

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc142861040)

[Table of Contents 2](#_Toc142861041)

[Document Revision History 2](#_Toc142861042)

[Project 1 4](#_Toc142861043)

[Executive Summary 4](#_Toc142861044)

[Requirements 4](#_Toc142861045)

[Design Constraints 4](#_Toc142861046)

[System Architecture View 4](#_Toc142861047)

[Domain Model 4](#_Toc142861048)

[Evaluation 5](#_Toc142861049)

[Recommendations 6](#_Toc142861050)

[Project 2 6](#_Toc142861051)

[Server Side 6](#_Toc142861052)

[Client Side 7](#_Toc142861053)

[Development Tools 7](#_Toc142861054)

[Project 3 8](#_Toc142861055)

[Operating Platform 8](#_Toc142861056)

[Operating System Architectures 8](#_Toc142861057)

[Storage Management 9](#_Toc142861058)

[Memory Management 9](#_Toc142861059)

[Distributed Systems and Networks 9](#_Toc142861060)

[Security 9](#_Toc142861061)

[Work Cited for Project 3 10](#_Toc142861062)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| Project 1.0 | 07/16/2023 | Riuthu Mathews | The Gaming Room blueprint for Draw it or Lose it |
| Project 2.0 | 07/30/2023 | Riuthu Mathews | Updating, The Gaming Room blueprint with project 2 requirements, specifically the client-server architecture. |
| Project 3.0 | 08/11/2023 | Riuthu Mathews | Updating, The Gaming Room blueprint with project 3 requirements. |

# Project 1

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room, the client, wants to develop a web version of the game called Draw it or Lose it. The game must have the ability to have multiple teams of people playing for different number of rounds. Each team has opportunity to guess the rendering picture. While the picture is being slowly reveled, within a 30 second time frame, one team gets to guess until till the time runs out. If not answered each opposing team member gets to answer till 15 seconds runs out.

## Requirements

The app should be available in web based version. The app should provide an engaging and enjoyable gaming experience while involving multiple teams. Each team has multiple player and each should have the ability to choose usernames. Additionally, there should only be one instance of the game, this is done via creating identifiers for everyone.

## [Design Constraints](#_2et92p0)

The app should have to be consistent across different platforms, mobile and web, to ensure intuitive experience for users; Each team must support multiple people; Usernames must be unique and it should check if the name is in use or not; Only one instance of the game can exist at any given time.

These are the requirements but also, they pose as constraint as well. The game must run on all devices and each device may favor a different framework of language. Also, narrowing the specification to single instance but simultaneously supporting different users limit the approach we have when it comes to coding.

## [System Architecture View](#_ilbxbyevv6b6)

The system architecture of the application will be based on a web based distributed model. It will consist of multiple attributes, including the front end (smartphone app), server side (backend infrastructure), and the database for storing game data. The communication between these components will be conducted via APIs and network protocols.

## [Domain Model](#_8h2ehzxfam4o)

The class Entity is the super class or the base class for Game, Team, and Player class; Meaning, the classes following class Entity inherit from Entity. This UML diagram shows the inheritance relationship between classes. We can also observe that the class GameService has a reference of class Games, class Games has a reference to class Team, and class Team a reference to class 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Has options for web based distribution but is not preferred. Also needs knowledge of macOS servers. | Cost efficient, most popular amongst developer but web based distribution is limited. | Most popular amongst users, and developer friendly but prone to security treats. | Most preferred way for users, cost efficient but different mobile platforms requirement more development. |
| **Client Side** | More knowledge but similar cost. Also, more suited for web version | Most cost efficient but requires more time because it’s not as compatible with web based distribution. | Cost is similar to mac os and is has more user than mac but vulnerable to virus. | More difficult and time consuming but has potential to reach more users. |
| **Development Tools** | Xcode, Visual Studio Code, Atom, Sublime Text | Eclipse, IntelliJ IDEA, NetBeans, Sublime Text | Visual Studio, Visual Studio Code, Sublime Text | Android Studio (Java/Kotlin), Xcode (Swift/Objective-C) |

## Recommendations

1. **Operating Platform**: The recommended way to make the web based distributed solution that is compatible with all major operating platforms, including Linux, Mac, and Windows. This will allow the gaming room to expand the game app to a wide range of environments. But if only one platform is available we recommend Mac OS based on the mobility between smartphone and web version and security.
2. **Operating Systems Architectures**: Mac has Xcode that contains architectures and frameworks to support the game development. Mac also uses a hybrid architecture, combining features of monolithic and microkernel systems making it efficient for the clients need.
3. **Storage Management**: Mac users can leverage cloud storage solutions like iCloud to store and access data across devices. The choice of storage management system should consider the scalability and security with Mac's file systems.
4. **Memory Management**: Mac has a virtual memory management techniques, allowing efficient use of physical memory and optimizing memory allocation for running applications. Mac also provides tools and APIs for developers to manage memory effectively in their web based app.
5. **Distributed Systems and Networks**: Mac can communicate with other platforms using networking technologies like RESTful APIs. We can leverage these networking capabilities to build robust and scalable web based app.
6. **Security**: Mac prioritizes security and provides various security features to protect user information and ensure secure communication, it also includes built in encryption mechanisms. We can also use security frameworks and APIs to implement robust security measures in the games app.

# Project 2

## Server Side

Hosting a web application on the Mac offers developers an easy and smooth experience due to it IOS architecture, which is UNIX based operating system. UNIX is an operating system that makes web development easier for developers. And due to the recent new application of Mac chips, the power and performance plus the added benefit of IOS being mobile friendly, developer prefer the IOS ecosystem for web development. Furthermore, the advantages to using a Mac platform is its popularity, making it the second largest user base, which resulted in compatibility with many web application friendly platforms such as Apache, Nginx, ruby on rails, react etc; These platforms can be utilized on Linux and windows as well. A disadvantage of using Mac is the servers may be limited when it comes to web based distribution options compared to Linux; Back end development is more streamlined in Linux compared to windows and mac. The main disadvantage is the cost of using the IOS ecosystem.

Linux is very popular and most known for its cost-effectiveness, stability, security, and the preferred choice for server deployments/backend development; It also offers for web hosting and scaling just like macs IOS. The advantage of Linux is the wide range of server based deployment options such as Apache, Nginx, Ruby on rails and Node.js, which are well-suited for hosting web applications. Note that all platforms mentioned such as Node.js, is suitable across all platforms. The main advantage of utilizing Linux is the cost effectiveness and when it comes to managing the servers. Like Mac and PC, Linux is also a great platform to run and develop web-based application. The main disadvantage of Linux is that it may require a higher level of technical expertise for server configuration and administration, which could lead to higher cost, and that lack of cross platform compatibility to other platforms like mobile and windows. Linux is somewhat compatible to Mac.

Like mac and Linux, windows can utilize wide range of platforms like React, Node, Angular and Ruby on rails. Windows server is also widely used in enterprise environments and supports various web hosting solutions, including IIS. The main advantages include windows offers a user-friendly interface and is integrated well with Microsoft development tools and technologies, like Visual studios etc. The popularity makes it the largest platform, follows by Mac. Like Linux, windows are very efficient at backend structures. The disadvantages are licensing costs for Windows server can be higher compared to Linux, especially for larger-scale deployments, there is a lack of cross compatibility with mobile devices.

IOS mobile servers need to support communication with IOS devices running the game application, which typically involves using Apple's push notification service and other IOS specific APIs. The main Advantages of IOS servers can leverage the seamless integration with Apple's ecosystem, allowing efficient data organization between the server and IOS devices. The primary disadvantage of IOS mobile is developing and maintaining IOS servers may require specialized expertise in Apple development technologies. For android, the main advantages of android are the servers can support a wide range of android devices and versions, warranting broad compatibility with different mobile devices. The main disadvantage of android is the server development may require dealing with fragmentation challenges due to the diverse range of devices.

## Client Side

For the client, the experience across Mac, Linux and PC, may be very similar due to all three platforms hosting the frameworks used for web based development. There is more of a difference for the developer, choosing between Mac, Linux and PC, then there is for the client. Web browsers offer a standard interface for web applications, warranting compatibility across different platforms. Web based deployment allows seamless access to the game application from any device with a modern web browser but ensuring consistent user experience across various web browsers may require additional development and testing efforts.

There are two options for mobile platforms, IOS and android. Each have different development environments, programming languages they are better suited for such as swift/objective-C for IOS and Java/Kotlin for android. They also have application stores for distribution, Apple has the app store and android has Google play store. Supporting multiple mobile platforms allows the game to reach a broader audience, but development for multiple mobile platforms may increase the complexity, cost, and time required for the development process.

## Development Tools

Using Macs development tools, Xcode would be the primary development environment for Mac and iOS applications, supporting Swift and Objective-C languages. Xcode offers a very rich and diverse set of development tools and can seamlessly integrate with Apple's ecosystem, but licensing costs may apply for certain features in Xcode.

For Linux development tools, Linux supports a wide range of development tools and programming languages like Python, Node.js, and Java. Therefore, many Linux development tools are open source and freely available, reducing licensing costs. The primary disadvantage is the possibility of hiring multiple development teams to cover expertise in different programming languages.

Windows development tools include Visual Studio, which is the primary development environment for windows applications, and it supports languages like C# and VB.NET. Visual Studio also provides a comprehensive set of tools and strong integration with Microsoft technologies. Licensing costs for Visual Studio may apply, especially for enterprise editions.

In conclusion, each operating platform offers its unique advantages and challenges for hosting and supporting the web-based game application. The choice of platform will depend on factors such as cost, technical expertise, scalability, and target audience. It is essential to consider the requirements and constraints of each platform to ensure the successful deployment and functioning of the game application in a distributed environment.

# Project 3

## Operating Platform

As the client’s consultant, the operating OS should be a Mac OS, as the preferred platform. The goal of expanding, Draw It or Lose It, is made easier via Mac OS due to its popularity and easy translation from Mac OS to iOS. Specifically, one can utilize the Mac OS X Server to host the game server. It provides native integration with Apples ecosystem and its tools, like Xcode, this also goes hand in hand with the Unix based foundation that Apple and Mac OS shares. Mac also host various other tools that makes web development easy, such as JavaScript, CSS and frameworks associated with these tools. Furthermore, one can also utilize a cloud based server platform for hosting the game which will make the client side and the server side work seamlessly together. Cloud platforms like Amazon Web Services (AWS) or Microsoft Azure provide the ability and scalability needed to accommodate various environments and platforms. Cloud based server also offer virtual machines, databases, and networking, allowing easy expansion to different operating systems.

## Operating System Architectures

The software architecture chosen for the web game is client server architecture. Similarly, the architecture of the chosen platform, Mac OS, is composed of four main layers, the core, graphics, application, and interface. At the base is what’s called the Darwin core, which provides the fundamental OS services. Darwin is based on BSD Unix and features protected memory, automatic memory management, preemptive multitasking and advanced virtual memory for stability and security. Built on top of Darwin is the graphics subsystem with components for 2D, 3D and digital media to enable advanced graphics capabilities. The application layer supports running older Classic Mac applications via emulation as well as newer native apps built with Carbon and Cocoa APIs. Java is also supported at this layer. Finally, the last layer, Aqua user interface, provides the desktop environment with drop shadows, colors, textures and detailed icons for an excellent user experience. In summary, the layered Mac OS architecture combines the power of Unix with graphics, application support and user interface design that made the Mac experience great while enabling compatibility with older software.

## Storage Management

If one was to only utilize the Apple or Mac ecosystem, the HFS+ filesystem on Mac OS has evidence for reliable storage. A RAID, or, redundant array can be used along with time machine backups; Core Data or SQLite can be used for the database needs. Using external services within the Mac OS for storage means adopting a distributed file storage system like Amazon S3 or Azure Blob Storage as a means of cloud storage alternative. These cloud-based storage solutions provide reliable storage for game assets, images, and user data. They offer easy integration with various platforms and provide data redundancy to ensure data integrity, additionally it is a great substitute to iCloud storage options. The recommended storage option would be Amazon or Azure cloud storage due to its selections but iCloud offers more intuitive and integrative experience with Mac OS.

## Memory Management

Mac OS uses uncompressed memory pages and enhanced allocation to ensure efficiency. The virtual memory system shares physical memory between processes; This allows the game, Draw It or Lose It, to efficiently manage memory resources, allocate memory, and protect processes from unauthorized access. The operating system automatically handles memory paging, minimizing the risk of memory related crashes.

## Distributed Systems and Networks

Internal communication can be handled by distributed objects and remote messaging APIs provided by Mac OS. Networking can be done via TCP/IP sockets in the BSD layer, these sockets make it easier to communicate with Mac OS. Web sockets would enable real time multiplayer features. Alternatively, to achieve communication between various platforms, a RESTful API architecture can be implemented. This allows Draw It or Lose It to expose endpoints for communication over the network. The game's components can interact through API calls, enabling cross-platform compatibility.

## Security

Mac OS provides great security features that can help protect user information and ensure data privacy. To enhance security between platforms, implement secure communication protocols such as HTTPS and OAuth for user authentication. And utilizing encryption techniques for data transmission and storage. Regularly update the software and apply security patches to address potential vulnerabilities.

Mac OS has great built in firewall, encryption, secure boot capabilities, and access control lists for security, which negates the implementation of further security features.

## Work Cited for Project 3

Mac OS X Architecture and Terminology. (2023). etutorials. Retrieved August 13, 2023, from <https://etutorials.org/Mac+OS/using+mac+os+x+v10.3+panther/Part+I+Mac+OS+X+Exploring+the+Core/Chapter+1.+Mac+OS+X+Foundations/Mac+OS+X+Architecture+and+Terminology/>

Chapter 7. Sockets. (2023, April 8). freebsd. Retrieved August 13, 2023, from <https://docs.freebsd.org/en/books/developers-handbook/sockets/>