**Memory and Storage Management**

The Game Room's project 'Draw It or Lose It' requires exchanging and managing data on multiple platforms. The Gaming Room will have 200 high-definition images that it will serve, all of which are approximately 8 megabytes in size. The challenge is to find a solution in terms of memory (RAM) and storage. The main difference between memory and storage is that memory (RAM) is volatile, and the RAM is temporary storage for data while the CPU executes other tasks on the fly. To compare the RAM and CPU relationship to the human brain, RAM would be much like keeping an idea or a memory in the "back of your mind" for implementation later. The CPU would be your current state of thought and all the processing your brain is doing in the current moment. RAM is volatile because it will not be saved if no power is sent to the components. A more practical example would be when you are typing characters into your word document, the information is saved but in a temporary state because the Operating System is actively allocating resources to make it possible to edit the file. That is why it's important to save data. Storage is non-volatile and can retain data without any power. Until you click on the ol-floppy disk or hit ctrl+s, your data will not be saved!

The first factor that should be considered is memory allocation and management. Knowing the specific amount that the client will need to properly utilize the service will be essential to the application's success. Since the game will be cross-platform, we must make considerations for each device. Looking towards the lower-end devices such as mobile, there are mobile devices on the market that are cheap and have a minimum of 2 gigabytes of RAM, while you can look at a higher-end mobile device capable of having 16 gigabytes of RAM. These days, the most common smartphones come built with at least 4 gigabytes of RAM and efficient processing power. However, mobile users do not have the same capability as those playing on a console or PC, so accommodations must be met for these players. PC and console players have the capabilities of multi-core processors and expansive RAM slots. With that in mind, this game was initially developed on Android devices, so its capabilities may not be hindered if it was ported to PC/Console; however, Android might be affected if they began making more additions and updates that require more memory.

Memory allocation in computing is a method that is automated by the machine's operating system. An OS will allocate portions of RAM to each actively running application; however, the application will utilize however much of the RAM is provided for whatever resources it needs. Managing your memory effectively within the applications you develop will reflect its performance on the system. The game will have two hundred files each, 8 megabytes in approximate size which will be rendered over thirty seconds at a fixed rate. The solution in this scenario would be to maintain the performance of the application, and this could be done by loading the chosen files into the system cache. The cache is a unique storage space for temporary files that aids in running applications. This will allow for the fastest rate of data transfer. This method will ensure that the processing units do not stand-by waiting for further instruction from the application. All images that are not in-use will not be loaded into the system memory because they are not presently necessary to the function of the application. Once the game instance is loaded, the image should be wiped from the memory to free space for other resources requiring memory. Another way to create an efficient user experience would be to load images into memory during each round, if necessary. Since the game is multiplayer, there will be so many variations and different turn-outs for each round, so this must be taken into account for the organization factor. It may be possible that a team will not guess the right pictures within the time limit, but the image will be displayed to the rival team.

We must consider the practicality of storage and how two hundred images will be stored within a file server. Cloud storage has the potential to be a solution for this problem, but it will have to be reworked into the client's budget, yet it is not expensive as a physical network server. With a physical server, you also have the responsibility of maintaining the hardware, and most of the time, these servers do not last very long, so eventually, they will need to invest in better equipment, either that or future-proof their systems which will be even more costly. Cloud storage is typically subscription-based and can be upgraded very easily. This will be the simplest solution once the game becomes more extensive and the client's demands grow. If Draw It or Lose It were to remain on a physical server or switch to the cloud, the game will require data to be encoded on the storage device it's installed on. This means all of the application files for usernames, the data associated with each player, network communication, graphical user interface, and much more. The amount of data in the image files alone is a gigabyte and a half, so to put that in perspective, the game will need enough data to store itself and the accumulation of data from user input. However, the images may be able to be contained by a cloud service or compressing the images to a smaller size.

As mentioned in the first paragraph in my examples, RAM is temporary storage, whereas something like a hard disk or the cloud would permanently store the data. When a developer has to make changes on the spot, a developer can add elements to the stack and manipulate the stack as they please, and these changes will only be stored in the RAM until the changes are saved. Once the data is saved, it will be retrieved from the storage and pulled for future cases. From the user's end, a user will log back into the game using their credentials stored in the servers. As the player continues to play, the software will require data to be sent to the user, and the user might change their password, or their stats for the number of games played will increase requiring the data to be sent back to the server. It would be optimal if little data was saved onto the user's device in this case.