Computing Coursework

Henry Mason

March 12, 2015

Contents

Chapter 1

Analysis

1.1 Introduction

1.1.1 Client Identification

My client is Susannah Mason, she is 50 years old and has little usage of computers, except when having to order new stock for the pharmacy. currently the pharmacy uses computerised methods to submit orders to the warehouse.

Suesannah is a pharmacutical manager at spire healthcare in impington

by creating this program it would speed up the process making leeping track of and ordering of new equipment and stock alot easier for her

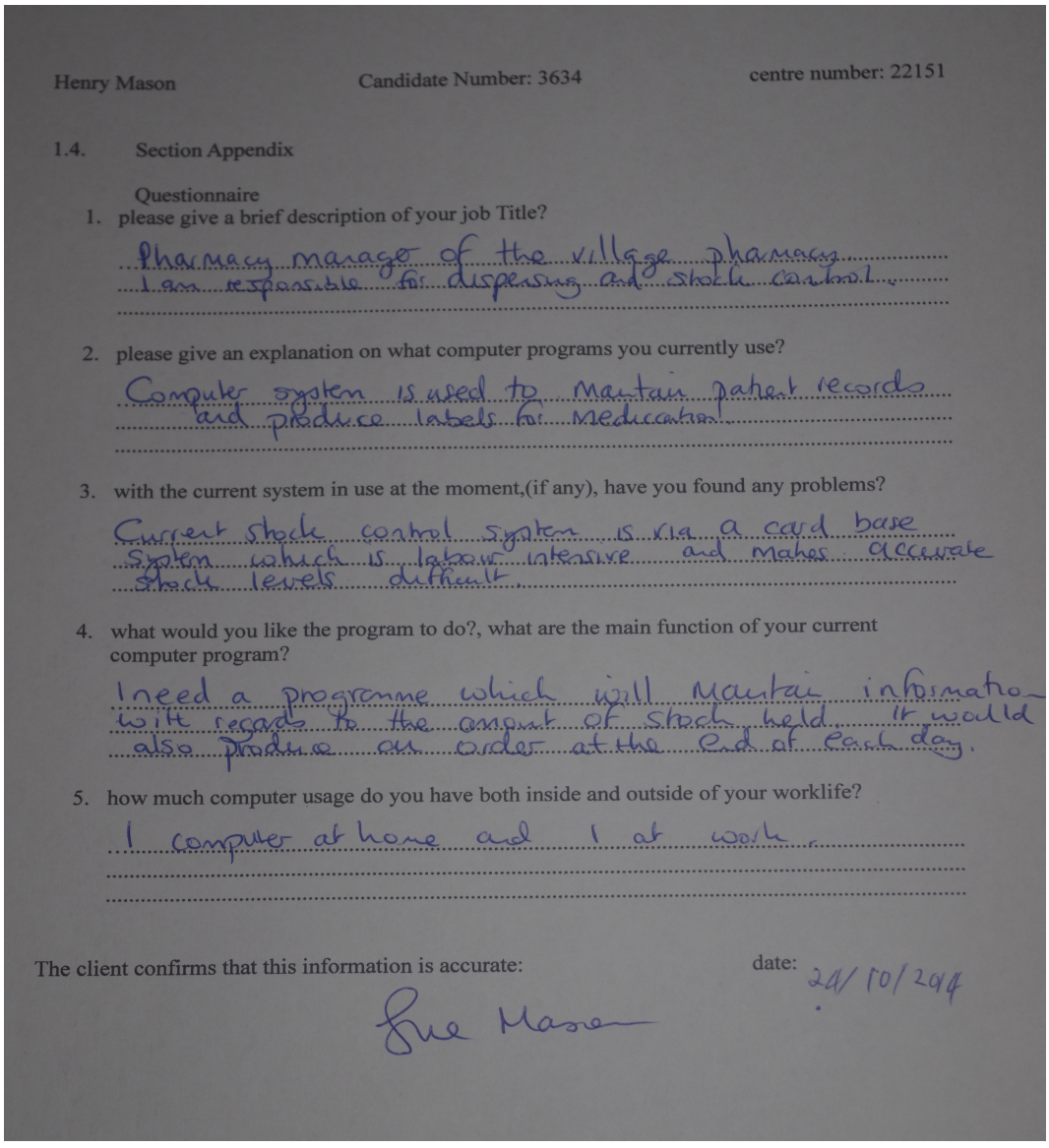
1.1.2 Define the current system

the current system uses mostly computer based order submission and price checks but the orders have to be put through the computer manually

1.1.3 Describe the problems

the orders for the stock take too long to submit and all stock has to be conted by hand

1.1.4 Section appendix



Henry Mason Candidate Number: 3634 centre number: 22151

1.4. Section Appendix

Questionnaire

1. please give a brief description of your job Title?

Pharmacy manage of the village pharmacy. I am responsible for dispensing and stock control.

2. please give an explanation on what computer programs you currently use?

Computer system is used to maintain patient records and produce labels for medication.

3. with the current system in use at the moment,(if any), have you found any problems?

Current stock control system is via a card base system which is labour intensive and makes accurate stock levels difficult.

4. what would you like the program to do?, what are the main function of your current computer program?

I need a programme which will maintain information with regards to the amount of stock held. It would also produce an order at the end of each day.

5. how much computer usage do you have both inside and outside of your worklife?

1 computer at home and 1 at work.

The client confirms that this information is accurate: date: 24/10/2014

Sue Mason

Figure 1.1: questionaire

1.2 Investigation

1.2.1 The current system

the current system at the pharmacy is a data base that holds the information of over 500 items. the data base holds the price the mass the desription and how much is in the pharmacy at that point in time. when an item is taken out of stock the pharmacist has a card to say that an item has been removed from the storage cupboard. sometimes the system deosn't update even when the card is swiped to say a product has been removed

Data sources and destinations

Data Source	Travels via	destination
doctor	gives prescription	patient
patient	requests medicine	pharmacist
pharmacist	checks stock	stock system
stock system	gives information	pharmasict
pharmacist	collects medication	medicine cupboard
pharmacist	gives medicine	patient

Table 1.1:

Algorithms

i will be using quite a few algorithums for this assignment

Algorithm 1 if statement

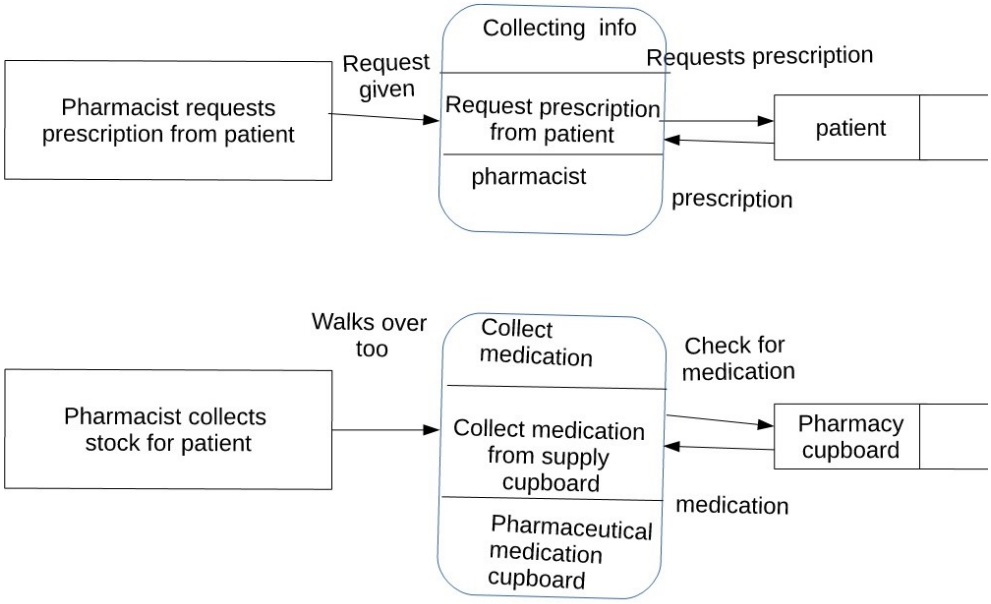
```
1: FOR EACH item to check = 0 to 50 IN
2:     IF  THENDOitem = lowest minimum amount
3:         "you don't need any more tablets"
4:     ELSE
5:         "you need more tablets"
6:     END IF
7: END FOR
```

this other algorithm will be used to calculate the exact price of all of the order using the information in the list Items the exact price is calulated

Data flow diagram

```
IF order Submitted = True THEN
    calculate Order
ELSE
    Restart stock check
END IF
```

```
IF item in items = True THEN
    total ← total + itemPrice
ELSE
    total ← total
END IF
```



Input Forms, Output Forms, Report Formats

pres.JPG

1.2.2 The proposed system

the proposed system will be used to order, check stock and be informed as soon as anything leaves the pharmacy the data base will be updated of the removal, as well as if the product falls below a certain point it will be program to replace the stock by ordering new stock form the wearhouse automatically but the order will go through a master contol point before being sent off

Data sources and destinations

Data Source	Travels via	destination
doctor	sends email	pharmacy
pharmacist	checks stock	stock system
stock system	gives information	pharmasict
pharmacist	collects medication	medicine cupboard
patient	collects from	pharmacy

Data flow diagram

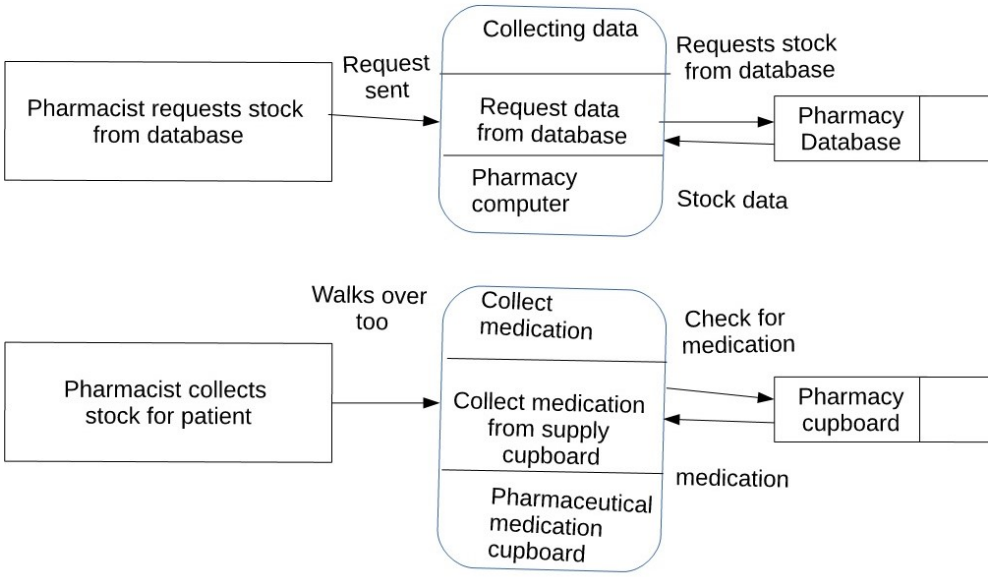


Figure 1.2: data flow diagram

Data	Uses	Name
stock detail	stock number	stock check
prescription information	mediction needed	prescription
enough item in stock False	order more item	update stock

Data dictionary

Volumetrics

this system should only be used by pharmacutical staff in hospitals my client uses this system normally after every patient has gone through to update the stock. so my predicted amount of memory used by the system should calculate upt around 64 bytes to 256 bytes.

1.3 Objectives

1.3.1 General Objectives

- to make a stable system that checks, updates, restocks and sends payment for the ordered items
- to give the system to auto restock items when they fall below a certain number of items
- to graph which items are being bought or used faster and updates the resocking system acordingly

1.3.2 Specific Objectives

- to design a program that will make sorting through the items at the pharmacy as well as store the price and item location in the pharmacy as well as the amount.

1.3.3 Core Objectives

- self updating stock system
- easy accessablilty
- order more items to refill stock

1.3.4 Other Objectives

- the stock keeping on the program should be accurate. E.G. showing how much on tablet of paracitamol costs
- the system must have a automatic communication between the wholesale (warehouse) and the pharmacy

1.4 ER Diagrams and Descriptions

1.4.1 ER Diagram

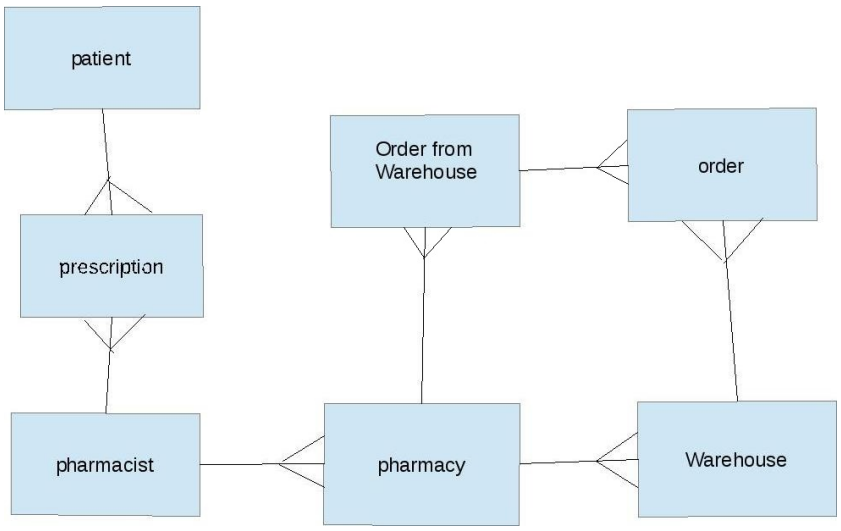


Figure 1.3: entity relationship diagram

1.4.2 Entity Descriptions

- Client(clientID, PharmacyNum, surname, FirstName, PhoneNumber, Address,Postcode)
- Pharmacist(PharamacistID,*PharmacyNum*,Surname,FirstName,PhoneNumber,Address,Email)
- Pharmacy(PharmacyNum,PharmacyAddress,PharmacyPhoneNumber)
- Warehouse(WareHouseNum,*PharmacyAddress*,WareHouseAddress)
- Order(OrderNum, *WareHouseNum*,*PharmacyLocation*,OrderDate,size)

1.5 Object Analysis

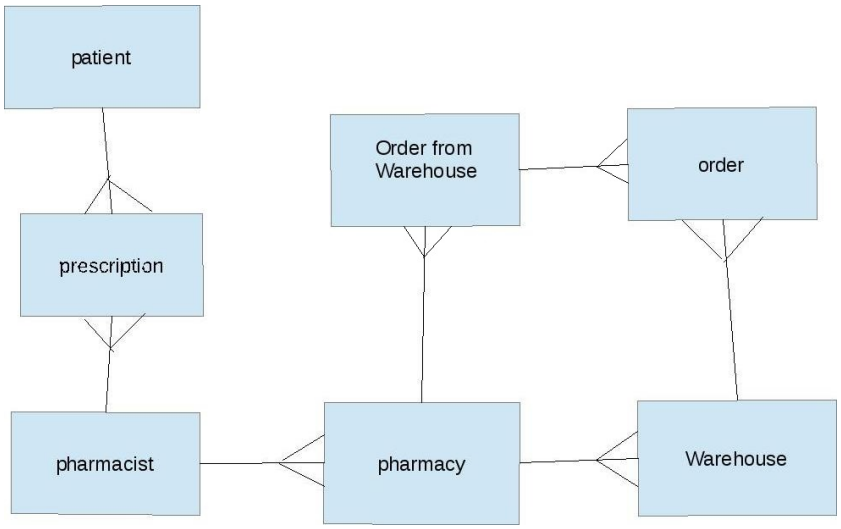


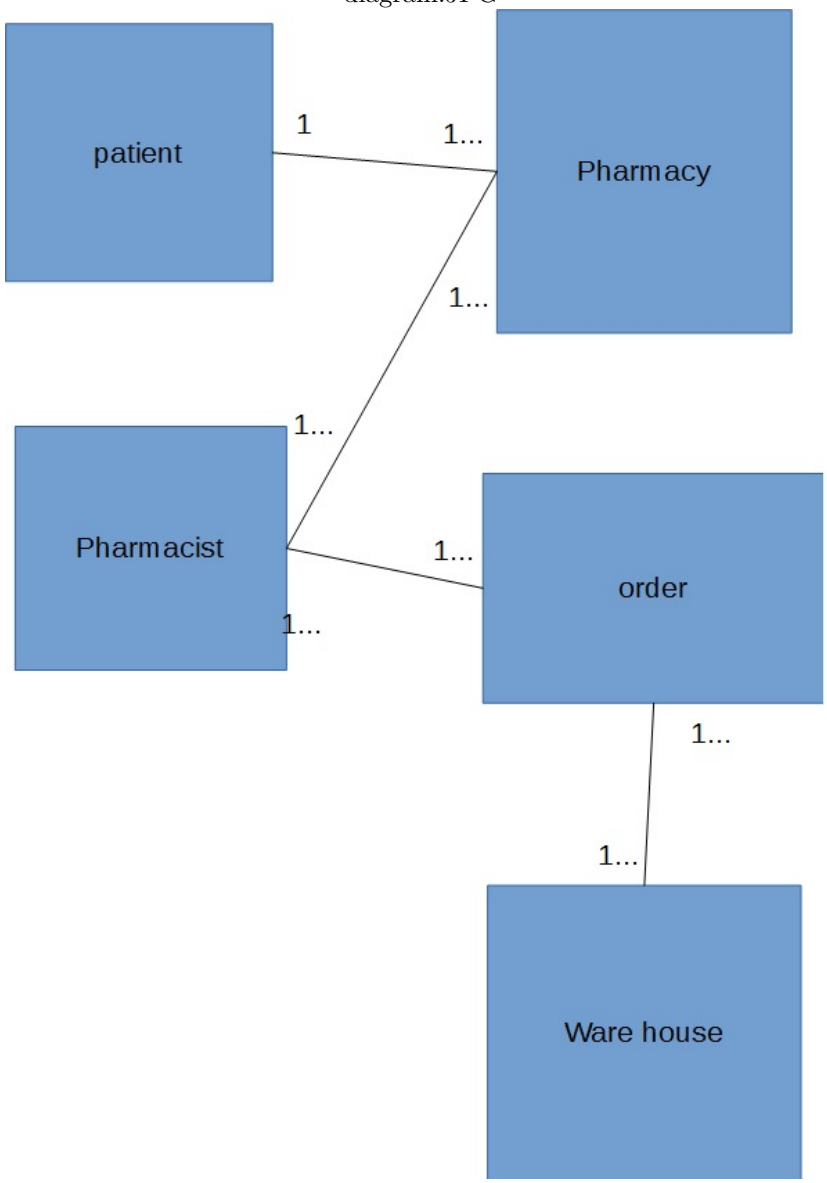
Figure 1.4: entity relationship diagram

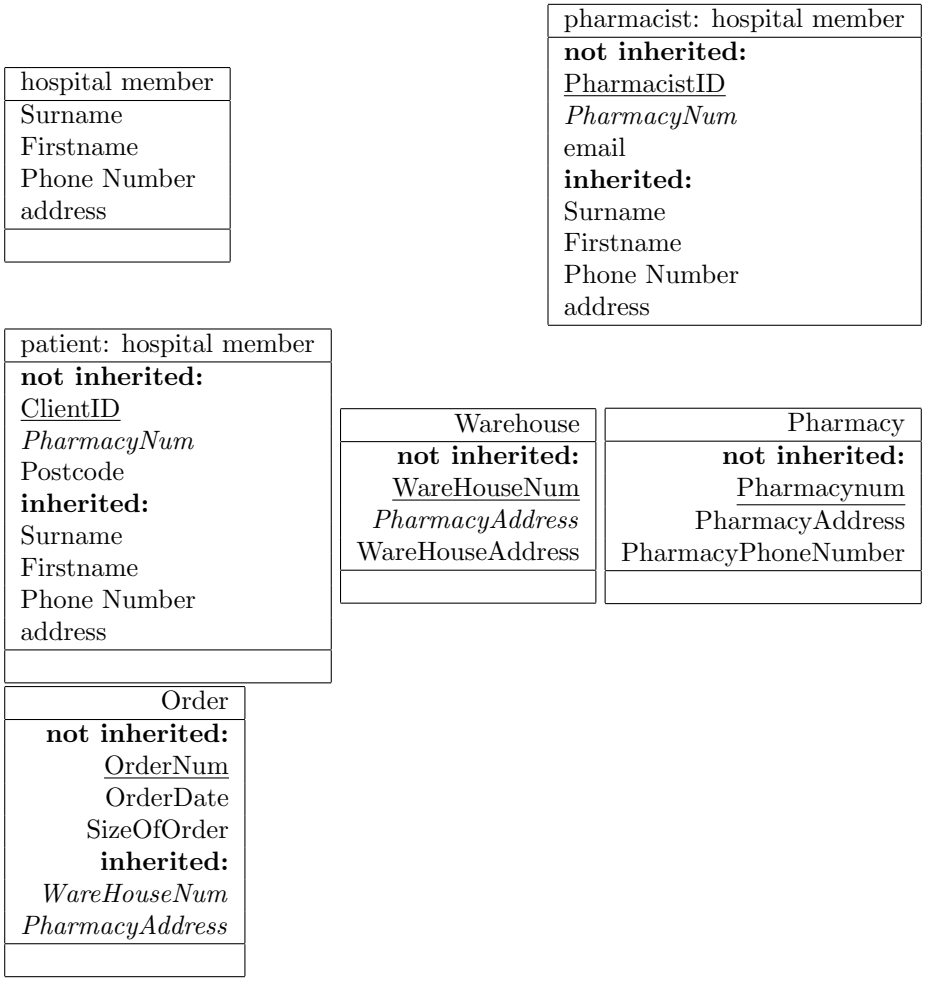
1.5.1 Object Listing

- Client
- Pharmacist
- Pharmacy
- Warehouse
- Order

1.5.2 Relationship diagrams

diagram.JPG





1.5.3 Class definitions

1.6 Constraints

1.6.1 Hardware

Susannah uses a small laptop computer. the components of that laptop are listed below:

- 10.1" display
- intel core N270 atom 1.6 ghz
- 1.00GB DDR3 RAM

- 160GB HDD

The proposed system should work on this laptop because of the laptops fast processor it will run through the calculations fast enough.

If using the laptop doesn't work i will switch back to my desktop computer which has:

- 17" display 1024 x 768 pixels
- 34" display 1360 x 768 pixels
- amd A6-3500 APU 3.0 ghz
- 4.00GB DDR3 RAM
- 0.5TB HDD

There should be no problems running the program on this computer.

1.6.2 Software

The operating system used on the laptop is windows xp. Whereas the operating system used on the desktop is running windows 7. The programs that i will be using will be python 3.2.

1.6.3 Time

The final submission of the implimentation of the program must be in by the 13th of February

1.6.4 User Knowledge

The system i will be building will require at least a little background into the pharmacutical area. This is due to some of the medicines in the system do not have abbreviated names. The installation should take about 5 to 10 minutes.

1.6.5 Access restrictions

the proposed system should only be accessable and privileges to the people in pharmacy, as well as the system should be password protectedto ensure no body outside the system can access the stock information.

1.7 Limitations

1.7.1 Areas which will not be included in computerisation

the prescriptions are not given in electronic format so when the patients come to the pharmacy to collect there medicine it has to be collected by the pahrmacists once the patient has got to the pharmacy

1.7.2 Areas considered for future computerisation

the prescriptions should be sent by the doctors to the pharmacy before the patient leaves the doctor so the pharmacy have time to prepare for the patient so they can pick up there prescription and pay and leave all within the space of one minute.

1.8 Solutions

1.8.1 Alternative solutions

solution	advantages	disadvantages
created program that checks the stock after an item is removed	this will keep a continually accurate stock check	this will take up more space to program in

Table 1.2:

1.8.2 Justification of chosen solution

I have chosen to the Python 3.2 desktop application with a GUI and SQL' solution. My reason for using this method is:

- the application will be specific for pharmacy which will be updated at the start of every week and will continuously keep track of the database where the old system.
- the database used will take up less space required to store the data.
- due to the databases size making back ups is very easy so if the system.

Chapter 2

Design

2.1 Overall System Design

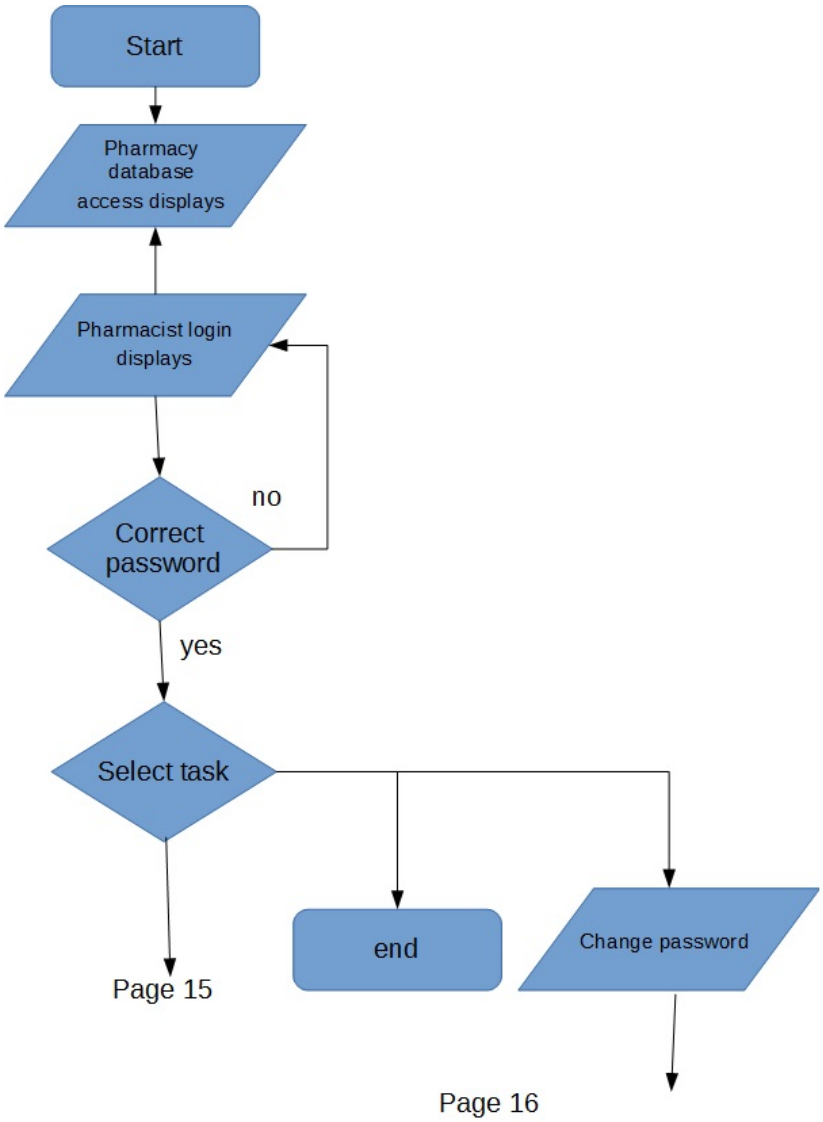
2.1.1 Short description of the main parts of the system

my system will consist of 5 main parts, using this information i have given a short description of what the system is meant to do:

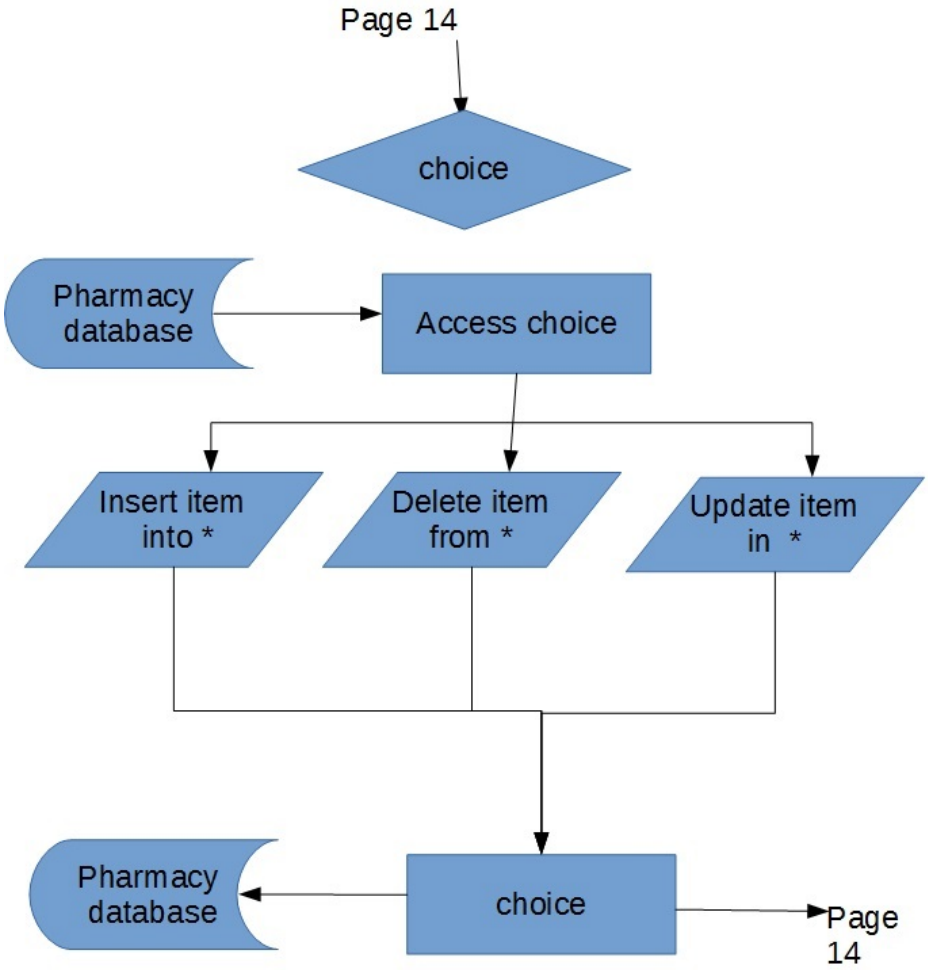
- Log in window
  - this will display a login window and the central widget has input boxes for the user to input there user name and password
  - checks in the pharmacy's database to find if the user is registered to the system and has prority to enter the database.
  - the system then checks the users name and finds the password registered to that user. if the password entered is not the same the user will be asked to renter the password after the input box has been cleared.
- Database display
  - once the user has been accepted into the database the stock will come up on screen for the user
  - the system will then check the stock to see if the amount of items is up to the minimum level and if the stock isn't to the minimum level then the system begins to request for more stock.
  - if the system find's an error e.g. missing stock or calculation problem it will give an error report to the user.
- stock order form

- if the system find's a product low the system will bring up a order request form which will be sent to the warehouse.
-

2.1.2 System flowcharts showing an overview of the complete system



flow 1.JPG



flow 2.JPG

Figure 2.1: system flow

2.2 User Interface Designs

2.3 hardware specification

2.4 Program Structure

2.4.1 Top-down design structure charts

2.4.2 Algorithms in pseudo-code for each data transformation process

2.4.3 Object Diagrams

2.4.4 Class Definitions

2.5 Prototyping

2.6 Definition of Data Requirements

2.6.1 Identification of all data input items

2.6.2 Identification of all data output items

2.6.3 Explanation of how data output items are generated

2.6.4 Data Dictionary

Data	Uses	Name
stock detail	stock number	stock check
prescription information	mediction needed	prescription
enough item in stock False	order more item	update stock

2.6.5 Identification of appropriate storage media

2.7 Database Design

2.7.1 Normalisation

ER Diagrams

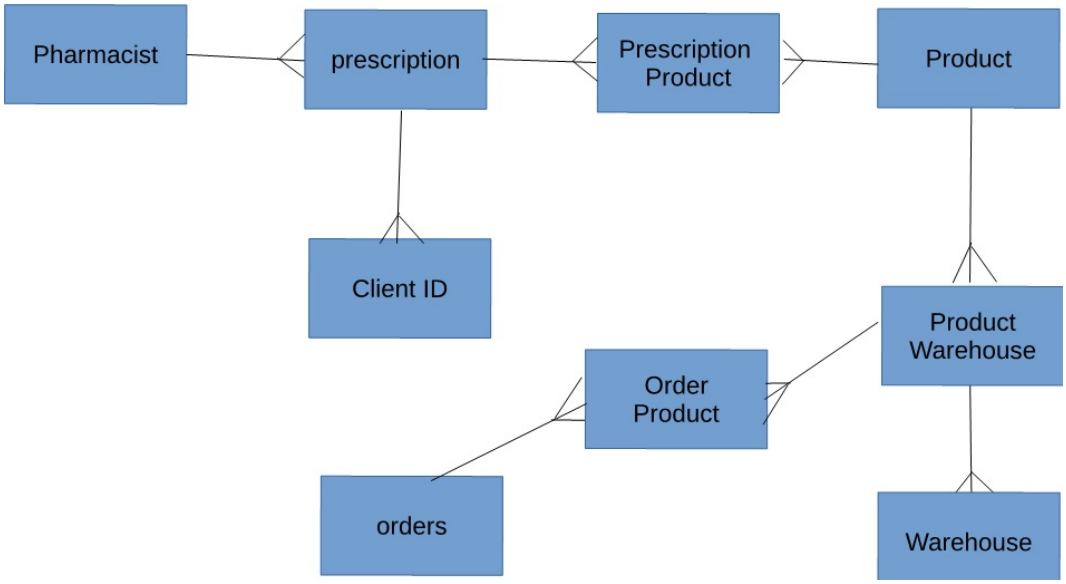


Figure 2.2: ERdiagram

Entity Descriptions

- Client(clientID, FirstName, surname, ClientPhoneNumber, Town, streetName,HouseNumber/name,Postcode)
- Product(ProductID,ProductName, ProductWeight, ProductCode, Manufacturer, Price)
- Pharmacist(PharamacistID, PharmacistNum, Pharmacistname, PharmacistEmail, PharmacistTown, PharmacistStreet, PharmacistPostcode)
- Warehouse(WareHouseNum, WarehouseTown, WarehouseStreet, WarehousePostcode)

- PrescriptionCode(PrescriptionCode, *PharmacistID*, *ClientID*, Quantity-OfMed)
- Order(OrderNum, *WareHouseNumber*, *ProductID*, OrderDate, size)
- Product on prescription(PrescritionProduct, *ProductID*, *PrescritionCode*)
- product delivered from warehouse(ProductWareHouse, *ProductID*, *Ware-HouseNumber*)
- Product ordered (OrderProduct, *ProductWareHouse*, *Order*)

UNF

Un-Normalised
ClientID
PharmacyNum
Surname
Firstname
ClientPhoneNumber
ClientAddress
Postcode
PrescriptionCode
PharmacistID
PharmacistTown
PharmacistStreet
PharmacistPostcode
PharmacistEmail
PharmacyAddress
PharmacyPhoneNumber
OrderNum
OrderDate
size
ProductCode
QuantityOfMed
Weight
ProductName
Manufacturer
Price
Town
Postcode
StreetName
HouseNumber/Name
WareHouseNumber
WareHouseStreet
WareHouseTown
WarehousePostcode

1NF to 3NF

1NF

non-repeating	**repeating**
PharmacyNum	ClientID
OrderDate	PharmacyNum
OrderNum	FirstName
Pharmacystreet	Surname
PharmacyTown	ClientPhoneNumber
PharmacyPostcode	Town
	Postcode
	HouseNumber/Name
	StreetName
	ProductCode
	QuantityOfMed
	ProductWeight
	ProductName
	PrescriptionCode
	Manufacturer
	Price
	size
	PharmacistName
	PharmacistNumber
	PharmacistTown
	PharmacistStreet
	PharmacistPostcode
	WarehouseNumber
	WarehouseTown
	WarehouseStreet
	WarehousePostcode

2NF

repeating	**Non repeating**
ClientID	PharmacyNum
PharmacyNum	OrderDate
PrescriptionCode	Size
	OrderNum
ClientID	PharmacyStreet
FirstName	PharmacyTown
Surname	PharmacyPostcode
ClientPhoneNumber	PharmacyPhoneNumber
HouseNumber/Name	
Town	
Postcode	
StreetName	
PharmacyNum	
PharmacistNumber	
PharmacistName	
PharmacistEmail	
PharmacistTown	
PharmacistStreet	
PharmacistPostcode	
QuantityOfMed	
ProductName	
ProductWeight	
ProductCode	
Manufacturer	
Price	
WarehouseTown	
WarehouseStreet	
WarehousePostcode	

3NF

ClientID FirstName Surname ClientPhoneNumber HouseNumber/Name Town Postcode StreetName	ProductID ProductName ProductWeight ProductCode Manufacturer Price	OrderNumber *WareHouseNumber* *ProductID* OrderDate Size	PharmacistID PharmacistNumber PharmacistName PharmacistEmail PharmacistTown PharmacistStreet PharmacistPostcode
PrescriptionCode *PharmacistID* *clientID* QuantityOfMed	WareHouseNumber WarehouseTown WarehouseStreet WarehousePostcode	PrescritionProduct *ProductID* *PrescritionCode*	ProductWareHouse *ProductID* *WareHouseNumber*
OrderProduct *ProductWareHouse* *Order*	Manufacturer ManufacturerPostcode ManufacturerStreet ManufacturerTown		

2.8 Security and Integrity of the System and Data

2.8.1 Security and Integrity of Data

2.8.2 System Security

2.9 Validation

2.10 Testing

2.10.1 Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
Example	Example	Example	Example

2.10.2 Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
Example	Example	Example	Example	Example	Example	Example	Example

Chapter 3

Testing

3.1 Test Plan

3.1.1 Original Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
Example	Example	Example	Example

3.1.2 Changes to Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
Example	Example	Example	Example

3.1.3 Original Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
Example	Example	Example	Example	Example	Example	Example	Example

3.1.4 Changes to Detailed Plan

Test Series	**Purpose of Test**	**Test Description**	**Test Data**	**Test Data Type (Normal/ Erroneous/ Boundary)**	**Expected Result**	**Actual Result**	**Evidence**
Example	Example	Example	Example	Example	Example	Example	Example

3.2 Test Data

3.2.1 Original Test Data

3.2.2 Changes to Test Data

3.3 Annotated Samples

3.3.1 Actual Results

3.3.2 Evidence

3.4 Evaluation

3.4.1 Approach to Testing

3.4.2 Problems Encountered

3.4.3 Strengths of Testing

3.4.4 Weaknesses of Testing

3.4.5 Reliability of Application

3.4.6 Robustness of Application

Chapter 4

System Maintenance

4.1 Environment

4.1.1 Software

- Python 3.4
  - Idle for writing the code for the program
  -
  - SQLite3 module for interaction with the database
-
-

4.10.1 Main Edit

```
def run_main_menu():
    print("which area of the database do you wish to access")
    print()
    print("1. customer")
    print("2. products")
    print("3. prescription data")
    print("4. Orders")
    print("5. manufacturers")
    print("0. close the database")
    print("choice : ", end = "")
    try:
        choice = int(input())
    except ValueError:
        print()
        choice, choice1 = run_main_menu()
    if choice == 0:
        choice1 = 0
        return choice, choice1
    elif choice in range(1,6):
        choice1 = run_sub_menu(choice)
    else:
        choice, choice1 = run_main_menu()
    return choice, choice1
def run_sub_menu(choice):
    if choice == 1:
        print("1. insert data into {0}".format("customer"))
```

```
27          print("2. update data in {0}".format("customer"))
28          print("3. display data from {0}".format("customer"))
29          print("4. find item in {0}".format("customer"))
30          print("5. delete item in {0}".format("customer"))
31          print("choice : ", end = "")
32      elif choice == 2:
33          print("1. insert data into {0}".format("product"))
34          print("2. update data in {0}".format("product"))
35          print("3. display data from {0}".format("product"))
36          print("4. find item in {0}".format("product"))
37          print("5. delete item in {0}".format("product"))
38          print("choice : ", end = "")
39      elif choice == 3:
40          print("1. insert data into {0}".format("prescription"))
41          print("2. update data in {0}".format("prescription"))
42          print("3. display data from {0}".format("prescription"))
43          print("4. find item in {0}".format("prescription"))
44          print("5. delete item in {0}".format("prescription"))
45          print("choice : ", end = "")
46      elif choice == 4:
47          print("1. insert data into {0}".format("orders"))
48          print("2. update data in {0}".format("orders"))
49          print("3. display data from {0}".format("orders"))
50          print("4. find item in {0}".format("orders"))
51          print("5. delete item in {0}".format("orders"))
52          print("choice : ", end = "")
53      elif choice == 5:
54          print("1. insert data into {0}".format("manufacturer"))
```

```
55          print("2. update data in {0}".format("manufacturer"))
56          print("3. display data from {0}".format("manufacturer"))
57          print("4. find item in {0}".format("manufacturer"))
58          print("5. delete item in {0}".format("manufacturer"))
59          print("choice : ", end = "")
60      try:
61          choice1 = int(input())
62      except ValueError:
63          print()
64          choice1 = run_sub_menu(choice)
65      return choice1
66
67
68  def run_main():
69      close = False
70      while not close:
71          connect = None
72          choice, choice1 = validate_choice()
73          if choice == 0:
74              close = True
75          elif choice == 1:
76              connect = customer_menu()
77              if choice1 == 1:
78                  connect.run_menu(choice1)
79              elif choice1 == 2:
80                  connect.run_menu(choice1)
81              elif choice1 == 3:
82                  connect.run_menu(choice1)
```

```
83              elif choice1 == 4:
84                  connect.run_menu(choice1)
85              elif choice1 == 5:
86                  connect.run_menu(choice1)
87          elif choice == 2:
88              connect = product_menu()
89              if choice1 == 1:
90                  connect.run_menu(choice1)
91              elif choice1 == 2:
92                  connect.run_menu(choice1)
93              elif choice1 == 3:
94                  connect.run_menu(choice1)
95              elif choice1 == 4:
96                  connect.run_menu(choice1)
97              elif choice1 == 5:
98                  connect.run_menu(choice1)
99          elif choice == 3:
100             connect = prescription_menu()
101             if choice1 == 1:
102                 connect.run_menu(choice1)
103             elif choice1 == 2:
104                 connect.run_menu(choice1)
105             elif choice1 == 3:
106                 connect.run_menu(choice1)
107             elif choice1 == 4:
108                 connect.run_menu(choice1)
109             elif choice1 == 5:
110                 connect.run_menu(choice1)
```

```
111             elif choice == 4:
112                 connect = order_menu()
113                 if choice1 == 1:
114                     connect.run_menu(choice1)
115                 elif choice1 == 2:
116                     connect.run_menu(choice1)
117                 elif choice1 == 3:
118                     connect.run_menu(choice1)
119                 elif choice1 == 4:
120                     connect.run_menu(choice1)
121                 elif choice1 == 5:
122                     connect.run_menu(choice1)
123             elif choice == 5:
124                 connect = manufacturer_menu()
125                 if choice1 == 1:
126                     connect.run_menu(choice1)
127                 elif choice1 == 2:
128                     connect.run_menu(choice1)
129                 elif choice1 == 3:
130                     connect.run_menu(choice1)
131                 elif choice1 == 4:
132                     connect.run_menu(choice1)
133                 elif choice1 == 5:
134                     connect.run_menu(choice1)
135 
136 
137 def validate_choice():
138     choicechecked = False
```

```
139     choice, choice1 = run_main_menu()
140     while not choicechecked:
141         if choice in range(0,6):
142             choicechecked = True
143         else:
144             print()
145             choice, choice1  = run_main_menu()
146     return choice, choice1
147 
148 if __name__ == "__main__":
149     run_main()
```


4.10.2 customer

customer menu

```
1 
2 #menu for managing customer
3 class customer_menu():
4     def __init__(self):
5         self.running = None
6         self.active_detail = customer_manage()
7 
8     def run_menu(self,choice):
9         if choice == 1:
10            done = False
11            while not done:
```

```
            true = False
            NHSNumber = input("please enter the patients NHSnumber: ")
            true = self.run_NHS_validate(NHSNumber)
            if true == True:
                done = True
            else:
                done = False
                print("that is not a valid NHS number")
        FirstName = input("please enter the clients first name: ")
        LastName = input("please enter the clients last name: ")
        HouseNumber= input("please enter the clients house number: ")
        Street = input("please enter the clients street name: ")
        town = input("please enter the clients town name: ")
        postcode = input("please enter the clients Postcode: ")
        print("does the client wish to give their phone number and email address:
            ", end = "")
        done = False
        while not done:
            answer = input()
            answer = answer.lower()
            if answer in ["no","n"]:
                values =
                    (NHSNumber,FirstName,LastName,HouseNumber,Street,town,postcode,
                    "NA","NA")
                done = True
            elif answer in ["yes","y"]:
                phone_number = input("please enter the clients Phone number: ")
                email = input("please enter the clients email: ")
```

```
37                      values = (NHSNumber, FirstName, LastName, HouseNumber, Street,
                            town, postcode, phone_number, email)
38                      done = True
39                  else:
40                      done = False
41                      print("please enter a valid answer: ")
42              self.active_detail.insert_customer_data(values)
43          elif choice == 2:
44              choice2 = input("what id do you want to update: ")
45              print("pass 1-1")
46              choice1 = self.get_answers()
47              print("pass 1-2")
48              if choice1 == 1:
49                  FirstName = input("please enter a first name: ")
50                  data = (FirstName,choice2)
51                  self.active_detail.update_customer_first(data)
52              elif choice1 == 2:
53                  FirstName = input("please enter a first name: ")
54                  LastName = input("please enter a last name: ")
55                  data = (FirstName,LastName,choice2)
56                  self.active_detail.update_customer_lastfirst(data)
57              elif choice1 == 3:
58                  FirstName = input("please enter a first name: ")
59                  LastName = input("please enter a last name: ")
60                  TelephoneNumber = input("please enter their new telephone number: ")
61                  data = (FirstName,LastName,TelephoneNumber,choice2)
62                  self.active_detail.update_customer_telelastfirst(data)
63              elif choice1 == 4:
```

```
64                  FirstName = input("please enter a first name: ")
65                  LastName = input("please enter a last name: ")
66                  TelephoneNumber = input("please enter their new telephone number: ")
67                  EmailAddress = input("please enter their new email address: ")
68                  data = (FirstName,LastName,TelephoneNumber,EmailAddress,choice2)
69                  self.active_detail.update_customer_emailtelelastfirst(data)
70              elif choice1 == 5:
71                  FirstName = input("please enter a first name: ")
72                  LastName = input("please enter a last name: ")
73                  TelephoneNumber = input("please enter their new telephone number: ")
74                  EmailAddress = input("please enter their new email address: ")
75                  HouseNumber = input("please enter their new house number: ")
76                  data = (FirstName, LastName, TelephoneNumber, EmailAddress,
                        HouseNumber, choice2)
77                  self.active_detail.update_customer_housefirst(data)
78              elif choice1 == 6:
79                  FirstName = input("please enter a first name: ")
80                  LastName = input("please enter a last name: ")
81                  TelephoneNumber = input("please enter their new telephone number: ")
82                  EmailAddress = input("please enter their new email address: ")
83                  HouseNumber = input("please enter their new house number: ")
84                  Street = input("please enter their new street name: ")
85                  postcode = input("please enter their new postcode: ")
86                  data = (FirstName, LastName, TelephoneNumber, EmailAddress,
                        HouseNumber, Street, postcode, choice2)
87                  self.active_detail.update_customer_housestreetpostfirst(data)
88              elif choice1 == 7:
89                  LastName = input("please enter a last name: ")
```

```
90                  data = (LastName,choice2)
91                  self.active_detail.update_customer_Last(data)
92              elif choice1 == 8:
93                  LastName = input("please enter a last name: ")
94                  TelephoneNumber = input("please enter their new telephone number: ")
95                  data = (LastName,TelephoneNumber,choice2)
96                  self.active_detail.update_customer_TeleLast(data)
97              elif choice1 == 9:
98                  LastName = input("please enter a last name: ")
99                  TelephoneNumber = input("please enter their new telephone number: ")
100                 EmailAddress = input("please enter their new email address: ")
101                 data = (LastName,TelephoneNumber,EmailAddress,choice2)
102                 self.active_detail.update_customer_EmailTeleLast(data)
103             elif choice1 == 10:
104                 FirstName = input("please enter a first name: ")
105                 LastName = input("please enter a last name: ")
106                 TelephoneNumber = input("please enter their new telephone number: ")
107                 EmailAddress = input("please enter their new email address: ")
108                 HouseNumber = input("please enter their new house number: ")
109                 data = (LastName,TelephoneNumber,EmailAddress,HouseNumber,choice2)
110                 self.active_detail.update_customer_HouseEmailTeleLast(data)
111             elif choice1 == 11:
112                 LastName = input("please enter a last name: ")
113                 TelephoneNumber = input("please enter their new telephone number: ")
114                 EmailAddress = input("please enter their new email address: ")
115                 HouseNumber = input("please enter their new house number: ")
116                 Street = input("please enter their new street name: ")
117                 postcode = input("please enter their new postcode: ")
```

```
118                 data = (LastName, TelephoneNumber, EmailAddress, HouseNumber, Street,
                        postcode, choice2)
119                 self.active_detail.update_customer_housestreetpostLast(data)
120             elif choice1 == 12:
121                 LastName = input("please enter a last name: ")
122                 TelephoneNumber = input("please enter their new telephone number: ")
123                 EmailAddress = input("please enter their new email address: ")
124                 HouseNumber = input("please enter their new house number: ")
125                 Street = input("please enter their new street name: ")
126                 Town = input("please enter their new town name: ")
127                 postcode = input("please enter their new postcode: ")
128                 data = (LastName, TelephoneNumber, EmailAddress, HouseNumber, Street,
                        postcode, town, choice2)
129                 self.active_detail.update_customer_housetownstreetpostLast(data)
130             elif choice1 == 13:
131                 TelephoneNumber = input("please enter their new telephone number: ")
132                 data = (email,choice2)
133                 self.active_detail.update_customer_Tele(data)
134             elif choice1 == 14:
135                 TelephoneNumber = input("please enter their new telephone number: ")
136                 EmailAddress = input("please enter their new email address: ")
137                 data = (phone_number,email,choice2)
138                 self.active_detail.update_customer_EmailTele(data)
139             elif choice1 == 15:
140                 TelephoneNumber = input("please enter their new telephone number: ")
141                 EmailAddress = input("please enter their new email address: ")
142                 HouseNumber = input("please enter their new house number: ")
143                 data = (phone_number,email,HouseNumber,choice2)
```

```
144                 self.active_detail.update_customer_HouseEmailTele(data)
145             elif choice1 == 16:
146                 TelephoneNumber = input("please enter their new telephone number: ")
147                 EmailAddress = input("please enter their new email address: ")
148                 HouseNumber = input("please enter their new house number: ")
149                 Street = input("please enter their new street name: ")
150                 postcode = input("please enter their new postcode: ")
151                 data = (phone_number,email,HouseNumber,Street,postcode,choice2)
152                 self.active_detail.update_customer_housestreetposttele(data)
153             elif choice1 == 17:
154                 TelephoneNumber = input("please enter their new telephone number: ")
155                 EmailAddress = input("please enter their new email address: ")
156                 HouseNumber = input("please enter their new house number: ")
157                 Street = input("please enter their new street name: ")
158                 Town = input("please enter their new town name: ")
159                 postcode = input("please enter their new postcode: ")
160                 data =
                        (phone_number,EmailAddress,HouseNumber,Street,postcode,Town,choice2)
161                 self.active_detail.update_customer_housetownstreetposttele(data)
162             elif choice1 == 18:
163                 EmailAddress = input("please enter their new email address: ")
164                 data = (EmailAddress,choice2)
165                 self.active_detail.update_customer_Email(data)
166             elif choice1 == 19:
167                 EmailAddress = input("please enter their new email address: ")
168                 HouseNumber = input("please enter their new house number: ")
169                 data = (EmailAddress,HouseNumber,choice2)
170                 self.active_detail.update_customer_HouseEmail(data)
```

```
171             elif choice1 == 20:
172                 EmailAddress = input("please enter their new email address: ")
173                 HouseNumber = input("please enter their new house number: ")
174                 Street = input("please enter their new street name: ")
175                 postcode = input("please enter their new postcode: ")
176                 data = (EmailAddress,HouseNumber,Street,postcode,choice2)
177                 self.active_detail.update_customer_housestreetpost(data)
178             elif choice1 == 21:
179                 EmailAddress = input("please enter their new email address: ")
180                 HouseNumber = input("please enter their new house number: ")
181                 Street = input("please enter their new street name: ")
182                 Town = input("please enter their new town name: ")
183                 postcode = input("please enter their new postcode: ")
184                 data = (EmailAddress,HouseNumber,Street,postcode,town,choice2)
185                 self.active_detail.update_customer_housetownstreetpost(data)
186             elif choice1 == 22:
187                 HouseNumber = input("please enter their new house number: ")
188                 data = (HouseNumber,choice2)
189                 self.active_detail.update_customer_House(data)
190             elif choice1 == 23:
191                 HouseNumber = input("please enter their new house number: ")
192                 Street = input("please enter their new street name: ")
193                 postcode = input("please enter their new postcode: ")
194                 data = (HouseNumber,Street,postcode,choice2)
195                 self.active_detail.update_customer_housestreetpost(data)
196             elif choice1 == 24:
197                 HouseNumber = input("please enter their new house number: ")
198                 Street = input("please enter their new street name: ")
```

```
199                 Town = input("please enter their new town name: ")
200                 postcode = input("please enter their new postcode: ")
201                 data = (HouseNumber,Street,postcode,town,choice2)
202                 self.active_detail.update_customer_housetownstreetpost(data)
203             elif choice1 == 25:
204                 FirstName = input("please enter their newfirst name: ")
205                 LastName = input("please enter their new last name: ")
206                 TelephoneNumber = input("please enter their new telephone number: ")
207                 EmailAddress = input("please enter their new email address: ")
208                 HouseNumber = input("please enter their new house number: ")
209                 Street = input("please enter their new street name: ")
210                 Town = input("please enter their new town name: ")
211                 postcode = input("please enter their new postcode: ")
212                 data = (FirstName, LastName, HouseNumber, Street, postcode, town,
                        EmailAddress, phone_number, choice2)
213                 self.active_detail.update_customer_data(data)
214         elif choice == 3:
215             customer = self.active_detail.customer_data()
216             print(customer)
217         elif choice == 4:
218             done = False
219             while not done:
220                 print("would you like to search by ID or by firstname: ",end = "")
221                 choices = input()
222                 if choices in ["ID","Id","id"]:
223                     print("please enter the ID you wish to view: " ,end = "")
224                     id = input()
225                     rename = self.active_detail.display_customer_data(id)
```

```
226                     print(rename)
227                     done = True
228                 elif choices in ["Firstname","firstname"]:
229                     print("please enter the Name you wish to view: ",end = "")
230                     name = input()
231                     rename = self.active_detail.display_customer_data(name)
232                     print(rename)
233                     done = True
234                 else:
235                     print("please enter a valid choice")
236                     done = False
237         elif choice == 5:
238             choice = input("which id do you want to delete: ")
239             self.active_detail.delete_customer_data(choice)
240 
241     def run_secondary_menu(self,NHSNumber):
242             done = False
243             NHSNumber = input("please enter the patients NHSnumber: ")
244             FirstName = input("please enter the clients first name: ")
245             LastName = input("please enter the clients last name: ")
246             HouseNumber= input("please enter the clients house number: ")
247             Street = input("please enter the clients street name: ")
248             town = input("please enter the clients town name: ")
249             postcode = input("please enter the clients Postcode: ")
250             print("does the client wish to give their phone number and email address:
                    ", end = "")
251             while not done:
252                 answer = input()
```

```
                answer = answer.lower()
                if answer in ["no","n"]:
                    values = (NHSNumber, FirstName, LastName, HouseNumber, Street,
                        town, postcode, "NA", "NA")
                    done = True
                elif answer in ["yes","y"]:
                    phone_number = input("please enter the clients Phone number: ")
                    email = input("please enter the clients email: ")
                    values = (NHSNumber, FirstName, LastName, HouseNumber, Street,
                        town, postcode, phone_number, email)
                    done = True
                else:
                    done = False
                    print("please enter a valid answer: ")
            self.active_detail.insert_customer_data(values)

    def get_answers(self):
        print("what do you want to update?")
        print()
        print("1.First_Name")
        print("2.First_Name and Last_name")
        print("3.First_Name, Last_name and Phone_Number")
        print("4.First_Name, Last_name, Phone_Number and email_address")
        print("5.First_Name, Last_name, Phone_Number, email_address and HouseNumber")
        print("6.First_Name, Last_name, Phone_Number, email_address, HouseNumber,
            Street and Postcode")
        print("7.Last_name")
        print("8.Last_name and Phone_Number")
```

```
278             print("9.Last_name, Phone_Number and email_address")
279             print("10.Last_name, Phone_Number, email_address and HouseNumber")#
280             print("11.Last_name, Phone_Number, email_address, HouseNumber, Street and
                    Postcode")
281             print("12.Last_name, Phone_Number, email_address, HouseNumber, Town, Street
                    and Postcode")
282             print("13.Phone_Number")
283             print("14.Phone_Number and email_address")
284             print("15.Phone_Number, email_address and HouseNumber")
285             print("16.Phone_Number, email_address, HouseNumber, Street and Postcode")
286             print("17.Phone_Number, email_address, HouseNumber, Town, Street and Postcode")
287             print("18.email_address")
288             print("19.email_address and HouseNumber")
289             print("20.email_address, HouseNumber, Street and Postcode")
290             print("21.email_address, HouseNumber, Town, Street and Postcode")
291             print("22.HouseNumber")
292             print("23.HouseNumber, Street and Postcode")
293             print("24.HouseNumber, Town, Street and Postcode")
294             print("25.update all")
295             print()
296             print("what is your choice: ",end = "")
297             try:
298                 choice = int(input())
299             except ValueError:
300                print()
301                self.get_answers()
302             return choice
303
```

```
304     def run_NHS_validate(self,NHSNumber):
305         count = 1
306         for each in NHSNumber:
307             each1 = ord(each)
308             if each1 in range(48,57):
309                 if count in range(1,4) or count in range(5,8) or count in range(9,13):
310                     valid1 = True
311                 elif count == 4:
312                     valid2 = False
313                 elif count == 8:
314                     valid3 = False
315             elif each1 == 32:
316                 if count in range(1,4) or count in range(5,8) or count in range(9,13):
317                     valid1 = False
318                 elif count == 4:
319                     valid2 = True
320                 elif count == 8:
321                     valid3 = True
322             count+=1
323         if valid1 == True and valid2 == True and valid3 == True:
324             true = True
325         else:
326             true = False
327         return true
```

customer management

```
    def __init__(self):
        self.accessed = None

    def insert_customer_data(self,values):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "insert into customer (ClientNHSNumber, FirstName, LastName,
               HouseNumber, Street, Town, Postcode, TelephoneNum, EmailAddress) values
               (?,?,?,?,?,?,?,?,?)"
            cursor.execute(sql,values)
            db.commit()

    def update_customer_first(self,data):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update customer set FirstName=? where customerID=?"
            cursor.execute(sql,data)
            db.commit()

    def update_customer_lastfirst(self,data):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update customer set FirstName=?, LastName=? where customerID=?"
            cursor.execute(sql,data)
            db.commit()

    def update_customer_telelastfirst(self,data):
        with sqlite3.connect("pharmacy_database.db") as db:
```

```
27              cursor = db.cursor()
28              sql = "update customer set FirstName=?, LastName=?, TelephoneNum=? where
                    customerID=?"
29              cursor.execute(sql,data)
30              db.commit()
31
32      def update_customer_emailtelelastfirst(self,data):
33          with sqlite3.connect("pharmacy_database.db") as db:
34              cursor = db.cursor()
35              sql = "update customer set FirstName=?, LastName=?, TelephoneNum=?,
                    Emailaddress=? where customerID=?"
36              cursor.execute(sql,data)
37              db.commit()
38
39      def update_customer_housefirst(self,data):
40          with sqlite3.connect("pharmacy_database.db") as db:
41              cursor = db.cursor()
42              sql = "update customer set FirstName=?, LastName=?,TelephoneNum=?,
                    Emailaddress=?, HouseNumber=? where customerID=?"
43              cursor.execute(sql,data)
44              db.commit()
45
46      def update_customer_housestreetpostfirst(self,data):
47          with sqlite3.connect("pharmacy_database.db") as db:
48              cursor = db.cursor()
49              sql = "update customer set FirstName=?, LastName=?,TelephoneNum=?,
                    Emailaddress=?, HouseNumber=?, Street=?, Postcode=? where customerID=?"
50              cursor.execute(sql,data)
```

```
51              db.commit()
52
53      def update_customer_Last(self,data):
54          with sqlite3.connect("pharmacy_database.db") as db:
55              cursor = db.cursor()
56              sql = "update customer set LastName=? where customerID=?"
57              cursor.execute(sql,data)
58              db.commit()
59
60      def update_customer_TeleLast(self,data):
61          with sqlite3.connect("pharmacy_database.db") as db:
62              cursor = db.cursor()
63              sql = "update customer set LastName=?, TelephoneNum=? where customerID=?"
64              cursor.execute(sql,data)
65              db.commit()
66
67      def update_customer_EmailTeleLast(self,data):
68          with sqlite3.connect("pharmacy_database.db") as db:
69              cursor = db.cursor()
70              sql = "update customer set LastName=?,TelephoneNum=?, Emailaddress=? where
                    customerID=?"
71              cursor.execute(sql,data)
72              db.commit()
73
74      def update_customer_HouseEmailTeleLast(self,data):
75          with sqlite3.connect("pharmacy_database.db") as db:
76              cursor = db.cursor()
```

```
77              sql = "update customer set LastName=?,TelephoneNum=?, Emailaddress=?,
                    HouseNumber=? where customerID=?"
78              cursor.execute(sql,data)
79              db.commit()
80
81      def update_customer_housestreetpostLast(self,data):
82          with sqlite3.connect("pharmacy_database.db") as db:
83              cursor = db.cursor()
84              sql = "update customer set LastName=?,TelephoneNum=?, Emailaddress=?,
                    HouseNumber=?, Street=?, postcode=? where customerID=?"
85              cursor.execute(sql,data)
86              db.commit()
87
88      def update_customer_housetownstreetpostLast(self,data):
89          with sqlite3.connect("pharmacy_database.db") as db:
90              cursor = db.cursor()
91              sql = "update customer set LastName=?,TelephoneNum=?, Emailaddress=?,
                    HouseNumber=?, Street=?, postcode=?, town=? where customerID=?"
92              cursor.execute(sql,data)
93              db.commit()
94
95      def update_customer_Tele(self,data):
96          with sqlite3.connect("pharmacy_database.db") as db:
97              cursor = db.cursor()
98              sql = "update customer set TelephoneNum=? where customerID=?"
99              cursor.execute(sql,data)
100             db.commit()
101
```

```
102     def update_customer_EmailTele(self,data):
103         with sqlite3.connect("pharmacy_database.db") as db:
104             cursor = db.cursor()
105             sql = "update customer set TelephoneNum=?, Emailaddress=? where
                    customerID=?"
106             cursor.execute(sql,data)
107             db.commit()
108 
109     def update_customer_HouseEmailTele(self,data):
110         with sqlite3.connect("pharmacy_database.db") as db:
111             cursor = db.cursor()
112             sql = "update customer set TelephoneNum=?, Emailaddress=?, HouseNumber=?
                    where customerID=?"
113             cursor.execute(sql,data)
114             db.commit()
115 
116     def update_customer_housestreetposttele(self,data):
117         with sqlite3.connect("pharmacy_database.db") as db:
118             cursor = db.cursor()
119             sql = "update customer set TelephoneNum=?, Emailaddress=?, HouseNumber=?,
                    Street=?, postcode=? where customerID=?"
120             cursor.execute(sql,data)
121             db.commit()
122 
123     def update_customer_housetownstreetposttele(self,data):
124         with sqlite3.connect("pharmacy_database.db") as db:
125             cursor = db.cursor()
```

```
126             sql = "update customer set TelephoneNum=?, Emailaddress=?, HouseNumber=?,
                    Street=?, postcode=?, town=? where customerID=?"
127             cursor.execute(sql,data)
128             db.commit()
129
130     def update_customer_Email(self,data):
131         with sqlite3.connect("pharmacy_database.db") as db:
132             cursor = db.cursor()
133             sql = "update customer set Emailaddress=? where customerID=?"
134             cursor.execute(sql,data)
135             db.commit()
136
137     def update_customer_HouseEmail(self,data):
138         with sqlite3.connect("pharmacy_database.db") as db:
139             cursor = db.cursor()
140             sql = "update customer set Emailaddress=?, HouseNumber=? where
                    customerID=?"
141             cursor.execute(sql,data)
142             db.commit()
143
144     def update_customer_housestreetpost(self,data):
145         with sqlite3.connect("pharmacy_database.db") as db:
146             cursor = db.cursor()
147             sql = "update customer set Emailaddress=?, HouseNumber=?, Street=?,
                    postcode=? where customerID=?"
148             cursor.execute(sql,data)
149             db.commit()
150
```

```
151     def update_customer_housetownstreetpost(self,data):
152         with sqlite3.connect("pharmacy_database.db") as db:
153             cursor = db.cursor()
154             sql = "update customer set  Emailaddress=?, HouseNumber=?, Street=?,
                    postcode=?, town=? where customerID=?"
155             cursor.execute(sql,data)
156             db.commit()
157 
158     def update_customer_House(self,data):
159         with sqlite3.connect("pharmacy_database.db") as db:
160             cursor = db.cursor()
161             sql = "update customer set HouseNumber=? where customerID=?"
162             cursor.execute(sql,data)
163             db.commit()
164 
165     def update_customer_housestreetpost(self,data):
166         with sqlite3.connect("pharmacy_database.db") as db:
167             cursor = db.cursor()
168             sql = "update customer set HouseNumber=?, Street=?, postcode=? where
                    customerID=?"
169             cursor.execute(sql,data)
170             db.commit()
171 
172     def update_customer_housetownstreetpost(self,data):
173         with sqlite3.connect("pharmacy_database.db") as db:
174             cursor = db.cursor()
175             sql = "update customer set HouseNumber=?, Street=?, postcode=?, town=?
                    where customerID=?"
```

```
176             cursor.execute(sql,data)
177             db.commit()
178
179     def update_customer_data(self,data):
180         with sqlite3.connect("pharmacy_database.db") as db:
181             cursor = db.cursor()
182             sql = "update customer set FirstName=?, LastName=?, HouseNumber=?,
                    Street=?, Postcode=?, Town=?, Emailaddress=?, TelephoneNum=? where
                    customerID=?"
183             cursor.execute(sql,data)
184             db.commit()
185
186     def customer_data(self):
187         with sqlite3.connect("pharmacy_database.db") as db:
188             cursor = db.cursor()
189             cursor.execute("select customerID, FirstName, LastName, ClientNHSNumber
                    from customer ")
190             customer = cursor.fetchall()
191             return customer
192
193     def display_customer_data(self,FirstName):
194         with sqlite3.connect("pharmacy_database.db") as db:
195             cursor = db.cursor()
196             cursor.execute("select * from customer where FirstName=?",(FirstName,))
197             customer = cursor.fetchone()
198             return customer
199
200     def display_customer_data(self,id):
```

```
201         with sqlite3.connect("pharmacy_database.db") as db:
202             cursor = db.cursor()
203             cursor.execute("select * from customer where customerID=?",(id,))
204             customer = cursor.fetchone()
205             return customer
206 
207     def delete_customer_data(self,data):
208         with sqlite3.connect("pharmacy_database.db") as db:
209             cursor = db.cursor()
210             cursor.execute("delete from customer where customerID=?",(data,))
211         db.commit()
```

4.10.3 prescription

prescription menu

```
1  from customer_menu import*
2  from product_management import*
3  
4  
5  #menu for managing customer
6  class prescription_menu():
7      def __init__(self):
8          self.running = None
9          self.active_detail = prescription_manage()
10         self.active_replacement = customer_menu()
11         self.active_products = product_manage()
```

```
12
13      def run_menu(self,choice):
14          if choice == 1:
15              pharmacist_id= "suesannah9635"
16              customer_ID = []
17              Customer_NHS = []
18              count=0
19              customer_ID.append(self.customer_ID_check(customer_ID))
20              number = len(customer_ID)
21              print(number)
22
23              Customer_NHS = self.Customer_NHS_check(Customer_NHS)
24              print(Customer_NHS)
25              customer_Nhs = input('please enter the customer NHS number: ')
26
27              if customer_Nhs in Customer_NHS:
28                  QuantityOfMed = self.run_add_products()
29              elif customer_Nhs not in Customer_NHS:
30                  self.active_replacement.run_secondary_menu(customer_Nhs)
31                  QuantityOfMed = self.run_add_products()
32              values = (pharmacist_id,customer_Nhs,QuantityOfMed)
33              self.active_detail.insert_prescription_data(values)
34          elif choice == 2:
35              id = input("please enter the id of the product you wish to change: ")
36              choice = self.get_answers()
37              prescription_code = self.get_prescription_code(id)
38              print(prescription_code)
39              pharmacist_id = self.get_pharmacist_id(id)
```

```
40              print(pharmacist_id)
41              Customer_id = self.get_customer_id(id)
42              print(Customer_id)
43              QuantityOfMed = self.get_prescription_quantity(id)
44              print(QuantityOfMed)
45              if choice == 1:
46                  pharmacist_id = input("please enter the new pharmacist_id of the the
                        prescription: ")
47                  value = (pharmacist_id,id)
48                  self.active_detail.update_prescription_Pharmacist(value)
49              elif choice == 2:
50                  pharmacist_id = input("please enter the new pharmacist_id of the the
                        prescription: ")
51                  customer_id = input("please enter the new customer_id of the
                        prescription: ")
52                  value = (pharmacist_id,Customer_id,id)
53                  self.active_detail.update_prescription_Pharmacustom(value)
54              elif choice == 3:
55                  customer_id = input("please enter the new customer_id of the
                        prescription: ")
56                  value = (Customer_id,id)
57                  self.active_detail.update_prescription_customer(value)
58              elif choice == 4:
59                  customer_id = input("please enter the new customer_id of the
                        prescription: ")
60                  QuantityOfMed = input("please enter the new QuantityOfMed of the the
                        prescription: ")
61                  value = (Customer_id,QuantityOfMed,id)
```

```
62                  self.active_detail.update_prescription_quatitycustomer(value)
63              elif choice == 5:
64                  QuantityOfMed = input("please enter the new QuantityOfMed of the the
                        prescription: ")
65                  value = (QuantityOfMed,id)
66                  self.active_detail.update_prescription_quantity(value)
67              elif choice == 6:
68                  pharmacist_id = input("please enter the new pharmacist_id of the the
                        prescription: ")
69                  customer_id = input("please enter the new customer_id of the
                        prescription: ")
70                  QuantityOfMed = input("please enter the new QuantityOfMed of the the
                        prescription: ")
71                  value = (pharmacist_id,Customer_id,QuantityOfMed,id)
72                  self.active_detail.update_prescription_data(value)
73          elif choice == 3:
74              prescription = self.active_detail.prescription_data()
75              print(prescription)
76          elif choice == 4:
77              done = False
78              while not done:
79                  print("would you like to search by pharmacist id or by customer id:
                        ",end = "")
80                  choices = input()
81                  choices = choices.lower()
82                  if choices in ["pharmacist id"]:
83                      print("please enter the pharmacist id you wish to view: " ,end =
                            "")
```

```
84                      id = input()
85                      rename = self.active_detail.display_prescription_data(id)
86                      print(rename)
87                      done = True
88                  elif choices in ["customer id"]:
89                      print("please enter the customer id you wish to view: ",end = "")
90                      name = input()
91                      rename = self.active_detail.display_prescription_data(name)
92                      print(rename)
93                      done = True
94                  else:
95                      print("please enter a valid choice")
96                      done = False
97          elif choice == 5:
98              choice = input("which id do you want to delete: ")
99              self.active_detail.delete_prescription_data(choice)
100
101
102     def get_customer_ID(self):
103         with sqlite3.connect("pharmacy_database.db") as db:
104             cursor = db.cursor()
105             cursor.execute("select CustomerID from Customer")
106             CustomerID = cursor.fetchall()
107             return CustomerID
108
109     def get_customer_NHS(self):
110         with sqlite3.connect("pharmacy_database.db") as db:
111             cursor = db.cursor()
```

```
112             cursor.execute("select ClientNHSNumber from Customer ")
113             CustomerID = cursor.fetchall()
114             return CustomerID
115
116
117     def get_answers(self):
118         print("what do you want to update?")
119         print()
120         print("1.pharmacist id")
121         print("2.pharmacist id and client id")
122         print("3.client id")
123         print("4.client id and QuantityOfMed")
124         print("5.QuantityOfMed")
125         print("6.update all")
126         print("what is your choice: ",end = "")
127         try:
128             choice = int(input())
129         except ValueError:
130            print()
131            self.get_answers()
132         return choice
133
134
135     def run_add_products(self):
136         complete = False
137         while not complete:
138             products = self.active_products.product_data()
139             print(products)
```

```
            complete = True
        quantityofmeds = "salt drip"
        return quantityofmeds

    def Customer_NHS_check(self,customer_NHS):
        NHS1 = self.get_customer_NHS()
        for each in NHS1:
            for each1 in each:
                customer_NHS.append(each1)
        return customer_NHS


    def customer_ID_check(self,customer_ID):
        count1=1
        count = 1
        name1 = self.get_customer_ID()
        for each in name1:
            for each1 in each:
                pass
        return each1
```

prescription management

```
class prescription_manage():
    def __init__(self):
        self.accessed = None

```

```
5       def insert_prescription_data(self,values):
6           with sqlite3.connect("pharmacy_database.db") as db:
7               cursor = db.cursor()
8               sql = "insert into Prescription (PharmacistID,CustomerID,QuantityOfMed)
                    values (?,?,?)"
9               cursor.execute(sql,values)
10              db.commit()
11
12
13      def update_prescription_Pharmacist(self,data):
14          with sqlite3.connect("pharmacy_database.db") as db:
15              cursor = db.cursor()
16              sql = "update Prescription set PharmacistID=? where PrescriptionCode=?"
17              cursor.execute(sql,data)
18              db.commit()
19
20
21      def update_prescription_Pharmacustom(self,data):
22          with sqlite3.connect("pharmacy_database.db") as db:
23              cursor = db.cursor()
24              sql = "update Prescription set PharmacistID=?, CustomerID=?where
                    PrescriptionCode=?"
25              cursor.execute(sql,data)
26              db.commit()
27
28      def update_prescription_customer(self,data):
29          with sqlite3.connect("pharmacy_database.db") as db:
30              cursor = db.cursor()
```

```
31              sql = "update Prescription set CustomerID=? where PrescriptionCode=?"
32              cursor.execute(sql,data)
33              db.commit()
34
35      def update_prescription_quatitycustomer(self,data):
36          with sqlite3.connect("pharmacy_database.db") as db:
37              cursor = db.cursor()
38              sql = "update Prescription set CustomerID=?, QuantityOfMed=? where
                    PrescriptionCode=?"
39              cursor.execute(sql,data)
40              db.commit()
41
42      def update_prescription_quantity(self,data):
43          with sqlite3.connect("pharmacy_database.db") as db:
44              cursor = db.cursor()
45              sql = "update Prescription set QuantityOfMed=? where PrescriptionCode=?"
46              cursor.execute(sql,data)
47              db.commit()
48
49      def update_prescription_data(self,data):
50          with sqlite3.connect("pharmacy_database.db") as db:
51              cursor = db.cursor()
52              sql = "update Prescription set PharmacistID=?, CustomerID=?,
                    QuantityOfMed=? where PrescriptionCode=?"
53              cursor.execute(sql,data)
54              db.commit()
55
56      def prescription_data(self):
```

```
57          with sqlite3.connect("pharmacy_database.db") as db:
58              cursor = db.cursor()
59              cursor.execute("select PharmacistID,CustomerID,QuantityOfMed from
                    Prescription ")
60              prescription = cursor.fetchall()
61              return prescription
62
63      def display_prescription_data(self,name):
64          with sqlite3.connect("pharmacy_database.db") as db:
65              cursor = db.cursor()
66              cursor.execute("select * from Prescription where CustomerID=?",(name,))
67              prescription = cursor.fetchone()
68              return prescription
69
70      def display_prescription_data(self,id):
71          with sqlite3.connect("pharmacy_database.db") as db:
72              cursor = db.cursor()
73              cursor.execute("select * from Prescription where PharmacistID=?",(id,))
74              prescription = cursor.fetchone()
75              return prescription
76
77      def delete_prescription_data(self,data):
78          with sqlite3.connect("pharmacy_database.db") as db:
79              cursor = db.cursor()
80              cursor.execute("delete from Prescription where PrescriptionCode=?",(data,))
81              db.commit()
```

4.10.4 product

product menu

```
1
2  #menu for managing customer
3  class product_menu():
4      def __init__(self):
5          self.running = None
6          self.active_detail = product_manage()
7
8      def run_menu(self,choice):
9          if choice == 1:
10             product_name = input("please enter the name of the product: ")
11             product_weight = input("please enter the products weight: ")
12             product_code = input("please enter the product code: ")
13             price = input("please enter the products price: ")
14             values = (product_name,product_weight,product_code,price)
15             self.active_detail.insert_product_data(values)
16         elif choice == 2:
17             choice1 = input("please enter the id of the product you wish to change: ")
18             choice = self.get_answers()
19             if choice == 1:
20                 product_name = input("please enter the name of the product: ")
21                 value = (product_name,choice1)
22                 self.active_detail.update_product_name(value)
23             elif choice == 2:
24                 product_name = input("please enter the name of the product: ")
```

```
25                  product_weight = input("please enter the new weight of the the
                        product: ")
26                  value = (product_name,product_weight,choice1)
27                  self.active_detail.update_product_weightname(value)
28              elif choice == 3:
29                  product_name = input("please enter the name of the product: ")
30                  product_weight = input("please enter the new weight of the the
                        product: ")
31                  ProductCode = input("please enter the new ProductCode of the product:
                        ")
32                  value = (product_name,product_weight,product_code,choice1)
33                  self.active_detail.update_product_codeweightname(value)
34              elif choice == 4:
35                  product_weight = input("please enter the new weight of the the
                        product: ")
36                  value = (product_weight,choice1)
37                  self.active_detail.update_product_weight(value)
38              elif choice == 5:
39                  product_weight = input("please enter the new weight of the the
                        product: ")
40                  ProductCode = input("please enter the new ProductCode of the product:
                        ")
41                  value = (product_weight,product_code,choice1)
42                  self.active_detail.update_product_codeweight(value)
43              elif choice == 6:
44                  product_weight = input("please enter the new weight of the the
                        product: ")
```

```
45                 ProductCode = input("please enter the new ProductCode of the product:
                       ")
46                 price = input("please enter the new price of the the product")
47                 value = (product_weight,product_code,price,choice1)
48                 self.active_detail.update_product_pricecodeweight(value)
49             elif choice == 7:
50                 productCode = input("please enter the new ProductCode of the product:
                       ")
51                 value = (product_code,choice1)
52                 self.active_detail.update_product_code(value)
53             elif choice == 8:
54                 productCode = input("please enter the new ProductCode of the product:
                       ")
55                 price = input("please enter the new price of the the product")
56                 value = (product_code,price,choice1)
57                 self.active_detail.update_product_pricecode(value)
58             elif choice == 9:
59                 price = input("please enter the new price of the the product")
60                 value = (price,choice1)
61                 self.active_detail.update_product_price(value)
62             elif choice == 10:
63                 product_name = input("please enter the name of the product: ")
64                 product_weight = input("please enter the new weight of the the
                       product: ")
65                 productCode = input("please enter the new ProductCode of the product:
                       ")
66                 price = input("please enter the new price of the the product")
67                 value = (product_name,product_weight,product_code,price,choice1)
```

```
68                  self.active_detail.update_product_data(value)
69              elif choice == 3:
70                  product = self.active_detail.product_data()
71                  print(product)
72              elif choice == 4:
73                  done = False
74                  while not done:
75                      print("would you like to search by ID or by productname: ",end = "")
76                      choices = input()
77                      choices = choices.lower()
78                      if choices in ["id"]:
79                          print("please enter the ID you wish to view: " ,end = "")
80                          id = input()
81                          rename = self.active_detail.display_product_data(id)
82                          print(rename)
83                          done = True
84                      elif choices in ["productname"]:
85                          print("please enter the Name you wish to view: ",end = "")
86                          name = input()
87                          rename = self.active_detail.display_product_data(name)
88                          print(rename)
89                          done = True
90                      else:
91                          print("please enter a valid choice")
92                          done = False
93              elif choice == 5:
94                  choice = input("which id do you want to delete: ")
95                  self.active_detail.delete_product_data(choice)
```

```
96
97      def get_product_name(self,id):
98          with sqlite3.connect("pharmacy_database.db") as db:
99              cursor = db.cursor()
100             cursor.execute("select ProductName from Product where ProductID=?",(id,))
101             Product = cursor.fetchone()
102             return Product
103
104     def get_product_weight(self,id):
105         with sqlite3.connect("pharmacy_database.db") as db:
106             cursor = db.cursor()
107             cursor.execute("select ProductWeight from Product where ProductID=?",(id,))
108             Product = cursor.fetchone()
109             return Product
110
111     def get_product_code(self,id):
112         with sqlite3.connect("pharmacy_database.db") as db:
113             cursor = db.cursor()
114             cursor.execute("select Productcode from Product where ProductID=?",(id,))
115             Product = cursor.fetchone()
116             return Product
117
118     def get_product_price(self,id):
119         with sqlite3.connect("pharmacy_database.db") as db:
120             cursor = db.cursor()
121             cursor.execute("select Price from Product where ProductID=?",(id,))
122             Product = cursor.fetchone()
123             return Product
```

```
124             db.commit()
125
126         def get_answers(self):
127             print("what do you want to update?")
128             print()
129             print("1.product name")
130             print("2.product name and product weight")
131             print("3.product name, product weight and product code ")
132             print("4.product weight")
133             print("5.product weight and product code")
134             print("6.product weight, product code and price")
135             print("7.product code")
136             print("8.product code and price")
137             print("9.price")
138             print("10.update all")
139             print("what is your choice: ",end = "")
140             try:
141                 choice = int(input())
142             except ValueError:
143                print()
144                self.get_answers()
145             return choice
```


product management

```
1  class product_manage():
2      def __init__(self):
```

```
3           self.accessed = None
4
5       def insert_product_data(self,values):
6           with sqlite3.connect("pharmacy_database.db") as db:
7               cursor = db.cursor()
8               sql = "insert into Product (ProductName,ProductWeight,ProductCode,Price)
                    values (?,?,?,?)"
9               cursor.execute(sql,values)
10              db.commit()
11
12      def update_product_name(self,data):
13          with sqlite3.connect("pharmacy_database.db") as db:
14              cursor = db.cursor()
15              sql = "update Product set ProductName=? where ProductID=?"
16              cursor.execute(sql,data)
17              db.commit()
18
19      def update_product_weightname(self,data):
20          with sqlite3.connect("pharmacy_database.db") as db:
21              cursor = db.cursor()
22              sql = "update Product set ProductName=?,ProductWeight=? where ProductID=?"
23              cursor.execute(sql,data)
24              db.commit()
25
26      def update_product_codeweightname(self,data):
27          with sqlite3.connect("pharmacy_database.db") as db:
28              cursor = db.cursor()
```

```
29              sql = "update Product set ProductName=?,ProductWeight=?,ProductCode=?
                    where ProductID=?"
30              cursor.execute(sql,data)
31              db.commit()
32
33      def update_product_weight(self,data):
34          with sqlite3.connect("pharmacy_database.db") as db:
35              cursor = db.cursor()
36              sql = "update Product set ProductWeight=? where ProductID=?"
37              cursor.execute(sql,data)
38              db.commit()
39
40      def update_product_codeweight(self,data):
41          with sqlite3.connect("pharmacy_database.db") as db:
42              cursor = db.cursor()
43              sql = "update Product set ProductWeight=?,ProductCode=? where ProductID=?"
44              cursor.execute(sql,data)
45              db.commit()
46
47      def update_product_pricecodeweight(self,data):
48          with sqlite3.connect("pharmacy_database.db") as db:
49              cursor = db.cursor()
50              sql = "update Product set ProductWeight=?,ProductCode=?,Price=? where
                    ProductID=?"
51              cursor.execute(sql,data)
52              db.commit()
53
54      def update_product_code(self,data):
```

```
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update Product set ProductCode=? where ProductID=?"
            cursor.execute(sql,data)
            db.commit()

    def update_product_pricecode(self,data):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update Product set ProductCode=?,Price=? where ProductID=?"
            cursor.execute(sql,data)
            db.commit()

    def update_product_price(self,data):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update Product set Price=? where ProductID=?"
            cursor.execute(sql,data)
            db.commit()

    def update_product_data(self,data):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update Product set
               ProductName=?,ProductWeight=?,ProductCode=?,Price=? where ProductID=?"
            cursor.execute(sql,data)
            db.commit()

```

```
82      def product_data(self):
83          with sqlite3.connect("pharmacy_database.db") as db:
84              cursor = db.cursor()
85              cursor.execute("select ProductID, ProductName, ProductCode from Product ")
86              Product = cursor.fetchall()
87              return Product
88
89      def display_product_data(self,name):
90          with sqlite3.connect("pharmacy_database.db") as db:
91              cursor = db.cursor()
92              cursor.execute("select * from Product where ProductName=?",(name,))
93              Product = cursor.fetchone()
94              return Product
95
96      def display_product_data(self,id):
97          with sqlite3.connect("pharmacy_database.db") as db:
98              cursor = db.cursor()
99              cursor.execute("select * from Product where ProductID=?",(id,))
100             Product = cursor.fetchone()
101             return Product
102
103     def delete_product_data(self,data):
104         with sqlite3.connect("pharmacy_database.db") as db:
105             cursor = db.cursor()
106             cursor.execute("delete from Product where ProductID=?",(data,))
107             db.commit()
```

4.10.5 order

order menu

```
1
2  #menu for managing customer
3  class order_menu():
4      def __init__(self):
5          self.running = None
6          self.active_detail = orders_manage()
7
8      def run_menu(self,choice):
9          if choice == 1:
10             order_date = input("please enter the order date: ")
11             order_size = input("please enter the size of the order: ")
12             values = (order_date,order_size)
13             self.active_detail.insert_order_data(values)
14         elif choice == 2:
15             id = input("please enter the id of the product you wish to change: ")
16             choice = self.get_answers()
17             if choice == 1:
18                 order_date = input("please enter the date of the order: ")
19                 value = (order_date,id)
20                 self.active_detail.update_order_date(value)
21             elif choice == 2:
22                 order_size = input("please enter the new size of the order: ")
23                 value = (order_size,id)
24                 self.active_detail.update_order_size(value)
```

```
25              elif choice == 3:
26                  order_date = input("please enter the date of the order: ")
27                  order_size = input("please enter the new size of the order: ")
28                  value = (order_date,order_size,id)
29                  self.active_detail.update_order_sizedate(value)
30          elif choice == 3:
31              order = self.active_detail.order_data()
32              print(order)
33          elif choice == 4:
34              done = False
35              while not done:
36                  print("would you like to search by order_num or by order_date: ",end =
                        "")
37                  choices = input()
38                  choices = choices.lower()
39                  if choices in ["order_num","order num","order number","order_number"]:
40                      print("please enter the order number you wish to view: " ,end = "")
41                      id = input()
42                      rename = self.active_detail.display_order_data(id)
43                      print(rename)
44                      done = True
45                  elif choices in ["order_date","order date"]:
46                      print("please enter the customer id you wish to view: ",end = "")
47                      name = input()
48                      rename = self.active_detail.display_order_data(name)
49                      print(rename)
50                      done = True
51                  else:
```

```
52                  print("please enter a valid choice")
53                  done = False
54          elif choice == 5:
55              choice = input("which id do you want to delete: ")
56              self.active_detail.delete_order_data(choice)
57      def get_order_date(self,id):
58          with sqlite3.connect("pharmacy_database.db") as db:
59              cursor = db.cursor()
60              cursor.execute("select OrderDate from Orders where OrderNum=?",(id,))
61              Product = cursor.fetchone()
62
63      def get_order_size(self,id):
64          with sqlite3.connect("pharmacy_database.db") as db:
65              cursor = db.cursor()
66              cursor.execute("select OrderSize from Orders where OrderNum=?",(id,))
67              Product = cursor.fetchone()
68              return Product
69
70      def get_answers(self):
71          print("what do you want to update?")
72          print()
73          print("1.order_date")
74          print("2.order_size")
75          print("3.update all")
76          print("what is your choice: ",end = "")
77          try:
78              choice = int(input())
79          except ValueError:
```

```
            print()
            self.get_answers()
        return choice
```

order management

```
class orders_manage():
    def __init__(self):
        self.accessed = None

    def insert_order_data(self,values):
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "insert into Orders (OrderDate,OrderSize) values (?,?)"
            cursor.execute(sql,values)
            db.commit()

    def update_order_date(self,data):
        print(data)
        with sqlite3.connect("pharmacy_database.db") as db:
            cursor = db.cursor()
            sql = "update Orders set OrderDate=? where OrderNum=?"
            cursor.execute(sql,data)
            db.commit()

    def update_order_size(self,data):
        print(data)
```

```
22          with sqlite3.connect("pharmacy_database.db") as db:
23              cursor = db.cursor()
24              sql = "update Orders set OrderSize=? where OrderNum=?"
25              cursor.execute(sql,data)
26              db.commit()
27
28      def update_order_sizedate(self,data):
29          print(data)
30          with sqlite3.connect("pharmacy_database.db") as db:
31              cursor = db.cursor()
32              sql = "update Orders set OrderDate=?, OrderSize=? where OrderNum=?"
33              cursor.execute(sql,data)
34              db.commit()
35
36      def order_data(self):
37          with sqlite3.connect("pharmacy_database.db") as db:
38              cursor = db.cursor()
39              cursor.execute("select OrderNum, OrderDate, OrderSize from Orders")
40              Product = cursor.fetchall()
41              return Product
42
43      def display_order_data(self,date):
44          with sqlite3.connect("pharmacy_database.db") as db:
45              cursor = db.cursor()
46              cursor.execute("select * from Orders where OrderDate=?",(date,))
47              Product = cursor.fetchone()
48              return Product
49
```

```
50      def display_order_data(self,id):
51          with sqlite3.connect("pharmacy_database.db") as db:
52              cursor = db.cursor()
53              cursor.execute("select * from Orders where OrderNum=?",(id,))
54              Product = cursor.fetchone()
55              return Product
56
57      def delete_order_data(self,data):
58          with sqlite3.connect("pharmacy_database.db") as db:
59              cursor = db.cursor()
60              cursor.execute("delete from Orders where OrderNum=?",(data,))
61              db.commit()
```

Chapter 5

User Manual

5.1 Introduction

5.2 Installation

5.2.1 system requirements

in the design section of this work i put in some system requirements to make this program work on most computers those requirements were:

- AMD dual core graphics processor running at 3.0GHz
- 4GB DDR3
- Sata III 600GB internal HDD
- CD-Rom Drive
- motherboard onboard graphics

the operating system for the computer doesn't matter

5.2.2 Prerequisite Installation

the application doesn't require many programs to be instatlled to the computer apart from itself and the programs listed below

Installing Python

Installing PyQt

Etc.

5.2.3 System Installation

5.2.4 Running the System

5.3 Tutorial

5.3.1 Introduction

5.3.2 Assumptions

5.3.3 Tutorial Questions

Question 1

Question 2

5.3.4 Saving

5.3.5 Limitations

5.4 Error Recovery

5.4.1 Error 1

5.4.2 Error 2

5.5 System Recovery

5.5.1 Backing-up Data

5.5.2 Restoring Data

Chapter 6

Evaluation

6.1 Customer Requirements

6.1.1 Objective Evaluation

6.2 Effectiveness

6.2.1 Objective Evaluation

6.3 Learnability

6.4 Usability

6.5 Maintainability

6.6 Suggestions for Improvement

6.7 End User Evidence

6.7.1 Questionnaires

6.7.2 Graphs

6.7.3 Written Statements