Draw It or Lose It

CS 230 Project Software Design Template

Version 3.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	07/15/21	Mark Webster	Edited Executive Summary, Design Constraints, Sys Architecture View, Domain Model, Evaluation, and Recommendation Sections
2.0	07/29/21	Mark Webster	Updated Design Constraints and Evaluations to implement feedback and new information
3.0	08/20/21	Mark Webster	Updated Recommendation Section

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

The company has currently released their game “Draw It or Lose It” on Android, and want to expand to a have a web-based version of the application. The game sketches an image for 30 seconds, and if a team has not successfully guessed by the end of that time, other teams will have the opportunity to guess for 15 seconds. Each game can have one or more teams involved; each team will have multiple players assigned to it; and game and team names must be unique so that players can check whether a name is in use when choosing a team name. Only one instance of the game can exists in memory at any one time.

Design Constraints

Different platforms may require different development kits and code may not be able to be reused between platforms. The main challenge is ensuring uniqueness of game, team, and player. Only one particular game can exist at any given time, so we have to set-up means of making sure that a game doesn't already exist. Furthermore, we have to make sure that teams and players are also not duplicated. A secure login process will need to be developed so that the system can authenticate each individual user, and the game itself will need means to identify a player as unique.

There will be a software component (the application) and a hardware component (the physical server or cloud-based solution) on which the platform will be hosted. The server-side component will hold a larger amount of images for the rendering process, so we will need to ensure a fairly large, non-volatile storage means, and well as an expedient way to send those images to the client-side application for rendering.

System Architecture View

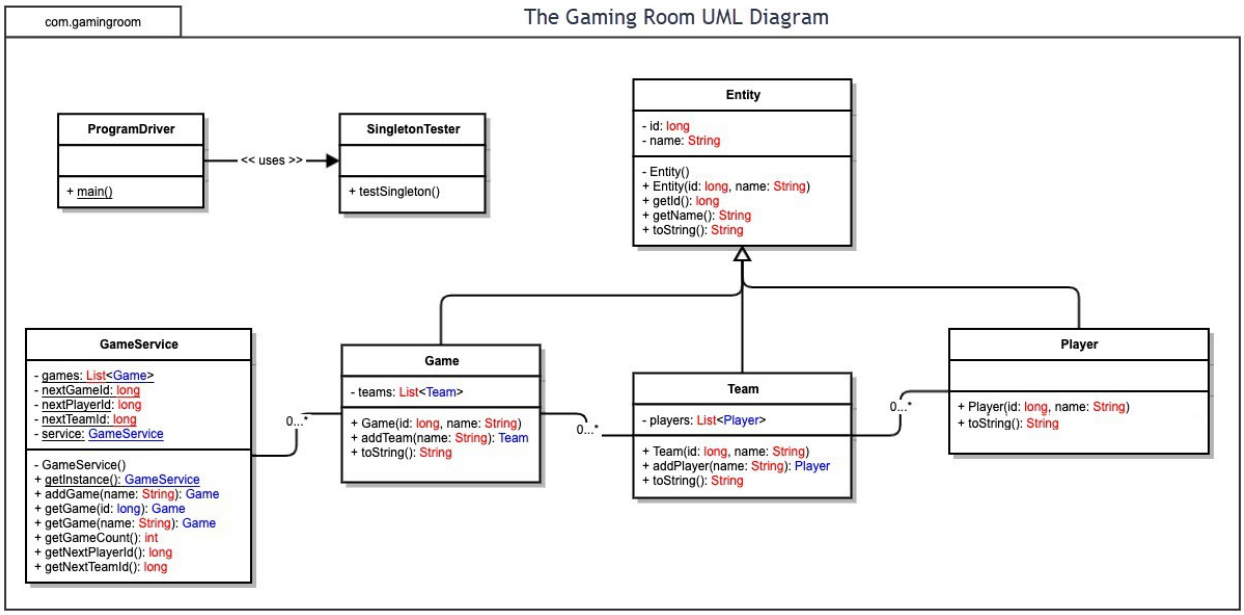
Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

Domain Model

The Entity class provides a basis which Game, Team, and Player then extend. Entity provides the basic id and name attributes and a basic constructor and getter functions. The Game class extends Entity and adds a teams ArrayList, a Game constructor and an addTeam feature. The Team class extends Entity and adds a player ArrayList, a Team constructor and an addPlayer feaure. Finally, the Player class extends Entity with a Player constructor.

Game Service is in a “has-a” relationship with the Game class, and as such keeps track of the instances of Game objects using a games ArrayList, providing iterator-based functions to ensure that Game instances are unique in name and id, as well as providing the next player and team ids when called.

The Program Driver runs the program and calls the Singleton Tester class to test the Singleton Pattern implementation in GameService.



Evaluation

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 is surprisingly adept at web-hosting as it has Apache baked into the platform. While it is more costly, it is the most secure of the options. While it is reliable, there are fewer hosts that offer Mac servers and most other platforms can run the software. If the company wants to set up their own server hardware, they would need to buy the server and the macOS server software, which costs $20. However, the company would have to shoulder all costs for their own servers, administrators, and support.	Linux is considered the go-to system for web-hosting, especially cloud solutions. Linux is open-source, however, which can make it a little less secure. In terms of cost, many flavors of Linux are free, but for large-scale systems which will need support, it is considered best to go with a corporate product like Red Hat Linux. An average cloud solution on the Red Hat platform costs about $1600.	Like Linux, Microsoft has been running server versions of Windows platforms for decades. They are more expensive than Linux servers , but Windows Server 2019 or Microsoft Azure make web-hosting very simple. Windows Server costs anywhere from $20 to $125 per month depending on which package is selected. Windows DataCenter 2019 is $6200 for licensing. The Gaming Room can deploy with either for fast, virtual server solutions which can be customized for our storage and throughput needs.	Hosting on mobile devices is tricky, especially given that different phone may be running different versions of the same OS, like variants of Anroid and IOS. Also, phones may be on Wi-Fi or mobile data, making web-hosting through mobile devices even trickier. While Mobile Web Server software exists, it would be tricky to build reliability over such a distributed system. Given the amount of memory and speed needed for the rendering library, the option is not a strong contender.
Client Side	Developing for Mac uses SwiftUI, and we would have to bring in people who know how to use that framework. Hiring new people will be costly monetarily, and the time cost as well as they are onboarded will be high. We would also have to use web devs familiar with Safari so that we can ensure compatibility. Thorough cross-browser testing of the web app is needed for each revision . While time-consuming , this ensures that functionality remains the same and all users have access to the same features.	We can develop the web-app in Linux using Python, which is a fairly common language. Python-based web-apps can also be run on most of the big name hosting services (AWS, Azure, etc) with little difficulty. Cross-browser testing will require web-dev knowledge for Linux browsers and devices. Python is widespread so finding engineers will not be a problem. The testing team will have to deploy and investigate each revision across all targeted platforms to ensure cross-functionality.	Windows is everywhere, and designing a web-based application for Edge, Chrome, Firefox, et cetera will be straightforward as Python, JS, and many more platforms can all run in VS Code. Windows supports a multitude of browsers and, given the established nature of the platform, finding developers should be straightforward. Strenuous cross-browser testing will be necessary with each change to make sure that implementation works for each platform feature. This will take more time, but will ensure consistency over all platforms.	Porting to IOS anytime in the near future will be tricky as it uses a different SDK than Android. Aside from a stand-alone downloadable app, there will be users who access the website-based application on mobile to consider as well.
Development Tools	Designing web-based apps in Mac is done in SwiftUI which is provided for free. For web development, many IDEs exist which can build on HTML, CSS, and JavaScript frameworks. We will need browser emulators for Safari on both macOS and iOS to ensure cross-browser success.	The number one method is using Python plus Flask, Django, or other frameworks. PyCharm or IDLE works well. VS Code by Microsoft is also designed for Linux.. Emulation for the Linux web browser is also needed, or a dedicated linux machine, to test the web application's cross-browser functionality.	'Visual Studio Code is the number one editor for Windows. It is available for free and numerous libraries and extensions are also available for free. Browser emulation software, as well as a variety of Windows devices , will be needed to test the web app in Edge, Chrome, Safari, Opera, and others.	Coding for IOS uses Swift; for Android, Java or Kotlin are widely used. We will need to ensure that the iOS version is consistent with the version that Safari users access, as well as the Android app , mobile Chrome users, and computer browsers.

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**: I recommend using Linux as the operating platform for server-side operations. Linux servers are by far the cheapest option, and can integrate into all of the hosting companies platforms, including AWS and Azure. Linux also easily allows cloud implementations, which will allow us to lower cost while increasing ability to handle traffic.

2. **Operating Systems Architectures**: Running Linux on our server-side means support for a wide variety of architectures, including x86 and ARM. We can also take advantage of Linux's wide native range of support for various cloud environments to help mitigate storage needs and bridge over distributed networked devices.

3. **Storage Management**: As Linux allows from easy implementation of cloud-based solutions, a large portion of storage will be in the cloud if we use AWS or Azure. Ideally, we should maintain an array of physical storage in the form of SSDs for their faster read/write times, lower energy usage, and performance-to-cost ratio. This physical array can serve as backup for

4. **Memory Management**: Both AWS and Azure pledge that servers will not suffer from any memory problems as memory allotment in the cloud will increase during heavy usage periods. Linux uses virtual memory and demand paging to manage memory, but ideally we want to avoid paging for faster access. We want enough cache memory for images; our cloud-based memory should be able to handle this without issue. Linux also supports a large number of various database framworks, including MongoDB and Postgres, so maintaining our databases of user information and images should not be problematic.

5. **Distributed Systems and Networks**: The means of communication between server and client will be primarily through RESTful APIs using the secure HTTP configurations. To that end, we can tailor the size of images sent to the devices the client is using based on platform (mobile or web), internet connection, and memory on the client device. Devices using mobile bandwidth or lower-speed connections can receive smaller file-size, more compressed images to save space. Linux allows use of network binding to link physical network interfaces together, creating redundancies in case of failure, as well as round robin support to ensure fair scheduling of access to resources. Linux also support orchestration tools like Docker and Kubernetes which allow users to manage containers which are on multiple host machines. Tapping into these tools will allow users on any devices to access and return information to our cloud servers with quick communication and scalability.

6. **Security**: Cross-platform apps require more attentive coding to ensure security, but they can be just as secure as native-apps. In addition to extra scrutiny during coding, regular updates will ensure we stay ahead of any possible exploits. Linux supports vendor security updates as well as a vast multitude of security scanning softwares Linux also has the widest support of encryption protocols, certificate authentications, and SSL protocols. SSL secures HTTPS communications and allows encrypted data to be sent from the server-side to our client devices.