Task1 : Client-to-Client File Transfer Project

Objective:
Develop a simple file transfer system that allows **Client A** to send a 1 GB file to **Client B** using chunking, parallel transfer, and data integrity checks.

Requirements

1. **Client A and Client B Setup**
    - Implement **Client A** (sender) and **Client B** (receiver) to enable direct file transfer.
    - Client A will handle file partitioning, sending chunks, and manifest creation.
    - Client B will handle chunk reception, integrity verification, and reassembly.
2. **File Transfer Implementation**
    - **Chunking Process**
        - Partition the 1 GB file on Client A into 100 MB chunks, creating 10 chunks.
        - Send all chunks concurrently to Client B to optimize transfer speed.
    - **Data Integrity Check**
        - For each chunk:
            - Calculate the MD5 hash at Client A before transmission.
            - Upon receiving each chunk at Client B, verify the MD5 hash to confirm integrity.
            - If the MD5 hash does not match, log the error and trigger re-transmission of the specific chunk.
    - **Manifest and Reassembly**
        - Before starting the transfer, generate a *manifest file* on Client A:
            - This file should describe the chunk order and expected MD5 hashes.
            - Send this manifest as the first packet to Client B to guide the reassembly process.
        - Client B should reassemble the file using the manifest, verifying each chunk's integrity as it arrives.

Deliverables

- **Source Code** for both clients with comments for function and class clarity.
- **Logs** generated during transfer showing any successful chunk verifications or errors, including re-transmission details if needed.

Task 2 : Task Mgmt Dev task

The task is focused on the ability to run local development environment effectively for a complex application and show skills related to a single back end functionality that our system does. Imagine we have over 200 tasks of this nature in the application, how would you effectively manage the codebase to structure these type of tasks and complex business logic they run.

1. Create a NodeJs App for making and copying 100 files of size 1 GB each to AWS s3 bucket. Our system works on tasks model and lets say you are going to make a task called "**MakeAndUploadFiles**"
    a. MakeAndUploadFiles
        i. Input Params
            1. fileCount
            2. fileSize
            3. S3Destination Path
        ii. Output Param
            1. timeToCopy
            2. fileSize Created
            3. copySpeed in gbps
    b. This task Spawns another Tasks called "**MakeUploadFile**" for the length of fileCount and listen for all tasks to complete **Parallely**

2. You need to envision that initial task (MakeAndUploadFiles) request happens on container1, but then **MakeUploadFile** is spawned to other containers (n)
    a. You need to make the application to accomplish it with RabbitMQ as the queue service
3. Create a docker compose with below services to demonstrate this task
    a. App1
    b. App2 (with scale factor)
    c. rabbitMq
    d. <any other things> you may need.
4. Most Importantly, I would like you to do a docker based debug/development, i.e. you make your code in your IDE such as VSCode, but debug (step through) in IDE while running code in docker containers. Lot of times we have n number of non node libs that we had install while they work great on linux images, we may not have those libs in mac or windows.

Deliverables:

1. Code for this project for review
2. Video of the runtime execution demo of app

Resources:

- AWS S3 bucket : mast-dev-intw-test01
- AWS Region: US West (Oregon) us-west-2
- Amazon Resource Name (ARN) : arn:aws:s3:::mast-dev-intw-test01

Task 3: Create a Video with iMovie and Document Key Details

Objective:
Use iMovie to create a short video by stitching together several favorite video clips and adding background audio. Document the process and key details to understand how a video editing program could automate this.

Steps

1. **Create the Video**
    - **Collect Clips**: Choose 4 video clips, such as family or friends moments.
    - **Edit in iMovie**:
        - **Stitch the Clips Together**: Combine the clips in a way that creates a smooth flow.
        - **Add Background Audio**: Overlay an audio track that complements the clips. Adjust audio levels for clarity.
        - **Adjust Transitions**: Use fade-ins, fade-outs, or other transitions to make the video seamless.
        - **Check Sync and Alignment**: Make sure audio aligns with key points in the video and that transitions look natural.
2. **Review Key Editing Elements**
    - **Clips Information**: Note the order, start and end times, and any transitions used.
    - **Audio Details**: Record start and end times, volume adjustments, and any key sync points.
    - **Effects and Enhancements**: List any filters, color adjustments, or additional visual effects.
3. **Summarize the Process**
    - Provide ER diagram which captures all the steps that how you edited and aligned the clips and audio which should help us build a system that can perform the same operation in an automated way.