Python CLI Application

Presented by Edward Doughery

→ Next

Overview

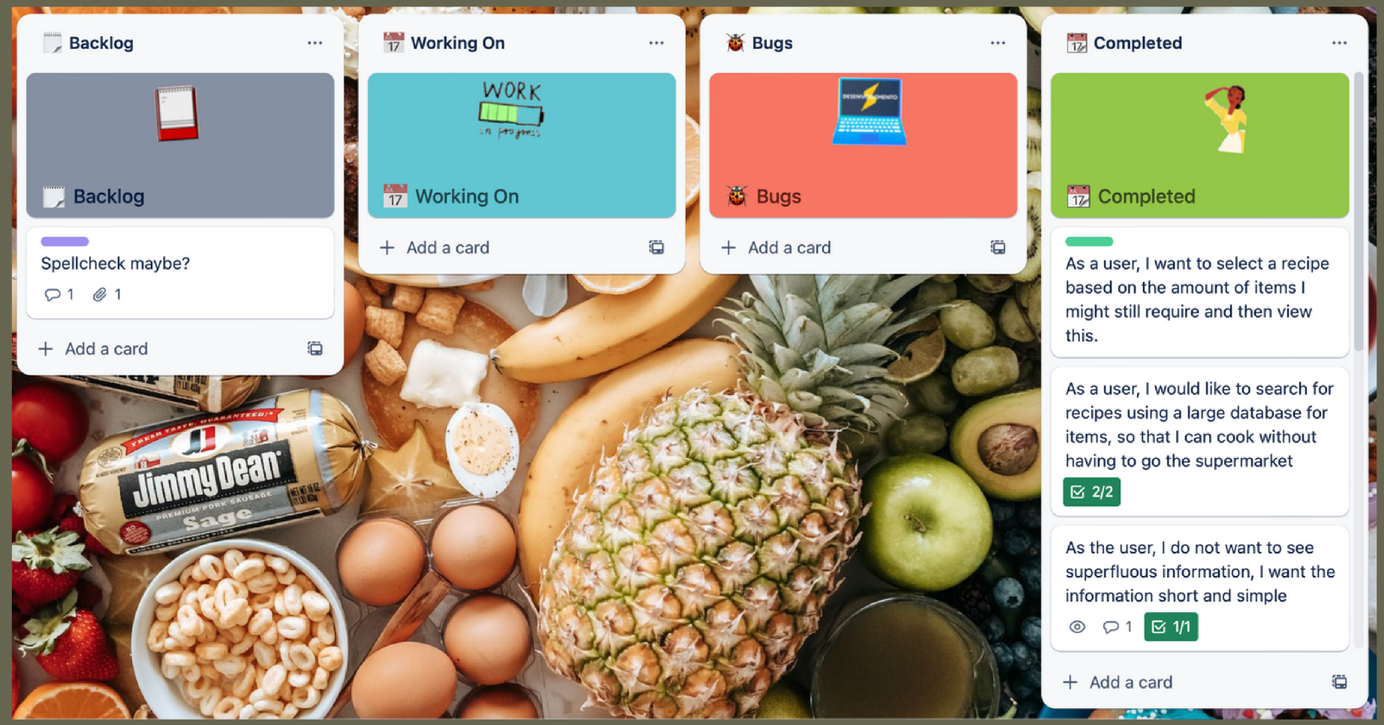
- Design process and decisions
- Meal Mate App and its features
- Review of build process.

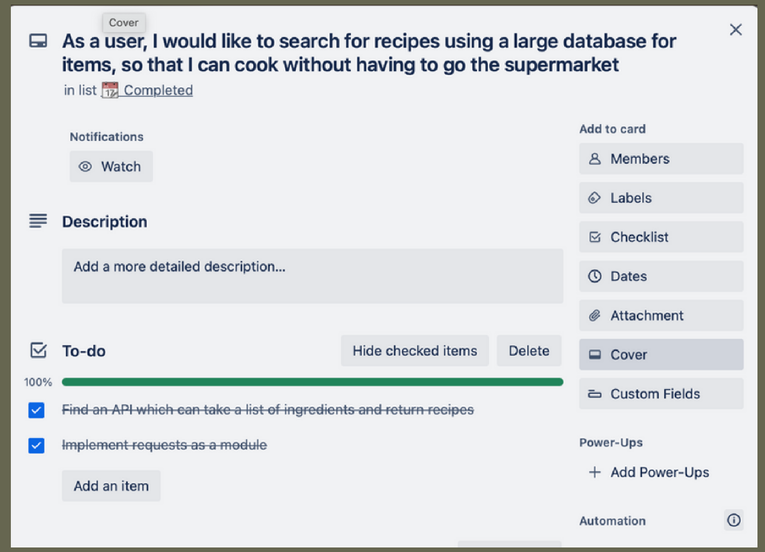
Meal Mate

Design process

Project Management

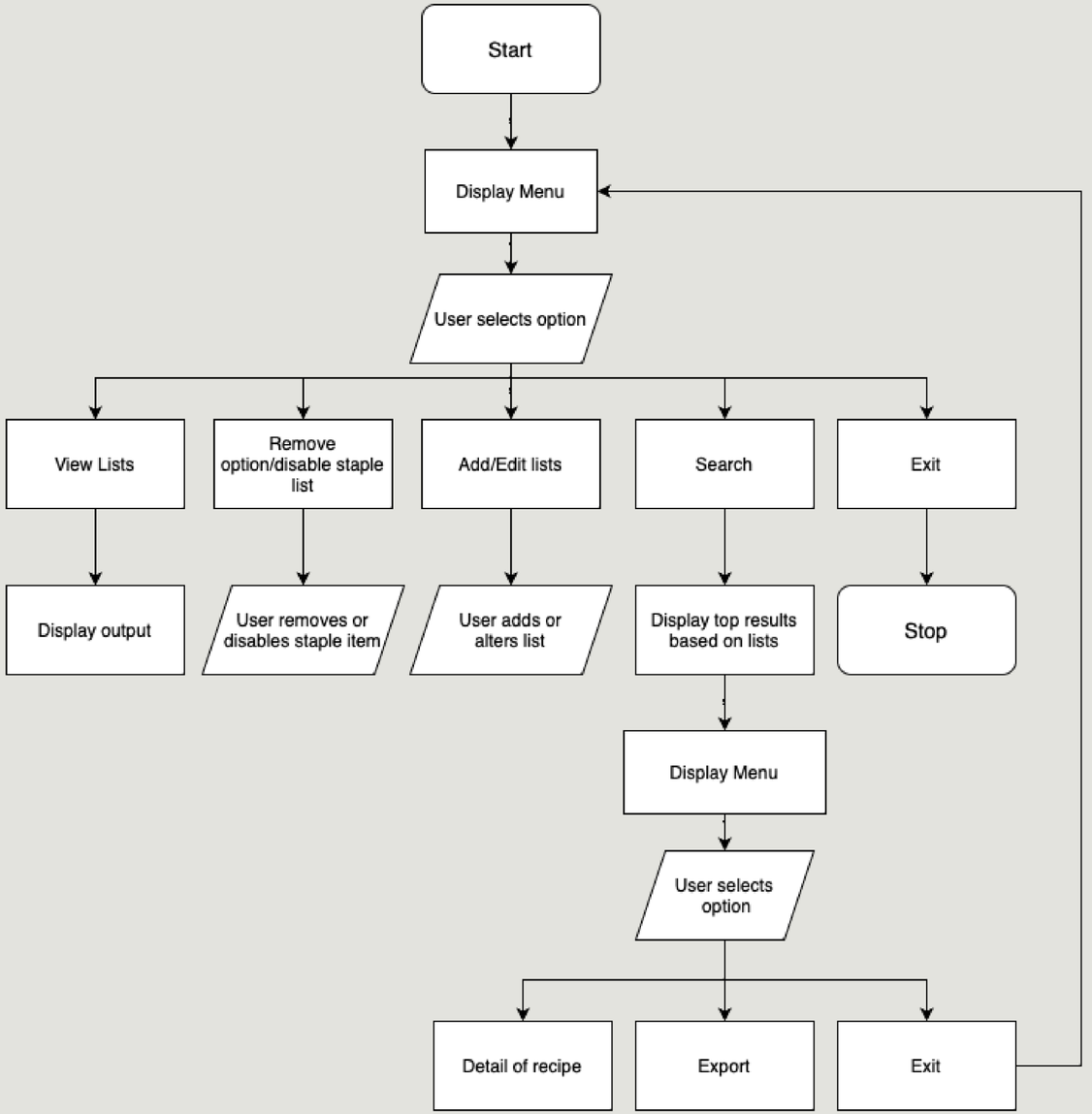
- Trello was selected as my management platform.
- I started with some user stories and added some checklists based around the functionality





Navigation

- Flowchart of the process that the user would use
- Text based, so colour and some formatting would be added afterwards



Meal Mate App and its features

Checklist of staple ingredients

- Staple ingredients are created based on a default list.
- This contains a True/False value on stock availability.

Editable list of pantry ingredients

- Ingredients are created as a CSV.
- This is able to be edited within or outside of the application.

Fetch a list of recipes via API

- Search for recipes from an external source (Spoonacular).
- User can select from based on name and number of ingredients.

Export recipe from CLI to PDF.

- Select the recipe to export.
- Spoonacular HTML webpage to convert to PDF and saves to the directory of the user.

Application Build

1. Project Management tool
    a. Trello
2. Application folder structure
    a. including github connection
3. Navigation decisions
    a. Menu and number selection
4. Finding an appropriate API source
    a. 1st API searched based on recipe title and not ingredients
    b. 2nd API (Spoonacular) was exactly what I wanted and had few limitations
5. Features added based on Test Driven Development
6. Automated tests using pytest were utilised for specific areas of the application
7. Bash Scripts created
8. User tests

Main menu

```
Welcome to Meal Mate
'The recipe finder for your pantry items you don't know what to with!'

Pantry items
1. to view you ingredient list
2. to add a new item to your list
3. to remove item from your list

Staple items ie. salt, pepper, oil, vinegar
4. to view you staple list
5. to change a staple item
6. to ignore staple items in search

Search
7. to search for recipes

Exit
8. to exit
Enter your selection:
```

- 'while' loop utilised as the menu input. With functions being called when the corresponding option is picked.
- 'Match-case' used which follows DRY principal, but reduces compatibility.
- CSV lists are used as the ingredient containers

```python
# user selection for main menu
while user_choice != "8":
    user_choice = create_menu()

    match user_choice:
        case "1":
            view_ingr(ingr_file_name)
        case "2":
            add_ingr(ingr_file_name)
        case "3":
            remove_ingr(ingr_file_name)
        case "4":
            staple_view_ingr(staple_file_name)
        case "5":
            staple_edit_ingr(staple_file_name)
        case "6":
            # ignore setting
            staple_ignore_response = input(
                f"Use the staple items list in your search? (y/n): "
            )
            staple_setting = staple_ignore(staple_ignore_response)
            print(
                f"\nThe search function is set to {fg(5)}{'use' if staple_setting else 'ignore'}{attr(0)} the staple list"
            )
        case "7":
            try:
                get_recipes(ingr_file_name, staple_file_name, staple_setting)
            except NameError:
                get_recipes(ingr_file_name, staple_file_name, True)
        case "8":
            continue
        case _:
            print(f"{bg(1)}Invalid input{attr(0)}\n")
            time.sleep(1)
            continue

    input(f"\n{bg(177)}Press Enter to continue...{attr(0)}\n")

print(f"{bg(2)}Thank you for using Meal Mate{attr(0)}\n")
```

Pantry and Staple Items view

```
View staple items
Olive oil (in stock)
Vegetable oil (in stock)
Vinegar (in stock)
Salt (in stock)
Pepper (in stock)
Oregano (in stock)
Parsley (in stock)
Chilli Powder (out of stock)
Ground Cumin (in stock)
Paprika (in stock)
Cinnamon (in stock)
Curry Powder (out of stock)
Garlic (in stock)
Onion (in stock)
Ginger (in stock)
```

```python
def staple_view_ingr(staple_file_name):
    print(f"\n{bg(90)}View staple items{attr(0)}")
    with open(staple_file_name, "r") as f:
        reader = csv.reader(f)
        reader.__next__()
        for row in reader:
            if row[1] == "True":
                print(f"{fg(2)}{row[0]} (in stock){attr(0)}")
            else:
                print(f"{fg(1)}{row[0]} (out of stock){attr(0)}")
```

- Simple functions to display CSV files

Pantry and Staple Items Add/Edit

```
Add pantry item
Enter your ingredient: Chicken

Press Enter to continue...
```

```
View pantry items
Chicken

Press Enter to continue...
```

```python
def add_ingr(file_name):
    print(f"\n{bg(2)}Add pantry item{attr(0)}")
    ingr_title = input("Enter your ingredient: ")
    with open(file_name, "a") as ingr_file:
        writer = csv.writer(ingr_file)
        writer.writerow([ingr_title])

```

```python
def staple_edit_ingr(staple_file_name):
    print(f"\n{bg(90)}Modify staple item{attr(0)}")
    staple_view_ingr(staple_file_name)
    ingr_title = input("Enter the item that you want to check/uncheck: ")
    staple_lists = []
    with open(staple_file_name, "r") as f:
        reader = csv.reader(f)
        for row in reader:
            if (ingr_title == row[0]):
                if row[1] == "False":
                    staple_lists.append([row[0], "True"])
                else:
                    staple_lists.append([row[0], "False"])
            else:
                staple_lists.append(row)
    with open(staple_file_name, "w") as f:
        writer = csv.writer(f)
        writer.writerows(staple_lists)
    staple_view_ingr(staple_file_name)

```

- Similar functionality between add and editing the lists
- User enters a name to corresponding value

Staple Items Ignore

```
Use the staple items list in your search? (y/n): n

The search function is set to ignore the staple list

Press Enter to continue...
```

- Ignore function returns False
- This has to effects:
  - The Staple Items get ignored in the search parameter
  - Adds a parameter in with API as well which ignores staple items.
  - This makes the results a bit less specific, so the results will be a bit broader when not using the staple items.

```python
        case "6":
            # ignore setting
            staple_ignore_response = input(
                f"Use the staple items list in your search? (y/n): "
            )
            staple_setting = staple_ignore(staple_ignore_response)
            print(
                f"\nThe search function is set to {fg(5)}{'use' if staple_setting else 'ignore'}{attr(0)} the staple list"
            )
```

```python
def staple_ignore(staple_ignore_response):
    match staple_ignore_response:
        case 'y':
            return True
        case 'n':
            return False
        case _:
            print(f"\n{bg(1)}Please enter y (for yes) or n (for no){attr(0)}")
            time.sleep(1)

```

Initial Search

```
case "7":
    try:
        get_recipes(ingr_file_name, staple_file_name, staple_setting)
    except NameError:
        get_recipes(ingr_file_name, staple_file_name, True)
```

- Utilises the lists and places them in a comma seperated string
- Depending on the 'staple_setting' the staple list will be added to the list or not.
- A parameter on the API endpoint returns 5 recipes, this could be increased, but I didn't want to overwhelm the user as they are likely to only look at the top result as this utilises the most ingredients

```
def get_recipes(ingr_file_name, staple_file_name, staple_setti
    p = ""  # pantry + staples
    with open(ingr_file_name, "r") as f:
        reader = csv.reader(f)
        reader.__next__()
        for row in reader:
            p = p + str(row[0]) + ","
    if staple_setting:
        with open(staple_file_name, "r") as f:
            reader = csv.reader(f)
            reader.__next__()
            for row in reader:
                if row[1] == "True":
                    p = p + row[0] + ","
```

```
r = requests.get(
    "https://api.spoonacular.com/recipes/findByIngredients?apiKey=3e06d892f3044bab8b766176ccd0e18c&ingredients="
    + p
    + "&ranking=2&number=5"
    + ("&ignorePantry=true" if not staple_setting else "")
)
json = r.json()
```

Search menu

```
Searching for recipes with 3 items and ignoring staple items

The following recipes utilise your existing ingredients!
1. Zucchini Chicken Omelette
Utilises: 3 items, Requires: 0 items

The following recipes may require a trip to the shops
2. Square Deviled Eggs
Utilises: 1 item, Requires: 2 items
3. Nutella Buttercream Cupcakes with Hidden Cadbury Egg
Utilises: 1 item, Requires: 2 items
4. Toasted" Agnolotti (or Ravioli)
Utilises: 1 item, Requires: 2 items
5. Scotch Eggs
Utilises: 1 item, Requires: 2 items

Recipe options
1. View more details about a recipe
2. Export a recipe
Exit
3. to exit
Enter your selection:
```

- Results listed as a numbered list with amount of items used and required
- 5 recipes returned they can choose from.
- The View function uses the initial search results to show more details
    - limits the amount of API results.
- The Export function uses the recipe specific API to return the URL of the recipe and in turn convert that to a PDF

init_run.sh

```
#!/bin/bash

printf "Installing required packages"
sleep 1

# Install required packages
python3 -m pip install -r requirements.txt
# MacOS homebrew install for wkhtmltopdf
brew install homebrew/cask/wkhtmltopdf

printf "Install complete"

python3 main.py
```

uninstall.sh

```
#!/bin/bash

printf "Thanks for using Meal Mate, the required packages are now uninstalling"
sleep 1

python3 -m pip uninstall -r requirements.txt
# MacOS homebrew install for wkhtmltopdf
brew uninstall homebrew/cask/wkhtmltopdf

printf "Uninstall complete. Have a nice day!"
```

- Two Scripts, which I find most useful
  - install and run application
  - uninstall components of application
- Requirements.txt document is used
- Homebrew is used to install 'wkhtmltopdf' which is the module used to have the export function.
  - this limits the use case to systems which can use the homebrew installer

Bash Scripts

Review

- Ethical issues
  - Gaining inspiration
  - Relying on the 3rd party API for the main use case of the application
- Favourite parts
  - Designing the lists and finding modules
  - Parsing the API
  - variable scope was a learning factor
  - when to actually use a function and when to just use lines of code
- What I would do differently?
  - Look at other options for user input
  - Think about another way of storing lists and how to make this user friendly
  - Use classes for some things
  - Add the option for the user to use their own API keys instead of leaving mine in the application

THANKYOU