Draw It or Lose It

CS 230 Project Software Design Template

Version 1.2

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	07/15/2022	Eric Slutz	Wrote the Executive Summary, Design Constraints, and Domain Model sections.
1.1	07/26/2022	Eric Slutz	Filled out the Evaluation table.
1.2	08/11/2022	Eric Slutz	Filled out the Recommendations section.

Executive Summary

The Gaming Room wants a web-based gaming application called Draw It or Lose It. It is loosely based on an 80s gameshow called *Win, Lose or Draw*. The team at The Gaming Room are not familiar with setting up the needed environment. The development team at CTS will help to simplify the development of the web-based version of the game application. Additionally, hardware requirements will not be assessed until after the software application decisions.

Design Constraints

- The application is written in Java
- The application is web based
- Only one instance of the game can exist in memory at any given time
- Ability for one or more team to play
- Multiple players on each team
- Game and team names must be unique
- The game consists of four rounds of play lasting one minute each
- Drawings fully render at a continuous rate for 30 seconds
- Images sourced from a large library of stock drawings
- The first team has those 30 seconds to guess correctly
- Otherwise, remaining teams each get 15 seconds to make one guess

The above-listed design constraints include the client requested software requirements. Furthermore, there are constraints dictated by the nature of the game and how it is supposed to be played. Lastly, given the web-based nature of the game application, those constraints need to be considered as well. The implications of these design constraints on the development process are that they help to set clear expectations for the application and how it is expected to work. The constraints will work to ensure that the client's needs are being met.

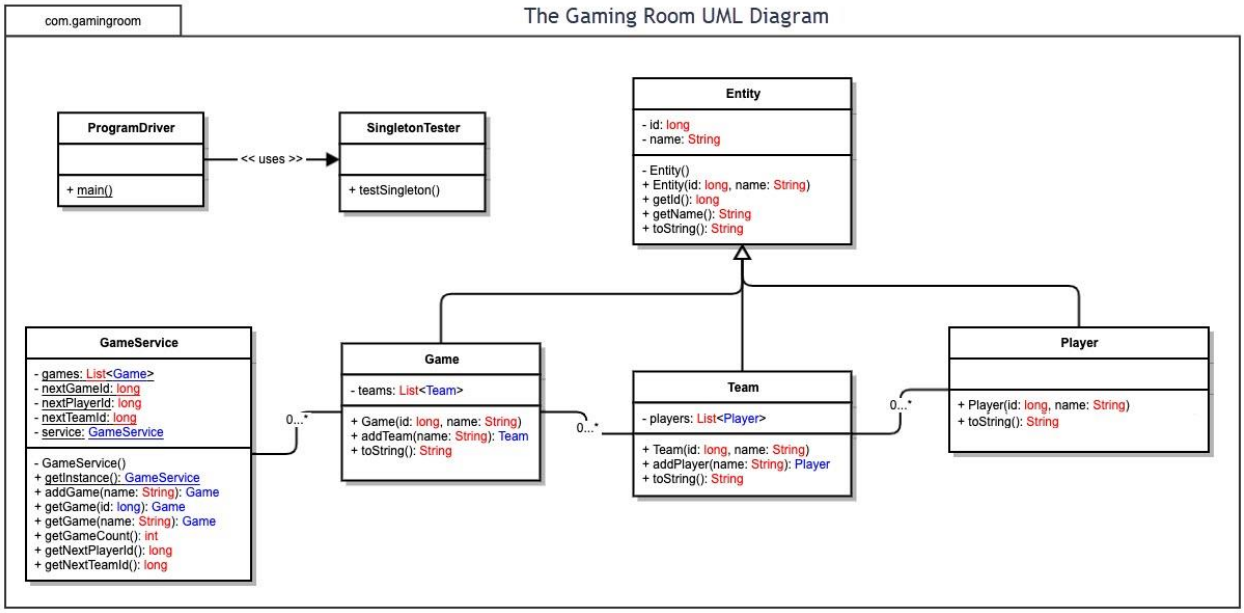
System Architecture View

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

The UML class diagram shows that the Game, Team, and Player class all inherit the Entity class; thus, showing the OOP principle of inheritance. Those four classes, along with the GameService class all exhibit encapsulation through using private attributes with public methods to access or modify them. Additionally, there are zero to many relationships between many of the classes. Such relationships exist between Team and Player, Game and Team, and GameService and Game. There are multiple examples of abstraction throughout the diagram as well. A few of them include addPlayer(), addTeam(), and addGame(). The process of adding these different

objects is hidden from the user; all they see is that the object has been added. Finally, the UML Class Diagram shows the relationship of the ProgramDriver class using the SingletonTester class.



Evaluation

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	• It is possible to use macOS as a server to host a web application. • You would be required to use Apple hardware which can be cost prohibitive. • There is no licensing cost for macOS (comes with the hardware). • Due to cost, would be difficult to scale up for more users.	• It is possible to use Linux as a server to host a web application. • There are many varieties of Linux available from general use to dedicated server versions. • Licensing fees vary from free to having yearly fees. • Overall, licensing can be much cheaper with this option. • Can use commonly available hardware.	• It is possible to use Windows as a server to host a web application. • A specific version of Windows, Windows Server, should be used as it is designed for this purpose. • Of all the options, this has the highest licensing fee for any version of the Windows OS. • Licensing costs could make scaling up cost prohibitive. • Can use commonly available hardware.	• It is technically possible to use a mobile device as a server and host a web application. • Cost of scaling up and the number of devices needed, reliability, and performance are just the tip of the reasons why you should not.

Development Requirements	Mac	Linux	Windows	Mobile Devices
Client Side	• The cost of developing a web app for this platform is shared with the other desktop OSs. • This, with the other desktop OSs, are the most cost effective because you get to cover 3 operating systems with one browser-based client. • Must ensure compatibility with major browsers. • Slight additional cost, skill, and time to include Safari (only available on Mac).	• The cost of developing a web app for this platform is shared with the other desktop OSs. • This, with the other desktop OSs, are the most cost effective because you get to cover 3 operating systems with one browser-based client. • Must ensure compatibility with major browsers.	• The cost of developing a web app for this platform is shared with the other desktop OSs. • This, with the other desktop OSs, are the most cost effective because you get to cover 3 operating systems with one browser-based client. • Must ensure compatibility with major browsers.	• Additional time to developer a client for mobile devices. • Different skills needed to develop client for mobile devices. • Within mobile devices there is further division between iOS and Android • Different skills needed between iOS and Android clients. • The different skills and additional time will increase cost to develop client.
Development Tools	• For server development macOS can handle any language that is used for web app development. • For client development macOS can handle any language that is used for web app development. • Most IDEs have versions compatible with macOS. • iOS apps are typically developed with Swift. • Xcode, VS Code, or AppCode are IDEs that can be used for iOS development. • Java or Kotlin are primarily used for Android development.	• For server development Linux can handle any language that is used for web app development. • For client development Linux can handle any language that is used for web app development. • The selection of commonly used IDEs for Linux is limited. • iOS apps are typically developed with Swift. • VS Code can be used for iOS development on Linux, but it is a limited experience. • Java or Kotlin are primarily used for	• For server development Windows can handle any language that is used for web app development. • For client development Windows can handle any language that is used for web app development. • Most IDEs have versions compatible with Windows. • iOS apps are typically developed with Swift. • VS Code can be used for iOS development on Windows, but it is a limited experience. • Java or Kotlin are primarily used for	• It is not recommended to do development on a mobile device. • Both screen size and computing power would limit the ability to do work. • While there is a browser-based version of VS Code and some other online IDEs they are still designed for a desktop experience. • A lot of the languages are not available for mobile, so if you write code, you will not be able to compile, build, or test the code.

	• Most major IDEs can be used for Android development.	Android development. • Android studio can be used for Android development.	Android development. • Most major IDEs can be used for Android development.	

Recommendations

1. **Operating Platform**: For the first part of the operating platform, the operating system, of the four options, the three desktop operating systems are the only ones of any real merit. Of those three, I would recommend Linux as the operating system of choice. Linux gives The Gaming Room the best return on their investment and is a well-supported operating system for use with web applications hosted on servers. For the second part of the operating platform, the hardware, I would recommend using a cloud provider, such as AWS or Azure, for the server hardware to run the operating system.

2. **Operating Systems Architectures**: The Linux operating system architecture is comprised of multiple elements that include the kernel, system libraries, and system utilities. The list below, from Silberschatz et al. (2008) describes those three components.

    1. **Kernel.** The kernel is responsible for maintaining all the important abstractions of the operating system, including such things as virtual memory and processes.
    2. **System libraries.** The system libraries define a standard set of functions through which applications can interact with the kernel. These functions implement much of the operating-system functionality that does not need the full privileges of kernel code.
    3. **System utilities.** The system utilities are programs that perform individual, specialized management tasks. Some system utilities may be invoked just once to initialize and configure some aspect of the system; others—known as daemons in UNIX terminology—may run permanently, handling such tasks as responding to incoming network connections, accepting logon requests from terminals, and updating log files.

3. **Storage Management**: The primary storage concern for the Draw It or Lose It game is storing the game images and storing game, team, and player information. None of that on its own would require much storage. The questions then become, how many different teams and players will there be, and will the game be expanded with more images? These storage issues can be addressed through going with a cloud provider, so you are only using the amount of storage that is needed, with a quick and effortless way to scale up or down the amount of storage being used. When looking at storage management options, for this game, you would want a RAID configuration, making data loss less likely. Another benefit of a RAID configuration is an increase in game performance through parallelism (Silberschatz et al., 2008).

4. **Memory Management**: The operating platform has two areas of memory management, physical memory, and virtual memory. The memory management techniques of paging and segmentation can be used for efficiently running the Draw It or Lose It software. This works by keeping only the needed parts of the program in the physical memory. Parts can be swapped back and forth between the physical memory and virtual memory as needed. This keeps the size requirements for physical memory low, while still allowing the program to run smoothly (Silberschatz et al., 2008). Also in this case, by minimizing the memory storage needed, it will allow for more instances of the game to be run at once than it would be able to otherwise.

5. **Distributed Systems and Networks**: Since Draw It or Lose It is designed as a web application, for the game to communicate between various platforms, access to the internet will be a requirement for the distributed system to work. Externally, by using RESTful APIs, the client, i.e., the user's browser or mobile app, will be able to easily make calls over a network connection to the various endpoints created for the game to play the game. This includes some form of authentication to verify that the user is set up and allowed to access a game instance. Internally, the various distributed systems in the cloud environment the application is hosted on will also use network connections to make the needed connections between the services. Typically, that would mean the backend running the RESTful endpoints would take the requests from the client, process the request, and make direct calls to the tools or service that is needed to complete the request.

6. **Security**: As always, security should be a primary concern to keep the application, and all its related data, including client information, restricted to those with correct access. From the client/user side, this part of the process involves authentication to verify which game instances the user has access to, what team they are a part of, and their own player information. On the distributed system side, since the various parts of the application are communicating over a network connection, authentication is needed between them to ensure proper access control of the systems and data contained therein. The Linux OS part of the operating platform has the required capabilities built into it to perform authentication activities. Additionally, all user data, both at rest and in transit, should be encrypted. Once again this is another capability that the operating platform can handle.

References

Silberschatz, A., Galvin, P. B., & Gagne, G. (2008). *Operating System Concepts* (8th ed.) [E-book]. Wiley. Retrieved August 10, 2022, from https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/