

<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 | 7/23/2023 | Shayla Vincent | Basic Overview of the Application Design |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

The gaming producer, *The Gaming Room,* is looking to develop a web-based game that will serve multiple platforms. This will help them reach more people and in turn increase revenue. The problem is that they will need help developing the environment to run their game. CTS will generate a java application for their use.

## 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.>*

* 2+ teams
  + Multiple players assigned
* Render image from stock
  + Render at steady rate
  + Complete at 30 sec mark
* 4 rounds
  + 1 minute each
  + If team does not guess before end the other team has 15 sec to answer
* Game
  + Names and game must be unique
    - Check whether name is in use
  + Only one instance of game in memory at a time
    - Create unique identifier for game instance, team, or player

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

*Technical constraints*

* Must be available to use on multiple platforms but remains as a web-based application.
* Only one instance of a game can be stored in memory at a time in order to reduce the load on the system.

*Business constraints*

* Make sure that there are no copyright infringement when selecting stock images
* Ensure there are no copyright problems because the game is loosely based on an old television game

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

First off, the diagram is contained within com.gamingroom. The relationship between ProgramDriver and SingletonTester is a directional association, the programDriver uses the SingletonTester.

The main chunk of the diagram is the other connections. Starting from the bottom left, Game service is what creates the instances of the game itself and is how id(s) are obtained. GameService is associated with a zero to many connection to Game. Game is where the list of teams are held add can add to that list. The game is also named here. Game is connected to team with another zero to many association. Team holds a list of players and can add to that list. The team is also named here. Team is then connected to Player by a zero to many association. This basically means that without a player there is no team and without a team there is no game.

Game, Team, and Player classes inherit from the Entity Class.

**"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** | <Evaluate Mac for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Linux for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Windows for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.> |
| **Development Tools** | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mac.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Linux.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Windows.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.> |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

Given the choice between Mac, Linux, and Microsoft, the best operating system for the *Draw it or Lose it* application would be Microsoft. Microsoft Azure, cloud computing, is one of the most versatile gaming development platforms. Microsoft will enable deployment to all major systems, such as: iOS, Windows, Linux, etc. They have many tools that make it more efficient to create an application, such as the Windows app SDK. Microsoft also has advanced security measures and relatively lower costs. According to Jay Chapel, Microsoft’s costs are as follows: Block bobs – $0.002 GB/month, Data Lake Storage - $0.001 GB/month, Managed Disks - $1.54 per month, and files $0.06/GB per month. [[1]](#footnote-1)

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

Microservices is the best architecture to use to develop a small application like *Draw it or Lose it.* This architecture type is “resilient, highly scalable, independently deployable, and able to evolve quickly”[[2]](#footnote-2). There cannot be a system wide failure because each service is decoupled from the other. Microsoft architecture can be broken into programs, processes, and threads. The program is the instructions to carry out. The thread is “what Windows schedules for execution within a process”. The thread has two stacks: kernel mode and user mode. The process contains: a program, at least one executable thread, virtual address space, etc.[[3]](#footnote-3)

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

The most appropriate storage management system that Windows offers would be Azure Blobs. This system enables the storage of images into a file system that is easily accessible through URL links. Blob storage is also easily integrated with other Microsoft development tools. There are also different pricing levels to choose from to best accommodate our needs. Plus, there are also always options to scale up.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

Like most operating systems, Microsoft employes the use of virtual machines to host many operations and applications. Hypervisors are the physical machines that allocate how much memory each VM can use. Hypervisors can reallocate memory based on demand. Microsoft has tools such as Azure advisor and Azure monitor, to monitor performance and identify potential issues in memory management.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

The application will need to communicate with different platforms so we need to consider how it will do so. Microsoft Azure functions enable developers to code in many different languages. APIs are how the application communicates with the client and each tool. Since we will be using a microservice architecture each blob will have to communicate with the others. We will also need to consider what communication tools to use. We will have the URLs for each image from the blobs, but we may also need to use Https to communicate with clients. The system will need to have back-up dependencies to reduce connectivity or outage issues. This can be accomplished with refresh buttons, breakers, authenticators and more.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

There are many layers of security that can be added to the application. At the most basic client-side level, we can require a password or additional verification procedures. For a web application like *Draw it or lose it* we will need a firewall and additional encryption to protect developer and user information. Our website should be hosted on HTTPS instead of HTTP to ensure security. For the back end we can utilize a shared access signature to prevent unauthorized users from accessing the applications data. Azure security center can provide an easy way to detect vulnerabilities and active threats. Microsoft also provides other tools such as Windows Defender for additional security.

1. Chapel, J. (2019, July 1). *Cloud storage cost comparison: AWS vs. Azure vs. google*. Medium. https://jaychapel.medium.com/cloud-storage-cost-comparison-aws-vs-azure-vs-google-844dfff3d324 [↑](#footnote-ref-1)
2. Martinekuan. (n.d.). *Microservices Architecture Design - Azure Architecture Center*. Azure Architecture Center | Microsoft Learn. https://learn.microsoft.com/en-us/azure/architecture/microservices/ [↑](#footnote-ref-2)
3. *Windows Architecture - The Basics*. TECHCOMMUNITY.MICROSOFT.COM. (2019, March 16). https://techcommunity.microsoft.com/t5/ask-the-performance-team/windows-architecture-the-basics/ba-p/372345 [↑](#footnote-ref-3)