tare-csv

Terminal Application Read / Edit Comma Separated Values

Evelyn Paplauskas

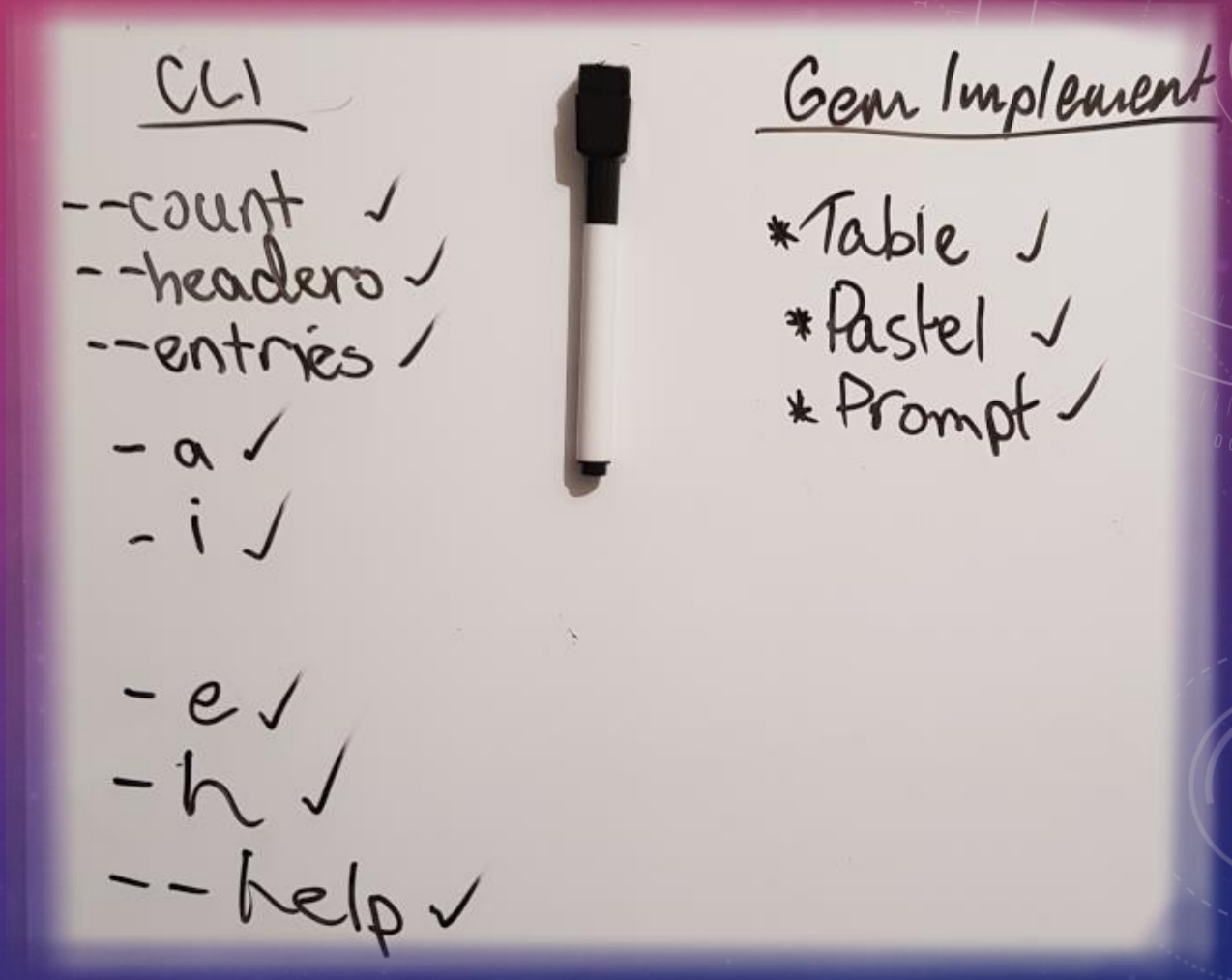
https://github.com/evey-pea/T1A2

BUILD WORKFLOW

DESIGN

As the project was a freeform design, workflow was kept simple by following these steps

- Establish the program idea in a reference document (README.md)
- Write out in detail each individual option the user had available in the document
- Write down each of the gems to be used
- Based on how each part of the implementation of the build performed, re-evaluate the program features, scope, functions and required components
- Upon each re-evaluation, update the documentation accordingly



BUILD WORKFLOW

IMPLEMENTATION

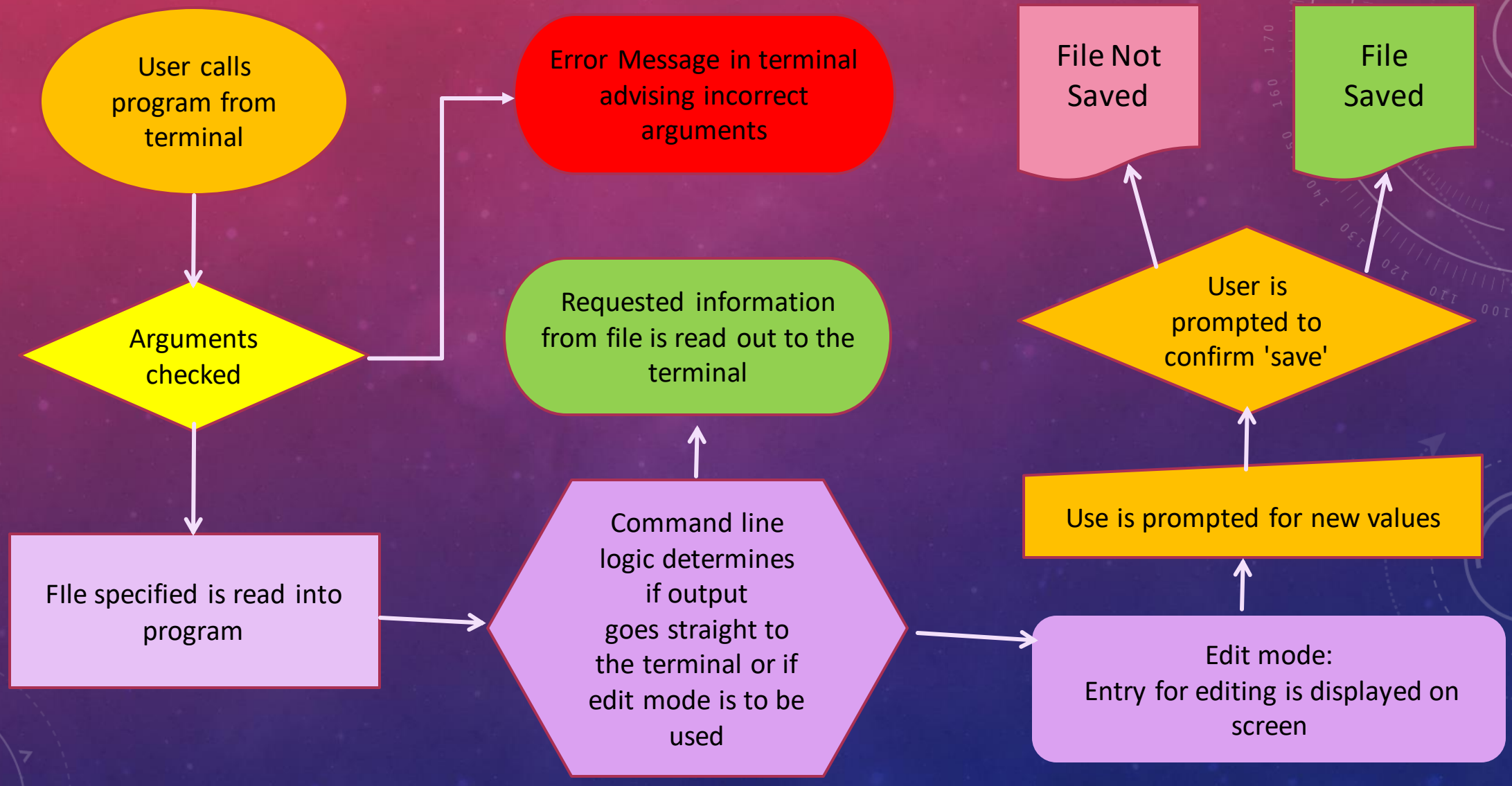
- The initial part of the build was the main program class and the command line logic
- The command line logic was then moved to its own module to reduce code in the main file for better readability.
- Each feature of the application was coded one after the other.
- For testing purposes, test data was used locally in each file for testing each method
- Once all the methods for each module was deemed working, it was linked in to the main program and tested once again

```ruby
require "pastel"
pastel = Pastel.new
$accepted = pastel.white.on_green.detach
$rejected = pastel.white.on_red.detach
$warning = pastel.black.on_yellow.detach

require "./modules/CSV_IO.rb"
# Counts number of items in first line of file
def file_count_headers
    return "File header count: #{@data[0].length}"
end

# Returns a count of entries in file
def file_count_entries
    return "File entry count: #{@data.length - 1}"
end

def file_count_headers_and_entries
    return [file_count_headers, file_count_entries]
end
# Output index values without headers as a single array
def index_output_read(index)
    output = []
    entry = @data[index]
    for field in entry do
        output.push(field)
    end
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1: ruby

```
Installing ri documentation for wisper-2.0.1
Parsing documentation for tty-screen-0.7.1
Installing ri documentation for tty-screen-0.7.1
Parsing documentation for tty-cursor-0.7.1
```

USER OPTIONS AND PROGRAM FLOW



PERSONAL BENEFITS FROM THE PROJECT

Learnings

- Reading and writing files in Ruby is sometimes better done in pure core Ruby standard constructs rather than relying gems
- The difficulty of implementing a gem is directly proportionate to amount of documentation available for the gem
- Modules are awesome for the DRY principle
- Checking what data (and its structure) has been passed to a method is the best thing to do first when debugging errors
- Sithu Aye has some awesome albums that are great to use as a mental focus tool

Favourite Milestones and Parts

- When code just worked without having to debug an error (it was a rare and welcome sight)
- Getting the ARGV parsing logic right for the command line (This made things so much easier to integrate later on)
- Working out how to transpose multidimensional arrays
- Working with the pastel gem –using aliases made putting together preset themed colours so much easier than doing it manually
- Sleeping after it was finished
- Waking up and realising I wrote approximately 320 lines of freeform refactored code in four days – a new P.B.