

<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 | <10/10/2023> | <Richard Pashko> | <Module seven project.> |

**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 Gaming Room has tasked Creative Technology Solutions (CTS) with developing a web-based game, "Draw It or Lose It," based on their existing Android app. The challenge lies in streamlining the development process and meeting the client's software requirements effectively. These requirements include the ability to have multiple teams with multiple players, ensuring unique game and team names, and managing unique identifiers to prevent multiple instances of the game in memory. To address these issues, CTS will design a robust software solution that encompasses team management, name validation, and instance control, ensuring a seamless gaming experience for The Gaming Room's users.

**Requirements**

The client, The Gaming Room, seeks to expand its Android app-based game, "Draw It or Lose It," into a web-based multi-platform version. The game involves multiple teams with several players per team and emphasizes unique names for both games and teams to prevent naming conflicts. A crucial technical requirement is the development of cross-platform compatibility for seamless gameplay across various devices. Additionally, the system must manage teams effectively, validate and ensure the uniqueness of game and team names, and establish unique identifiers to prevent multiple instances of the game running simultaneously. These requirements form the foundation for the development of the web-based version of "Draw It or Lose It."

**Design Constraints**

Developing the game application "Draw It or Lose It" in a web-based distributed environment requires addressing several design constraints. Cross-platform compatibility is essential to ensure the game runs smoothly on different devices and browsers. Scalability is crucial to accommodate multiple teams and players, while data integrity and security measures are necessary to protect user data and ensure a seamless gaming experience.

**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**

<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.>

ProgramDriver uses the SingletonTester. GameService is a singleton class. GameService uses Game zero or more times. Game uses Team zero or more times. Team uses Player zero or more times. All three (Game, Team and Player) are child classes of the parent class Entity.



**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** | MacOS offers a stable platform with a user-friendly interface for hosting web-based software applications, but it may have limited scalability and support compared to dedicated server operating systems like Linux. | Linux has an open-source nature, customization capabilities, and security features. It is a strong choice for hosting web-based software applications, providing advantages like a diverse selection of server software, strong performance, and cost-efficiency. However, Linux may have a steeper learning curve for administrators, and compatibility with some proprietary software could be limited. | Windows has a user-friendly interface and robust support for .NET technologies, offers characteristics that make it suitable for hosting web-based software applications. However, it may have higher licensing costs and be more vulnerable to security threats compared to open-source alternatives like Linux. | Mobile devices have characteristics of portability, diverse hardware, and touch interfaces which make them suitable for accessing web-based software applications. This offers advantages like accessibility, on-the-go usage, and a broad user base. However, they have limitations in terms of processing power, screen size, and resource constraints, which can impact the complexity and performance of hosted applications. |
| **Client Side** | Supporting multiple client types on Mac can have relatively low development costs due to free development tools but may require investments in Mac hardware. However, it can increase development time and expertise requirements because of platform-specific development and testing needs. | Supporting multiple client types on Linux can be cost-effective due to open-source development tools but may extend development time because of the need to account for various distributions and desktop environments. Expertise requirements can vary based on the complexity of cross-platform compatibility. | Supporting multiple client types on Windows can have moderate development costs due to licensing requirements for some development tools, and it may involve additional time for testing and compatibility with various Windows versions, while expertise requirements will be lowered due to the platform's popularity. | Supporting multiple client types on mobile devices typically involves moderate to high development costs due to the need for platform-specific tools and testing across various devices. Development time is extended, especially when targeting both iOS and Android. Expertise requirements encompass platform-specific knowledge and potentially cross-platform development frameworks. |
| **Development Tools** | To build software for deployment on Mac, developers commonly use programming languages such as Swift, Objective-C, and C/C++, along with Apple's official integrated development environment Xcode. Additional tools like Git may also be employed to facilitate the development process. | To build software for deployment on Linux, developers often use programming languages like C, C++, Python, and JavaScript, with code editors like Visual Studio Code, Sublime Text, or Vim, coupled with GNU Compiler Collection (GCC) or other compilers. Package managers like APT, YUM, or Snapcraft are used to distribute software, and tools like Docker assist with containerization and deployment on various Linux distributions. | To build software for deployment on Windows, developers frequently use programming languages such as C#, C++, and JavaScript, along with integrated development environments (IDEs) like Visual Studio and Visual Studio Code. Other relevant tools include .NET Framework or .NET Core for application development, Windows Presentation Foundation (WPF) for desktop applications, and Windows Installer (MSI) for packaging and distribution. | For developing software for deployment on mobile devices, developers commonly use Swift and Objective-C for iOS using Xcode, and Java or Kotlin for Android using Android Studio. Cross-platform development options include languages like JavaScript, TypeScript, and Dart with frameworks such as React Native, Flutter, Xamarin, or PhoneGap/Cordova. |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**: To expand Draw It or Lose It to various computing environments, using a cloud-based server platform like AWS, Azure, or Google Cloud, will offer scalability, cross-platform compatibility, efficient resource management, and high availability for a smooth game experience.
* **Operating Systems Architectures**: Cloud operating platform architectures typically involve virtualization, resource orchestration, distributed data centers, and a network fabric to enable scalable and reliable cloud services delivery.
* **Storage Management** An appropriate storage management system to be used with the recommended cloud operating platform would be a distributed and scalable NoSQL database system like Apache Cassandra.
* **Memory Management**: The recommended operating platform for Draw It or Lose It utilizes memory management techniques such as resource pools, caching, dynamic scaling, and memory monitoring to optimize game performance.
* **Distributed Systems and Networks**: To enable communication between various platforms, a distributed software architecture can be implemented. This involves creating a service-oriented architecture (SOA) or microservices where each component of the Draw It or Lose It application represents a service that communicates through APIs. To ensure robust connectivity, redundancy and load balancing can be employed, reducing the impact of outages or network interruptions, and implementing robust error handling mechanisms to maintain communication integrity across multiple computing environments.
* **Security:** To keep user information safe on different platforms, the Draw It or Lose It app should make sure that data is scrambled when it's sent and when it's saved, use strong passwords and control who gets access to what, and follow the safety rules of the chosen platform. It's also important to check for problems often and have a plan for what to do if there's a security issue.