Terminal App Project Management

To Do:

- Create idea and get confirmation ✔️
- Setup project, install bundle, commit and push✔️
- Create a list of classes, subclasses and superclasses ✔️
- Design vehicle and options list with prices ✔️
- Design features ✔️
- Choose gems and install: Rubocop ✔️ TTY Progressbar ✔️ TTY Prompt ✔️ Colorize✔️
- README ✔️
- Create Instructions: ✔️
  - steps to install the application
  - any dependencies required by the application to operate
  - any system/hardware requirements
  - how to use any command line arguments made for the application
- Clean up code ✔️
- Add instructions ✔️
- PDF slide deck and project management
- Test

Other things to put into the app:

- Ascii art for main page title
- A main page with developer info, ascii art, and initialization from the user
- TTY Progressbar for submit ✔️

Fix:

- Metallic Navy - Miami Blue don't return full spec sentence ✔️

Feature Implementation

Feature 1: HIGH PRIORITY
Estimated delivery date Friday 15th April: Completed on time ✔️
Feature 1 is to build the basic usability of the app and is high priority
Using TTY Prompt a selection of vehicle options will be displayed. A basic option list will consist of a few items, where only one item can be chosen. A case statement will take the user input from each vehicle option item list and add it to a fully specified car object which will contain each option item and the price.
To do:
Populate each class of vehicle option with:

- class info ✔️
- TTY Prompt selection method for the index.rb to call ✔️
- case statement for accepting a selection ✔️ and passing the data to the option object (unsure)

Design welcome page with :

- Initial menu for about, base price, spec a new car and later for saved session ✔️
- Case statement for above menu ✔️
- Require relative links to each options methods on its on page ✔️

Test:

At this point the basic app is complete ✔️

Feature 2: HIGH PRIORITY
Estimated delivery date by Friday 22nd April: ✔️
Feature 2 is high priority as it is part of the main functionality of the app.
Feature 2 adds the ability to add a specification on top of another. In order to specify the color of your wheels, you first need to select wheels. If you select a color first, the configurator will offer you a wheel choice to begin with, then color. Also, this feature has the ability to ensure incompatible options are not chosen, in this case Carbon Fibre wheels are not compatible with yellow rim borders. If you select these options the program will tell you they are not compatible and lets you reselect another color.
To do:

- Create classes ✔️
- Link with inheritance - not finished
- Create conditional case statement to exclude yellow rims on carbon fibre wheels ✔️
- Enforce wheel type option before wheel color ✔️
- Utilise TTY Prompt and Colorize Gems ✔️

Feature 3: MEDIUM PRIORITY
Estimated start date: Thursday 21st April: not finished
The fully optioned car object will have each items cost added to create a total cost which will be added to the base price of the car. The full specification and total cost will be given to the user to save in a database, download for themselves and send a confirmation to the dealer to begin the build process of the vehicle.
To do:

- Create car options array/hash ✔️
- Get instance variable options value stored in $my_spec - doesn't work
- Iterate through my_spec and sum the values - unfinished
- Create "Spec Cost" menu item and save cost of options inside
- Add "Spec Cost" to "Base Price and print to user a total vehicle cost