

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 | 03/18/2023 | Jordan Schutte | First commit |

**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 wants to develop a web-based version of their existing Android app team-based drawing game, "Draw It or Lose It". The new web-based version will be able to serve all platforms.

**Requirements**

*Serve multiple platforms.*

*Ability for a game to have one or more teams involved with unique names for games and teams.*

*have only one instance of the game in memory at any given time.*

**Design Constraints**

Develop the game application in a web-based environment.

Ability to accommodate multiple users concurrently.

Each game will have the ability to have one or more teams involved and each team will have multiple players.

Game and team names must be unique.

Only one instance of the game can exist.

**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 Entity class is the handler for Game, Team, and Player classes. From those three classes they inherit characteristics from the Entity class. The ProgramDriver class is how the project is built. From the ProgramDriver class the program can access all the classes created and execute them. There is a SingletonTester which allows the program to run according to the design constraints of only having one instance at a time.



**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 reliable platform for server-side development, with support for popular server-side technologies like Node.js, Django, and Ruby on Rails. Its Unix-based foundation offers stability and ease of use. However, Mac hardware can be expensive, which might not be ideal for setting up large-scale server deployments. | Linux is a popular choice for server-side development, due to its open-source nature, cost-effectiveness, and wide support for server-side technologies like Node.js, Django, and Ruby on Rails. Linux offers excellent stability, security, and flexibility, which makes it suitable for deploying servers to meet the requirements of the web-based game. | Windows supports various server-side technologies, such as ASP.NET, Node.js, and Django. It is known for its user-friendly interface and wide range of applications. However, Windows may not be as cost-effective as Linux for server deployments, and it can be less secure than Linux or Mac due to its larger user base and target for cyber-attacks. | Mobile devices are not typically used for server-side development, as they are designed for client-side use. However, cloud-based services like Firebase, AWS, and Azure can be utilized to provide server-side functionality for mobile applications, ensuring that the web-based game runs smoothly on mobile platforms. |
| **Client Side** | Mac offers an excellent development environment for client-side technologies such as HTML, CSS, JavaScript, and popular web frameworks like React. Web applications developed on Mac will run smoothly on major browsers and are generally compatible with other platforms. However, Mac's limited market share may require extra testing for cross-platform compatibility. | Linux is a flexible platform for developing client-side applications, offering support for HTML, CSS, JavaScript, and popular web frameworks like React. Although Linux's market share is small, web applications developed on Linux are typically compatible with other platforms. Developers must ensure that their applications work well on a variety of browsers for the best user experience. | Windows provides a robust environment for client-side development, with support for HTML, CSS, JavaScript, and popular web frameworks like React. Web applications developed on Windows are generally compatible with other platforms, but developers need to ensure that their applications run smoothly on various browsers for optimal user experience. | For mobile devices, WebView-based apps, React Native, and Flutter can be used for client-side development. These technologies allow web applications to run smoothly on mobile platforms and offer a native-like experience. Developers need to ensure that the web-based game is responsive and compatible with various mobile devices to cater to the diverse user base. |
| **Development Tools** | Mac offers a variety of development tools, such as Git, GitHub, Bitbucket, Jira, and Trello. These tools are essential for efficient development, collaboration, and project management. The availability of these tools on Mac ensures that the development process for the web-based game will be streamlined and well-organized. | Linux also provides access to essential development tools like Git, GitHub, Bitbucket, Jira, and Trello. These tools facilitate efficient development, collaboration, and project management, ensuring that the web-based game's development process remains organized and efficient on the Linux platform. | Windows supports development tools like Git, GitHub, Bitbucket, Jira, and Trello, which are crucial for efficient development, collaboration, and project management. Using these tools on Windows will ensure a well-organized and streamlined development process for the web-based game. | Mobile devices can utilize development tools such as Git, GitHub, Bitbucket, Jira, and Trello, facilitating efficient development, collaboration, and project management. |

**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**: I think utilizing a cloud-based platform such as Amazon Web Services or Google Cloud Platform would offer scalability, flexibility, and cost-effectiveness, which would allow the game to run efficiently on various devices.
* **Operating Systems Architectures**: The cloud platform should support various operating systems, such as Linux, Windows, and macOS, which would enable the game to run on a wide range of server environments.
* **Storage Management**: For storage management, I recommend using a managed storage service like Amazon S3 or Google Cloud Storage. These services provide reliable, scalable, and secure storage options that can accommodate the large image library required for Draw It or Lose It.
* **Memory Management**: Cloud platforms like AWS and GCP provide memory management features, such as auto-scaling and load balancing, which can help ensure that the game runs efficiently and effectively on all supported devices. The platform should also support garbage collection and caching mechanisms to optimize memory usage and performance.
* **Distributed Systems and Networks**: The game application can be designed using microservices architecture and RESTful APIs, which can be deployed and scaled independently across distributed systems.
* **Security**: To protect user information on and between various platforms, the chosen cloud platform should offer robust security features such as data encryption at rest and in transit, access control, and identity management. Implementing proper authentication mechanisms such as OAuth 2.0.