**Project Title:** Photo Backup and Organizer

**What is the goal of your script?**  
The goal is to automate the backup and organization of image files by sorting them into directories based on the date they were taken (from EXIF metadata or file modification date). It optionally compresses the backup for long-term storage.

**Who is the intended user?**  
The intended users are individuals who regularly manage digital photos, including students, bloggers, hobbyists, and professional photographers. Users are expected to have basic familiarity with the terminal environment on Linux or macOS systems.

**What does it automate or solve?**  
It automates the time-consuming task of manually sorting and organizing photos. It solves issues like inconsistent file naming, lost image timestamps, unorganized photo libraries, and prevents accidental overwrites by renaming duplicates. It also adds an optional compression feature for easier archival.

**2. Design Rationale**

**EXIF Date-Based Sorting**

**Design Decision:** Used exiftool to extract the DateTimeOriginal tag.  
**Why:** Provides an accurate record of when a photo was taken.  
**Alternative Considered:** Using stat to retrieve file modification time. Rejected because modification times can change when copying files, which may not reflect the actual photo date.

**Folder Organization by Date**

**Design Decision:** Photos are placed in backup/YYYY/MM/ folders.  
**Why:** Improves organization and searchability of photo collections. Makes it easy to find photos from a specific year or month.  
**Alternative Considered:** A single folder with renamed files. Rejected due to scalability and user inconvenience.

**Duplicate Filename Handling**

**Design Decision:** If a file with the same name exists in the destination, the script renames the new file using a timestamp.  
**Why:** Prevents overwriting original files while maintaining all versions.  
**Alternative Considered:** Hash comparison to check for actual content duplicates (using md5sum). Rejected due to added complexity not necessary for most use cases.

**Optional Compression**

**Design Decision:** Add --compress option to generate a .tar.gz archive of the backup.  
**Why:** Simplifies storage or transfer of large photo collections.  
**Alternative Considered:** Always compress or use .zip. Rejected to give user control and because .tar.gz is more efficient in Unix-based systems.

**Use of exiftool**

**What:** A command-line tool for reading and writing image metadata.  
**Why Needed:** Bash alone cannot parse binary EXIF metadata from images. exiftool offers reliability and wide format support.  
**Why Built-In Bash Was Insufficient:** No native Bash tool can parse EXIF headers without external utilities.  
**Verification:** Compared results of exiftool with stat. Also tested with various JPEGs and PNGs to confirm accurate date extraction.

**3. Execution Instructions**

1. Open a terminal.
2. Ensure exiftool is installed:
   * On Ubuntu/Debian:
   * sudo apt install libimage-exiftool-perl
   * On macOS:
   * brew install exiftool
3. Give the script execute permission:
4. chmod +x 12345678\_script.sh
5. Run the script:
6. ./12345678\_script.sh photos/ backup/
7. Optional: Run with compression
8. ./12345678\_script.sh photos/ backup/ --compress

**4. Testing Summary**

**How did you test your script?**

* Created a photos/ folder with .jpg, .jpeg, and .png files, including:
  + Images with EXIF data
  + Images without EXIF data
  + Images with duplicate names
* Ran the script in various scenarios:
  + Normal run
  + With compression
  + With a missing source folder

**What edge cases did you consider?**

* Missing EXIF data
* Duplicate file names
* Non-image files
* Empty directories
* Incorrect input arguments

**Were any limitations or bugs identified?**

* Limitation: Only supports common image types (.jpg, .jpeg, .png).
* Limitation: Only processes images in the top level of the source folder (no recursion).
* Bug: None identified in the final version.

**5. Reflection**

**What was challenging?**

* Making the script work both with and without EXIF metadata.
* Handling duplicate files without overwriting.
* Ensuring compatibility across systems with different versions of core utilities.

**What would you improve or do differently next time?**

* Add recursive search to include subfolders.
* Include logging and progress bar.
* Support more formats (e.g., RAW, HEIC).
* Add GUI prompts with tools like zenity.
* Use md5sum or similar to detect true duplicates by file content.