STEFFS PHARMACY

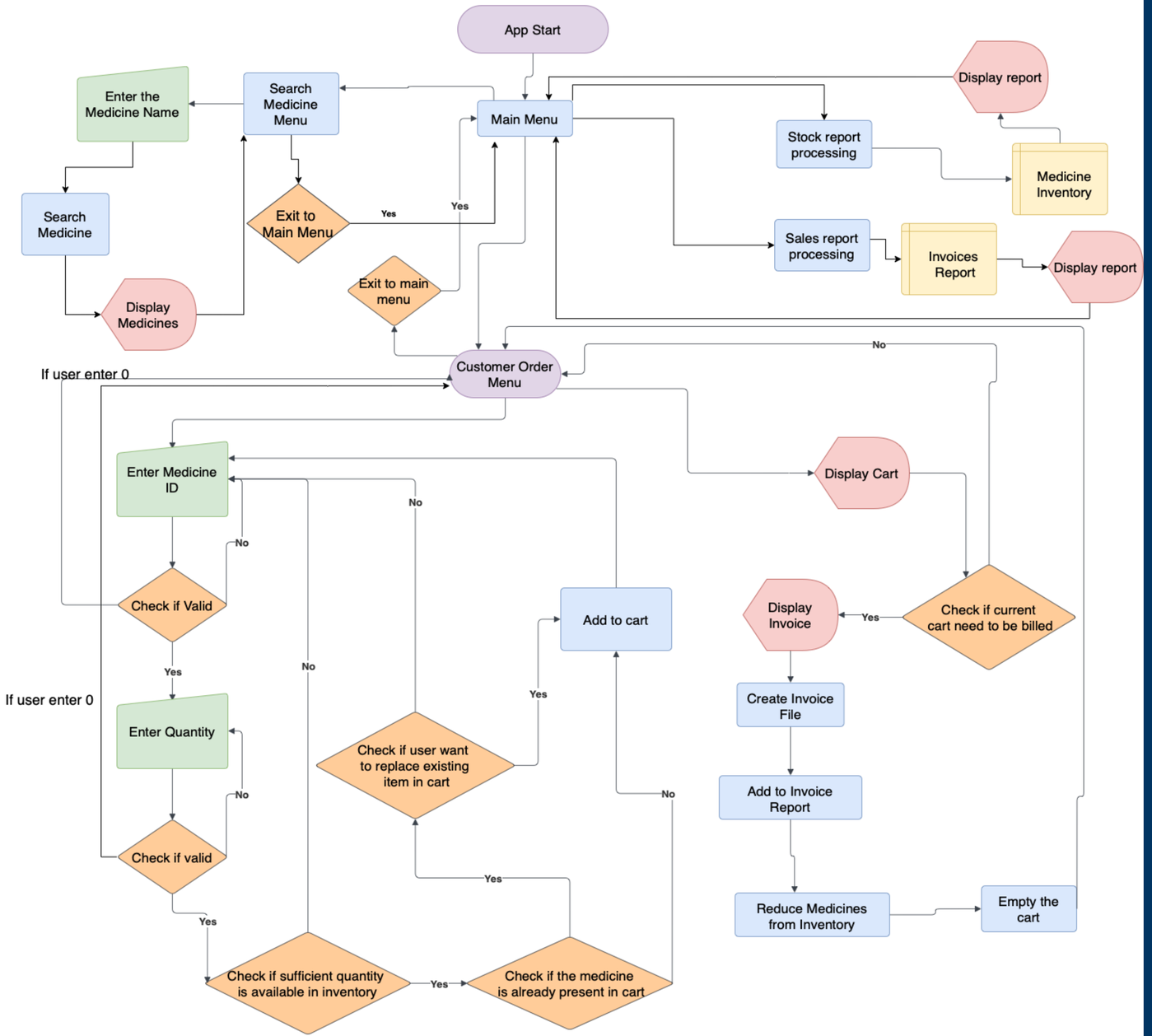
T1A3 - PythonTerminal App

By - Steffy Johnson

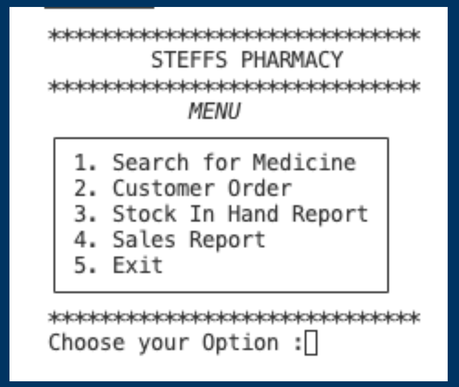
Introduction

STEFFS PHARMACY app, created using python is intended for the management of the daily activities of a pharmacy. The application facilitates medicine inventory management and billing of customer orders. It helps the pharmacist to keep track of the medicines that are currently available. The precise stock quantity report that is available in this app helps to forecast and place new stock replenishment orders on timely manner. The customer's enquiries for availability of any medicine can be easily found by the inventory search feature. Customer orders can be placed with ease along with the generation of invoice with a unique invoice number attached to each customer order. The sales report feature gives the list of sales made along with the invoice number and medicines that were purchased. This helps in easier tracking of the sales made at any point in time.

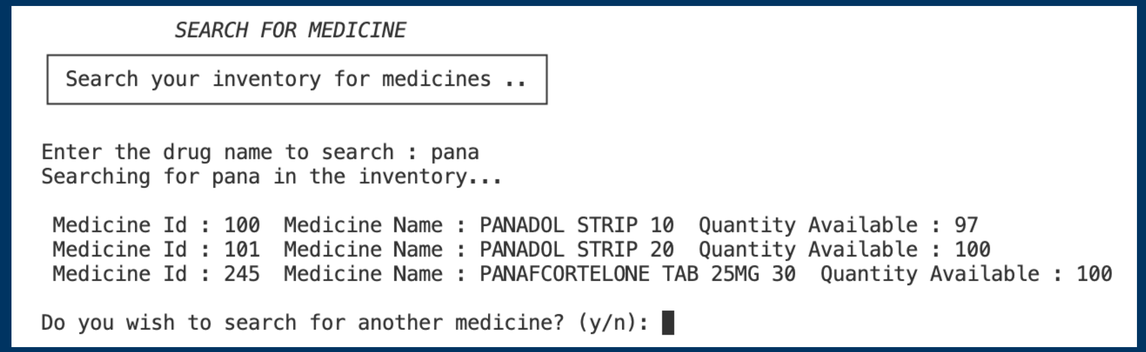
STEFF'S PHARMACY FLOWCHART



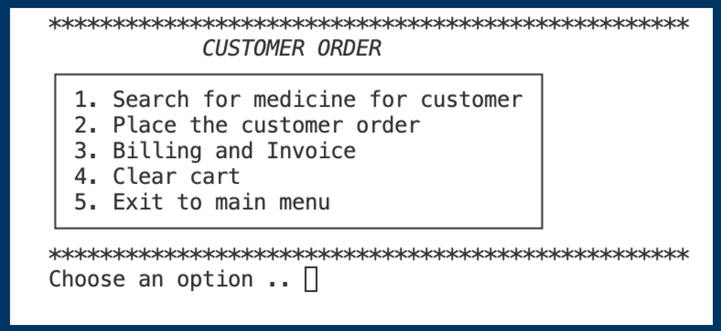
Terminal Walkthrough



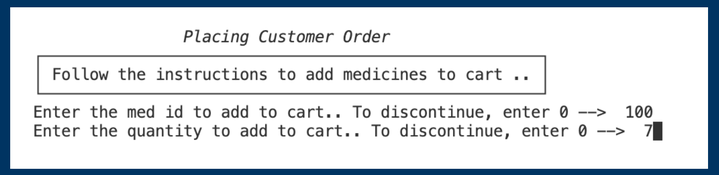
Main Menu:
The application main menu is displayed here. There are options to perform search . Place customer order. There are couple of reports that can be generated which are the stock and sales reports.



Search Medicines :
Regular expressions are used to match the user input against the medicine inventory. All matching medicines are displayed. User can perform multiple searches. The user need to take note of the Medicine id of the medicines that customers need , this medicine id is used when placing the customer order . The display also includes the medicine name, quantity available and the price details.

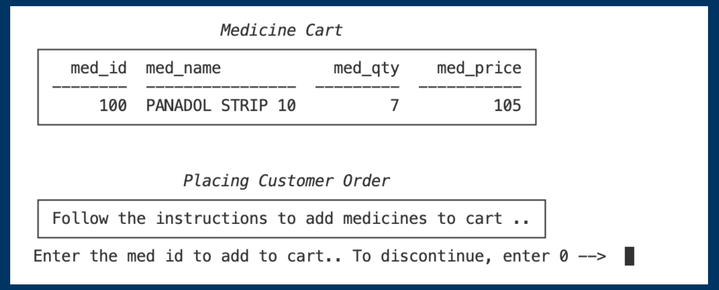


Customer Order:
The customer order option (2) from main menu points to the customer order menu . This is where the user can place the order for the customer. Medicines can be added to cart , the items in the cart can be billed and the invoice is generated . An invoice file is also created that is saved into the local disk. There is also option to clear the cart in case the user wants to redo their cart additions.



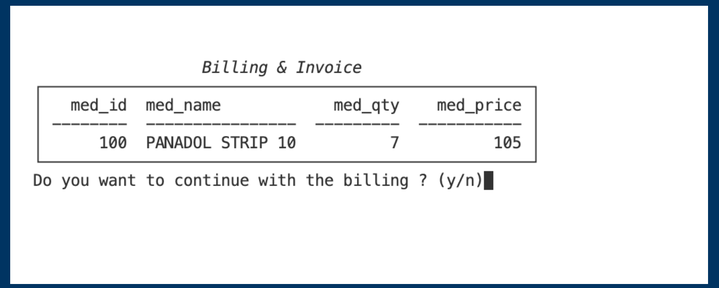
Placing customer Order:

This screen helps the user in adding the medicines to the cart. It is essential that the user is aware of the MED-ID beforehand.



Placing customer order —> Medicine Cart :

This screen displays the cart while placing each medicines to cart. In order to exit the customer order menu , the user need to enter ‘0’.



Billing & invoice:

This screen is to do the billing for the customer order . The items present in the cart is displayed . If the user is satisfied, then user can continue with the invoice generation.

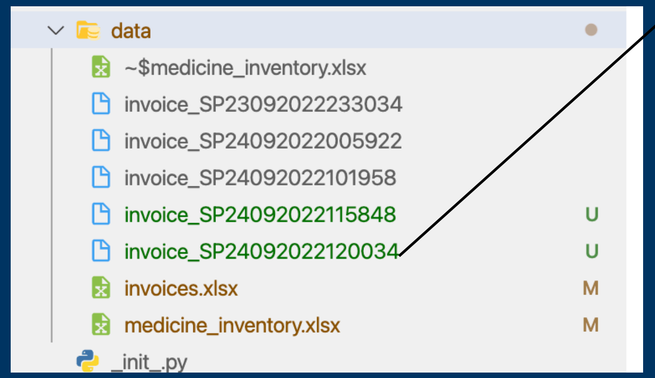
Billing & Invoice

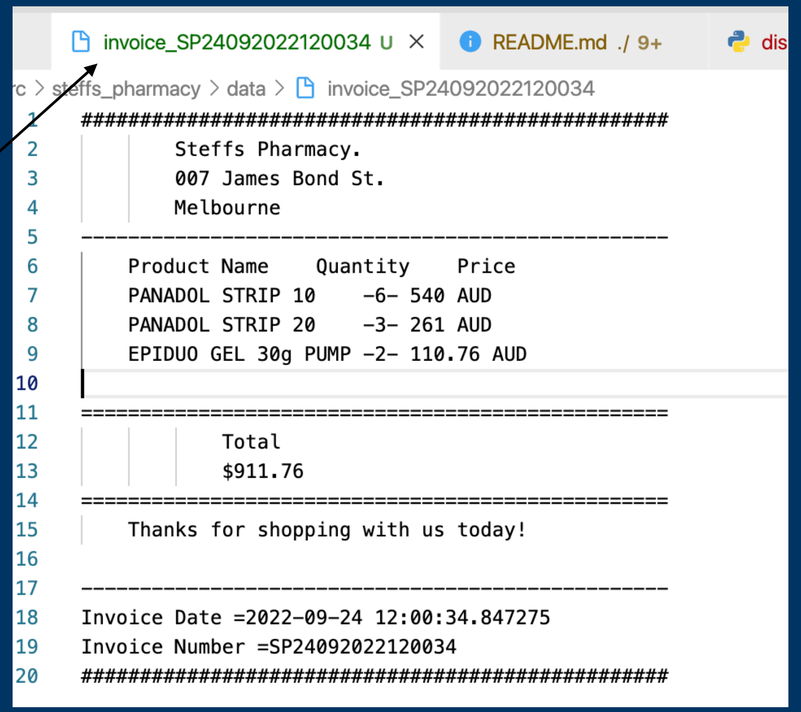
med_id	med_name	med_qty	med_price
100	PANADOL STRIP 10	6	90
101	PANADOL STRIP 20	3	87
180	EPIDUO GEL 30g PUMP	2	55.38

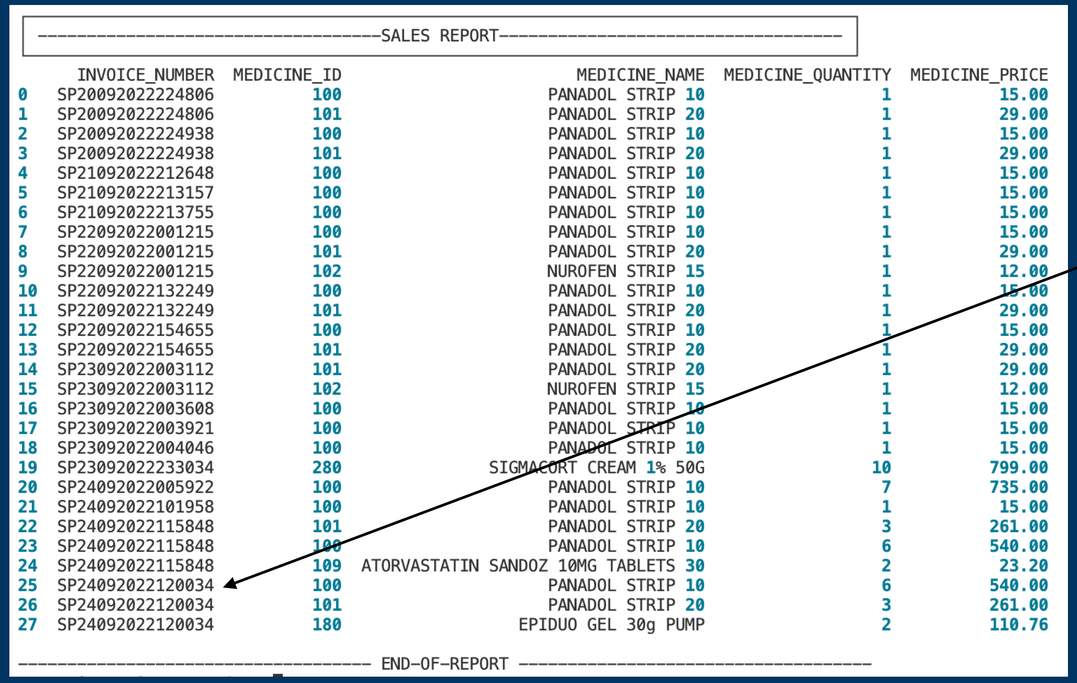
```
Do you want to continue with the billing ? (y/n)y
################################################
                    INVOICE

                Steffs Pharmacy.
                007 James Bond St.
                Melbourne
------------------------------------------------
        Product Name       Quantity     Price
        PANADOL STRIP 10        -6-     540 AUD
        PANADOL STRIP 20        -3-     261 AUD
        EPIDUO GEL 30g PUMP     -2-     110.76 AUD
================================================
                        Total
                        $911.76
================================================

        Thanks for shopping with us today!


------------------------------------------------
Invoice Date =2022-09-24 12:00:34.847275

Invoice Number =SP24092022120034

################################################
Press 'Enter' to continue..
```

Billing & Invoice:
This screen shows the invoice that is generated. It contains the list of medicines, the quantity and the individual price. There is also total price displayed. The invoice date provides current date . There is a unique invoice number generated for each of the purchases of the format SP<ddmmyyyyHHMMSS>

After you press the enter key , the invoice file is created in the / data folder with the name as 'invoice_<invoicennumber>' . The sample is displayed below.







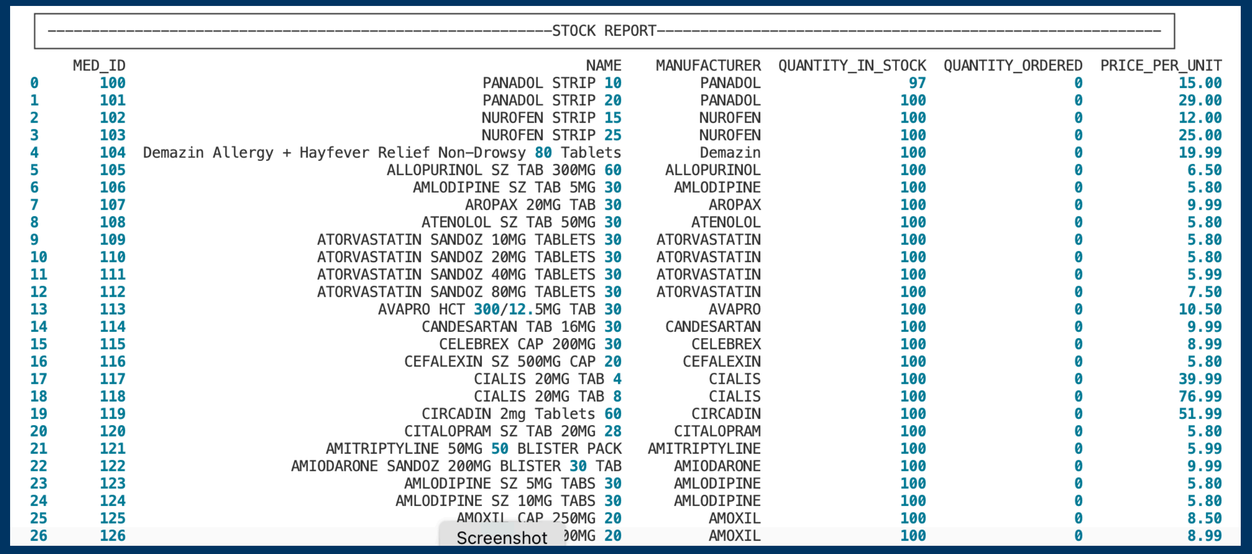
----------SALES REPORT----------

	INVOICE_NUMBER	MEDICINE_ID	MEDICINE_NAME	MEDICINE_QUANTITY	MEDICINE_PRICE
0	SP20092022224806	100	PANADOL STRIP 10	1	15.00
1	SP20092022224806	101	PANADOL STRIP 20	1	29.00
2	SP20092022224938	100	PANADOL STRIP 10	1	15.00
3	SP20092022224938	101	PANADOL STRIP 20	1	29.00
4	SP21092022212648	100	PANADOL STRIP 10	1	15.00
5	SP21092022213157	100	PANADOL STRIP 10	1	15.00
6	SP21092022213755	100	PANADOL STRIP 10	1	15.00
7	SP22092022001215	100	PANADOL STRIP 10	1	15.00
8	SP22092022001215	101	PANADOL STRIP 20	1	29.00
9	SP22092022001215	102	NUROFEN STRIP 15	1	12.00
10	SP22092022132249	100	PANADOL STRIP 10	1	15.00
11	SP22092022132249	101	PANADOL STRIP 20	1	29.00
12	SP22092022154655	100	PANADOL STRIP 10	1	15.00
13	SP22092022154655	101	PANADOL STRIP 20	1	29.00
14	SP23092022003112	101	PANADOL STRIP 20	1	29.00
15	SP23092022003112	102	NUROFEN STRIP 15	1	12.00
16	SP23092022003608	100	PANADOL STRIP 10	1	15.00
17	SP23092022003921	100	PANADOL STRIP 10	1	15.00
18	SP23092022004046	100	PANADOL STRIP 10	1	15.00
19	SP23092022233034	280	SIGMACORT CREAM 1% 50G	10	799.00
20	SP24092022005922	100	PANADOL STRIP 10	7	735.00
21	SP24092022101958	100	PANADOL STRIP 10	1	15.00
22	SP24092022115848	101	PANADOL STRIP 20	3	261.00
23	SP24092022115848	100	PANADOL STRIP 10	6	540.00
24	SP24092022115848	109	ATORVASTATIN SANDOZ 10MG TABLETS 30	2	23.20
25	SP24092022120034	100	PANADOL STRIP 10	6	540.00
26	SP24092022120034	101	PANADOL STRIP 20	3	261.00
27	SP24092022120034	180	EPIDUO GEL 30g PUMP	2	110.76

---------- END-OF-REPORT ----------

Sales Report:

This screen shows the sales report. All purchases made are displayed here. The invoice number denotes each purchases made.



----------STOCK REPORT----------

	MED_ID	NAME	MANUFACTURER	QUANTITY_IN_STOCK	QUANTITY_ORDERED	PRICE_PER_UNIT
0	100	PANADOL STRIP 10	PANADOL	97	0	15.00
1	101	PANADOL STRIP 20	PANADOL	100	0	29.00
2	102	NUROFEN STRIP 15	NUROFEN	100	0	12.00
3	103	NUROFEN STRIP 25	NUROFEN	100	0	25.00
4	104	Demazin Allergy + Hayfever Relief Non-Drowsy 80 Tablets	Demazin	100	0	19.99
5	105	ALLOPURINOL SZ TAB 300MG 60	ALLOPURINOL	100	0	6.50
6	106	AMLODIPINE SZ TAB 5MG 30	AMLODIPINE	100	0	5.80
7	107	AROPAX 20MG TAB 30	AROPAX	100	0	9.99
8	108	ATENOLOL SZ TAB 50MG 30	ATENOLOL	100	0	5.80
9	109	ATORVASTATIN SANDOZ 10MG TABLETS 30	ATORVASTATIN	100	0	5.80
10	110	ATORVASTATIN SANDOZ 20MG TABLETS 30	ATORVASTATIN	100	0	5.80
11	111	ATORVASTATIN SANDOZ 40MG TABLETS 30	ATORVASTATIN	100	0	5.99
12	112	ATORVASTATIN SANDOZ 80MG TABLETS 30	ATORVASTATIN	100	0	7.50
13	113	AVAPRO HCT 300/12.5MG TAB 30	AVAPRO	100	0	10.50
14	114	CANDESARTAN TAB 16MG 30	CANDESARTAN	100	0	9.99
15	115	CELEBREX CAP 200MG 30	CELEBREX	100	0	8.99
16	116	CEFALEXIN SZ 500MG CAP 20	CEFALEXIN	100	0	5.80
17	117	CIALIS 20MG TAB 4	CIALIS	100	0	39.99
18	118	CIALIS 20MG TAB 8	CIALIS	100	0	76.99
19	119	CIRCADIN 2mg Tablets 60	CIRCADIN	100	0	51.99
20	120	CITALOPRAM SZ TAB 20MG 28	CITALOPRAM	100	0	5.80
21	121	AMITRIPTYLINE 50MG 50 BLISTER PACK	AMITRIPTYLINE	100	0	5.99
22	122	AMIODARONE SANDOZ 200MG BLISTER 30 TAB	AMIODARONE	100	0	9.99
23	123	AMLODIPINE SZ 5MG TABS 30	AMLODIPINE	100	0	5.80
24	124	AMLODIPINE SZ 10MG TABS 30	AMLODIPINE	100	0	5.80
25	125	AMOXIL CAP 250MG 20	AMOXIL	100	0	8.50
26	126	00MG 20	AMOXIL	100	0	8.99

Screenshot

Stock Report:

This screen shows the stock report. It shows all the medicines that are available in stock. After each purchase , corresponding quantity of medicines are reduced from the inventory.

Coding Walkthrough

```python
def main():
    console = Console()
    cart = []
    while True:
        clearing.clear()
        print("*****************************")
        table = Table(show_header=False, header_style="bold blue",
                      title="MENU", title_justify="center")
        table.add_row("1. Search for Medicine")
        table.add_row("2. Customer Order")
        table.add_row("3. Stock In Hand Report")
        table.add_row("4. Sales Report")
        table.add_row("5. Exit")
        console.print(table)
        print("*****************************")
        # Try block for user input
        try:
            choice = int(input("Choose your Option :"))
        except ValueError:
            print("Invalid Option !")
            time.sleep(2)
            continue
        if choice == 1:
            # Call funtion to seach medicine
            search_drugs()
        elif choice == 2:
            # call function to place customer order
            cart = customer_order(cart)
        elif choice == 3:
            # Call function to display stock report
            display_stock_report()
        elif choice == 4:
            # Call function to display invoice report
            display_invoice_report()
        elif choice == 5:
            # exit
```

Main Menu

- Uses rich library to display menus in a tabular form
- The menu code is placed in a while loop.
- The loop is exited only with a choice ‘5’
- The user input errors are caught by placing this in a try-except block.
- Each function corresponding to the user choice is called by using a If-elif-else control statements.

```python
def customer_order(cart):
    console = Console()
    while True:
        clearing.clear()
        # Menu for getting customer order
        print("*"*50)
        table = Table(show_header=False, header_style="bold blue",
                      title="CUSTOMER ORDER", title_justify="center")
        table.add_row("1. Search for medicine for customer")
        table.add_row("2. Place the customer order")
        table.add_row("3. Verify medicines in cart and confirm order")
        table.add_row("4. Clear cart")
        table.add_row("5. Exit to main menu")
        console.print(table)
        print("*"*50)
        # capture user input
        try:
            user_option = int(input("Choose an option .. "))
        # Catch  any invalid input
        except ValueError:
            print("Invalid Option !")
            time.sleep(2)
            continue
        # user options
        if user_option == 1:
            # Call function to  search for medicine
            search_drugs()
            time.sleep(1)
        elif user_option == 2:
            # Call function for placing customer order
            cart = placing_customer_order(cart)
            time.sleep(1)
        elif user_option == 3:
            if cart:
                # Invoice generation
                invoice_number = billing_invoice_generation(cart)
                if invoice_number is not None:
                    # call function to update inventory
                    inventory_update(cart)
                    # call function to update invoice
                    invoice_update(cart, invoice_number)
                    # clear the cart after the order is complete
                    cart.clear()
            else:
                # ALert that the cart is empty
                print("Cart is empty !")
            time.sleep(1)
```

Customer Order

- Uses rich library to display customer order menu in a tabular form
- The menu code is placed in a while loop.
- The loop is exited only with a choice ‘5’
- The user input errors are caught by placing this in a try-except block.
- Each function corresponding to the user choice is called by using a If-elif-else control statements.

```
# User input of the med id
while True:
    try:
        med_id_user_input = int(
            input("Enter the med id to add to cart.. To discontinue, enter 0 -->  ").strip())
        break
    except ValueError:
        # When user input is invalid display alert
        print("Invalid med id !")
        time.sleep(2)
        continue
# Return to Customer Order Menu
if med_id_user_input == 0:
    return medicine_cart
# User input of med quantity
while True:
    try:
        med_qty_user_input = int(input(
            "Enter the quantity to add to cart.. To discontinue, enter 0 -->  ").strip())
        break
    except ValueError:
        print("Invalid Quantity !")
        time.sleep(2)
        continue
```

Placing Customer Order

- The user input is received within a while loop. Exited when user enters ‘0’
- Invalid user input is handled by placing it in try-except block.
- Here Med-id and Med quantity are collected as inputs.

Contd to next slide ..

```
medicine_inventory = openpyxl.load_workbook(
    "src/steffs_pharmacy/data/medicine_inventory.xlsx")
mi = medicine_inventory.active
# Search Inventory for the Medicine and quantity
for i in range(2, mi.max_row+1):
    # Get the Med id for the row
    med_id = mi.cell(row=i, column=1)
    # Get the Med name for the row
    med_name = mi.cell(row=i, column=2)
    # Get the Med quantity for the row
    med_qty = mi.cell(row=i, column=4)
    # Get the Med price per unit for the row
    med_price = mi.cell(row=i, column=6)
    # Checking if the medicine is available
    if (med_id_user_input == int(med_id.value)):
        # Set the med availability flag to true
        found_med_flag = True
        available_med_qty = med_qty.value
        # Checking if quantity is less than the available quantity
        if (med_qty_user_input <= int(med_qty.value)):
            # Set quantity availability flag to true
            qty_available_flag = True
            # Checking if the medicine is already available in cart.
            if (next((i for i, x in enumerate(medicine_cart) if x["med_id"] == med_id.value), None) is not None):
                while True:
                    try:
                        # If the Med is already available in cart, check if it need to be replaced with new entry
                        med_duplicate_user_input = input(
                            "Med already present in cart. Do you want to replace the existing item in the cart? (y/n)")
                        # If yes then replace the existing item in cart with the new item
                        if (med_duplicate_user_input.lower() == 'y'):
                            medicine_cart = list(
                                filter(lambda i: i['med_id'] != med_id.value, medicine_cart))
                            medicine_cart.append({'med_id': med_id.value, 'med_name': med_name.value,
                                                  'med_qty': med_qty_user_input, 'med_price': med_price.value * med_qty_user_input})
                            print("Medicine replaced !")
                            break
                        # if no then skip the step to add to  cart
                        elif (med_duplicate_user_input.lower() == 'n'):
                            print(
                                "Operation cancelled . Item not added to cart...")
                            break
                        else:
                            # Raise a value error for any invalid inputs.
                            raise ValueError
                    # Catch any value  error for user inputs
```

Placing customer order

- The medical inventory is read using openpyxl library.
- The rows of inventory is traversed within a for loop.
- If-else loop checks the conditions whether the med-id and quantity is valid.
- There is an additional check to prevent duplicate medicines being added to the same cart. In such a case, the user is prompted to confirm whether to replace the item in cart or to cancel the operation.
- Item is added to cart only after these if-elif conditions are satisfied.

```
        # Creating the invoice
        # create a company name and information
        company_name = 'Steff''s Pharmacy.'
        company_address = '007 James Bond St.'
        company_city = 'Melbourne'
        # declare ending message
        message = 'Thanks for shopping with us today!'

        # create a top border
        print('#' * 50)
        # print company information first using format
        table = Table(show_header=False, header_style="bold blue",
                      title="INVOICE", title_justify="center")
        table.add_row('\t\t{}'.format(company_name.title()))
        table.add_row('\t\t{}'.format(company_address.title()))
        table.add_row('\t\t{}'.format(company_city.title()))
        # create a top border
        table.add_row('-' * 50)
        # print out header for section of items
        table.add_row('\tProduct Name\t   Quantity\tPrice')
        # create a print statement for each item
        for i in cart:
            table.add_row(
                '\t{}\t-{}-\t{} AUD'.format(i['med_name'], i['med_qty'], i['med_price']*i['med_qty']))
        # print a line between sections
        table.add_row('=' * 50)
        # print out header for section of total
        table.add_row('\t\t\tTotal')
        # calculate total price and print out
        for i in cart:
            total = total + (float(i['med_price'])*i['med_qty'])
        table.add_row('\t\t\t${}'.format(total))
        # print a line between sections
        table.add_row('=' * 50)
        # output thank you message
        table.add_row('\n\t{}\n'.format(message))
        console.print(table)
        # create a bottom border
        print('-' * 50)
        # Creating an Invoice file in project folder
        now = datetime.now()
        print(f"Invoice Date ={now}\n")
        dt_string = now.strftime("%d%m%Y%H%M%S")
        invoice_number = 'SP'+dt_string
        print(f"Invoice Number ={invoice_number}\n")
        print('#' * 50)
```

Billing and Invoice

- Uses rich library to display invoice in a tabular form to the terminal
- The items in the cart is displayed first.
- Once the user confirms to proceed with the billing, the invoice is displayed.
- Unique invoice number is generated as of the format ‘SP<ddmmyyyyHHMMSS>’
- After the invoice is displayed , the invoice file is generated in the next step.

```
        invoice_file_name = 'src/steffs_pharmacy/data/invoice_'+invoice_number
        with open(invoice_file_name, 'w') as f:
            # create a top border
            f.write('#' * 50)
            # print company information first using format
            f.write('\n\t\t{}'.format(company_name.title()))
            f.write('\n\t\t{}'.format(company_address.title()))
            f.write('\n\t\t{}'.format(company_city.title()))
            # create a top border
            f.write('\n')
            f.write('-' * 50)
            # print out header for section of items
            f.write('\n\tProduct Name\tQuantity\tPrice\n')
            # create a print statement for each item
            for i in cart:
                f.write(
                    '\t{}\t-{}-\t{} AUD'.format(i['med_name'], i['med_qty'], i['med_price']*i['med_qty']))
                f.write('\n')
            # print a line between sections
            f.write('\n')
            f.write('=' * 50)
            # print out header for section of total
            f.write('\n\t\t\tTotal')
            # calculate total price and print out
            f.write('\n\t\t\t${}'.format(total))
            # print a line between sections
            f.write('\n')
            f.write('=' * 50)
            # output thank you message
            f.write('\n\t{}\n'.format(message))
            # create a bottom border
            f.write('\n')
            f.write('-' * 50)
            f.write(f"\nInvoice Date ={now}\n")
            f.write(f"Invoice Number ={invoice_number}\n")
            f.write('#' * 50)
            enter_key = input("Press 'Enter' to continue..")
            return invoice_number
    elif billing_user_input.lower() == "n":
        # Return to  customer order menu when the input is 'n'
        print("Returning to the Cutomer Order Menu..")
        time.sleep(1)
        return None
```

Billing and Invoice : Generate Invoice file

- The invoice file is opened in the write mode.
- The cart details , the invoice date and number is written to the invoice file.
- The invoice file is of the format ‘invoice_<invoice number>’
- This file is placed in the /data/ folder.
- The billing screen is exited once the user presses the ‘Enter’ key

```python
def display_invoice_report():
    console = Console()
    # clear screen
    clearing.clear()
    # Read the invoices  report
    sales_report = pd.read_excel("src/steffs_pharmacy/data/invoices.xlsx")
    table_stock = Table(show_header=False, header_style="bold blue")
    table_stock.add_row(
        "----------------------------------SALES REPORT----------------------------------")
    console.print(table_stock)
    # print the report to terminal
    print(f"{sales_report.to_string()}\n")
    print("-"*36, "END-OF-REPORT", "-"*36)
    # Hit enter key to return
    enter_key = input("Press 'Enter' to continue.")
```

Display Sales report

- The invoices excel sheet is read using pandas library function ‘read_excel’.
- The dataframes are displayed to the terminal screen.

```python
def display_stock_report():
    console = Console()
    # clear screen
    clearing.clear()
    # read the inventory data from  the xls file
    stock_report = pd.read_excel(
        "src/steffs_pharmacy/data/medicine_inventory.xlsx")
    table_stock = Table(show_header=False, header_style="bold blue")
    table_stock.add_row(
        "-------------------------------------------------------------STOCK REPORT------
    # display the heading
    console.print(table_stock)
    # display the stock report
    print(f"{stock_report.to_string()}\n")
    print("-"*39, "END-OF-REPORT", "-"*39)
    # press enter to continue
    enter_key = input("Press 'Enter' to continue.")
```

Display stock report

- The inventory excel sheet is read using pandas library function ‘read_excel’.
- The dataframes are displayed to the terminal screen.

Development / Build Process:

- Used Trello for project management . Checklist for each feature created to track the progress.
- Draft flowchart was created to give the direction in coding.
- Each feature was developed and tested thoroughly . Including negative test cases.
- PEP8 Style is used throughout the code as the standard.

Challenges:

- Arriving at a format for the inventory was a bit challenging. Arrived at excel/csv format for inventory and sales data as it is easier to create the sample data.
- Another challenge was on how to display the data into the terminal . Pandas, Rich libraries are used extensively for this purpose.

Favourite Part:

- The most interesting part of the whole development is the testing of each feature and fixing apparent issues and unhandled errors.