<Name-of-Software-Application>

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	11/12/23	Andrew Torrez	Initial software design requirements
2.0	12/1//23	Andrew Torrez	Updated server/client/development tool analyses
3.0	12/10/23	Andrew Torrez	Included recommendations for the client

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

For the software design of "Draw It or Lose It", we propose transforming an existing Android game into a versatile web-based application, catering to multiple platforms. This development aims to preserve the essence of the game while enhancing accessibility and user experience. The project focuses on unique challenges such as ensuring unique identifiers for games, teams, and players, and implementing efficient software design patterns to manage a single instance in memory and real-time name availability checking. This approach is aimed at delivering a scalable, user-friendly, and robust gaming experience that aligns with The Gaming Room's vision and technical requirements.

Requirements

Creative Technology Solutions (CTS) is tasked with transforming The Gaming Room's "Draw It or Lose It" from an Android app into a web-based game accessible across multiple platforms. Key business requirements include ensuring multi-platform accessibility, maintaining user engagement similar to the original game, implementing a scalable monetization strategy, and accommodating a growing user base. Technically, the game must support multiple teams with unique identifiers, ensure a single instance in memory, and provide real-time name availability checks for games and teams. The development will focus on cross-platform compatibility, encompassing responsive design for various devices and operating systems. Backend development will handle game states and data management, while the frontend ensures a user-friendly interface. Database management, stable networking, and robust security measures are also crucial to protect user data and ensure smooth gameplay. This concise approach aims to meet The Gaming Room's goals while offering an engaging and secure gaming experience.

Design Constraints

Web-Based Distributed Environment:

- **Browser Compatibility:** The game must function seamlessly across various web browsers, considering different JavaScript engines, CSS interpretations, and HTML rendering.
- **Network Dependencies and Latency:** Web applications depend on network stability and speed. Latency can affect game performance, so efficient network usage and minimal latency are crucial.
- **Security Considerations:** Protecting user data and game integrity in an online environment is critical. This includes securing communications, data storage, and handling user authentication.

Single Instance Restriction:

- **Memory Management:** Only one instance of the game can exist in memory at any given time, necessitating careful memory management and garbage collection strategies.
- **Data Integrity:** Ensuring that the single instance maintains data integrity, especially when accessed by multiple users simultaneously.

Unique Naming for Games and Teams:

- **Real-Time Name Availability Checking:** Implementing a system to check and confirm the uniqueness of game and team names in real-time, without causing significant delays or server load.

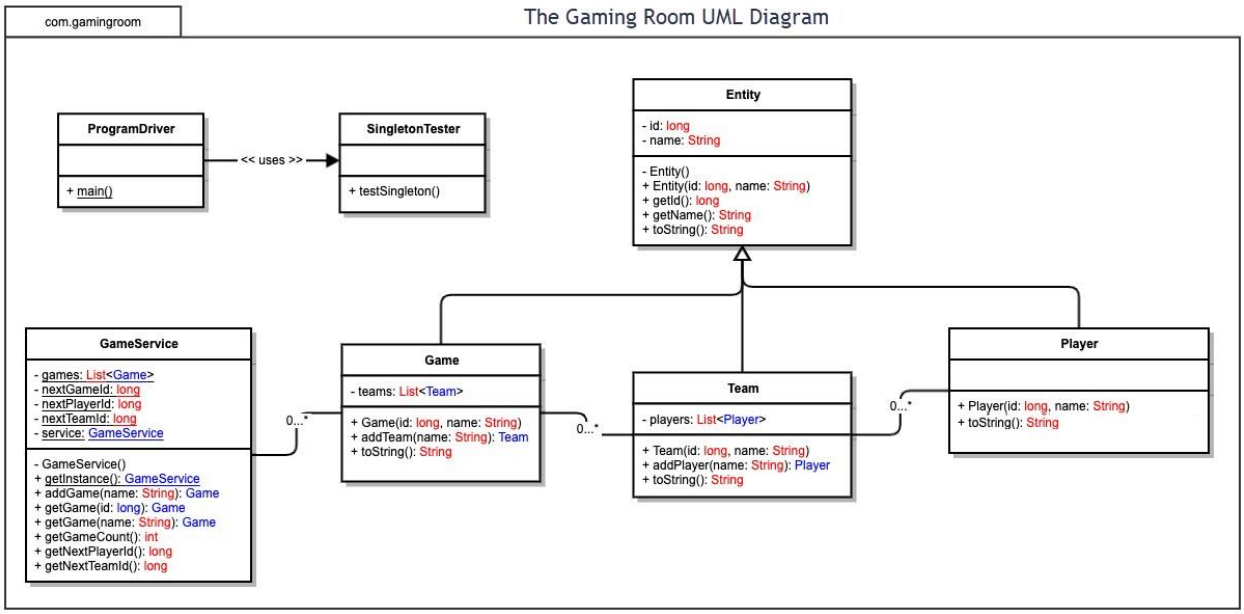
- **Database Design and Scalability:** Designing a database that can efficiently handle the uniqueness constraint and scale as the number of users grows.

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

In the domain model of our software design document, the Unified Modeling Language (UML) diagram is key. It outlines the structure of "Draw It or Lose It," with the Entity class linking to Game, Team, and Player classes, demonstrating inheritance for code efficiency. The diagram highlights the ProgramDriver Class's use of SingletonTester, ensuring only one game instance exists in memory. The GameService class, as the game's backbone, contains critical methods for functionality and supports unique identification for games, teams, and players. This UML diagram serves as a flexible blueprint for development, ensuring all key requirements of The Gaming Room are met efficiently.



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	For hosting a web-based software application on Mac, key characteristics include a stable and secure environment, with an intuitive user interface, making it user-friendly for administrators. Advantages are its reliability and integration with Apple's ecosystem. However, it has weaknesses like higher costs due to licensing and hardware requirements, and limited flexibility and software compatibility compared to other server platforms.	Linux is an open-source platform offering customization and a wide range of support for web technologies, making it a popular choice for server environments. Its advantages include no licensing costs, robust security features, and scalability to handle high traffic loads. The main weaknesses are the need for more technical expertise for management and potential compatibility issues with non-Linux software.	Windows servers are known for their ease of use and integration with Microsoft's ecosystem. The advantages include widespread support, familiarity among administrators, and compatibility with a broad range of software. However, the downsides are higher licensing costs and a greater vulnerability to security threats compared to platforms like Linux.	Mobile devices typically don't function as traditional servers but can support specific, lightweight server tasks. They offer advantages in decentralized, small-scale applications, enhancing accessibility and mobility. However, they are limited by hardware capabilities and are not suitable for large-scale server demands.

Client Side	Developing for Mac clients requires attention to compatibility with macOS-specific features and browsers like Safari. It involves higher costs due to specific hardware and software requirements, time-intensive testing for macOS configurations, and necessitates expertise in macOS-specific development tools and design principles.	Development for Linux clients focuses on compatibility across various Linux distributions and browsers. It has lower costs due to free development tools, but additional time is needed for testing across diverse distributions. Expertise in open-source technologies and the Linux environment is crucial.	For Windows clients, the development must ensure compatibility with different Windows versions and browsers. It involves moderate costs, given the availability of resources and tools, requires thorough testing across Windows versions, and demands knowledge of Windows-specific development environments.	Mobile development should prioritize responsive design and touch interface optimization, considering various screen sizes and operating systems. It involves higher costs and time due to extensive testing and optimization for diverse devices and requires expertise in mobile development frameworks and UX/UI design principles.
Development Tools	For Mac development, common programming languages include Swift, Objective-C, and JavaScript for web applications. Essential tools are Xcode, Atom, and Visual Studio Code, catering to the unique development environment of macOS.	Linux development commonly employs programming languages like Python, Java, PHP, and JavaScript. Tools like Eclipse, GNOME Builder, Sublime Text, and Visual Studio Code are widely used, leveraging the open-source nature of Linux.	For Windows, popular programming languages include C#, .NET, JavaScript, and Python. Development tools such as Visual Studio, Atom, Sublime Text, and Eclipse are essential for creating applications compatible with the Windows environment.	Mobile development utilizes languages like Swift for iOS, Kotlin for Android, and JavaScript for web-based applications. Tools include Xcode for iOS, Android Studio for Android, and cross-platform frameworks like React Native and Flutter to cater to the diverse mobile ecosystem.

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**: For expanding "Draw It or Lose It" to other computing environments, I recommend using a **Linux-based server platform**. Linux offers flexibility, robustness, and is widely supported, making it ideal for hosting web-based applications. Its compatibility with various hardware and scalability makes it suitable for a game that needs to expand and evolve.

2. **Operating Systems Architectures**: Linux is an open-source operating system known for its modular architecture. It comprises a kernel that manages core functionalities like networking, process management, and file systems, while the user space provides a wide range of applications and utilities. Linux's modular design allows for customization and optimization specific to the needs of "Draw It or Lose It," ensuring efficient performance and scalability.

3. **Storage Management**: For storage management with the Linux platform, utilizing a combination of **filesystems like Ext4** (for general storage needs) and **database systems like MySQL or MongoDB** (for handling game data) is recommended. Ext4 offers reliability and good performance for general file storage, while the chosen database system will efficiently manage game-related data, ensuring quick access and data integrity.

4. **Memory Management**: Linux uses advanced memory management techniques, including virtual memory with paging and swapping. This allows efficient allocation of memory to different processes, crucial for a game application like "Draw It or Lose It" which may have varying memory requirements. Linux's memory management ensures that the game runs smoothly, even in high-demand situations.

5. **Distributed Systems and Networks**: To enable "Draw It or Lose It" to communicate across various platforms, we can use distributed software architecture. This would involve deploying the game's backend on a centralized server (Linux-based) while allowing different client devices to connect and interact with it. We'll use web technologies (like HTML5, CSS3, and JavaScript) for the client-side, ensuring compatibility across devices. For the network, employing RESTful APIs over HTTPS ensures secure and efficient communication between the clients and the server. Attention will be needed to handle dependencies, such as network connectivity and handling outages with fallback mechanisms and redundancy.

6. **Security**: Securing user information on various platforms is paramount. On the Linux server, we can implement firewalls, use SSH for secure access, and configure SELinux for advanced access control. Data encryption (both at rest and in transit) and regular security audits are crucial. For client devices, ensuring that the application adheres to the security standards

each platform (iOS, Android, Windows, etc.) is essential. Implementing features like OAuth for secure authentication and TFA (Two-Factor Authentication) enhances user protection across platforms.