CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	03/23/2025	Dustin L	Initial draft of software design document for Draw It or Lose It web application.

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 currently offers Draw It or Lose It as an Android app, but they want to expand it to a web-based version that works across multiple platforms. My goal is to help them make this transition by outlining a software design plan that addresses both the technical and business needs. I'm proposing a solution that supports scalable deployment, avoids duplicate game instances, and allows unique identification of games, teams, and players. To do this, we'll use design patterns like Singleton and Iterator to manage resources and maintain data integrity.

Requirements

- The game must support one or more teams.
- Each team must include multiple players.
- Game and team names must be unique.
- Only one instance of the game can exist in memory at a time.
- The application must be accessible across platforms.

Design Constraints

One major constraint is the need to support a web-based distributed environment. This means everything needs to run efficiently across different systems, networks, and devices. Because we're limiting memory use to one instance of the game, we need to use the Singleton pattern to manage access. Data consistency is another constraint—names for games and teams must remain unique. Also, managing different user devices adds complexity because not all browsers and hardware behave the same. These constraints affect how we store and retrieve data, ensure fast performance, and maintain secure sessions.

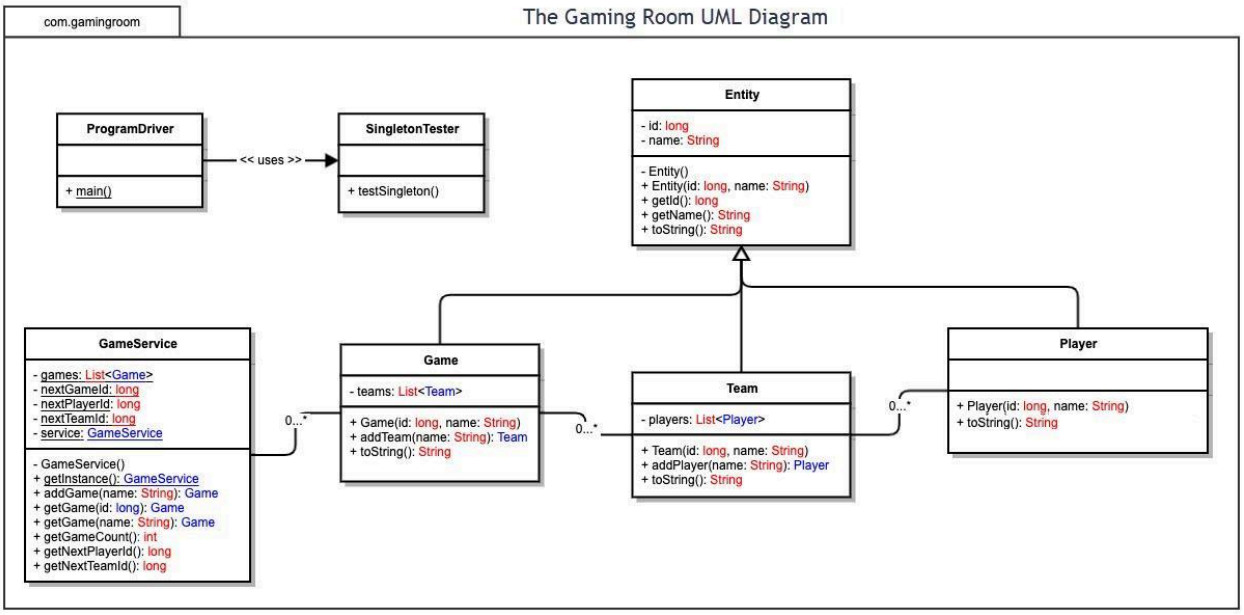
System Architecture View

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.>

The UML diagram shows how the different classes interact in the application. Entity is a base class that Game, Team, and Player inherit from. This helps reduce repeated code and makes it easier to track common attributes like id and name. GameService manages all game-related logic and uses the Singleton pattern so only one instance runs at a time. Teams belong to games, and players belong to teams. The structure supports object-oriented principles like inheritance, encapsulation, and modularity, which help keep the code organized and efficient.



Evaluation

Project 1:

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 servers provide reliable performance and security, but they are rarely used for hosting large-scale web applications. They are expensive, harder to scale, and lack community support for server-side deployment. Best suited for development or design-related workflows, not web hosting.	Linux is widely used for web servers because it's lightweight, secure, open source, and scalable. Distributions like Ubuntu and CentOS support popular stacks (LAMP, MEAN) and offer great uptime and flexibility. It's cost-effective and easy to automate using shell scripts and tools like Docker.	Windows Server supports .NET applications and has a familiar interface, making it easier for developers coming from a Windows background. However, it requires licensing and can be resource-heavy. It is suitable for enterprise environments but not always ideal for scalable web hosting.	Mobile devices can't host server-side applications. Their role is to connect to cloud services and present data to users. They depend on efficient client-server communication and backend performance.
Client Side	Developing for macOS clients often requires Swift and Xcode. Testing can be limited unless using Apple hardware. Apps must follow Apple's App Store review process, which adds time and costs. Ideal for reaching a high-end user base.	Linux desktop users are a small market share, but still relevant in open-source circles. Development tools are free, but varied hardware and distributions can make testing more complex. It's good for web-based interfaces but rarely a priority for desktop-specific client builds.	Most desktop users run Windows, making it essential to support. Tools like Visual Studio simplify development. Cost is relatively low, and wide compatibility with browsers and hardware makes deployment reliable.	To reach the most users, development must support both Android and iOS. Tools like Flutter, React Native, or native SDKs (Android Studio, Xcode) help manage this. Mobile clients must be responsive and optimized for performance, especially in real-time games.

Development Tools	Xcode is required for iOS/macOS native apps. Swift is the primary language. Developers may also use Flutter, React Native, or Unity for cross-platform needs. Terminal and Git are often used in development workflows.	Developers often use Eclipse, NetBeans, or VS Code. Tools like Docker, Git, and terminal-based build systems (Make, Gradle) are common. Languages include Java, Python, JavaScript, and C++.	Visual Studio is the most popular IDE on Windows. Developers use C#, .NET, and also cross-platform tools like Node.js, Java, and Python. Windows Subsystem for Linux (WSL) adds flexibility for developers working in hybrid environments.	Android development typically uses Android Studio with Java or Kotlin. iOS development uses Xcode with Swift. For cross-platform projects, Flutter and React Native are widely supported, helping manage one codebase across platforms.

Project 2:

Server Side: The client has asked you to create a web-based application. This implies a server-style configuration for hosting the website and allowing it to scale up to thousands of players. What does this mean for your ability to host the software application on each operating platform listed above?

Evaluate various platforms for their characteristics, advantages, and weaknesses for hosting a web-based software application. Consider the following in your evaluation and articulate your findings in the software design template:
Does each of the operating platforms offer a server-based deployment method where the website will be hosted?
What are the potential licensing costs to the client, The Gaming Room, for the server operating system?
Client Side: The client wishes to move beyond their current Android-only application to supporting players on iOS and Android mobile platforms, as well as traditional desktop-based operating systems. The application must be delivered as a modern, responsive HTML interface running inside the web browser for desktop clients (Linux, Mac, Windows), as well as on mobile platforms. Each will be capable of communicating with the back-end web application running on the server.

Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients. Consider the following in your evaluation and articulate your findings in the software design template:
What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices?
Development Tools

Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on each operating platform. Consider the following and articulate your findings in the software design template:
What impact do these technical requirements have on a development team? Consider whether multiple development teams may be needed.
Are there licensing costs related to the development tools?

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	macOS can be used as a server, but it's not common. It's more expensive and doesn't have as much support for hosting websites as Linux. Most companies don't use Mac servers for big applications. Licensing costs and hardware costs make it harder to recommend (Apple, 2023).	Linux is one of the best options for running websites and web apps. It's free, open source, and widely used. Tools like Apache, NGINX, and MySQL work well with it. There are no licensing costs, and it's great for scaling to lots of users (Red Hat, 2023).	Windows Server is used by many businesses. It supports web apps using ASP.NET or IIS. However, it has licensing fees and usually takes more system resources than Linux. It's a good option if you're using Microsoft tools already (Microsoft, 2023).	Mobile devices don't host servers. Instead, they connect to a server to get data or send info. The game app will connect to a server that runs the main logic of the game (Android Developers, 2023).
Client Side	Apps for macOS are usually written in Swift and built using Xcode. You need a Mac computer to build and test apps. Apple also has strict rules for getting apps on the App Store, which can take time and cost money (Apple Developer, 2023).	Not many users run Linux on desktops, but Linux supports web browsers like Firefox and Chrome. A responsive web app will work just fine on Linux without needing a special version (Canonical, 2023).	Most people use Windows, so it's very important to support. Web apps work well in Chrome, Edge, or Firefox. Tools like Visual Studio make it easy to build desktop or web apps (NetMarketShare, 2023).	The app needs to run on Android and iOS. Android uses Java or Kotlin, and iOS uses Swift. You can also use cross-platform tools like Flutter or React Native to build one app that runs on both systems (Google, 2023).

Development Tools	Developers use Xcode for native apps. If building a web app or mobile app, tools like Flutter or React Native also work. Xcode is free, but you need a $99/year Apple Developer account to publish apps on the App Store (Apple Developer, 2023).	Linux supports many free tools like VS Code, Eclipse, and terminal tools like Git and Docker. It's great for web development and testing server code. Most tools are free and easy to install (The Linux Foundation, 2022).	Visual Studio is popular on Windows and supports many languages like C#, JavaScript, and Python. Visual Studio has a free Community Edition version and paid versions for bigger teams (Microsoft, 2023).	Android Studio and Xcode are the main tools for Android and iOS. Flutter and React Native also work for both. Most tools are free, but iOS requires a $99/year developer account to publish to the App Store (Flutter, 2023; Apple Developer, 2023).

Sources

Apple Developer. (2023). *Apple Developer Program*. https://developer.apple.com/programs/
Android Developers. (2023). *App architecture*. https://developer.android.com/jetpack/guide
Canonical. (2023). *Ubuntu Desktop Overview*. https://ubuntu.com/desktop
Flutter. (2023). *Build apps for any screen*. https://flutter.dev/
Google. (2023). *Flutter for mobile*. https://flutter.dev/multi-platform/mobile
Microsoft. (2023). *Visual Studio Editions*. https://visualstudio.microsoft.com/
NetMarketShare. (2023). *Desktop Operating System Market Share*. https://netmarketshare.com
Red Hat. (2023). *Why Linux is the Best OS for Developers*. https://www.redhat.com
The Linux Foundation. (2022). *Open Source Tools for Developers*. https://www.linuxfoundation.org

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**: I recommend Linux for the server-side operating system due to its high performance, low resource requirements, robust security features, and cost-effectiveness. Linux distributions such as Ubuntu Server or CentOS are widely used in production environments and support web servers like Apache and NGINX. This makes them ideal for hosting the Draw It or Lose It application in a scalable and efficient manner. On the client side, the application should support Windows, macOS, Android, and iOS through a browser-based interface, ensuring cross-platform compatibility.

2. **Operating Systems Architectures**: The server should implement a standard LAMP (Linux, Apache, MySQL, PHP) or MEAN (MongoDB, Express.js, Angular, Node.js) stack. Linux's kernel architecture supports loadable kernel modules and efficient system calls, allowing high performance and modular design. For client devices, the application will run in a web browser using HTML, CSS, and JavaScript, which are supported by all modern operating systems. This architecture allows for efficient communication between front-end clients and the Linux server over HTTPS.

3. **Storage Management**: The best approach for managing game data, user accounts, and session states is to use cloud-based object storage like Amazon S3 or Firebase. These systems allow for fast data retrieval, horizontal scaling, and data redundancy. Cloud storage also simplifies integration with CDNs (Content Delivery Networks), ensuring that users experience minimal latency regardless of location.

4. **Memory Management**: Linux uses advanced memory management techniques such as virtual memory, paging, and caching to efficiently allocate and deallocate system resources. In the Draw It or Lose It application, memory use is minimized by enforcing a single game instance in memory via the Singleton pattern. Linux’s slab allocator and out-of-memory (OOM) killer also ensure stability under heavy loads, making it a strong choice for a high-traffic web game.

5. **Distributed Systems and Networks**: To support real-time interactions across devices and platforms, the game will use WebSocket connections or RESTful APIs over HTTPS. This allows continuous communication between clients and the backend server. Distributed software elements, such as load balancers, failover servers, and redundant databases, will ensure uptime and scalability. Techniques like heartbeats, retry logic, and circuit breakers will help maintain reliability across network partitions or outages.

6. **Security**: Security is always a top priority. The recommended server will enforce HTTPS, employ OAuth 2.0 for authentication, and store sensitive data in encrypted formats. Firewalls, intrusion detection systems (IDS), and secure socket layer (SSL) certificates will protect server communications. On the client side, input validation and role-based access control will prevent unauthorized actions. Linux's access control lists and regular patching will ensure ongoing protection against vulnerabilities.