**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 5**

Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/16/2024 | Jay Morocho | Initial draft of the software design document. |
| 1.1 | 12/14/2024 | Jay Morocho | Completed Project Three Recommendations section with improved formatting. |

Executive Summary

The Gaming Room, a client of Creative Technology Solutions (CTS), plans to port its game, Draw It or Lose It, which is currently available on Android platforms, to the web, where it can be played across multiple devices. The essence of the game is multiplayer teams that take turns and try to guess images from provided clues.

This paper presents the terms of the application design and application software, the object-oriented software design principles, and patterns that have been used to meet the specified objectives of the client, including:

* Supporting many teams with various names.
* Enforcing the Singleton pattern concerning the number of game instances in memory.
* Creating a web-based distributed ‘cloud’ application that is likely to be used by many clients across the globe.

The proposed solution employs sound object-oriented concepts and system architecture to address the client’s needs while ensuring the code is clean and easy to manage.

Requirements

The Gaming Room has laid down a number of business and technical requirements that should be suited to the new game application.

* **Multiple Teams:** The game should allow one or more teams, and each team should have several players.
* **Unique Names:** The game and team names should be such that no two names are similar.
* **Single Instance:** The game in memory should exist in one instance only, at any point in time.
* **Scalability:** The number of games, the number of teams, and the number of players should be on the increasing side.
* **Security:** Ensure user and game information is secure and safe amidst multi-platform interactions.

Design Constraints

* **Concurrency:** Ensuring that more than one user can play concurrently allows massive simultaneous plays, which warrants thread-safe operations. Otherwise, this will result in data inconsistency and race conditions.
* **Scalability:** Changes in the structure should not affect performance speeds as the number of games, players, and teams increases in the future.
* **Data Synchronization:** The updates across all platforms should be instant and take little time. This is necessary for maintaining the integrity of the game across platforms.
* **Web-Based Environment:** When the application is hosted on the web, there should be no limitations. The application should be able to work on mobile devices and desktops.
* **Security:** Data should be made unreadable if located outside the server, and possible intrusions should be checked through some authentication methods.

These constraints will affect the design and implementation. Patterns such as the Singleton pattern, which helps in memory management, and the Iterator pattern, which enforces a unique name, are also needed.

System Architecture View

Although this section is not explicitly required, the architecture for this project would involve:

* **Client-Side Interface:** The interface will be available for the users to interact with through mobile applications and web browsers.
* **Server-Side Logic:** This will be implemented on a web server such as Apache Tomcat that contains the game's content, players, and teams.
* **Database Layer:** This relational database, such as MySQL, will make game, team, and player data persistent.

Domain Model

Using a UML diagram, the relationships between the Game, Team, and Player classes are outlined, and all three inheritance classes are derived from the Entity base class. Other principles of object-oriented programming illustrated include:

* **Inheritance:**  
   The Entity Class provides common characteristics, i.e., attributes (id and name), which become inheritable by Game, Team, and Player, reducing redundancy.
* **Encapsulation:**  
   Attributes are made private and protected and are accessed via getters.
* **Polymorphism:**  
   The method toString() has been overridden by each subclass to display particular information.
* **Singleton Pattern:**  
   This prevents the alteration of instances so that only one example of the game service class exists.
* **Iterator Pattern:**  
   Applied to ensure games and team names are not repeated by going through already created lists before new ones are added.

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** | Stable environment; not ideal for high-performance hosting. | It has high scalability and preservation, particularly for web hosting. | This is better for the smaller level of hosting but not that viable when it needs to be scaled. | There are limitations regarding how many server-side applications can be hosted through this channel. |
| **Client Side** | External sources are incorporated but do come with some cost. | It is economical and offers varied surroundings. | Developers are conversant with it; its licensing costs are higher. | The app can be processed with no difficulties. |
| **Development Tools** | IDEs like IntelliJ IDEA or Eclipse are compatible with Java. | Eclipse, or IntelliJ IDEA tools, are open source; Java is prepared. | There is Eclipse and Tenable Visual Studio; Java is ready. | Other desktop applications, such as Android Studio, are available for use on specific app needs. |

**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**

For the cross-multiplying game “Draw It or Lose It,” the suggested operating platform is Linux. Linux, with its remarkable scalability, economy, and reliable performance, is a robust option for hosting games on multiple platforms. Its open-source nature not only cuts down on licensing fees but also provides excellent support for server-side applications. Moreover, Linux offers various tools and frameworks, including but not limited to Apache Tomcat and MySQL, which are perfectly suited for the technical aspects of this project.

### **Operating Systems Architectures**

The selected research approach, on the other hand, is a Linux operating system server and the usage of a kernel architecture that is modular in nature. This type of structure allows more efficiency in the acquisition of the system since the system will only acquire components that it requires to be in memory space.

The conceived game will be run on the web by Apache Tomcat, which can work in conjunction with REST APIs to allow connections to many applications and platforms. Users will interact with the game using React Native, enabling usage across many platforms, such as iOS, Android, Windows, and Mac OS. The advantages of this structure include scalability and ease of being made to fit.

### **Storage Management**

A relational database management system (RDBMS), specifically MySQL, is recommended and adopted to manage the data of games, teams, and players. The advantages of using MySQL include:

* **Data Persistence:** This ensures that all information, including user accounts, session details, and teams, will be securely stored within the application.
* **Data Validation:** Implementing unique constraints prevents the alteration of team and player names.
* **Backup and Recovery:** Allowing offsite mirroring with encryption will hinder data loss and assurance of security.

Also, query caching will optimize the database, especially for pages with a high frequency of use or large databases.

### **Memory Management**

It is recommended that the platform uses virtual memory management techniques to optimize the use of resources at hand for the platform. Some of the features include:

* **Singleton Pattern:** Multiple instances of the game may exist in memory, but only one is allowed. This is to reduce high memory use.
* **Garbage Collection:** To reduce memory leaks, the Java Virtual Machine (JVM) automatically recovers unused objects.
* **Memory Profiling Tools:** It is recommended that Android Studio’s Memory Profiler and Xcode’s Instruments (macOS) be used to detect and fix memory bottlenecks to ensure seamless application functionality on different platforms.

### **Distributed Systems and Networks**

By using RESTful APIs over HTTPS, "Draw It or Lose It "will be able to communicate across different platforms, enhancing the gaming experience and making the audience feel more connected and engaged in the game's development. Major Components comprise of:

* **WebSockets Protocol:** Making it possible to use lower latencies such as games in both the game chat and other live game events.
* **Load Balancing:** Inbound requests are relayed by way of Heroku-based load balancers, ensuring that no one server is overloaded and that the resources are highly available.
* **Dependency Management:** The architecture will consider the possibility that clients may lose connectivity and make provision for retry and caching strategies for client-sided operations.

This distributed architecture maintains the three major East Coast principles as a document consistency model while providing reliable inter-communication across all devices.

### **Security**

User interactions across different platforms necessitate adequate protection to guarantee that the user's data is protected. Such criteria that can be considered necessary include the following:

* **Encryption:** SSL/TLS protocols will cover all users’ active data during the transport stage.
* **User Authentication:** Secure login and management of a particular session with the user's data will be achieved using OAuth 2.0 and JSON Web Tokens (JWT).
* **Access Control:** Using the Least Privilege Principle for data and sensitive functionality access limitation.
* **Input validation:** Any input provided by the user is thoroughly validated to protect against injections and related vulnerabilities.
* **Firewall Configurations:** Enforcing or relating to host-based firewalls will provide attempts and actions to deny and attempt to contact unauthorized users.

On that note, one can be sure that there could be no better gaming experience throughout all the platforms while playing “Draw It or Lose It,” as these measures will guarantee utmost safety.