Ancestry Tree Downloader

CLI Tool Specification

Project Overview

Create a command-line tool that downloads a user's complete family tree from Ancestry.com, including the GEDCOM file and all attached media (photos, documents, records). This tool is for personal data portability - allowing users to backup and preserve their own genealogical research.

Technical Requirements

Language & Stack

Primary: Golang (user preference)
Browser automation: Use chromedp or go-rod for headless Chrome control
CLI framework: Use urfave/cli (github.com/urfave/cli/v2)

Core Functionality

1. Authentication

```
ancestry-dl login --username <email> --password <password>
# Store credentials securely (encrypted local config or keyring)
```

2. Tree Export

```
ancestry-dl export --tree-id <id> --output ./my-family-tree
```

3. Download Process

Phase 1: Export GEDCOM
• Automate clicking through Ancestry's export process
• Download the .ged file
• Parse it to get list of all individuals and their IDs

Phase 2: Media Download
• For each person in GEDCOM:
  - Navigate to their Ancestry profile page
  - Identify all media attachments (photos, documents, records)
  - Download each with systematic naming: {personID}_{mediaType}_{index}.{ext}
  - Save metadata JSON: who, what, when, source URL

Phase 3: Generate Index
• Create manifest.json mapping people to their media files
• Include GEDCOM data + media references

Project Structure

```
ancestry-dl/
■■■ main.go                  # urfave/cli app setup
■■■ commands/
■   ■■■ login.go
■   ■■■ export.go
■   ■■■ list.go
■■■ pkg/
■   ■■■ ancestry/
■   ■   ■■■ client.go       # Browser automation
■   ■   ■■■ auth.go         # Login handling
■   ■   ■■■ gedcom.go       # GEDCOM export
■   ■   ■■■ media.go        # Media scraping
■   ■■■ gedcom/
■   ■   ■■■ parser.go       # Parse GEDCOM files
■   ■■■ storage/
■       ■■■ organizer.go    # File organization
■■■ go.mod
■■■ go.sum
■■■ README.md
■■■ LICENSE
```

CLI Implementation with urfave/cli

Main Application Setup (main.go)

```go
package main

import (
    "log"
    "os"

    "github.com/urfave/cli/v2"
)

func main() {
    app := &cli.App{
        Name:  "ancestry-dl",
        Usage: "Download your family tree data from Ancestry.com",
        Version: "1.0.0",
        Commands: []*cli.Command{
            {
                Name:    "login",
                Aliases: []string{"l"},
                Usage:   "Authenticate with Ancestry.com",
                Flags: []cli.Flag{
                    &cli.StringFlag{
                        Name:     "username",
                        Aliases:  []string{"u"},
                        Usage:    "Ancestry.com email/username",
                        Required: true,
                    },
                    &cli.StringFlag{
                        Name:     "password",
                        Aliases:  []string{"p"},
                        Usage:    "Ancestry.com password",
                        Required: true,
                    },
                },
                Action: loginCommand,
            },


            {
                Name:    "list-trees",
                Aliases: []string{"ls"},
                Usage:   "List all available family trees",
                Action:  listTreesCommand,
            },
            {
                Name:    "export",
                Aliases: []string{"e"},
                Usage:   "Export a family tree with all media",
                Flags: []cli.Flag{
                    &cli.StringFlag{
                        Name:     "tree-id",
                        Aliases:  []string{"t"},
                        Usage:    "Ancestry tree ID",
                        Required: true,
                    },
                    &cli.StringFlag{
                        Name:    "output",
                        Aliases: []string{"o"},
                        Usage:   "Output directory",
                        Value:   "./ancestry-export",
                    },
                    &cli.StringSliceFlag{
                        Name:    "media-types",
                        Aliases: []string{"m"},
                        Usage:   "Media types to download",
                        Value:   cli.NewStringSlice("photos",
                                  "documents", "records"),
                    },
                    &cli.BoolFlag{
                        Name:    "slow",
                        Aliases: []string{"s"},
                        Usage:   "Use extra delays between requests",
                        Value:   false,
                    },
                    &cli.BoolFlag{
                        Name:  "resume",
                        Usage: "Resume interrupted download",
                        Value: false,
                    },
                    &cli.BoolFlag{
                        Name:  "dry-run",
                        Usage: "Show what would be downloaded",
                        Value: false,
                    },
                    &cli.IntFlag{
                        Name:  "delay",
                        Usage: "Delay between requests in seconds",
                        Value: 2,
                    },
                },
                Action: exportCommand,
            },
        },
    }

    if err := app.Run(os.Args); err != nil {
        log.Fatal(err)
    }
}
```

Key Implementation Details

Rate Limiting

• Add configurable delays between requests (default: 2-3 seconds)
• Use --slow flag for extra cautious downloading (5+ second delays)
• Use --delay flag for custom delay timing
• Respect any rate limit responses

Error Handling

• Graceful failure with resume capability
• Save progress state to allow resuming interrupted downloads
• Log all errors with context

User Agent

- Identify as 'AncestryDownloader/1.0 (Personal Data Export)' or similar
- Be transparent about automation

Output Organization

```
output/
■■■ family.ged
■■■ media/
■   ■■■ photos/
■   ■   ■■■ I001_photo_01.jpg
■   ■   ■■■ I001_photo_02.jpg
■   ■■■ documents/
■   ■■■ records/
■■■ metadata/
■   ■■■ manifest.json
■   ■■■ person_I001.json
■■■ logs/
    ■■■ download.log
```

CLI Commands

Login and save credentials:

```
ancestry-dl login --username user@example.com --password secret
```

List available trees:

```
ancestry-dl list-trees
ancestry-dl ls  # alias
```

Export specific tree:

```
ancestry-dl export --tree-id 123456 --output ./backup
```

Export with options:

```
ancestry-dl export --tree-id 123456 \
  --output ./backup \
  --media-types photos,documents \
  --slow \
  --resume
```

Custom delay:

```
ancestry-dl export -t 123456 -o ./backup --delay 5
```

Dry run:

```
ancestry-dl export --tree-id 123456 --dry-run
```

Configuration File

Location: ~/.ancestry-dl/config.yaml

```
credentials:
  encrypted: true

download:
  delay_seconds: 2
  concurrent_downloads: 1
  retry_attempts: 3

output:
  organize_by_person: true
  include_metadata: true
```

Responsible Use Guidelines

Include in README:
1. This tool is for downloading YOUR OWN data only
2. Respect Ancestry.com's Terms of Service
3. Use reasonable rate limiting
4. Do not redistribute downloaded content that includes others' private information
5. This tool automates what you could do manually - it's for convenience and data portability

Testing Strategy

• Test with small tree first (<10 people)
• Verify GEDCOM download works
• Verify media identification works
• Test resume functionality
• Test error handling

Future Enhancements

• Support for other genealogy sites (MyHeritage, FamilySearch)
• Convert output directly to static website
• Generate PDF family book
• Validate downloaded data completeness

Open Source Considerations

• MIT or Apache 2.0 license
• Clear disclaimer about responsible use
• Contribution guidelines
• Document Ancestry.com's page structure (it may change)

Dependencies

```
require (
    github.com/urfave/cli/v2 v2.27.0
    github.com/chromedp/chromedp v0.9.0  // or go-rod
    // other dependencies as needed
)
```

Implementation Instructions

Please implement this CLI tool in Golang with clean, documented code using urfave/cli for the command-line interface. Start with the authentication and GEDCOM export, then build out the media downloading functionality. Include a comprehensive README with setup instructions and usage examples.