Lister App

R4 Link to Source Control

https://github.com/Yearnsmith/DerickYearnsmith_t1a3

R5 Software Development Plan

Purpose and Scope

Describe at a high level what app will do

Lister is a list manager. It allows users to create, edit, and save lists.

identify the problem it will solve and explain why you are developing it

1. Maintaining Lists

Makers and Creators use many kinds of lists: - Kanban Planning - Dependencies - issues - future projects - new features - lists of important lists!

Often ideas coem to us in the middle of an unrelated task. We have a choice to either let the idea go free or break our flow, load up a .txt, .doc, or pull out an old fashioned notepad and pen to write them down. For the creative type who uses a CLI, Lister enables them to quickly add to, remove from, or export a list from their CLI without changing directory, or opening a new terminal.

Lister also allows users to interact with lists in an interactive, distraction-free environment, with options to edit list items, and re-order them.

I am developing this app to help people like myself keep on task and not be distracted by new, exciting thoughts as they arise. It also will allow me to: 1. Work with file I/O 2. Feed inputs from the command line into the list 3. Perform interesting tasks with arrays, hashes, and other data-types 4. I can eventually port it to a Rails app, or other GUI interface for the Interactive Mode

identify the target audience

Lister's target audience is creators who work in a CLI, and don't enjoy being distracted or discarding cool ideas.

It's also aimed at people who like making lists.

explain how a member of the target audience will use it

There are two ways to use Lister: *Interactive mode* and *Express mode*. Lister automatically detects which mode is being used.

Interactive Mode

`lister [list-title] [list-item]`

Interactive mode clears the screen, and allows the user to edit a list by following prompts.

`lister list-title`
will load lister in Interactive mode, along with a file with a name that matches `list-title`.

If a matching file name doesn't exit, Lister will create the file, and interact with it.

`lister list-title list-item`
Will behave the same as above, and also append `list-item` to the list.

`lister`
Running `lister` without arguments will load lister into a cleared screen and the user will be prompted to create or load a new list

Stretch Goal *Use `xterm` to run `smcup` (or `\e[?1049h`) to act like less and open an altscreen.*

Operating within Interactive mode

Using Interactive Mode, Lister makes use of TTY-Prompt to give user an interactive list of options to choose between.

There is a new screen each time the user selects an option.

Express mode

`lister [option] [list-title] [list-item]`

`lister [ -a | -r list-title list-item ]`

`lister -c list-title [list-option]`

`lister -e list-title`

Express mode runs a specific feature of Lister, based on the options given.

Options

- -a, –**a**dd

  Append an item to a list. Requires user to give list-title and list-option. ¿If list-title doesn't exist, a new list will be created?

- -c, –**c**reate

  Create a new list. Requires user to give list-title. A first list item is optional. ¿combine with add?

- -e, –**e**cho

  print a list to command line. Requires user to give list-title

- -r, –**r**emove

  remove an item from a list. Requires user to give list-title and list-option.

- -x, –e**x**port

  Stretch Goal export a list to `Lister` folder in `My Documents`. Requires user to give list-title.

`list-title`

`list-title` may be entered two ways:

1. Single words for titles with 1 word *e.g.* `Ideas`
2. Strings surrounded by quotes for `list-title`s comprising multiple words
   e.g. `"Killer Novel Ideas"`

`list-item`

The item to be added or removed. It may also be a single word, or a string surrounded by quotes.

R6 Develop a list of features that will be included in the application.

Interactive Mode

Interactive Mode is just that: the terminal will be cleared, and users will be presented with a menu for interacting with Lister. This feature will make heavy use of the `TTY-Prompt` and `colorize` gems. It utilises a `case when` statement to map the users' selection to methods contained in `List` class.

The user will first be presented with a menu informing of the following options:

- Create List
- Edit List
- Exit

Create List

This is the main feature of Lister. The app will prompt the user for a title and first item. Lister will then construct a hash out of the answers, feeding them to a new `List` object. The hash will be in the following format to enable it to eventually be stored as a YAML file: `{ "list_title" => [list_items] }`

List objects have the following instance variables:

- `@list_title`: A string containing the title of the list

- `@list_contents`: An array with each list-item.
- `@list_hash`: A hash in the format -
  - key: `@list_title`
  - value: `@list_contents`

Once the list has been created, Lister will invoke the `list_edit_menu` `TTY::Prompt` object, prompting the user to continue modifying the list.

Edit List

Users of Lister are able to edit lists they have saved to file.

Users will select "Edit List" from the Interactive Mode menu and Lister invokes the `.edit_list` within Lister's `State` class.

`.edit_list` prompts the user to select from an array of list titles passed from `State` class' `@curent_state` hash. This is an array containing strings that map to every .yml file in Lister's `/lists` directory (`../src/lists/`)

Once a user selects a list to edit, the YAML file is passed as a hash into the `list` object, invoking the `list_edit_menu` `TTY::Prompt` object.

This object consists of the following options:

- Add item
- Remove item
- Change list title
- Save list
- Exit to main menu

Add Item

Lister users may add items to their list one at a time.

Lister will utilise the `.add_item` method in the `List` class. This behaves in the following way:

`.add_item` will prompt the user for the item they wish to add, store this local variable `item_to_add`, then append the item to the `@list_items` array. `item_to_add` will future-proof the app for a possible `.undo` method.

Remove Item

Lister users may remove items to their list one at a time.

Lister will utilise the `.remove_item` method in the `List` class. This behaves in the following way:

`.remove_item` prints the current iteration of the list for the user's convenience. It then prompts the user for the item they wish to add, stores this to local variable `item_to_remove`, matches

`item_to_remove` to a value in the `@list_items` array, and performs `.remove` on the matched item. `item_to_remove` will also future-proof the app for a possible `.undo` method.

Change List Title

It may occur that when a user is part-way through, or finished populating their list, they wish to chnage it's name. Upon selecting "Change List Title", Lister will utilise the `.update_title` method in the `List` class.

`.update_title` will prompt the user for a new title, and store it to local variable `new_list_title`. The current `List` object's `@list_title` will then be updated with `new_list_title`'s value. This local variable will also prepare for a possible `.undo` method.

Save List

Some lists are never completed — but users can't live their entire lives sitting in front of Lister! Not consecutively. `.save_list` allows users to save their lists to work on at a later time.

`.save_list` is a method within `List` class. `.save_list` converts the `@list_hash` to YAML formatting utilising *Psych's* `Psych.dump()`. Then writes the returned string to a YAML file using `IO.write` (From Ruby's `IO` class). This handy method opens, writes to, and closes a file, before returning the number of bytes written.

Once the file has been saved, Lister will ask the user if they wish to continue editing

Express Mode

`lister <option> <list-title> [list-item]`

In *Express mode*, the user will instruct Lister without an interface appearing.

One of the first things Lister does on opening is read the syntax of ARGV, and direct the program to the appropriate `control flow`. If the user has input any of the Express mode options (`-a`, `-r`, `-e`, `-x`) Lister will invoke a `.find_list` method in the `State` class. This matches `list-title` to a file with the same name and creates a new `List` object.

Lister will read the other items in ARGV, and feed these to the appropriate method.

Lister will automatically save the list, and provide confirmation that an item has been added or removed.

If Lister encounters any errors (missing arguments, non-existent list or items for -r), Lister will explain the error and close, to keep with expected behaviours of line-mode applications.

`lister -a <list-title> <list-item>`
Lister invokes `.add-item` method in the `List` class.

R7 Develop an outline of the user interaction and experience for the application.

Your outline must include:

- *how the user will find out how to interact with / use each feature*
- *how the user will interact with / use each feature*
- *how errors will be handled by the application and displayed to the user*

```
lister --add <list> <list-item>
```

R8 Develop a diagram which describes the control flow of your application.

- *show the workflow/logic and/or integration of the features in your application for each feature.*
- *utilise a recognised format or set of conventions for a control flow diagram, such as UML.*

R9 Develop an implementation plan

- *outlines how each feature will be implemented and a checklist of tasks for each feature*
- *prioritise the implementation of different features, or checklist items within a feature*
- *provide a deadline, duration or other time indicator for each feature or checklist/checklist-item*

Utilise a suitable project management platform to track this implementation plan

Trello Link

> *Your checklists for each feature should have at least 5 items.*

R10 Design help documentation which includes a set of instructions which accurately describe how to use and install the application.

You must include:

- *steps to install the application*
- *any dependencies required by the application to operate*
- *any system/hardware requirements*