

<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 | <mm/dd/yy> | <grant-sorenson> | <Added support for web accessible devices.> |

**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 game company wants to introduce a web-based game that’s available on multiple platforms. The developers are unsure about how to properly set up the environment for development. In order to solve this problem, the developers will need a detailed explanation about how to set up their environment including version control and platform emulators. The development team will also need experience in programming cross platform web-based games and an experienced UI designer to handle the user interface.

**Requirements**

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

**Design Constraints**

The main known design constraint Is developing a web based game for multiple platforms. The size of the screen of the device will be variable, so UI design will be paramount for a smooth user experience. The speed at which devices can process information will also be variable. If two people are playing on opposite ends of the planet there will be data lag, so choosing what to expose to the other player will also be a design decision based on the capabilities of the app.

**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 game service class houses the gameplay state logic as well as relevant methods that can alter game instances. The game,team, and player classes inherit from the Entity class, and contain a unique identifier for locating each instance of an object as well as relevant object details. The program driver can be used to test the program and the Singleton Tester creates a single game instance to manipulate.



**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 OS’s architecture allows free built in server application for web based software. Server services are available out of the box and provide plenty of documentation. | Linux is able to host web-based servers easily. The Linux framework is very versatile and provides the programmer with complete control for advanced integration. | Windows can host web-based servers but Linux is far more common. Windows services are also considerably more expensive than Linux hosting options. | While hosting web based servers is possible from mobile devices, its simply not realistic. They will have limited functionality and wont be able to achieve the same performance as a desktop. |
| **Client Side** | Mac OS operates at different versions on different devices. The OS for the mac laptop will be slightly different for the desktop, but the underling functionality will be similar. Apple devices are known for being compatible, so development should be straightforward. | The programmers will need knowledge in Linux and time to set up and configure the web server. While services can be bought for hosting, they could acquire their own machines and setup a domain to host their websites and servers. | The programmer s will need experience with .NET framework and enough time to get the site up. Windows server hosting is not free and while windows provides paid services it will add to the total cost. | The programmers will need expertise in mobile development and server based mobile application. Most services are paid so that will be another cost to the total. Because mobile based OS are less functional it will take longer to implement. |
| **Development Tools** | The programmer will need whatever favored IDE, and the Mac OS should come with Apache, the software used to launch the server. Programmer is done in html so experience is needed. | Like Mac, Apache is utilized for server connection as well as mySQL for database needs and PHP for executing applications. Programmers will need experience with Linux OS. | Programmers will need experience with web development software like windows VB, as well as .NET framework. Preferred IDE can be used. | The server will most likely take only PHP script or HTML files. Developers may be limited in IDE and may find speedier development time with an android OS emulator. |

**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 Recommend using Linux as the primary operating platform for the webserver. It is free to use and setup and has a large user/developmental base which makes finding experts easy.
* **Operating Systems Architectures**: While Linux has multiple characteristics, its known for having a monolithic OS structure. This means the entire process runs in the kernel and is dependent on all its processes running smoothly, or else the overall process will crash. Even though this can make the system seem brittle, it gives the user full control over security and memory management.
* **Storage Management**: I would recommend using a combination of cloud and physical management system. This would give the Gaming Room the freedom of using redundant process servers in the event the primary process fails on one, while also having 24/7 access to data for the game housed in a cloud system.
* **Memory Management: Linux operates processes within the kernel for internal structures and user space programs.** Memory management is complex, consisting of processes utilizing physical and virtual memory with pointers along with caches and buffers that hold and purge dirty data, respectively. TGR (The Gaming Room) developers will be able to control these systems and provide adequate enhancements when extraneous data needs arise.
* **Distributed Systems and Networks**: TGR should use a Restful API for server-based communication and to facilitate communication between devices. This way, rules can be established for valid communication to prevent corruption of the system. Outages and network performance can be handled with connection pings and proper safeguards can be established for users who lose connection to their network. TGR only wants valid users sending requests to the server, so implementing authorization and authentication is important for maintaining security of the system.
* **Security**: Linux runs it's processes in the kernel, which uses RAM for memory. The user can edit RAM and corrupt the system by sending bad requests, falsifying data, even implementing custom functionality not intended by the developers. They are also vulnerable to attacks on their web server. Despite these vulnerabilities, TGR developers can still protect their application. Using proper encapsulation and abstraction techniques will allow them to successfully hide data from bad users and maintain the security of their codebase. Utilizing Lunix, they will have complete control over the security of the webserver and can implement proper checks to authenticate and authorize users to send requests and access data.