Google Photos Tagging Assistant - Desktop App Project

Project Overview

I need a **desktop application** that serves as a semi-automated assistant for adding names to photo descriptions in Google Photos. Instead of full automation, this tool will provide an efficient UI that streamlines my manual tagging process while maintaining accuracy and control.

Current Situation

- I have thousands of photos in Google Photos that need people's names added to descriptions
- Currently at year 1998 with thousands more to go
- Manual process is slow: navigate to photo, click info, type name(s), save, navigate to next photo
- Many photos have multiple people in them
- I need to maintain accuracy while dramatically improving speed

Solution: Semi-Automated Desktop Assistant

A Python desktop application that:

1. Controls a Chrome/Edge browser window showing Google Photos
2. Provides a separate, clean UI window for me to work in
3. Makes tagging fast and easy with clickable name buttons
4. Learns and remembers names as I work

Core Functionality

Two-Window System

Window 1: Assistant UI (My control panel)

- Clean, simple interface I interact with
- Shows current photo information
- Displays pre-populated name buttons
- Has input field for new names
- Navigation controls (Next, Previous, Skip)

Window 2: Automated Browser (Google Photos)

- Chrome/Edge controlled by the app via Selenium/Playwright
- Displays Google Photos website
- App automatically navigates and enters data here
- I can watch it work but don't interact with it directly

Starting Point Configuration

- Ability to specify starting date or photo (e.g., "12/1/1998")
- Navigate to specific face group or date range
- Resume from last session automatically

Name Management System

Pre-populated Names:

- Display as clickable buttons in the UI
- Stored persistently (survive app restarts)
- Sorted by frequency of use (most-used names at top) or alphabetically
- Allow multiple name selection for group photos

Adding New Names:

- Text input field for names not in the list
- When new name is added:
  - Immediately adds to current photo's description
  - Adds to pre-populated name list for future use
  - Becomes available as clickable button for subsequent photos
  - Persists to storage for future sessions

Name List Features:

- Search/filter functionality if list gets long
- Ability to edit or remove names from the list
- Visual indication of which names have been selected for current photo
- Clear selection button (start over on current photo)

Tagging Workflow

For Each Photo:

1. App loads photo in Google Photos browser
2. Assistant UI shows:
   - Photo thumbnail or identifier (date/time)
   - Current description (if any exists)
   - All pre-populated name buttons
   - Input field for new names
3. I click one or more names (for people in the photo)
4. Or type a new name and click "Add New Name"
5. I click "Save & Next" or "Next"
6. App writes the selected names to Google Photos description field
7. App navigates to next photo
8. Repeat

Description Format:

- Names should be added as comma-separated list (e.g., "John, Mary, Susan")
- Option to append to existing description or replace
- Configurable prefix/suffix (e.g., "People: John, Mary")

Progress Tracking

- Display current photo number / total photos
- Show date range being processed
- Estimated photos remaining
- Session statistics (photos tagged this session, total time)
- Auto-save progress to resume later

Navigation Controls

- **Next**: Save current tags and move to next photo
- **Previous**: Go back to previous photo (to correct mistakes)
- **Skip**: Move to next without saving (photo already tagged, uncertain, etc.)
- **Jump to Date**: Quick navigation to specific date
- **Pause/Resume**: Stop and continue later

Technical Requirements

Platform

- **Primary**: Python 3.8+
- **UI Framework**: Tkinter (built-in) or PyQt5 (more polished)
- **Browser Automation**: Selenium or Playwright
- **Operating Systems**: Windows and macOS

Browser Control

- Use Chrome or Edge (Chromium-based)
- Headful mode (visible browser window)
- Handle Google authentication (I'll log in manually first session)
- Navigate Google Photos interface
- Locate and fill description/info field
- Handle dynamic content loading

Data Persistence

- **Name List**: Store in local file (JSON or SQLite)
- **Progress**: Save current position (date, photo index)
- **Session History**: Track what's been processed
- **Settings**: Starting date, description format preferences

Error Handling

- Gracefully handle network issues
- Detect if Google Photos layout changes
- Alert me if can't find description field
- Log errors for troubleshooting
- Don't lose data if app crashes

User Interface Design

Main Window Layout

```
┌─────────────────────────────────────────┐
│ Google Photos Tagging Assistant         │
├─────────────────────────────────────────┤
│                                         │
│ Current Photo: 12/15/1998  [Photo #: 145] │
│ Existing Description: [shows any text]  │
│                                         │
│ ┌─────────────────────────────────────┐ │
│ │ Selected Names: John, Mary          │ │
│ └─────────────────────────────────────┘ │
│                                         │
│ Pre-populated Names (click to select):  │
│ ┌─────────────────────────────────────┐ │
│ │ [John] [Mary] [Susan] [Tom]         │ │
│ │ [Linda] [Mike] [Sarah] [Bob]        │ │
│ │ [David] [Karen] [Paul] [Lisa]       │ │
│ └─────────────────────────────────────┘ │
│                                         │
│ Add New Name: [____________] [Add]      │
│                                         │
│ [Clear Selection]  [← Previous] [Skip]  │
│ [Save & Next →]                         │
│                                         │
│ Progress: 145/3,247 photos (4.5%)       │
│ Session: 23 photos tagged               │
└─────────────────────────────────────────┘
```

Settings/Configuration Window

- Starting date/photo location
- Description format (prefix, suffix, separator)
- Name list management (edit, remove names)
- Browser selection (Chrome/Edge)
- Face group selection (optional)

Required Features Checklist

Core Functionality

- [ ] Two-window system (UI + automated browser)
- [ ] Navigate to specific starting date/photo
- [ ] Display pre-populated clickable name buttons
- [ ] Support multiple name selection per photo
- [ ] Add new names on-the-fly
- [ ] Write names to Google Photos descriptions
- [ ] Navigate between photos (Next/Previous/Skip)

Name Management

- [ ] Persistent name list (survives restarts)
- [ ] Add new names to list automatically
- [ ] Sort names by usage frequency or alphabetically
- [ ] Search/filter name list
- [ ] Edit/remove names from list
- [ ] Visual feedback for selected names

Progress & State

- [ ] Save progress automatically
- [ ] Resume from last position
- [ ] Display progress statistics
- [ ] Session tracking
- [ ] Jump to specific date

User Experience

- [ ] Clear, intuitive interface
- [ ] Keyboard shortcuts (Enter = Save & Next, etc.)
- [ ] Visual confirmation of actions
- [ ] Error messages that make sense
- [ ] Help/instructions within app

Reliability

- [ ] Handle network interruptions
- [ ] Detect Google Photos layout changes
- [ ] Don't lose data on crash
- [ ] Log errors for debugging
- [ ] Validate data before writing

Success Criteria

1. **Speed Improvement**: Can tag 200-400 photos per hour (vs ~50-100 manual)
2. **Accuracy**: I maintain full control over tagging decisions
3. **Ease of Use**: Intuitive enough to use without constant reference to docs
4. **Reliability**: Runs for hours without crashing or losing data
5. **Resumability**: Can stop and start across multiple sessions
6. **Name Learning**: Builds comprehensive name list as I work
7. **Flexibility**: Can handle group photos, corrections, skips easily

---

Implementation Priorities

Phase 1 (MVP - Minimum Viable Product)

1. Basic two-window setup
2. Navigate to starting date in Google Photos
3. Display simple UI with name buttons
4. Click name → write to description → next photo
5. Add new name functionality
6. Basic progress tracking

Phase 2 (Enhanced)

1. Multiple name selection
2. Previous/Skip navigation
3. Name list persistence
4. Session resume capability
5. Better error handling

Phase 3 (Polish)

1. Name sorting/search
2. Keyboard shortcuts
3. Progress statistics
4. Settings configuration
5. Help documentation

Setup Requirements

User Setup (What I'll do)

- Install Python 3.8+
- Install required libraries (will be provided)
- Install Chrome/Edge and WebDriver
- Log into Google Photos manually (first time)
- Configure starting date/location

Developer Deliverables (What I need)

1. **Complete Python application** with:
   - Main app script
   - UI code
   - Browser automation code
   - Data persistence code
2. **Installation guide** (step-by-step for Windows/macOS)
3. **User manual** (how to use the app)
4. **requirements.txt** (Python dependencies)
5. **Troubleshooting guide** (common issues)

Google Photos Interaction Details

Initial Setup & Manual Control

- **Browser Window 2**: User must be able to type/click in this window for:
  - Initial Google sign-in
  - Manual selection of first photo to start from
  - Manual fixes if something goes wrong
- App should not lock out manual interaction with the browser
- User navigates to photos.google.com and signs in manually
- User manually selects the first photo they want to start tagging from

Photo Information Panel Workflow

Step 1: Opening Info Panel

- User manually clicks the **(i)** info button in Google Photos
- This opens the right panel showing photo details
- Description field appears at the top of this panel

Step 2: Reading Existing Description

- App reads current content of description field
- If **blank**: App can directly add names
- If **populated**: App must give user options:
  - **Replace** existing content with new names
  - **Append** new names to existing content (e.g., "old text, John, Mary")
  - **Skip** this photo (leave description as-is)
- Display existing description in Window 1 UI for user review

Step 3: Updating Description

- User selects name(s) from Window 1
- App writes selected names to description field in Google Photos
- Format: Comma-separated (e.g., "John, Mary, Susan")
- If appending: Add separator between old and new content

Step 4: Navigation to Next Photo

- User clicks **"Next"** button in Window 1 (UI)
- App clicks the **>** (next arrow) button in Google Photos
  - Located on the right middle side of the photo
- App waits for next photo to fully load
- Info panel remains open (should persist)
- Repeat process

Navigation Between Photos

- **Next Photo**: Click **>** arrow on right middle of photo
- **Previous Photo**: Click **<** arrow on left middle of photo (if needed)
- Info panel **(i)** should stay open between photos
- If panel closes, app should detect and reopen it

Handling Slow Loading Photos

- **User responsibility**: If photos load slowly, user will manually wait/handle
- App should implement reasonable wait after clicking next arrow (2-3 seconds)
- If app tries to interact before photo loads, user can pause and retry manually
- No complex timeout or retry logic needed - keep it simple

Data Storage (Local)

- **All data stored in the program's working directory** (where app is run from)
- **Name List**: `names.json` - stores all tagged names
- **Progress**: `progress.json` - tracks session info and last position
- **Settings**: `settings.json` - user preferences
- Simple JSON files that user can view/edit if needed
- Easy to backup (just copy the directory)

Detecting Layout Changes

- **User responsibility**: If Google Photos changes layout and app breaks, user will report issue
- App should use reasonable selectors but no need for complex fallback logic
- If app can't find elements, it will fail gracefully and user will troubleshoot
- Accept that major Google Photos updates may require app updates

Browser Crash Recovery

- **Manual restart**: If browser crashes, user will manually restart the app
- App should save progress after each photo is tagged
- On restart, app can show last saved progress
- User manually navigates back to where they left off and continues
- No automatic recovery needed - keep it simple

Navigation Method

- **Only navigation method**: Click the **>** (next) arrow button
- Located on right middle side of photo in Google Photos
- **If this doesn't work, the app won't work** - this is the critical dependency
- No alternative navigation methods needed
- App just needs to reliably find and click this > button

Technical Constraints & Preferences

- **No cloud services**: All data stored locally
- **Human in the loop**: I make all tagging decisions
- **Visible process**: I want to see what's happening
- **Safe operation**: No risk of data loss or corruption
- **Standard Python**: Avoid exotic dependencies if possible
- **Simple installation**: Minimal setup steps

Additional Context

- I'm committed to tagging thousands of photos over many sessions
- I have basic technical skills but need clear setup instructions
- Many photos have 2-5 people in them (group shots)
- Some faces won't be in face groups (need manual identification)
- I want this tool to grow with me (name list builds over time)
- Willing to run this for 1-2 hours per session, multiple sessions per week

---

When ready, please provide:

1. Complete setup instructions for Windows and macOS
2. The desktop application code with clear comments
3. Step-by-step user guide with screenshots/examples
4. Troubleshooting section for common issues
5. Instructions for updating name list, changing settings, etc.

APPENDIX: Proof of Concept Test App

Before building the full application, create a **simple proof-of-concept** to validate the core navigation mechanism.

POC Objective

Verify that we can reliably click the **>** (next) button in Google Photos to navigate between photos.

POC Scope (Minimal)

What it does:

1. Opens a browser window (Window 2) to photos.google.com
2. Pauses for user to:
   - Log into Google Photos manually
   - Navigate to starting photo manually
   - Click (i) button to open info panel
3. Opens a simple UI window (Window 1) with single button: **"NEXT"**
4. When user clicks "NEXT":
   - App finds and clicks the **>** arrow in Google Photos
   - Waits 2-3 seconds
   - Ready for another click

What it does NOT do:

- No description reading
- No description writing/modifying
- No name management
- No progress tracking
- No error handling
- Just navigation testing

POC Success Criteria

- Can reliably find and click the **>** button
- Photos advance to next photo when clicked
- Works consistently across 20-30+ photo navigations
- User can manually intervene if something goes wrong

POC UI (Window 1)

```
┌─────────────────────────────────┐
│ Google Photos POC Test      │
├─────────────────────────────────┤
│                     │
│ Instructions:            │
│ 1. Log into Google Photos   │
│ 2. Navigate to first photo  │
│ 3. Click (i) to open panel  │
│ 4. Click NEXT to test       │
│                     │
│   [   NEXT   ]         │
│                     │
│ Clicks: 0              │
│                     │
│   [   STOP   ]         │
│                     │
└─────────────────────────────────┘
```

POC Deliverables

1. **Simple Python script** (poc_test.py)
2. **Setup instructions** (just install Selenium/Playwright)
3. **Quick test guide** (how to run and validate)

Why POC First?

- Validates the critical assumption (can we click >?)
- Quick to build (1-2 hours vs full app)
- Fails fast if Google Photos structure prevents this
- Gives confidence before building full application
- Helps identify any unexpected issues with browser automation

After POC Success

If POC works reliably:

- Proceed with full application build
- Use same navigation logic in final app
- Add all the name management and description features

If POC fails:

- Investigate alternative approaches
- May need different solution architecture
- Saves time before building complete application

POC Request: Please build this proof-of-concept test app first, before starting on the full application.