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	05/26/24	Daniel Aguiar	Initial version
2.0	06/13/24	Daniel Aguiar	Added scenario and evaluation
3.0	06/23/24	Daniel Aguair	Review recommendation section

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, a client of Creative Technology Solutions (CTS), seeks to develop a web-based version of their Android game, Draw It or Lose It. This game is similar to the 1980s television game show "Win, Lose or Draw," where teams guess what is being drawn. The new web-based game must support multiple platforms and include capabilities for team play, unique naming for games and teams, and single-instance game memory management. This document outlines the design constraints, domain model, and evaluations necessary to meet the client's requirements effectively.

The client is interested in expanding their gaming app to multiple platforms using various software patterns in a distributed environment. Specifically, before making a decision, The Gaming Room has asked for you to evaluate the three traditional operating platforms (Linux, Mac, and Windows), as well as mobile platforms, for how the game application software could be deployed and run and what would be required to do so.

Requirements

The client requires the following for the game application:

- Ability to add one or more teams in a game
- Add multiple players to each team
- Unique game and team names to avoid duplicates
- A single instance of the game in memory at any given time, with unique identifiers for each game, team, or player.

Design Constraints

1. **Scalability**: The system must handle multiple concurrent users and games without performance degradation.

2. **Consistency**: Ensuring data consistency across distributed systems is crucial, especially for team scores and player data.

3. **Security**: Protecting user data and preventing unauthorized access are paramount.

4. **Responsiveness**: The application must provide real-time feedback to users, requiring efficient network communication and low latency.

5. **Maintainability**: The design should facilitate easy updates and bug fixes.

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.

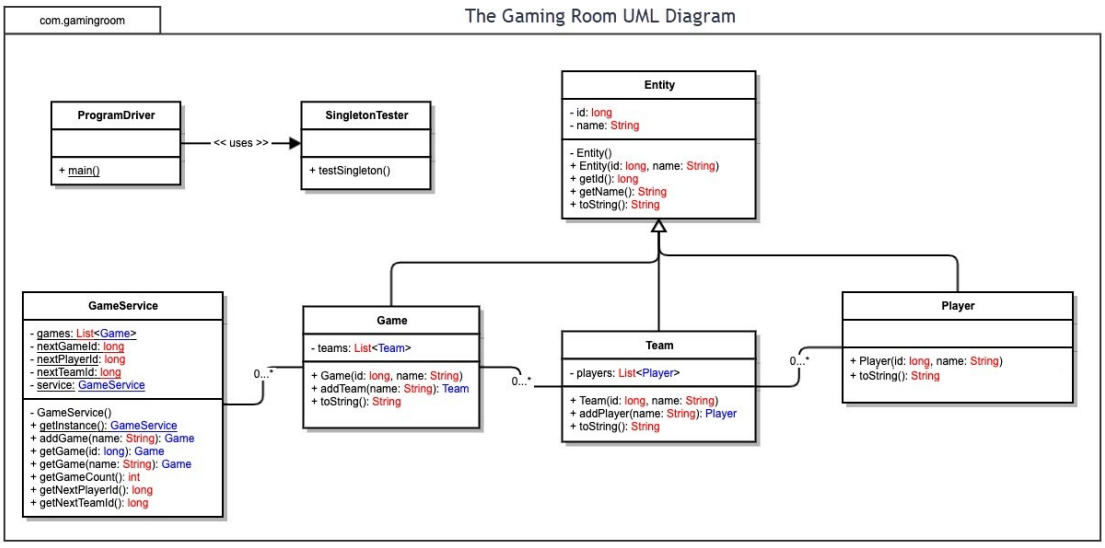
Domain Model

The UML class diagram provided illustrates the following classes and their relationships:

- **Entity**: A base class with common attributes (*id* and *name*) and methods (*getId()*, *getName()*, and *toString()*).
- **GameService**: Manages the games, including methods to add games, get game information, and maintain next identifiers for games, teams, and players.
- **Game**: Contains a list of teams and methods to add teams and provide a string representation.
- **Team**: Contains a list of players and methods to add players and provide a string representation.
- **Player**: Represents individual players with methods to get player information.
- **ProgramDriver** and **SingletonTester**: Classes for testing and initializing the game instance.

Object-Oriented Programming Principles:

1. **Inheritance**: The `Entity` class serves as a base class for `Game`, `Team`, and `Player`, promoting code reuse and consistency.
2. **Encapsulation**: Each class encapsulates its attributes and provides methods to interact with them.
3. **Singleton Pattern**: `GameService` likely implements the Singleton pattern to ensure only one instance exists, fulfilling the requirement for single-instance game management.



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.

Server Side

1. **Linux**
    - **Characteristics**: Linux is known for its stability, security, and open-source nature. It is the most popular choice for web servers.
    - **Advantages**:
        - Cost-effective: No licensing fees.
        - Scalability: Easily scales to handle large numbers of users.
        - Security: Strong security features and regular updates.
        - Flexibility: Supports a wide range of server software and configurations.
    - **Weaknesses**:
        - Requires skilled administrators.
        - Steeper learning curve for new users.
    - **Server-Based Deployment**: Widely supported and highly recommended for web-based applications.
    - **Licensing Costs**: Free (open-source).
2. **Mac**
    - **Characteristics**: Mac offers a stable, UNIX-based environment, similar to Linux, but is less common for server deployment.
    - **Advantages**:
        - Stability and security.
        - Excellent development tools.
        - Integration with other Apple products.
    - **Weaknesses**:
        - Higher hardware costs.
        - Less commonly used in server environments.
    - **Server-Based Deployment**: Available but not as common or recommended as Linux.
    - **Licensing Costs**: Higher hardware costs, no additional licensing fees for macOS server.
3. **Windows**

- **Characteristics**: Windows is widely used, especially in enterprise environments.
- **Advantages**:
    - Familiar interface and environment.
    - Extensive support and documentation.
    - Integration with other Microsoft products.
- **Weaknesses**:
    - Higher licensing costs.
    - Susceptible to malware.
- **Server-Based Deployment**: Supported and commonly used, especially in enterprises.
- **Licensing Costs**: Requires Windows Server licenses, which can be costly.

4. **Mobile Devices**

- **Characteristics**: Mobile devices can act as lightweight servers but are not suitable for high-performance needs.
- **Advantages**:
    - Portability.
    - Suitable for specific lightweight tasks.
- **Weaknesses**:
    - Limited performance and scalability.
    - Not suitable for hosting large-scale web applications.
- **Server-Based Deployment**: Not recommended for primary hosting of web applications.
- **Licensing Costs**: Generally not applicable for server use.

Client Side

1. **Mac**

- **Characteristics**: Macs provide a seamless development environment for iOS and macOS applications.
- **Advantages**:
    - Excellent development tools like Xcode.
    - High-quality hardware and stable OS.
- **Weaknesses**:
    - Higher initial cost.
    - Limited hardware options.
- **Development Requirements**: Development on Mac is straightforward with robust tools and support, but higher costs need to be considered.
- **Compatibility**: High compatibility with web standards and modern browsers.

2. **Linux**

- **Characteristics**: Linux offers a cost-effective and powerful development environment.
- **Advantages**:
    - Flexibility and extensive open-source tools.
    - Robust performance and stability.
- **Weaknesses**:
    - Steeper learning curve for new users.
    - Compatibility issues with some proprietary software.
- **Development Requirements**: Linux provides extensive tools for web development but requires experienced developers.
- **Compatibility**: High compatibility with web standards and modern browsers.

3. **Windows**

- **Characteristics**: Windows provides a familiar and versatile development environment.
- **Advantages**:
    - Wide range of development tools.
    - Strong support for various languages and frameworks.
- **Weaknesses**:
    - Higher vulnerability to malware.
    - Licensing costs for development tools.
- **Development Requirements**: Windows offers extensive tools and support, making it a strong choice for development.
- **Compatibility**: High compatibility with web standards and modern browsers.

4. **Mobile Devices**

- **Characteristics**: Mobile devices require apps optimized for touch and screen size.
- **Advantages**:
    - Essential for modern accessibility.
    - Huge market potential.
- **Weaknesses**:
    - Fragmentation across devices.
    - Different OS requirements (iOS vs. Android).
- **Development Requirements**: Requires knowledge of mobile development platforms and tools.
- **Compatibility**: Needs responsive design to ensure compatibility across different devices and screen sizes.

Development Tools

1. **Mac**

- **Languages and Tools**: Uses Swift, Objective-C, JavaScript, and Xcode.

- **Impact**: Higher cost for hardware and development tools, but seamless integration for Apple products.
- **Licensing Costs**: Xcode is free, but Apple Developer Program costs $99/year.

2. **Linux**
    - **Languages and Tools**: Uses Java, Python, JavaScript, and open-source tools like Eclipse, IntelliJ IDEA, Neovim, etc..
    - **Impact**: Lower cost due to open-source tools, but requires skilled developers.
    - **Licensing Costs**: Generally free or low cost due to open-source nature.
3. **Windows**
    - **Languages and Tools**: Uses C#, .NET, JavaScript, and Visual Studio.
    - **Impact**: Familiarity for many developers, extensive documentation, but potential higher costs.
    - **Licensing Costs**: Visual Studio can be free or require a subscription for advanced features.
4. **Mobile Devices**
    - **Languages and Tools**: Uses Swift, Kotlin, Java, JavaScript, and tools like Android Studio, Xcode.
    - **Impact**: Need to manage different development environments for iOS and Android.
    - **Licensing Costs**: Android Studio is free, Apple Developer Program costs $99/year.

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**: Recommend using Linux for its cost-effectiveness, stability, and scalability for server-side development. Linux has strong support for web servers and is highly customizable.

2. **Operating Systems Architectures**: The Linux operating system architecture is based on a monolithic kernel which provides efficient management of hardware resources, file systems, and system calls.

3. **Storage Management**:
   An appropriate storage management system for the recommended Linux platform is a Relational Database Management System (RDBMS) such as PostgreSQL. PostgreSQL is known for its reliability, data integrity, and support for complex queries.

4. **Memory Management**: Linux's memory management techniques, including virtual memory and caching, will ensure efficient use of resources.

5. **Distributed Systems and Networks**: To facilitate communication between various platforms, I recommend using RESTful APIs for the distributed software architecture. RESTful APIs enable seamless and scalable communication between distributed components by leveraging standard HTTP methods.

6. **Security**: Security is paramount, especially when handling user data. To protect user information:

   Encryption: Use HTTPS for secure communication between clients and servers, ensuring data is encrypted during transit.
   Authentication: Implement strong authentication mechanisms such as OAuth 2.0 to verify user identities.
   Access Control: Enforce role-based access control (RBAC) to restrict access to sensitive data.
   Data Protection: Use encryption for data at rest and implement regular security audits to identify and mitigate vulnerabilities.