

Draw It or Lose It

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 11/20/23 | Asia Mayfield | This revision centers on the software design for Draw It or Lose It. The executive summary outlines the high level plan and highlights core design constraints of a web-based environment. Operating platforms were assessed on compatibility, tools, and other factors to recommend the ideal system for development. |

**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**

Draw It or Lose It, is a competitive online guessing game using stock images as visual clues. This document outlines the plan for meeting software requirements like unique IDs, real-time multiplayer interaction, and concurrent access. Strict design constraints come from the distributed, web-based nature of the game. The domain model employs object oriented principles to structure relationships between game components and enable efficient performance. Each operating platform will be evaluated on factors like compatibility, tools, and server robustness to recommend the optimal system for secure, and scalable deployment that satisfies all client requirements.

**Requirements**

The client outlined some key business and technical needs, which includes, support for multiple teams, several players per team, unique game and team names, and one game instance at a time. These are core to the software design, ensuring a fun and engaging experience.

**Design Constraints**

The online distributed nature of the game introduces challenges like managing concurrent access, state data, and info accuracy. The unique IDs and single active game requirement need a robust backend structure and careful planning. This affects the app's scalability and speed.

**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 illustrates the application's structure using object-oriented programming OOP principles such as inheritance, encapsulation, and the Singleton pattern. Inheritance is evident as the Game, Team, and Player classes all extend from the Entity base class, allowing them to share common properties and reduce code duplication. Encapsulation is shown by how these classes manage and expose their data and behavior through methods, safeguarding data integrity. The Singleton pattern is implemented in the GameService class to ensure only one game instance is active, directly aligning with the game's requirement to prevent duplicate game sessions. These OOP principles help organize the system to meet the gaming application's requirements efficiently.



**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** | Is known for its stable and secure environment, suitable for hosting web based applications. However, it is not commonly used for servers due to its proprietary nature and hardware requirements. | Is widely recognized for its robustness and is commonly used for server environments. It is open source, which allows for extensive customization. | Offers a user friendly interface and integrates well with other Windows applications and services. However, it requires licensing fees and may be more susceptible to malware than other platforms. | Hosting a web based application on mobile devices isn't typical due to hardware limitations, though it could be suitable for lightweight applications with low traffic. |
| **Client Side** | Provides excellent support for web based applications from a client perspective, offering modern browsers and good hardware capabilities, but can be costly. | Can be more complex to support due to the variety of distributions and configurations, which might require more specialized knowledge. | Is the most widely used OS for clients, thus offering a large audience. It requires consideration for different versions and their compatibility. | Mobile devices are crucial for reaching a broad audience. They require a responsive design and consideration for different operating systems like iOS and Android. |
| **Development Tools** | Supports a range of development tools including Xcode, which is well suited for creating web based applications, especially if integrating with Apple services. | Supports a wide array of open source development tools and environments, such as Eclipse and NetBeans, making it a versatile platform for development. | Supports a variety of development tools such as Visual Studio, which is a comprehensive suite for building web-based application | Development for mobile devices requires specific tools like Android Studio for Android apps and Xcode for iOS apps, as well as cross platform frameworks like React Native. |

**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**: Linux is the chosen operating platform for hosting Draw It or Lose It. It should be selected for its robust security, stability, and commonality in server environments.
* **Operating Systems Architectures** It is advised to adopt a microservices architecture within the Linux environment to enable a scalable and distributed deployment of the application.
* **Storage Management**: Cloud based storage solutions, like AWS S3 or Google Cloud Storage, are recommended to provide the necessary flexibility and scalability for the application's data storage needs.
* **Memory Management**: Memory management can be effectively handled through the implementation of containerization such as using Docker, complemented by orchestration with Kubernetes, to facilitate resource management on Linux.
* **Distributed Systems and Networks**: The application is expected to leverage RESTful APIs or GraphQL for cross-platform interoperability. Network reliability and data transfer efficiency are to be prioritized.
* **Security**: Security measures including the use of HTTPS, secure authentication with OAuth 2.0, and routine security audits are recommended to safeguard user information on varying platforms.