

<Name-of-Software-Application>

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 5/21/23 | Christian Cromer | First Revision |

**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 is wanting to transition their already existing app on Android OS to a web based game that can be used on multiple platforms. I think the best way to develop this app is my launching the web based application on a linux based server due to cost being lower and due to linux being very optimized for this kind of project.

**Requirements**

Must run on multiple platforms while all platforms being to communicate and use the applications at once.

**Design Constraints**

Different development kits for the different OS platforms.

The API should be able to plug into all platforms.

Sharing of data between platforms.

storage for images and game information

must be web based

Unique istances of game and teams

**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.>

**We can Start with the Entity class and how it is parent class to 3 other classes game, team, and player. Each one of these is an enitity which will have an id and a name. Doing this instead of each class having to state their own name they would also have to use boiler plate code and repeat lines of got for getting names, ids and contructors. Then we go into game service and how it is related to the classes game, team and player. Game service is going to be the most important class here because it is going to hold List of games in it. This is important because the Game class holds the list of teams in the game, and then in turn team holds all the list of all the players on a given team. Of couse all the values are unique. All together this helps form an object from other objects which, this UML shows why Object Oriented Design can be so useful in these type of applications.**



**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** | The benefits include being able to easily integrate Into IOS apps and and Mac os Applications, but it does not scale well and requires mac to develop the server. | linux is ideal for hosting web based servers. They remove a lot of the bloat and with good customizability it has a lot of advantages. The downside is the expertise of the development team to be able to develop the server, but once launched, low cost. | Windows advantages is that it is a very common way to launch servers and has many options to do so just like linux. I think this is also a very viable option, the downside is up cost and setup. | Could be very efficient due to not needing a lot of hardware power for the application, but not very reliable as if battery died on the mobile device being used it could shut down the whole server and the scaleability of the mobile storage is not great. along with the speed of the processing and the memory would not be great with performance for the game once it is launched. |
| **Client Side** | Mac is not great for developing for other OS's but can do the job for web based applications, It can take some more advanced set up but setting up a java environment and launching on a spring boot or tomcat server is doable, but just expensive as it scales and the software and hardware for Mac is expensive. | It is going to take more time and expertise to develop on the linux platform but due to the low cost I think this could be a very viable option. The OS itself is free and the hardware requirements are low. | The time would be really low for development here and the cost would be average and most developers are comfortable on windows for development. | You would need to have an expert in mobile development for this to really work and the time would be quick. The cost would be low due to the mobile device sometimes not costing near as much as enterprise level computers. |
| **Development Tools** | Swift and the CodeX is going to be used for developing. You can download a JDK and Java onto the machine but that requires advanced set up and is not native to mac. | There are many options for linux for development tools since it is an open source OS. It almost always take more knowledge to set this up but it gives many options such as netbeans or inteliJ for IDE and python or Java for the language used to develope the application | Using Java with VS Code or inteliJ would probably be the best options for this game program. Scalability is great with windows just as it is linux. | depending on IOS or android os you would use different apps and IDE's to develope. This would not be a very viable option since it is limited and usually meant to develope apps for that specific platform not web based applications. |

**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 would recommend Windows.
* **Operating Systems Architectures**: Windows has a modular structure made of modules, hardware Abstraction,Kernel, Executive Services, Environment Subsystem, and the Integral subsystem. There can me modes to this system which are Kernel and user. Kernel mode allows direct access to CPU and Memory while the user mode has many things locked off for security reasons and to prevent crashes that are fatal to the system. The reason for the modular layers to the System is mostly for security and efficiency. With out the HAL layer we would only be able to run one program at a time. The kernel layer is important for scheduling of task and priority of tasks to be run by the HAL.
* **Storage Management**: I would recommend a SQL server for this storage management system. It can handle storage of all of our users game data, and audio files, while keeping a secure backend especially if hosted on a known service as azure or AWS that have built in security measures for their data bases.
* **Memory Management**: It would block out a section of the memory required for the server or program and use physical and virtual memory to keep this steady and limit the ram usage so not to crash the application. It does this by only allocating what is needed and also limiting other programs memory use if the game would need more.
* **Distributed Systems and Networks**:

I would distribute on a Restful Spring Boot application so that any of the platforms with most browsers would be able to access the application. It would use HTTP protocols which are very customizable for security and access. We could also look to host the system on a service such as Azure or AWS which have a very reliable network and systems and redundancy built in that would save from outages for the game.

* **Security**: Windows has built in security protocols for network access and but if the program was developed using Restful Spring Boot Application using Oauth2 Security protocols, this should prevent most security risks to information of client or business.