for user login and authentication with a RESTful approach.

**Hardware Constraint: Cross-Platform Development**

The project requires a team skilled in React Native for web and familiar with all modern browsers across Mac, Windows, Linux, Android, and iOS.

**Software Constraint: Mobile and Desktop Landing Pages**

The app, available on Google Play and the Apple App Store, needs a landing page with "Get it On the X Store" on mobile interfaces. On desktop browsers, the game should be playable directly from the homepage.

## [System Architecture View](#_heading=h.44sinio)

**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](#_heading=h.2jxsxqh)

The UML diagram comprises seven classes: Game, Team, Player, GameService, SingletonTester, ProgramDriver, and Entity.

**Generalization and Inheritance:**

- Classes Game, Team, and Player inherit from the Entity class, demonstrating multiple inheritances and abstraction.

**Associations and Multiplicity:**

- Game, Team, and Player classes have direct associations with GameService, which can manage zero to many instances of these classes. Multiple players can be added to a team, each identified by an id and name. Multiple teams can be added to a game, each identified by a name.

**Singleton Pattern:**

- The GameService class uses a singleton method to ensure only one instance exists, accessed via the public getInstance method, demonstrating encapsulation and the Singleton pattern. The SingletonTester class verifies this single instance, linked to the ProgramDriver class that holds the main method.

**Polymorphism and Method Overloading:**

- GameService demonstrates polymorphism through method overloading of the getGame function, invoking the correct method based on method signatures.

**Iterator Pattern:**

- The addGame, addPlayer, and addTeam methods use the Iterator pattern to manage unique instances within encapsulated lists without exposing underlying data types.

**Composition:**

- GameService has a composition relationship with Game (List<Game>), Game has-a Team, and Team has-a Player, each reinforcing the structural hierarchy and ensuring unique identification.

**Testing and Execution:**

- The SingletonTester class ensures only one instance of the game runs at a time, associated with the ProgramDriver class where the main method is stored.

The UML diagram shows key OOP principles like abstraction, encapsulation, inheritance, polymorphism, and the design patterns Singleton and Iterator.

**"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](#_heading=h.z337ya)

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** | Mac servers seem pretty reliable and user-friendly. They'd work great with other Apple devices, so if my team uses Macs, it might be easier. But they can be pricey, and some web hosting tools might not work as well on Mac. Also, not as many people know how to manage Mac servers compared to Linux or Windows. | Linux is popular for web servers. It's free, which helps keep costs down. It's known for being stable and secure, with high customizability. Many web hosting tools work well with Linux. The main challenge is that it might require specialized knowledge to manage effectively. | Windows servers are common in businesses. They're user-friendly and integrate well with other Microsoft products. However, they can be expensive, especially for smaller projects. They generally require more powerful hardware than Linux servers. | Mobile devices aren't typically used as servers for web apps. They're more for accessing web apps through browsers or dedicated apps. I'll need to consider how my server handles requests from mobile devices, optimizing for varying connection speeds and screen sizes. |
| **Client Side** | For Mac clients, I'll need to ensure my web app works well in Safari, which can differ from other browsers. It might take extra time to test and fix Mac-specific issues. If I'm using any special Mac features, I'll need to either learn them or find developers who know Mac development, which could increase costs. | For Linux clients, I'll mostly focus on how my web app performs in browsers like Firefox or Chrome. It's similar to developing for other desktop systems. I might need to test on different Linux distributions to ensure compatibility, which could be time-consuming. | For Windows clients, I'll need to ensure my web app works well in Edge and possibly Internet Explorer for older systems. Most developers are familiar with Windows, which is helpful. Testing might take longer due to the various Windows versions in use. | Developing for mobile clients involves dealing with various screen sizes and operating systems (mainly iOS and Android). It will require more time for testing and ensuring compatibility across devices. If creating native apps, I might need to learn both iOS and Android development, which is challenging and time-intensive. |
| **Development Tools** | For Mac development, I can use languages like Swift for native apps or JavaScript, HTML, and CSS for web development. Xcode is the primary IDE for Mac development, and it's free. Visual Studio Code is another option. For web development, tools like WebStorm or Atom are available. | Linux offers a wide range of development options. I can use various programming languages like Python, Java, PHP, or Ruby. For IDEs, Visual Studio Code is popular, as are JetBrains products like PyCharm or IntelliJ IDEA. Vim or Emacs are options for those who prefer them. | Windows offers numerous development tools. Visual Studio is excellent for .NET development. For web development, options include Visual Studio Code and Sublime Text. Windows supports most programming languages, like C#, Java, JavaScript, and Python. | For iOS, Xcode and Swift or Objective-C are necessary. Android development uses Android Studio with Java or Kotlin. For web apps on mobile, I'll use HTML, CSS, and JavaScript, possibly with frameworks like React Native or Flutter for a more native feel. Cross-platform tools like Xamarin are an option, but require looking at the learning curve vs the benefits. |

## 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 a good operating platform for the game, I will choose a cloud-based platform. Something like AWS or Azure because they're flexible enough to handle all the different devices and operating systems we need to support. Also, it'll make it a lot easier when it comes to managing the game across these different platforms as well.
2. **Operating Systems Architectures**: For the server side, I think we should stick with Linux because It's reliable, performs well, and is cheap to run and maintain. On the client side, we'll be running in web browsers, so we don't need to worry too much about the underlying OS. This way, we can focus on creating a great web experience that works everywhere.
3. **Storage Management**: for storage management we could use a solid relational database for user accounts and game state and potentially also use a non-relational database for quicker access to game data. For all the game assets and images, we can use object storage for fast load times.
4. **Memory Management**: Most of the heavy lifting for memory management will be handled by our cloud platform. That said, we still need to be able to handle memory in our web app. We'll need to keep an eye on our data structures and caching, and make sure our JavaScript is working well. On the server side, we'll optimize based on the programming language.
5. **Distributed Systems and Networks**: To get all our platforms talking to each other correctly, we'll set up a RESTful API as it is a trusted method. We'll also use Web Sockets for real-time loading for multiplayer games. We will also build in fallbacks for when connections drop, and make sure everything stays in sync.
6. **Security**: For security, we can encrypt everything in transit with HTTPS, use robust authentication methods, and make sure we're encrypting sensitive data. We'll also need to stay on top of updates and patches, and regularly check for any vulnerabilities for safety. We will also make sure we're following all the necessary data protection regulations.