<Name-of-Software-Application>

CS 230 Project Software Design Template

Version 1.0

Table of Contents

CS 230 Project Software Design Template 1

Table of Contents 2
Document Revision History 2
Executive Summary 3
Requirements 3
Design Constraints 3
System Architecture View 3
Domain Model 3
Evaluation 4
Recommendations 7

Document Revision History

Version	Date	Author	Comments
1.0	<07/23/23>	Kaylee Johan	Project One

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

In this Java game, players take turns drawing a given word or phrase while their teammates try to guess it within a time limit. The challenge lies in designing software that accurately recognizes and evaluates the drawings, ensuring a fair and enjoyable gaming experience for all participants. Addressing this problem is crucial, as it directly impacts the gameplay experience and overall satisfaction of the players. Well-designed software that accurately recognizes and evaluates the drawings will enhance the game's fairness, prevent any potential biases, and ensure that all participants have an equal opportunity to guess and enjoy the game.

Requirements

The requirements for the game are that it will have the option to incorporate one or more teams. Each squad will be allotted a number of players. When choosing a team name, users must be able to verify whether a name is already in use. At any given time, only one instance of the game can exist in memory. This is done by creating unique identifiers for each iteration of a game, team, or player. What is more, the game should have a user-friendly interface that allows players to easily navigate through different game modes and options. It is also important to implement a robust error handling system to gracefully oversee any unexpected errors or exceptions that may occur during gameplay. The Domain Model UML diagram for the game application is the proposed solution. The Domain Model UML diagram for the game application illustrates the structure and relationships of the software design templates. It contains classes such as Participant, Hostile, and Item to illustrate inheritance principles and their relationships. The design incorporates polymorphism, allowing classes to be interchangeable, and adheres to the SOLID principles, thereby encouraging code reuse, maintainability, and scalability. This method enables code flexibility and extensibility.

Design Constraints

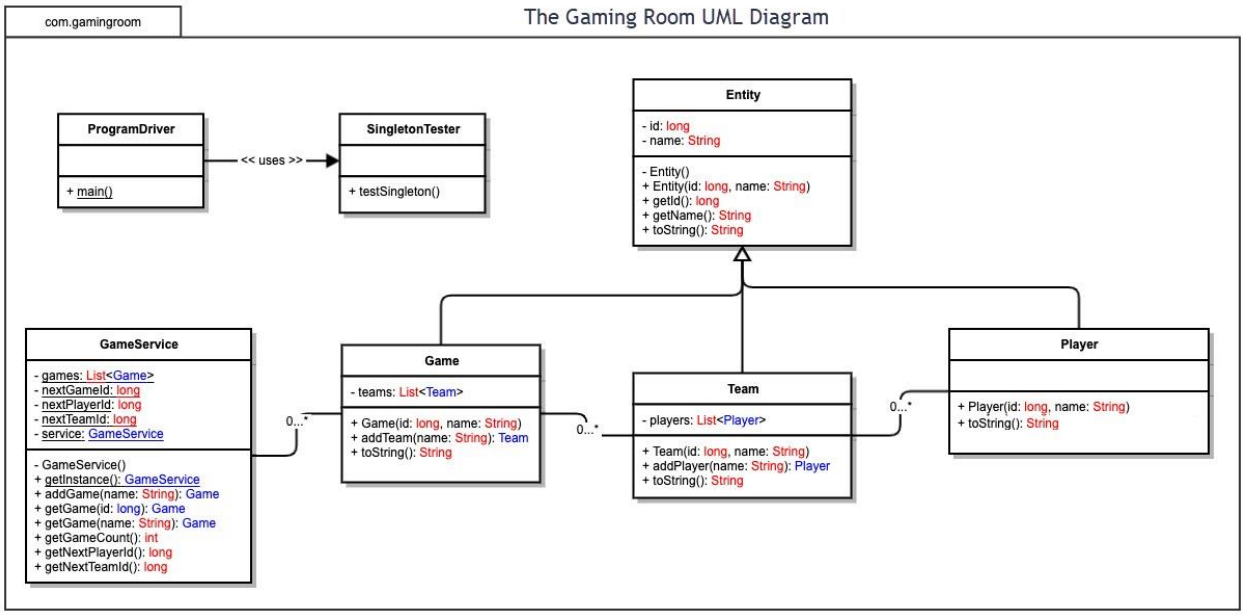
Design restrictions may include hardware or software platform limitations as well as any specific performance requirements or memory constraints. Prioritizing and streamlining the development process is critical to ensuring the game fulfills the tight time restrictions. Setting realistic timelines, streamlining code and assets, and making optimal use of existing resources are all examples of this. Furthermore, regular testing and iteration can assist in identifying and resolving any potential issues or bottlenecks that may occur due to time constraints. When developing an online game, it is critical to consider web-based limitations. These constraints guarantee that users with different internet connections and device capabilities can easily access and play the game. Developers can build a seamless gaming experience for a larger audience by compressing file sizes, lowering processing power requirements, and designing for low network connectivity.

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 demonstrates the software architecture of a game application, including classes inherited from the Entity class, such as Participant, Enemy, and Item. This structure demonstrates the inheritance principle, where subclasses inherit properties and behaviors from their base class. The design incorporates polymorphism, allowing distinct classes to be used interchangeably through a common interface, promoting flexibility and extensibility in code. The SOLID principles ensure that each class has a single responsibility and manages dependencies appropriately, promoting code reuse, maintainability, and scalability. The game system consists of various classes, each with unique attributes and methods. The SingletonTester class contains import statements for necessary packages or libraries, the main method for entry, and method calls to execute desired functionality. The "ProgramDriver" class utilizes the SingletonTester class to access its methods and functionality. The GameService class serves as a central hub for managing game data and executing games, representing an instance of a game with attributes such as gameID, gamename, and gametype. The Team class represents a group of players working together towards a common goal, with attributes such as teamID, teamName, and teamMembers. The Player class represents individual players within the game, providing information about their unique characteristics and performance, as well as methods for actions specific to individual players. The Entity class serves as a parent class for these classes, providing a common structure and functionality for all entities in the game system. This inheritance allows for code reusability and consistency across different entities in the game system. The Game class can interact with instances of the Player and Team classes, allowing for the management and organization of data related to players, teams, and games as a whole.



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 a popular choice for web-based software application hosting due to its sleek design, user-friendly interface, and seamless integration with Apple devices. However, it is expensive and has limited software compatibility compared to Windows systems.	Linux is a dependable, stable, open-source operating system ideal for hosting web-based software applications. It offers customization and flexibility but may require technical expertise and compatibility with proprietary software or hardware.	Windows is a popular choice for hosting web-based software applications due to its user-friendly interface, robust security features, and robust firewalls. However, it is vulnerable to malware attacks and can be more expensive than alternatives.	Evaluate mobile devices for web-based software application hosting by considering operating systems, processing power, memory capacity, portability, and screen size, but also considering potential weaknesses.
Client Side	Supporting multiple clients on Mac requires software development considerations, including higher costs, specialized expertise, and compatibility across macOS and hardware configurations, requiring additional time and resources.	To support multiple clients on Linux, consider the cost, time required for development and deployment, and expertise in Linux-based systems. Evaluate the complexity of client requirements and customization levels, and consider the time required for each type.	To support multiple Windows clients, consider cost, time, and expertise in software development. Determine budget, estimate duration, and ensure compatibility and optimal performance across various platforms. Developing and maintaining software requires expertise and skill sets.	Software development considerations include cost, time, and expertise for supporting multiple mobile clients. Ensuring compatibility across operating systems and optimizing user experience for different screen sizes and resolutions is crucial for seamless functionality.

Development Tools	For deploying on Mobile Devices, relevant Java programming languages include Java ME (Micro Edition) and JavaFX. IDEs like Eclipse, NetBeans, and IntelliJ IDEA are commonly used for development on Mac. Additionally, tools like Android Studio can be utilized for building mobile applications specifically for Android devices running on the Mac platform.	Some relevant Java programming languages for building software for deploying on Windows Mobile Devices include Java ME (Micro Edition) and JavaFX. IDEs like Eclipse, NetBeans, and IntelliJ IDEA are commonly used for development. Additionally, tools like Android Studio can be utilized for building mobile applications specifically for Android devices running on the Windows platform.	C#, Visual Basic.NET, and C++ are popular programming languages for Windows deployment, often used with IDEs like Visual Studio and Microsoft.NET Framework. Other tools include WPF, UWP, and Azure.	Java programming languages and tools commonly used for building software for deploying on mobile devices include Java SE, Java ME, Android Studio, Eclipse, and IntelliJ IDEA. These tools provide developers with the necessary resources and features to create efficient and functional mobile applications. Additionally, frameworks such as Flutter and React Native can also be utilized to build cross-platform mobile apps using Java.

Server Side

In order to evaluate the characteristics, advantages, and weaknesses of each operating platform for hosting a web-based software application, it is important to consider if each platform offers a server-based deployment method. This will determine if the website can be hosted on the platform. Additionally, it is crucial to evaluate the potential licensing costs for the server operating system that The Gaming Room would incur because this will have an impact on their financial resources and overall budget. The licensing costs for a server operating system can vary greatly depending on the platform chosen, and it is essential for The Gaming Room to carefully evaluate and compare these costs to ensure they are within their means. Furthermore, it is important to consider the level of technical support and security features provided by each server operating system, as these factors can greatly impact the performance and reliability of the web-based platform. Additionally, The Gaming Room should also consider the scalability and compatibility of the server operating system with their existing infrastructure and future growth plans. This will ensure that they can easily expand their operations and integrate modern technologies without incurring significant additional costs or disruptions to their business.

Client Side

To ensure compatibility with all web browser platforms and mobile devices, the application development process needs to prioritize cross-platform compatibility. This includes using responsive design techniques to ensure the application adapts to different screen sizes and resolutions. Additionally, thorough testing on various browsers and devices is crucial to identifying and fixing any compatibility issues that may arise. It is also important to stay updated with the latest web standards and technologies to ensure optimal performance across different platforms and devices. This may involve staying informed about current updates and releases from operating systems like iOS and Android, as well as keeping up with advancements in web development frameworks and languages. By taking initiative-taking in adopting these updates, developers can ensure that their applications remain compatible and performant across a wide range of devices, providing a seamless user experience for all users. Additionally, developers should also consider accessibility guidelines and best practices to ensure that their applications are usable by individuals with disabilities. This includes incorporating features such as screen reader compatibility and keyboard navigation options.

Development Tools

The technical requirements of using specific programming languages and tools for building software for different operating platforms can have a significant impact on a development team. It may require the team to have expertise in multiple programming languages and tools, which could potentially increase the complexity of development and require additional training or the hiring of specialized developers. Additionally, different operating platforms may have their own unique development environments and frameworks, further adding to the complexity. In terms of licensing costs, it is important to consider that some development tools may require the purchase of licenses for each individual developer, which can significantly increase the overall cost of development. Moreover, integrating different programming languages and tools may also introduce compatibility issues and potential conflicts, leading to additional time and effort spent on troubleshooting and resolving these issues. Additionally, the need for continuous training and updating of skills for developers working with multiple languages and tools can also impact productivity and increase costs overall.

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:** The Gaming Room can expand Draw It or Lose It to various computing environments using Unity, a cross-platform game engine with powerful graphics, intuitive interface, and extensive documentation for easy development.

2. **Operating Systems Architectures**: Unity supports various operating system architectures, including x86, x64, ARM, and ARM64, enabling seamless gaming across various devices and platforms. It supports graphics APIs like DirectX and OpenGL for optimal performance.

3. **Storage Management**: Cloud storage is a suitable storage management system for operating platforms, enabling online access to game data and progress, scalability, and backup options for safety and availability.

4. **Memory Management**: The Draw It or Lose It software uses memory management techniques to optimize performance and allocate resources efficiently, minimizing memory leaks and ensuring smooth gameplay without lag or crashes.

5. **Distributed Systems and Networks**: Draw It or Lose It uses a client-server architecture, with a central server managing game logic and data. Clients connect to the server for real-time updates, and distributed software can manage connectivity issues with fault-tolerant mechanisms like redundancy and replication.

6. **Security**: Implement robust security measures, including encryption and authentication, to protect user information across platforms. Regular audits and updates address vulnerabilities. Consider operating platform capabilities like built-in firewalls, secure boot processes, and app sandboxing. Implement multi-factor authentication, such as biometric or token-based, for enhanced security.

Conclusion

The recommendations given for storage management, memory management, distributed systems and networks, and security are crucial aspects of ensuring the overall functionality and safety of a system. For storage management, it is important to regularly audit and update systems to address vulnerabilities. Additionally, considering operating platform capabilities like built-in firewalls, secure boot processes, and app sandboxing can further enhance security measures. Lastly, implementing multi-factor authentication, such as biometric or token-based methods, can provide an extra layer of protection for the system. By following these recommendations, the overall functionality and safety of the system can be ensured. For memory management, it is important to regularly monitor and optimize the usage of system resources. This can help prevent crashes or slowdowns caused by excessive memory usage. Additionally, implementing secure coding practices and regularly updating software can help mitigate potential memory-related vulnerabilities and ensure the system's stability and reliability. For distributed systems and networks, it is crucial to establish robust communication protocols and implement fault-tolerant mechanisms. This can ensure seamless data transmission and minimize the impact of network failures or congestion. Additionally, employing load balancing techniques and redundancy measures can help distribute workloads efficiently and enhance system performance. Regular monitoring and analysis of network traffic can also help identify potential bottlenecks or security threats, allowing for timely mitigation actions to be taken. Lastly, for security purposes, it is crucial to implement strong authentication and encryption protocols to protect sensitive data from unauthorized access or interception. Regular security audits and updates should also be conducted to stay ahead of emerging threats and vulnerabilities in the network. In other words, these recommendations were given because they are essential for maintaining the integrity and confidentiality of network communications. By following these best practices, organizations can ensure that their networks are resilient against potential attacks and unauthorized access attempts. Additionally, implementing these recommendations can help organizations comply with industry regulations and standards related to data protection and privacy.