gavalion - Project Portfolio Page

Overview

Travel Diary is a desktop application for managing trips and travel memories, designed for use through a Command Line Interface (CLI). It enables users to efficiently log and organize trips, photos, and personal experiences — providing a more structured and customizable alternative to traditional travel journaling apps.

Contributions

💥 New Features

1. Parser

- **What it does**: Parse user input from string to hashmap<String, String> based on the command and tags
- **Justification**: To send data to the commandFactory easier and made the code more modular.
- **Highlights**:
  - Designed the parsing structure and create a skeletal code for others so that they can create command easier.
  - Includes exception handling for invalid tags, duplicated tags and missing tags
  - Reject invalid inputs due to wrong command name
- **Technologies**: Java I/O, JSON serialization

2. Trip Class

- **What it does**: Represents the core data model for a trip, including its name, description, and associated photo album.
- **Highlights**:
  - Added missing compulsory parameter error to detect missing input.

3. TripManager Class

- **What it does**: Manages a collection of `Trip` objects, including logic for adding, removing, and listing trips.
- **Highlights**:
  - Add mechanism where the trip can not be added to trip manager due to duplicated trip name

4. Photo Class

- **What it does**: Represents the core data model for a photo, including its photoname, caption, and filepath.
- **Highlights**:
  - Make sure that the user does not have any missing input

5. PhotoMetadataExtractor Class

- **What it does**: Extract photo gps latitude, longitude and date when the photo was taken.
- **Highlights**:
  - Create the constructor to extract the photo metadata.
  - Create exceptions to reject photos with missing metadata values, eg. missing gps data.

6. TravelDairy Class

- **What it does**: Runs the main software function.
- **Highlights**:
  - Collaborated with `Ojassurana` to create the main function of the code based on his previous IP.
  - Design the logic on how to track the fsm value.

7. Commands and commandFactory Classes

- **What it does**: Execute commands based on parser output and connects them to trip, tripmanager and photos.
- **Highlights**:
  - Collaborated with `Ojassurana` to create commands (addphoto, addtrip, closephoto, delete, list, menu)
  - Make sure the logic for the fsmvalue is correct in the commands

8. Tracker

- **What it does**: Track datetime of all photos inside a trip, track datetime difference between photos
- **Highlights**:
  - Created getPeriod to print out the minimum and maximum date of photos inside an album (originally it was implemented at tripmanager)

9. Storage

- **What it does**: Implements persistent data storage by saving and loading trip data from local files, ensuring that user data is retained across sessions.
- **Highlights**:
  - Hunt and debug the Storage class for any edge cases
  - Found a major bug where when a line of storage can not be processed, the rest of the text will not be transferred to trip manager
  - Added exception to gracefully reject trip with corrupted line
  - Added exception to gracefully reject photos and trips with duplicated name and filepath

---

💻 Code Contribution

- View my contributions

🚀 Project Management

- Managed GitHub releases: **v1.0**, **v2.0**, **v2.1**

- Integrated the trip, tripmanager, parser (which back then was combined with command), photo and album developed in **v1.0**
- Hunt bugs and solve them in **v2.1**, mainly on storage
- Coordinated team milestones and documentation updates
- Facilitated pull request reviews and issue triaging

Documentation

User Guide

- Added:
  - **Quick Start** section
  - Comprehensive **Command Summary**
  - Detailed usage documentation for the `help` and `close` commands

Developer Guide

- Documented:
  - **Main components (Parser, Commands, CommandFactory)**
  - **Interactions between components**
  - **Implementation details for Parser**

Community Involvement

- Reviewed PRs with constructive, non-trivial feedback: #22, #24, #34, #65
- Reported and suggested improvements for other teams: 29, 30, 136
