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	11/16/2021	Elorha Newcomb	This revision presents technology requirements, including possible design constraints for the multi-platform game **Draw It or Lose It**, ordered by The Gaming Room.
2.0	11/26/2021	Elorha Newcomb	Upgraded evaluation table with more information regarding each platform option for the project development.
3.0	12/09/2021	Elorha Newcomb	Refactoring to include updated final recommendations with memory and storage management, systems and network, security, system architecture and operating platforms.

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 game **Draw It or Lose It** consists of a multi-platform game played in teams. The goal is to have each of the players guess the correct answers from image puzzles to accumulate points for their respective teams. We will pull the images from a large library database. The images will be shown to the participants in separate pieces, like a jigsaw puzzle. The full image will appear when 30 seconds have passed whether the team guessed it or not. There will be only one instance of the game at a time with a total of four rounds. At the moment, we do not have a limit set by the client for the total number of active teams and players in one game.

Design Constraints

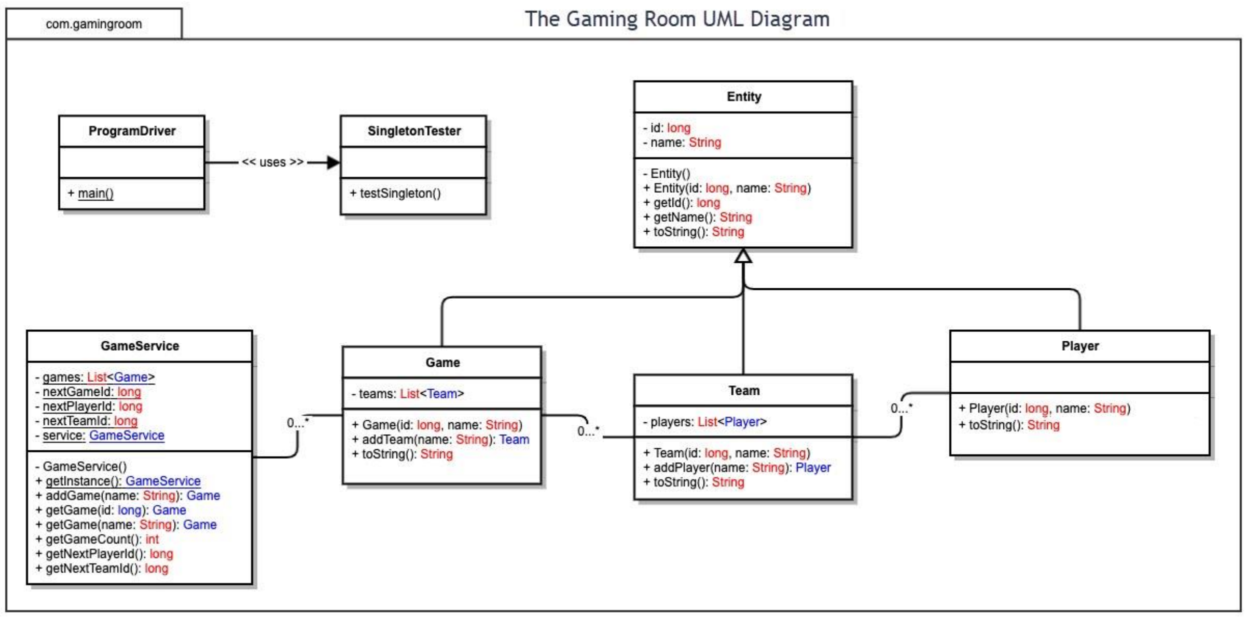
- We will require two teams for the development of this project. One team will work in the web-based application using JavaScript as the main language, and the other team is responsible for the mobile application, compatible with Android and iOS, and using React Native.
- Budget must account for possible delays, since the project is multi-platform and is being created from scratch.
- Scrum with sprints of two weeks duration and daily meetings will be held for the technical team. Time and division of feature-focused teams to be decided by CTO. Operations and the staff of **The Gaming Room** will be joining the pre-release meetings before product is submitted to quality assurance to align with the client's goals.
- The documentation for mobile and web-application will be created with the respective products, and will be updated frequently by the group leaders of the technical teams.
- Database created must account for image-heavy usage and quickly process calls to the server, since games need a fast response from the users.

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 was extremely helpful to the team while designing the server foundation. The main classes have an established relationship with the superclass *Entity*, which allows them to inherit the attributes **name** and **id.** This way we ensure that there will be no duplicates for any of the elements in our *GameService.* The *ProgramDriver* uses *SingletonTester* to ensure that the correct information is being sent to the server and that there are no duplicates being created when more instances of the game are created. The *Player* class will be called in *Team* in order to separate players in different teams. The *Game* class will aggregate the previous classes and will generate teams with the available players in the database. The *GameService* class relationship with *Games* will aggregate the teams that were generated in *Games* and will create new games and allocate the resources from *Team* and *Player* accordingly. Once a game is generated, it cannot exist somewhere else. Every game is unique to the pertaining team and players in the current instance.



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	• Built-in server that speed up the process of installation. There's the option to set up the server without the program, running everything through the terminal. • Many features of Mac OS server are now disabled, due to its decline in usage by users. • MacOS Server costs $20 a month. • Used in local networks to ensure that all computers are up to date and backed-up; thus, a small to medium-scale tool. • You can rent a macOS server in the cloud (Mac in the cloud) for around $25 per month without having to acquire the hardware. • Not as used for web hosting as Linux and Windows. • Intuitive Graphical User Interface (GUI) to allow easy interaction with the computer without the command line.	• Linux is the most secure operating system currently, and is more reliable than Windows for the server side. • Open-source and many libraries and distributions can be found online without any cost. • Modular and open-source architecture allow for an application-centric view of the database rather than a network-focused view. • Integrates much better with Microsoft products than Apple. It is harder to find web hosting support compared to Windows. • To enable a GUI in your Linux server, you need to install it separately. • The user will have to access the terminal more frequently than the other platform options.	• Windows is the most used operating system in the world. • Windows servers are popular and this ensures that more products and frameworks, like ASP and ASP.Net are available for the system compared to the others. • Windows contains more features and tools for server creation. • More susceptible to viruses, hacking and system errors. • Since Microsoft Server is a paid license, there is also 24/7 support from the company in case something is not functioning how it should. • The user is limited to using only Microsoft products and web framework, which can delay the production process. • Intuitive Graphical User Interface (GUI) to allow easy interaction with the computer without the command line. • Comes with a built-in remote desktop that can process all the information without requiring the user to know how to use the command line.	• Server apps are not as popular as the other options but have a much higher reach, considering that many users prefer to use apps on mobile devices. • A thin client is installed in the device and manipulates the data from the server remotely, keeping the information safe. • Native apps cannot be exported to other operating systems and would require separate versions of the game for mobile and desktop. • Scalability is easier since the costs of local hardware do not apply. • Most development tools are compatible with mobile systems currently. • Having the server-side handling the database separately from the layout facilitates updating and adding information.

Client Side	• Production cost is high and requires enough expertise to operate the terminal. • If the team already develop products using Mac computers, they will not require extra equipment for the job • The layout could be handled by a ready-to-use template in JavaScript, and there are plenty of layouts to choose from. • If the database becomes too big, rendering information will become considerably slower. • Data leaks are more prone to happen since there is only one side handling the totality of the application, with a lot of user information being stored by the client.	• Production cost is much lower than the others because it's opensource, but requires high expertise to operate the terminal. • Open-source and many libraries and distributions can be found online without any cost. • The layout could be handled by a ready-to-use template in JavaScript, and there are plenty of layouts to choose from. • If the database becomes too big, rendering information will become considerably slower. • Data leaks are more prone to happen since there is only one side handling the totality of the application, with a lot of user information being stored by the client. • Parameters and other aspects of the server can be adjusted without the need for the whole website to be unavailable for users.	• Windows production cost is high but the usability is much easier than the other UNIX-based systems. • The layout could be handled by a ready-to-use template in JavaScript, and there are plenty of layouts to choose from. • If the database becomes too big, rendering information will become considerably slower. • Data leaks are more prone to happen since there is only one side handling the totality of the application, with a lot of user information being stored by the client. • Deployment is easy and comes with built-in security. It allows deployment even over firewall to any servers around the world. • No contact between client and server besides the HTTP and HTTPS requests. If client is down, the server will not know. It could even deem it	• Easier to develop and there are many free tools online that help export the apps to mobile stores. • Great tool to be used with advertisement, because it provides a more dynamic user experience. • Handling the whole application just on the client side would make the development much faster since everything would be contained within the mobile application. • Possibility of creating a user version and an advertisement version for business clients that want to invest in the app be able to control their own advertisement campaigns. • The layout could be handled by a ready-to-use template in JavaScript, and there are plenty of layouts to choose from. • Uses the GPU of the computer (Graphics processing unit) instead of the CPU. That optimizes system for use in multiple platforms. • Data leaks are more prone to happen since there is only one side handling the totality of the application, with a

		• Compressing JavaScript into one document is fast and can be done from the command line.	inactive and eventually remove all the information pertaining client from database.	lot of user information being stored within the application.

Development Tools	• Web-based written in JavaScript. • HTML and CSS where React component styles are not applicable. • Changes in real-time available in any browser. • Frameworks: React for client side and Next.js for server side. We chose Next.js instead of Express.js because it comes ready for development. • We evaluated Remix.js as well, but since it is a very recent tool, Next.js is more stable. • There's no limitation to only one platform and production can be done anywhere from any desktop OS in web-based applications. • Visual Studio Code is the preferred tool for code editing in the production phase. • No need to install the application locally to play and user can play with their friends from anywhere with an online invitation.	• Web-based written in JavaScript. • HTML and CSS where React component styles are not applicable. • Changes in real-time available in any browser. • Frameworks: React for client side and Next.js for server side. We chose Next.js instead of Express.js because it comes ready for development. • We evaluated Remix.js as well, but since it is a very recent tool, Next.js is more stable. • There's no limitation to only one platform and production can be done anywhere from any desktop OS in web-based applications. • Visual Studio Code is the preferred tool for code editing in the production phase. • No need to install the application locally to play and user can play with their friends from anywhere with an online invitation.	• Web-based written in JavaScript. • HTML and CSS where React component styles are not applicable. • Changes in real-time available in any browser. • Frameworks: React for client side and Next.js for server side. We chose Next.js instead of Express.js because it comes ready for development. • We evaluated Remix.js as well, but since it is a very recent tool, Next.js is more stable. • There's no limitation to only one platform and production can be done anywhere from any desktop OS in web-based applications. • Visual Studio Code is the preferred tool for code editing in the production phase. • No need to install the application locally to play and user can play with their friends from anywhere with an online invitation.	• React Native will be adopted for the mobile developments. React native can be used in any OS and is a great tool to save time and money in the production for mobile platforms. • React Native supports both iOS and Android and almost the entirety of the code can be reused. • Changes can be done faster and there is no need to rewrite the whole application when adding new components in the future. • Most of the code for the web application made in JavaScript / React will be reused in the mobile, like the CSS stylesheets and styled-components, in order to maintain similar layout cross-platform. • The performance is not affected by the fact that it is a cross-platform application and it optimizes for 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**:

    Even though the cost is usually higher on Macs, we have performed a productivity evaluation, and chose Mac as our major development OS for the web application and the mobile app.

    iOS vs Windows: The team will work on MacOS during development because of its stability and built-in terminal that is preferred over Git for BASH emulation on Windows OS. Brew also makes installations easier on Macs compared to Windows.

    iOS vs Linux: Better tools for testing cross-platform solutions, which will speed up the process for the mobile application. Because of the popularity of iPhone, Safari is one of the most used browsers and is natively on Macs, facilitating testing of the web application for multiple browsers. Not all devices are responsive with Linux and the team would have to purchase laptops for the engineers that do not have a compatible machine. Also, most users of the game will not be using Linux natively, since it comes as the fifth option for consumers, according to data provided by WGU (2021), after Windows, MacOS, Android, and iOS.

2. **Operating Systems Architectures**:

    When we talk about the architecture of the Mac OS, we are talking about multiprocessor systems. The processors communicate and share resources with each other, like the clock, memory, and peripheral devices. Nowadays it is impossible to think of operating systems with a single core unit. This is due to all the advantages that the multiprocessors brought to their evolution. Mac uses symmetric multiprocessing (SMP) where each processor performs all tasks within the operating system. SMP means that all processors are peers; no master-slave relationship exists between processors (Silberschatz et al., p. 13). A multiprocessor system of this form will allow processes and resources—such as memory—to be shared dynamically among the various processors and can lower the variance among the processors. This brings a series of advantages to the table. According to Silberschatz et al., they are:

    Increased throughput: if there are more processors, tasks take less time to be completed and the entirety of the system benefits from that.

    Economy of scale: multiprocessor systems can cost less, because they share peripherals, mass storage, and power supplies. It is cheaper to store those data on one disk and share them among the processors than to have many computers with local disks and many copies of the data.

    Increased reliability: when tasks are distributed among processors, one failure will not halt the system completely, but it might slow it down. “Providing service proportional to the level of surviving hardware is called graceful degradation. Some systems go beyond graceful degradation and are called fault tolerant, because they can suffer a failure of any single component and still continue operation". (Silberschatz et al., p. 14)

3. **Storage Management**:

The server side will use a hybrid model of local database pointing to the files in the cloud. For the image database, we will store the image files in AWS S3. They will be contained in a bucket and will be selected at random by the server. The reason the client is not used in this case is because randomizing files in the folder would end up showing different files to each person in the gaming room. The images won't rely on the server's OS RAM loading time for the images, since they will all be stored remotely, and there will be no need to increase hardware capacity and purchase more licenses for the current available OS's, in case the image bank becomes too big. Buying more space in the cloud is effective and does not require refactoring of the server or the database. The mobile application will use the same principle, so it will be light and responsive, as mobile games should be.

4. **Memory Management**:

Since the first mock-up application was made in Java, we will discuss memory management in this language, but the concept of garbage collection is true for all high-level languages that do not require direct memory management by the developer, like C and C++. Working with a high-level language like Java facilitates memory management, because the garbage collecting is dealt automatically. In Java, it is handled by the JVM, or Java Virtual Machine. This managed runtime environment (Tyson, 2020) optimizes memory and establishes compatibility between different platforms in place of the operational system of the physical machine. The garbage collection clears unused objects from the heap space to allocate newly created objects, while it separates elements in order of survivability. In each runtime, JVM will move elements to one of the heap spaces "Young Generation, Old or Tenured Generation, and Permanent Generation" (Oracle, 2021). The young generation contains objects that are used just once and are then discarded, and elements that can survive over one cycle. The survivors will be categorized in the older generation, as long as they still exist in the space. If they are immutable elements pertinent to JVM, they will belong to the permanent generation. The more an object survives, the longer it takes for the process to be completed. This can affect runtime for a responsive application like Draw It or Lose It, where games are time-dependent. This problem would likely only happen at a global scale, but at that point, multiple regional servers (cloud servers in this matter) could help mitigate lag in the server-client communications.

5. **Distributed Systems and Networks**:

We will combine Java and a NoSQL database, Mongo DB, to store an index of the files from AWS. Instead of hard coding the paths on the code, there will be an element

"Path" at the team schema (using JSON protocol) to pick a random file in the AWS image folder and show it by using a "GET" HTTP request. Game encapsulates team, and team encapsulates player in the database. There's one instance of the game, two teams, and multiple players. Each team will be composed of half of the total amount of players (A will have +1 in case of odd numbers). This way, the same image will be handled by the server and sent to the client in order to show the provided image to all members of only one group.

There will be different server-points for the operation to avoid full-outages in case one of the server locations goes down (either due to network traffic or natural disasters and occurrences). The updates in the two versions will always follow the standard guidelines for dealing with layout, branding, and in-app advertising from marketing partners of **The Gaming Room**.

6. **Security**:

For the files in Amazon S3 storage, called buckets, the least-privilege principle is the best resource to control unauthorized access. "Identity and Access Management (IAM) directly enables fine-grained access controls" (Pandey, 2020). The main reason for data breaches in S3 buckets, as noted by Pandey in his article, is misconfiguration that can happen when inexperienced users accidentally set access to "Everyone" to allow users to read and write in every available file (Pandey, 2020). In AWS, only the profile that originally created the account has control over all access permissions and can set the access control lists to private, everyone, or to a case-by-case basis. This account in **The Gaming Room** will be the designated super admin and will be used only to configure permissions. It will be protected under a 2-factor authentication wall and only managed by the CTO for extra security. There will also be server-side encryption in the images that are retrieved from S3 during server requests.

Since our SQL database will be disconnected from the actual storing of the images on S3, the tampering of files from hacking will encounter multiple barriers of security. For the server security, we will run Apache on Linux Ubuntu, because of its virtual hosting capability, facilitating testing for the development team in charge. It is important to limit the amount of unnecessary ports open in the Linux server besides the HTTP and HTTPS, to make sure the firewall and the server antivirus are up and running, and to always keep it up-to-date and backed-up (Kumar, 2020). These are the bare minimum needs of a server for a web application. Carrying out security audits and vulnerability updates from time to time is also a great way to ensure that something will not strike the server and take it down by surprise. Disabling the root login, enforcing regular password rotation and enabling 2-factor authentication (Avast, 2021) are advanced options to keep the hardware of the local server from being accessed by unauthorized users.

User passwords will be hashed to ensure there will be no damage to the users, should any data breach happen to the application in the future. Administrators will be assigned specific roles and have special ranked access (super admin, admin, game master, support rep) in the same least-privilege principle as S3 for file management and other activities in the app. In case any personal data of a role leaks, that role can be suspended with no damage to the others and another one can replace the compromised profile. Users will have a simple role that will allow them to form teams, manage their own profiles and play the game.

References

Avast Business Team. (n.d.). How to secure your Linux server. Avast. Retrieved December 12, 2021, from https://blog.avast.com/secure-your-linux-server-avast.

AWS. (n.d.). Identity and access management in Amazon S3 - Amazon ... Retrieved December 11, 2021, from https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-access-control.html.

Kumar, C. (2020, February 29). 8 essential tips to secure web application server. Geekflare. Retrieved December 11, 2021, from https://geekflare.com/secure-web-application-server/.

Oracle. (2010, January 27). Understanding Memory Management. Oracle. Retrieved December 2, 2021, from https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html.

Oracle. (n.d.). Java Garbage Collection Basics. Retrieved December 3, 2021, from https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html.

Pandey, K. (2020, January 28). How to secure S3 buckets effectively. Medium. Retrieved December 10, 2021, from https://medium.com/panther-labs/how-to-secure-s3-buckets-effectively-9c1a3a7178bb.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2008). Operating system concepts. Hoboken, NJ: Wiley.

Tyson, M. B. M., & Tyson, M. (2020, January 17). What is the JVM? introducing the Java Virtual Machine. InfoWorld. Retrieved December 4, 2021, from https://www.infoworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html.