Raynaldo Young

CS-230

**Module Six Assignment - Draw It or Lose It**

**Context**

Draw It or Lose It is moving from Android only to a web version that runs on desktop and mobile. The image set is two hundred files at about eight megabytes each, which is about one point six gigabytes in total. The game shows drawings on a fixed timeline, so players need steady, smooth updates across platforms.

**What considerations and specific approaches would it take to ensure that memory is effectively managed in Draw It or Lose It?**

Keep the working set small on the client. Hold the image on screen and the next image, then release the previous one when the round ends. Preload the next image during idle time, and use versions that match the screen, so the browser is not holding extra pixels. Draw on the browser animation loop so updates feel steady, and clear timers and event listeners after each round to avoid slow memory growth.

On the server, avoid buffering entire files in memory. Stream files to the client, keep only a small cache of frequently used images, and set normal cache headers so browsers and the content delivery network can reuse files. Track memory use and set simple limits for caches so the system stays predictable when many games are active at once. If traffic grows, scale the number of stateless servers or handlers rather than increasing per process memory.

**What considerations and specific approaches would you take to determine how much storage is needed and how to manage storage for Draw It or Lose It?**

Start with the originals, which are about one point six gigabytes. Add a few web friendly sizes for desktop and mobile plus a small thumbnail, which usually brings the total image footprint to about three to four gigabytes. Add room for puzzle data, team and match records, logs, and backups. A starting target of five to ten gigabytes leaves headroom and is easy to grow.

Store images in cloud object storage for durability and cost control and serve them through a content delivery network for speed. Use a clear folder and naming pattern such as originals, desktop, mobile, and thumbs. Turn on lifecycle rules to move old or rebuildable copies to cheaper tiers after some time. Back up the originals and the database on a schedule and monitor storage growth on a regular cadence.

**What are the differences in how memory and storage are used in terms of the game application functionality?**

Memory is the live work area. It holds only what is needed right now so the round feels responsive, usually the current image and the next one. Storage is the long-term library. It keeps the full image set, resized copies, game records, logs, and backups organized and safe. In simple terms, memory supports speed during play, while storage preserves assets and data over time. Managing both cleanly lets the game feel quick without wasting resources.

**References**

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating system concepts* (10th ed.). Wiley.  
freeCodeCamp. (n.d.). *What is serverless architecture? What are its pros and cons?* <https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/>

Microtica. (n.d.). *Microservices vs. SOA: Is there any difference at all?* <https://medium.com/microtica/microservices-vs-soa-is-there-any-difference-at-all-2a1e3b66e1be>

Microsoft. (n.d.). *Windows Server*. <https://www.microsoft.com/en-us/windows-server>

Microsoft. (n.d.). *Windows for developers: What’s new in Windows 11 (build 22000).* <https://learn.microsoft.com/en-us/windows/apps/whats-new/windows-11-build-22000>

Chapel, J. (n.d.). *Cloud storage cost comparison: AWS vs. Azure vs. Google.* <https://jaychapel.medium.com/cloud-storage-cost-comparison-aws-vs-azure-vs-google-844dfff3d324>