Contents

Analysis

1. Introduction

1.1. Client Identification

My Client is Niki Cox; Niki uses her computer for emails, word processing and the software package "MidexPro" (a medical billing system). She has two touch screen desktop computers on her desk which both run 64-bit windows 7.

She runs her own secretarial business called "Cambridge Medical Practice" and currently employs four other people. The business manages all of the paper work and appointments for private consultants and is run from a small office attached to Niki's house. There are currently 9 consultants and 6 anaesthetists using her services.

1.2. Define the current system

Any medical system uses set operation codes to summarise a particular procedure carried out during an operation. If an operation contains multiple procedures then up to 3 codes can be assigned to a patient to summarise the procedures they have had during their operation.

The current system is manual and involves visiting multiple webpages to retrieve values before manually calculating two final prices. Niki has to do this for every patient as different codes are used for different operational procedures.

After retrieving a list of patients relating to the previous week from her filing cabinet, she has to get two prices related to the operation codes, one for the surgeon and the other for the anaesthetist. To do this she finds out which insurance company the patient is with and their operation codes from her MidexPro system. She then goes to the corresponding insurance company website, where she manually has to find and record the two prices for all of the codes being used by the patients from the previous week.

Secondly, if there is more than one code, she has to check the compatibility of the codes. This is done using the CCSD[1] website. CCSD are a group containing representatives from the major private medical insurance companies and have agreed a set of common procedures for the private medical sector. Niki uses the website to look up each of the codes being used to see if they collide with any of the others. For example if you were to look up the operation code T7910 you would see that it collides with W9192 along with many others.

Finally she must use the calculation methods for multiple codes, provided by the insurance companies, to work out the price to charge. These methods are different for some of the insurance companies and the calculation also depends on the code combination selected above. The calculation methods for each of the insurance companies are shown in the table below.

[1] CCSD – The Clinical Coding and Schedule Development Group (http://www.ccsd.org.uk/)

Insurance Company	Number of Codes	Calculation method
BUPA*	1	Charge 100% of the code.
	2	Charge 125% of the most expensive code.
	3	Charge 140% of the most expensive code.
WPA	1, 2 or 3	See BUPA*
PPP (only allow 2)	1	Charge 100% of the code.
	2	Charge 100% of the most expensive code, plus 50% of the second most expensive code.
AVIVA	1	Charge 100% of the code.
	2	Charge 100% of the most expensive code, plus 50% of the second most expensive code.
	3	Charge 100% of the most expensive code, plus 50% of the second most expensive code, plus 25% of the third most expensive code.
Pru Health	1, 2 or 3	See BUPA*
Cigna	1, 2 or 3	See BUPA*
Simply Health	1, 2 or 3	See BUPA*

1.3. Define the problems

One of the main problems with the current system is the time consumption. It takes Niki around 5 minutes to calculate the two prices for one patient. This process is used between 30 and 40 times a week, taking up approximately 3 hours.

The system is also has a large room for error as the calculation methods are reasonably complex. This is a problem as if a price is calculated incorrectly the business could lose out financially or get in trouble with the insurance companies for trying to charge codes which are not compatible.

As practice manager Niki feels that because of this process is error prone, it is important that she carries out the task of calculating the prices herself. Whereas she would prefer having a simpler system which everyone in the office can use, freeing up her time for other important tasks.

Finally, all of the data comes from different places and has to be compiled by the user which can be confusing. For example the patient's information is within MidexPro but the list of patients is in the filing cabinet, the prices for the codes are on the insurances companies' website and the compatibility is located on yet another site.

1.4. Section appendix

Client Interview

Client Name: Niki Cox

Business Name: Cambridge Medical Practice

What do you and your business do?
Provide a secretarial management service to consultants - Surgeons and Anaesthetists

What does the current system do? Is it manual or computerised?
Patient data is retrieved using the computer but the calculation is manual

What are the problems with the current system?
- Too Slow
- Many Processes
- Large Error Scope
- Has to be done by Niki
- Potential loss of revenue
- Manual Calculation

How does the current system work?
Physical list of patients, details manually retrieve from Midex Pro database. Code prices and combinations are retrieved from the web for manual calculation.

What would you like the new system to do?
Calculate code combinations individually or our stored lists of patients. Lists generated by week, consultant, date or name. Create invoices from these lists.

Do you have any other preferences regarding the new system?
eg. Completed before a certain date?
Would be nice to include company logo.
No time preferences - sooner the better.

How many different insurance companies are there, their names and web addresses?

BUPA	bupa.co.uk
WPA	wpa.org.uk
PPP	axapppphealthcare.co.uk
AVIVA	aviva.co.uk/Health
PruHealth	pruhealth.co.uk
Simply Health	simplyhealth.co.uk
Cigna	cigna.co.uk

Do the operation codes all have the same format? How many codes do you require the system to have?

All have A9999 format
Approximatly 300

Averagely how many codes collide with each code?

Each code normally has between 5 and 15 incompatible codes.

What are the calculation methods for the different Insurance companies?

BUPA, WPA, Pru, Cigna, Simply all use:
100% of highest code for 1, 125% for 2, 140% for 3
PPP + AVIVA – 100% of the highest code for 1, plus 50% of the second highest for 2, plus 25% of the lowest for 3. However PPP only allow 2 codes.

How often do you use your computer and what for?

12-15 hours a day (Mon-Fri)
Emails, Word Processing, MidexPro

Do you or any of the staff have any computer related qualifications?

No.

Who would you like to have access to the new system?

All staff members in the office and the technical support

How many computers does the office currently use? 5

Are there any other hardware devices in the office?

Printer, Server, Rater, Scanner, Fax Machine

Signed: Date: 05.07.12

After interviewing my client she was unable to provide me with details regarding the hardware used in the office. So I had a second interview with the technical support manager Nick Cox to get the necessary hardware information.

Technical Support Interview

Name: Nick Cox

What role do you provide to Cambridge Medical Practice?
Technical Support Manager

Computer specifications:

Reference ID	Manufacturer	Model	Processor	RAM	Operating System
1	Dell	Optiplex 380	Intel Core Duo	2GB	Windows 7 Progessional 32-bit
2	Dell	Optiplex 380	Intel Core Duo	4GB	Windows 7 Progessional 32-bit
3	Dell	Vastro 360	Intel i3 3.3GHz	4GB	Windows 7 Progessional 64-bit
4	Dell	Vastro 360	Intel i5 2.5GHz	4GB	Windows 7 Progessional 64-bit
5	Sony	PCV-A1112M	Intel i5 2.93GHz	4GB	Windows 7 Progessional 64-bit

Can you give me more details about the Server, Printer and Router?

Server - DELL 110, Intel Xeon, 2.4GHz, 4GB, RAID5, windows small business server 2008, 64-bit
Printer - HP 2055DN Network printer
Router - Virgin Superhub, NetGear

Signed: NCox. Date: 29/9/12

2. Investigation

2.1. The current system

2.1.1. Data sources and destinations

The 4 main data sources in this system are:

1) The patient information (including their operation and associated operation codes) which is retrieved from the filing cabinet.

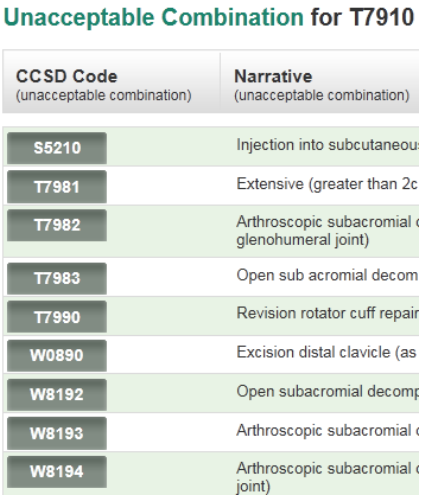
2) The prices for each code which are identified from the insurance company websites and recorded onto paper ready for calculations.

3) The combinations of codes which are identified from the CCSD website and recorded on paper for use in the calculation.

4) The amended patient information which now contains the surgeon and anaesthetist prices and is ready for invoicing.

The table below shows these data transitions more clearly.

Source	Data	Example Data	Destination
Filing Cabinet	Patient Information	First Name Last Name Patient ID Insurance Company Operation Codes	User
Insurance company website.	Prices	Surgeon: £870 Anaesthetist: £425	User
CCSD website	Combinations	Code T7910 collides with code W8194	User
User	Amended Patient Information	First Name Last Name Patient ID Insurance Company Operation Codes Surgeon Price Anaesthetist Price	MidexPro



Unacceptable Combination for T7910

CCSD Code (unacceptable combination)	Narrative (unacceptable combination)
S5210	Injection into subcutaneou
T7981	Extensive (greater than 2c
T7982	Arthroscopic subacromial (glenohumeral joint)
T7983	Open sub acromial decom
T7990	Revision rotator cuff repair
W0890	Excision distal clavicle (as
W8192	Open subacromial decomp
W8193	Arthroscopic subacromial (
W8194	Arthroscopic subacromial (joint)

Example of CCSD webpage, showing unacceptable combinations

Example of insurance company (WPA) webpage, showing the procedure (surgeon) and the anaesthetist prices.

Code	Description	Procedure	Anaesthetic
W8194	Arthroscopic sub-acromial decompression and excision of distal clavicle (including arthroscopic procedures in glenohumeral joint)	£870.00	£425.00

2.1.2. Algorithms

The algorithms in this section concentrate mainly on the calculation methods as this is the most important function within the system. This algorithm shows the most common type of calculation in the current system (using BUPA as the example insurance company). This demonstrates calculating the final prices after the individual codes and their prices have been input, and would be carried out twice for both the surgeon and anaesthetist.

Algorithm 1 – BUPA Calculation method

```
CodeList ← RETRIEVE codeList
PriceList ← RETRIEVE priceList
MaxPrice ← 0
FOR Count IN RANGE  LENGTH(CodeList)
        IF PriceList[Count] > MaxPrice
        MaxPrice ← PriceList[Count]
END FOR
IF LENGTH CodeList = 3
        FinalPrice ← (MaxPrice / 100) * 140
ELIF LENGTH CodeList = 2
        FinalPrice ← (MaxPrice/ 100) * 125
ELSE
        FinalPrice ← MaxPrice
```

Algorithm 2 – AVIVA Calculation method

A similar algorithm demonstrates the calculation method for AVIVA.

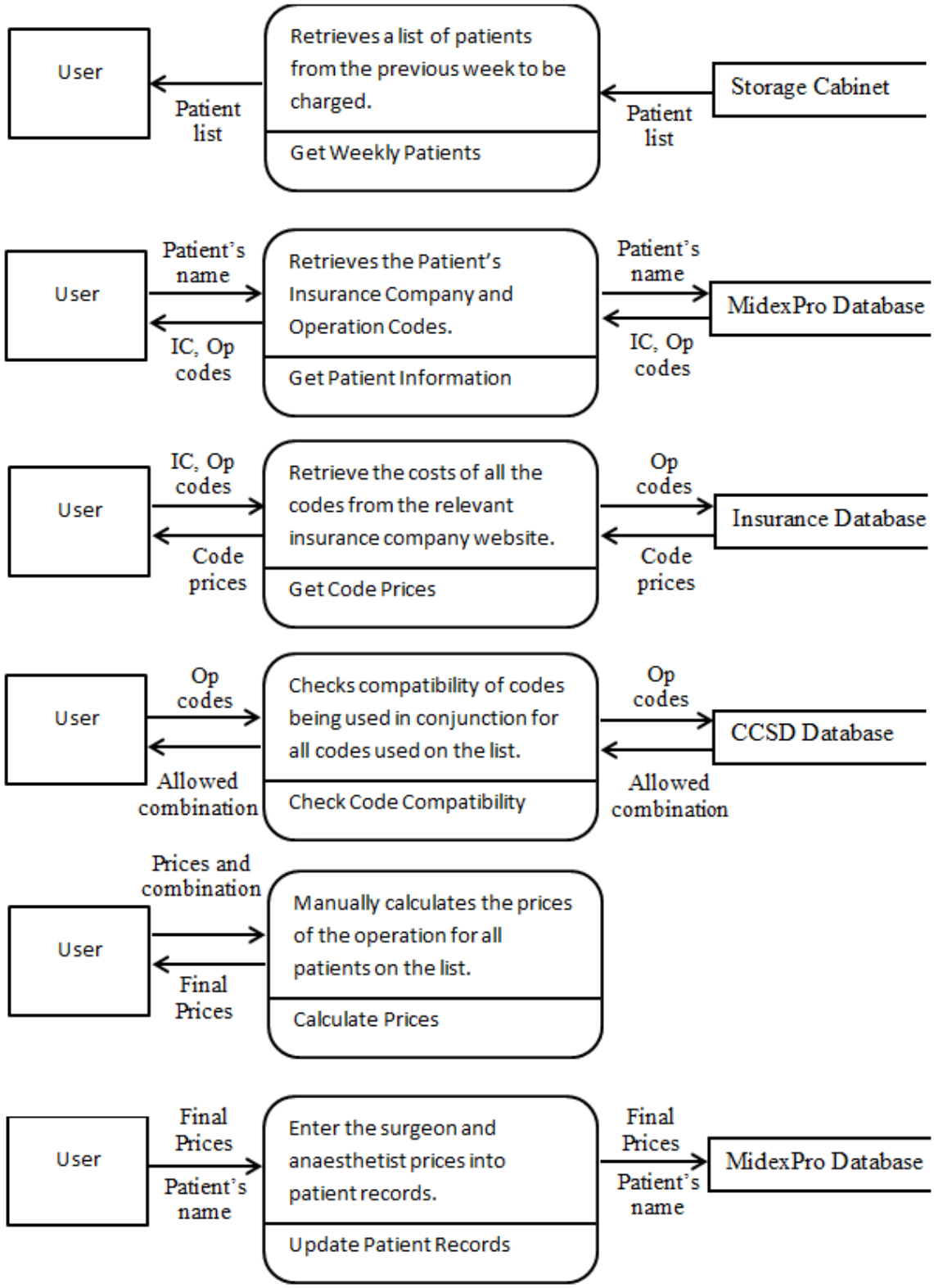
```
CodeList ← RETRIEVE codeList
PriceList ← RETRIEVE priceList
MaxPrice ← 0
MidPrice ← 0
LowPrice ← 0
FOR Count IN RANGE  LENGTH(CodeList)
        IF PriceList [Count] > MaxPrice
                MaxPrice ← PriceList [Count]
        ELIF PriceList [Count] < MaxPrice AND PriceList [Count] > LowPrice
                MidPrice ← PriceList[Count]
        ELSE
                LowPrice ← PriceList [Count]
END FOR
IF LENGTH CodeList = 1
        FinalPrice ← MaxPrice
ELIF LENGTH CodeList = 2
        FinalPrice ← MaxPrice + (MidPrice * 0.5)
ELSE
        FinalPrice ← MaxPrice + (MidPrice * 0.5) + (LowPrice * 0.25)
```

Algorithm 3 – All Calculation methods

If we combine these to algorithms by introducing an Insurance Company variable we can see how the system will decide which calculation method to use.

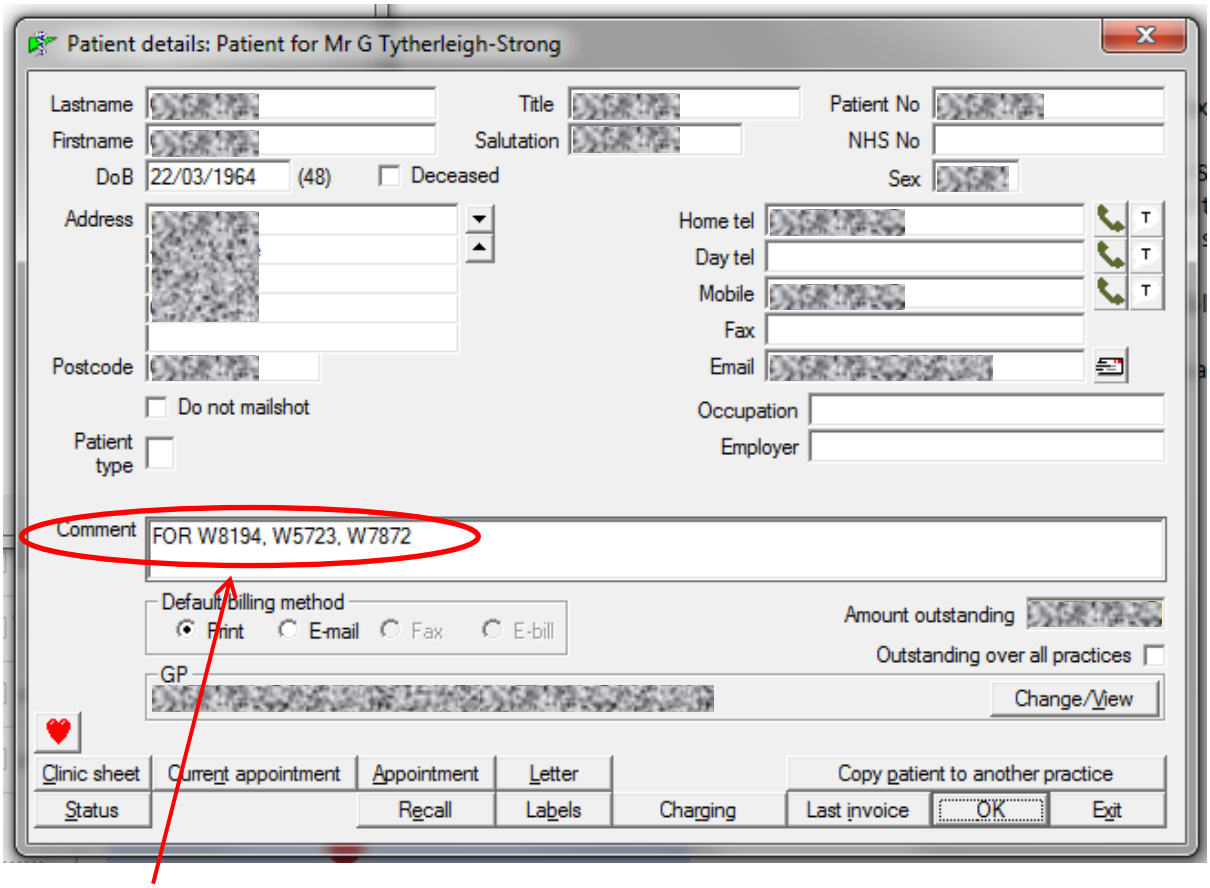
```
CodeList ← RETRIEVE codeList
PriceList ← RETRIEVE priceList
InsuranceCompany ← RETRIEVE InsuranceCompany
IF InsuranceCompany IN ["BUPA","WPA", "Pru Health", "Cigna", "Simply Health"]
        MaxPrice ← 0
        FOR Count IN RANGE  LENGTH(CodeList)
                IF PriceList [Count] > MaxPrice
                MaxPrice ← PriceList [Count]
        END FOR
        IF LENGTH CodeList = 3
                FinalPrice ← (MaxPrice / 100) * 140
        ELIF LENGTH CodeList = 2
                FinalPrice ← (MaxPrice/ 100) * 125
        ELSE
                FinalPrice ← PriceList[1]
ELIF InsuranceCompnay = "AVIA"
        MaxPrice ← 0
        MidPrice ← 0
        LowPrice ← 0
        FOR Count IN RANGE  LENGTH(CodeList)
                IF PriceList [Count] > MaxPrice
                        MaxPrice ← PriceList [Count]
                ELIF PriceList [Count] < MaxPrice AND PriceList [Count] > LowPrice
                        MidPrice ← PriceList [Count]
                ELSE
                        LowPrice ← PriceList [Count]
        END FOR
        IF LENGTH CodeList = 1
                FinalPrice ← MaxPrice
        ELIF LENGTH CodeList = 2
                FinalPrice ← MaxPrice + (MidPrice * 0.5)
        ELSE
                FinalPrice ← MaxPrice + (MidPrice * 0.5) + (LowPrice * 0.25)
ELSE
        MaxPrice ← 0
        LowPrice ← 0
        FOR Count IN RANGE  LENGTH(CodeList)
                IF PriceList [Count] > MaxPrice
                        MaxPrice ← PriceList [Count]
                ELSE
                        LowPrice ← PriceList [Count]
        END FOR
        IF LENGTH CodeList = 1
                FinalPrice ← MaxPrice
        ELIF LENGTH CodeList = 2
                FinalPrice ← MaxPrice + (LowPrice * 0.5)
```

2.1.3 Data flow diagram

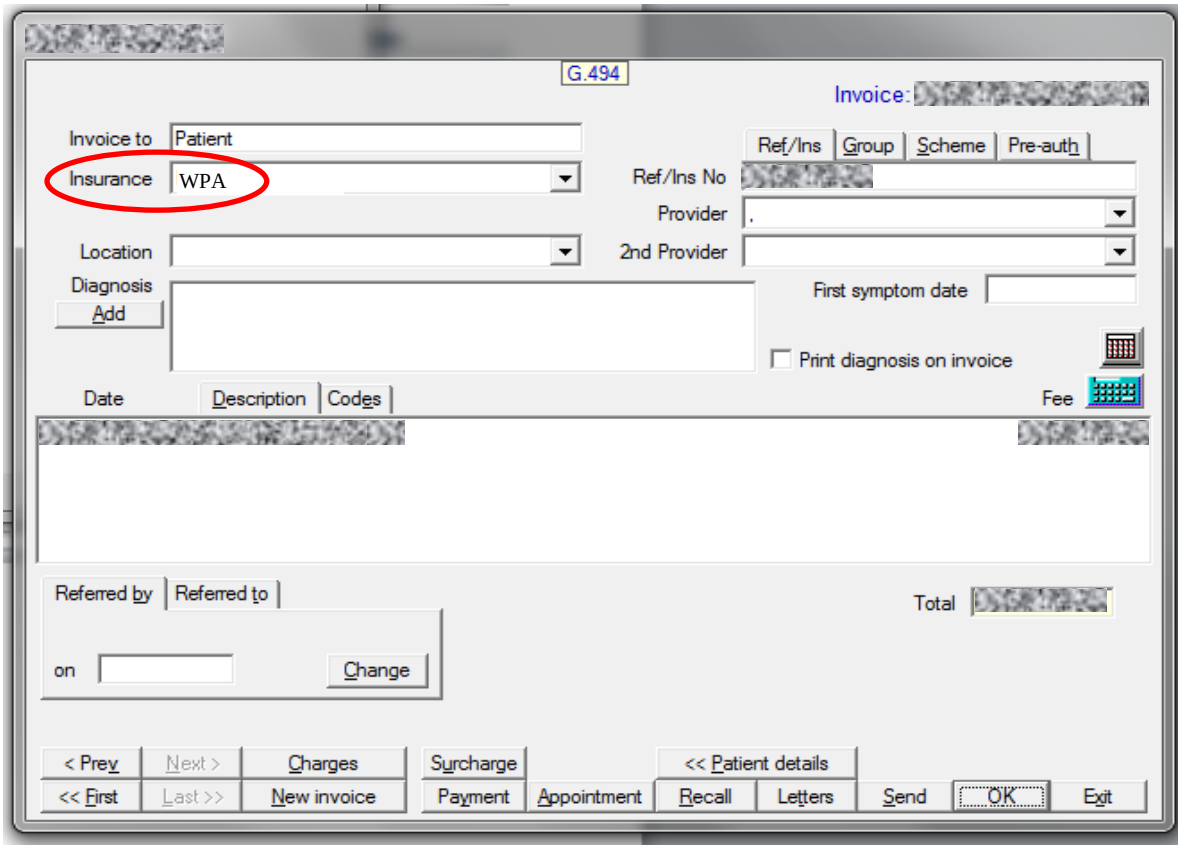


2.1.4. Input Forms, Output Forms, Report Formats

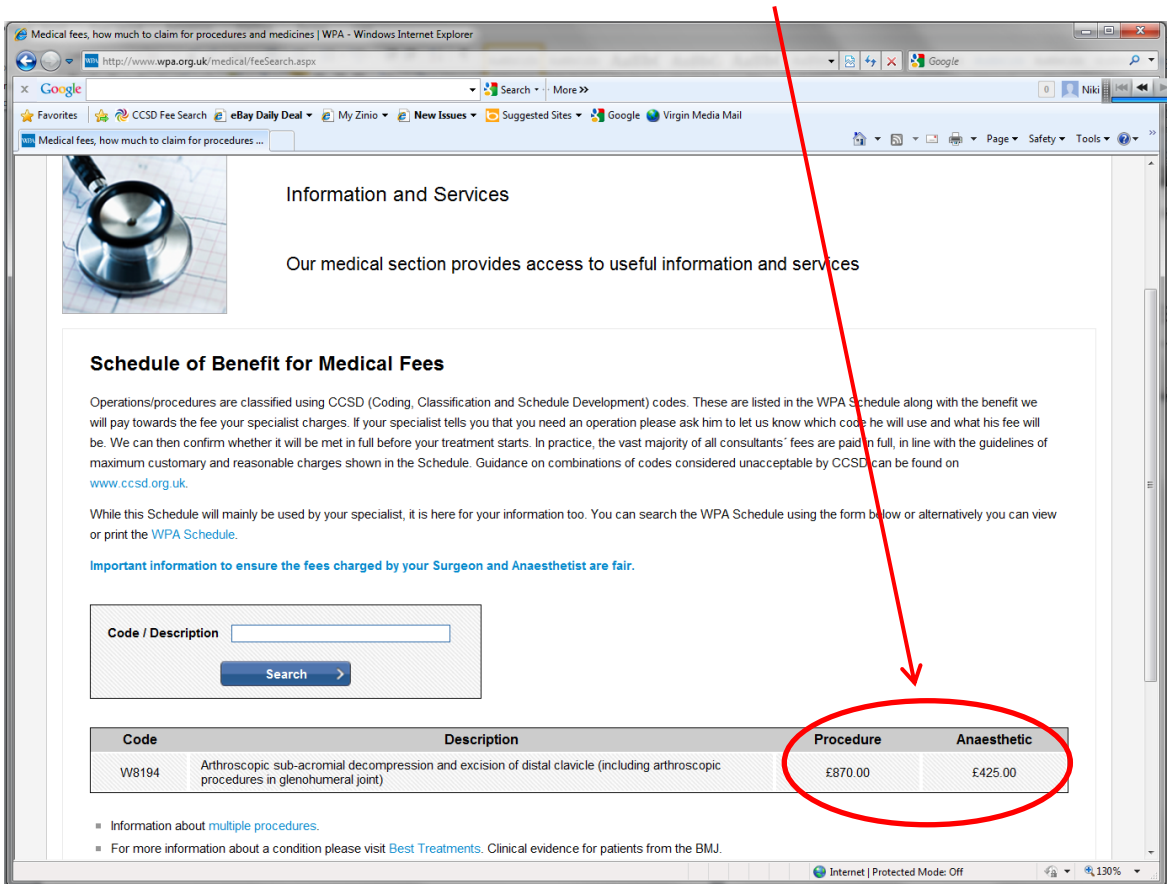
The first form is from the MidexPro software which Niki uses. It shows the window she has to visit in order to retrieve a patient's operation codes. The second window shows the output form she receives from the first window in order to find which insurance company the patient is with.



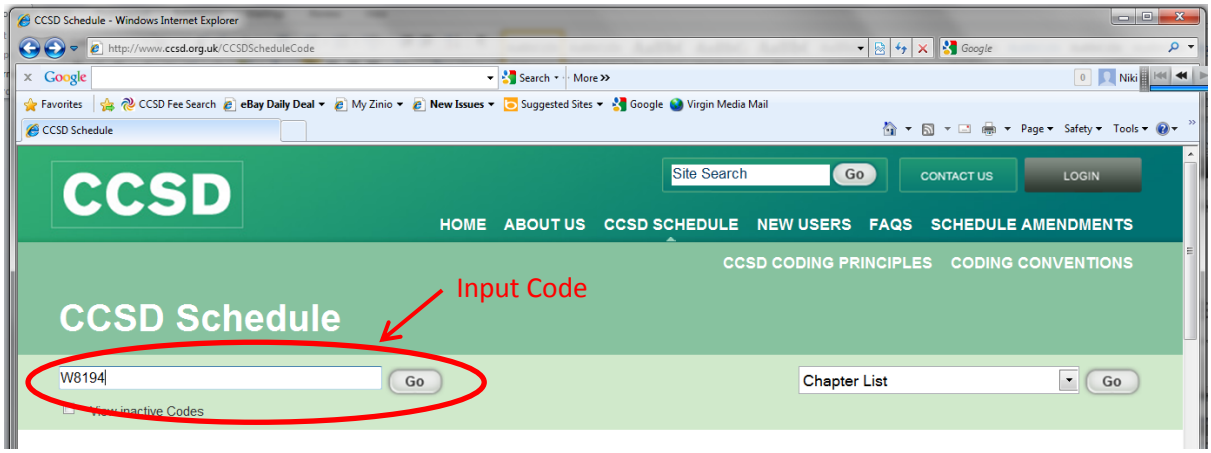
As you can see from the first window there is no official data store of the operation codes the patients have had. Niki has to use the comment box to store this information.



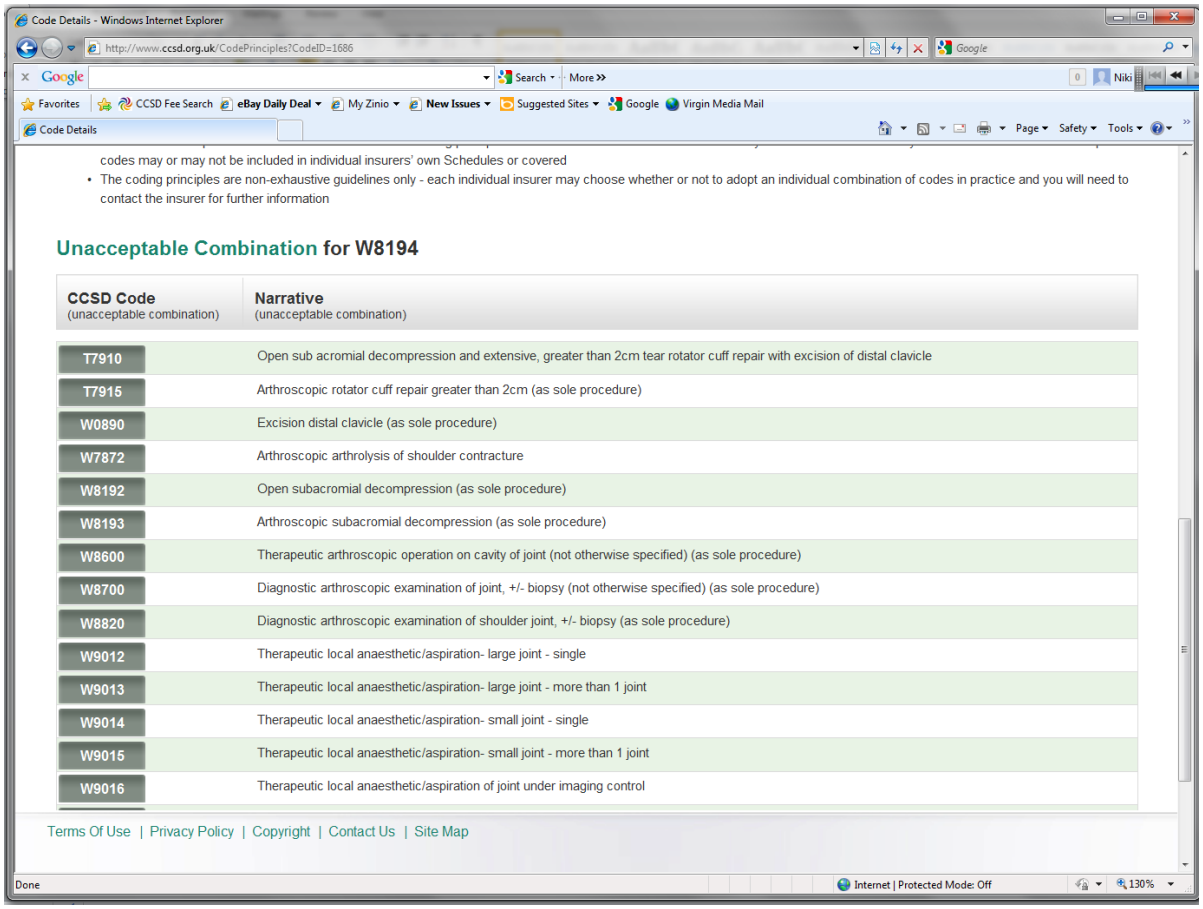
The next window shows an example of inputting a code into an insurance company website (WPA in this case), in order to retrieve the Procedure and Anaesthetists prices.



The 2 screens below show the report produced after inputting a code into the CCSD website in order to check it's compatibility with the other codes.



This then leads to the output form shown below which lists the incompatible codes related to the searched code from the previous input.



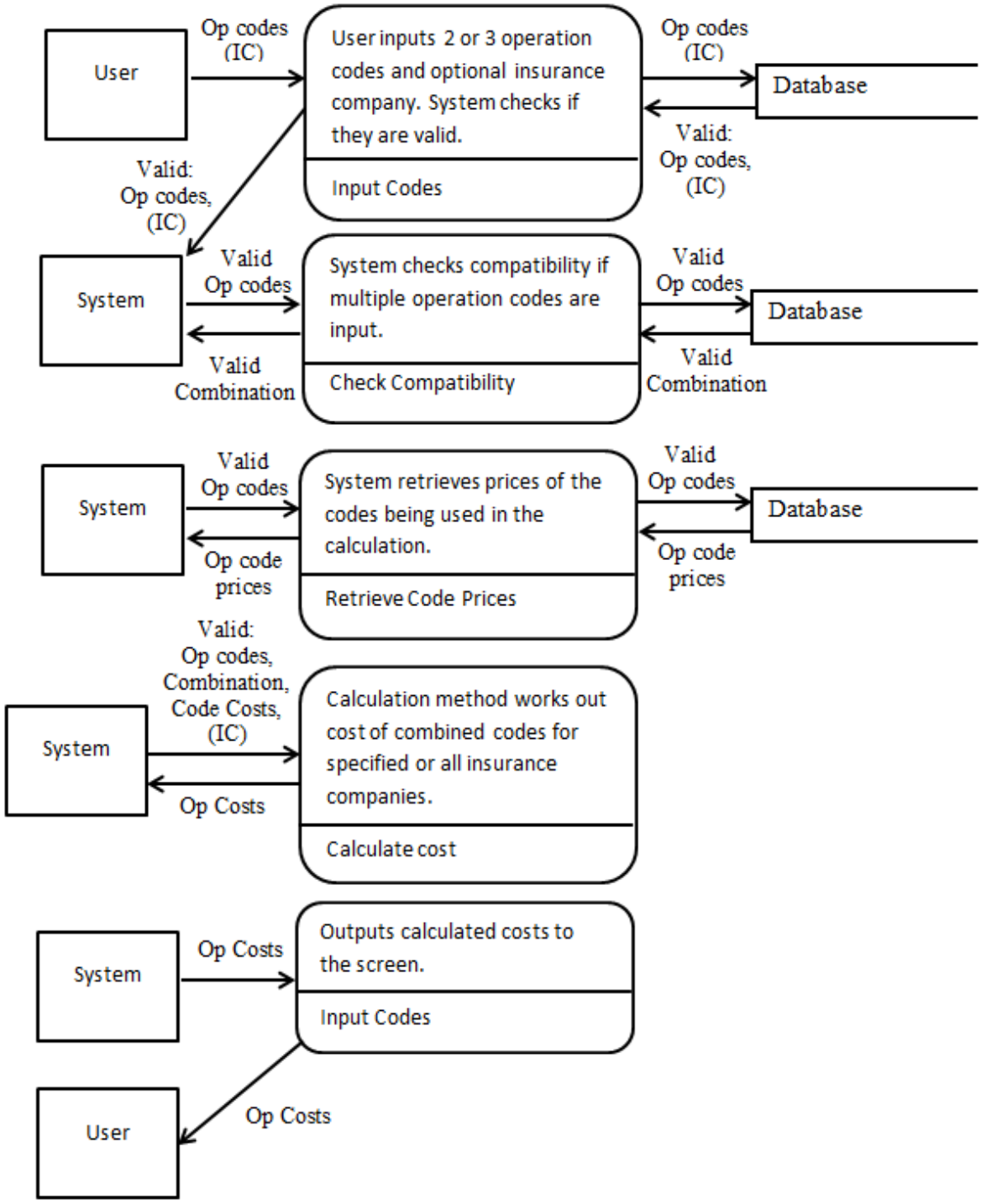
2.2. The proposed system

2.2.1 Data sources and destinations

Source	Data	Data Type	Destination
List of Patients	First Name	List - String	User
List of Patients	Last Name	List - String	User
User	First Name	String	Calculation Method
User	Last Name	String	Calculation Method
Price Database	Code Price	Integer	Calculation Method
Combination Database	Disallowed combinations	String	Calculation Method
Calculation Method	Surgeon Price	Integer	User
Calculation Method	Anaesthetist Price	Integer	User
Calculation Method	Total Price	Integer	User
Calculation Method	Surgeon Price	Integer	Database
Calculation Method	Anaesthetist Price	Integer	Database
Calculation Method	Total Price	Integer	Database
User	Surgeon Price	Integer	MidexPro Patient Records
User	Anaesthetist Price	Integer	MidexPro Patient Records

2.2.2 Data flow diagram

This first data flow diagram shows the main feature if the system, the 'code calculator'. The diagram shows the data flow when the calculator is used individually from any stored patient records.

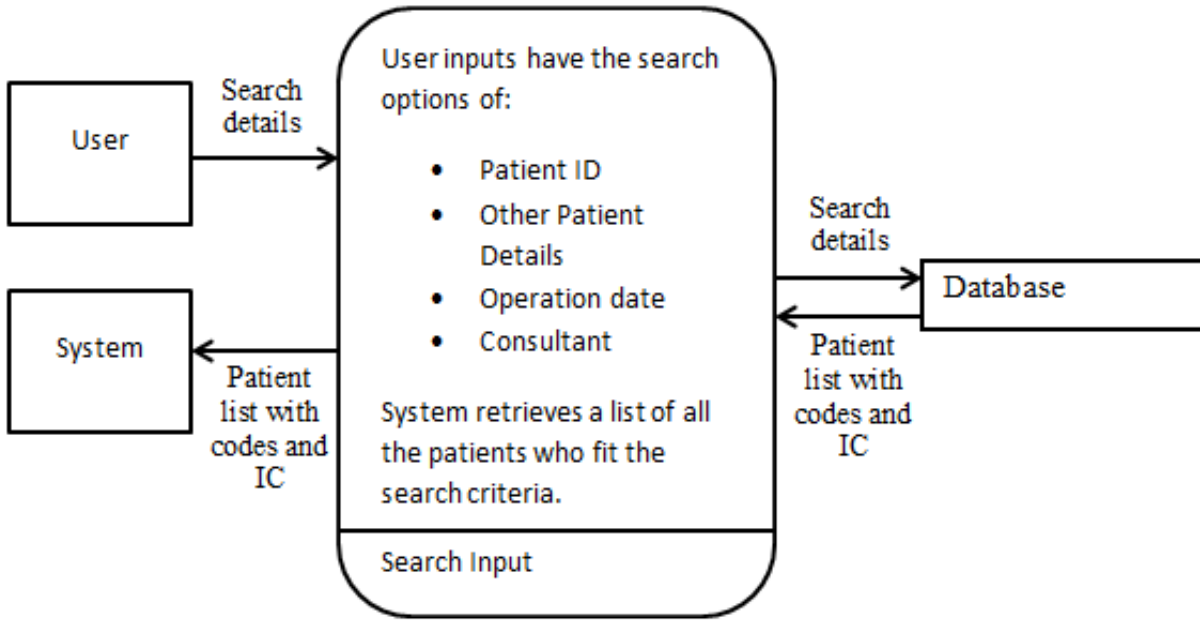


**(IC) = Optional Insurance Company input/output, Op Codes = Operation Codes, Op Costs = Operation Costs (Surgeon and Anaesthetist fee)*

The system also requires codes to be calculated for existing patients stored within the database. These patients can be called to the calculating method based on the following search attributes:

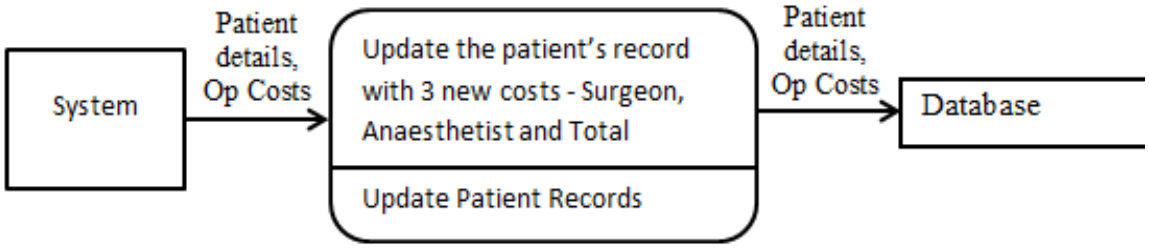
- Patient ID
- Other Patient Details (eg. First name, Last name, Date of Birth)
- Operation Date
- Consultant

Therefore the first process in my data flow diagram could be changed to the following.

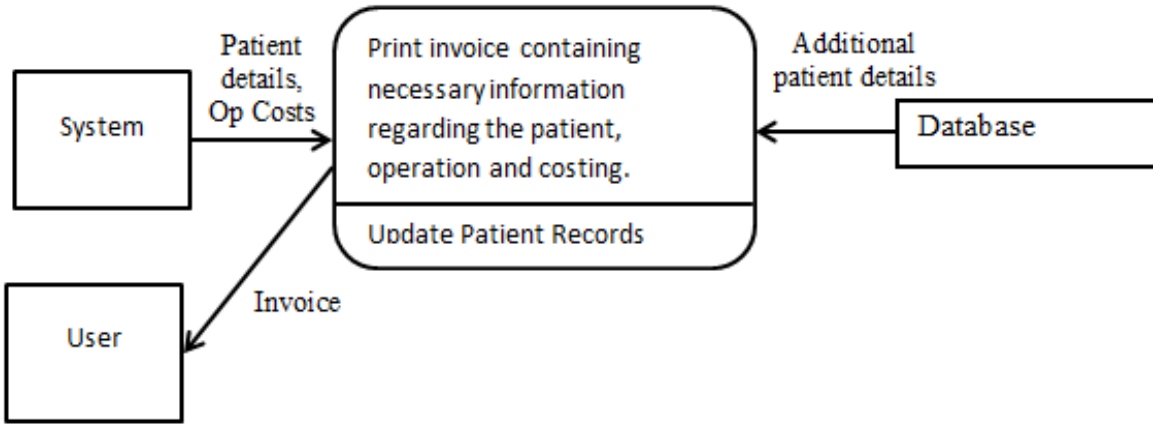


**IC = Insurance Company*

Using the 'code calculator' the user may wish to update the patient's records or print off invoices for the patients. This means that the last process in the data flow diagram could be changed to either of the following.



**Op Costs = Operation Costs*



**Op Costs = Operation Costs, Additional patient details = details which were not required for calculation but are required to produce an invoice, such as date of birth.*

As you can see from the proposed data flow diagrams the manual processes have been significantly reduced which in turn will reduce the scope for errors which Niki mentioned was one of the main problem areas with the current system.

2.2.3 Data dictionary

Name	Data Type	Length	Validation	Example
Patient ID	Integer	1+	-	1
Title	String	4	Look Up	Mr
First Name	String	25	Length/Presence	Fred
Last Name	String	25	Length/Presence	Barnes
Address Line 1	String	30	Length /Presence	45 Example Rd
Town	String	20	Length /Presence	Cambridge
Post Code	String	8	Format/Presence	AA99 9AA
Date Of Birth	Date	10	Format/Presence	1994/12/23
Gender	String	6	Look Up	Female
Telephone Number	String	12	Length/Presence	01223 324283
Email	String	50	Format/Presence	abc@edf.com
Consultant ID	Integer	1+	-	1
Consultant Title	String	4	Look Up	Mr
Consultant First Name	String	25	Length/Presence	Fred
Consultant Last Name	String	25	Length/Presence	Barnes
Consultant Email	String	50	Format/Presence	abc@edf.com
Code ID	Integer	1+	-	1
Code Name	String	5	Format/Look Up	A9999
Patient ID	Integer	1+	-	1
Operation ID	Integer	1+	-	1
Date Of Operation	Date	10	Format/Presence	1994/12/23
Price ID	Integer	1+	-	1
Surgeon Price	Integer	4	Range	576
Anaesthetist Price	Integer	4	Range	843
Code Name	String	5	Format	A9999
Insurance ID	Integer	1+	-	1
Company Name	String	13	Look Up	Simply Health
Web Address	String	30	Format	www.ccsd.co.uk
Username	String	20	Look Up	User102
Password	String	20	Length	Password1
User ID	Integer	1+	-	1

**IDs are automatically given therefore their length is unknown and validation unrequired as it is not a user input.*

2.2.4 Volumetric

The table below shows the number of bytes required by each entity to store the data which the system needs. I used 12,000 as the patient, operation and price quantity required as this will be the approximately the amount of patients passed through the system in a year.

Entity	Total Attributes	Quantity Required	Total Characters (Bytes)
Patient	12	500*	95,500
Insurance Company	3	7	266
Operation	6	500*	11,000
Consultant	5	15	1,605
Price	5	500*	8,500
Code	2	300*	2,400
Invalid Combinations	2	3000*	30,000
User	6	5	715
Total			149986

**IDs were assumed to be 3 characters long*
"200" = approximate value*

3. Objectives

3.1. General Objectives

- Data is to be protected
- Clear layout with appropriate buttons
- Edit information relating to patients
- Edit information relating to the operations codes
- Edit information relating to staff using the system
- Calculate prices individually and for a given week and consultant
- Print report sheets containing invoice information

3.2. Specific Objectives

- Log In system to allow different access levels for data protection
- Confidential patient and staff data must be encrypted for data protection
- Automatic password update emails sent to staff members to increase security
- Add new patients to the database
- Edit existing patient information
- Allow system administrators to delete patient information
- Add/Delete operation codes in the database
- Edit pricing of codes within the database
- Calculate a surgeon, anaesthetist, and total price for any given or all insurance companies in a one off situation
- Calculate a surgeon, anaesthetist, and total prices for a specified week and consultant
- Print a list of patient information, including the pricing, for filing.
- Print invoices for individual or list of patients.

3.3. Core Objectives

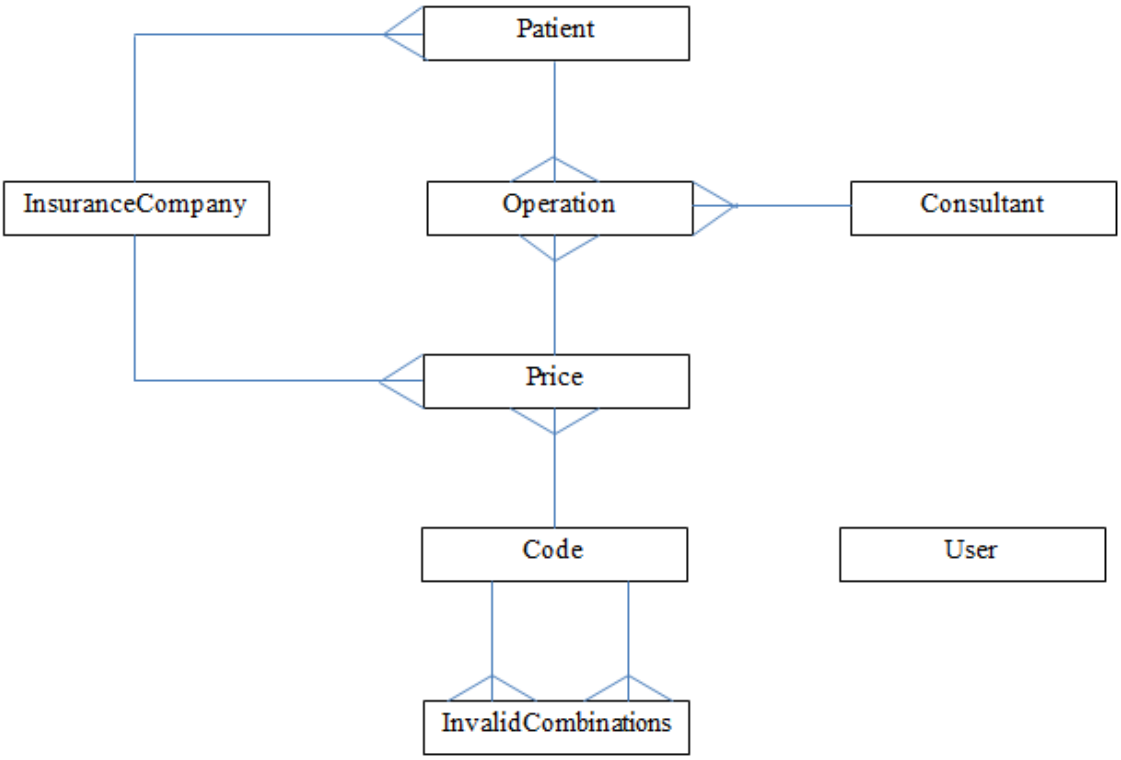
- Log In system to allowed different access levels for data protection
- Confidential patient data must be encrypted for data protection
- Add new patients to the database
- Edit existing patient information
- Add/Delete operation codes in the database
- Edit pricing of codes within the database
- Calculate a surgeon, anaesthetist, and total price for any given or all insurance companies in a one off situation
- Print invoices for individual or list of patients.

3.4. Other Objectives

- Calculate a surgeon, anaesthetist, and total prices for a specified week and consultant
- Allow system administrators to delete patient information
- Print a list of patient information, including the pricing, for filing.

4. E-R Diagrams and Descriptions

4.1. E-R Diagrams



4.2. Entity Descriptions

Patient (PatientID, Title, FirstName, LastName, AddressLine1, AddressLine2, Town, PostCode, DateOfBirth, Gender, *InsuranceID*, TelephoneNo, Email)

Operation (OperationID, *PriceID, PatientID, ConsultantID*, DateOfOperation)

Consultant (ConsultantID, ConsultantTitle, ConsultantFirstName, ConsultantLastName, ConsultantEmail)

Price (PriceID, *CodeID, InsuranceID*, SurgeonPrice, AnaesthetistPrice)

Code (CodeID, CodeName)

InsuranceCompany (InsuranceID, CompanyName, WebAddress)

InvalidCombinations (SourceCode, DisallowedCode)

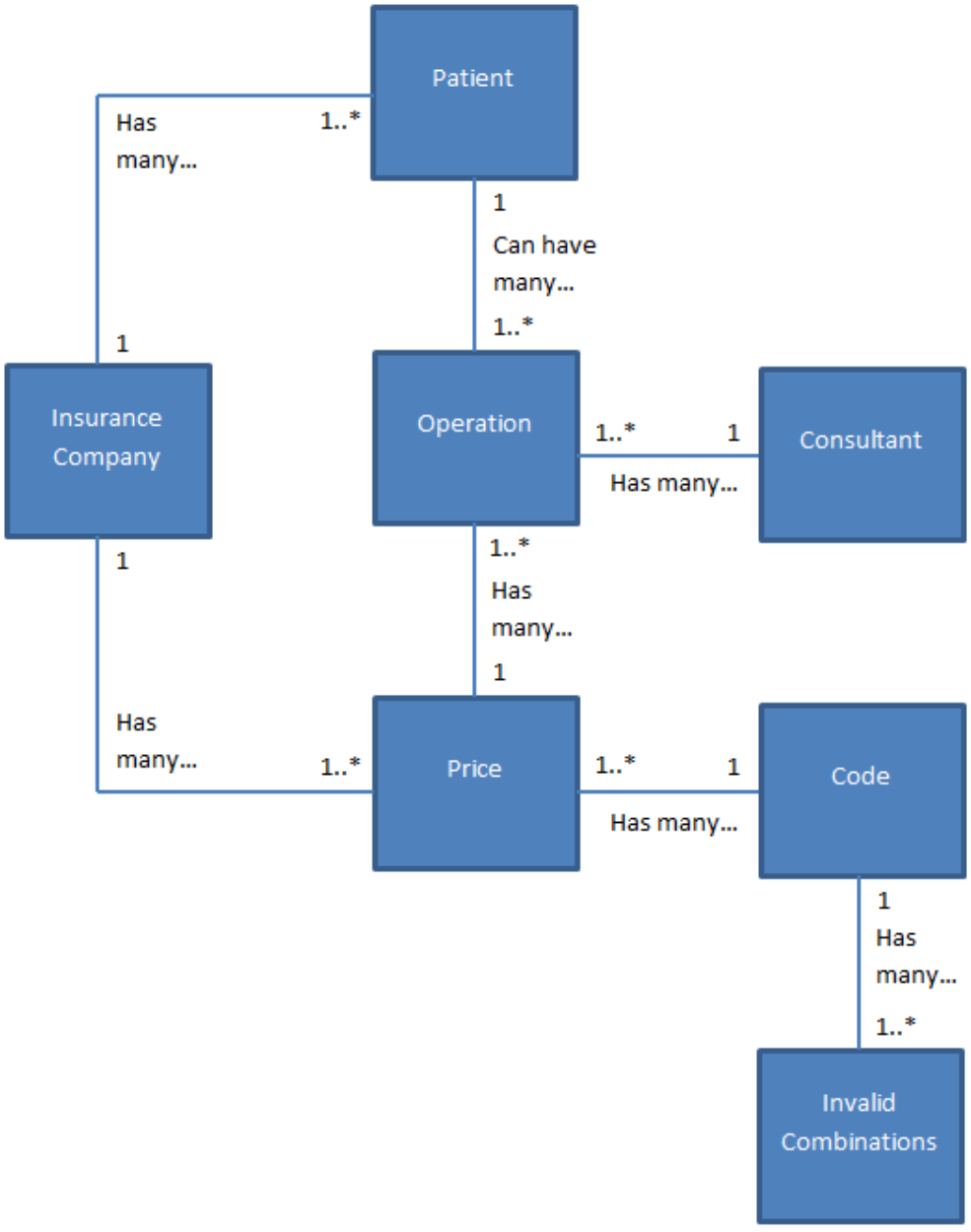
User (StaffID, FirstName, LastName, Email, UserName, Password)

5. Object Analysis

5.1. Object Listing

- Patient
- Insurance Company
- Operation
- Consultant
- Price
- Code
- Invalid Combinations

5.2. Relationship diagrams



5.3. Class definitions

Key:

Object Label
Attributes
Behaviours

Patient
Title FirstName LastName AddressLine1 AddressLine2 Town PostCode DateOfBirth Gender TelephoneNumber Email
getPatientReport addPatient editPatient deletePatient addOperation

Insurance Company
CompanyName WebAddress
addInsuranceCompany edidInsuranceCompany deleteInsuranceCompany

Operation
DateOfOperation
addOperation editOperation

Consultant
Title FirstName LastName Email
addConsultant editConsultant deleteConsultant

Price
SurgeonPrice AnaesthetistPrice
getIncuranceCompany getOperationCodes calculateSurgeonPrice calculateAnaesthetistPrice

Code
CodeName
addCode deleteCode

Invalid Combinations
SourceCode DisallowedCode
addDisallowedCombination deleteDisallowedCombination

6. Other Abstractions

6.1. Graphs

No graphs are required.

7. Constraints

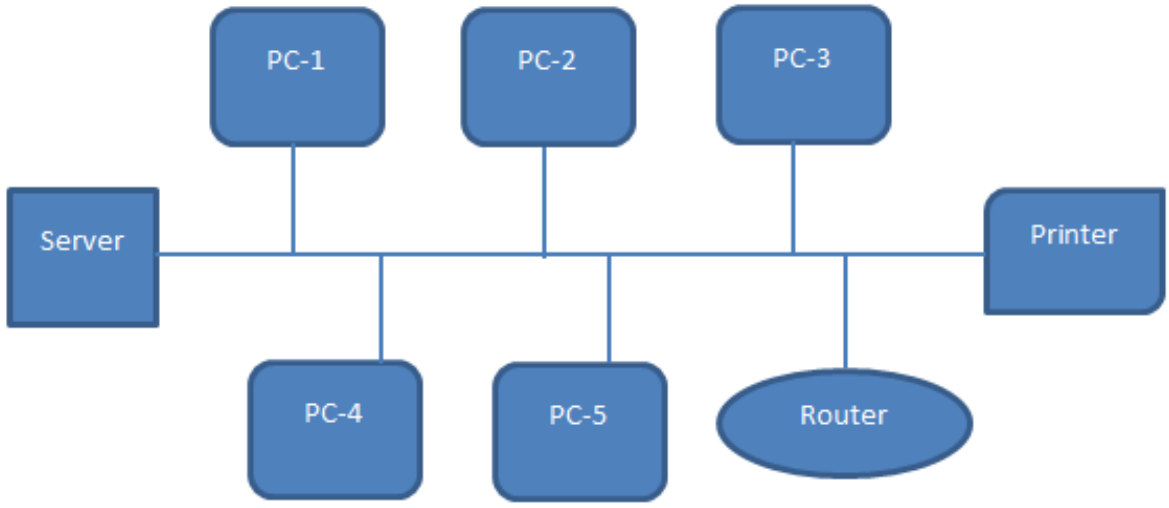
7.1. Hardware

There are currently 5 computers in Niki's office which all run a strand of windows 7 professional. All of the computers have internet connection. Details of these computers are given below.

Reference ID	PC-1	PC-2	PC-3	PC-4	PC-5
Manufacturer	DELL	DELL	DELL	DELL	SONY
Model	Optiplex 380	Optiplex 380	Vostro 360	Vostro 360	PCV-A1112M
Processor	Intel Core Duo	Intel Core Duo	Intel i3 running at 3.3GHz	Intel i5 running at 2.5GHz	Intel i5 running at 2.93GHz
RAM	2GB	4GB	4GB	4GB	4GB
Operating System	Windows 7 Professional, 32-bit	Windows 7 Professional, 32-bit	Windows 7 Professional, 64-bit	Windows 7 Professional, 64-bit	Windows 7 Professional, 64-bit

There is also a DELL server 110 in the office which has a Intel Xeon processor running at 2.4 GHz. The server has 4GB of RAM and runs the 64-bit Windows Small Business Server 2008 operating system. The 2 disks within the server are configured as a RAID 5 array.
There is also a HP 2055DN network printer.
On the network there is also a Virgin Super hub router which is manufactured by NetGear.

The diagram below illustrates the office LAN configuration:-



None of this hardware should be a constraint as my system is reasonably small and simplistic therefore it should run fine on all of the computers within the office. However in order to use the printer on the network I will need to learn some network and printer protocols, this could be considered a constraint.

7.2. Software

Niki does not require the program to be linked to MidexPro in anyway. However she would like the program to be available for everyone in the office to use. This is a constraint because it requires the database behind the system to be linked to multiple computers. Therefore I will need to ensure the SQLite Python library can be used with a database installed on a LAN server.

7.3. Time

My client does not require the system by any particular time, which leaves the only time restriction as the deadline of this project which is around the middle of February 2013.

7.4. User knowledge

None of the staff have any computer related qualifications other than being privately trained in the use of MidexPro. This means my system should be relatively simple to use to prevent confusion. Using a similar style and layout to MidexPro with clearly organised and labelled buttons would make the program easy to use.

There should be a clear set of instructions provided with the program to assist the users.

7.5. Access restrictions

As the database behind the system will have private information regarding patients there will need to be some access restrictions for data protection. This will be enforced using a log in system for anyone trying to access the program. The user accounts will need to be monitored, and because of this Niki's IT support will have a higher administrator access level allowing them to create, delete and view all of the staff accounts and their system activity.

8. Limitations

8.1. Areas which will not be included in computerisation

There are multiple areas which I feel Niki would like to be involved in computerisation, such as invoices and appointments. However she is perfectly happy with the professionally designed MidexPro system that she already uses for these features, therefore they will not be included.

8.2. Areas considered for future computerisation

Niki informed me that very occasionally some of the insurance companies update their prices. An area which could be considered for future computerisation could be a screen scraping method which would check the prices against the existing database and perform updates. This would be useful to the system in order to keep it in date; however I do not have the time to create this method for all 7 insurance companies. Therefore this could be considered for future computerisation.

9. Solutions

9.1. Alternate solutions

Solution	Advantages	Disadvantages
Desktop application with a GUI	Simple to use without any extra training. Inputs will be obvious and easy to enter. Output will be clearly displayed on screen.	Without access to the machine the program will not be accessible. Each computer would need a separate copy of the program.
Creating a spreadsheet	Information such as prices and combinations will be displayed clearly for each of the insurance companies. Would be easy to view large quantities of data.	Would not be able to calculate the price for the user. Prices and combinations are sometimes updated which would make the spreadsheet out of date.
Command line application	Quicker to produce as a GUI would not be needed.	Training would be needed to understand some of the syntax. Not very user-friendly as data is pushed up the screen.
Web application	Would be available for use by everyone in the office There is already a website in place which is run on the office server. Can be accessed from anywhere.	Website would require extra design/programming knowledge. Anyone would be able to access the program unless a password system was put in place.

9.2. Justification of chosen solution

I have decided to make a Desktop application with a GUI front end and the database hosted on the server. My main reason for this is that I am familiar with the programing language Python and I believe all aspects required would be achievable using this method. Although a web application would allow use of the system from any computer, Niki was trying to move away from constantly visiting webpages as this relies heavily on there being an Internet connection. I am also unfamiliar with web coding and design.

I dismissed the other two ideas as neither would give Niki the full system she requested. A spread sheet would not be able to stay up to date or do the calculations for here, which was a requirement for the proposed system. The command line application would be difficult to use as training would be required and the people she wishes to have access to the system will not have this training.

Design

1. Overall System Design

1.1. Short description of the main parts of the system

There will be 9 main parts to the new system I am designing for Niki. This are listed below and followed with a more detail description of their functionality.

- Cambridge Medical Practice Billing System
    - Log In
    - General User Interface
    - Manage Weekly Operations
    - Individual Code Calculator
    - Patient Information
    - Code Information
    - User Information
    - Consultant Information
    - Output patient information

Log In

- This will ask the user for a username and password as the system will contain confidential information.

- These will be checked against a database to ensure that only certain users can access the system.

- Access levels will be placed on different accounts allowing different members of staff, such as the practice manager or a general member of staff, access to certain features.

General User Interface

- This part of the system will display navigational methods required for moving between the programs functions.

- It will use appropriately placed buttons, prompts and text windows so it is easy to use.

Manage Weekly Operations

- Will ask the user which set of patients they wish to view by asking for a consultant and/or week date.

- The option to generate the prices using the same method as the individual code calculator will be available.

Individual Code Calculator

- Prompts the user to input up to three operation codes with an optional input of multiple insurance companies.

- Calculates three prices, surgeon, anaesthetist and total, for operations after the user inputs a one to three codes.

- It will allow the user to input a specific insurance company or receive the price of the operation for all insurance companies.

Patient Information

- Ask the user if they want to edit existing patient information or add a new patient to the system.

- If the user chooses to edit existing patient records then they will be prompted with a search algorithm asking for the patient's name.

- This will then display the patient's records or display a list of patients if there is more than one patient of the searched name.

- If the user chooses to add a new patient record a blank patient record will be created for them to fill out.

- After checking the necessary fields are filled out and that the validation is correct the new patient will be added to the database. If not then the user will be shown an appropriate error message and will to be allowed to continue until the record is updated.

Code Information

- Will have 3 main features: add code, delete code and update.

- The add and delete options will simply allow the addition of new codes to be added to the database, or the removal of existing codes from the database.

- The updating of the codes will give the user to option to assign incompatible codes to a particular code. It will also allow the some users to change any of the insurance company prices associated with a code.

- Adding a new code will also ask the user to input the corresponding prices for the insurance company and any incompatible codes.

User Information

- Similarly to "Code Information" this section of the system will allow the addition and removal of new and existing users. However these features will only be available to users with top level access.

- Some updates will be available to all users such as password or email updates.

Consultant Information

- Will allow addition and deletion of consultants as they leave and join the practice.

- An update feature will let the user change some consultant details such as email and name.

Output patient information

- This part of the system has 2 output features, to the screen and printing.

- Screen outputs will be used for current system operations and will display only the relevant data regarding the user's use of the system.

- Printing will allow any data displayed on the screen to be put into report form and sent to the office printer. This will allow for invoices and patient reports for filing.

1.2. System flowcharts showing an overview of the complete system

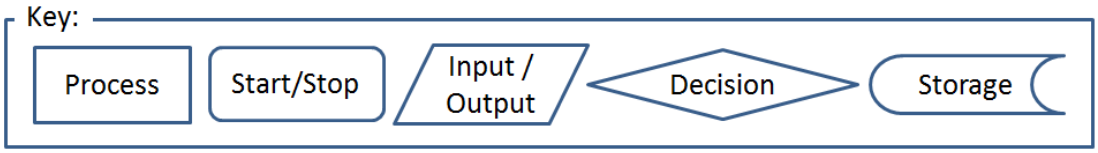


Fig 1.

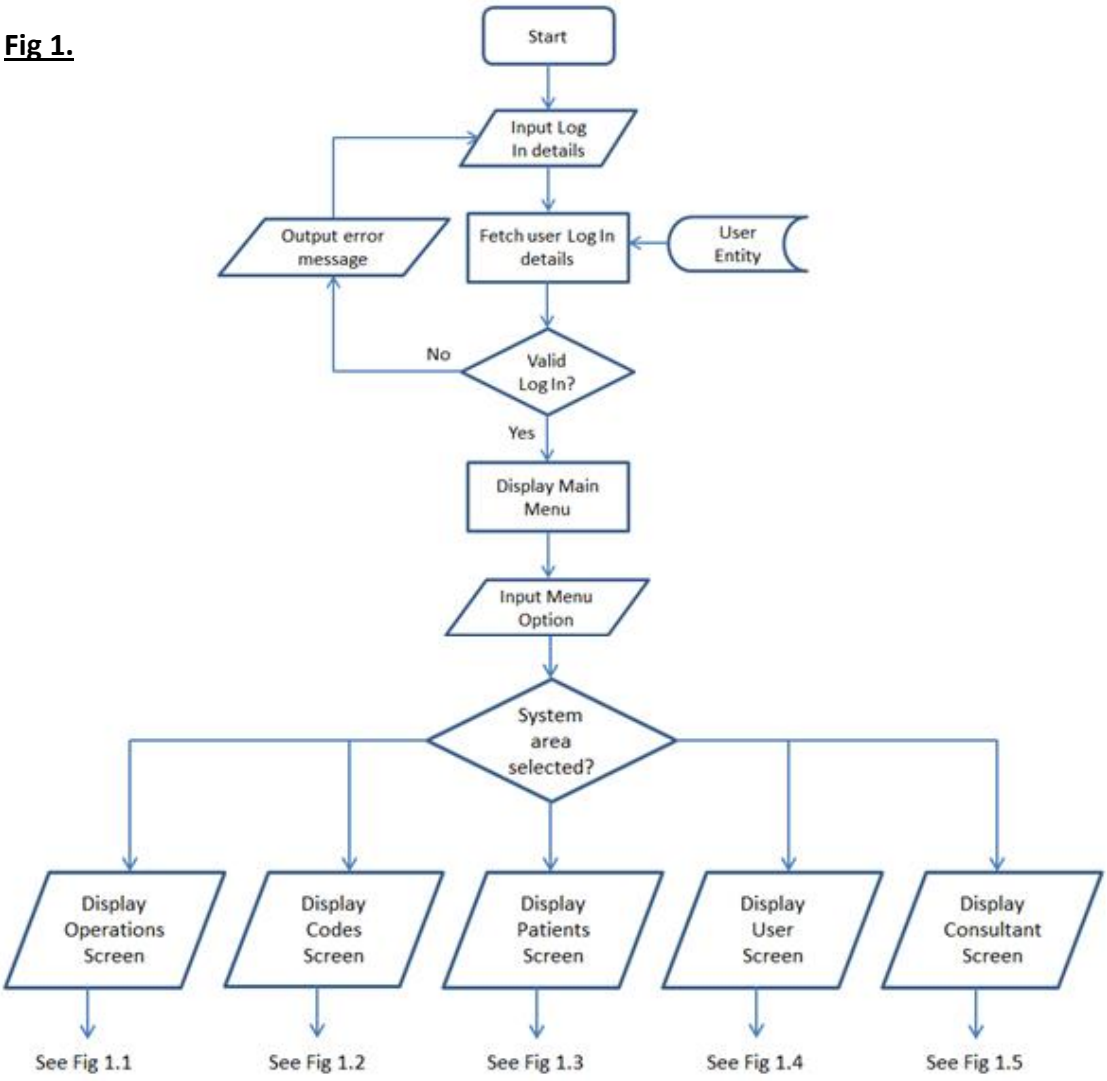


Fig 1.1

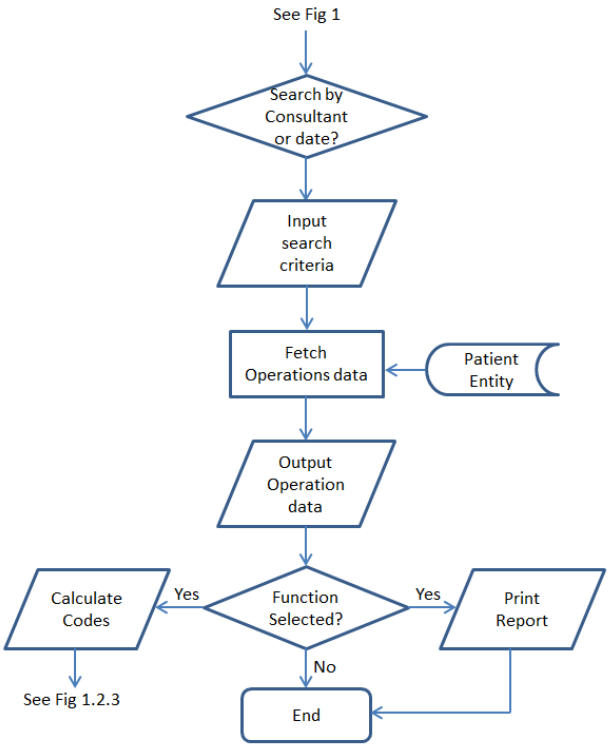


Fig 1.2

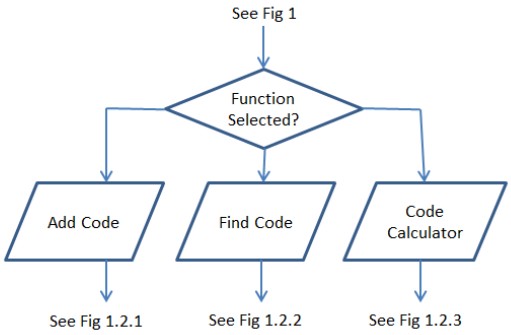


Fig 1.2.1

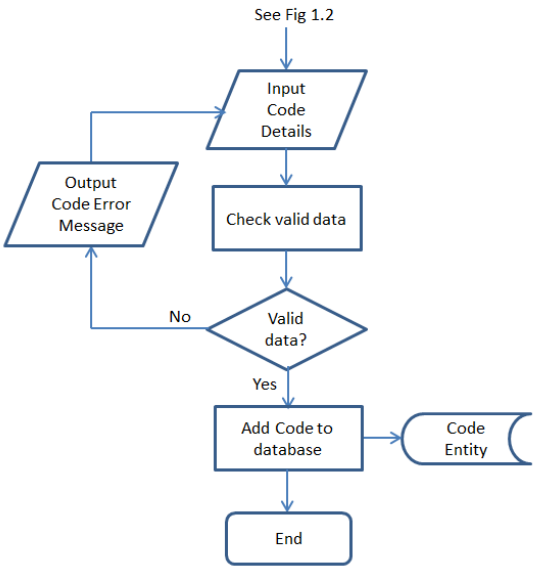


Fig 1.2.2

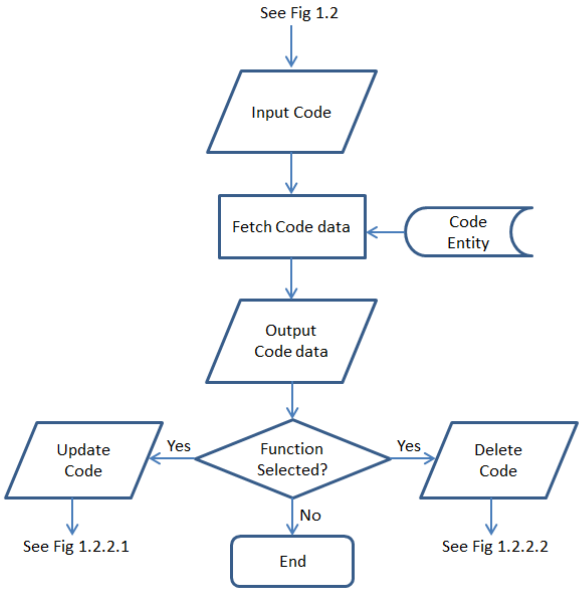


Fig 1.2.2.1

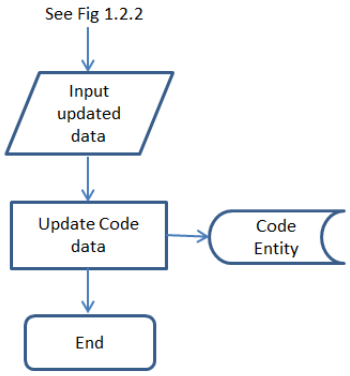


Fig 1.2.2.2

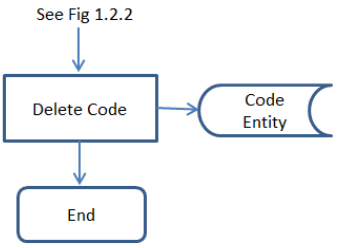


Fig 1.2.3

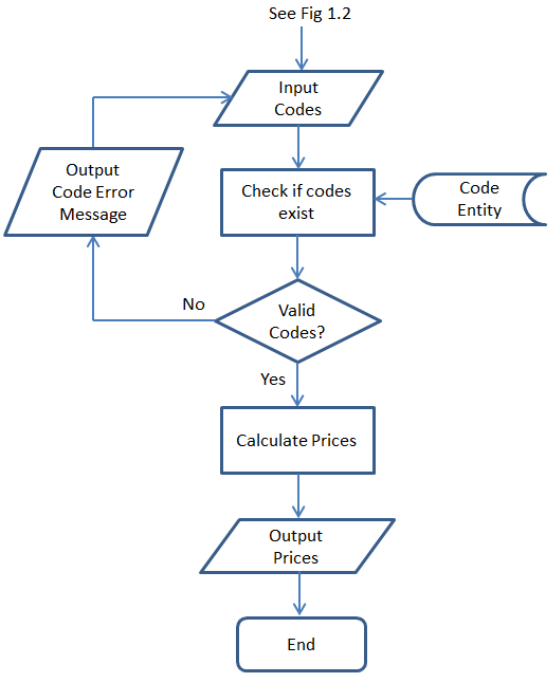


Fig 1.3

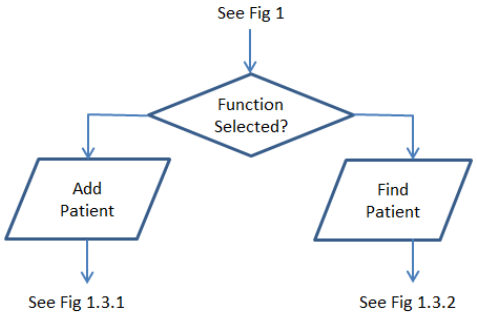


Fig 1.3.1

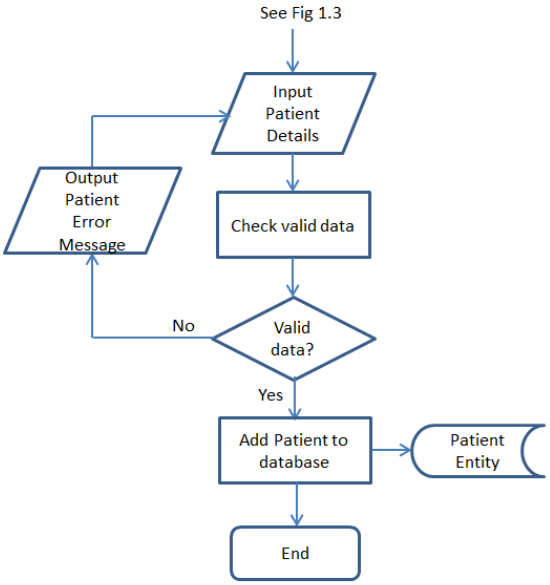


Fig 1.3.2

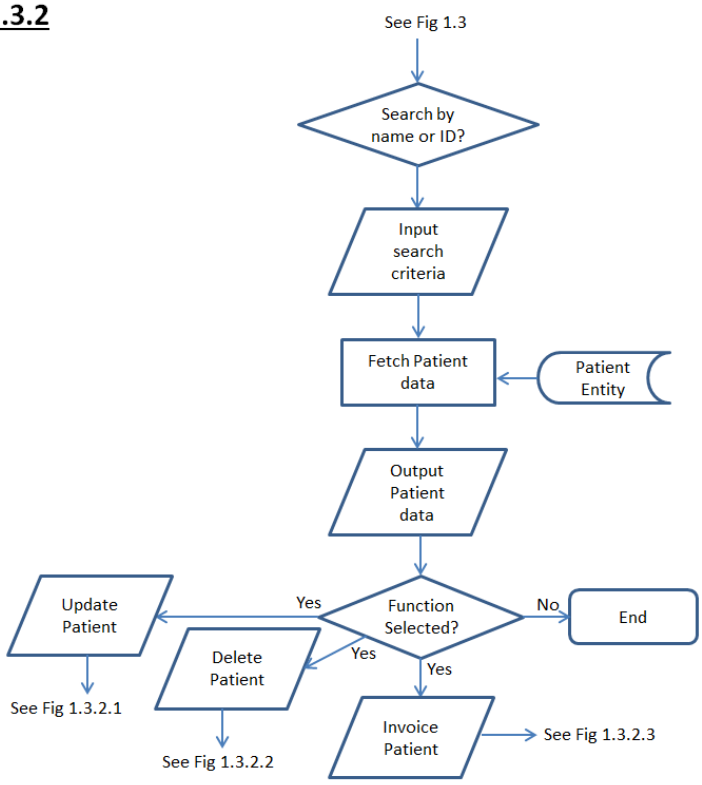


Fig 1.3.2.1

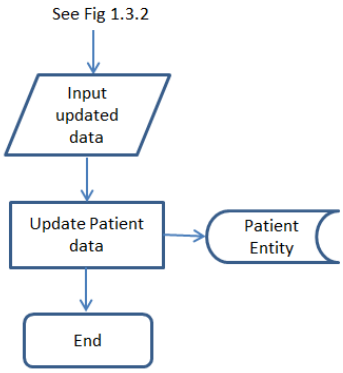


Fig 1.3.2.2

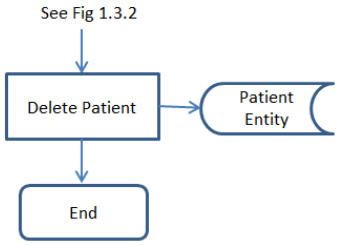


Fig 1.3.2.3

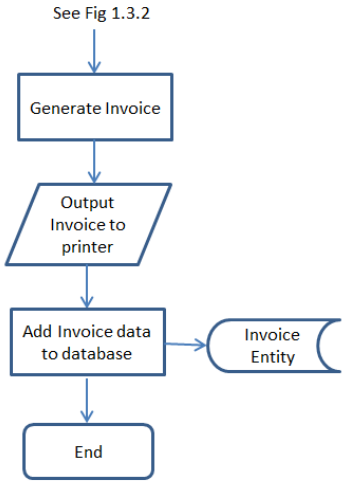


Fig 1.4

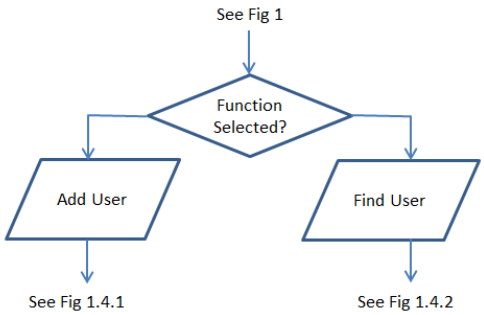


Fig 1.4.1

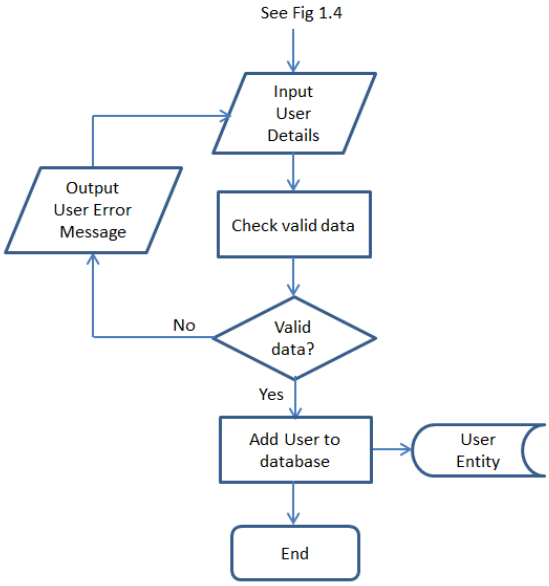


Fig 1.4.2

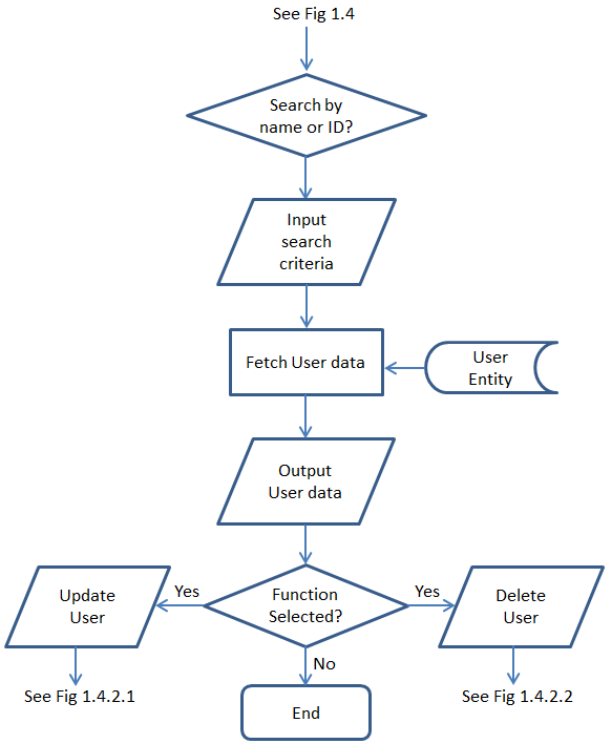


Fig 1.4.2.1

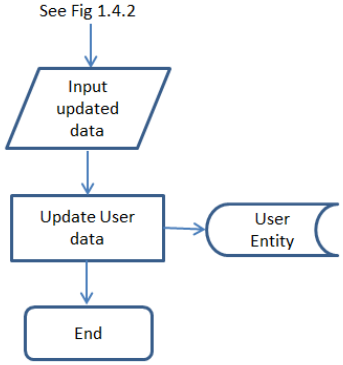


Fig 1.4.2.2

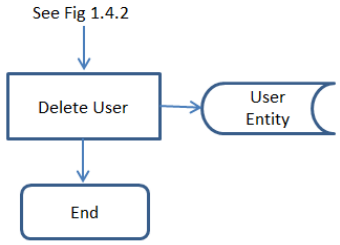


Fig 1.5

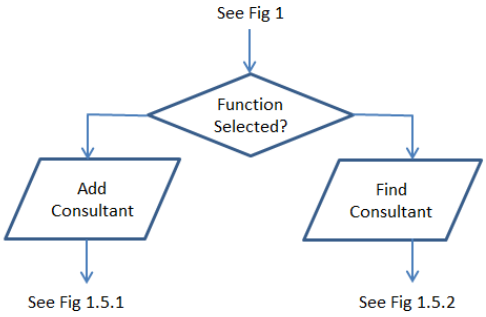


Fig 1.5.1

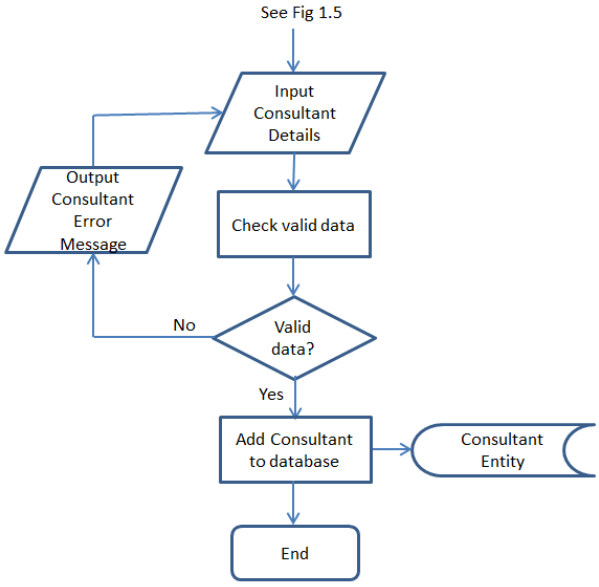


Fig 1.5.2

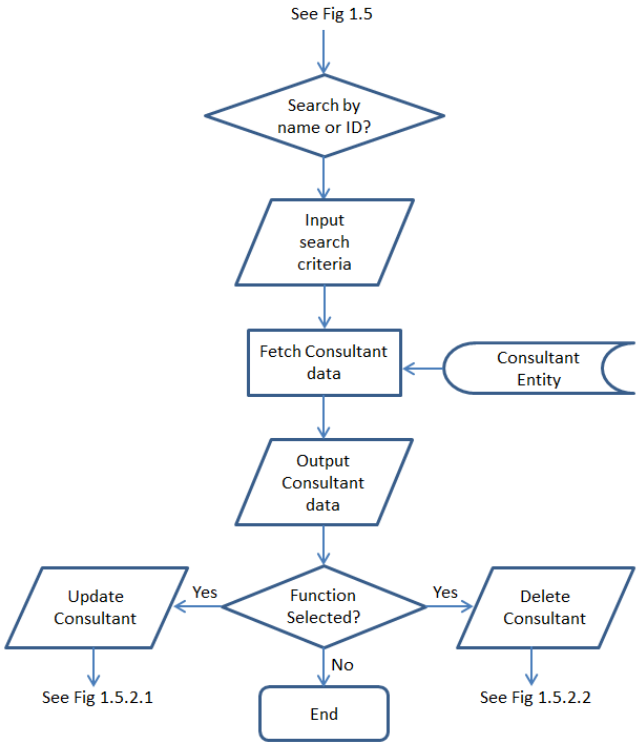


Fig 1.5.2.1

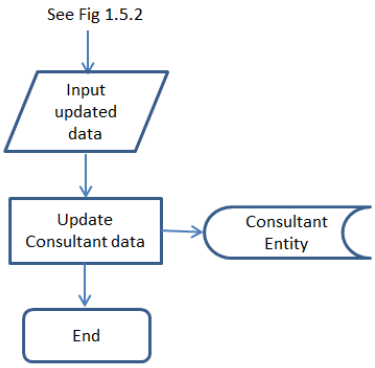
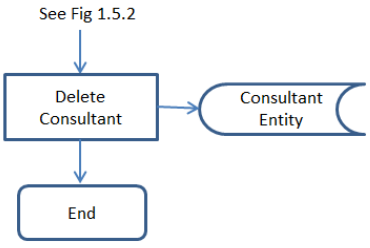
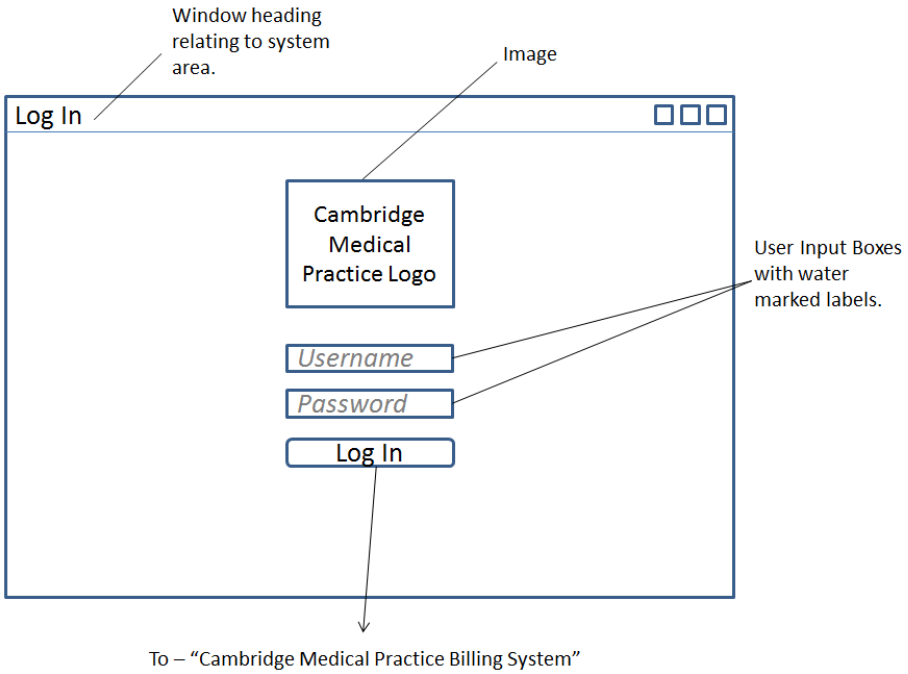
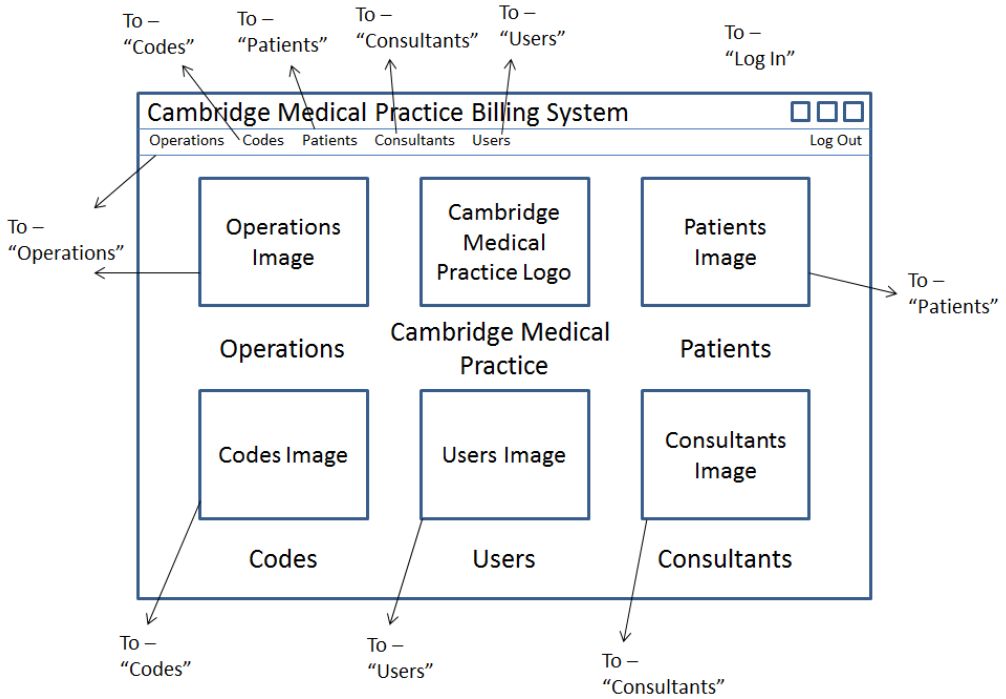


Fig 1.5.2.2



2. User Interface Designs





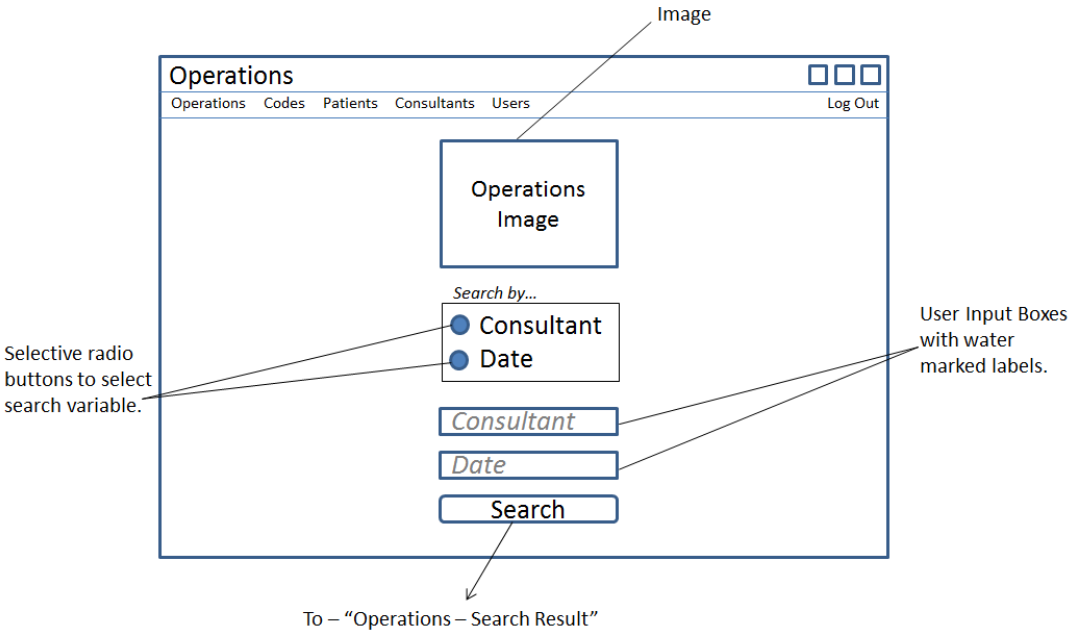
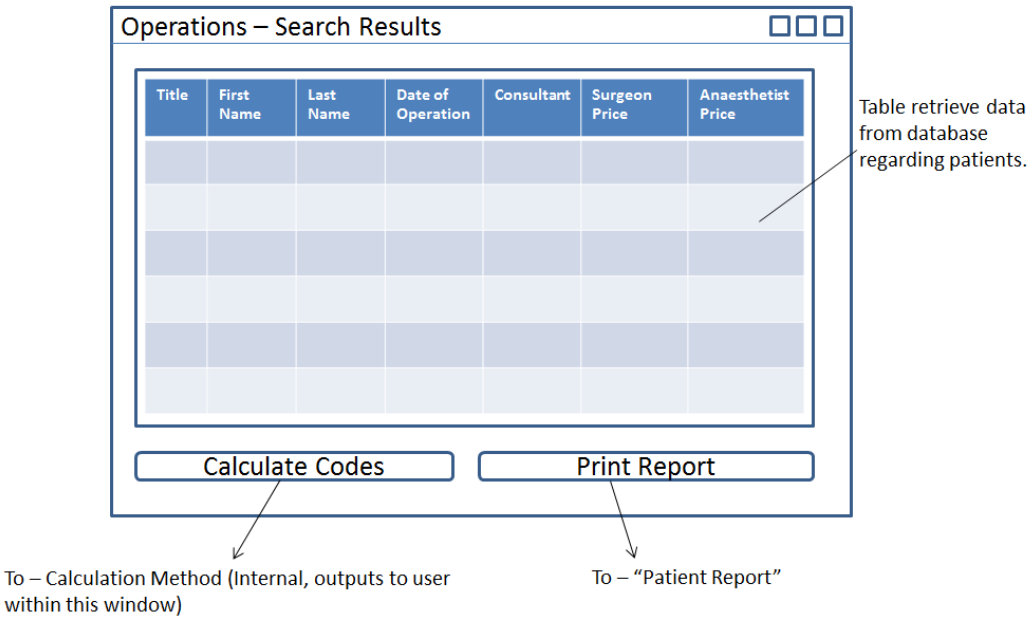


Image
Operations
Operations Codes Patients Consultants Users Log Out
Operations Image
Search by...
Consultant
Date
Consultant
Date
Search
Selective radio buttons to select search variable.
User Input Boxes with water marked labels.
To – "Operations – Search Result"


Operations – Search Results
Title
First Name
Last Name
Date of Operation
Consultant
Surgeon Price
Anaesthetist Price
Table retrieve data from database regarding patients.
Calculate Codes
Print Report
To – Calculation Method (Internal, outputs to user within this window)
To – "Patient Report"

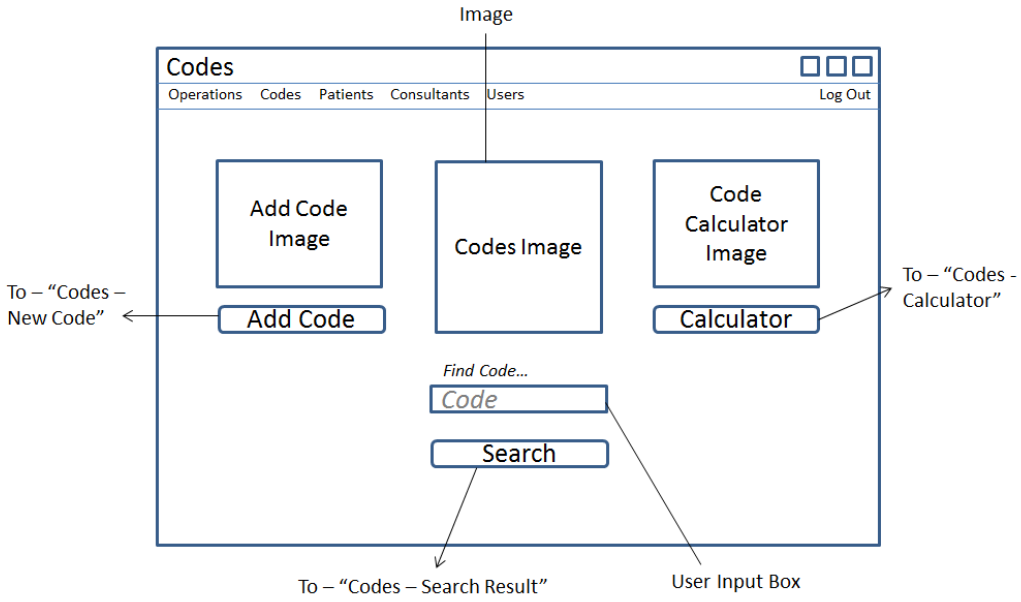
Patient Report – <Date or Consultant>

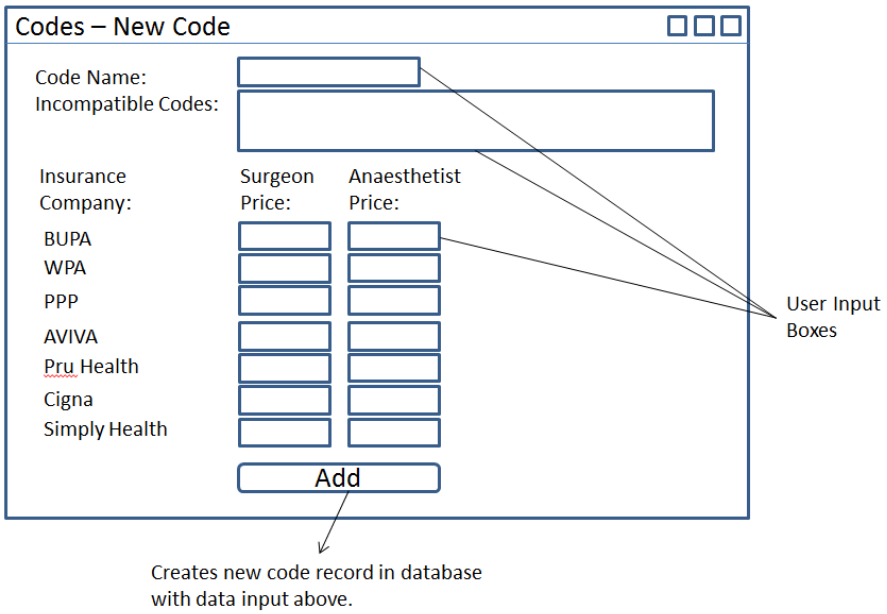
<title> <first name> <last name>
<consultant> - <date of operation>
Surgeon Price - <surgeon price>
Anaesthetist Price - <anaesthetist price>
Total Price - <anaesthetist price + surgeon price >

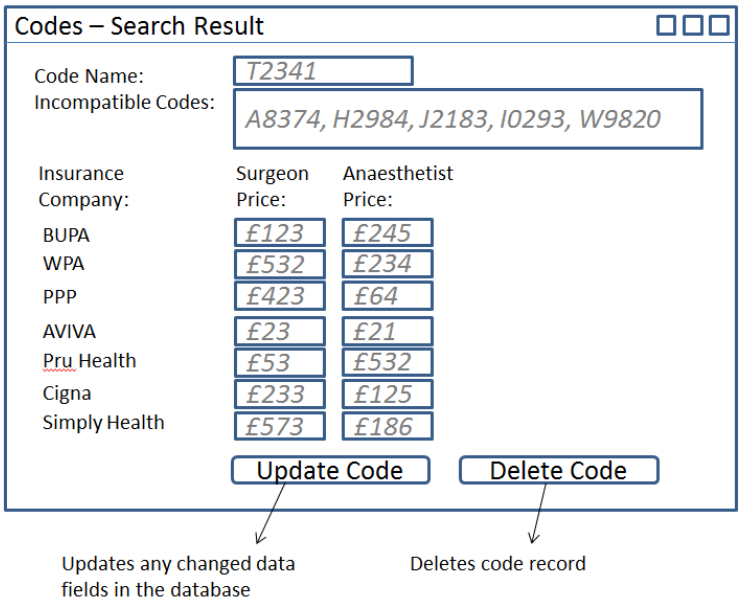
<title> <first name> <last name>
<consultant> - <date of operation>
Surgeon Price - <surgeon price>
Anaesthetist Price - <anaesthetist price>
Total Price - <anaesthetist price + surgeon price >

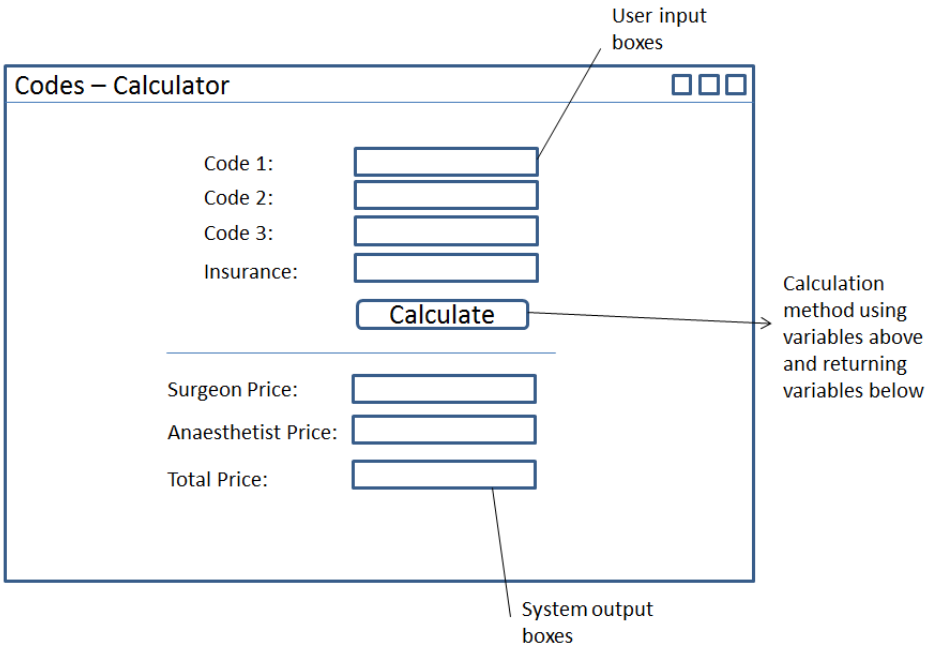
<title> <first name> <last name>
<consultant> - <date of operation>
Surgeon Price - <surgeon price>
Anaesthetist Price - <anaesthetist price>
Total Price - <anaesthetist price + surgeon price >

Produced by - <username> Date - <date>




Codes – New Code
Code Name:
Incompatible Codes:
Insurance Company:
Surgeon Price:
Anaesthetist Price:
BUPA
WPA
PPP
AVIVA
Pru Health
Cigna
Simply Health
Add
User Input Boxes
Creates new code record in database with data input above.


Codes – Search Result
Code Name:
T2341
Incompatible Codes:
A8374, H2984, J2183, I0293, W9820
Insurance Company:
Surgeon Price:
Anaesthetist Price:
BUPA £123 £245
WPA £532 £234
PPP £423 £64
AVIVA £23 £21
Pru Health £53 £532
Cigna £233 £125
Simply Health £573 £186
Update Code
Delete Code
Updates any changed data fields in the database
Deletes code record


User input boxes
Codes – Calculator
Code 1:
Code 2:
Code 3:
Insurance:
Calculate
Calculation method using variables above and returning variables below
Surgeon Price:
Anaesthetist Price:
Total Price:
System output boxes

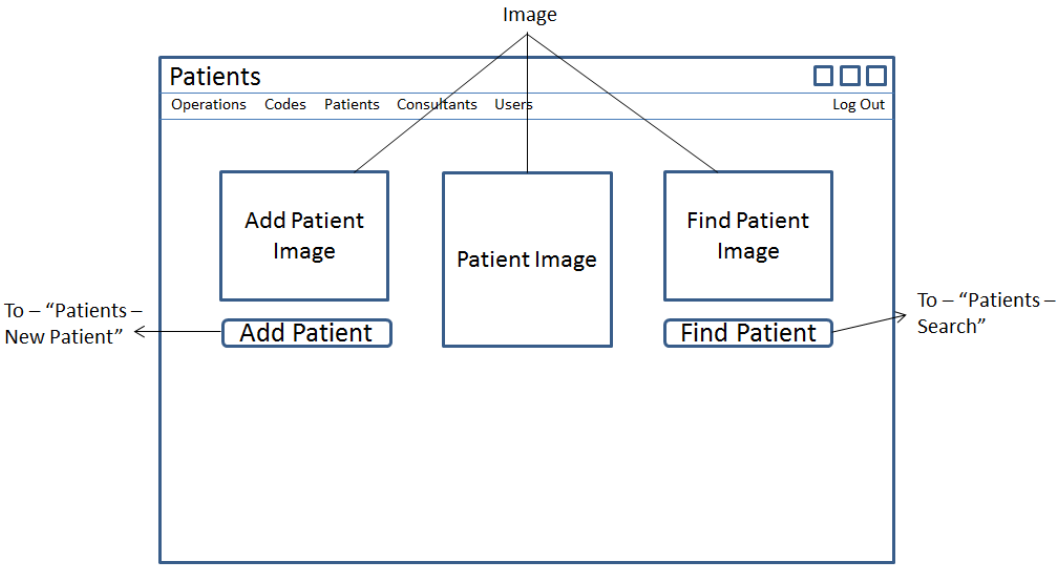
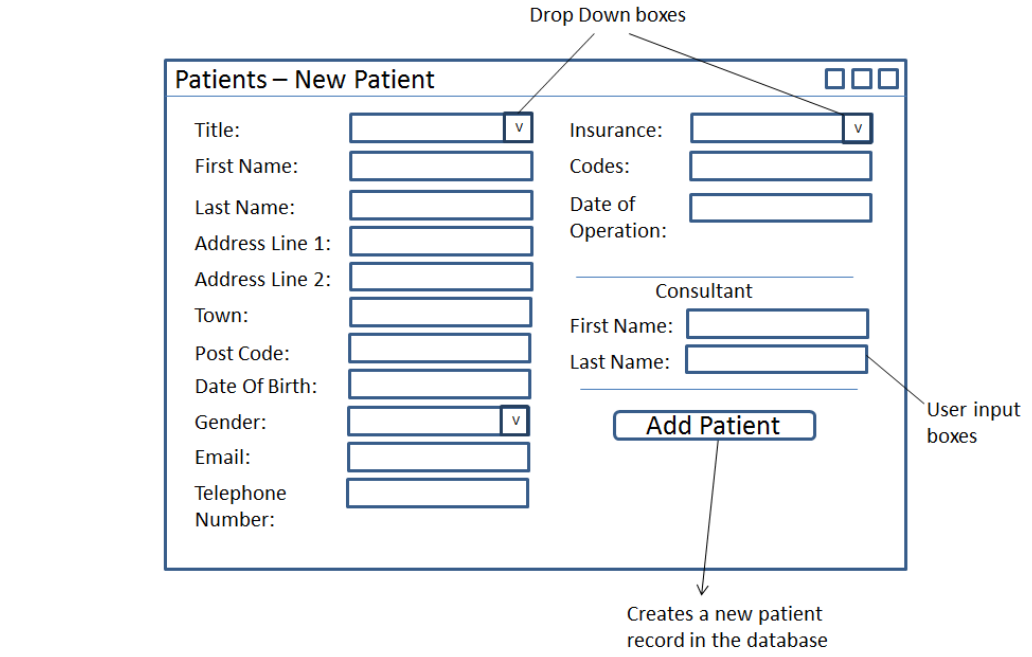


Image
Patients
Operations Codes Patients Consultants Users Log Out
Add Patient Image
Patient Image
Find Patient Image
To – "Patients – New Patient"
Add Patient
Find Patient
To – "Patients – Search"


Drop Down boxes
Patients – New Patient
Title:
First Name:
Last Name:
Address Line 1:
Address Line 2:
Town:
Post Code:
Date Of Birth:
Gender:
Email:
Telephone Number:
Insurance:
Codes:
Date of Operation:
Consultant
First Name:
Last Name:
Add Patient
User input boxes
Creates a new patient record in the database

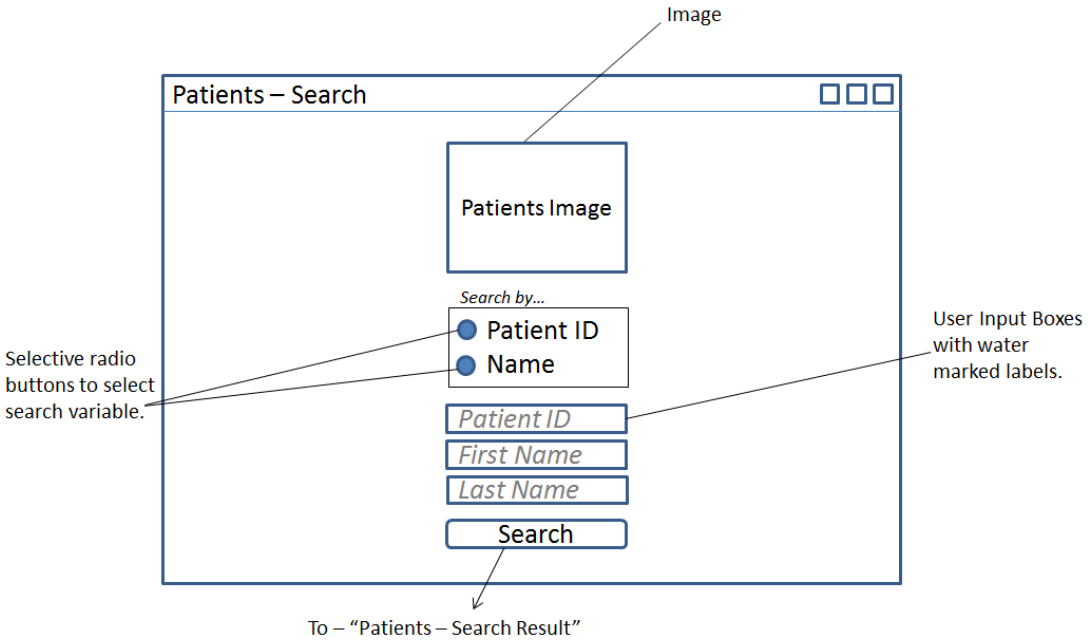
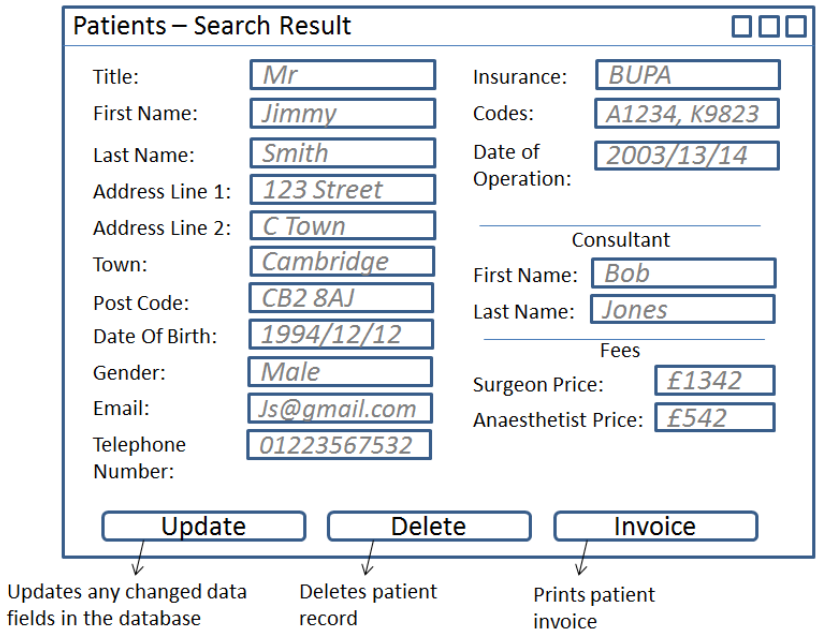


Image
Patients – Search
Patients Image
Search by...
Patient ID
Name
Patient ID
First Name
Last Name
Search
Selective radio buttons to select search variable.
User Input Boxes with water marked labels.
To – "Patients – Search Result"


Patients – Search Result
Title: Mr
First Name: Jimmy
Last Name: Smith
Address Line 1: 123 Street
Address Line 2: C Town
Town: Cambridge
Post Code: CB2 8AJ
Date Of Birth: 1994/12/12
Gender: Male
Email: Js@gmail.com
Telephone Number: 01223567532
Insurance: BUPA
Codes: A1234, K9823
Date of Operation: 2003/13/14
Consultant
First Name: Bob
Last Name: Jones
Fees
Surgeon Price: £1342
Anaesthetist Price: £542
Update
Delete
Invoice
Updates any changed data fields in the database
Deletes patient record
Prints patient invoice

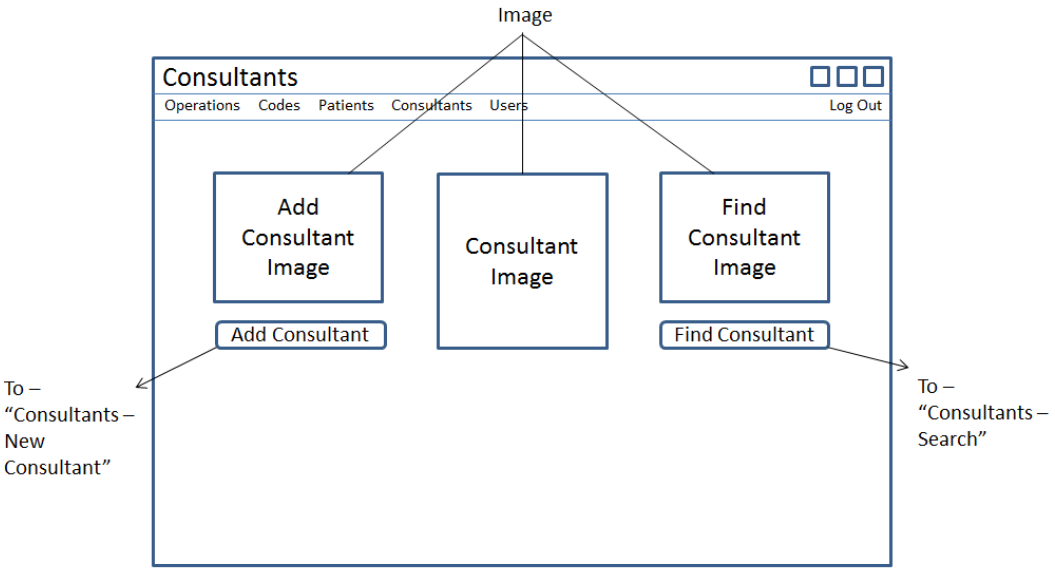
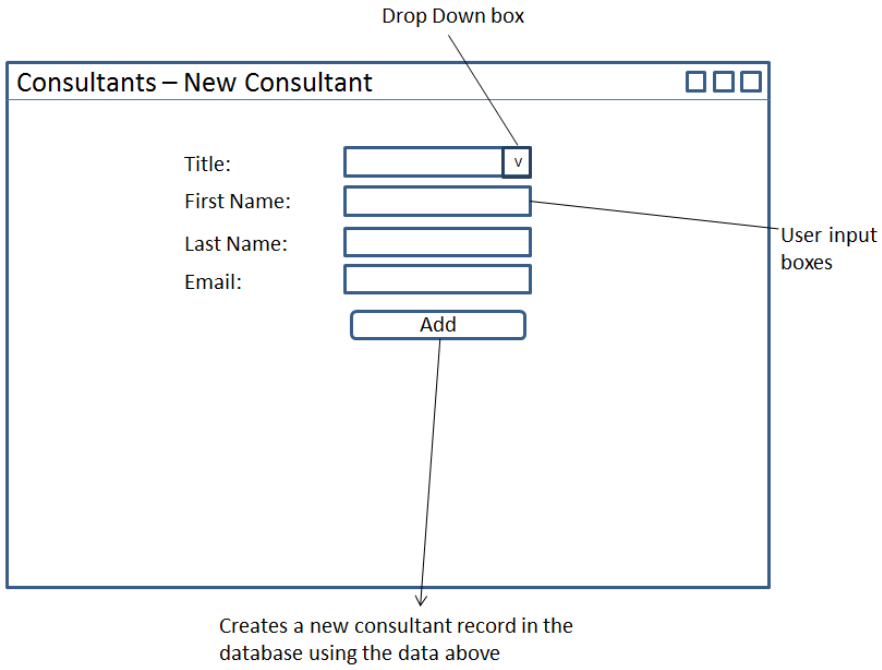


Image
Consultants
Operations Codes Patients Consultants Users Log Out
Add Consultant Image
Consultant Image
Find Consultant Image
Add Consultant
Find Consultant
To – "Consultants – New Consultant"
To – "Consultants – Search"


Drop Down box
Consultants – New Consultant
Title:
First Name:
Last Name:
Email:
Add
User input boxes
Creates a new consultant record in the database using the data above

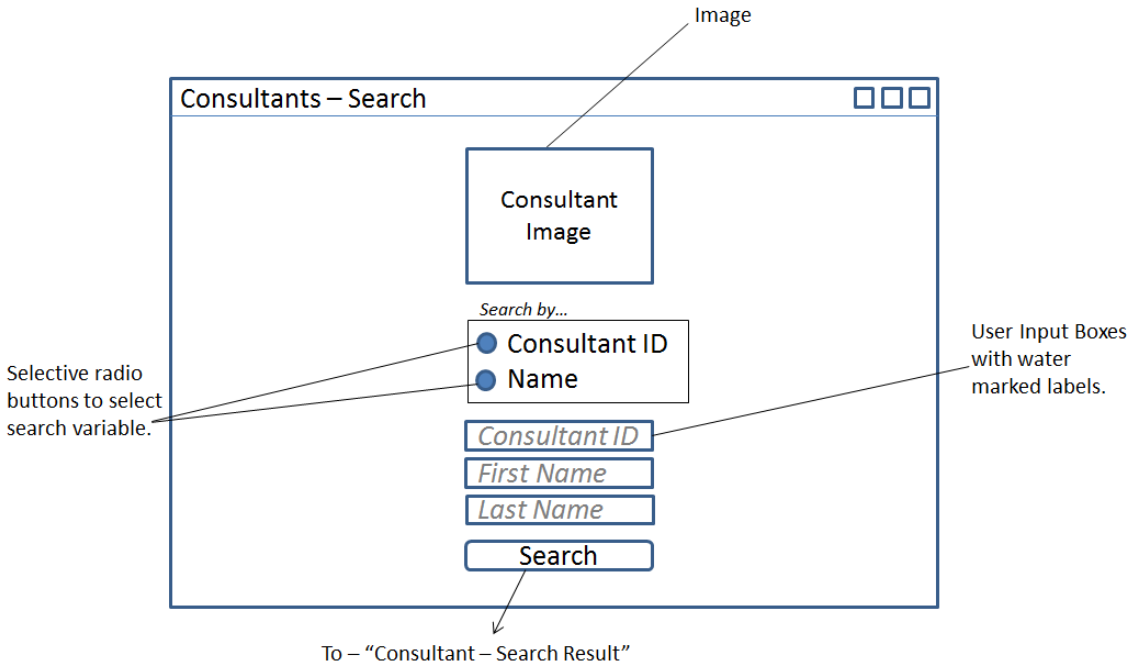
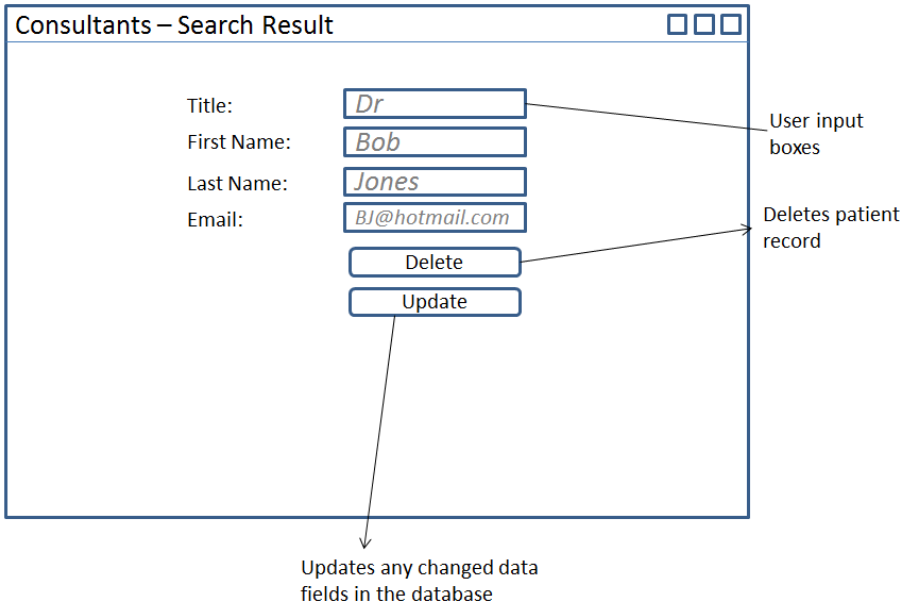


Image
Consultants – Search
Consultant Image
Search by...
Consultant ID
Name
Consultant ID
First Name
Last Name
Search
Selective radio buttons to select search variable.
User Input Boxes with water marked labels.
To – "Consultant – Search Result"


Consultants – Search Result
Title:
Dr
First Name:
Bob
Last Name:
Jones
Email:
BJ@hotmail.com
Delete
Update
User input boxes
Deletes patient record
Updates any changed data fields in the database

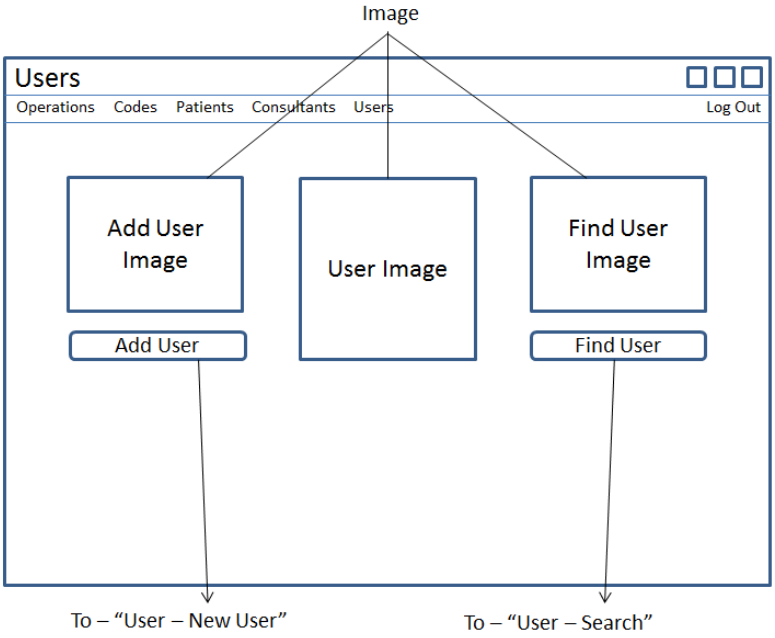
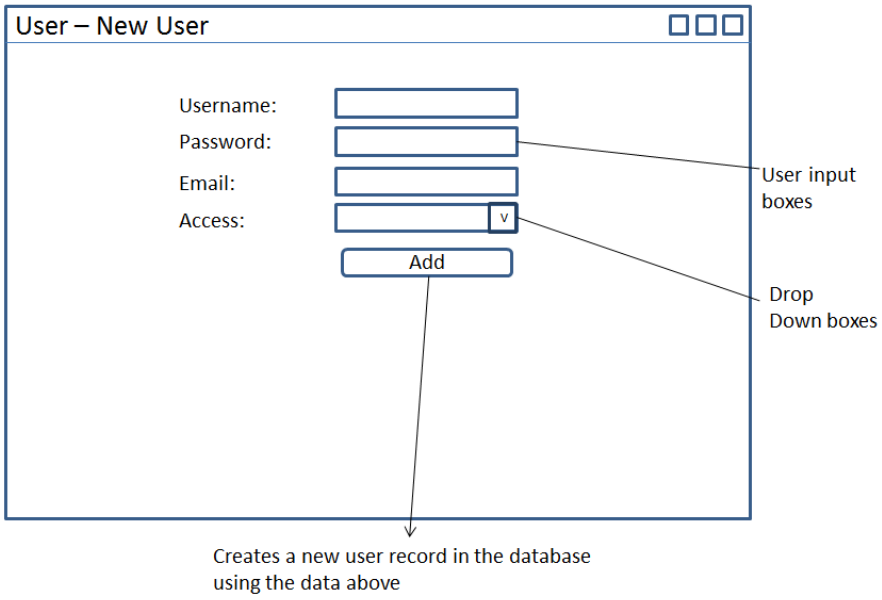


Image
Users
Operations
Codes
Patients
Consultants
Users
Log Out
Add User Image
User Image
Find User Image
Add User
Find User
To – "User – New User"
To – "User – Search"


User – New User
Username:
Password:
Email:
Access:
Add
User input boxes
Drop Down boxes
Creates a new user record in the database using the data above

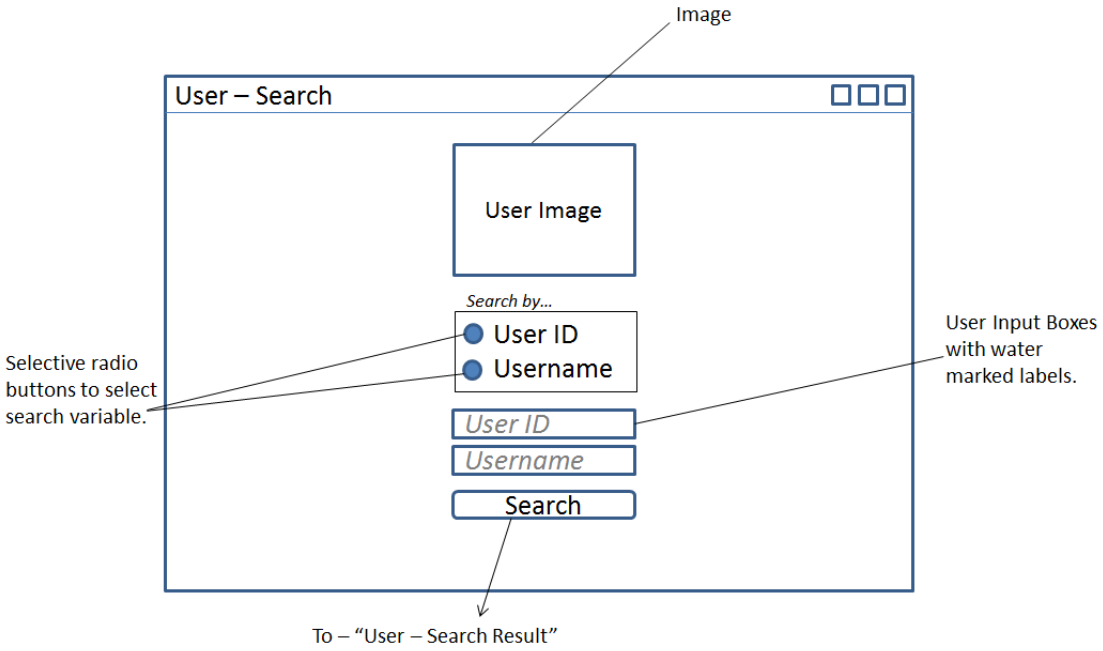
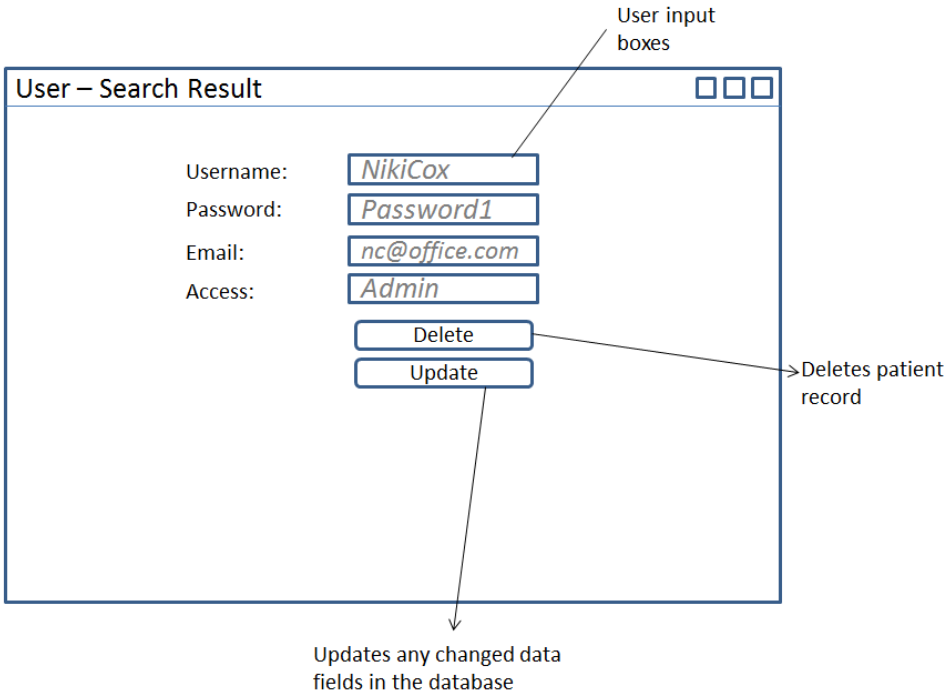


Image
User – Search
User Image
Search by...
User ID
Username
User ID
Username
Search
Selective radio buttons to select search variable.
User Input Boxes with water marked labels.
To – "User – Search Result"



3. Hardware Specifications

Below is a list of hardware devices which will be used in the running the system:

- A keyboard as it is a simple input method which the system users are already familiar with.
- A mouse which in combination with the keyboard will allow the user to input all data required and navigate through the system.
- A visual display will be required to output information back to the user.
- The server will be required to host the database; this is because the database will need to be accessed from multiple computers.
- A printer which will be used to output report lists and invoices as hard copies for filing.

All of these devices are already available and in use within the office, so there will be no need of new hardware.

4. Program Structure

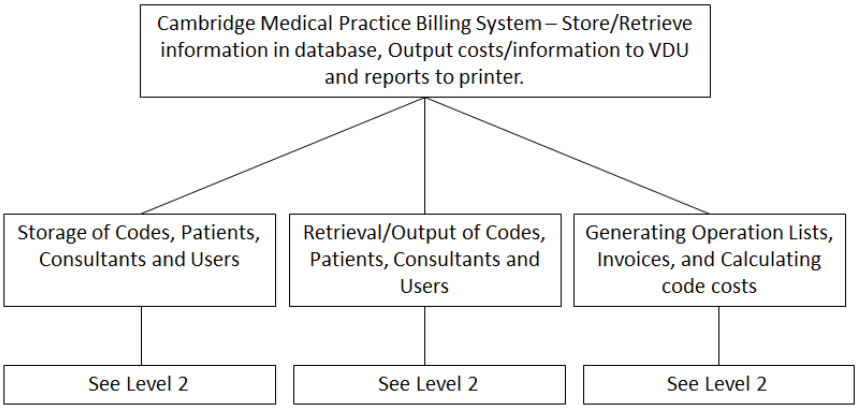
4.1. Top down design structure charts

The structure charts bellow, break down the problems within my system into processes. Processes which "Get" – retrieve data from the user. Whilst processes that "Fetch" – retrieve data from the database.

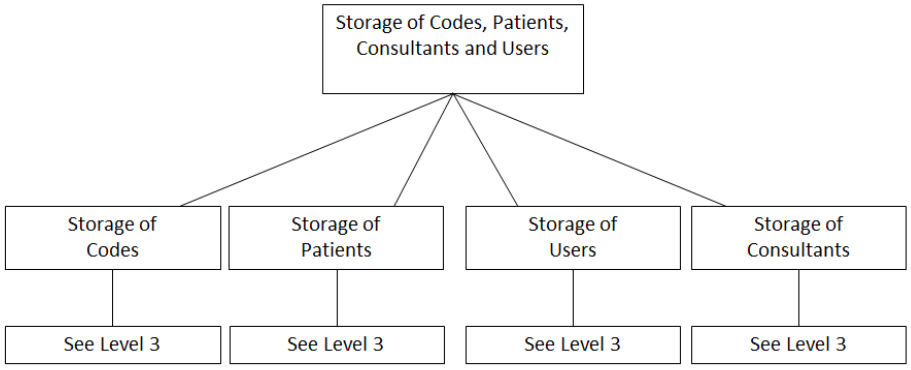
This table is the key to all of the data symbols found with the structure charts.

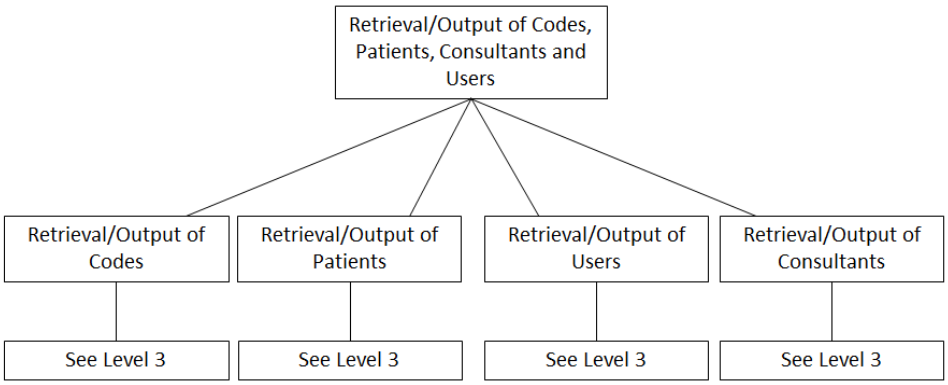
Data Symbol	Definition
C	Code name, for example – A1234
Co	The Surgeon and Anaesthetist cost for the code for each Insurance Company
P	Patient Details, for example – Title, First Name etc.
Con	Consultant Details, for example – Title, First Name etc.
I	Insurance Details, for example – Company Name
U	User Details, for example – Username, Password etc.
D	Date, used as a possible query parameter
Da	Data, retrieved from the database
R	Report, produced from data or inputs
Up	Update, used as a system function selector
De	Delete, used as a system function selector

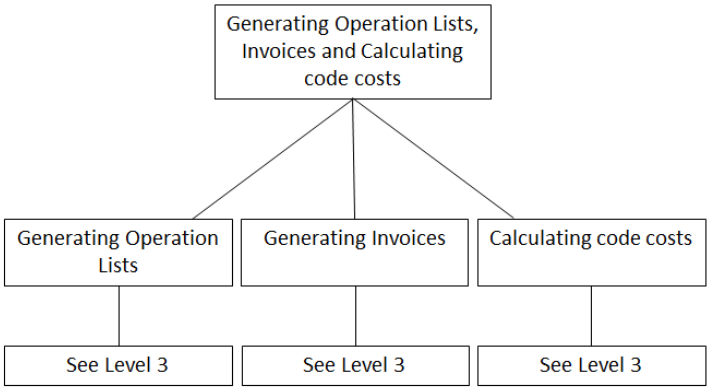
Level 1



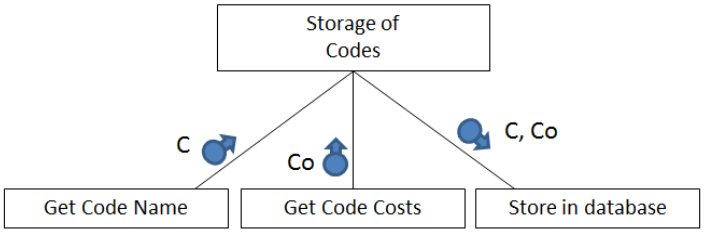
Level 2

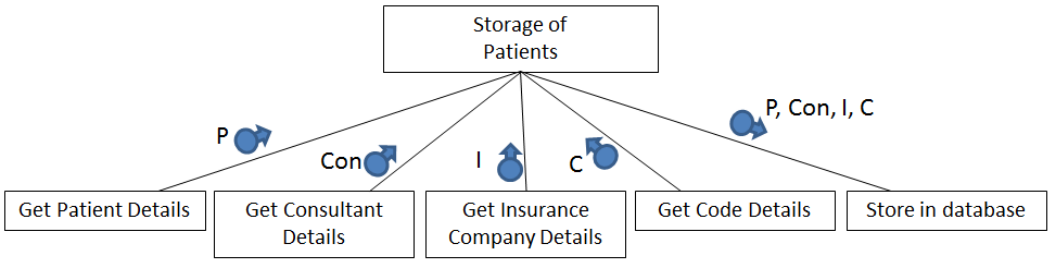


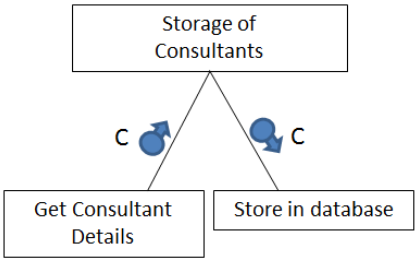

Retrieval/Output of Codes, Patients, Consultants and Users
Retrieval/Output of Codes
Retrieval/Output of Patients
Retrieval/Output of Users
Retrieval/Output of Consultants
See Level 3
See Level 3
See Level 3
See Level 3

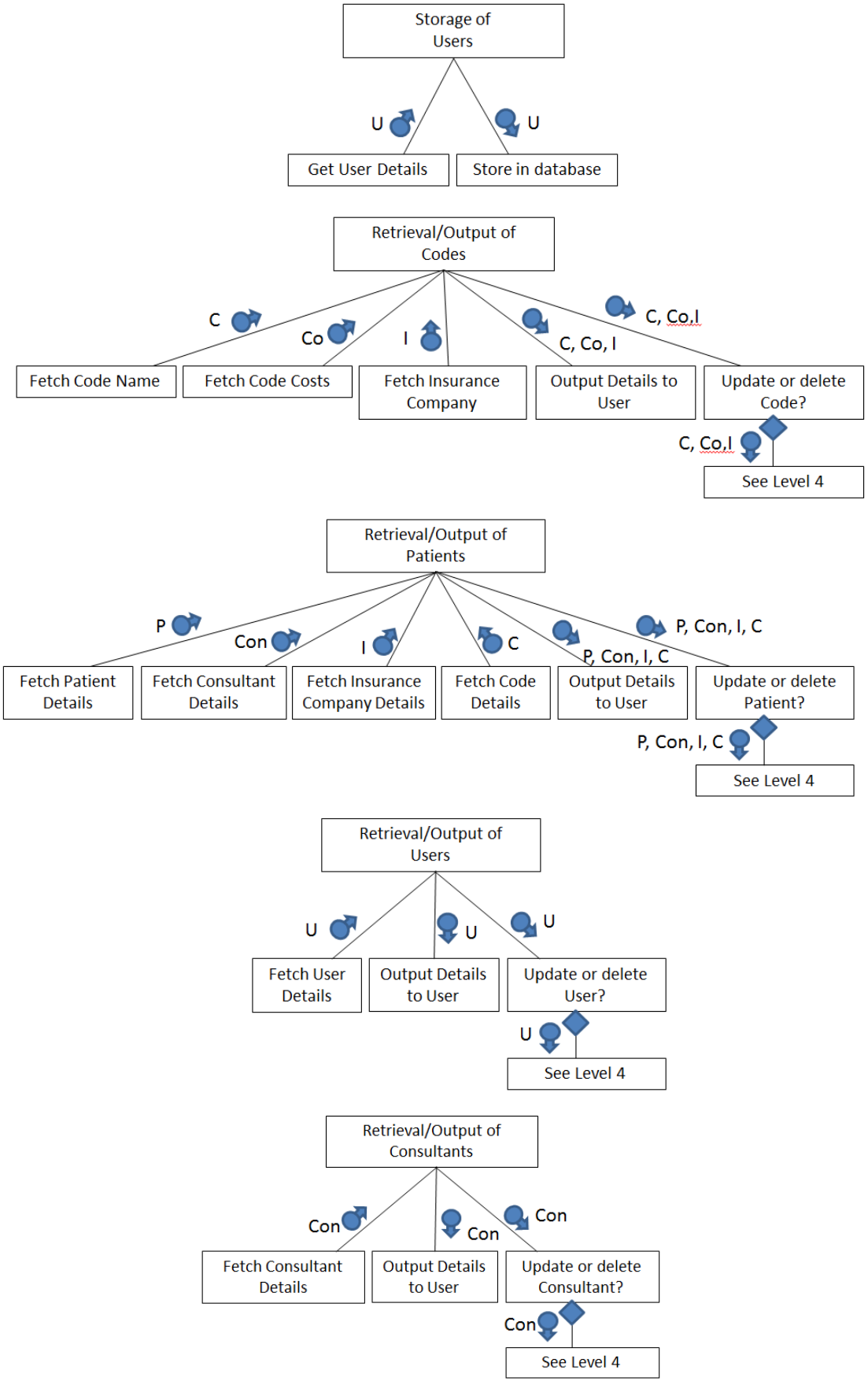

Generating Operation Lists, Invoices and Calculating code costs
Generating Operation Lists
Generating Invoices
Calculating code costs
See Level 3
See Level 3
See Level 3

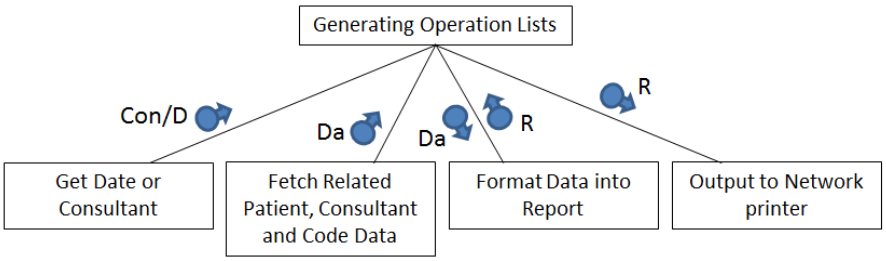
Level 3

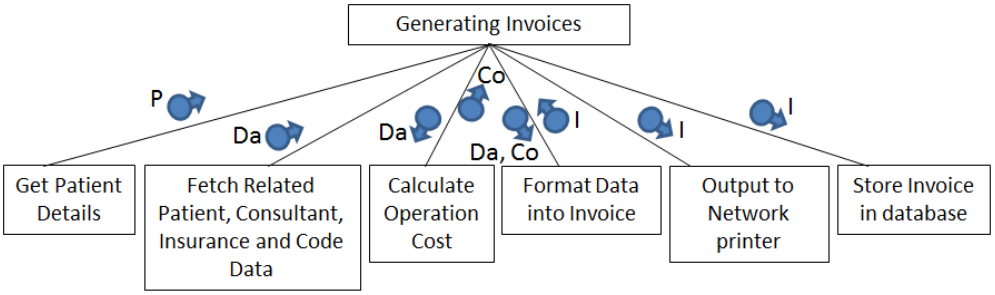

Storage of Codes
C
Co
C, Co
Get Code Name
Get Code Costs
Store in database

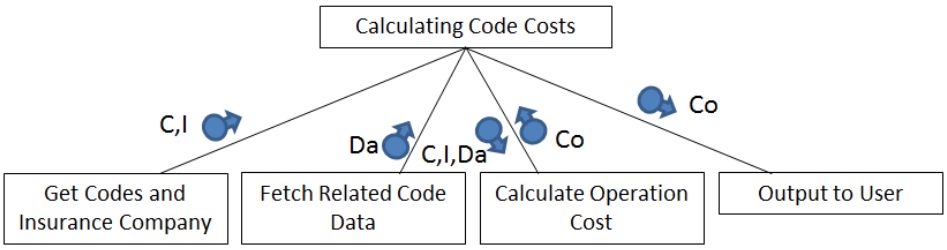

Storage of Patients
P
Con
I
C
P, Con, I, C
Get Patient Details
Get Consultant Details
Get Insurance Company Details
Get Code Details
Store in database

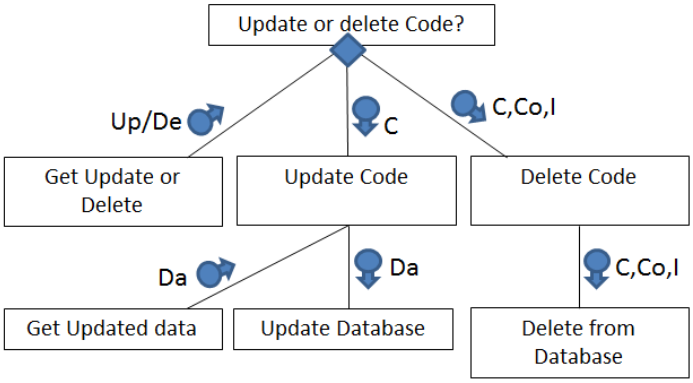

Storage of Consultants
C
C
Get Consultant Details
Store in database

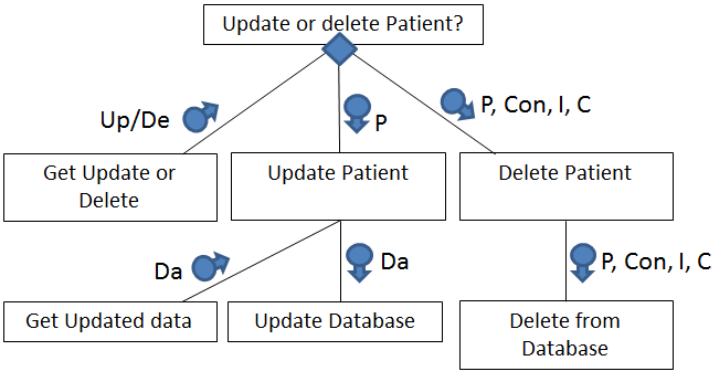

Storage of Users
U
U
Get User Details
Store in database
Retrieval/Output of Codes
C
Co
I
C, Co, I
C, Co,I
Fetch Code Name
Fetch Code Costs
Fetch Insurance Company
Output Details to User
Update or delete Code?
C, Co,I
See Level 4
Retrieval/Output of Patients
P
Con
I
C
P, Con, I, C
P, Con, I, C
Fetch Patient Details
Fetch Consultant Details
Fetch Insurance Company Details
Fetch Code Details
Output Details to User
Update or delete Patient?
P, Con, I, C
See Level 4
Retrieval/Output of Users
U
U
U
Fetch User Details
Output Details to User
Update or delete User?
U
See Level 4
Retrieval/Output of Consultants
Con
Con
Con
Fetch Consultant Details
Output Details to User
Update or delete Consultant?
Con
See Level 4

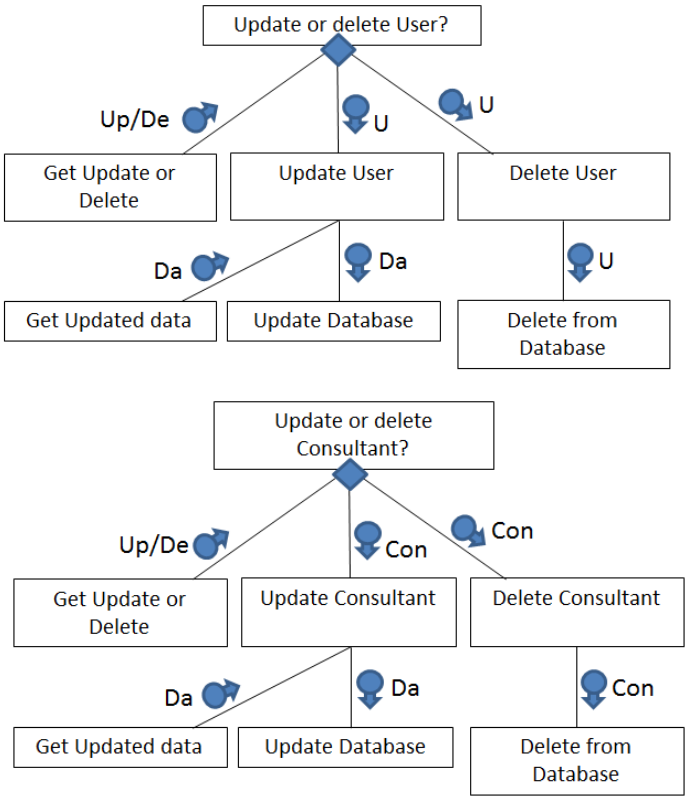

Generating Operation Lists
Con/D
Da
Da
R
R
Get Date or Consultant
Fetch Related Patient, Consultant and Code Data
Format Data into Report
Output to Network printer


Generating Invoices
P
Da
Da
Co
Da, Co
I
I
I
Get Patient Details
Fetch Related Patient, Consultant, Insurance and Code Data
Calculate Operation Cost
Format Data into Invoice
Output to Network printer
Store Invoice in database


Calculating Code Costs
C,I
Da
C,I,Da
Co
Co
Get Codes and Insurance Company
Fetch Related Code Data
Calculate Operation Cost
Output to User


Update or delete Code?
Up/De
C
C,Co,I
Get Update or Delete
Update Code
Delete Code
Da
Da
C,Co,I
Get Updated data
Update Database
Delete from Database


Update or delete Patient?
Up/De
P
P, Con, I, C
Get Update or Delete
Update Patient
Delete Patient
Da
Da
P, Con, I, C
Get Updated data
Update Database
Delete from Database

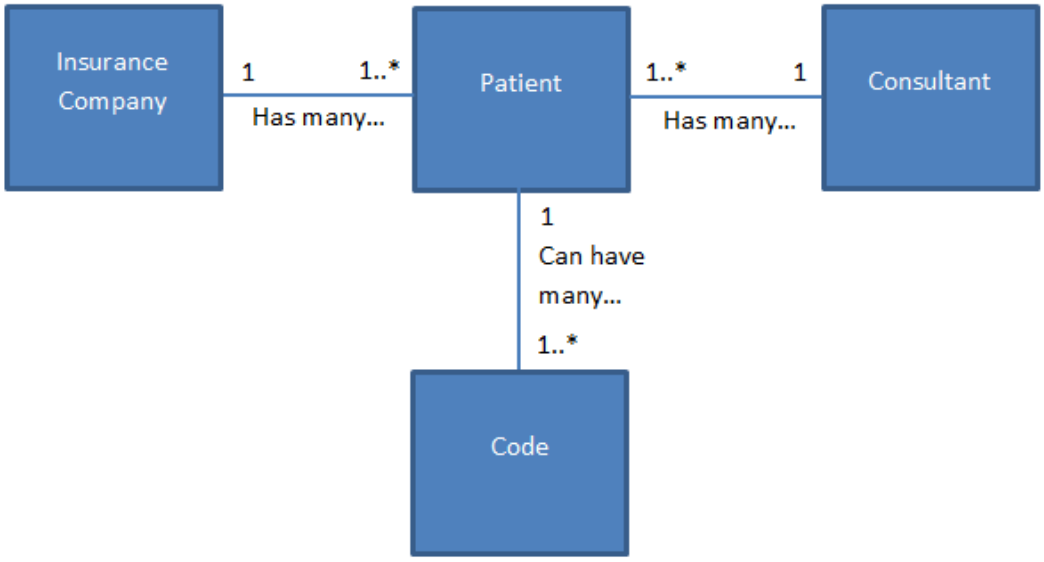

Update or delete User?
Up/De
U
U
Get Update or Delete
Update User
Delete User
Da
Da
U
Get Updated data
Update Database
Delete from Database
Update or delete Consultant?
Up/De
Con
Con
Get Update or Delete
Update Consultant
Delete Consultant
Da
Da
Con
Get Updated data
Update Database
Delete from Database

4.2. Algorithms in pseudo-code for each data transformation process

This first algorithm shows how will decide which calculation method to apply based on given codes and insurance company. It will then calculate the surgeon and anaesthetist price using the selected method and the price of the codes.

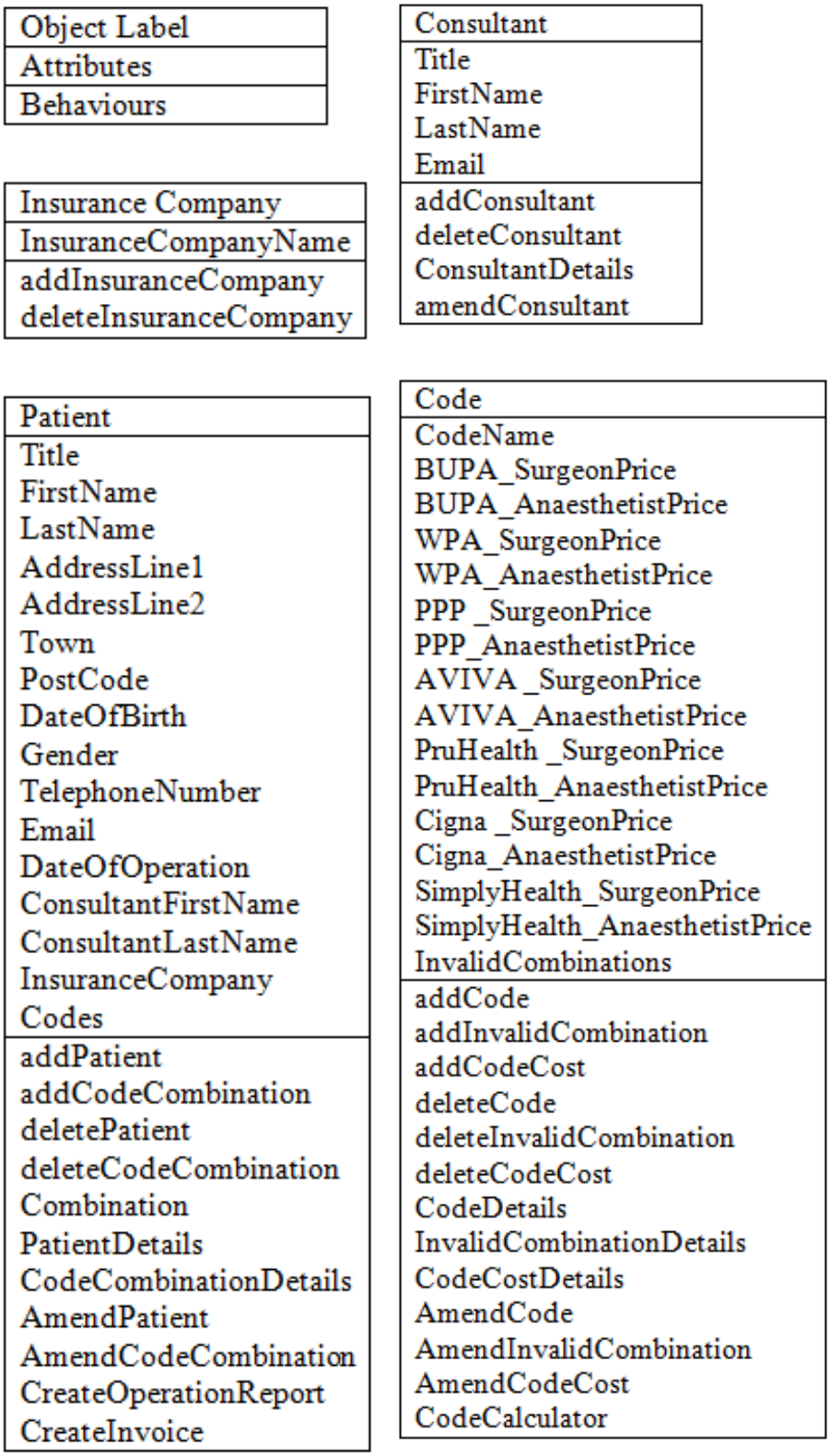
```
CodeList ← RETRIEVE codeList
PriceList ← RETRIEVE priceList
InsuranceCompany ← RETRIEVE InsuranceCompany
IF InsuranceCompany IN [“BUPA”,”WPA”, “Pru Health”, “Cigna”, “Simply Health”]
        MaxPrice ← 0
        FOR Count IN RANGE  LENGTH(CodeList)
                IF PriceList [Count] > MaxPrice
                MaxPrice ← PriceList [Count]
        END FOR
        IF LENGTH CodeList = 3
        FinalPrice ← (MaxPrice / 100) * 140
        ELIF LENGTH CodeList = 2
                FinalPrice ← (MaxPrice/ 100) * 125
        ELSE
                FinalPrice ← PriceList[1]

ELIF InsuranceCompnay = “AVIA”
        MaxPrice ← 0
        MidPrice ← 0
        LowPrice ← 0
        FOR Count IN RANGE  LENGTH(CodeList)
                IF PriceList [Count] > MaxPrice
                        MaxPrice ← PriceList [Count]
                ELIF PriceList [Count] < MaxPrice AND PriceList [Count] > LowPrice
                        MidPrice ← PriceList [Count]
                ELSE
                        LowPrice ← PriceList [Count]
        END FOR
        IF LENGTH CodeList = 1
                FinalPrice ← MaxPrice
        ELIF LENGTH CodeList = 2
                FinalPrice ← MaxPrice + (MidPrice * 0.5)
        ELSE
                FinalPrice ← MaxPrice + (MidPrice * 0.5) + (LowPrice * 0.25)

ELSE
        MaxPrice ← 0
        LowPrice ← 0
        FOR Count IN RANGE  LENGTH(CodeList)
                IF PriceList [Count] > MaxPrice
                        MaxPrice ← PriceList [Count]
                ELES
                        LowPrice ← PriceList [Count]
        END FOR
        IF LENGTH CodeList = 1
                FinalPrice ← MaxPrice
        ELIF LENGTH CodeList = 2
                FinalPrice ← MaxPrice + (LowPrice * 0.5)
```

4.3. Object Diagrams



4.4. Class Definitions

Key:



5. Prototyping

5.1. Consideration of impact on design and development

In my system there are some new features which I do not have experience in programming. These functions would benefit from being prototyped before fully designed in order for them to meet the programs specifications.

These areas would be the following:

- Formatting on screen data into a report and sending the report to the network printer.
- Accessing and updating a database hosted on a server from multiple computers, as I have no knowledge of python network protocols.
- Encrypting data within a SQL database as I have never used this feature of SQL.
- Creating PyQt features such as a drop down tool bar for navigation.

Below is an example of one of my prototypes. The code shows how to create a drop down tool bar using PyQt in python.

```
#menu bar
self.menuBar = QMenuBar()
    #single tab
self.tab1 = self.menuBar.addMenu("tab1")
    #drop down tab
self.tab2 = self.menuBar.addMenu("tab2")
self.tab2_1 = self.tab2.addAction("tab2_1")
#set menu widget
self.setMenuWidget(self.menuBar)
#connections
self.tab1.triggered.connect(self.menuTest1)
self.tab2.triggered.connect(self.menuTest2)
self.tab2_1.triggered.connect(self.menuTest2_1)
```

6. Definition of Data Requirements

6.1. Identification of all data input items

The system will require multiple data to be input by the user, these data items are listed below:

- Username
- Password
- Title
- First Name
- Last Name
- Address Line 1
- Address Line 2
- Town
- Post Code
- Date of Birth
- Gender
- Telephone Number
- Email
- Consultant Title
- Consultant First Name
- Consultant Last Name
- Consultant Email
- Code
- Incompatible Code
- Surgeon Price
- Anaesthetist Price

6.2. Identification of all data output items

- Title
- First Name
- Last Name
- Address Line 1
- Address Line 2
- Town
- Post Code
- Date of Birth
- Gender
- Telephone Number
- Email
- Consultant Title
- Consultant First Name
- Consultant Last Name
- Consultant Email
- Code
- Incompatible Code
- Surgeon Price
- Anaesthetist Price
- Total Price
- Insurance Company
- Date
- User Email
- User Access

6.3. Explanation of how data output items are generated

Output	Generated
Title	Input by user, retrieve from database
First Name	Input by user, retrieve from database
Last Name	Input by user, retrieve from database
Address Line 1	Input by user, retrieve from database
Address Line 2	Input by user, retrieve from database
Town	Input by user, retrieve from database
Post Code	Input by user, retrieve from database
Date of Birth	Input by user, retrieve from database
Gender	Input by user, retrieve from database
Telephone Number	Input by user, retrieve from database
Email	Input by user, retrieve from database
Consultant Title	Input by user, retrieve from database
Consultant First Name	Input by user, retrieve from database
Consultant Last Name	Input by user, retrieve from database
Consultant Email	Input by user, retrieve from database
Code	Retrieved from database (Can be input by user)
Incompatible Code	Retrieved from database (Can be input by user)
Surgeon Price	Calculated using calculation method dependent on Code and Insurance Company.
Anaesthetist Price	Calculated using calculation method dependent on Code and Insurance Company.
Total Price	Surgeon Price + Anaesthetist Price
Insurance Company	Retrieved from database
Date	Retrieved by system clock
Username	Retrieved from database
User Email	Retrieved from database
User Access	Retrieved from database

6.4. Data Dictionary

Variable Name	Data Type	Length	Validation	Example Data	Comment
UserName	String	15 chars	Look Up	Niki_Cox	Required for user log in
Password	String	15 chars	Look Up	Password1	Required for user log in
Title	String	4	Presence	Mr	Relates to a Patient
FirstName	String	50	Presence	Jack	Relates to a Patient
LastName	String	50	Presence	Cox	Relates to a Patient
AddressLine 1	String	50	Presence	24 Mill End Road	Relates to a Patient

AddressLine 2	String	50		Cherry Hinton	Relates to a Patient
Town	String	30	Presence	Cambridge	Relates to a Patient
PostCode	String	8	Presence / Format	CB1 9KP	Relates to a Patient
DateOfBirth	Date	3 bytes	Presence / Format	1994/12/4	Relates to a Patient
Gender	String	6	Presence	Male	Relates to a Patient
TelephoneNumber	String	13	Presence	01234567891	Relates to a Patient
Email	String	50	Presence	jcox@ hotmail.com	Relates to a Patient
ConsultantTitle	String	4	Presence	Mr	Relates to Consultant
ConsultantFirstName	String	50	Presence	Bob	Relates to Consultant
ConsultantLastName	String	50	Presence	Johnson	Relates to Consultant
ConsultantEmail	String	50	Presence	BJ@ gmail.com	Relates to Consultant
Code	String		Format	[W2489, W7548, T9283]	Operation code relating to a procedure
IncompatibleCodes	String		Format	[W2489, W7548, T9283]	Operation code not allowed to be charge with the initial code
SurgeonPrice	Integer	3 bytes	-	984	
AnaesthetistPrice	Integer	3 bytes	-	257	
TotalPrice	Integer	4 bytes	-	1241	
InsuranceCompany	String	20	-	BUPA	
Date	Date	3 bytes	-	1994/12/4	Used in invoices
UserEmail	String	50	Presence	ncox@ office.co.uk	
UserAccess	String	5	Look Up	Admin/User	

6.5. Identification of appropriate storage media

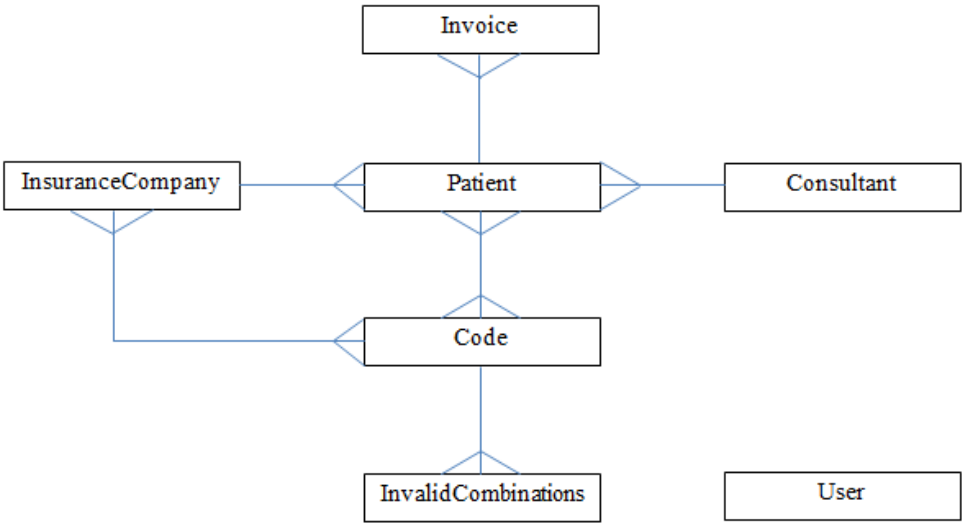
The appropriate storage which the system will use will be the server's hard drive. This is because the device has a large enough open memory space for my database and has an automatic back up system. This backup is run automatically every night and will ensure that a minimal amount of data will be lost if anything were to happen to the database.

7. Database Design

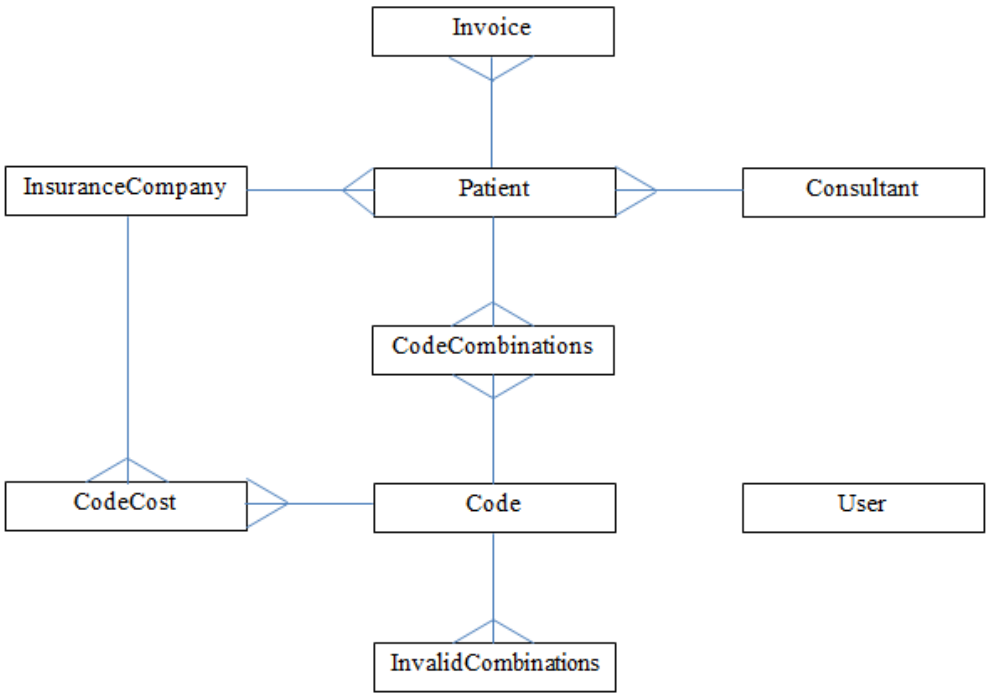
7.1. Normalisation

7.1.1. ER Diagrams

Below is my first Entity Relationship diagram which shows the relationships in my database.



This second Entity Relationship diagram shows the breakdown of the many to many relationships from the first diagram by introducing the 2 new entities: CodeCost and InvalidCombinations.



7.1.2. UNF to 3NF

UNF	1NF	2NF	3NF
Patient ID	Patient ID	Patient ID	Patient ID
Title	Title	Title	Title
First Name	First Name	First Name	First Name
Last Name	Last Name	Last Name	Last Name
Address Line 1	Address Line 1	Address Line 1	Address Line 1
Address Line 2	Address Line 2	Address Line 2	Address Line 2
Town	Town	Town	Town
Post Code	Post Code	Post Code	Post Code
Date of Birth	Date of Birth	Date of Birth	Date of Birth
Gender	Gender	Gender	Gender
Telephone Number	Telephone Number	Telephone Number	Telephone Number
Email	Email	Email	Email
Date of Operation	Date of Operation	Date of Operation	Date of Operation
Insurance Company	Insurance Company	Insurance Company	Consultant ID*
Consultant Title	Consultant Title	Consultant Title	Insurance Company ID*
Consultant First Name	Consultant First Name	Consultant First Name	
Consultant Last Name	Consultant Last Name	Consultant Last Name	Consultant ID
Consultant Email	Consultant Email	Consultant Email	Consultant Title
Code	Surgeon Price	Surgeon Price	Consultant First Name
Incompatible Code	Anaesthetists Price	Anaesthetists Price	Consultant Last Name
Surgeon Price	Invoice Date	Invoice Date	Consultant Email
Anaesthetists Price	Tax Rate	Tax Rate	
Invoice Date			Insurance Company ID
Tax Rate	Code ID	Code ID	Insurance Company Name
	Patient ID	Patient ID	
	Code	Code	Invoice ID
	Incompatible Code		Invoice Date
		Code ID	Tax Rate
		Incompatible Code	Patient ID*
			Code ID
			Code
			Incompatible ID
			Code ID*
			Code ID*

						Cost ID
						Insurance Company ID*
						Code ID*
						Surgeon Price
						Anaesthetist Price
						Code Combination ID
						Patient ID*
						Code ID*
Username		User ID		User ID		User ID
Password		Username		Username		Username
User Access		Password		Password		Password
User Email		User Access		User Access		User Access
		User Email		User Email		User Email

** = Foreign Key,* = *Primary Key*

As you can see from the normalisation table and entity relationship diagram there will be 9 entities in my database with a total of 46 attributes. I chose to show the final entity, "User", separately at the bottom of the normalisation as the attributes it contains are not related in any way to anything from other entities.

Below are the entity descriptions showing the distribution of attributes between the 9 entities.

Invoice(InvoiceID, InvoiceDate, TaxRate, *PatientID*)

InsuranceCompany(InsuranceCompanyID, InsuranceCompanyName)

Patient(PatientID, Title, FirstName, LastName, AddressLine1, AddressLine2, Town, PostCode, DateOfBirth, Gender, TelephoneNumber, Email, DateOfOperation, *ConsulantID*, *InsuranceCompayID*)

Consultant(ConsultantID, ConsultantTitle, ConsultantFirstName, ConsultantLastName, ConsultantEamil)

CodeCombinations(CodeCombinationID, *PatientID*, *CodeID*)

CodeCost(CostID, *InsuranceCompanyID*, *CodeID*, SurgeonPrice, AnaesthetistPrice)

Code(CodeID, Code)

User(UserID, Username, Password, UserAccess, UserEmail)

InvalidCombination(IncompatibleID, *CodeID*, *CodeID*)

7.2. SQL Queries

This query demonstrates how the system will retrieve incompatible codes related to a given code.

“””SELECT *

FROM InvalidCombinations

WHERE CodeID = (‘{0}’) OR CodeID = (‘{1}’) OR CodeID = (‘{2}’)”””.format(Code1, Code2, Code3)

The following SQL query shows how a patient will be added to the database.

“””INSERT INTO Patient(Title, FirstName, LastName, AddressLine1, AddressLine2, Town, PostCode, DateOfBirth, Gender, TelephoneNumber, Email, ConsultantID, InsuranceCompany, InvoiceID) VALUES

(‘{0}’, ‘{1}’, ‘{2}’, ‘{3}’, ‘{4}’, ‘{5}’, ‘{6}’, ‘{7}’, ‘{8}’, ‘{9}’, ‘{10}’, ‘{11}’, ‘{12}’, ‘{13}’)”””.format(Title, FirstName, LastName, AddressLine1, AddressLine2, Town, PostCode, DateOfBirth, Gender, TelephoneNumber, Email, ConsultantID, InsuranceCompany, InvoiceID)

The system will also need to update some existing values, an example of the SQL for this is shown below.

“””UPDATE Patient

SET FirstName = ‘Adam’

WHERE PatientID = ‘2’ “””

Some records will also need to be removed from the database; this is shown below in SQL.

“””DELETE FROM Code

WHERE CodeID = ‘1’ ”””

Some of the SQL statements for my system use algorithms and loops to calculate values for passing into the database. For example the code below shows a SQL statement within a loop using a calculated value from the IC assignment.

```
def add_code_cost(self,ID,costs):
    for count in range(0,13,2):
        IC = (count+2)//2
        sql = """insert into codeCost
            (InsuranceCompanyID,CodeID,SurgeonPrice,AnaesthetistPrice)
            values
            ('{0}','{1}','{2}','{3}')""".format(IC,ID,costs[count],costs[count+1])
```

**Where the parameters would be: ID-Any integer, costs-A list of 14 integers*

8. Security and Integrity of the System and Data

8.1. Security and Integrity of Data

As the data in the systems database will be confidential patient information there will need to be some level of encryption to ensure the data is protected in compliance with the Data Protection Act 1998.

To ensure the integrity of data is accurate there will be validation in place wherever there is a user input. This will ensure that all data of the same type is consistent and has the correct format.

8.2. System Security

As the systems database will hold confidential information regarding patients, and any procedures they have had, it will give any user access to this private information. Therefore access restrictions will be put in place in the form of user accounts. This will restrict normal users from altering important parts of the database, which will only be available to admins.

This will give the office staff alone access to the system as the Data Protection Act 1998 states that stored personal information must be secure and only used for an appropriate purpose.

9. Validation

In my data dictionary (Section 6.4) I identified the validation I am going to use when the system is faced with user inputs. These 3 types of validation are Lookup, Presence and Format.

Lookup

The following 3 attributes have a lookup validation method, Username, Password and UserAccess. This is because these attributes can only have a specific set of pre-defined values. For example, whenever a user inputs a username and password at the log in screen these inputs need to be checked for existence within the database in order to allow the user to be logged in. Similarly UserAccess can only be 2 different values, user or admin, so when the access level of an account is set it must be one of these 2 options. The look up validation ensures this by searching for the user input value in the values are retuning whether the value is existing or non-existing.

Presence

Another form of validation the system will use is presence checks. This type of validation is important as this attributes are vital to the construction of a record within the database. The following attributes from the patient entity require this presence validation during the creation of a record: Title, FirstName, LastName, AddressLine1, Town, PostCode, DateOfBirth, Gender, TelephoneNumber, Email.

Likewise the 4 attributes ConsultantTitle, ConsultantFirstName, ConsultantLastName and ConsultantEmail, are all required to produce a new consultant record. Therefore these attributes also have the presence validation.

Format

Code, IncompatibleCode, DateOfBirth and PostCode are special attributes which require a specific input format, for example a code should be one letter followed by four numbers, giving A1234. As an IncompatibleCode is a Code it has the same format, however DateOfBirth should have the format YYYY/MM/DD and PostCode AB1 2CD or AB12 3CD. This is why I will use format validation on these attributes as it is important that they are all input in the same format to prevent errors in the system.

10. Testing

10.1. Outline Plan

Test Set	Purpose
1.	Testing all navigational features and function buttons within the user interface, in order to ensure all connections are correct.
2.	Test all user inputs to ensure that validation works appropriately. This prevents unreliable data from being input to the system.
3.	Test code calculation method is applied correctly and calculations are accurate.
4.	Test data updates and retrievals to and from the database to ensure proper functionality.

10.1.1. Identification and explanation of suitable test strategies

Test	Purpose/Description	Test Data (Input)	Expected Result	Test Result
1.1.	Check that the "Log In" button from the "Log In" window navigates to the correct page.	"Log In" Button Clicked.	"Cambridge Medical Practice Billing System" Window displayed.	
1.2.	Check that the "Operations" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	"Operations" Button Clicked.	"Operations" Window displayed.	
1.3.	Check that the "Codes" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	"Codes" Button Clicked.	"Codes" Window displayed.	
1.4.	Check that the "Users" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	"Users" Button Clicked.	"Users" Window displayed.	
1.5.	Check that the "Consultants" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	"Consultants" Button Clicked.	"Consultants" Window displayed.	

1.6.	Check that the "Patients" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	"Patients" Button Clicked.	"Patients" Window displayed.	
1.7.	Check that the "Operations" tab from the tool bar displayed in multiple windows navigates to the correct page.	"Operations" tab Clicked.	"Operations" Window displayed.	
1.8.	Check that the "Codes" tab from the tool bar displayed in multiple windows navigates to the correct page.	"Codes" tab Clicked.	"Codes" Window displayed.	
1.9.	Check that the "Users" tab from the tool bar displayed in multiple windows navigates to the correct page.	"Users" tab Clicked.	"Users" Window displayed.	
1.10.	Check that the "Consultants" tab from the tool bar displayed in multiple windows navigates to the correct page.	"Consultants" tab Clicked.	"Consultants" Window displayed.	
1.11.	Check that the "Patients" tab from the tool bar displayed in multiple windows navigates to the correct page.	"Patients" tab Clicked.	"Patients" Window displayed.	
1.12.	Check that the "Search" button from the "Operations" window navigates to the correct page.	"Search" button Clicked.	"Operations – Search Results" Window displayed.	
1.13.	Check that the "Add" button from the "Codes" window navigates to the correct page.	"Add" button Clicked.	"Codes – New Code" Window displayed.	
1.14.	Check that the "Search" button from the "Codes" window	"Search" button Clicked.	"Codes – Search Results" Window displayed.	

	navigates to the correct page.			
1.15.	Check that the "Calculator" button from the "Codes" window navigates to the correct page.	"Calculator" button Clicked.	"Codes – Calculator" Window displayed.	
1.16.	Check that the "Add Patient" button from the "Patients" window navigates to the correct page.	"Add Patient" button Clicked.	"Patients – New Patient" Window displayed.	
1.17.	Check that the "Find Patient" button from the "Patients" window navigates to the correct page.	"Find Patient" button Clicked.	"Patients - Search" Window displayed.	
1.18.	Check that the "Search" button from the "Patients - Search" window navigates to the correct page.	"Search" button Clicked.	"Patients – Search Results" Window displayed.	
1.19.	Check that the "Add Consultant" button from the "Consultants" window navigates to the correct page.	"Add Consultant" button Clicked	"Consultants – New Consultant" Window displayed.	
1.20.	Check that the "Find Consultant" button from the "Consultants" window navigates to the correct page.	"Find Consultant" button Clicked	"Consultants – Search" Window displayed.	
1.21.	Check that the "Search" button from the "Consultants - Search" window navigates to the correct page.	"Search" button Clicked	"Consultants – Search Results" Window displayed.	
1.22.	Check that the "Add User" button from the "Users" window navigates to the correct page.	"Add User" button Clicked	"User – New User" Window displayed.	
1.23.	Check that the "Find User" button from the "User" window navigates to the correct page.	"Find User" button Clicked	"User – Search" Window displayed.	

1.24.	Check that the "Search" button from the "User - Search" window navigates to the correct page.	"Search" button Clicked	"User – Search Results" Window displayed.	
2.1.	Check that the "Consultant" input box from the "Operations" window only accepts valid data.	0 characters	Error message displayed	
		51 characters	Error message displayed	
		10 characters	Accepted	
2.2.	Check that the "Date" input box from the "Operations" window only accepts valid data.	'1-2-2012'	Error message displayed	
		'2012-2-1'	Accepted	
		'201221'	Error message displayed	
		'a'	Error message displayed	
2.3.	Check that the "Code" input box from the "Codes" window only accepts valid data.	'A1234'	Accepted	
		'a1234'	Error message displayed	
		'a123'	Error message displayed	
		'12345'	Error message displayed	
2.4.	Check that the "Code Name" input box from the "Codes – New Code" window only accepts valid data.	'A1234'	Accepted	
		'a1234'	Error message displayed	
		'a123'	Error message displayed	
		'12345'	Error message displayed	
2.5.	Check that the "Incompatible Codes" input box from the "Codes – New Code" window only accepts valid data.	'A1234/B1234'	Accepted	
		'A'	Error message displayed	
		-1	Error message displayed	
2.6.	Check that the Insurance Company costs, input boxes from the "Codes – New Code" window only accepts valid data.	'a'	Error message displayed	
		-1	Error message displayed	
		234	Accepted	
2.7.	Check that the "Code" input box from the "Codes – New Code"	'A1234'	Accepted	
		'a1234'	Error message displayed	

	window only accepts valid data.	‘a123’	Error message displayed	
		‘12345’	Error message displayed	
2.8.	Check that the “Title” input box from the “Patients – New Patient” window only accepts valid data.	‘Mr’	Accepted	
		‘mr’	Error message displayed	
		-1	Error message displayed	
2.9.	Check that the “First Name” input box from the “Patients – New Patient” window only accepts valid data	‘Jack’	Accepted	
		‘jack’	Error message displayed	
		-1	Error message displayed	
2.10.	Check that the “Second Name” input box from the “Patients – New Patient” window only accepts valid data	‘Cox’	Accepted	
		‘cox’	Error message displayed	
		-1	Error message displayed	
2.11.	Check that the “Address Line 1” input box from the “Patients – New Patient” window only accepts valid data	’45 Hello Road’	Accepted	
		‘’	Error message displayed	
2.12.	Check that the “Address Line 2” input box from the “Patients – New Patient” window only accepts valid data	’45 Hello Road’	Accepted	
		‘’	Accepted	
2.13.	Check that the “Town” input box from the “Patients – New Patient” window only accepts valid data	‘Cambridge’	Accepted	
		‘cambridge’	Error message displayed	
2.14.	Check that the “Post Code” input box from the “Patients – New Patient” window only accepts valid data	‘CB8 8HN’	Accepted	
		‘CB22 8HN’	Accepted	
		‘CB88HN’	Error message displayed	
		-1	Error message displayed	
		‘a’	Error message displayed	

2.15.	Check that the "Date Of Birth" input box from the "Patients – New Patient" window only accepts valid data	'1-2-2012'	Error message displayed	
		'2012-2-1'	Accepted	
		'201221'	Error message displayed	
		'a'	Error message displayed	
2.16.	Check that the "Gender" input box from the "Patients – New Patient" window only accepts valid data	'Male'	Accepted	
		'Female'	Accepted	
		'Alien'	Error message displayed	
		-1	Error message displayed	
2.17.	Check that the "Email" input box from the "Patients – New Patient" window only accepts valid data	'wwwbobgmailcom'	Error message displayed	
		'www.bob@gmail.com'	Accepted	
		-1	Error message displayed	
2.18.	Check that the "Telephone Number" input box from the "Patients – New Patient" window only accepts valid data	'01234-567899'	Accepted	
		'01234 567899'	Accepted	
		a	Error message displayed	
		'01234 56789'	Error message displayed	
2.19.	Check that the "Insurance" input box from the "Patients – New Patient" window only accepts valid data	'BUPA'	Accepted	
		'abc'	Error message displayed	
		-1	Error message displayed	
2.20.	Check that the "Codes" input box from the "Patients – New Patient" window only accepts valid data	'A1234'	Accepted	
		'a1234'	Error message displayed	
		'a123'	Error message displayed	
		'12345'	Error message displayed	
2.21.	Check that the "Date Of Operation" input box from the "Patients – New Patient" window only accepts valid data	'1-2-2012'	Error message displayed	
		'2012-2-1'	Accepted	
		'201221'	Error message displayed	
		'a'	Error message displayed	
2.22.	Check that the "Consultant: First	'Jack'	Accepted	

	Name" input box from the "Patients – New Patient" window only accepts valid data	'jack'	Error message displayed	
		-1	Error message displayed	
2.23.	Check that the "Consultant: Last Name" input box from the "Patients – New Patient" window only accepts valid data	'Cox'	Accepted	
		'cox'	Error message displayed	
		-1	Error message displayed	
2.24.	Check that the "Patient ID" input box from the "Patients - Search" window only accepts valid data	-1	Error message displayed	
		0	Error message displayed	
		1	Accepted	
		'a'	Error message displayed	
2.25.	Check that the "First Name" input box from the "Patients - Search" window only accepts valid data	'Jack'	Accepted	
		'jack'	Error message displayed	
		-1	Error message displayed	
2.26.	Check that the "Last Name" input box from the "Patients - Search" window only accepts valid data	'Cox'	Accepted	
		'cox'	Error message displayed	
		-1	Error message displayed	
2.27.	Check that the "Title" input box from the "Consultants – New Consultant" window only accepts valid data	'Mr'	Accepted	
		'mr'	Error message displayed	
		-1	Error message displayed	
2.28.	Check that the "First Name" input box from the "Consultants – New Consultant" window only accepts valid data	'Jack'	Accepted	
		'jack'	Error message displayed	
		-1	Error message displayed	
2.29.	Check that the "Last Name" input box from the "Consultants – New Consultant" window only accepts valid data	'Cox'	Accepted	
		'cox'	Error message displayed	
		-1	Error message displayed	
2.30.	Check that the "Email" input box from the "Consultants	'wwwbobgmailcom'	Error message displayed	

	– New Consultant" window only accepts valid data	'www.bob@gmail.com'	Accepted	
		-1	Error message displayed	
2.31.	Check that the "Consultant ID" input box from the "Consultants – Search" window only accepts valid data	-1	Error message displayed	
		0	Error message displayed	
		1	Accepted	
		'a'	Error message displayed	
2.32	Check that the "First Name" input box from the "Consultants – Search" window only accepts valid data	'Jack'	Accepted	
		'jack'	Error message displayed	
		-1	Error message displayed	
2.33.	Check that the "Last Name" input box from the "Consultants – Search" window only accepts valid data	'Cox'	Accepted	
		'cox'	Error message displayed	
		-1	Error message displayed	
2.34.	Check that the "Username" input box from the "User – New User" window only accepts valid data	'user1'	Accepted	
		'User'	Accepted	
		'U$er'	Error message displayed	
2.35.	Check that the "Password" input box from the "User – New User" window only accepts valid data	'pass'	Error message displayed	
		'password'	Error message displayed	
		'password1'	Accepted	
2.36.	Check that the "Email" input box from the "User – New User" window only accepts valid data	'wwwbobgmailcom'	Error message displayed	
		'www.bob@gmail.com'	Accepted	
		-1	Error message displayed	
2.37.	Check that the "Access" input box from the "User – New User" window only accepts valid data	'User'	Accepted	
		'Admin'	Accepted	
		'Hello'	Error message displayed	
		-1	Error message displayed	
2.38.	Check that the "User ID" input box from the "User – Search"	-1	Error message displayed	
		0	Error message	

	window only accepts valid data		displayed	
		1	Accepted	
		‘a’	Error message displayed	
2.39.	Check that the “Username” input box from the “User – Search” window only accepts valid data	‘user1’	Accepted	
		‘User’	Accepted	
		‘U$er’	Error message displayed	
3.1.	Check that the ‘BUPA’ calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1120 Anaesthetist: £560	
		Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234	Surgeon: £1120 Anaesthetist: £560	
3.2.	Check that the ‘WPA’ calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1120 Anaesthetist: £560	
		Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234	Surgeon: £1120 Anaesthetist: £560	
3.3.	Check that the ‘PPP’ calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1100 Anaesthetist: £550	
		Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234	Surgeon: £1100 Anaesthetist: £550	
3.4.	Check that the ‘AVIVA’ calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1200 Anaesthetist: £600	
		Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234	Surgeon: £1100 Anaesthetist: £550	
3.5.	Check that the ‘PruHealth’ calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1120 Anaesthetist: £560	
		Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234	Surgeon: £1120 Anaesthetist: £560	
3.6.	Check that the ‘Cigna’ calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1120 Anaesthetist: £560	
		Codes : A1234, B1234, D1234	Surgeon: £1120 Anaesthetist:	

		Incompatible: A1234 and D1234	£560	
3.7.	Check that the 'SimplyHealth' calculation method returns the correct values	Codes : A1234, B1234, C1234	Surgeon: £1120 Anaesthetist: £560	
		Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234	Surgeon: £1120 Anaesthetist: £560	
4.1.	Check "Add Code" function works appropriately	Random but valid data	Matching data to appear in correct entity	
4.2.	Check "Update Code" function works appropriately	Code Name used in 4.1.	Returns data input in 4.1.	
4.3.	Check "Delete Code" function works appropriately	Code Name used in 4.1.	Removes code and corresponding data	
4.4.	Check "Add Patient" function works appropriately	Random but valid data	Matching data to appear in correct entity	
4.5.	Check "Update Patient" function works appropriately	Patient used in 4.4.	Returns data input in 4.4.	
4.6.	Check "Delete Patient" function works appropriately	Patient used in 4.4.	Removes Patient and corresponding data	
4.7.	Check "Add Consultant" function works appropriately	Random but valid data	Matching data to appear in correct entity	
4.8.	Check "Update Consultant" function works appropriately	Consultant used in 4.7.	Returns data input in 4.7.	
4.9.	Check "Delete Consultant" function works appropriately	Consultant used in 4.7.	Removes Consultant and corresponding data	
4.10.	Check "Add User" function works appropriately	Random but valid data	Matching data to appear in correct entity	
4.11.	Check "Update User" function works appropriately	User used in 4.10.	Returns data input in 4.10.	
4.12.	Check "Delete User" function works appropriately	User used in 4.10.	Removes User and corresponding data	

Testing

1. Test Plan

1.1. Detailed Test Plan

All tests marked with a red 'X' have been removed from the original test plan due to changes within the system made during the implementation phase.

Test Set	Purpose	Testing Strategy
1.	Testing all navigational features and function buttons within the user interface, in order to ensure all connections are correct.	White box testing
2.	Test all user inputs to ensure that validation works appropriately. This prevents unreliable data from being input to the system.	Functional testing
3.	Test code calculation method is applied correctly and calculations are accurate.	Black box testing
4.	Test data updates and retrievals to and from the database to ensure proper functionality.	Functional testing
5.	Test log in functionality to ensure access is only granted to existing users, and appropriate access level is enforced.	Security testing
6.	Test all search methods to ensure that non-existing data cause an error message and does not lead to a blank record being output.	Functional testing

Test	Purpose/Description	Test Data Type	Expected Result	Reference
1.1. 	Check that the "Log In" button from the "Log In" window navigates to the correct page.	Normal	"Cambridge Medical Practice Billing System" Window displayed.	
1.2.	Check that the "Operations" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	Normal	"Operations" Window displayed.	Figure 1 – Test 1.2. page 119
1.3.	Check that the "Codes" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	Normal	"Codes" Window displayed.	
1.4.	Check that the "Users" button from the	Normal	"Users" Window displayed.	

	"Cambridge Medical Practice Billing System" window navigates to the correct page.			
1.5.	Check that the "Consultants" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	Normal	"Consultants" Window displayed.	
1.6.	Check that the "Patients" button from the "Cambridge Medical Practice Billing System" window navigates to the correct page.	Normal	"Patients" Window displayed.	
1.7. 	Check that the "Operations" tab from the tool bar displayed in multiple windows navigates to the correct page.	Normal	"Operations" Window displayed.	
1.8. 	Check that the "Codes" tab from the tool bar displayed in multiple windows navigates to the correct page.	Normal	"Codes" Window displayed.	
1.9. 	Check that the "Users" tab from the tool bar displayed in multiple windows navigates to the correct page.	Normal	"Users" Window displayed.	
1.10. 	Check that the "Consultants" tab from the tool bar displayed in multiple windows navigates to the correct page.	Normal	"Consultants" Window displayed.	
1.11. 	Check that the "Patients" tab from the tool bar displayed in multiple windows navigates to the correct page.	Normal	"Patients" Window displayed.	
1.12.	Check that the "Search" button from the "Operations" window	Normal	"Operations – Search Results" Window displayed.	

	navigates to the correct page.			
1.13.	Check that the "Add" button from the "Codes" window navigates to the correct page.	Normal	"Codes – New Code" Window displayed.	Figure 2 - Test 1.13. page 119
1.14.	Check that the "Search" button from the "Codes" window navigates to the correct page.	Normal	"Codes – Search Results" Window displayed.	
1.15.	Check that the "Calculator" button from the "Codes" window navigates to the correct page.	Normal	"Codes – Calculator" Window displayed.	
1.16.	Check that the "Add Patient" button from the "Patients" window navigates to the correct page.	Normal	"Patients – New Patient" Window displayed.	
1.17.	Check that the "Find Patient" button from the "Patients" window navigates to the correct page.	Normal	"Patients - Search" Window displayed.	Figure 3 - Test 1.17. page 120
1.18.	Check that the "Search" button from the "Patients - Search" window navigates to the correct page.	Normal	"Patients – Search Results" Window displayed.	
1.19.	Check that the "Add Consultant" button from the "Consultants" window navigates to the correct page.	Normal	"Consultants – New Consultant" Window displayed.	
1.20.	Check that the "Find Consultant" button from the "Consultants" window navigates to the correct page.	Normal	"Consultants – Search" Window displayed.	
1.21.	Check that the "Search" button from the "Consultants - Search" window navigates to the correct page.	Normal	"Consultants – Search Results" Window displayed.	
1.22.	Check that the "Add User" button from the "Users" window	Normal	"User – New User" Window displayed.	

	navigates to the correct page.			
1.23.	Check that the "Find User" button from the "User" window navigates to the correct page.	Normal	"User – Search" Window displayed.	
1.24.	Check that the "Search" button from the "User - Search" window navigates to the correct page.	Normal	"User – Search Results" Window displayed.	
2.1. 	Check that the "Consultant" input box from the "Operations" window only accepts valid data.	Erroneous	Error message displayed	
		Extreme/Boundary	Error message displayed	
		Normal	Accepted	
2.2.	Check that the "Date" input box from the "Operations" window only accepts valid data.	Erroneous	Error message displayed	Figure 6 - Test 2.2. page 121
		Normal	Accepted	
2.3.	Check that the "Code" input box from the "Codes" window only accepts valid data.	Normal	Accepted	
		Erroneous	Error message displayed	
2.4.	Check that the "Code Name" input box from the "Codes – Add Code" window only accepts valid data.	Normal	Accepted	
		Erroneous	Error message displayed	
2.5.	Check that the "Incompatible Codes" input box from the "Codes – Add Code" window only accepts valid data.	Normal	Accepted	
		Erroneous	Error message displayed	
2.6.	Check that the Insurance Company costs, input boxes from the "Codes – Add Code" window only accepts valid data.	Erroneous	Error message displayed	
		Extreme/Boundary	Error message displayed	
		Normal	Accepted	
2.7.	Check that the "Code" input box from the "Codes – Add Code" window only accepts valid data.	Normal	Accepted	
		Erroneous	Error message displayed	
2.8.	Check that the "Title"	Normal	Accepted	

	input box from the "Patients – Add Patient" window only accepts valid data.	Erroneous	Error message displayed	
2.9.	Check that the "First Name" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.10.	Check that the "Second Name" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.11.	Check that the "Address Line 1" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.12.	Check that the "Address Line 2" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	Figure 7 – Test 2.12. page 122
		Erroneous	Accepted	
2.13.	Check that the "Town" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.14.	Check that the "Post Code" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.15.	Check that the "Date Of Birth" input box from the "Patients – Add Patient" window only accepts valid data	Erroneous	Error message displayed	
		Normal	Accepted	
2.16. 	Check that the "Gender" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.17.	Check that the "Email" input box from the "Patients – Add Patient" window only accepts valid data	Erroneous	Error message displayed	
		Normal	Accepted	

2.18.	Check that the "Telephone Number" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
		Extreme/Boundary	Error message displayed	
2.19. 	Check that the "Insurance" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.20.	Check that the "Codes" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.21.	Check that the "Date Of Operation" input box from the "Patients – Add Patient" window only accepts valid data	Erroneous	Error message displayed	
		Normal	Accepted	
2.22. 	Check that the "Consultant: First Name" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.23. 	Check that the "Consultant: Last Name" input box from the "Patients – Add Patient" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.24.	Check that the "Patient ID" input box from the "Patients - Search" window only accepts valid data	Extreme/Boundary	Error message displayed	
		Erroneous	Error message displayed	
		Normal	Accepted	
2.25.	Check that the "First Name" input box from the "Patients - Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.26.	Check that the "Last Name" input box from the "Patients - Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.27.	Check that the "Title"	Normal	Accepted	

	input box from the "Consultants – Add Consultant" window only accepts valid data	Erroneous	Error message displayed	
2.28.	Check that the "First Name" input box from the "Consultants – Add Consultant" window only accepts valid data	Normal	Accepted	Figure 8 - Test 2.28. page 123
		Erroneous	Error message displayed	
2.29.	Check that the "Last Name" input box from the "Consultants – Add Consultant" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.30.	Check that the "Email" input box from the "Consultants – Add Consultant" window only accepts valid data	Erroneous	Error message displayed	
		Normal	Accepted	
2.31.	Check that the "Consultant ID" input box from the "Consultants – Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
		Extreme/Boundary	Error message displayed	
2.32	Check that the "First Name" input box from the "Consultants – Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.33.	Check that the "Last Name" input box from the "Consultants – Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.34.	Check that the "Username" input box from the "User – Add User" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.35.	Check that the "Password" input box from the "User – Add User" window only accepts valid data	Erroneous	Error message displayed	
		Normal	Accepted	
2.36.	Check that the "Email" input box from the	Erroneous	Error message displayed	Figure 9 - Test 2.36.

	"User – Add User" window only accepts valid data	Normal	Accepted	page 123
2.37. 	Check that the "Access" input box from the "User – Add User" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
2.38.	Check that the "User ID" input box from the "User – Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
		Extreme/Boundary	Error message displayed	
2.39.	Check that the "Username" input box from the "User – Search" window only accepts valid data	Normal	Accepted	
		Erroneous	Error message displayed	
3.1.	Check that the 'BUPA' calculation method returns the correct values	Normal	Prices calculated correctly	Figure 10 - Test 3.1. page 124
		Erroneous	Error message displayed	
3.2.	Check that the 'WPA' calculation method returns the correct values	Normal	Prices calculated correctly	
		Erroneous	Error message displayed	
3.3.	Check that the 'PPP' calculation method returns the correct values	Normal	Prices calculated correctly	Figure 11 - Test 3.3. page 125
		Erroneous	Error message displayed	
3.4.	Check that the 'AVIVA' calculation method returns the correct values	Normal	Prices calculated correctly	Figure 12 - Test 3.4. page 126
		Erroneous	Error message displayed	
3.5.	Check that the 'PruHealth' calculation method returns the correct values	Normal	Prices calculated correctly	
		Erroneous	Error message displayed	
3.6.	Check that the 'Cigna' calculation method returns the correct values	Normal	Prices calculated correctly	
		Erroneous	Error message displayed	
3.7.	Check that the 'SimplyHealth' calculation method returns the correct values	Normal	Prices calculated correctly	
		Erroneous	Error message displayed	

4.1.	Check "Add Code" function works appropriately	Normal	Data stored correctly	Figure 13 - Test 4.1. page 127
4.2.	Check "Update Code" function works appropriately	Normal	Data updated correctly	Figure 14 - Test 4.2. page 128
4.3.	Check "Delete Code" function works appropriately	Normal	Data deleted correctly	
4.4.	Check "Add Patient" function works appropriately	Normal	Data stored correctly	
4.5.	Check "Update Patient" function works appropriately	Normal	Data updated correctly	
4.6.	Check "Delete Patient" function works appropriately	Normal	Data deleted correctly	
4.7.	Check "Add Consultant" function works appropriately	Normal	Data stored correctly	Figure 15 - Test 4.7. page 128
4.8.	Check "Update Consultant" function works appropriately	Normal	Data updated correctly	
4.9.	Check "Delete Consultant" function works appropriately	Normal	Data deleted correctly	Figure 16 - Test 4.9. page 129
4.10.	Check "Add User" function works appropriately	Normal	Data stored correctly	
4.11.	Check "Update User" function works appropriately	Normal	Data updated correctly	
4.12.	Check "Delete User" function works appropriately	Normal	Data deleted correctly	

1.2. Changes from the original plan

All tests marked with a red 'X' have been removed from the original test plan due to changes within the system made during implementation. The table below states the reasons of removal.

Test	**Reason for removal**
1.1	The log in window opens on top of the main window which is already displayed.
1.7 – 1.11	Menu bar structure was changed so that the tabs do not directly link to system areas, instead only the drop down options do.
2.1, 2.8, 2.16, 2.19, 2.22,	Input boxes were replaced with combo boxes, which only allow

2.23, 2.27, 2.37	the user to select a valid input.
	Input boxes were replaced with spin boxes, which only allow the user to select a valid input.

Amendments to my system during the implementation phase also require me to add new test to my test plan. These tests are shown below along with a new test series which was also added to my test plan. The justification for addition of new tests can be found below the additional test plan.

Test Set	**Purpose**
5.	Test log in functionality to ensure access is only granted to existing users, and appropriate access level is enforced.
6.	Test all search methods to ensure that non-existing data cause an error message and does not lead to a blank record being output.

Test	**Purpose/Description**	**Test Data Type**	**Expected Result**	**Reference**
1.25.	Check that the "Find Operations" button from the "Operation" tab in menu bar window navigates to the correct page.	Normal	"Operations" Window displayed.	
1.26.	Check that the "Code Main" button from the "Code" tab in menu bar window navigates to the correct page.	Normal	"Code" Window displayed.	
1.27.	Check that the "Add Code" button from the "Code" tab in menu bar window navigates to the correct page.	Normal	"Code- Add Code" Window displayed.	
1.28.	Check that the "Code Calculator" button from the "Code" tab in menu bar window navigates to the correct page.	Normal	"Code- Calculator" Window displayed.	Figure 4 - Test 1.28. page 120
1.29.	Check that the "User Main" button from the "User" tab in menu bar window navigates to the correct page.	Normal	"User" Window displayed.	
1.30.	Check that the "Add User" button from the "User" tab in menu bar window navigates to the correct page.	Normal	"User- Add User" Window displayed.	

1.31.	Check that the “Find User” button from the “User” tab in menu bar window navigates to the correct page.	Normal	“User - Search” Window displayed.	
1.32.	Check that the “Consultant Main” button from the “Consultant” tab in menu bar window navigates to the correct page.	Normal	“Consultant” Window displayed.	
1.33.	Check that the “Add Consultant” button from the “Consultant” tab in menu bar window navigates to the correct page.	Normal	“Consultants- Add Consultant” Window displayed.	
1.34.	Check that the “Find Consultant” button from the “Consultant” tab in menu bar window navigates to the correct page.	Normal	“Consultants- Search” Window displayed.	
1.35.	Check that the “Patient Main” button from the “Patient” tab in menu bar window navigates to the correct page.	Normal	“Patient” Window displayed.	
1.36.	Check that the “Add Patient” button from the “Patient” tab in menu bar window navigates to the correct page.	Normal	“Patient- Add Patient” Window displayed.	
1.37.	Check that the “Find Patient” button from the “Patient” tab in menu bar window navigates to the correct page.	Normal	“Patient- Find Patient” Window displayed.	
1.38.	Check that the “Main Menu” button from the “System” tab in menu bar window navigates to the correct page.	Normal	“CMP Window” Window displayed.	
1.39.	Check that the “Log In” button from the “System” tab in menu bar window navigates to the correct page.	Normal	“Log In” Window displayed.	
1.40.	Check that the “Log Out” button from the “System” tab in menu bar window navigates to the correct page.	Normal	Confirmation message box displayed.	Figure 5 - Test 1.40. page 121

1.41.	Check that the "Exit" button from the "System" tab in menu bar window navigates to the correct page.	Normal	System closure.	
2.40.	Check that the "Username" input box from the "Log In" window only accepts valid data.	Normal	Accepted	
		Erroneous	Error message displayed	
2.41.	Check that the "Password" input box from the "Log In" window only accepts valid data.	Normal	Accepted	
		Erroneous	Error message displayed	
5.1.	Check that the "Log In" window rejects non-existent users.	Normal	Error message displayed	Figure 17 - Test 5.1. page 129
5.2.	Check that the "Log In" window rejects incorrect passwords.	Normal	Error message displayed	Figure 18 - Test 5.2. page 130
5.3.	Check that access is restricted to "Operations" area when user is not logged in.	Normal	Error message displayed	
5.4.	Check that access is restricted to "Codes" area when user is not logged in.	Normal	Error message displayed	Figure 19 - Test 5.4. page 130
5.5.	Check that access is restricted to "Users" area when user is not logged in.	Normal	Error message displayed	
5.6.	Check that access is restricted to "Patients" area when user is not logged in.	Normal	Error message displayed	
5.7.	Check that access is restricted to "Consultants" area when user is not logged in.	Normal	Error message displayed	
5.8.	Check that access is granted to "Operations" area when user has "User" level access	Normal	"Operation" Window displayed	
5.9.	Check that access is granted to "Codes" area when user has "User" level access	Normal	"Code" Window displayed	
5.10.	Check that access is restricted to "Users" area when user has "User" level access	Normal	Error message displayed	Figure 20 - Test 5.10. page 131
5.11.	Check that access is granted to "Patients" area when user has "User" level access	Normal	"Patient" Window displayed	

5.12.	Check that access is granted to “Consultants” area when user has “User” level access	Normal	“Consultant” Window displayed	
5.13.	Check that access is granted to “Operations” area when user has “Admin” level access	Normal	“Operation” Window displayed	
5.14.	Check that access is granted to “Codes” area when user has “Admin” level access	Normal	“Code” Window displayed	
5.15.	Check that access is restricted to “Users” area when user has “Admin” level access	Normal	“User” Window displayed	
5.16.	Check that access is granted to “Patients” area when user has “Admin” level access	Normal	“Patient” Window displayed	
5.17.	Check that access is granted to “Consultants” area when user has “Admin” level access	Normal	“Consultant” Window displayed	Figure 21 - Test 5.17. page 132
6.1.	Check that the “Operation” search functionality requires a search method to be selected.	Erroneous	Error message displayed	Figure 22 - Test 6.1 page 133
6.2.	Check that the “Operation” search functionality requires an existing data input.	Normal	Error message displayed	
6.3.	Check that the “Code” search functionality requires an existing data input.	Normal	Error message displayed	
6.4.	Check that the “User” search functionality requires a search method to be selected.	Erroneous	Error message displayed	
6.5.	Check that the “User” search functionality requires an existing data input.	Normal	Error message displayed	Figure 23 - Test 6.5. page 134
6.7.	Check that the “Consultant” search functionality requires a search method to be selected.	Erroneous	Error message displayed	
6.8.	Check that the “Consultant” search functionality requires an existing data input.	Normal	Error message displayed	
6.9.	Check that the “Patient” search functionality requires a search method to be selected.	Erroneous	Error message displayed	

6.10.	Check that the “Patient” search functionality requires an existing data input.	Normal	Error message displayed	Figure 24 - Test 6.10. page 134

This table states the reasons for addition of the tests declared above.

Test	**Reason for addition**
1.25 – 1.41	The structure of the menu bar was altered leading in the removal of existing menu bar tests. Therefor these are the replacement tests.
2.40, 2.41	Required but missing from the original test plan.
5.1 – 5.17	Required but missing from the original test plan.
6.1 – 6.10	Required but missing from the original test plan.

2. Test Data

2.1. Test Data

Inconsistent numbering is due to the removal of the tests states in the previous section.

Test	Test Data Type	Test Data	Justification of Test Data
1.2.	Normal	"Operations" Button Clicked.	Should result in the "Operations" window being displayed.
1.3.	Normal	"Codes" Button Clicked.	Should result in the "Codes" window being displayed.
1.4.	Normal	"Users" Button Clicked.	Should result in the "Users" window being displayed.
1.5.	Normal	"Consultants" Button Clicked.	Should result in the "Consultants" window being displayed.
1.6.	Normal	"Patients" Button Clicked.	Should result in the "Patients" window being displayed.
1.12.	Normal	"Search" button Clicked.	Should result in the "Operations – Search Results" window being displayed.
1.13.	Normal	"Add" button Clicked.	Should result in the "Codes – Add Code" window being displayed.
1.14.	Normal	"Search" button Clicked.	Should result in the "Codes – Search Results" window being displayed.
1.15.	Normal	"Calculator" button Clicked.	Should result in the "Codes – Calculator" window being displayed.
1.16.	Normal	"Add Patient" button Clicked.	Should result in the "Patients – Add Patient" window being displayed.
1.17.	Normal	"Find Patient" button Clicked.	Should result in the "Patients - Search" window being displayed.
1.18.	Normal	"Search" button Clicked.	Should result in the "Patients – Search Results" window being displayed.
1.19.	Normal	"Add Consultant" button Clicked	Should result in the "Consultants – Add Consultant" window being displayed.
1.20.	Normal	"Find Consultant" button Clicked	Should result in the "Consultants – Search" window being displayed.
1.21.	Normal	"Search" button Clicked	Should result in the "Consultants – Search Results" window being displayed.
1.22.	Normal	"Add User" button Clicked	Should result in the "User – Add User" window being displayed.
1.23.	Normal	"Find User" button Clicked	Should result in the "User – Search" window being displayed.
1.24.	Normal	"Search" button Clicked	Should result in the "User – Search Results" window being displayed.
2.2.	Erroneous	'-1'/ 'a'/ '199-1119'	Should produce - Validation Error
	Normal	'1994-11-19'	Should be Accepted – Proceed to next window

2.3.	Normal	‘A1234’	Should be Accepted – Proceed to next window
	Erroneous	‘a1234’/ ‘-1’	Should produce - Validation Error
2.4.	Normal	‘A1234’	Should be Accepted – Proceed to next window
	Erroneous	‘a1234’/ ‘-1’	Should produce - Validation Error
2.5.	Normal	‘A1234,B1234’/ ‘A1234’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
2.6.	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Extreme/Boundary	‘0’	Should produce - Validation Error
	Normal	‘1’	Should be Accepted – Proceed to next window
2.7.	Normal	‘A1234’	Should be Accepted – Proceed to next window
	Erroneous	‘a1234’/ ‘-1’	Should produce - Validation Error
2.9.	Normal	‘Jack’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘jack’	Should produce - Validation Error
2.10.	Normal	‘Cox’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘cox’	Should produce - Validation Error
2.11.	Normal	‘45 Hello Road’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
2.12.	Normal	‘Abury’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
2.13.	Normal	‘Cambs’	Should be Accepted – Proceed to next window
	Erroneous	‘cambs’/ ‘-1’	Should produce - Validation Error
2.14.	Normal	‘CB8 8HN’/ ‘CB22 8HN’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
2.15.	Erroneous	‘-1’/ ‘a’/ ‘199-1119’	Should produce - Validation Error
	Normal	‘1994-11-19’	Should be Accepted – Proceed to next window
2.17.	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Normal	‘www.bob@gmail.com’	Should be Accepted – Proceed to next window
2.18.	Normal	‘01234-567899’/ ‘01234 567899’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Extreme/Boundary	‘01234 56789’/ ‘ ’	Should produce - Validation Error
2.20.	Normal	‘A1234’	Should be Accepted – Proceed to next window
	Erroneous	‘a1234’/ ‘-1’	Should produce - Validation Error
2.21.	Erroneous	‘-1’/ ‘a’/ ‘199-1119’	Should produce - Validation Error

	Normal	‘1994-11-19’	Should be Accepted – Proceed to next window
2.24.	Extreme/Boundary	‘0’	Should produce - Validation Error
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Normal	‘1’	Should be Accepted – Proceed to next window
2.25.	Normal	‘Jack’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘jack’	Should produce - Validation Error
2.26.	Normal	‘Cox’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘cox’	Should produce - Validation Error
2.28.	Normal	‘Jack’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘jack’	Should produce - Validation Error
2.29.	Normal	‘Cox’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘cox’	Should produce - Validation Error
2.30.	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Normal	‘www.bob@gmail.com’	Should be Accepted – Proceed to next window
2.31.	Extreme/Boundary	‘0’	Should produce - Validation Error
	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Normal	‘1’	Should be Accepted – Proceed to next window
2.32	Normal	‘Jack’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘jack’	Should produce - Validation Error
2.33.	Normal	‘Cox’	Should be Accepted – Proceed to next window
	Erroneous	‘-1’/ ‘cox’	Should produce - Validation Error
2.34.	Normal	‘User’/ ‘user1’	Should be Accepted – Proceed to next window
	Erroneous	‘U$er’	Should produce - Validation Error
2.35.	Erroneous	‘password’	Should produce - Validation Error
	Normal	‘password1’	Should be Accepted – Proceed to next window
2.36.	Erroneous	‘-1’/ ‘a’	Should produce - Validation Error
	Normal	‘www.bob@gmail.com’	Should be Accepted – Proceed to next window
2.38.	Extreme/Boundary	‘0’	Should produce - Validation Error

	Erroneous	'-1'/ 'a'	Should produce - Validation Error
	Normal	'1'	Should be Accepted – Proceed to next window
2.39.	Normal	'User'/ 'user1'	Should be Accepted – Proceed to next window
	Erroneous	'U$er'	Should produce - Validation Error
3.1.	Normal	Codes : A1234, B1234, C1234 Incompatible: None	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'BUPA'	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
3.2.	Normal	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'WPA'	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'WPA'	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
3.3.	Normal	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'PPP'	Should output correct values of: Surgeon Price: £1100 Anaesthetist Price: £550 Total Price: £1650
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'PPP'	Should output correct values of: Surgeon Price: £1100 Anaesthetist Price: £550 Total Price: £1650
3.4.	Normal	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'AVIVA'	Should output correct values of: Surgeon Price: £1200 Anaesthetist Price: £600 Total Price: £1800
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'AVIVA'	Should output correct values of: Surgeon Price: £1100 Anaesthetist Price: £550 Total Price: £1650
3.5.	Normal	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'PruHealth'	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560

		D1234 Insurance: ‘PruHealth’	Total Price: £1680
3.6.	Normal	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘Cigna’	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘Cigna’	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
3.7.	Normal	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘Simply Health’	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
	Normal	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘Simply Health’	Should output correct values of: Surgeon Price: £1120 Anaesthetist Price: £560 Total Price: £1680
4.1.	Normal	Code Name: ‘J9999’ Incompatible: ‘A1234’ Prices- BUPA: ‘1, 2’ WPA: ‘3, 4’ PPP: ‘5, 6’ AVIVA: ‘7, 8’ PruHealth: ‘9, 10’ Cigna: ‘11, 12’ Simply Health: ‘13, 14’	Matching data to appear in correct entities
4.2.	Normal	Code Name used in 4.1. ‘J9999’	Returns data input in test 4.1.
4.3.	Normal	Code Name used in 4.1. ‘J9999’	Removes code and corresponding data from the database
4.4.	Normal	Title: ‘Mr’ First Name: ‘Bob’ Surname: ‘Jones’ Address Line 1: ’24 Mill’ Address Line 2: ‘Cherry’ Town: ‘Cambridge’ Post Code: ‘JD9 5BJ’ Date of Birth: ‘1994-12-12’ Gender: ‘Male’ Email: ‘bj@google.com’ Telephone: ‘01210-123456’	Matching data to appear in correct entities

		Insurance: ‘BUPA’ Codes: ‘A1234,B1234’ Date of Operation: ‘2001-05-13’ Consultant: ‘Dr Jim Howard’	
4.5.	Normal	Patient Name used in 4.4. ‘Bob’, ‘Jones’	Returns data input in test 4.4.
4.6.	Normal	Patient Name used in 4.4. ‘Bob’, ‘Jones’	Removes code and corresponding data from the database
4.7.	Normal	Title: ‘Dr’ First Name: ‘Andy’ Surname: ‘Leese’ Email: ‘con@smart.com’	Matching data to appear in correct entities
4.8.	Normal	Consultant Name used in 4.7. ‘Andy’, ‘Leese’	Returns data input in test 4.7.
4.9.	Normal	Consultant Name used in 4.7. ‘Andy’, ‘Leese’	Removes code and corresponding data from the database
4.10.	Normal	Username: ‘Batman’ Passwrod: ‘Robin’ Email: ‘b@batcave.com’ Access: ‘Admin’	Matching data to appear in correct entities
4.11.	Normal	Username used in 4.10. ‘Batman’	Returns data input in test 4.10.
4.12.	Normal	Username used in 4.10. ‘Batman’	Removes code and corresponding data from the database

2.2. Changes from original data

Test	Test Data Type	Test Data	Justification of Test Data
1.25.	Normal	Click on the "Find Operations" tab in the menu bar.	Should result in the "Operations" window being displayed.
1.26.	Normal	Click on the "Code Main" tab in the menu bar.	Should result in the "Code" window being displayed.
1.27.	Normal	Click on the "Add Code" tab in the menu bar.	Should result in the "Code- Add Code" window being displayed.
1.28.	Normal	Click on the "Code Calculator" tab in the menu bar.	Should result in the "Code-Calculator" window being displayed.
1.29.	Normal	Click on the "User Main" tab in the menu bar.	Should result in the "User" window being displayed.
1.30.	Normal	Click on the "Add User" tab in the menu bar.	Should result in the "User- Add User" window being displayed.
1.31.	Normal	Click on the "Find User" tab in the menu bar.	Should result in the "User- Search" window being displayed.
1.32.	Normal	Click on the "Consultant Main" tab in the menu bar.	Should result in the "Consultant" window being displayed.
1.33.	Normal	Click on the "Add Consultant" tab in the menu bar.	Should result in the "Consultant-Add Consultant" window being displayed.
1.34.	Normal	Click on the "Find Consultant" tab in the menu bar.	Should result in the "Consultants - Search" window being displayed.
1.35.	Normal	Click on the "Patient Main" tab in the menu bar.	Should result in the "Patient" window being displayed.
1.36.	Normal	Click on the "Add Patient" tab in the menu bar.	Should result in the "Patient- Add Patient" window being displayed.
1.37.	Normal	Click on the "Find Patient" tab in the menu bar.	Should result in the "Patient- Find Patient" window being displayed.
1.38.	Normal	Click on the "Main Menu" tab in the menu bar.	Should result in the "CMP Window" being displayed.
1.39.	Normal	Click on the "Log In" tab in the menu bar.	Should result in the "Log In" window being displayed.
1.40.	Normal	Click on the "Log Out" tab in the menu bar.	Should result in Confirmation message box displayed.
1.41.	Normal	Click on the "Exit" tab in the menu bar.	Should result in system closure.
2.40.	Normal	'User'/ 'User1'	Should be Accepted – Proceed to next window
	Erroneous	'U$er'	Should produce - Validation Error

2.41.	Normal	'password'	Should be Accepted – Proceed to next window
	Erroneous	'password1'	Should produce - Validation Error
5.1.	Normal	A non-existent user, eg. 'abcdefg'	Should produce – Non-existent Error
5.2.	Normal	An existing user with incorrect password, eg. 'Jack', 'Lemons'	Should produce – Incorrect password Error
5.3.	Normal	Click "Operations" main button on the home screen, and all tabs within the "Operation" menu bar. Whilst not logged in.	Should produce – Log In Error
5.4.	Normal	Click "Codes" main button on the home screen, and all tabs within the "Code" menu bar. Whilst not logged in.	Should produce – Log In Error
5.5.	Normal	Click "Users" main button on the home screen, and all tabs within the "User" menu bar. Whilst not logged in.	Should produce – Log In Error
5.6.	Normal	Click "Patients" main button on the home screen, and all tabs within the "Patient" menu bar. Whilst not logged in.	Should produce – Log In Error
5.7.	Normal	Click "Consultants" main button on the home screen, and all tabs within the "Consultant" menu bar. Whilst not logged in.	Should produce – Log In Error
5.8.	Normal	Click "Operations" main button on the home screen, and all tabs within the "Operation" menu bar. Whilst logged in with 'User' level access.	Should result in the "Operations" window being displayed.
5.9.	Normal	Click "Codes" main button on the home screen, and all tabs within the "Code" menu bar. Whilst logged in with 'User' level access.	Should result in the "Codes" window being displayed.
5.10.	Normal	Click "Users" main	Should produce – Access Restriction

		button on the home screen, and all tabs within the “User” menu bar Whilst logged in with ‘User’ level access.	Error
5.11.	Normal	Click “Patients” main button on the home screen, and all tabs within the “Patient” menu bar. Whilst logged in with ‘User’ level access.	Should result in the “Patients” window being displayed.
5.12.	Normal	Click “Consultants” main button on the home screen, and all tabs within the “Consultant” menu bar. Whilst logged in with ‘User’ level access.	Should result in the “Consultants” window being displayed.
5.13.	Normal	Click “Operations” main button on the home screen, and all tabs within the “Operation” menu bar. Whilst logged in with ‘Admin’ level access.	Should result in the “Operations” window being displayed.
5.14.	Normal	Click “Codes” main button on the home screen, and all tabs within the “Code” menu bar. Whilst logged in with ‘Admin’ level access.	Should result in the “Codes” window being displayed.
5.15.	Normal	Click “Users” main button on the home screen, and all tabs within the “User” menu bar Whilst logged in with ‘Admin’ level access.	Should result in the “Users” window being displayed.
5.16.	Normal	Click “Patients” main button on the home screen, and all tabs within the “Patient” menu bar. Whilst logged in with ‘Admin’ level access.	Should result in the “Patients” window being displayed.
5.17.	Normal	Click “Consultants” main button on the home screen, and all tabs within the “Consultant”	Should result in the “Consultants” window being displayed.

		menu bar. Whilst logged in with ‘Admin’ level access.	
6.1.	Erroneous	Click the “Search” button with no method selected.	Should produce – Search Error
6.2.	Normal	Click the “Search” button after entering non-existing data.	Should produce – Non-existing Error
6.3.	Normal	Click the “Search” button after entering non-existing data.	Should produce – Non-existing Error
6.4.	Erroneous	Click the “Search” button with no method selected.	Should produce – Search Error
6.5.	Normal	Click the “Search” button after entering non-existing data.	Should produce – Non-existing Error
6.7.	Erroneous	Click the “Search” button with no method selected.	Should produce – Search Error
6.8.	Normal	Click the “Search” button after entering non-existing data.	Should produce – Non-existing Error
6.9.	Erroneous	Click the “Search” button with no method selected.	Should produce – Search Error
6.10.	Normal	Click the “Search” button after entering non-existing data.	Should produce – Non-existing Error

3. Annotated Samples

3.1. Actual Results

Test	Test Data Type	Expected Result	Test Data	Actual Result	Annotation Reference
1.2.	Normal	"Operations" Window displayed.	"Operations" Button Clicked.	Expected Result	Figure 1 – Test 1.2. page 119
1.3.	Normal	"Codes" Window displayed.	"Codes" Button Clicked.	Expected Result	
1.4.	Normal	"Users" Window displayed.	"Users" Button Clicked.	Expected Result	
1.5.	Normal	"Consultants" Window displayed.	"Consultants" Button Clicked.	Expected Result	
1.6.	Normal	"Patients" Window displayed.	"Patients" Button Clicked.	Expected Result	
1.12.	Normal	"Operations – Search Results" Window displayed.	"Search" button Clicked.	Expected Result	
1.13.	Normal	"Codes – New Code" Window displayed.	"Add" button Clicked.	Expected Result	Figure 2 - Test 1.13. page 119
1.14.	Normal	"Codes – Search Results" Window displayed.	"Search" button Clicked.	Expected Result	
1.15.	Normal	"Codes – Calculator" Window displayed.	"Calculator" button Clicked.	Expected Result	
1.16.	Normal	"Patients – New Patient" Window displayed.	"Add Patient" button Clicked.	Expected Result	
1.17.	Normal	"Patients - Search" Window displayed.	"Find Patient" button Clicked.	Expected Result	Figure 3 - Test 1.17. page 120
1.18.	Normal	"Patients – Search Results" Window displayed.	"Search" button Clicked.	Expected Result	
1.19.	Normal	"Consultants – New Consultant" Window displayed.	"Add Consultant" button Clicked	Expected Result	
1.20.	Normal	"Consultants – Search" Window displayed.	"Find Consultant" button Clicked	Expected Result	
1.21.	Normal	"Consultants – Search Results" Window displayed.	"Search" button Clicked	Expected Result	
1.22.	Normal	"User – New User" Window displayed.	"Add User" button Clicked	Expected Result	
1.23.	Normal	"User – Search"	"Find User" button	Expected	

		Window displayed.	Clicked	Result	
1.24.	Normal	"User – Search Results" Window displayed.	"Search" button Clicked	Expected Result	
1.25.	Normal	"Operations" Window displayed.	Click on the "Find Operations" tab in the menu bar.	Expected Result	
1.26.	Normal	"Code" Window displayed.	Click on the "Code Main" tab in the menu bar.	Expected Result	
1.27.	Normal	"Code- Add Code" Window displayed.	Click on the "Add Code" tab in the menu bar.	Expected Result	
1.28.	Normal	"Code- Calculator" Window displayed.	Click on the "Code Calculator" tab in the menu bar.	Expected Result	Figure 4 - Test 1.28. page 120
1.29.	Normal	"User" Window displayed.	Click on the "User Main" tab in the menu bar.	Expected Result	
1.30.	Normal	"User- Add User" Window displayed.	Click on the "Add User" tab in the menu bar.	Expected Result	
1.31.	Normal	"User - Search" Window displayed.	Click on the "Find User" tab in the menu bar.	Expected Result	
1.32.	Normal	"Consultant" Window displayed.	Click on the "Consultant Main" tab in the menu bar.	Expected Result	
1.33.	Normal	"Consultants- Add Consultant" Window displayed.	Click on the "Add Consultant" tab in the menu bar.	Expected Result	
1.34.	Normal	"Consultants- Search" Window displayed.	Click on the "Find Consultant" tab in the menu bar.	Expected Result	
1.35.	Normal	"Patient" Window displayed.	Click on the "Patient Main" tab in the menu bar.	Expected Result	
1.36.	Normal	"Patient- Add Patient" Window displayed.	Click on the "Add Patient" tab in the menu bar.	Expected Result	
1.37.	Normal	"Patient- Find Patient" Window displayed.	Click on the "Find Patient" tab in the menu bar.	Expected Result	
1.38.	Normal	"CMP Window" Window displayed.	Click on the "Main Menu" tab in the menu bar.	Expected Result	
1.39.	Normal	"Log In" Window displayed.	Click on the "Log In" tab in the menu bar.	Expected Result	
1.40.	Normal	Confirmation	Click on the "Log Out"	Expected	Figure 5 -

		message box displayed.	tab in the menu bar.	Result	Test 1.40. page 121
1.41.	Normal	System closure.	Click on the "Exit" tab in the menu bar.	Expected Result	
2.2.	Erroneous	Error message displayed	'-1'/ 'a'/ '199-1119'	Expected Result	Figure 6 - Test 2.2. page 121
	Normal	Accepted	'1994-11-19'	Expected Result	
2.3.	Normal	Accepted	'A1234'	Expected Result	
	Erroneous	Error message displayed	'a1234'/ '-1'	Expected Result	
2.4.	Normal	Accepted	'A1234'	Expected Result	
	Erroneous	Error message displayed	'a1234'/ '-1'	Expected Result	
2.5.	Normal	Accepted	'A1234,B1234'/ 'A1234'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'a'	Expected Result	
2.6.	Erroneous	Error message displayed	'-1'/ 'a'	Expected Result	
	Extreme/ Boundary	Error message displayed	'0'	Expected Result	
	Normal	Accepted	'1'	Expected Result	
2.7.	Normal	Accepted	'A1234'	Expected Result	
	Erroneous	Error message displayed	'a1234'/ '-1'	Expected Result	
2.9.	Normal	Accepted	'Jack'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'jack'	Expected Result	
2.10.	Normal	Accepted	'Cox'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'cox'	Expected Result	
2.11.	Normal	Accepted	'45 Hello Road'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'a'	Failed Result. System accepted invalid data.	
2.12.	Normal	Accepted	'Arbury'	Expected Result	Figure 7 – Test 2.12. page 122
	Erroneous	Accepted	'-1'/ 'a'	Failed	

				Result. System accepted invalid data.	
2.13.	Normal	Accepted	‘Cambs’	Expected Result	
	Erroneous	Error message displayed	‘cambs’/ ‘-1’	Expected Result	
2.14.	Normal	Accepted	‘CB8 8HN’/ ‘CB22 8HN’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
2.15.	Erroneous	Error message displayed	‘-1’/ ‘a’/ ‘199-1119’	Expected Result	
	Normal	Accepted	‘1994-11-19’	Expected Result	
2.17.	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Accepted	‘bob@gmail.com’	Expected Result	
2.18.	Normal	Accepted	‘01234-567899’/ ‘01234 567899’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Extreme/ Boundary	Error message displayed	‘01234 56789’/ ‘ ’	Expected Result	
2.20.	Normal	Accepted	‘A1234’	Expected Result	
	Erroneous	Error message displayed	‘a1234’/ ‘-1’	Expected Result	
2.21.	Erroneous	Error message displayed	‘-1’/ ‘a’/ ‘199-1119’	Expected Result	
	Normal	Accepted	‘1994-11-19’	Expected Result	
2.24.	Extreme/ Boundary	Error message displayed	‘0’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Accepted	‘1’	Expected Result	
2.25.	Normal	Accepted	‘Jack’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘jack’	Expected Result	
2.26.	Normal	Accepted	‘Cox’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘cox’	Expected Result	

2.28.	Normal	Accepted	‘Jack’	Expected Result	Figure 8 - Test 2.28. page 123
	Erroneous	Error message displayed	‘-1’/ ‘jack’	Expected Result	
2.29.	Normal	Accepted	‘Cox’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘cox’	Expected Result	
2.30.	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Accepted	‘bob@gmail.com’	Expected Result	
2.31.	Extreme/ Boundary	Accepted	‘0’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Error message displayed	‘1’	Expected Result	
2.32	Normal	Accepted	‘Jack’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘jack’	Expected Result	
2.33.	Normal	Accepted	‘Cox’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘cox’	Expected Result	
2.34.	Normal	Accepted	‘User’/ ‘user1’	Expected Result	
	Erroneous	Error message displayed	‘U$er’	Expected Result	
2.35.	Erroneous	Error message displayed	‘password’	Expected Result	
	Normal	Accepted	‘password1’	Expected Result	
2.36.	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	Figure 9 - Test 2.36. page 123
	Normal	Accepted	‘bob@gmail.com’	Expected Result	
2.38.	Extreme/ Boundary	Accepted	‘0’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Error message displayed	‘1’	Expected Result	
2.39.	Normal	Accepted	‘User’/ ‘user1’	Expected Result	
	Erroneous	Error message displayed	‘U$er’	Expected Result	
2.40.	Normal	Accepted	‘User’/ ‘User1’	Expected	

				Result	
	Erroneous	Error message displayed	‘U$er’	Expected Result	
2.41.	Normal	Accepted	‘password’	Expected Result	
	Erroneous	Error message displayed	‘password1’	Expected Result	
3.1.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘BUPA’	Expected Result	Figure 10 - Test 3.1. page 124
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘BUPA’	Expected Result	
3.2.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘WPA’	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘WPA’	Expected Result	
3.3.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘PPP’	Expected Result	Figure 11 - Test 3.3. page 125
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘PPP’	Expected Result	
3.4.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘AVIVA’	Expected Result	Figure 12 - Test 3.4. page 126
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘AVIVA’	Expected Result	
3.5.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘PruHealth’	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234	Expected Result	

			and D1234 Insurance: 'PruHealth'		
3.6.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'Cigna'	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'Cigna'	Expected Result	
3.7.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'Simply Health'	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'Simply Health'	Expected Result	
4.1.	Normal	Data stored correctly	Code Name: 'J9999' Incompatible: 'A1234' Prices- BUPA: '1, 2' WPA: '3, 4' PPP: '5, 6' AVIVA: '7, 8' PruHealth: '9, 10' Cigna: '11, 12' Simply Health: '13, 14'	Expected Result	Figure 13 - Test 4.1. page 127
4.2.	Normal	Data updated correctly	Code Name used in 4.1. 'J9999'	Failed Result. Button was inactive and nothing happened when clicked.	Figure 14 - Test 4.2. page 128
4.3.	Normal	Data deleted correctly	Code Name used in 4.1. 'J9999'	Expected Result	
4.4.	Normal	Data stored correctly	Title: 'Mr' First Name: 'Bob' Surname: 'Jones' Address Line 1: '24 Mill' Address Line 2:	Expected Result	

			'Cherry' Town: 'Cambridge' Post Code: 'JD9 5BJ' Date of Birth: '1994-12-12' Gender: 'Male' Email: 'bj@google.com' Telephone: '01210-123456' Insurance: 'BUPA' Codes: 'A1234,B1234' Date of Operation: '2001-05-13' Consultant: 'Dr Jim Howard'		
4.5.	Normal	Data updated correctly	Patient Name used in 4.4. 'Bob', 'Jones'	Expected Result	
4.6.	Normal	Data deleted correctly	Patient Name used in 4.4. 'Bob', 'Jones'	Expected Result	
4.7.	Normal	Data stored correctly	Title: 'Dr' First Name: 'Andy' Surname: 'Leese' Email: 'con@smart.com'	Expected Result	Figure 15 - Test 4.7. page 128
4.8.	Normal	Data updated correctly	Consultant Name used in 4.7. 'Andy', 'Leese'	Expected Result	
4.9.	Normal	Data deleted correctly	Consultant Name used in 4.7. 'Andy', 'Leese'	Expected Result	Figure 16 - Test 4.9. page 129
4.10.	Normal	Data stored correctly	Username: 'Batman' Passwrod: 'Robin' Email: 'b@batcave.com' Access: 'Admin'	Expected Result	
4.11.	Normal	Data updated correctly	Username used in 4.10. 'Batman'	Expected Result	
4.12.	Normal	Data deleted correctly	Username used in 4.10. 'Batman'	Expected Result	
5.1.	Normal	Error message displayed	A non-existent user, eg. 'abcdefg'	Expected Result	Figure 17 - Test 5.1. page 129
5.2.	Normal	Error message displayed	An existing user with incorrect password, eg. 'Jack', 'Lemon'	Expected Result	Figure 18 - Test 5.2. page 130
5.3.	Normal	Error message displayed	Click "Operations" main button on the home screen, and all	Expected Result	

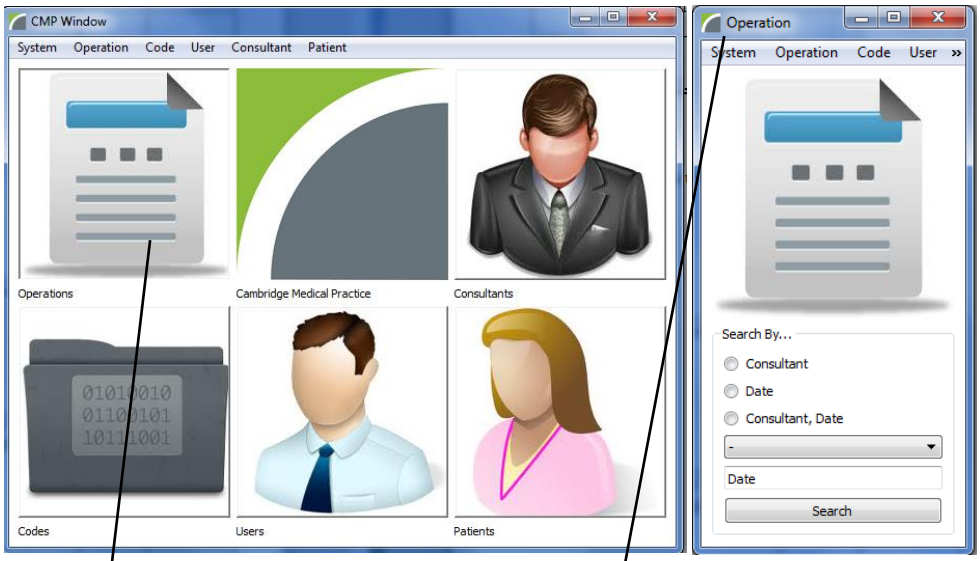
			tabs within the "Operation" menu bar. Whilst not logged in.		
5.4.	Normal	Error message displayed	Click "Codes" main button on the home screen, and all tabs within the "Code" menu bar. Whilst not logged in.	Expected Result	Figure 19 - Test 5.4. page 130
5.5.	Normal	Error message displayed	Click "Users" main button on the home screen, and all tabs within the "User" menu bar. Whilst not logged in.	Expected Result	
5.6.	Normal	Error message displayed	Click "Patients" main button on the home screen, and all tabs within the "Patient" menu bar. Whilst not logged in.	Expected Result	
5.7.	Normal	Error message displayed	Click "Consultants" main button on the home screen, and all tabs within the "Consultant" menu bar. Whilst not logged in.	Expected Result	
5.8.	Normal	"Operation" Window displayed	Click "Operations" main button on the home screen, and all tabs within the "Operation" menu bar. Whilst logged in with 'User' level access.	Expected Result	
5.9.	Normal	"Code" Window displayed	Click "Codes" main button on the home screen, and all tabs within the "Code" menu bar. Whilst logged in with 'User' level access.	Expected Result	
5.10.	Normal	Error message displayed	Click "Users" main button on the home screen, and all tabs within the "User" menu bar Whilst logged in with 'User' level access.	Expected Result	Figure 20 - Test 5.10. page 131
5.11.	Normal	"Patient" Window displayed	Click "Patients" main button on the home	Expected Result	

			screen, and all tabs within the "Patient" menu bar. Whilst logged in with 'User' level access.		
5.12.	Normal	"Consultant" Window displayed	Click "Consultants" main button on the home screen, and all tabs within the "Consultant" menu bar. Whilst logged in with 'User' level access.	Expected Result	
5.13.	Normal	"Operation" Window displayed	Click "Operations" main button on the home screen, and all tabs within the "Operation" menu bar. Whilst logged in with 'Admin' level access.	Expected Result	
5.14.	Normal	"Code" Window displayed	Click "Codes" main button on the home screen, and all tabs within the "Code" menu bar. Whilst logged in with 'Admin' level access.	Expected Result	
5.15.	Normal	"User" Window displayed	Click "Users" main button on the home screen, and all tabs within the "User" menu bar Whilst logged in with 'Admin' level access.	Expected Result	
5.16.	Normal	"Patient" Window displayed	Click "Patients" main button on the home screen, and all tabs within the "Patient" menu bar. Whilst logged in with 'Admin' level access.	Expected Result	
5.17.	Normal	"Consultant" Window displayed	Click "Consultants" main button on the home screen, and all tabs within the "Consultant" menu bar. Whilst logged in with 'Admin' level access.	Expected Result	Figure 21 - Test 5.17. page 132
6.1.	Erroneous	Error message displayed	Click the "Search" button with no method selected.	Expected Result	Figure 22 - Test 6.1 page 133

6.2.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	
6.3.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	
6.4.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	
6.5.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	Figure 23 - Test 6.5. page 134
6.7.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	
6.8.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	
6.9.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	
6.10.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	

3.2. Evidence

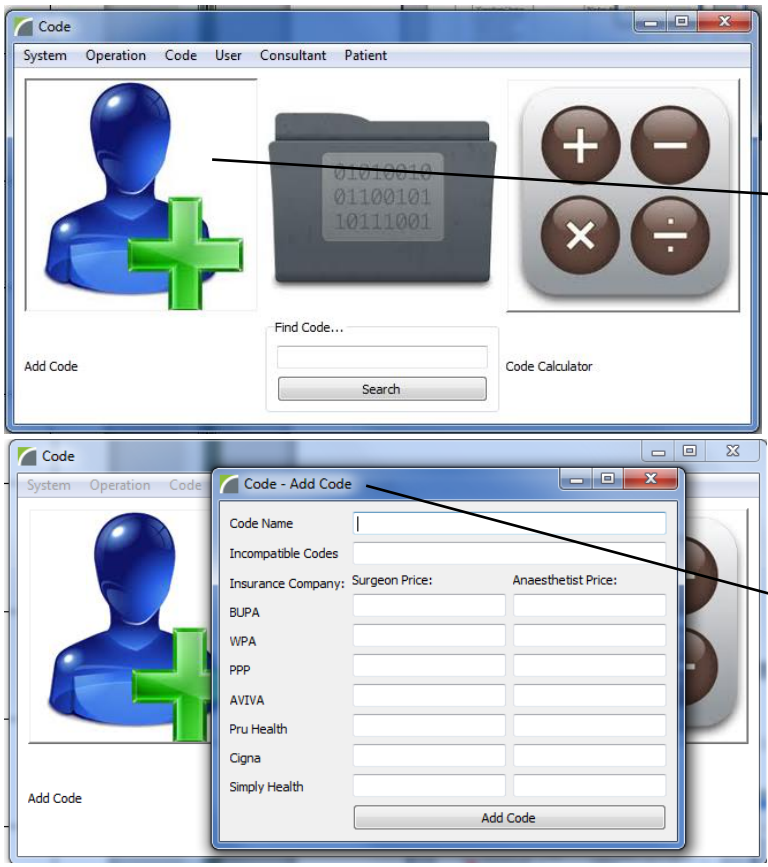
Figure 1 – Test 1.2.



Test data from test plan being input- "Operations" Button Clicked.

As expected the "Operation" window is displayed.

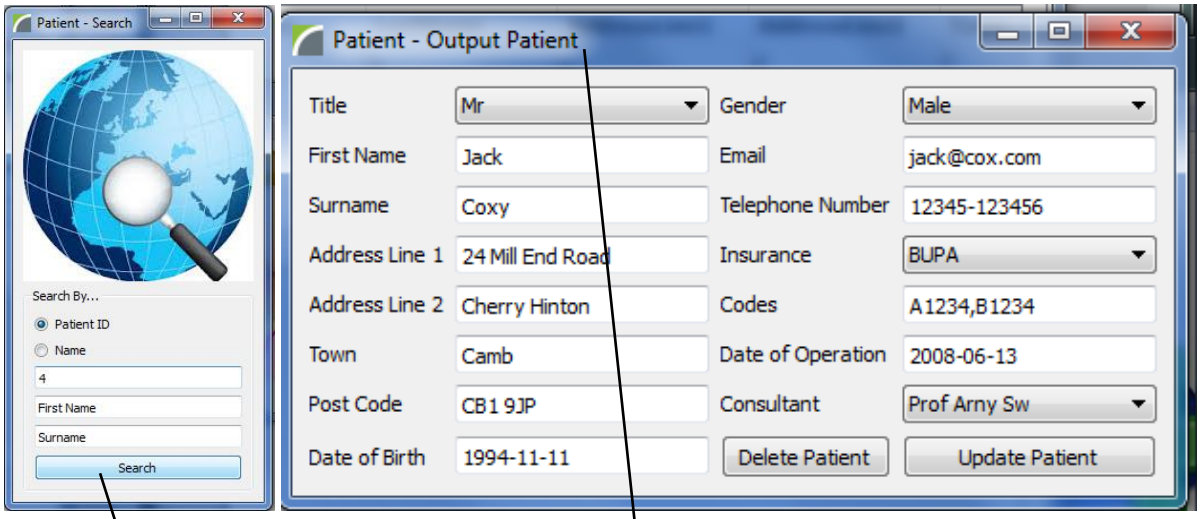
Figure 2 - Test 1.13.



Test data from test plan being input- "Add" button Clicked.

As expected the "Code- Add Code" window is displayed.

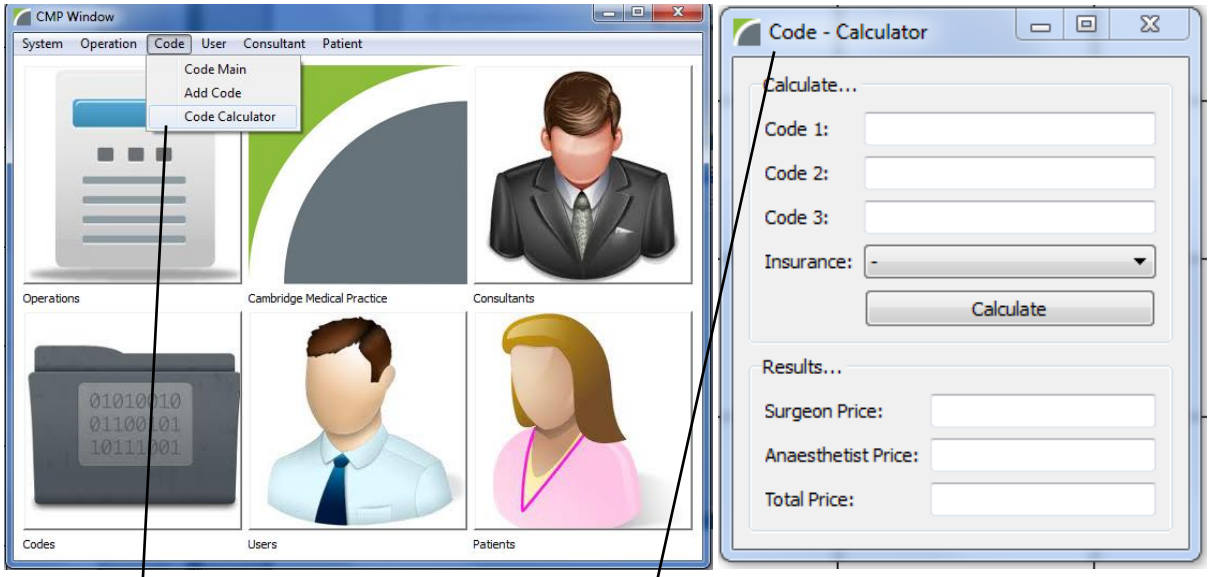
Figure 3 - Test 1.17.



Test data from test plan being input- "Search" Button Clicked.

As expected the "Patients- Output Patient" window is displayed.

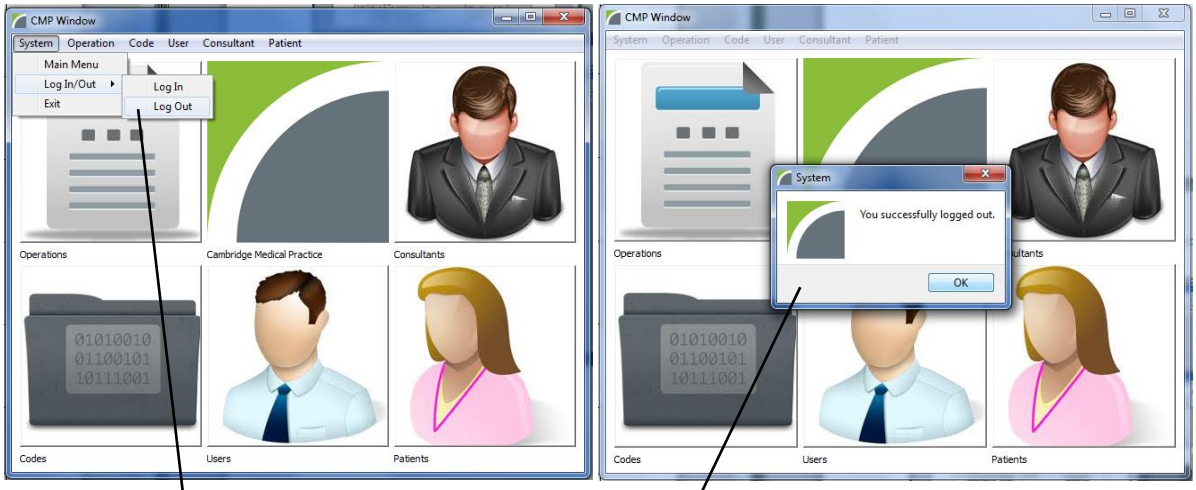
Figure 4 - Test 1.28.



Test data from test plan being input- "Code Calculator" tab in the menu bar clicked.

As expected the "Code-Calculator" window is displayed.

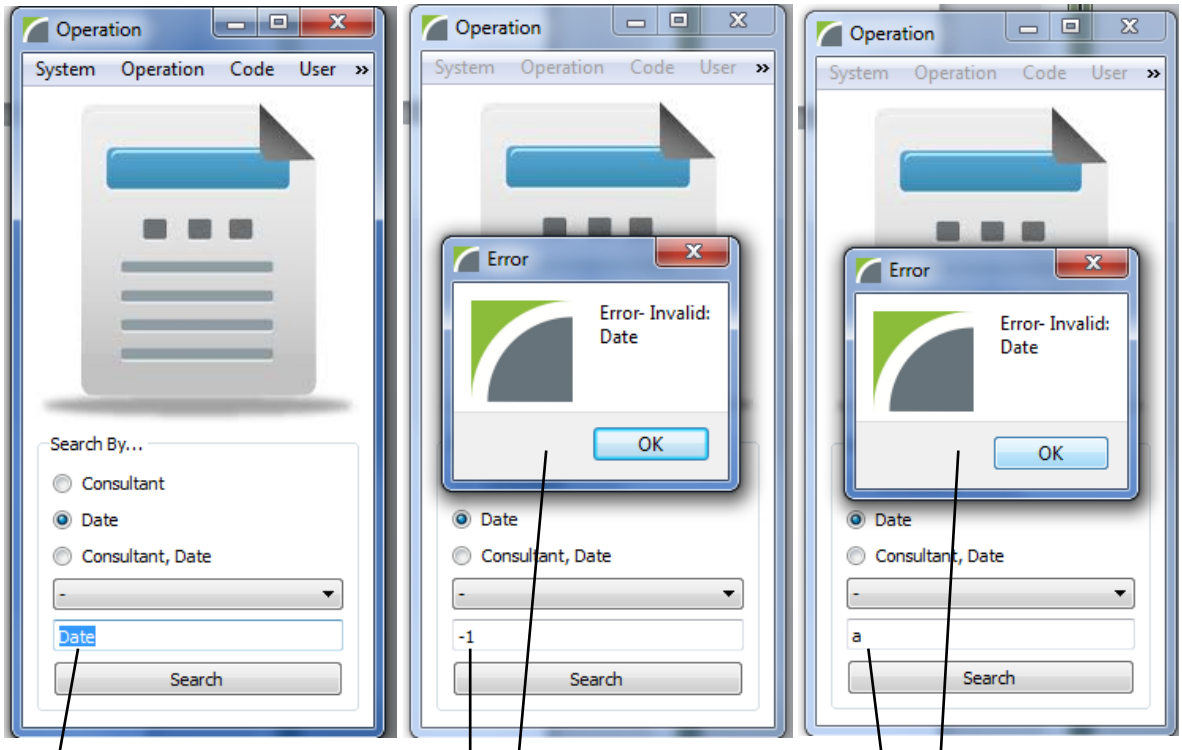
Figure 5 - Test 1.40.



Test data from test plan being input- "Code Calculator" tab in the menu bar clicked.

As expected the "Code-Calculator" window is displayed.

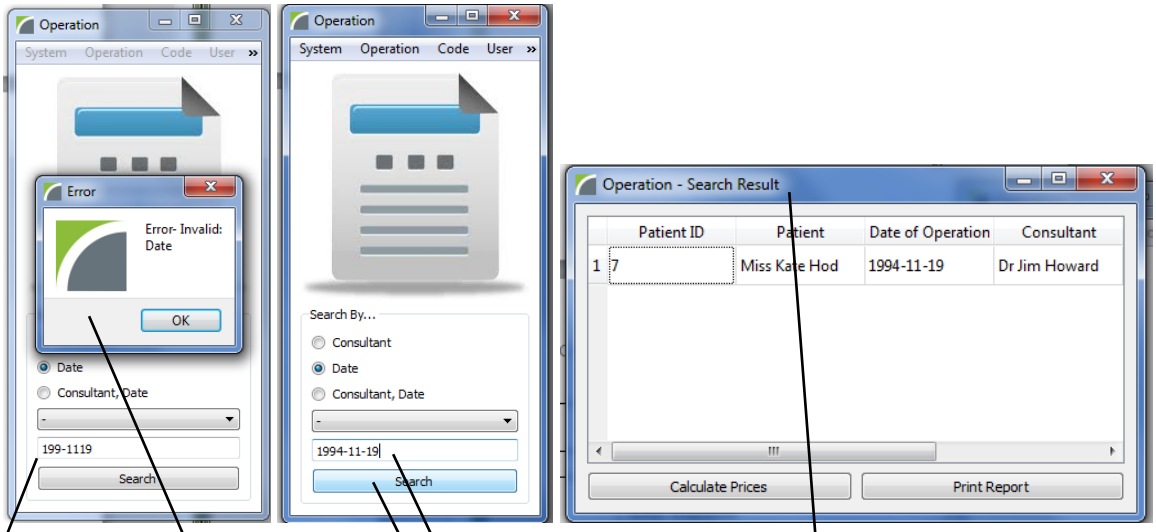
Figure 6 - Test 2.2.



This test validates the Date field from the "Operation" window. The following screen shots show the different data inputs stated in the test plan.

Erroneous test data- '-1'. As expected a validation error window is displayed preventing continuation.

Erroneous test data- 'a'. As expected a validation error window is displayed preventing continuation.

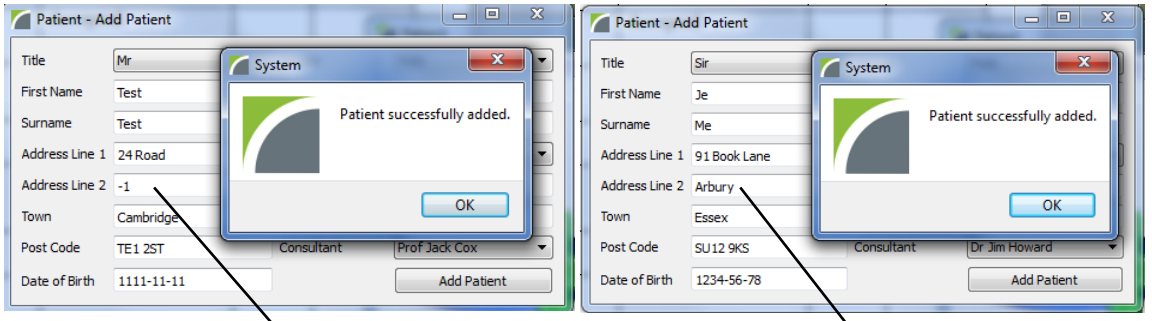


Erroneous test data- '199-1119'. As expected a validation error window is displayed preventing

Normal test data- '1994-11-19'. As expected no validation error window was allowing continuation.

As expected the input was excepted, and the "Operation- Search Result" was displayed.

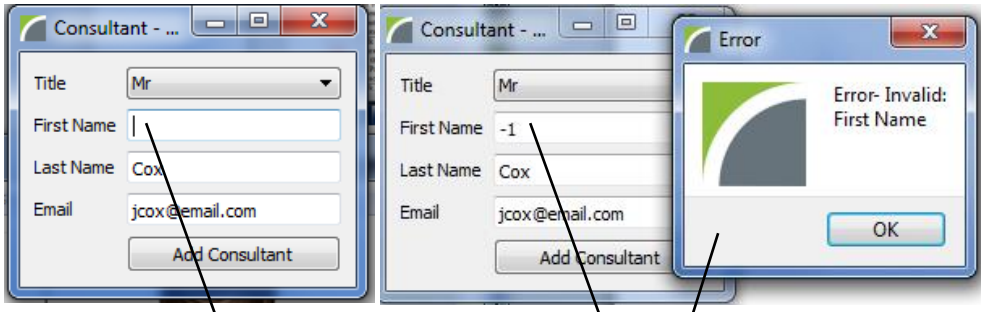
Figure 7 – Test 2.12.



This test validates the "Address Line 2" field in the "Patient- Add Patient" window. Erroneous test data- '-1'. Unexpected result as the system accepted the input. A validation error window should have been displayed preventing continuation.

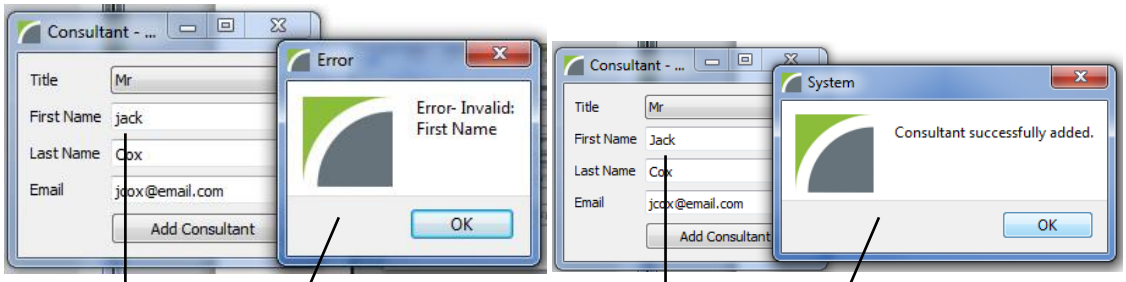
Normal test data- 'Arbury'. As expected the system accepted the input.

Figure 8 - Test 2.28.



This test validates the "First Name" field in the "Consultant- Add Consultant" window.

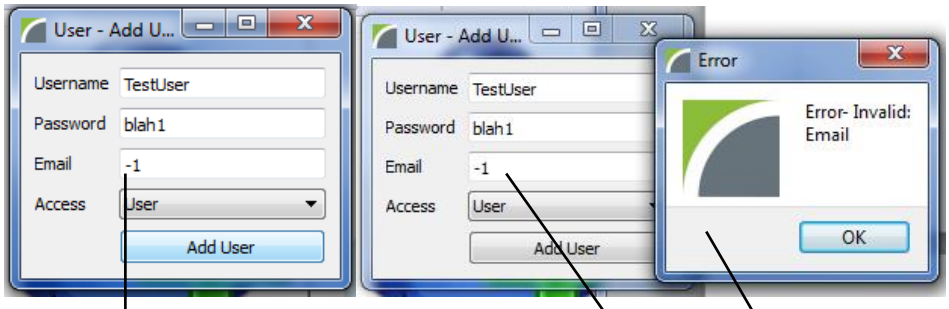
Erroneous test data- '-1'. As expected a validation error window is displayed preventing continuation.



Erroneous test data- 'jack'. As expected a validation error window is displayed preventing continuation.

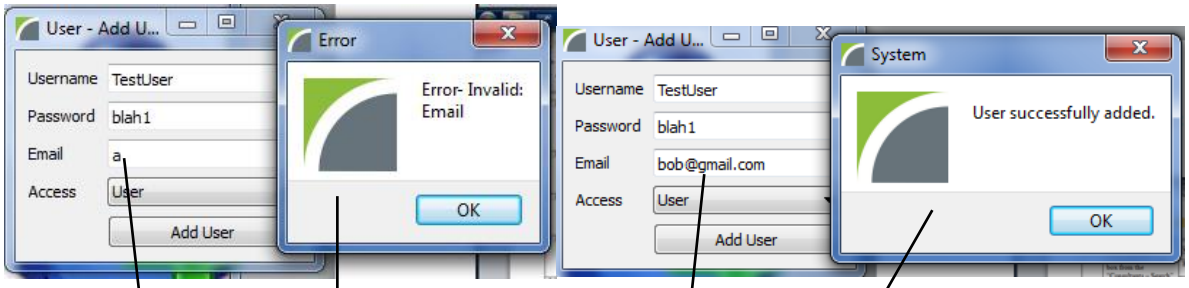
Normal test data- 'Jack'. As expected the system accepted the input.

Figure 9 - Test 2.36.



This test validates the "email" field in the "User- Add User" window.

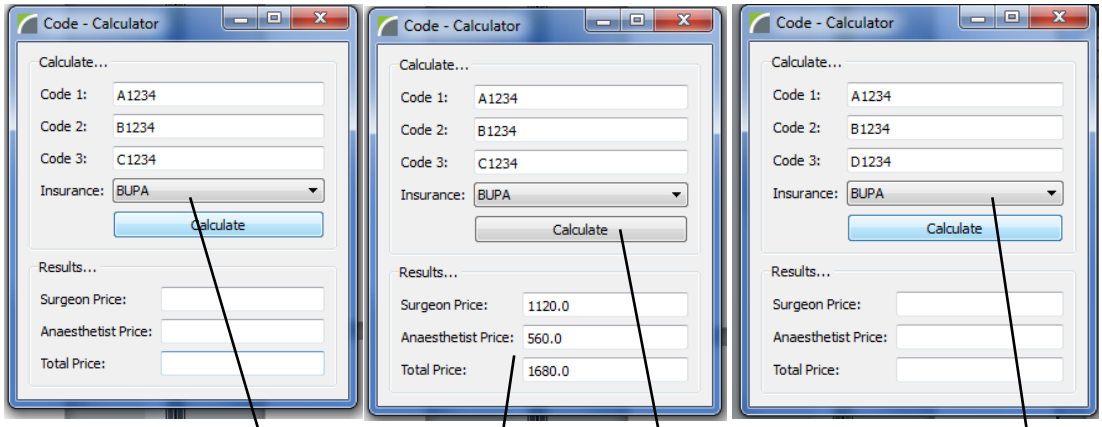
Erroneous test data- '-1'. As expected a validation error window is displayed preventing continuation.



Erroneous test data- 'a'. As expected a validation error window is displayed preventing continuation.

Normal test data- 'bob@gmail.com'. As expected the system accepted the input.

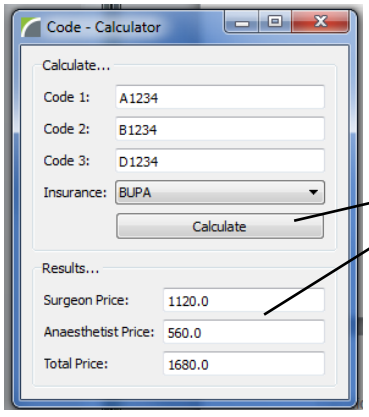
Figure 10 - Test 3.1.



This test checks the 'BUPA' calculation method used in the "Code-Calculator", where none of the codes are incompatible.

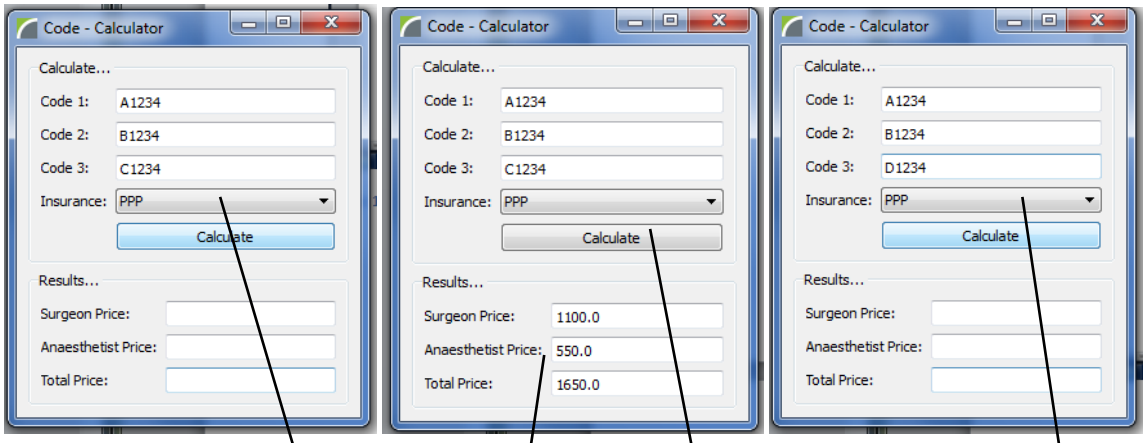
Using the test data from the test plan, the expected results were calculated after clicking the "calculate" button.

This test checks the 'BUPA' calculation method used in the "Code-Calculator", where there are incompatible codes.



Using the test data from the test plan, the expected results were calculated after clicking the "calculate" button.

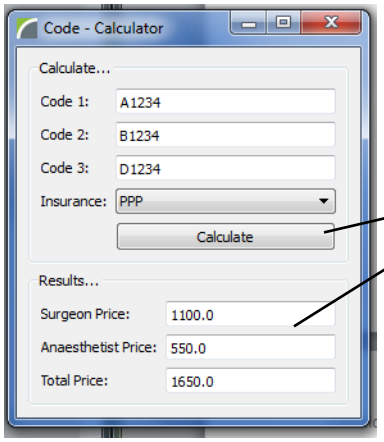
Figure 11 - Test 3.3.



This test checks the 'PPP' calculation method used in the "Code- Calculator", where none of the codes are incompatible.

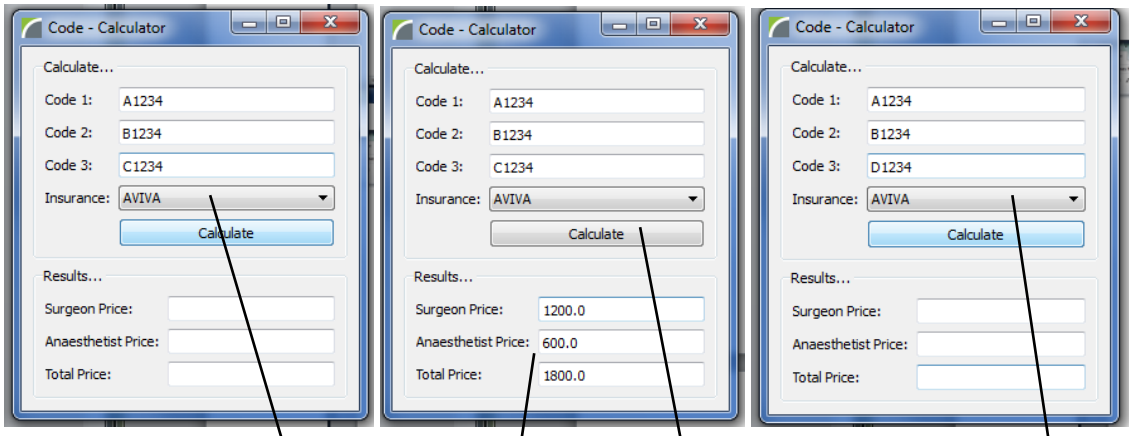
Using the test data from the test plan, the expected results were calculated after clicking the "calculate" button.

This test checks the 'PPP' calculation method used in the "Code- Calculator", where there are incompatible codes.



Using the test data from the test plan, the expected results were calculated after clicking the "calculate" button.

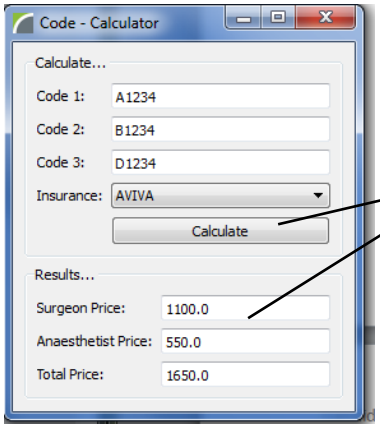
Figure 12 - Test 3.4.



This test checks the 'AVIVA' calculation method used in the "Code-Calculator", where none of the codes are incompatible.

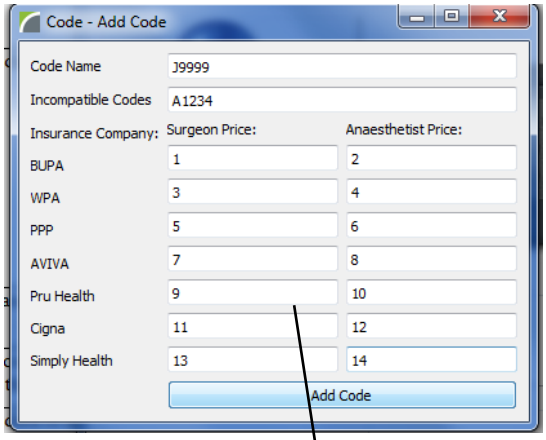
Using the test data from the test plan, the expected results were calculated after clicking the "calculate" button.

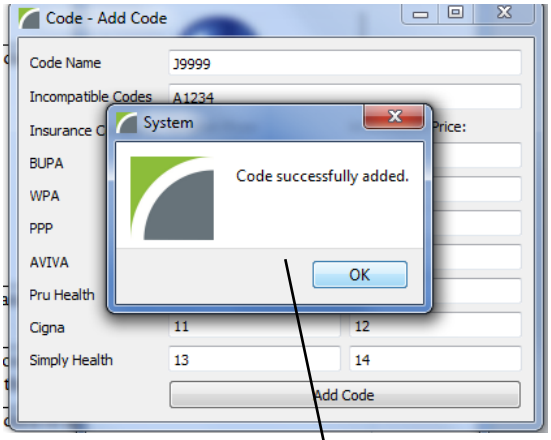
This test checks the 'AVIVA' calculation method used in the "Code-Calculator", where there are incompatible codes.



Using the test data from the test plan, the expected results were calculated after clicking the "calculate" button.

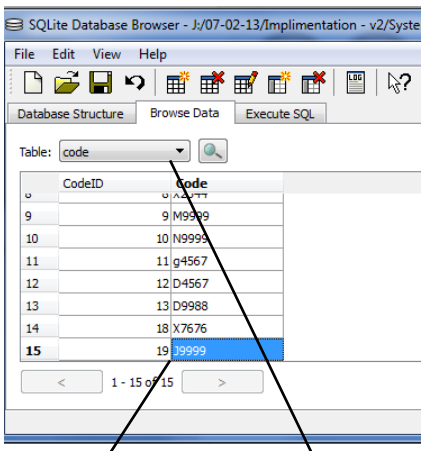
Figure 13 - Test 4.1.

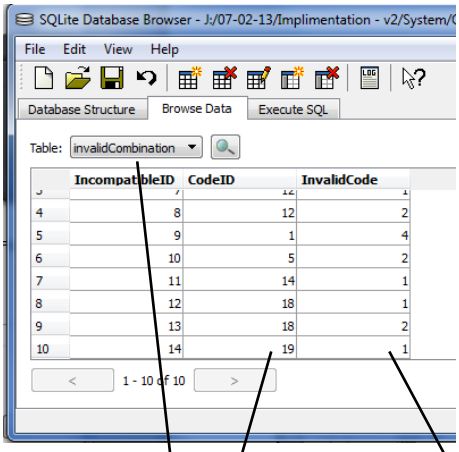




Using the normal test data stated in the test plan I clicked the add code button on in the "Code- Add Code" window.

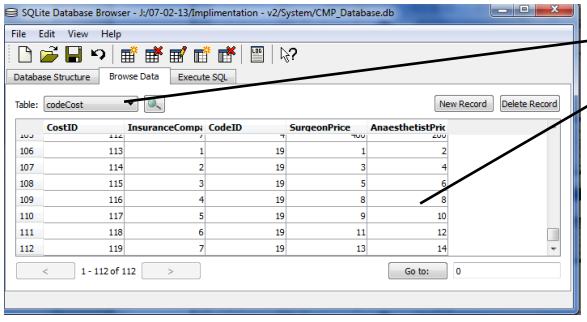
As expected the system accepted all of the inputs and the data should appear in the database.





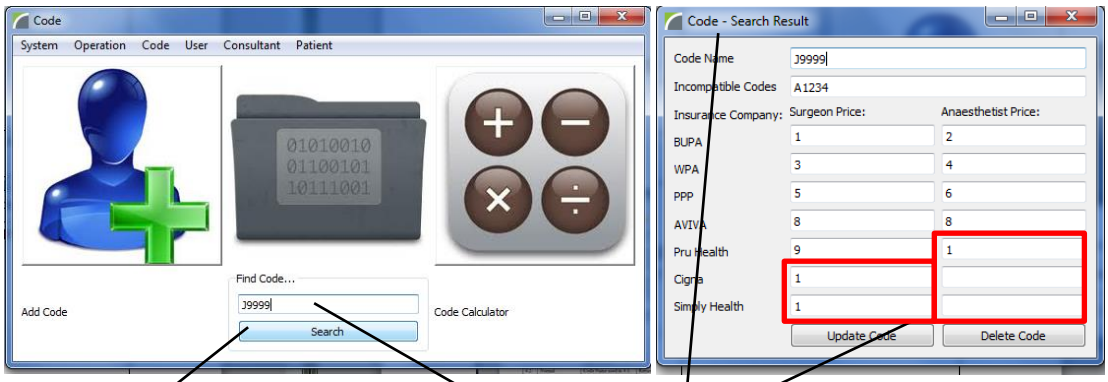
Using "SQLite Database Browser" to view the database more clearly, you can see that the correct code name, 'J9999' was stored in the "code" entity and assigned an ID of '19'.

The CodeID of 'A1234' is 1. This screenshot shows that the correct data was stored in the "invalidCombination" entity, as a new record was created storing the values for both the codes IDs, '19' and '1'.



Finally this screenshot shows that the correct costs were stored in the "codeCost" entity alongside the correct CodeID of '19'.

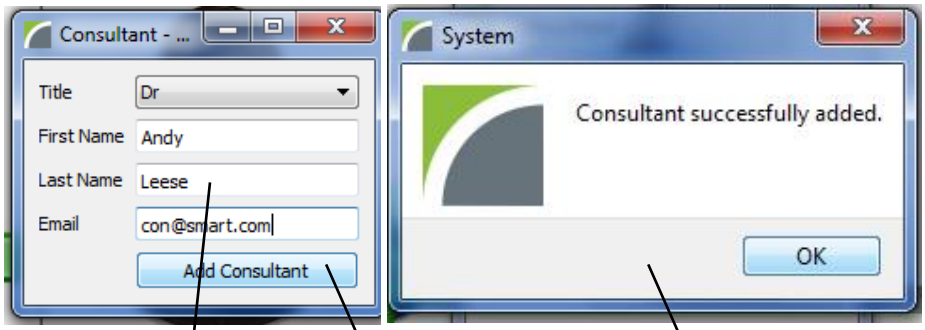
Figure 14 - Test 4.2.



Using the code name, 'J9999', from the previous test as the stated test data, this window attempts to retrieve data related to the code from the database.

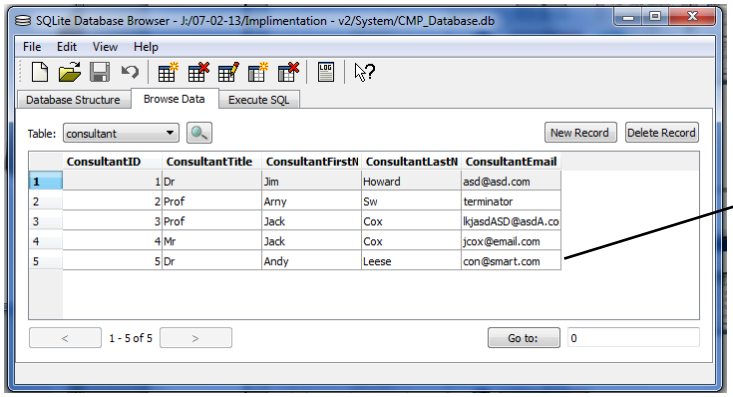
As expected the "Code- Search Result" window outputs data. However 2 of the fields are missing and 3 of the prices are incorrect (Highlighted). When the update button was clicked it was inactive and nothing happened. Therefore the test failed.

Figure 15 - Test 4.7.



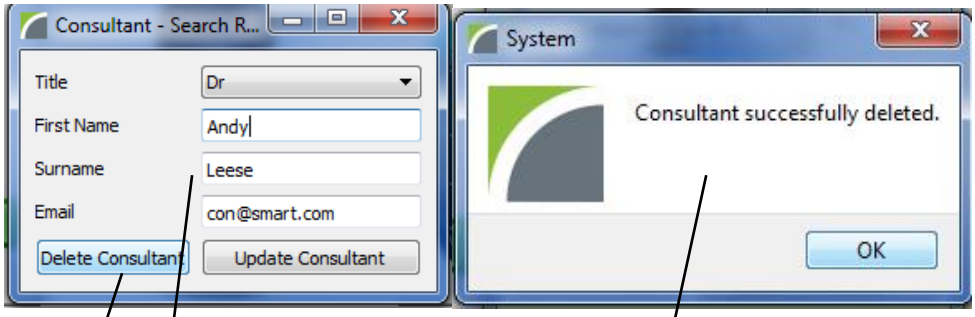
Using the normal test data stated in the test plan I clicked the add code button on in the "Code- Add Code" window.

As expected the system accepted all of the inputs and the data should appear in the database.



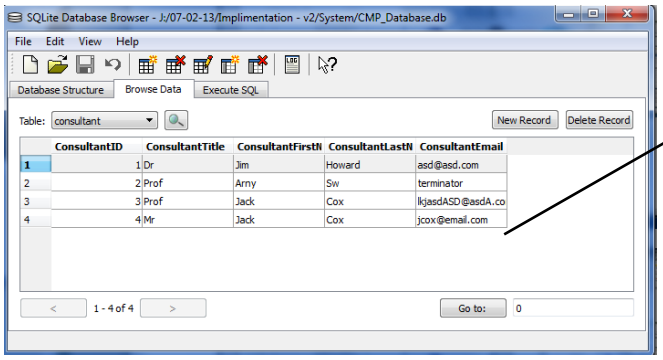
Using "SQLite Database Browser" to view the database more clearly, you can see that the correct Consultant, 'Dr Andy Leese' was stored in the "consultant" entity and assigned an ID of '5'.

Figure 16 - Test 4.9.



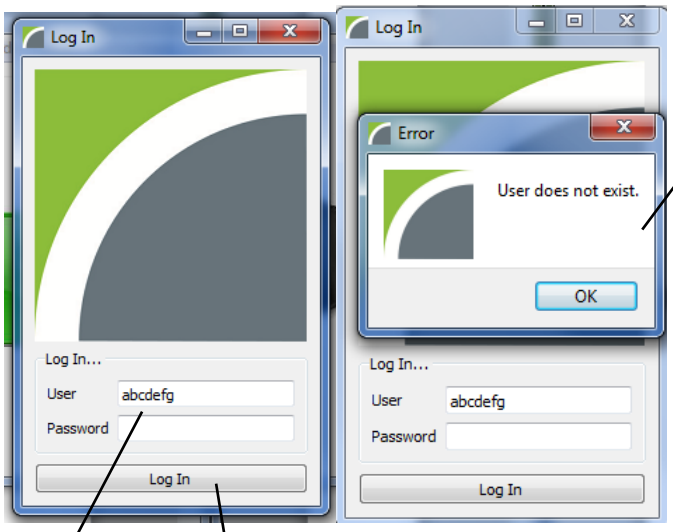
Using the consultant, 'Dr Andy Leese', from the previous test as the stated test data, this window attempts to delete data related to the consultant from the database.

As expected the system accepted all of the inputs and the data should appear in the database.



Using "SQLite Database Browser" to view the database more clearly, you can see that the correct Consultant, 'Dr Andy Leese' was deleted from the "consultant" entity as the record with an ID of '5' no longer exists.

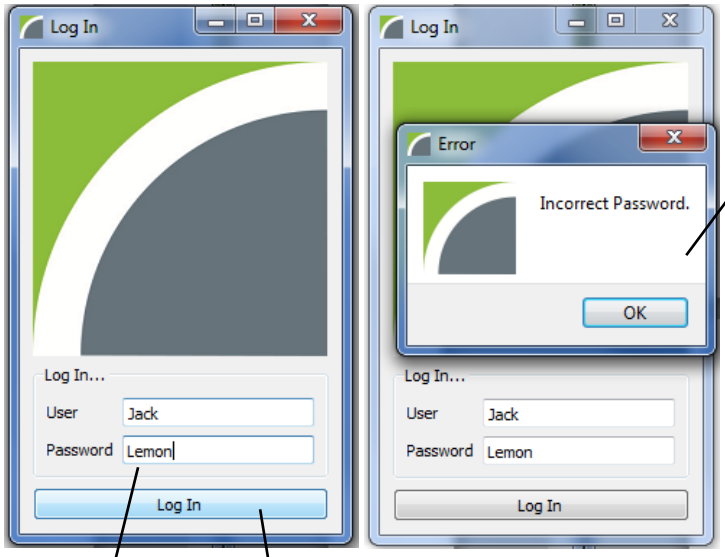
Figure 17 - Test 5.1.



As expected an error window was displayed to inform that the user does not exist.

The non-existent user 'abcdefg' was input into the log in window then the "Log In" button was clicked.

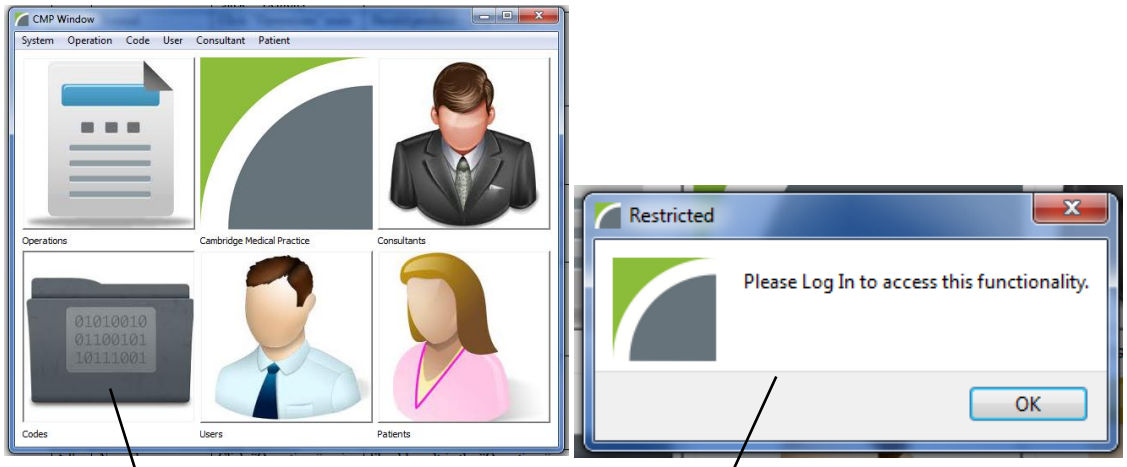
Figure 18 - Test 5.2.



As expected an error window was displayed to inform that the password for this user is incorrect.

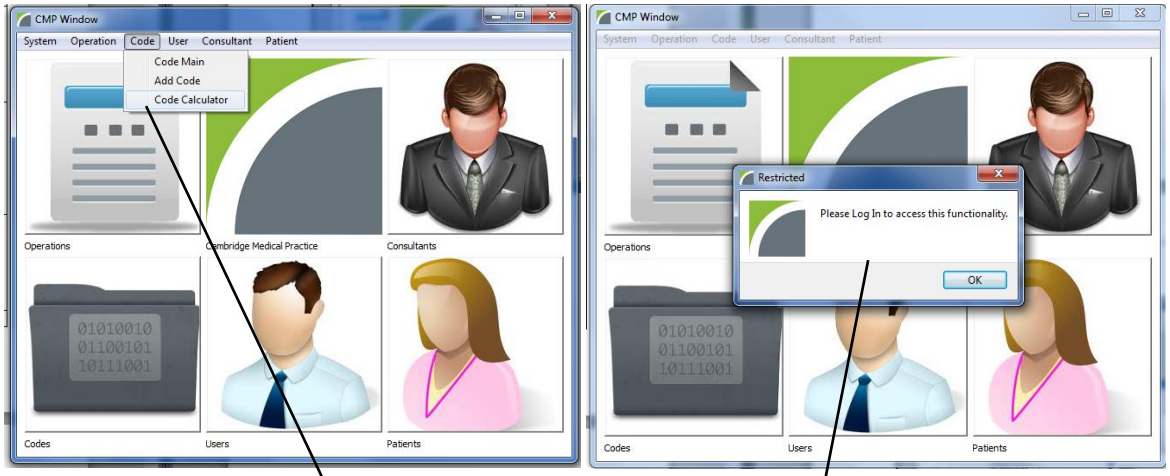
The user 'Jack' exists, but an incorrect password, 'Lemon' was input into the log in window then the "Log In" button was clicked.

Figure 19 - Test 5.4.



After opening the system and not logging in, the "Codes" button on the main menu was clicked.

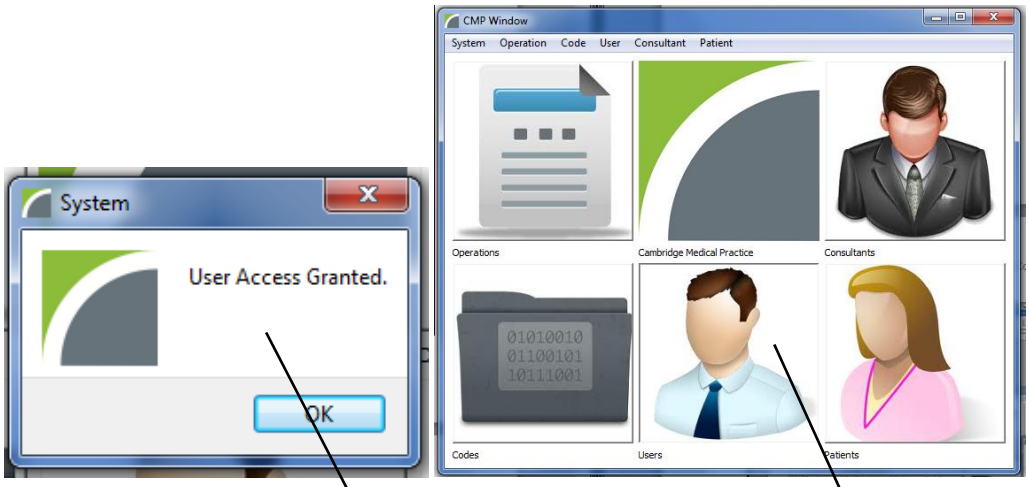
As expected an error window was displayed notifying the user that the feature is restricted without logging in.



After closing the restriction error the "Code Calculator" button from the menu bar was then clicked.

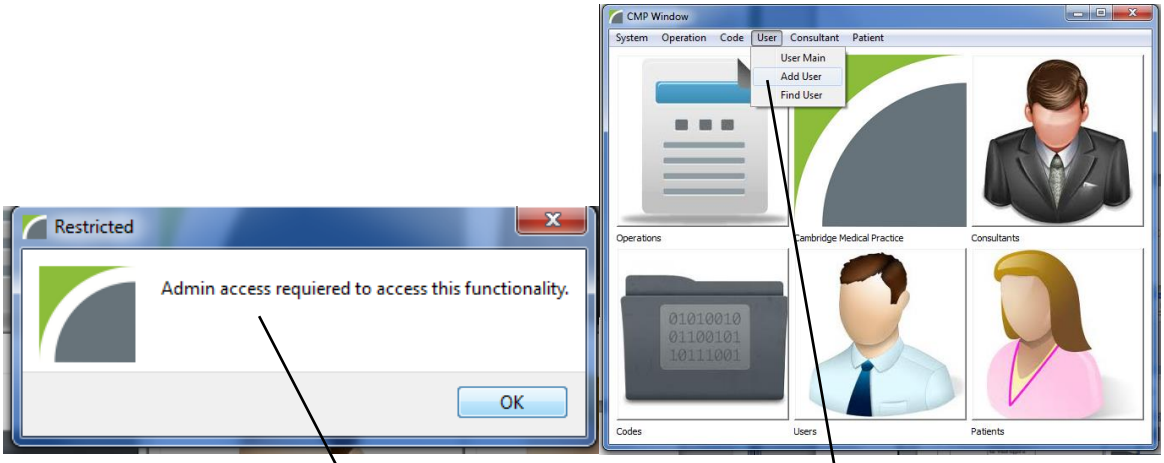
Again ass expected the same restriction error appears asking the user to log in before accessing the functionality.

Figure 20 - Test 5.10.



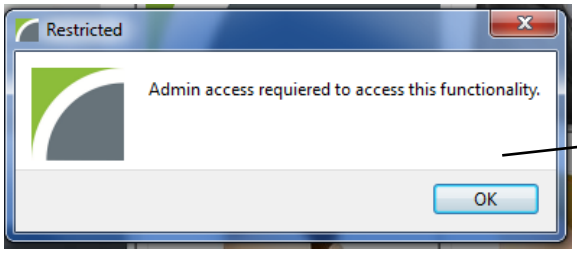
After opening the system I logged in to a user with 'User' level access.

The "Users" button on the main menu was then clicked.



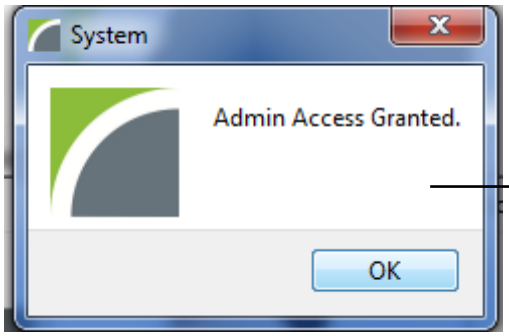
As expected a restriction error was displayed. Although I was logged in, this area of the system requires 'Admin' level access.

After closing the restriction error the "Users" button from the menu bar was then clicked.

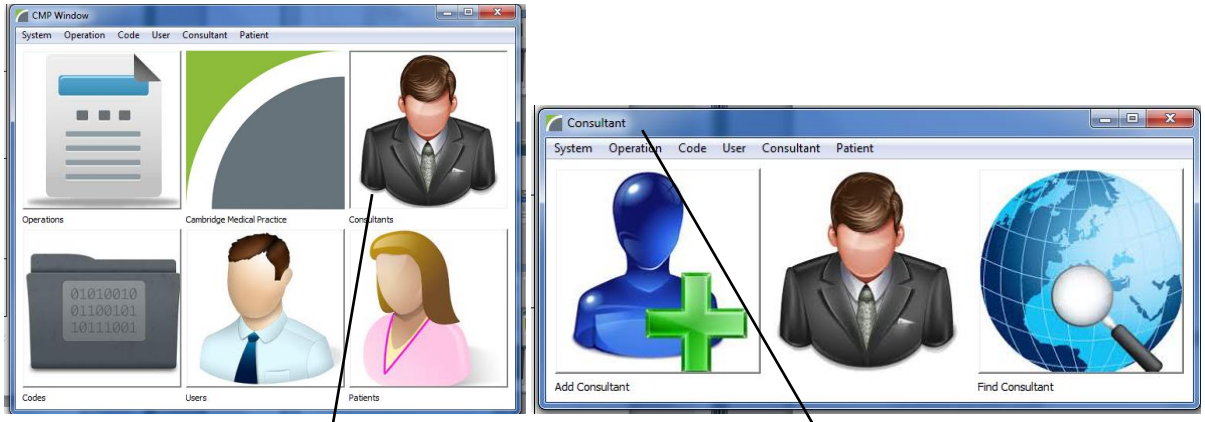


As expected the same restriction error was displayed preventing access to the system area.

Figure 21 - Test 5.17.



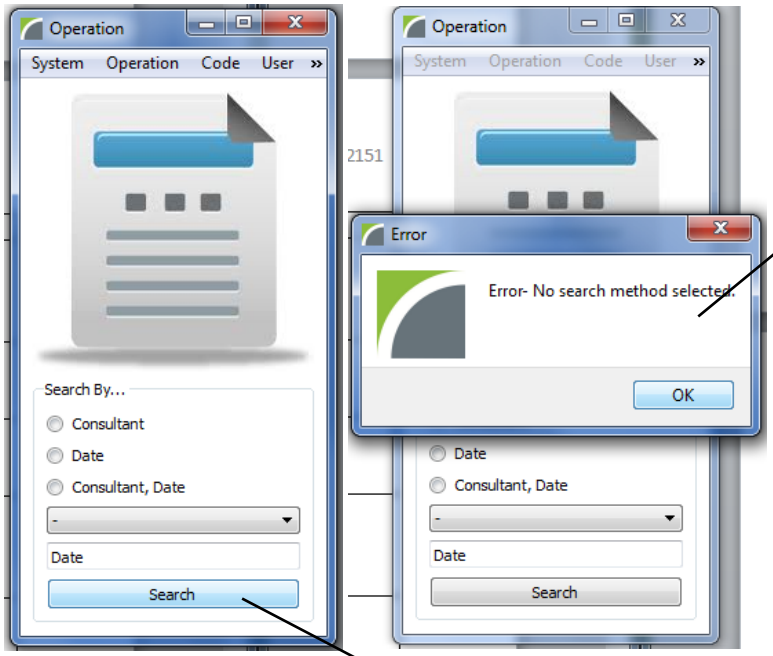
After opening the system I logged in to a user with 'Admin' level access.



The "Consultant" button from the main menu was then clicked.

As expected the "Consultant" window was displayed.

Figure 22 - Test 6.1



As expected an error message was displayed informing the user to select a search method.

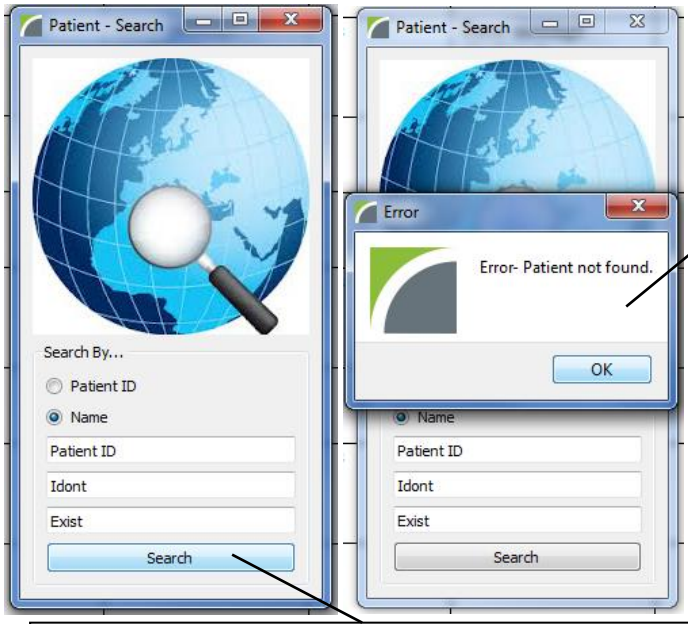
This is the default operations window, where the "Search" button was clicked without making any alterations to the window.

Figure 23 - Test 6.5.

As expected an error message was displayed informing the user that the 'User' does not exist.

After entering the data stated in the test plan, the "Search" button was clicked.

Figure 24 - Test 6.10.



As expected an error message was displayed informing the user that the 'Patient' does not exist.

After entering the data stated in the test plan, the "Search" button was clicked.

4. Evaluation

4.1. Approach to testing

I approached testing with multiple types of testing. For example my series 1 test are examples of white box testing where the navigational structure of the system is tested. Whereas series 2 and 6 demonstrates use of functional testing as they test the validation throughout the user inputs within the system. Series 5 uses security testing to prevent access to the system functionality without a log in. Finally black box testing is used in series 3 to test the calculation methods required for the code calculator. Overall I structured my test plan to an acceptance testing style as it is important the system performs as specified by the client.

4.2. Problems

The main issue with testing was the time given to complete the testing section. With more time I believe that each test could have been run more thoroughly, using repeats and more test data. This could of lead to a discovery of more issues which may not of been highlighted.

During testing I un-earthed some issues with the system causing faulty functionally. These tests are stated below with the actions completed to fix the issues.

Test 2.11. and Test 2.12. –

These were user input validation tests which accepted invalid data into the system. This was a minor issue as all other input fields were valid. After a small investigation I discovered that the regular expressions for these fields were missing, and the problem was quickly solved as the expressions where added.

Test 4.2. –

This test created a more serious problem as it informed me of an inactive database update and incorrect retrieval of data from the database. This was an issue as it is a major functionality of the system and compromised other objectives specified by the client. However the issue was quickly resolved as the problems were being caused by a missing IF statement, and an error in list formatting.

4.3. Strengths

Testing indicated that the system is highly effective as only 3 issues arouse throughout all 128 tests across all 6 test series. It makes clear that all navigation in the system is extremely robust, along with input validation. My tests also point out the accuracy in calculations required within the system. After fixing the error which occurred from Test 4.2. the test plan shows how the system reads and writes from the database accurately and effectively, which is a key feature to system functionality.

I would comment that the testing itself was also high in strength as all areas of the system where tested appropriately using different strategies. This allowed any issues within the system to be highlighted for adaptation, which proved effective as issues were discovered.

4.4. Weaknesses

The main weaknesses in my system occurred in section 4.2 of this document, and now the issues are no longer outstanding there is little weakness within my system. If any the main weakness could occur if a user enters strange or unique valid data which does not comply with my regular expressions and is rejected from the system.

However as I had limit time for testing I would of liked to of run more repeats of the tests in series 2 which required data inputs. Using a wider variety of data would of tested my regular expressions to an extensive amount to ensure only ideal data is passed through the system.

4.5. Reliability

Testing has shown that my system should prove reliable for the client, as all validation is functional which should prevent the user from proving the system with unexpected and unreliable data. This means that no user inputs in any area of the system could cause it to crash or corrupt the database. You can see this in section 3.1, series 2 of this section as it shows the results of my validation testing, and references to the evidence of this. I would say that this makes my system extremely reliable.

4.6. Robustness

During testing I establishes that my system is extremely robust as at no point did it crash as all data past through functions, which could cause a potential crash, have been validated first. At no point did I discover any method of input which could cause a system failure which leads me to believe that my system is robust enough for client usage. The 2 main sources of inputs, button clicks and data inputs, results of testing can be viewed in the first 2 test series of section 3.1.

System Maintenance

1. Environment

1.1. Software

During implementation I used a variety of software to create the system. These are listed below along with versions of the software used:

- Python 3.2 and Python 3.3
- IDLE
- PyQt 4
- SQLite3
- SQLite Database Browser
- Notepad++, version 6.3
- Internet Explorer

1.2. Usage Explanation

The table below states the usage explanation for the software listed in the previous section. All of the software within the table is available to download from the web for no cost which allows no expenditure for the client.

Software	**Usage Explanation**
Python 3.2 and Python 3.3	I used this language as it is the programming language which I learnt over the last two years through college.
IDLE	As this software is included with python it was a familiar development environment which made implementation an easier process.
PyQt 4	The software allows for the creation of a smooth and easily operated graphical interface.
SQLite3	This particular software is included with the python download and allows for a simple but effective single user use database structure to be created.
SQLite Database Browser	A database browser is not required for the client's system however it made observing data transactions during implementation and testing easier to monitor.
Notepad++, version 6.3	Similar to the database browser this software is not required for the client; however it made the creation of html, for invoices and reports, much similar.

Internet Explorer	Used to output reports and invoices as they are generated using html. The clients default browser which removes unnecessary installations and contains a print feature.

1.3. Features Used

Software	Features Used
Python 3.2 and Python 3.3	Python features allowed me to run my system which let me implement and test my system in a CLI or GUI form. This will be used to run my system in its GUI form for my client.
IDLE	I used IDLE to create my code and save it as python file which gave me the benefits of the built in code coloured system which made implementation clearer. I then used the 'Run Module' feature to run my system in Python (above).
PyQt 4	Multiple features from PyQt were used to create the Graphical User Interface (GUI) for the system. I used components such as main windows and widgets to create the GUI.
SQLite3	I used this software to link my entities together ensuring referential integrity was enforced. I then used these to store and retrieve data appropriately from the system. The ON DELETE CASCADE feature was used in some entities to ensure that related data was removed appropriately between linked entities.
SQLite Database Browser	During implementation I used the "Execute SQL" feature to check that my SQL statements would not cause the database to crash. During testing the "Browse Data" functionality was used to monitor the database clearly.
Notepad++, version 6.3	The main reason for using this software was because the language can be changed to html and run easily in any listed browser.
Internet Explorer	Used to display html created from notepad during implementation, and some of the outputs when the system is operational. Documents can be printed from this software which is a key feature.

2. System Overview

2.1. Log In

The log in part of the system prevents an unauthorised user from accessing functionality and prompts them to log in. When presented with a user name and password the system searches the database for the user and if found retrieves the relevant data. The system then checks whether the password matches the username and if so grants the user an appropriate access level ('Admin' or 'User'). If the user was not found or an incorrect password was input the system will display an appropriate error.

2.2. Graphical User Interface

This part of the system provides a graphical display to make the system more user friendly and gives more of a commercial software feel. It contains all navigational methods via buttons and then main tool bar. Operational buttons and input components, such as line edits, are included with the interface allowing the user to commit, retrieve and amend data to the database though the system. The interface displays appropriate labels and images to notify the user of their placement within the system.

2.3. Manage Weekly Operations

This area of the system prompts the user to input a variable on which they wish to view grouped patient data, these variables are date, consultant or both. A table is then output back to the user with two available functions. The calculate function completes the missing data in the table. The invoice function is described in section 2.9. of this document.

2.4. Individual Code Calculator

The individual calculator uses the same methods as the calculate function from the previous section. However as it is not attached to any other areas of the system it requires code and insurance company inputs from the user. The user then receives a surgeon, anaesthetist and total price.

2.5. Patient Data

The two main functions of the system which can be accessed in this area are the addition and location of a particular patient.

[1]The add functionality simple presents the user with an input form which prompts the entry of data. When the “Add” button is clicked the validation module checks for valid input and adds the data to the database or displays an error message.

[2]The search functionality presents the user with a form to input the desired search variable and data. Similar to the add function, when the “Search” button is clicked the validation module validates all user inputs and either progresses to the output window or displays an error message. The output screen retrieves data from the appropriate entities within the database and displays them in the correct components of the window. This then grants the user two new features, update and delete. Delete simply removes the data regarding the displayed record from the database. The update feature allows the user to make amendments to the displayed data. This works by checking for updated data and then using the same validation methods as the add function. Based on the result of validation, the data will be updated in the corresponding entity or an error message will be displayed.

2.6. Code Data

Similar to section 2.5. of this document this system area has the same functionality, addition and location. To see the detailed overview of these system areas please see the marked paragraphs [1] and [2] in the previous section.

2.7. User Data

Similar to section 2.5. of this document this system area has the same functionality, addition and location. To see the detailed overview of these system areas please see the marked paragraphs [1] and [2] in the previous section.

2.8. Consultant Data

Similar to section 2.5. of this document this system area has the same functionality, addition and location. To see the detailed overview of these system areas please see the marked paragraphs [1] and [2] in the previous section.

2.9. Report Output

Within two areas of the system, patients and operations, a printable output is required. These are generated using html which is then automatically opened in internet explorer on the users screen where they can print the invoice or report.

3. Code Structure

3.1. Database Controller Class

The class shown below is the ‘db_controller’ class which is the entire ‘db_controller.py’ module which can also be seen in section 10.12.

```
class db_controller:
    def __init__(self):
        self.dbtext = "CMP_Database.db"

    def query(self,sql):
        self.db = sqlite3.connect(self.dbtext)
        self.cursor = self.db.cursor()
        self.cursor.execute("PRAGMA foreign_keys = ON")
        self.cursor.execute(sql)
        self.db.commit()
        self.cursor.close()

    def select_query(self,sql):
        self.db = sqlite3.connect(self.dbtext)
        self.cursor = self.db.cursor()
        self.cursor.execute("PRAGMA foreign_keys = ON")
        self.cursor.execute(sql)
        results = self.cursor.fetchall()
        self.cursor.close()
        return results
```

This code is structured as a class so that it can parent the other controllers in the system. I did this the prevent duplication of code as all controllers require the two functions within this class. As well as saving time this also reduces errors and still provides all controllers with the ability to write and read the database.

3.2. Insurance Company Class: Add Insurance Company Function

```
class Insurance_Company_Controller(db_controller):
    """Creates controller for Insurance Companies"""
    def __init__(self):
        super().__init__()
```

```
    def add_Insurance_Company(self,name):
        sql = """insert into insuranceCompany (InsuranceCompanyName) values
('{0}')""".format(name)
        self.query(sql)
```

Shown above is the 'Insurance_company_controller' class and its first function 'add_Insurance_Company'. The class inherits from the 'db_controller.py' module as it is one of the controllers. The reason this class contains a function for adding a new insurance company is so it can be called from other modules such as the GUI modules.

3.3. Validation Function

```
def validate_data(data,pattern):
        if re.match(pattern,data):
            return True
        else:
            return False
```

Shown above is the 'validate_data' function which is part of the 'gui_validation.py' module which can be seen in section 10.13. The purpose of this function is to match a piece of data with a provided regular expression and return whether the data is valid or not. It was implemented as a separate function to save repetition of code, as it is called by various functions in the same module multiple times.

3.4. Main: Code Function

```
def Code(self):
        if self.access != None:
            self.codeLayout = code_gui.codeWindow()
            self.setCentralWidget(self.codeLayout)
            self.setWindowTitle("Code")
        else:
            QMessageBox.about(self, "Restricted", "Please Log In to access
this functionality.")
```

Shown above is the 'Code' function which is part of the 'main_gui.py' module which can be seen in section 10.17. This module contains lots of similar functions as each button from the main menu is linked to one. The functions check access levels the display the appropriate window, in this case the 'codeWindow', or display an error message.

3.5. Code: Delete Function

```
def delete(self):
        tempController = code_controller.Code_Controller()
        try:
            tempController.delete_code(self.id)
            QMessageBox.about(self, "System", "Code successfully deleted.")
            self.close()
        except:
            QMessageBox.about(self, "Error", "Code can not be deleted as it
is\nrequired for a Patient's record.")
```

Shown above is the 'delete' function which is part of the 'output_code.py' module which can be seen in section 10.20. This function displays an excellent example of exception handling as is the data is none existent the database can lock itself and become useless.

3.6. Module Main Function

```
if __name__ == "__main__":
```

```
application = QApplication(sys.argv)
codeWindow = codeWindow()
codeWindow.show()
codeWindow.raise_()
application.exec_()
```

The code above shows an example of the IF statement which is run at the start of almost every module. This allows the module to be run individually which is extremely useful for implementation and testing purposes.

4. Variable Listing

The table below contains a list of all the variables used throughout the system. The variables from the CLI functions are not included as the GUI overwrites these.

Variable Name	Purpose	Section	Line Numbers
code	Stores the code name as a string.	10.1.	94, 132, 136
cost	Stores the 14 costs related to a code as an array.	10.1.	95, 103, 104, 106, 128,136
count	A stepper variable used in FOR loops	10.1.	102, 103, 104, 106, 110, 111, 112, 114, 117, 118
invalid	Stores the input invalid data as a string, and then is used to store each item in the string in an array.	10.1.	107, 108, 113
letterlist	Stores the letters from the invalid string in an array.	10.1.	109, 112, 114, 117, 118
currentcode	Temporary store of a code compiled from the 'letterlist'.	10.1.	116
self.valid	Boolean variable which stores the current state of the data.	10.1.	120, 122, 125, 126, 129
self.errormsg	Stores a string containing the current error message	10.1.	123, 127
self.tempController	An instantiation of the Code Controller	10.1.	131, 132, 136
data	Stores returned data from the controller	10.1.	132, 133
self.valid	Boolean variable which stores the current state of the data.	10.2.	46, 49
self.errormsg	Stores a string containing the current error message	10.2.	46, 50
tempController	An instantiation of the Consultant Controller	10.2.	52, 54
tempController	An instantiation of the Consultant Controller	10.3.	106, 107
self.conData	Stores returned data from the controller	10.3.	107, 110
item	A stepper variable used in FOR	10.3.	110, 111, 134, 135,

	loops		138, 140, 141
con	Stores the compiled consultant name	10.3.	111, 112
tempController	An instantiation of the Patient Controller	10.3.	115, 153, 155
letterlist	Stores the letters from the invalid string in an array.	10.3.	116, 119, 122, 125, 132, 138, 140
count	A stepper variable used in FOR loops	10.3.	117, 122
self.codes	Stores an array of code names	10.3.	120, 127, 149, 160
self.valid	Boolean variable which stores the current state of the data.	10.3.	121, 129, 142, 143, 150
currentcode	Temporary store of a code compiled from the 'letterlist'.	10.3.	124, 127
self.errormsg	Stores a string containing the current error message	10.3.	130, 143, 151
con	Stores the compiled consultant name	10.3	131, 134
step	A stepper variable used in FOR loops	10.3.	133, 136, 137
conlname	Stores the consultant last name as a string	10.3.	139, 141, 148, 159
item	A stepper variable used in FOR loops	10.4.	30
self.valid	Boolean variable which stores the current state of the data.	10.4.	46, 49
self.errormsg	Stores a string containing the current error message	10.4.	46, 50
tempController	An instantiation of the User Controller	10.4.	52, 53
costs	Stores an array of 2 prices as integers	10.5.	4, 5, 6, 7, 8, 9, 10, 12, 13, 17, 18, 19, 20, 21, 22, 23, 25, 26, 30, 31, 32, 33, 35, 36, 95, 97, 99, 101
sp	Stores the surgeon price as an integer	10.5.	6, 9, 12, 14, 19, 22, 25, 27, 32, 35, 37, 97, 99, 101, 102
ap	Stores the anaesthetist price as an integer	10.5.	7, 10, 13, 14, 20, 23, 26, 27, 33, 36, 37, 97, 99, 101,102
controller	An instantiation of the Code Controller	10.5.	39, 40, 70, 71
codes	An array holding a code names	10.5.	39, 40, 42, 44, 45, 47, 50, 70, 71, 91, 93, 95, 108
IC	Stores the insurance company name as a string	10.5.	39, 40, 91, 92, 96, 98, 100, 109, 110

un_costs	Stores an array of costs which need to be ordered	10.5.	40, 43, 46, 49,
exist	Stores a Boolean value to indicate whether the costs are existent.	10.5.	40, 41
or_costs	Stores an array of costs which are in the process of being sorted and the final sorted array	10.5.	43, 46, 49, 51, 52, 53, 54, 55, 56, 61,62, 65
count	A stepper variable used in FOR loops	10.5.	51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63, 64, 73, 75, 79, 81, 82
temp	A temporary variable used to hold a costs during a swap	10.5.	54, 56, 57, 59
comp	Stores the compatibility of codes in a list.	10.5.	71, 71, 75
invalid	Stores all incompatible pairs as a list.	10.5.	72, 77, 80, 81
subinvalid	Stores a single incompatible pair which is added to ‘invalid’.	10.5.	74, 76, 77
item	A stepper variable used in FOR loops	10.5.	75, 76, 83, 85, 86, 88
step	A stepper variable used in FOR loops	10.5.	80, 81, 82
codes	Stores a list of code names as strings	10.6.	70, 72, 74, 76, 78, 81
valid	Boolean, determines if the data is valid or not	10.6.	78, 79
errormsg	Stores the error message as a string	10.6.	78
sp	Stores the surgeon price as an integer	10.6.	81, 82, 84
ap	Stores the anaesthetist price as an integer	10.6.	81, 83, 84
sql	Stores a SQLite instruction as a string.	10.7.	9, 10, 11, 12, 21, 25, 2841, 45, 50, 56, 59, 61, 64, 65, 74, 76, 82, 84, 91, 93, 99, 101, 107, 109, 111, 114, 116, 118, 120, 124, 141, 145, 150, 152, 157, 159, 162, 164, 174, 176, 179, 181, 186, 188
IDtuple	Stores the tuple of data retrieved from the database.	10.7.	12, 13, 25, 27, 35, 36, 39, 65, 67, 68, 152, 153, 155, 176, 177, 181, 182, 184

IDstr	Stores the data from the tuple as a string.	10.25.	130, 131, 136, 137, 142, 143
ID	Stores the ID of a database record as an integer.	10.7.	16, 44, 47, 58, 60, 68, 69, 70, 71, 73, 75, 87, 92, 97, 100, 108, 117, 144, 156, 163, 178, 185, 187
invalid	Stores all incompatible pairs as a list.	10.7.	8, 15, 16, 19, 69, 71, 97, 115, 119, 188, 189, 221
item	A stepper variable used in FOR loops	10.7.	20, 22, 31, 34, 78, 79, 81, 83, 119, 121, 130, 133, 149, 151, 154, 167, 173, 175, 184, 185
existing	Stores a Boolean value to indicate whether the search is valid.	10.7.	23, 24, 26, 38, 122, 123, 125, 138
count	A stepper variable used in FOR loops	10.7.	48, 49, 55, 90, 91, 104, 105, 106, 108, 110, 113
IC	Stores the insurance company name as a string	10.7.	49, 55, 88, 92, 105, 108, 113, 147, 158
costs	Stores an array of prices as integers	10.7.	47, 55, 70, 71, 89, 94, 95
codes	An array holding a code names	10.7.	80, 85, 147, 149, 171, 173
icList	Array of the 7 insurance company names	10.7.	97, 102, 106, 108, 110, 113
costList	Stores an array of prices as integers	10.7.	148, 168, 169
incompatible	Stores an array of lists of incompatible code pairs.	10.7.	172, 190, 192, 193
invalid	Stores all incompatible pairs as a list.	10.7.	8, 15, 16, 19, 20, 69, 71, 97, 115, 119, 188, 189
subinvalid	Stores a single incompatible pair which is added to ‘invalid’.	10.7.	183, 189, 190
valid	Determines whether or not input data is valid by storing a Boolean	10.8.	57, 59
errormsg	Stores the error message as a string	10.8.	57, 67
existing	Stores a Boolean value to indicate whether the search is valid.	10.8.	60, 61
data	Stores tuple of data returned from the database via a controller	10.8.	71, 72
code	An string holding a code name	10.8.	69, 71
sql	Stores a SQLite instruction as a	10.9.	9, 15, 18, 20, 23,

	string.		25, 28, 31, 34, 36, 37, 42, 43, 55,57, 58, 59, 60
title	Stores the 'Title' input as a string	10.9.	8, 14, 45, 47, 48
fname	Stores the 'First Name' input as a string	10.9.	8, 14, 22, 27, 29, 33, 35, 45, 49, 50
lname	Stores the 'Last Name' input, as a string	10.9.	8, 14, 22, 30, 33, 35, 45, 51, 52
email	Stores the 'Email' input, as a string	10.9.	8, 14, 45, 53, 54
ID	Stores the ID of a database record as an integer.	10.9.	17, 19, 22, 26, 45, 59
updates	Stores a list of fields and data to be included in a sql statement.	10.9.	46, 48, 50, 52, 54, 56
item	A stepper variable used in FOR loops	10.9.	56, 57
db	Stores the path to the database	10.11.	3, 14, 22, 47, 58, 71, 87, 95, 106, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132
cursor	The database navigator.	10.11.	3, 14, 22, 47, 58, 71, 87, 95, 106, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132
sql	Stores a SQLite instruction as a string.	10.11.	4, 11, 15, 19, 23, 44, 55, 59, 68, 72, 84, 88, 92, 96, 103, 107, 118
sql	Stores a SQLite instruction as a string.	10.12.	7, 11, 15, 19
self.db	Stores the path to the database	10.12.	8, 12, 16
self.cursor	The database navigator.	12.12.	9, 10, 11, 13, 17, 18, 9, 20, 21
db_controller	The class containing the connection methods for communicating with the database	10.14.	3
name	The name of an insurance company as a string	10.14.	8, 10
sql	Stores a SQLite instruction as a string.	10.14.	9, 11, 14, 16, 19, 21
ID	Stores the ID of a database record as an integer.	10.14.	13, 15, 18, 20
ic	Stores the insurance company name as a string	10.14.	21, 22
sp	Stores the surgeon price as an integer	10.15.	27, 29, 30
ap	Stores the anaesthetist price as an	10.15.	27, 30

	integer		
date	A date stored as a string.	10.15.	35, 37, 38
strdate	A formatted version of 'date' stored as a string	10.15.	36
codelist	An array holding a code names	10.15.	41, 43, 44
item	A stepper variable used in FOR loops	10.15.	42, 43, 48, 49
codes	An array holding code names	10.15.	40, 42
codeBullets	A string containing the html to generate a html list of codes	10.15.	47, 49, 67
html	A html string	10.15.	50, 69
htmlfile	The file in which the html string is written to	10.15.	68, 69, 70
new	The browser choice in which to open the html file	10.15.	73, 75
url	The combination of the html file and a path to the browser	10.15.	74, 75
tempController	An instantiation of a controller class.	10.15.	78, 79
invoice	An instantiation of the invoice class	10.15.	82, 83, 84, 88, 89, 90, 91
errormsg	Stores the error message as a string	10.15.	84, 85, 86
details	An array of lists, each list stores data regarding 1 record	10.15.	7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 81, 82, 83
db_controller	The class containing the connection methods for communicating with the database	10.16.	3
date	A date stored as a string.	10.16.	8, 12
pID	Stores the ID of a patient record as an integer.	10.16.	8, 12
sql	Stores a SQLite instruction as a string.	10.16.	9, 13
self.data	Stores tuple of data returned from the database via a controller	10.17.	215, 218, 220, 222, 228, 236
self.access	Stores the access level of the system as a string, either 'None', 'User' or 'Admin'	10.17.	18, 174, 177, 222, 234, 245, 254, 263, 273, 282, 291, 300, 309, 318, 328, 337, 346
tempController	An instantiation of a controller class.	10.18.	33, 34
item	A stepper variable used in FOR loops	10.18.	37, 38, 95, 96, 98, 99, 100, 103, 104, 106, 107
con	Stores a tuple containing data regarding a consultant record	10.18.	38, 39

existing	Stores a Boolean value to indicate whether the search is valid.	10.18.	56, 57
valid	Determines whether or not input data is valid by storing a Boolean	10.18.	58, 60
errormsg	Stores the error message as a string	10.18.	58, 67
date	Boolean to indicate whether a particular radio button is checked	10.18.	71, 77
both	Boolean to indicate whether a particular radio button is checked	10.18.	72, 81
letterlist	Stores the letters from the invalid string in an array.	10.18.	92, 99, 103
step	A stepper variable used in FOR loops	10.18.	94, 97, 98, 100
confname	Stores the 'First Name' input as a string	10.18.	102, 104, 108
conlname	Stores the 'Last Name' input, as a string	10.18.	105, 107, 108
tempController	An instantiation of a controller class.	10.19.	17,18
costs	Stores an array of prices as integers	10.19.	21, 31, 32
count	A stepper variable used in FOR loops	10.19.	22, 23, 24, 38, 39, 65, 68, 69, 70
IC	Stores the insurance company name as a string	10.19.	23, 29
codes	An array holding code names	10.19.	24, 26
codeList	An array holding code names	10.19.	25, 28, 29
item	A stepper variable used in FOR loops	10.19.	26, 27, 40, 41, 46, 49, 50
code	An string holding a code name	10.19.	27, 28
sp	Stores the surgeon price as an integer	10.19.	29, 30
ap	Stores the anaesthetist price as an integer	10.19.	29, 30
patient	A tuple containing data about a particular patient record	10.19.	41, 43, 44
details	Stores a SQLite instruction as a string.	10.19.	42, 43, 45, 46, 47
name	A compile name consisting of a title, first name and surname, as a string	10.19.	44, 45, 50, 51
step	A stepper variable used in FOR loops	10.19.	48, 51, 52
UN	'Username' data input, as a string	10.19.	56, 77
date	A date stored as a string.	10.19.	57, 58, 59, 77
strdate	A formatted version of 'date'	10.19.	58, 59

	stored as a string		
title	report not person	10.19.	61, 63
data	Stores tuple of data returned from the database via a controller	10.19.	64, 71, 77
currentData	Temporary store of a patient's data as a list.	10.19.	66, 71
html	A html string	10.19.	72, 79
htmlfile	The file in which the html string is written to	10.19.	78, 79, 80
new	The browser choice in which to open the html file	10.19.	83, 85
url	The combination of the html file and a path to the browser	10.19.	84, 85
code	A single code name as a string	10.20.	8, 10, 12, 13, 15, 20, 22, 40, 44, 124, 126, 165, 169
item	A stepper variable used in FOR loops	10.20.	24, 25, 29, 30, 32, 33
currentCost	A single cost, used in loops when iterating through multiples	10.20.	31, 32, 33, 34
tempController	An instantiation of a controller class.	10.20.	122, 169, 174, 176
costList	A list of costs as integers	10.20.	127, 136, 140
cost	Stores an array of prices as integers	10.20.	134, 137, 139, 141, 143
count	A stepper variable used in FOR loops	10.20.	135, 136, 137, 140, 141, 147, 148, 150, 152, 155, 156
valid	Determines whether or not input data is valid by storing a Boolean	10.20.	144, 159, 163, 164, 166, 168
letterlist	Stores the letters from the invalid string in an array.	10.20.	146, 150, 152, 155, 156
currentcode	Temporary store of a code compiled from the 'letterlist'.	10.20.	154, 157
errormsg	Stores the error message as a string	10.20.	160, 164, 167
invalid	Stores all incompatible pairs as a list.	10.20.	151, 157, 162, 165, 169
search	The variable on which to base a search, stores as a string	10.21.	8, 10, 12, 13, 20, 22
data	A string of data that is to be searched for	10.21.	8, 10, 12, 13, 20, 23, 26
tempController	An instantiation of a controller class.	10.21.	21, 23, 26, 86, 88, 94, 96
count	A stepper variable used in FOR loops	10.21.	36, 37, 38, 39
title	Stores the 'Title' input as a string	10.21.	61, 63, 77, 82, 88
fname	Stores the 'First Name' input as a string	10.21.	26, 77, 82, 88

lname	Stores the 'Last Name' input, as a string	10.21.	26, 69, 71, 77, 82, 88
email	Stores the 'Email' input, as a string	10.21.	73, 75, 77, 82, 88
update	Stores a list of fields and data to be included in a sql statement.	10.21.	76, 80, 85
valid	Determines whether or not input data is valid by storing a Boolean	10.21.	81, 83, 85
errormsg	Stores the error message as a string	10.21.	81, 84
search	The variable on which to base a search, stores as a string	10.22.	7, 10, 12, 13, 22, 23, 24, 26, 71
data	A string of data that is to be searched for	10.22.	7, 10, 14, 22, 24, 26, 71
user	Stores the current active username as a string	10.22.	7, 9, 71
row	A stepper variable used in FOR loops	10.22.	37, 38, 41, 45, 60, 62, 65
col	A stepper variable used in FOR loops	10.22.	38, 39, 41, 45
search	The variable on which to base a search, stores as a string	10.23.	10, 12, 14, 15, 22, 24
data	A string of data that is to be searched for	10.23.	10, 12, 14, 15, 22, 25, 28
tempController	An instantiation of a controller class.	10.23.	23, 25, 28, 134, 135, 259, 261, 268, 269
count	A stepper variable used in FOR loops	10.23.	56, 57, 58, 59, 63, 64, 65, 66, 71, 72, 73, 74, 138, 139, 140, 142, 143, 145, 225, 226, 227, 229, 232, 233
con	Stores the compiled consultant name	10.23.	201, 204
title	Stores the 'Title' input as a string	10.23.	153, 154, 156, 241, 251, 254, 261
fname	Stores the 'First Name' input as a string	10.23.	28, 158, 160, 241, 251, 254, 261
lname	Stores the 'Last Name' input, as a string	10.23.	28, 161, 162, 164, 241, 251, 254, 261
ad1	Stores the 'Address Line 1' input, as a string	10.23.	166, 168, 241, 251, 254, 261
ad2	Stores the 'Address Line 2' input, as a string	10.23.	170, 171, 242, 251, 254, 261
town	Stores the 'Town' input, as a string	10.23.	174, 176, 241, 251, 254, 261
pc	Stores the 'Post Code' input, as a string	10.23.	178, 180, 242, 251, 254, 261

DoB	Stores the 'Date of Birth' input, as a string	10.23.	182, 184, 242, 251, 254, 262
gender	Stores the 'Gender' input, as a string	10.23.	186, 188, 243, 251, 254, 262
telNo	Stores the 'Telephone Number' input, as a string	10.23.	190, 192, 243, 251, 255, 262
email	Stores the 'Email' input, as a string	10.23.	194, 196, 243, 251, 255, 262
DoO	Stores the 'Date of Operation' input, as a string	10.23.	198, 200, 243, 251, 255, 262
letterlist	Stores the letters from the invalid string in an array.	10.23.	202, 208, 210, 224, 227, 229, 232, 233
step	A stepper variable used in FOR loops	10.23.	203, 206, 207
item	A stepper variable used in FOR loops	10.23.	204, 205, 207, 208, 210, 211
conlname	Stores the 'Last Name' input, as a string	10.23.	209, 211, 212, 213, 125, 244, 251, 255, 262
icName	Stores the insurance company name as a string	10.23.	217, 219, 244, 252, 255, 262
valid	Determines whether or not input data is valid by storing a Boolean	10.23.	220, 236, 247, 249, 253, 256
codes	An array holding code names	10.23.	222, 223, 225, 226, 228, 234, 239, 244, 252, 255, 262
currentcode	Temporary store of a code compiled from the 'letterlist'.	10.23.	231, 234
errormsg	Stores the error message as a string	10.23.	237, 248, 253, 257
update	Stores a list of fields and data to be included in a sql statement.	10.23.	240, 246, 249
search	The variable on which to base a search, stores as a string	10.24.	8, 10, 19, 21
data	A string of data that is to be searched for	10.24.	8, 10, 12, 13, 19, 22, 24
tempController	An instantiation of a controller class.	10.24.	20, 22, 24, 84, 85, 91, 92
count	A stepper variable used in FOR loops	10.24.	38, 39, 40, 41
update	Stores a list of fields and data to be included in a sql statement.	10.24.	60, 79, 83
UN	Stores the 'Username' input, as a string	10.24.	24, 62, 64, 77, 80, 85
PW	Stores the 'Password' input, as a string	10.24.	66, 68, 77, 80, 85
email	Stores the 'Email' input, as a string	10.24.	70, 72, 77, 80, 85
AC	Stores the 'Access' input, as a	10.24.	74, 76, 77, 80, 85

	string		
valid	Determines whether or not input data is valid by storing a Boolean	10.24.	80, 81, 83
errormsg	Stores the error message as a string	10.24.	80, 82
title	Stores the 'Title' input as a string	10.25.	11, 35, 210, 213, 214
fname	Stores the 'First Name' input as a string	10.25.	11, 35, 74, 80, 82, 83, 86, 87, 120, 124, 126, 128, 132, 134, 184, 210, 215, 216
lname	Stores the 'Last Name' input, as a string	10.25.	11, 35, 39, 74, 80, 82, 83, 88, 90, 91, 114, 120, 124, 126, 128, 138, 140, 175, 284, 210, 211, 217, 218
ad1	Stores the 'Address Line 1' input, as a string	10.25.	11, 35, 210, 219, 220
ad2	Stores the 'Address Line 2' input, as a string	10.25.	11, 35, 210, 221, 222
town	Stores the 'Town' input, as a string	10.25.	11, 35, 210, 223, 224
pc	Stores the 'Post Code' input, as a string	10.25.	11, 35, 210, 225, 226
DoB	Stores the 'Date of Birth' input, as a string	10.25.	11, 35, 210, 227, 228
gender	Stores the 'Gender' input, as a string	10.25.	11, 35, 210, 229, 230
telNo	Stores the 'Telephone Number' input, as a string	10.25.	11, 35, 39, 210, 231, 232
email	Stores the 'Email' input, as a string	10.25.	11, 36, 210, 233, 234
DoO	Stores the 'Date of Operation' input, as a string	10.25.	11, 36, 211, 235, 236
conLname	Stores the 'Last Name' input, as a string	10.25.	12, 14, 211, 244, 246
icName	Stores the insurance company name as a string	10.25.	12, 21, 211, 253, 255
codes	An array holding code names	10.25.	12, 43, 46, 47, 79, 83, 87, 91, 109, 147, 149, 198, 201, 206, 211, 262, 264
sql	Stores a SQLite instruction as a string.	10.25.	13, 15, 20, 22, 27, 37, 38, 40, 48, 50, 55, 59, 64, 66, 75, 77, 81, 85,89, 93, 96, 98, 99, 116,

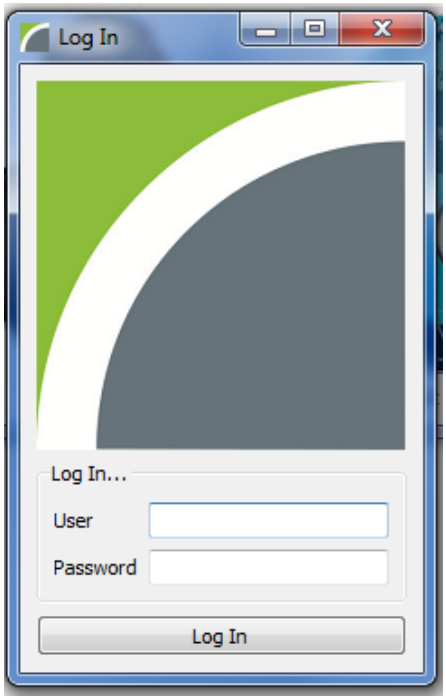
			119, 121, 135, 127, 129, 133, 135, 139, 141, 145, 147, 151, 177, 180, 186, 188, 238, 240, 241, 242, 245, 247, 150, 252, 254, 256, 259, 161
item	A stepper variable used in FOR loops	10.25.	47, 49, 52, 104, 105, 106, 107, 149, 150, 191, 194, 195, 199, 200, 239, 240
ID	Stores the ID of a database record as an integer.	10.25.	62, 63, 65, 68, 70, 74, 78, 79, 114, 115, 117, 124, 144, 146, 150, 152, 160, 168, 200
date	A date stored as a string.	10.25.	74, 95, 97
details	An array of lists, each list stores data regarding 1 record	10.25.	99, 105, 107, 108, 109, 110, 121, 122, 165, 168,
IDtuple	Stores the tuple of data retrieved from the database.	10.25.	129, 130, 135, 136, 141, 142
IDstr	Stores the data from the tuple as a string.	10.25.	130, 131, 136, 137, 142, 143
tempController	An instantiation of a controller class.	10.25.	159, 160, 166, 168, 172, 175, 184, 192, 194
con	Stores the compiled consultant name	10.25.	160, 161, 162, 163
ic	Stores the insurance company name as a string	10.25.	168, 169, 194, 196
data	Stores tuple of data returned from the database via a controller	10.25.	171, 175, 181, 184, 187, 202, 203, 204, 205, 206, 207
updates	Stores a list of fields and data to be included in a sql statement.	10.25.	212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 237, 239
existing	Stores a Boolean value to indicate whether the search is valid.	10.27.	50, 51, 56, 57,
valid	Determines whether or not input data is valid by storing a Boolean	10.27.	52, 55
errormsg	Stores the error message as a string	10.27.	52, 64
ID	Stores the ID of a database record as an integer.	10.27.	67, 69, 86
name	Stores a compiled name as a string	10.27.	68, 73

tempController	An instantiation of a controller class.	10.27.	84, 86, 89
existing	Stores a Boolean value to indicate whether the search is valid.	10.28.	50, 51, 55, 56
valid	Determines whether or not input data is valid by storing a Boolean	10.28.	52, 54
errormsg	Stores the error message as a string	10.28.	52, 63
ID	Stores the ID of a database record as an integer.	10.28.	66, 68, 85
name	Stores a compiled name as a string	10.28.	67, 72
tempController	An instantiation of a controller class.	10.28.	83, 85, 88
data	Stores tuple of data returned from the database via a controller	10.28.	85, 87, 89
existing	Stores a Boolean value to indicate whether the search is valid.	10.29.	48, 49, 53, 54
valid	Determines whether or not input data is valid by storing a Boolean	10.29.	50, 52
errormsg	Stores the error message as a string	10.29.	50, 61
ID	Stores the ID of a database record as an integer.	10.29.	64, 66, 82
UN	Stores the ‘Username’ input, as a string	10.29.	65, 70, 84
tempController	An instantiation of a controller class.	10.29.	80, 82, 84
data	Stores tuple of data returned from the database via a controller	10.29.	82, 84, 85
UN	Stores the ‘Username’ input, as a string	10.30.	8, 12, 20, 24, 26, 32, 34, 35
PW	Stores the ‘Password’ input, as a string	10.30.	8, 12, 32, 36, 37
AC	Stores the ‘Access’ input, as a string	10.30.	8, 12, 32, 38, 39
email	Stores the ‘Email’ input, as a string	10.30.	8, 12, 32, 40, 41
sql	Stores a SQLite instruction as a string.	10.30.	9, 13, 16, 18, 21, 23, 25, 27, 28, 42, 44, 45, 46, 47
ID	Stores the ID of a database record as an integer.	10.30.	15, 17, 20, 22, 32, 46
updates	Stores a list of fields and data to be included in a sql statement.	10.30.	33, 35, 37, 39, 41, 43
item	A stepper variable used in FOR loops	10.30.	43

`data`	Stores a single piece of input data	10.32.	3, 8
`encryptor`	The variable that converts data from normal to encrypted	10.32.	4
`encrypt_data`	The encrypted data	10.32.	5
`decryptor`	The variable that converts data from encrypted to normal	10.32.	9
`decrypt_data`	The normal data	10.32.	10
`switch`	A string which decides which method to call, encryption or decryption	10.32.	13, 15, 20, 25, 30
`UN`	Stores the 'Username' input, as a string	10.32.	13, 14, 16, 18, 34
`PW`	Stores the 'Password' input, as a string	10.32.	13, 19, 21, 23, 34
`AC`	Stores the 'Access' input, as a string	10.32.	13, 24, 26, 28, 34
`email`	Stores the 'Email' input, as a string	10.32.	13, 29, 31, 33, 34
`tempController`	An instantiation of a controller class.	10.33.	4, 5
`valid`	Determines whether or not input data is valid by storing a Boolean	10.33.	9, 10, 19
`username`	Stores the 'Username' input, as a string	10.33.	11
`password`	Stores the 'Password' input, as a string	10.33.	12, 17
`details`	Stores a SQLite instruction as a string.	10.33.	13, 14, 17, 22

5. System Evidence

5.1. User Interface

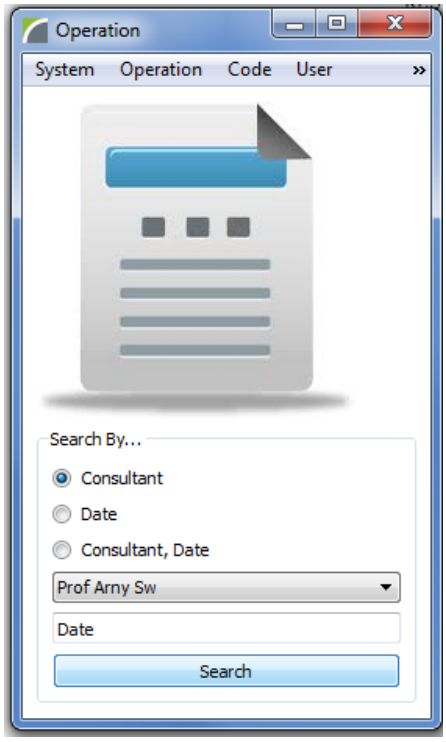


This is the log in window, it consists of:

An image of the company logo

2 labels and 2 line edits within a group box

A log in button



This is the Operation window, it consists of:

The standard menu bar for all main system areas

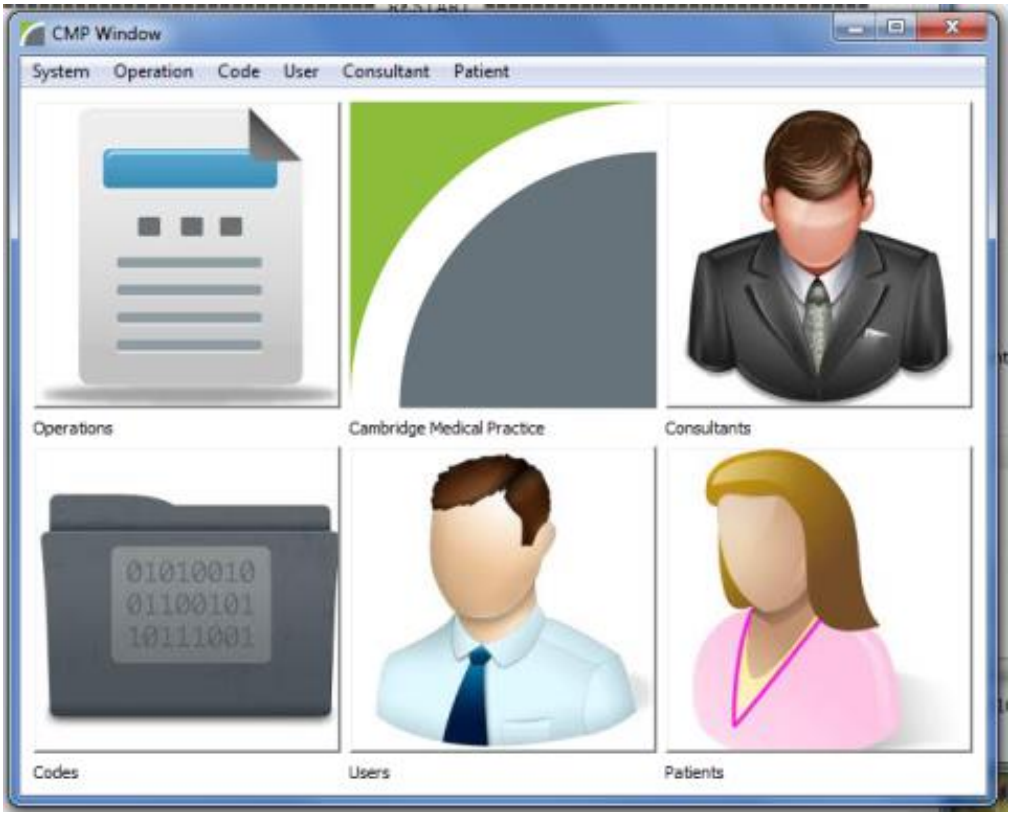
An image of the company logo

A group box containing:

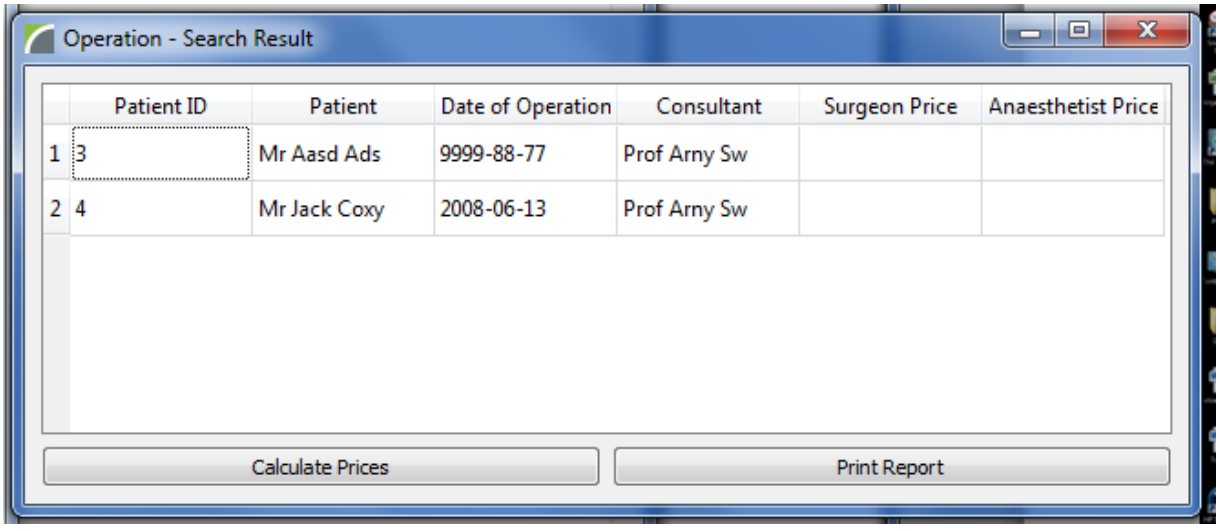
3 radio buttons

A combo box

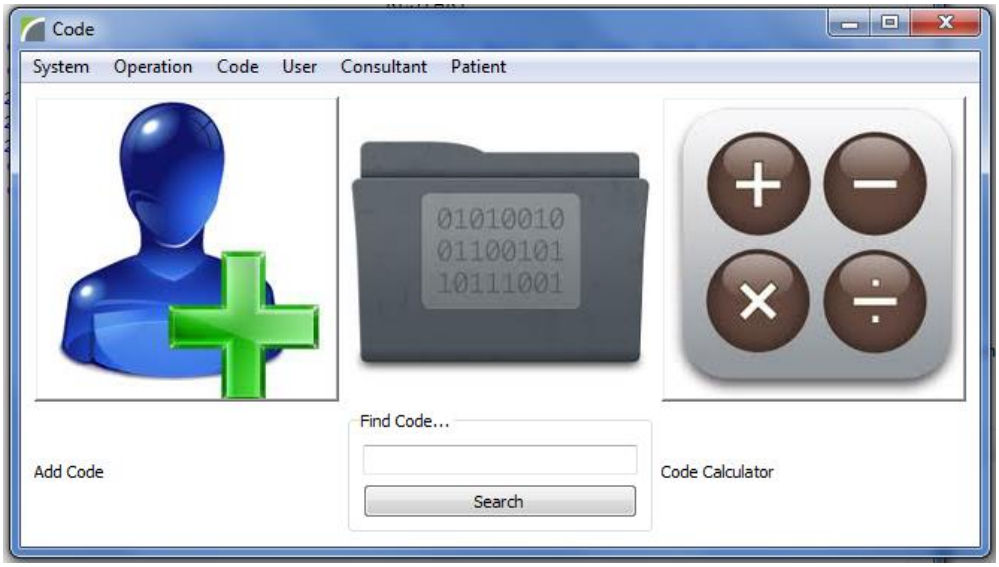
A line edit



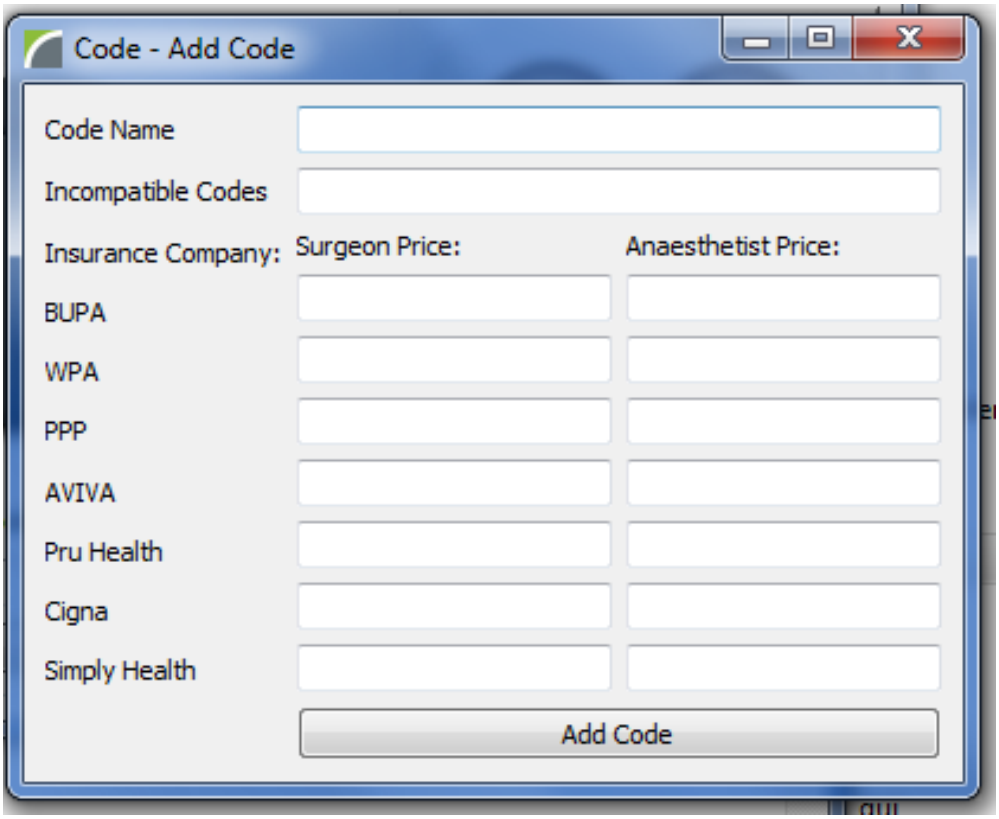
This is the CMP window, it consists of: the standard menu bar for all main system areas, an image of the company logo, 6 labels and 5 buttons with image backgrounds.



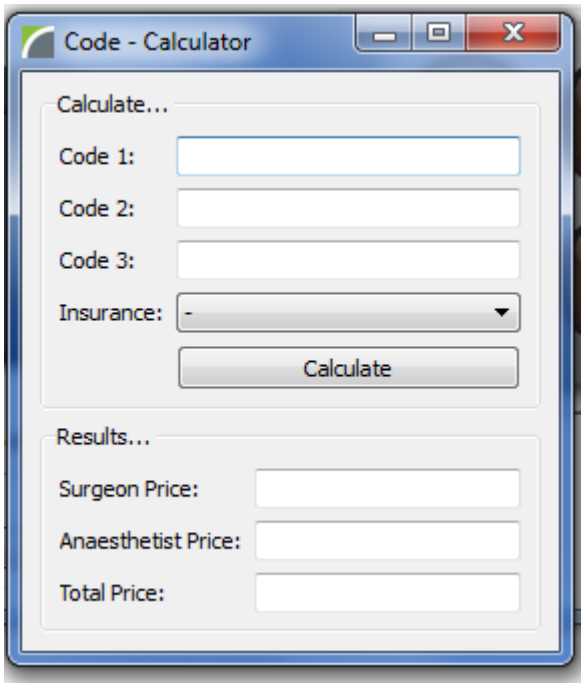
This is the Operations- Search Result window, it consists of: a table and 2 buttons



This is the Code window, it consists of: the standard menu bar for all main system areas, the Code image, 2 labels, 2 buttons with image backgrounds, a group box containing a line edit and a search button.



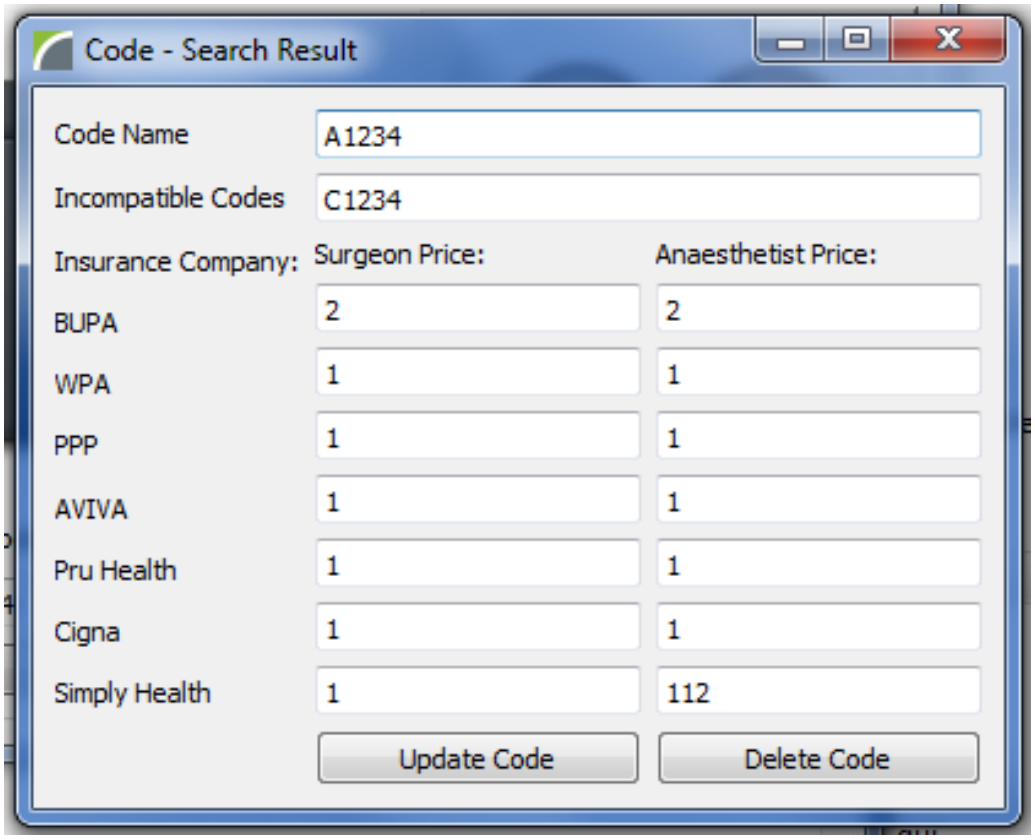
This is the Code- Add Code window, it consists of: 16 line edits, 12 line edits and an add button.



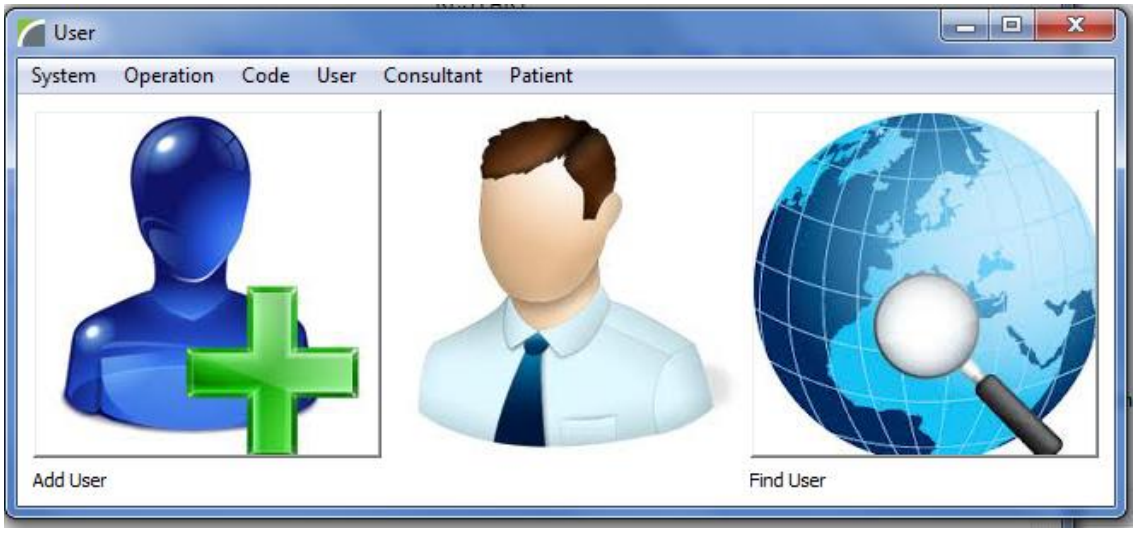
This is the log in window, it consists of:

A group box containing: 4 labels, 3 line edits, a combo box and a calculate button.

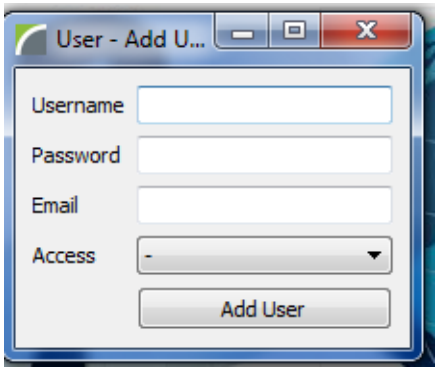
A group box containing: 3 labels and 3 line edits.



This is the Code- Search Result window, it consists of: 16 line edits, 12 line edits and 2 buttons.



This is the User window, it consists of: the standard menu bar for all main system areas, the user image, 2 labels and 2 buttons with image backgrounds.



This is the User- Add User window, it consists of:

4 Labels

3 Line Edits

A combo box

An add button

This is the User- Search window, it consists of:

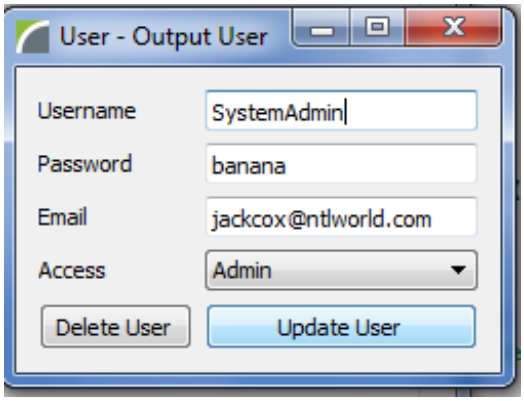
The search image

A group box containing:

2 radio buttons

2 line edits

A search button



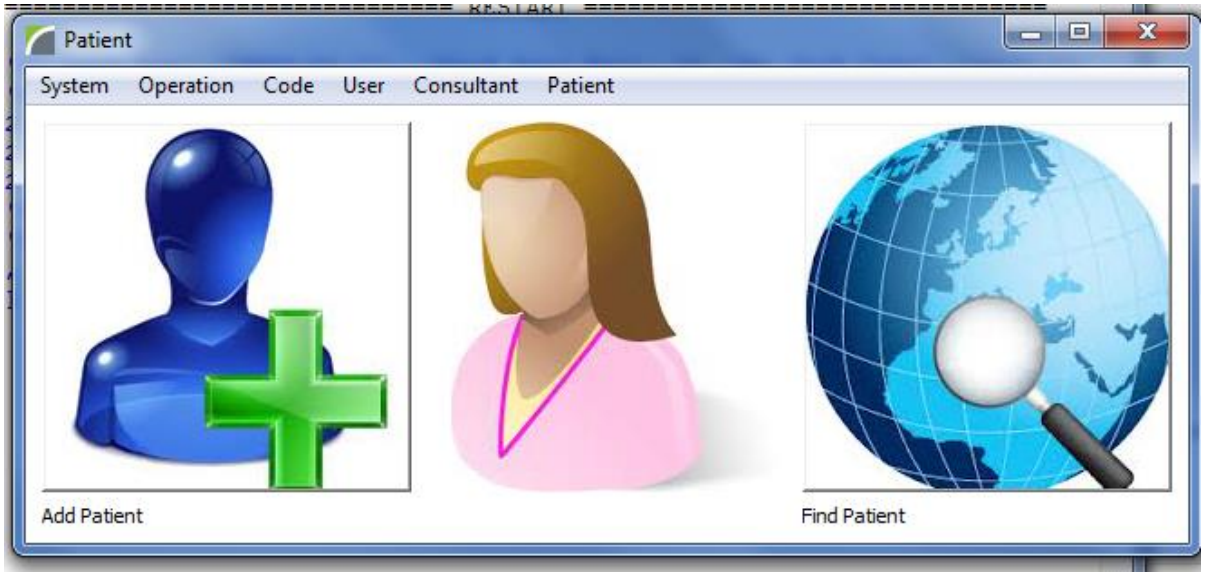
This is the User- Output User window, it consists of:

4 Labels

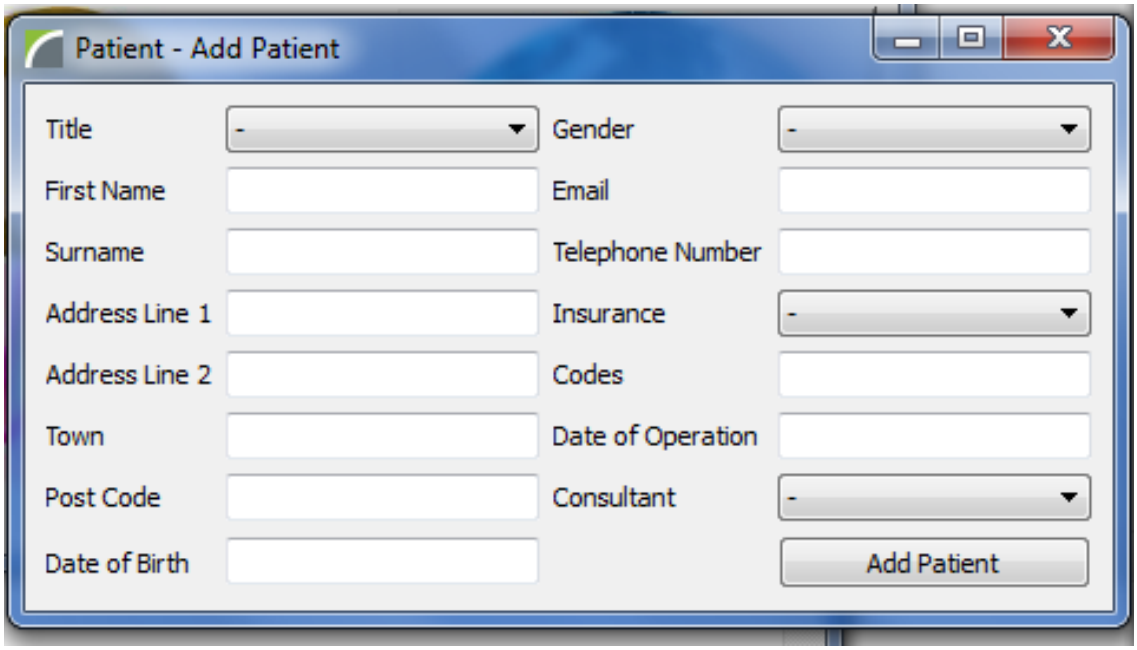
3 Line Edits

A combo box

2 buttons



This is the Patient window, it consists of: the standard menu bar for all main system areas, the patient image, 2 labels and 2 buttons with image backgrounds.



This is the Patient- Add Patient window, it consists of: 15 labels, 11 line edits, 4 combo boxes and an add button.

This is the Patient- Search window, it consists of:

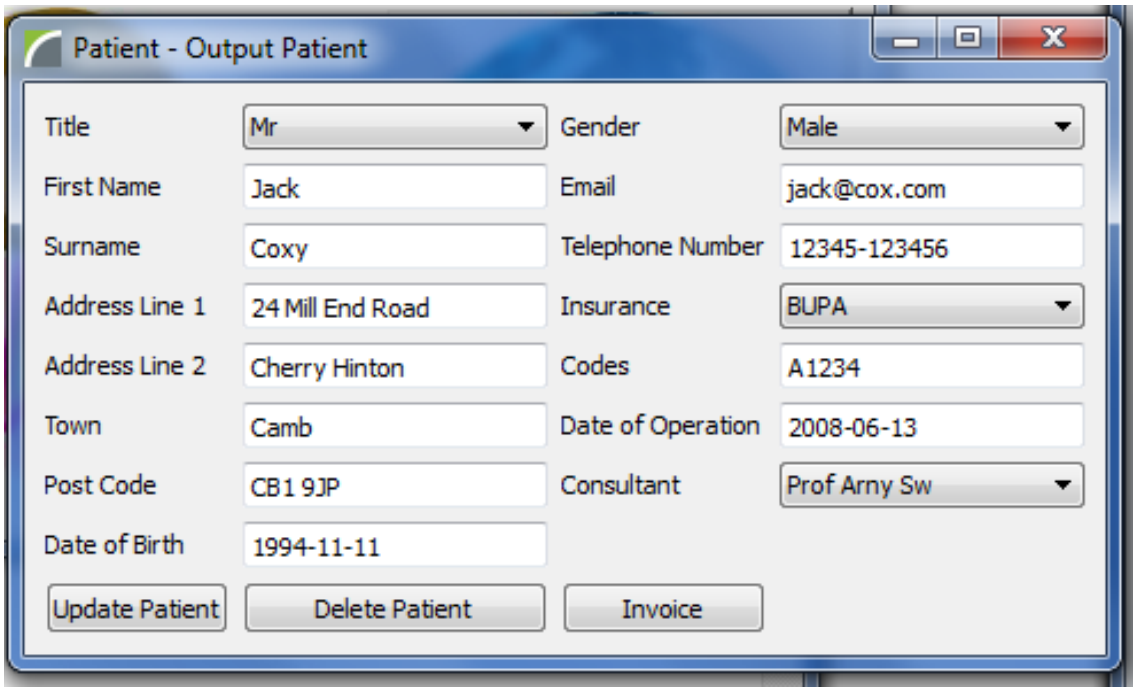
The search image

A combo box that consist of:

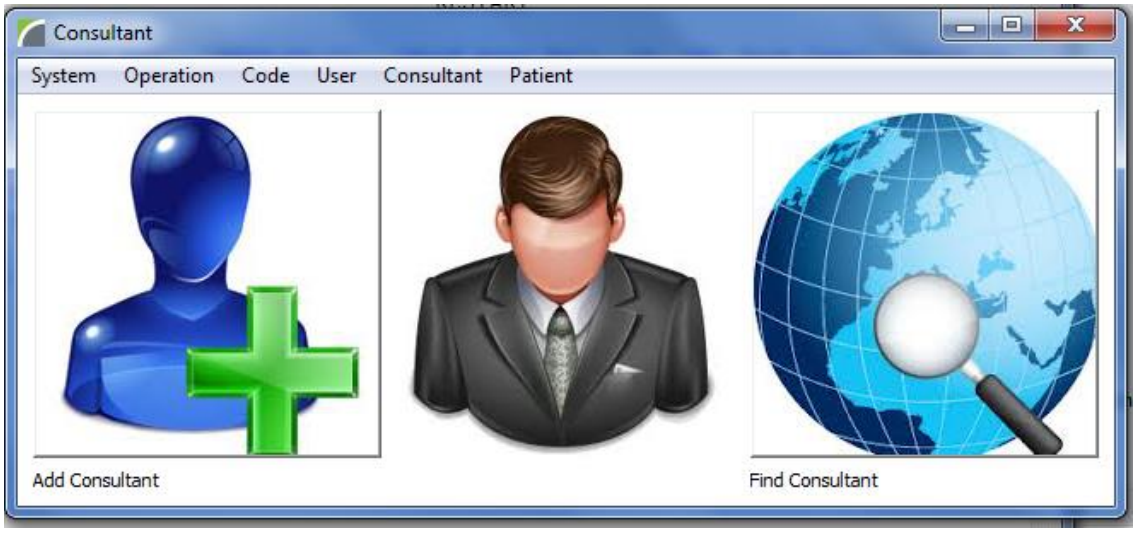
2 radio buttons

3 line edits

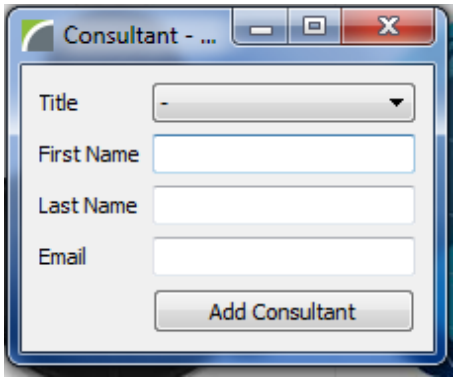
A search button



This is the Patient- Output Patient window, it consists of: 15 labels, 11 line edits, 4 combo boxes and 3 buttons.



This is the Consultant window, it consists of: the standard menu bar for all main system areas, the consultant image, 2 labels and 2 buttons with image backgrounds.



This is the Consultant- Add Consultant window, it consists of:

4 labels

A combo box

3 line edits

An add button

This is the Patient- Search window, it consists of:

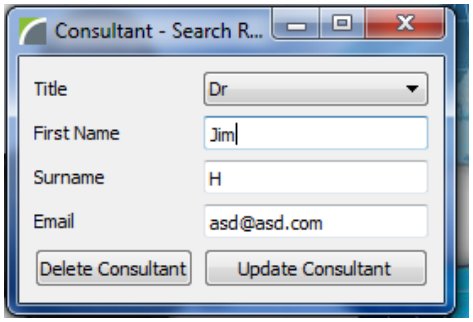
The search image

A combo box that consist of:

2 radio buttons

3 line edits

A search button



This is the Consultant- Search Result window, it consists of:

4 labels

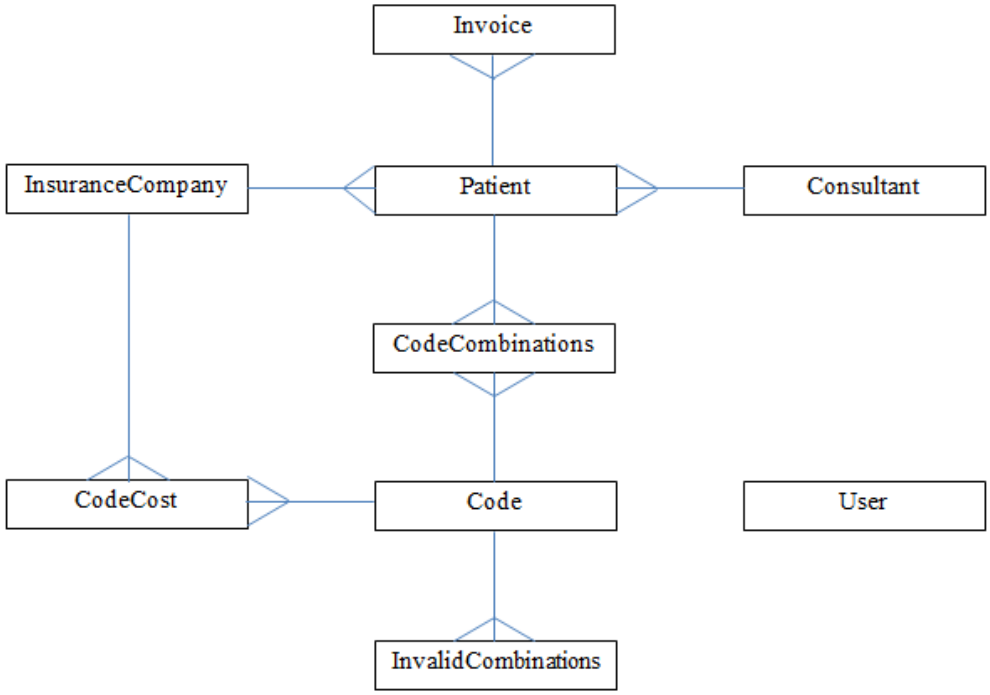
A combo box

3 line edits

2 buttons

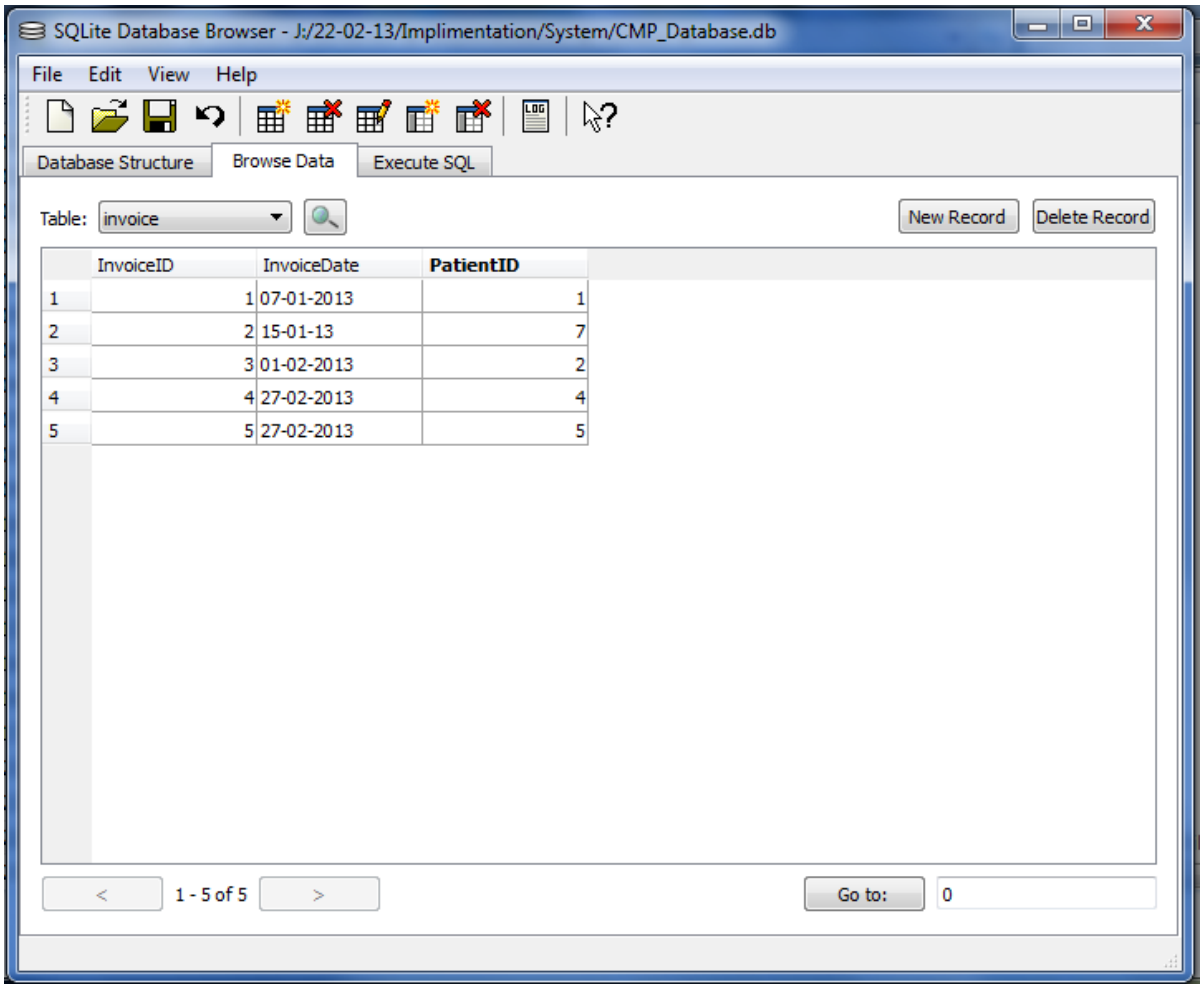
5.2. ER Diagram

The final database is illustrated below in the form of an Entity Relationship diagram.



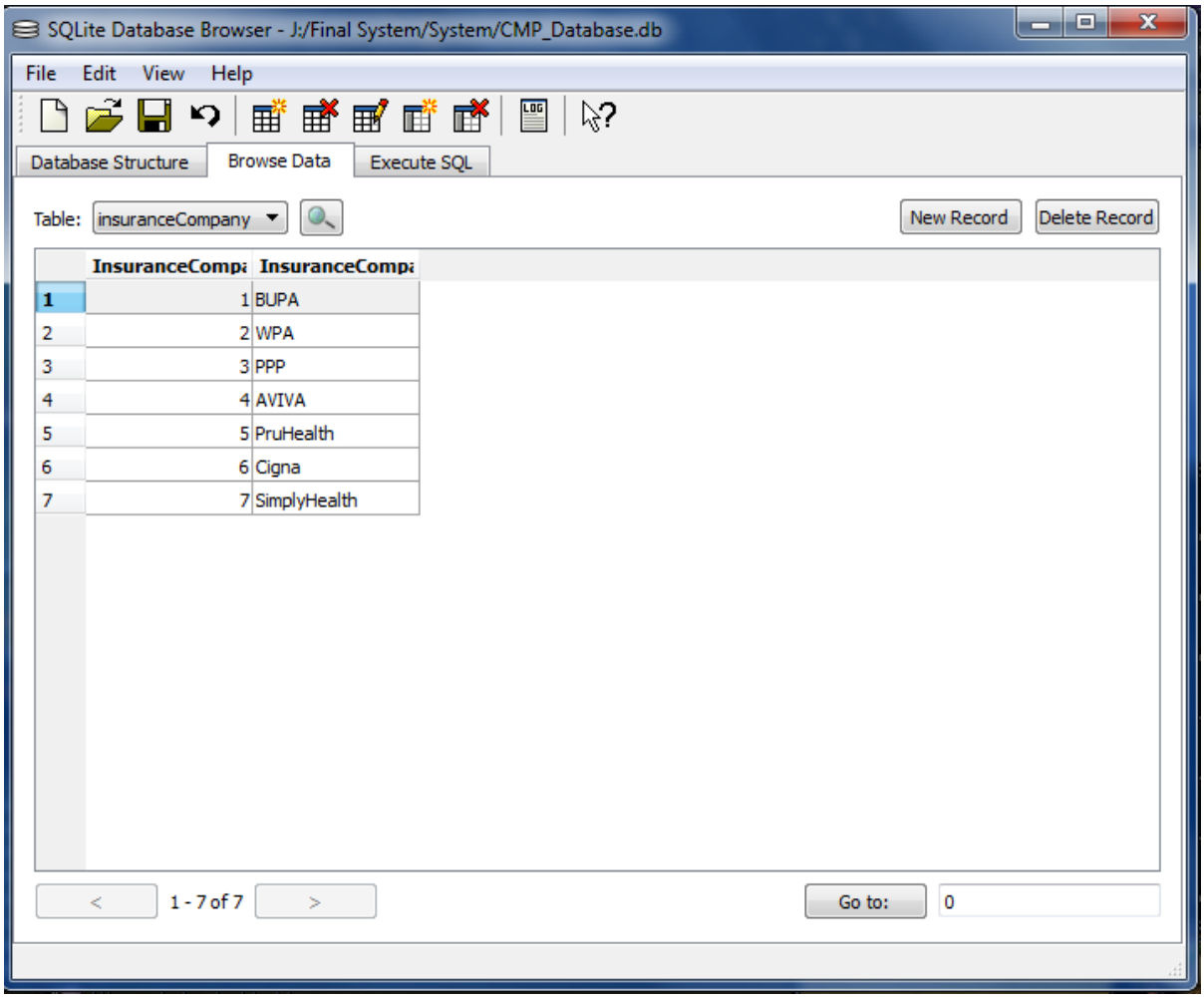
5.3. Database Table Views

5.3.1. ‘invoice’ Entity



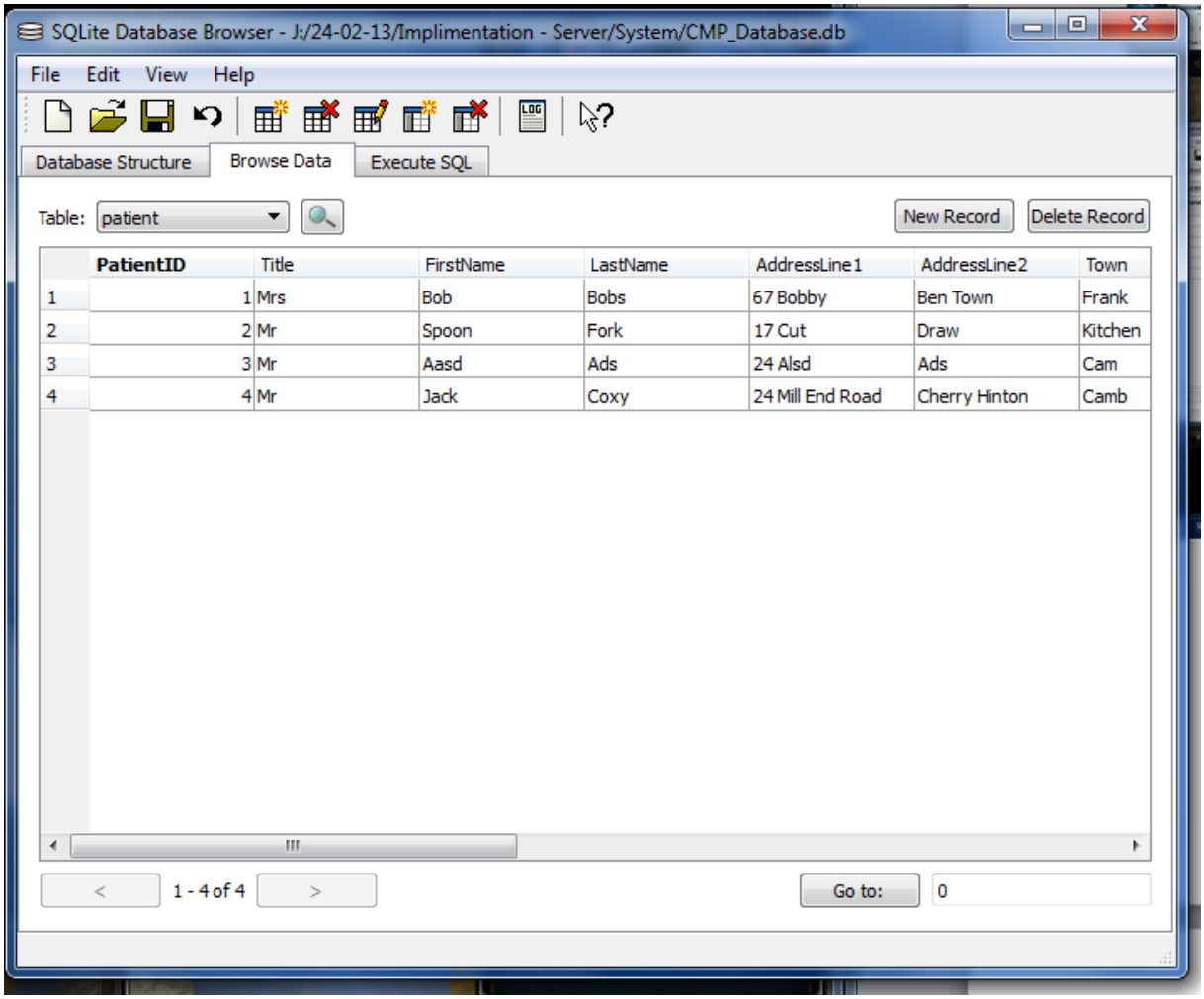
The ‘invoice’ entity has three fields. The primary key is the first field, ‘InvoiceID’. The second field contains regular data. The third, ‘PatientID’, is a foreign key which connects to the ‘patient’ entity which can be seen in section 5.3.3.

5.3.2. ‘insuranceCompany’ Entity



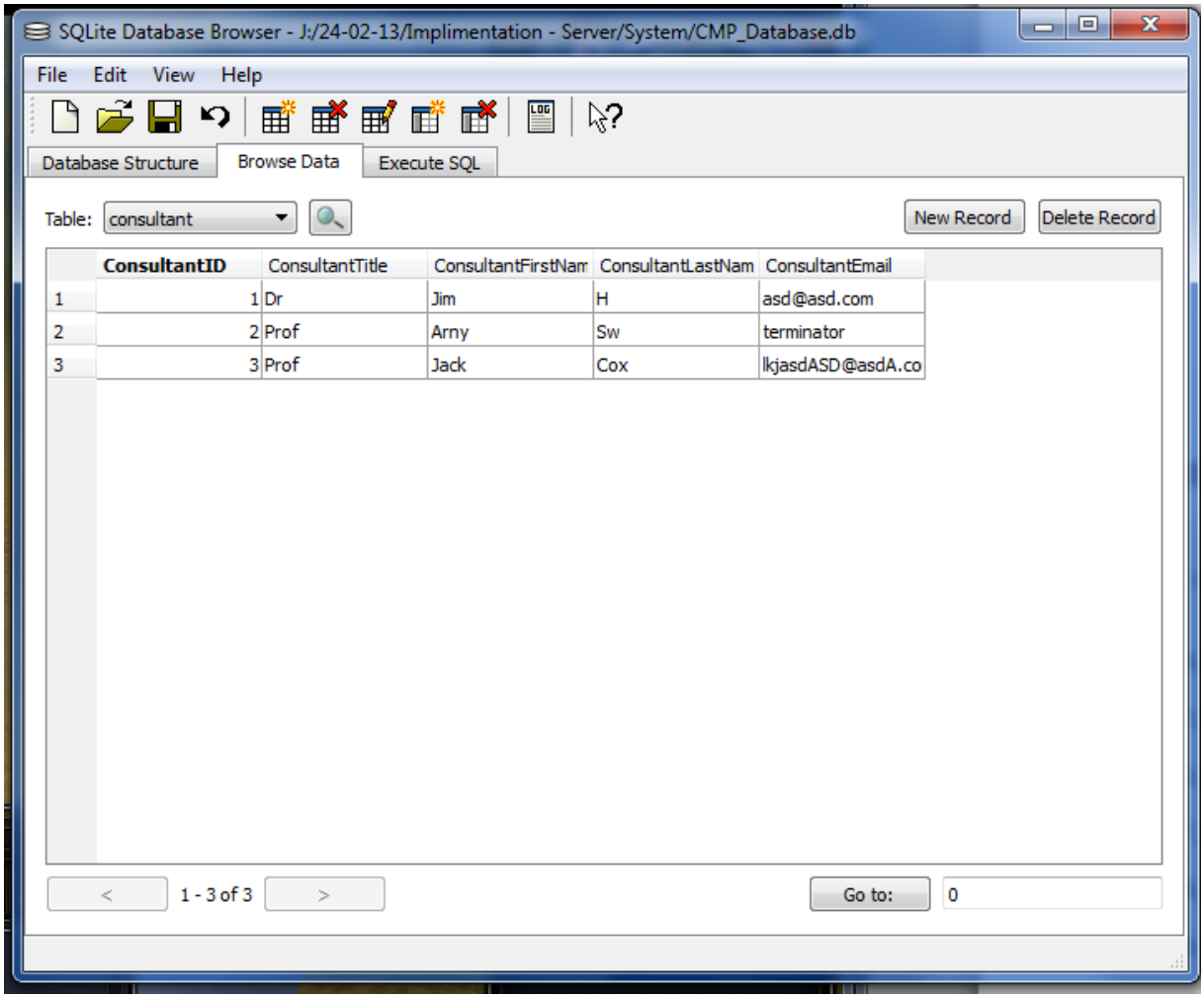
The first field, ‘InsuranceCompanyID’ is assigned a value automatically when a new record is created and is also the primary key of this entity. The ‘InsuranceCompanyName’ field stores the only data in this entity.

5.3.3. ‘patient’ Entity



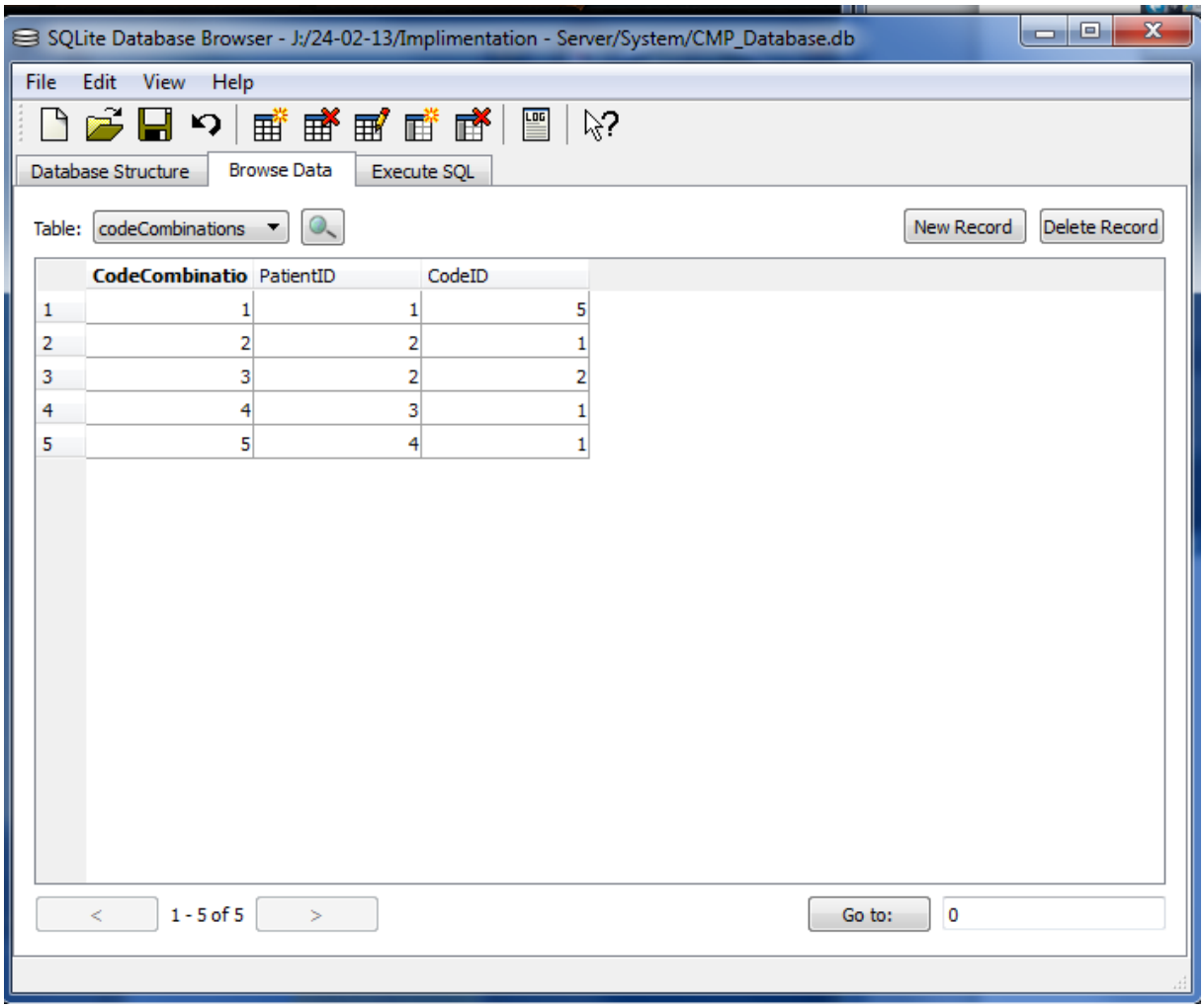
This is the ‘patient’ entity which stores all data regarding patients for the system. ‘PatientID ’ is the primary key and all the other fields are filled from the user via the system.

5.3.4. ‘consultant’ Entity



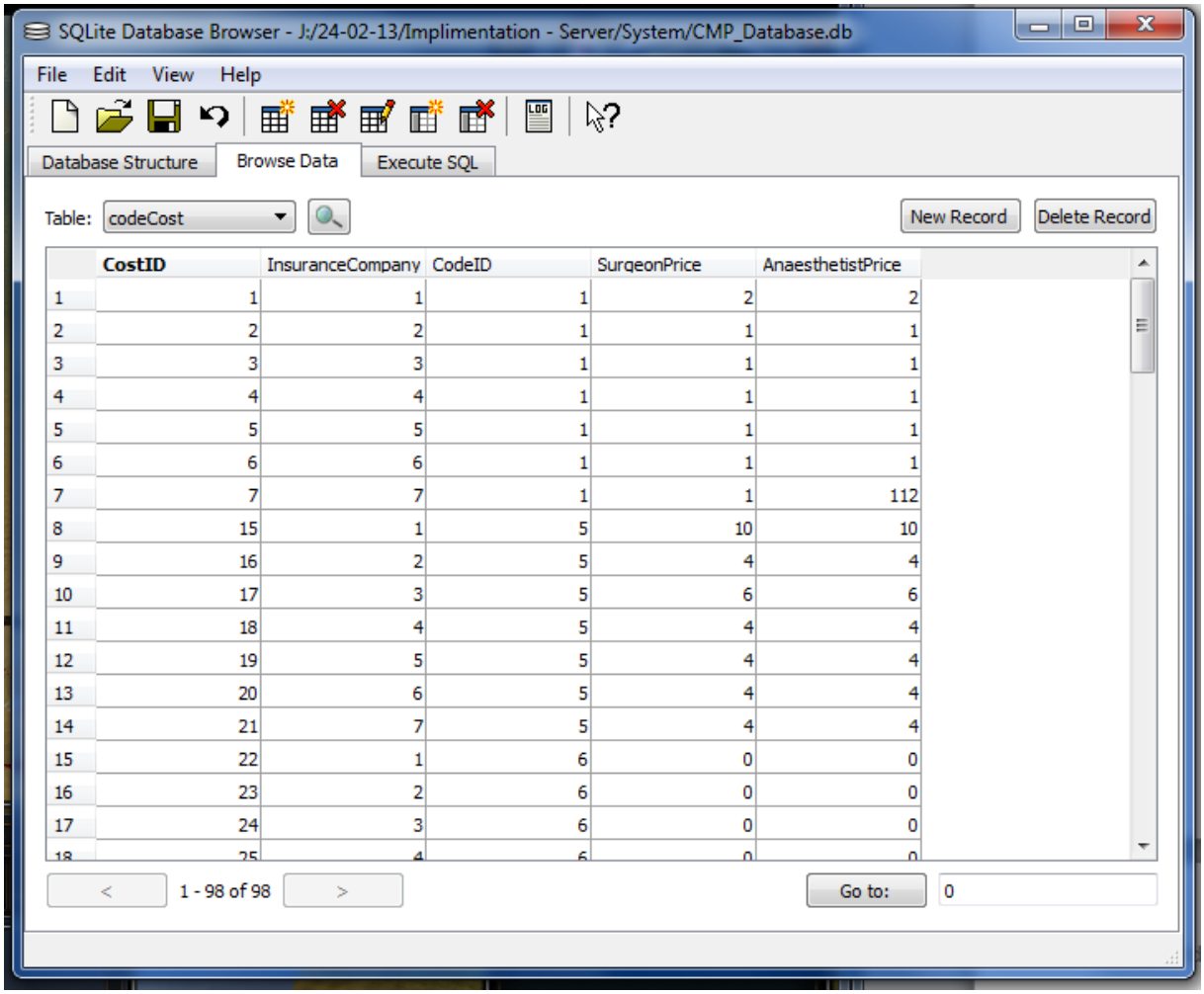
Shown above is the ‘consultant’ entity where ‘ConsultantID’ is the primary key and all of the other fields are regular data which gets past between the database and the system.

5.3.5. ‘codeCombinations’ Entity



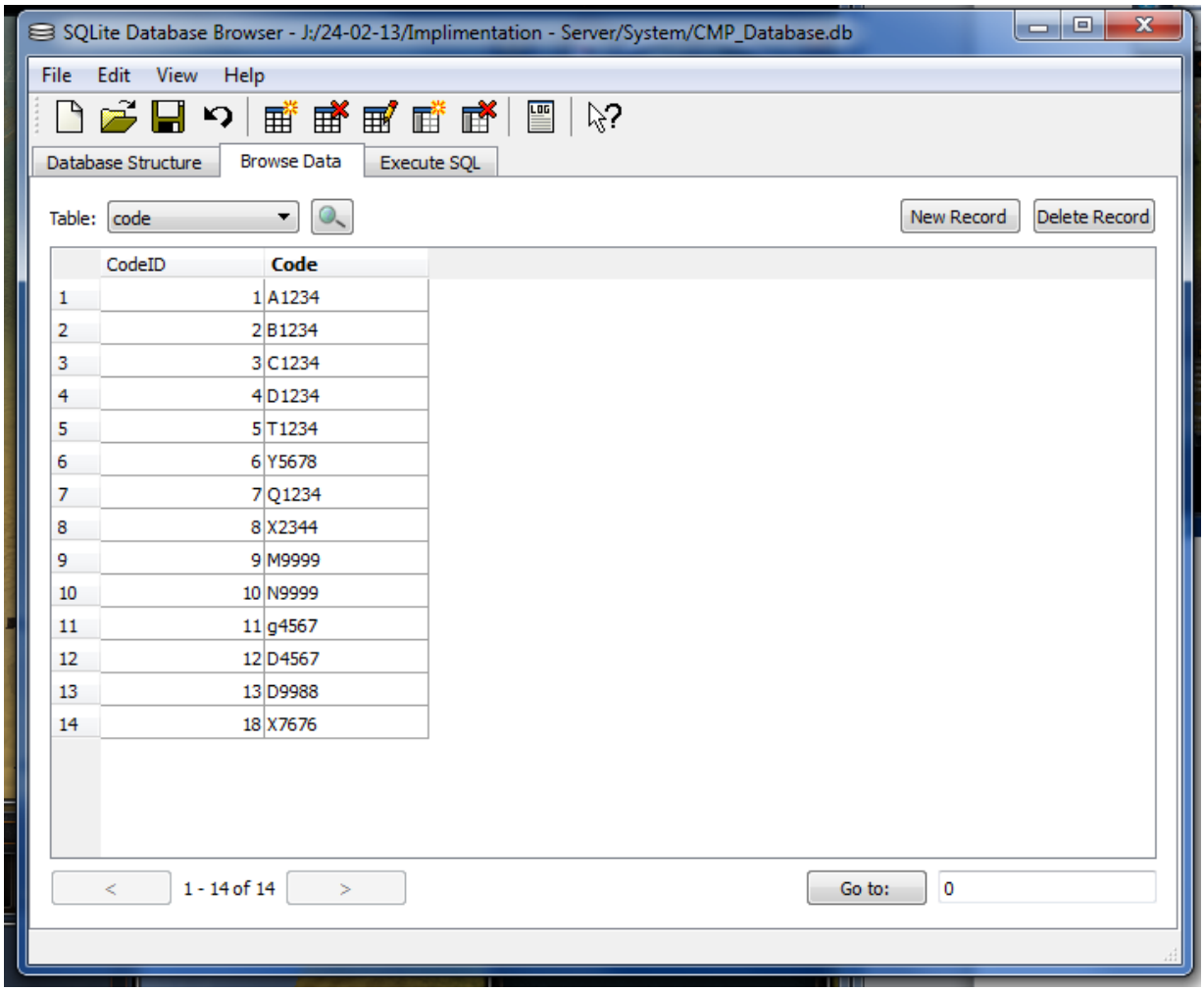
This entity consists of a primary key, ‘CodeCombinationID’, and two foreign keys, ‘PatientID’ and ‘CodeID’. The foreign keys reference the primary keys from the ‘patient’ and ‘code’ entities which can be found in sections 2.3.3. and 2.3.7. of this document.

5.3.6. ‘codeCost’ Entity



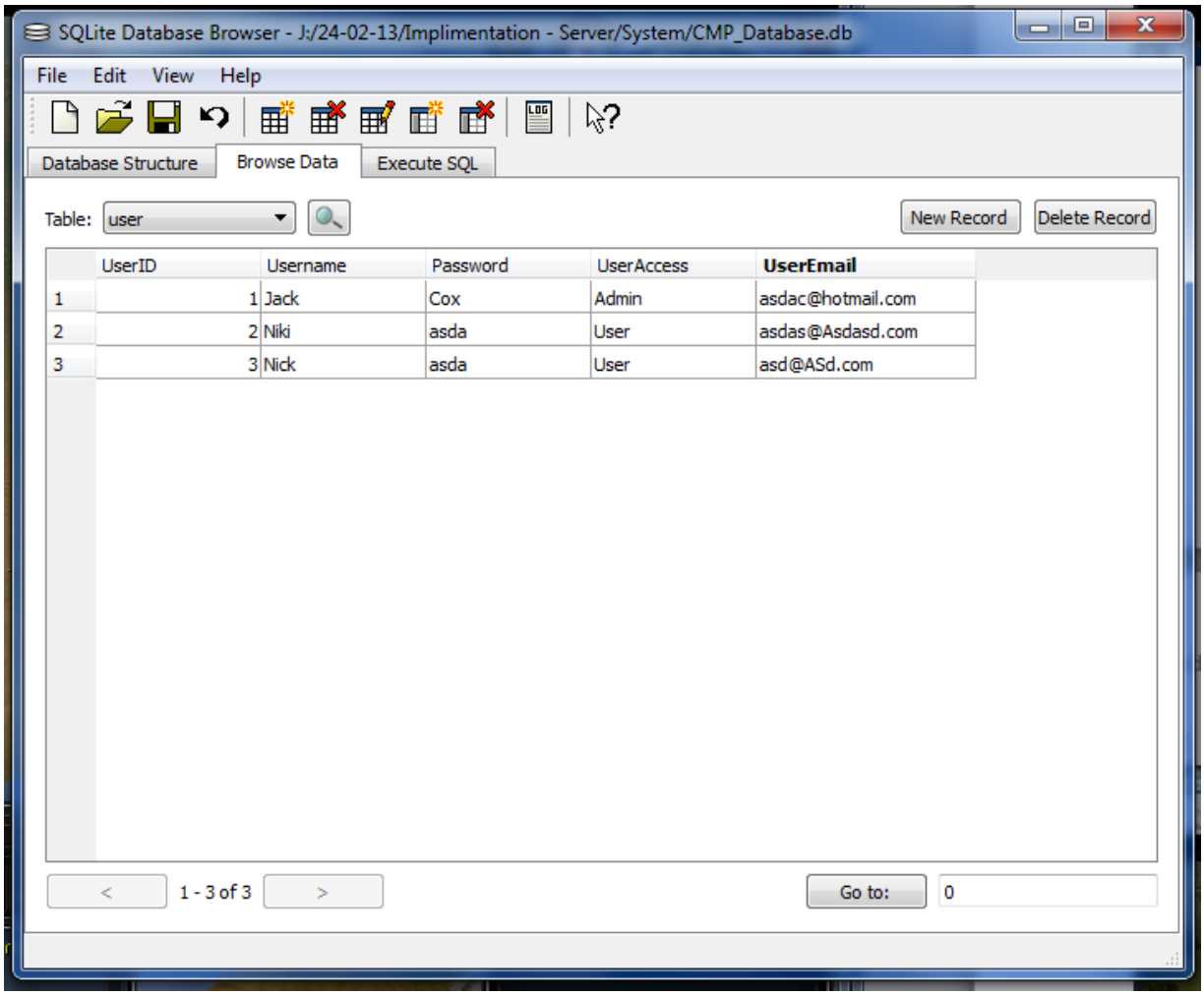
The ‘codeCost’ entity stores the two data fields which contain the costs for a particular code and insurance company. The primary key is the ‘CostID’ and the two foreign keys are ‘InsuranceCompanyID’ and ‘CodeID’. These reference respectively to the ‘insuranceCompany’ and ‘code’ entities which can be found in sections 5.3.2. and 5.3.7.

5.3.7. ‘code’ Entity



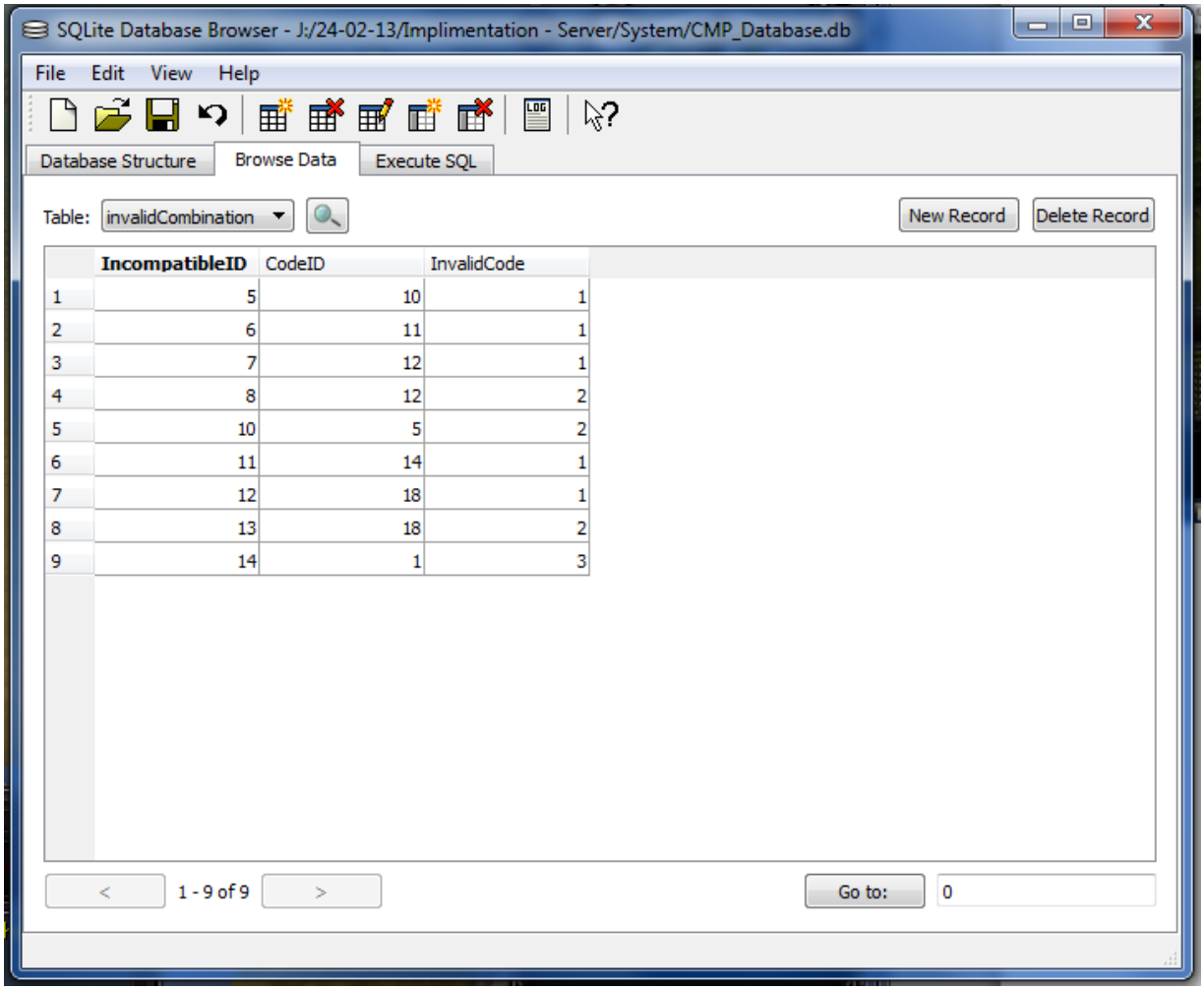
The ‘code’ entity only has 2 fields. The primary key field is ‘CodeID’ and the remaining data field is ‘Code’.

5.3.8. ‘user’ Entity



The ‘user’ entity consists of five fields, where the primary key is the ‘UserID’ field. The remaining fields all contain ordinary data

5.3.9. 'invalidCombination' Entity



The three fields in the 'invalidCombination' entity are shown above. The primary key is 'IncompatibleID' and the other fields, 'CodeID' and 'InvalidCode', are both foreign keys which reference 'CodeID' from the 'code entity' (section 5.3.7.).

5.4. Database SQL

This section of the document contains the SQL which generates the database. This is useful for testing and implementation as on some occasions the database can become locked which requires it to be recreated.

5.4.1. 'invoice' SQL

```
def create_invoice_entity(db,cursor):
    sql = """create table invoice (
             InvoiceID integer,
             InvoiceDate date,
             PatientID integer,
             primary key (InvoiceID),
             foreign key (PatientID)
             references patient(PatientID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.2. 'insuranceCompany' SQL

```
def create_insuranceCompany_entity(db,cursor):
    sql = """create table insuranceCompany (
             InsuranceCompanyID integer,
             InsuranceCompanyName integer,
             primary key (InsuranceCompanyID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.3. 'patient' SQL

```
def create_patient_entity(db,cursor):
    sql = """create table patient (
             PatientID integer,
             Title text,
             FirstName text,
             LastName text,
             AddressLine1 text,
             AddressLine2 text,
             Town text,
             PostCode text,
             DateOfBirth date,
             Gender text,
             TelephoneNumber text,
             Email text,
             DateOfOperation date,
             ConsultantID integer,
             InsuranceCompanyID integer,
             primary key (PatientID),
             foreign key (ConsultantID)
             references consultant(ConsultantID)
             foreign key (InsuranceCompanyID)
             references insuranceCompany(InsuranceCompanyID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.4. 'consultant' SQL

```
def create_consultant_entity(db,cursor):
    sql = """create table consultant (
             ConsultantID integer,
             ConsultantTitle text,
             ConsultantFirstName text,
             ConsultantLastName text,
             ConsultantEmail text,
             primary key (ConsultantID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.5. 'codeCombinations' SQL

```
def create_codeCombinations_entity(db,cursor):
    sql = """create table codeCombinations (
             CodeCombinationID integer,
             PatientID integer,
             CodeID integer,
             primary key (CodeCombinationID)
             foreign key (PatientID)
             references patient(PatientID)
             foreign key (CodeID)
```

```
            references code(CodeID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.6. 'codeCost' SQL

```
def create_codeCost_entity(db,cursor):
    sql = """create table codeCost (
            CostID integer,
            InsuranceCompanyID integer,
            CodeID integer,
            SurgeonPrice integer,
            AnaesthetistPrice integer,
            primary key (CostID)
            foreign key (InsuranceCompanyID)
            references insuranceCompany(InsuranceCompanyID)
            foreign key (CodeID)
            references code(CodeID)
            on delete cascade)"""
    cursor.execute(sql)
    db.commit()
```

5.4.7. 'code' SQL

```
def create_code_entity(db,cursor):
    sql = """create table code (
            CodeID integer,
            Code text,
            primary key (CodeID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.8. 'user' SQL

```
def create_user_entity(db,cursor):
    sql = """create table user (
            UserID integer,
            Username text,
            Password text,
            UserAccess text,
            UserEmail text,
            primary key (UserID))"""
    cursor.execute(sql)
    db.commit()
```

5.4.9. 'invalidCombination' SQL

```
def create_invalidCombination_entity(db,cursor):
    sql = """create table invalidCombination (
            IncompatibleID integer,
            CodeID integer,
            InvalidCode integer,
            primary key (IncompatibleID)
            foreign key (CodeID)
            references code(CodeID)
            on delete cascade
            foreign key (InvalidCode)
            references code(CodeID)
            on delete cascade)"""
    cursor.execute(sql)
```

```
db.commit()
```

5.5. SQL Queries

The SQL from the system has been broken down into the six controllers which each have a section below. The SQL statements can be seen between the quotation marks, the .format is the python feature which allows the transition on data into a string.

5.5.1. 'code_controller.py' SQL

Functionality	SQL Statement
Inserts a new code name into the database.	"""insert into code (Code) values ('{0}')""".format(code)
Retrieves a code ID from the database.	"""select CodeID from code where Code = '{0}'""".format(code)
Inserts a new invalid combination into the database.	"""insert into invalidCombination (CodeID,InvalidCode) values ('{0}','{1}')""".format(ID,invalidID)
Inserts a new code cost into the database.	"""insert into codeCost (InsuranceCompanyID,CodeID,SurgeonPrice,AnaesthetistPrice) values ('{0}','{1}','{2}','{3}')""".format(IC,ID,costs[count],costs[count+1])
Removes a code from the database.	"""delete from code where CodeID = '{0}'""".format(ID)
Retrieves an invalid code from the database.	"""select InvalidCode from invalidCombination where CodeID = '{0}'""".format(ID)
Retrieves a code name from the database.	"""select Code from code where CodeID = '{0}'""".format(item)
Retrieves the costs of a code from the database.	"""select SurgeonPrice, AnaesthetistPrice from codeCost where CodeID = '{0}' and InsuranceCompanyID = '{1}'""".format(ID,IC[count])
Amends the code name in the database.	"update code set Code = '{0}' where CodeID = '{1}'".format(code,ID)
Amends the one of	"update codeCost set SurgeonPrice = '{0}' where CodeID = '{1}' and

the code costs in the database.	InsuranceCompanyID = '{2}'".format(icList[count],ID,IC)
Amends the one of the code costs in the database.	"update codeCost set AnaesthetistPrice = '{0}' where CodeID = '{1}' and InsuranceCompanyID = '{2}'".format(icList[count+1],ID,IC)
Removes an invalid combination from the database.	"delete from invalidCombination where CodeID = '{0}'".format(ID)
Retrieves an insurance company ID from the database.	"""select InsuranceCompanyID from insuranceCompany where InsuranceCompanyName = '{0}'""".format(IC)
Retrieves an insurance company ID from the database.	"""select SurgeonPrice, AnaesthetistPrice from codeCost where CodeID = '{0}' and InsuranceCompanyID = '{1}'""".format(ID,icID)

5.5.2. 'consultant_controller.py' SQL

Functionality	**SQL Statement**
Inserts a new consultant into the database.	"""insert into consultant (ConsultantTitle,ConsultantFirstName,ConsultantLastName, ConsultantEmail) values ('{0}','{1}','{2}','{3}')""".format(title,fname,lname,email)
Removes a consultant from the database.	"""delete from consultant where ConsultantID = '{0}'"""
Retrieves a consultant from the database.	"""select * from consultant where ConsultantID = '{0}'""".format(ID)
Retrieves a consultant from the database.	"""select * from consultant where ConsultantFirstName = '{0}'""".format(fname)
Retrieves a consultant from the database.	"""select * from consultant where ConsultantLastName = '{0}'""".format(lname)

Retrieves a consultant from the database.	"""select * from consultant where ConsultantFirstName = '{0}' and ConsultantLastName = '{1}'""".format(fname,lname)
Retrieves a consultant from the database.	"select * from consultant"

5.5.3. 'insurance_company_controller.py' SQL

Functionality	**SQL Statement**
Inserts an insurance company into the database.	"""insert into insuranceCompany (InsuranceCompanyName) values ('{0}')""".format(name)
Removes an insurance company from the database.	"""delete from insuranceCompany where InsuranceCompanyID = '{0}'""".format(ID)
Retrieves an insurance company name from the database.	"""select InsuranceCompanyName from insuranceCompany where InsuranceCompanyID = '{0}'""".format(ID)

5.5.4. 'invoice_controller.py' SQL

Functionality	**SQL Statement**
Inserts a new invoice into the database.	"""insert into invoice (InvoiceDate,PatientID) values ('{0}','{1}')""".format(date,pID)

5.5.5. 'patient_controller.py' SQL

Functionality	**SQL Statement**
Retrieves a consultant ID from the database.	"""select ConsultantID from consultant where ConsultantLastName = '{0}'""".format(conLname)
Retrieves an insurance ID from the	"""select InsuranceCompanyID from insuranceCompany where InsuranceCompanyName = '{0}'""".format(icName)

database.	
Inserts a new patient into the database.	"""insert into patient (Title,FirstName,LastName,AddressLine1,AddressLine2,Town,PostCode,DateOfBirth,Gender,TelephoneNumber,Email,DateOfOperation,ConsultantID,InsuranceCompanyID) values ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}','{12}','{13}')""".format(title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,email,DoO,conID,icID)
Retrieves a patient ID from the database.	"""select PatientID from patient where LastName = '{0}' and TelephoneNumber = '{1}'""".format(lname,telNo)
Retrieves a code ID from the database.	"""select CodeID from code where Code = '{0}'""".format(item)
Inserts a new combination of codes into the database.	"""insert into codeCombinations (PatientID,CodeID) values ('{0}','{1}')""".format(patientID,codeID)
Removes a patient from the database.	"""delete from patient where PatientID = '{0}'""".format(ID)
Removes a combination of codes from the database.	"""delete from codeCombinations where PatientID = '{0}'""".format(ID)
Retrieves a patient from the database.	"""select * from patient where PatientID = '{0}'""".format(ID)
Retrieves a patient from the database.	"""select * from patient where FirstName = '{0}' and LastName = '{1}'""".format(fname,lname)
Retrieves a patient from the database.	"""select * from patient where FirstName = '{0}'""".format(fname)
Retrieves a patient from	"""select * from patient where LastName = '{0}'""".format(lname)

the database.	
Retrieves a patient from the database.	"""select * from patient where Consultant = '{0}'""".format(conid)
Retrieves a patient from the database.	"""select * from patient where DateOfOperation = '{0}'""".format(date)
Retrieves a patient ID from the database.	"""select PatientID from patient where FirstName = '{0}' and LastName = '{1}'""".format(fname,lname)
Retrieves a patient ID from the database.	"""select PatientID from patient where FirstName = '{0}'""".format(fname)
Retrieves a patient ID from the database.	"""select PatientID from patient where LastName = '{0}'""".format(lname)
Retrieves a Code ID from the database.	"""select CodeID from codeCombinations where PatientID = '{0}'""".format(ID)
Retrieves a code name from the database.	"""select Code from code where CodeID = {0}""".format(ID)
Retrieves a patient from the database.	"select * from patient where ConsultantID = '{0}'".format(conID)
Retrieves a patient from the database.	"select * from patient where DateOfOperation = '{0}' and ConsultantID = '{1}'".format(data[0],conID)
Retrieves a consultant ID from the database.	"select ConsultantID from consultant where ConsultantLastName = '{0}'".format(conLname)
Amends the consultant ID	"update patient set ConsultantID='{0}' where PatientID='{1}'".format(conID,ID)

in the database.	
Retrieves an insurance ID from the database.	"select InsuranceCompanyID from insuranceCompany where InsuranceCompanyName = '{0}'".format(icName)
Amends the insurance ID name in the database.	"update patient set InsuranceCompanyID='{0}' where PatientID='{1}'".format(icID,ID)

5.5.6. 'user_controller.py' SQL

Functionality	**SQL Statement**
Inserts a new user into the database.	"""insert into user (Username,Password,UserAccess,UserEmail) values ('{0}','{1}','{2}','{3}')""".format(UN,PW,AC,email)
Removes a user from the database.	"""delete from user where UserID = '{0}'""".format(ID)
Retrieves a user from the database.	"""select * from user where UserID = '{0}'""".format(ID)
Retrieves a user from the database.	"""select * from user where Username = '{0}'""".format(UN)

6. Testing

6.1. Summary of Results

Testing proved my system to be relatively strong and reliable as only two problems occurred during testing. These can be seen in section 4.2. of the testing section of this document. Now that these have been fixed the system is robust as none of the tests caused the system to crash or fail to complete a task appropriately and accurately.

The system meets the majority of the key objectives which the client desires make the system reliable. However as the encryption module was not completed, due to lack of time and knowledge, the data in the system will not be protected which was a primary objective stated by the client.

Below is my entire results table from the testing section.

Test	Test Data Type	Expected Result	Test Data	Actual Result	Annotation Reference
1.2.	Normal	"Operations" Window displayed.	"Operations" Button Clicked.	Expected Result	Figure 1 – Test 1.2. page 119
1.3.	Normal	"Codes" Window displayed.	"Codes" Button Clicked.	Expected Result	
1.4.	Normal	"Users" Window displayed.	"Users" Button Clicked.	Expected Result	
1.5.	Normal	"Consultants" Window displayed.	"Consultants" Button Clicked.	Expected Result	
1.6.	Normal	"Patients" Window displayed.	"Patients" Button Clicked.	Expected Result	
1.12.	Normal	"Operations – Search Results" Window displayed.	"Search" button Clicked.	Expected Result	
1.13.	Normal	"Codes – New Code" Window displayed.	"Add" button Clicked.	Expected Result	Figure 2 - Test 1.13. page 119
1.14.	Normal	"Codes – Search Results" Window displayed.	"Search" button Clicked.	Expected Result	
1.15.	Normal	"Codes – Calculator" Window displayed.	"Calculator" button Clicked.	Expected Result	
1.16.	Normal	"Patients – New Patient" Window displayed.	"Add Patient" button Clicked.	Expected Result	
1.17.	Normal	"Patients - Search" Window displayed.	"Find Patient" button Clicked.	Expected Result	Figure 3 - Test 1.17. page 120
1.18.	Normal	"Patients – Search Results" Window displayed.	"Search" button Clicked.	Expected Result	
1.19.	Normal	"Consultants – New Consultant" Window displayed.	"Add Consultant" button Clicked	Expected Result	
1.20.	Normal	"Consultants – Search" Window displayed.	"Find Consultant" button Clicked	Expected Result	
1.21.	Normal	"Consultants – Search Results" Window displayed.	"Search" button Clicked	Expected Result	
1.22.	Normal	"User – New User" Window displayed.	"Add User" button Clicked	Expected Result	
1.23.	Normal	"User – Search" Window displayed.	"Find User" button Clicked	Expected Result	
1.24.	Normal	"User – Search Results" Window	"Search" button Clicked	Expected Result	

		displayed.			
1.25.	Normal	"Operations" Window displayed.	Click on the "Find Operations" tab in the menu bar.	Expected Result	
1.26.	Normal	"Code" Window displayed.	Click on the "Code Main" tab in the menu bar.	Expected Result	
1.27.	Normal	"Code- Add Code" Window displayed.	Click on the "Add Code" tab in the menu bar.	Expected Result	
1.28.	Normal	"Code- Calculator" Window displayed.	Click on the "Code Calculator" tab in the menu bar.	Expected Result	Figure 4 - Test 1.28. page 120
1.29.	Normal	"User" Window displayed.	Click on the "User Main" tab in the menu bar.	Expected Result	
1.30.	Normal	"User- Add User" Window displayed.	Click on the "Add User" tab in the menu bar.	Expected Result	
1.31.	Normal	"User - Search" Window displayed.	Click on the "Find User" tab in the menu bar.	Expected Result	
1.32.	Normal	"Consultant" Window displayed.	Click on the "Consultant Main" tab in the menu bar.	Expected Result	
1.33.	Normal	"Consultants- Add Consultant" Window displayed.	Click on the "Add Consultant" tab in the menu bar.	Expected Result	
1.34.	Normal	"Consultants- Search" Window displayed.	Click on the "Find Consultant" tab in the menu bar.	Expected Result	
1.35.	Normal	"Patient" Window displayed.	Click on the "Patient Main" tab in the menu bar.	Expected Result	
1.36.	Normal	"Patient- Add Patient" Window displayed.	Click on the "Add Patient" tab in the menu bar.	Expected Result	
1.37.	Normal	"Patient- Find Patient" Window displayed.	Click on the "Find Patient" tab in the menu bar.	Expected Result	
1.38.	Normal	"CMP Window" Window displayed.	Click on the "Main Menu" tab in the menu bar.	Expected Result	
1.39.	Normal	"Log In" Window displayed.	Click on the "Log In" tab in the menu bar.	Expected Result	
1.40.	Normal	Confirmation message box displayed.	Click on the "Log Out" tab in the menu bar.	Expected Result	Figure 5 - Test 1.40. page 121
1.41.	Normal	System closure.	Click on the "Exit" tab	Expected	

			in the menu bar.	Result	
2.2.	Erroneous	Error message displayed	'-1'/ 'a'/ '199-1119'	Expected Result	Figure 6 - Test 2.2. page 121
	Normal	Accepted	'1994-11-19'	Expected Result	
2.3.	Normal	Accepted	'A1234'	Expected Result	
	Erroneous	Error message displayed	'a1234'/ '-1'	Expected Result	
2.4.	Normal	Accepted	'A1234'	Expected Result	
	Erroneous	Error message displayed	'a1234'/ '-1'	Expected Result	
2.5.	Normal	Accepted	'A1234,B1234'/ 'A1234'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'a'	Expected Result	
2.6.	Erroneous	Error message displayed	'-1'/ 'a'	Expected Result	
	Extreme/ Boundary	Error message displayed	'0'	Expected Result	
	Normal	Accepted	'1'	Expected Result	
2.7.	Normal	Accepted	'A1234'	Expected Result	
	Erroneous	Error message displayed	'a1234'/ '-1'	Expected Result	
2.9.	Normal	Accepted	'Jack'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'jack'	Expected Result	
2.10.	Normal	Accepted	'Cox'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'cox'	Expected Result	
2.11.	Normal	Accepted	'45 Hello Road'	Expected Result	
	Erroneous	Error message displayed	'-1'/ 'a'	Failed Result. System accepted invalid data.	
2.12.	Normal	Accepted	'Arbury'	Expected Result	Figure 7 – Test 2.12. page 122
	Erroneous	Accepted	'-1'/ 'a'	Failed Result. System accepted	

				invalid data.	
2.13.	Normal	Accepted	‘Cambs’	Expected Result	
	Erroneous	Error message displayed	‘cambs’/ ‘-1’	Expected Result	
2.14.	Normal	Accepted	‘CB8 8HN’/ ‘CB22 8HN’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
2.15.	Erroneous	Error message displayed	‘-1’/ ‘a’/ ‘199-1119’	Expected Result	
	Normal	Accepted	‘1994-11-19’	Expected Result	
2.17.	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Accepted	‘bob@gmail.com’	Expected Result	
2.18.	Normal	Accepted	‘01234-567899’/ ‘01234 567899’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Extreme/ Boundary	Error message displayed	‘01234 56789’/ ‘ ’	Expected Result	
2.20.	Normal	Accepted	‘A1234’	Expected Result	
	Erroneous	Error message displayed	‘a1234’/ ‘-1’	Expected Result	
2.21.	Erroneous	Error message displayed	‘-1’/ ‘a’/ ‘199-1119’	Expected Result	
	Normal	Accepted	‘1994-11-19’	Expected Result	
2.24.	Extreme/ Boundary	Error message displayed	‘0’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Accepted	‘1’	Expected Result	
2.25.	Normal	Accepted	‘Jack’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘jack’	Expected Result	
2.26.	Normal	Accepted	‘Cox’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘cox’	Expected Result	
2.28.	Normal	Accepted	‘Jack’	Expected Result	Figure 8 - Test 2.28. page 123
	Erroneous	Error message	‘-1’/ ‘jack’	Expected	

		displayed		Result	
2.29.	Normal	Accepted	‘Cox’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘cox’	Expected Result	
2.30.	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Accepted	‘bob@gmail.com’	Expected Result	
2.31.	Extreme/ Boundary	Accepted	‘0’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Error message displayed	‘1’	Expected Result	
2.32	Normal	Accepted	‘Jack’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘jack’	Expected Result	
2.33.	Normal	Accepted	‘Cox’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘cox’	Expected Result	
2.34.	Normal	Accepted	‘User’/ ‘user1’	Expected Result	
	Erroneous	Error message displayed	‘U$er’	Expected Result	
2.35.	Erroneous	Error message displayed	‘password’	Expected Result	
	Normal	Accepted	‘password1’	Expected Result	
2.36.	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	Figure 9 - Test 2.36. page 123
	Normal	Accepted	‘bob@gmail.com’	Expected Result	
2.38.	Extreme/ Boundary	Accepted	‘0’	Expected Result	
	Erroneous	Error message displayed	‘-1’/ ‘a’	Expected Result	
	Normal	Error message displayed	‘1’	Expected Result	
2.39.	Normal	Accepted	‘User’/ ‘user1’	Expected Result	
	Erroneous	Error message displayed	‘U$er’	Expected Result	
2.40.	Normal	Accepted	‘User’/ ‘User1’	Expected Result	
	Erroneous	Error message displayed	‘U$er’	Expected Result	

2.41.	Normal	Accepted	‘password’	Expected Result	
	Erroneous	Error message displayed	‘password1’	Expected Result	
3.1.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘BUPA’	Expected Result	Figure 10 - Test 3.1. page 124
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘BUPA’	Expected Result	
3.2.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘WPA’	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘WPA’	Expected Result	
3.3.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘PPP’	Expected Result	Figure 11 - Test 3.3. page 125
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘PPP’	Expected Result	
3.4.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘AVIVA’	Expected Result	Figure 12 - Test 3.4. page 126
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘AVIVA’	Expected Result	
3.5.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: ‘PruHealth’	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: ‘PruHealth’	Expected Result	
3.6.	Normal	Prices calculated	Codes : A1234, B1234,	Expected	

		correctly	C1234 Incompatible: None Insurance: 'Cigna'	Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'Cigna'	Expected Result	
3.7.	Normal	Prices calculated correctly	Codes : A1234, B1234, C1234 Incompatible: None Insurance: 'Simply Health'	Expected Result	
	Normal	Prices calculated correctly	Codes : A1234, B1234, D1234 Incompatible: A1234 and D1234 Insurance: 'Simply Health'	Expected Result	
4.1.	Normal	Data stored correctly	Code Name: 'J9999' Incompatible: 'A1234' Prices- BUPA: '1, 2' WPA: '3, 4' PPP: '5, 6' AVIVA: '7, 8' PruHealth: '9, 10' Cigna: '11, 12' Simply Health: '13, 14'	Expected Result	Figure 13 - Test 4.1. page 127
4.2.	Normal	Data updated correctly	Code Name used in 4.1. 'J9999'	Failed Result. Button was inactive and nothing happened when clicked.	Figure 14 - Test 4.2. page 128
4.3.	Normal	Data deleted correctly	Code Name used in 4.1. 'J9999'	Expected Result	
4.4.	Normal	Data stored correctly	Title: 'Mr' First Name: 'Bob' Surname: 'Jones' Address Line 1: '24 Mill' Address Line 2: 'Cherry' Town: 'Cambridge' Post Code: 'JD9 5BJ'	Expected Result	

			Date of Birth: '1994-12-12' Gender: 'Male' Email: 'bj@google.com' Telephone: '01210-123456' Insurance: 'BUPA' Codes: 'A1234,B1234' Date of Operation: '2001-05-13' Consultant: 'Dr Jim Howard'		
4.5.	Normal	Data updated correctly	Patient Name used in 4.4. 'Bob', 'Jones'	Expected Result	
4.6.	Normal	Data deleted correctly	Patient Name used in 4.4. 'Bob', 'Jones'	Expected Result	
4.7.	Normal	Data stored correctly	Title: 'Dr' First Name: 'Andy' Surname: 'Leese' Email: 'con@smart.com'	Expected Result	Figure 15 - Test 4.7. page 128
4.8.	Normal	Data updated correctly	Consultant Name used in 4.7. 'Andy', 'Leese'	Expected Result	
4.9.	Normal	Data deleted correctly	Consultant Name used in 4.7. 'Andy', 'Leese'	Expected Result	Figure 16 - Test 4.9. page 129
4.10.	Normal	Data stored correctly	Username: 'Batman' Passwrod: 'Robin' Email: 'b@batcave.com' Access: 'Admin'	Expected Result	
4.11.	Normal	Data updated correctly	Username used in 4.10. 'Batman'	Expected Result	
4.12.	Normal	Data deleted correctly	Username used in 4.10. 'Batman'	Expected Result	
5.1.	Normal	Error message displayed	A non-existent user, eg. 'abcdefg'	Expected Result	Figure 17 - Test 5.1. page 129
5.2.	Normal	Error message displayed	An existing user with incorrect password, eg. 'Jack', 'Lemon'	Expected Result	Figure 18 - Test 5.2. page 130
5.3.	Normal	Error message displayed	Click "Operations" main button on the home screen, and all tabs within the "Operation" menu bar. Whilst not logged in.	Expected Result	

5.4.	Normal	Error message displayed	Click “Codes” main button on the home screen, and all tabs within the “Code” menu bar. Whilst not logged in.	Expected Result	Figure 19 - Test 5.4. page 130
5.5.	Normal	Error message displayed	Click “Users” main button on the home screen, and all tabs within the “User” menu bar. Whilst not logged in.	Expected Result	
5.6.	Normal	Error message displayed	Click “Patients” main button on the home screen, and all tabs within the “Patient” menu bar. Whilst not logged in.	Expected Result	
5.7.	Normal	Error message displayed	Click “Consultants” main button on the home screen, and all tabs within the “Consultant” menu bar. Whilst not logged in.	Expected Result	
5.8.	Normal	“Operation” Window displayed	Click “Operations” main button on the home screen, and all tabs within the “Operation” menu bar. Whilst logged in with ‘User’ level access.	Expected Result	
5.9.	Normal	“Code” Window displayed	Click “Codes” main button on the home screen, and all tabs within the “Code” menu bar. Whilst logged in with ‘User’ level access.	Expected Result	
5.10.	Normal	Error message displayed	Click “Users” main button on the home screen, and all tabs within the “User” menu bar Whilst logged in with ‘User’ level access.	Expected Result	Figure 20 - Test 5.10. page 131
5.11.	Normal	“Patient” Window displayed	Click “Patients” main button on the home screen, and all tabs within the “Patient” menu bar. Whilst	Expected Result	

			logged in with ‘User’ level access.		
5.12.	Normal	“Consultant” Window displayed	Click “Consultants” main button on the home screen, and all tabs within the “Consultant” menu bar. Whilst logged in with ‘User’ level access.	Expected Result	
5.13.	Normal	“Operation” Window displayed	Click “Operations” main button on the home screen, and all tabs within the “Operation” menu bar. Whilst logged in with ‘Admin’ level access.	Expected Result	
5.14.	Normal	“Code” Window displayed	Click “Codes” main button on the home screen, and all tabs within the “Code” menu bar. Whilst logged in with ‘Admin’ level access.	Expected Result	
5.15.	Normal	“User” Window displayed	Click “Users” main button on the home screen, and all tabs within the “User” menu bar Whilst logged in with ‘Admin’ level access.	Expected Result	
5.16.	Normal	“Patient” Window displayed	Click “Patients” main button on the home screen, and all tabs within the “Patient” menu bar. Whilst logged in with ‘Admin’ level access.	Expected Result	
5.17.	Normal	“Consultant” Window displayed	Click “Consultants” main button on the home screen, and all tabs within the “Consultant” menu bar. Whilst logged in with ‘Admin’ level access.	Expected Result	Figure 21 - Test 5.17. page 132
6.1.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	Figure 22 - Test 6.1 page 133
6.2.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	

6.3.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	
6.4.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	
6.5.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	Figure 23 - Test 6.5. page 134
6.7.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	
6.8.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	
6.9.	Erroneous	Error message displayed	Click the “Search” button with no method selected.	Expected Result	
6.10.	Normal	Error message displayed	Click the “Search” button after entering non-existing data.	Expected Result	

6.2. Known Issues

There are no known issues with the existing code as the issues were rectified during the initial testing phase of the project. However the known issues lie in the incomplete module that was not tested, the encryption module which can be seen in section 10.32. This would be the main concern and area of the system to develop further first if the system is to be maintained.

7. Code Explanations

Each of the subheadings contains a reference to section 10. of this document and line numbers of the selected code.

7.1. Difficult Sections

The algorithms in this section have been broken down into sections indicated with [#] for a step by step explanation.

7.1.1. Check Compatibility (Section: 10.5, Lines: 70-88)

```
def check_compatibility(controller,codes,cost):
    [1]
    comp = controller.get_compatibilty(codes)
    [2]
    invalid = []
    for count in range(len(comp)):
        subinvalid = []
        for item in comp[count]:
            subinvalid.append(item[0][0])
        invalid.append(subinvalid)
    [3]
    incompatible = []
    for count in range(len(codes)):
        for step in range(len(invalid)):
            if codes[count] in invalid[step]:
                incompatible.append([count,step])
    [4]
    for item in incompatible:
        if len(cost) != 1:
            if item[0]>item[1]:
                del cost[item[0]]
            else:
                del cost[item[1]]
    return cost
```

This function proved challenging to implement there are various occasions with loops inside loops which can get extremely confusing.

[1] The first step of the function simply retrieves an array of incompatible list data and assigns it to the ‘comp’ variable.

[2] After assigning an empty array to ‘invalid’, the first FOR loop creates a second empty array called ‘subinvalid’ before entering a second FOR loop. This loop adds the first list in ‘comp’ array to ‘subinvalid’, which is then added to ‘invalid’, creating an array of lists. The outer FOR loop then repeats this creating a formatted array of lists.

[3] After assigning an empty array to ‘incompatible’, the two loops give a stepper variable within a stepper variable, which allows the iteration through each list in the array (outer loop), followed by the iteration through each item within that list (inner loop). The stepper variables are then used separately to search within each list for each of the original codes, if found the locations are stored as a list in the ‘incompatible’ array.

[4] The finally section of the function only uses one FOR loop to look at each location in the costs list using the locations previously stored in ‘incompatible’. The IF and ELSE statements

compare the costs from each location and remove the smaller code. Finally the function returns the amended cost array.

7.1.2. Add Invalid Combination (Section: 10.7, Lines: 19-45)

```
def add_invalid_combination(self,ID,invalid):
        [1]
        for item in invalid:
            sql = """select CodeID from code where Code =
'{0}'""".format(item)
            existing = False
            [2]
            while existing != True:
                invalidIDtuple = self.select_query(sql)
                existing = True
                [3]
                if invalidIDtuple == []:
                    sql2 = """insert into code
                              (Code)
                             values
                              ('{0}')""".format(item)
                    self.query(sql2)
                    sql2 = """select CodeID from code where Code =
'{0}'""".format(item)
                    [4]
                    newIDtuple = self.select_query(sql2)
                    newID = newIDtuple[0][0]
                    self.add_code_cost(newID,[0,0,0,0,0,0,0,0,0,0,0,0,0,0])
                    existing = False
            [5]
            invalidIDstr = str(invalidIDtuple[0])
            invalidID = int(invalidIDstr[1])
            sql = """insert into invalidCombination
                      (CodeID,InvalidCode)
                     values
                      ('{0}','{1}')""".format(ID,invalidID)
            self.query(sql)
```

This function caused multiply problems during implementation as when adding a code it would only allow existing codes to be incompatible. This was solved by creating the code if it was non-existent.

[1] The loop repeats the entire function for each of the codes in the ‘invalid array’. The first step within the loop is to declare the SQL statement to retrieve the matching code ID and set the ‘existing’ variable to False.

[2] The WHILE loop is based on the ‘existing’ variable being set as False which allows the loops to repeat either once or twice as we will see. ‘existing’ is then set to True after the execution of the previous stated SQL.

[3] If the code is non-existent the code inside the IF statement us run, if not the system jumps to part [5]. The first step in the IF statement declares a new SQL string which adds the code to the database, after execution the new SQL string is then re-declared to select the ID of the added code.

[4] The first line fetches the tuple containing the ID of the added code, and the following line formats the tuple to a string. This ID is then used to add code costs of 0 to the database to prevent the system from crashing, these can then be amended by the user later. ‘existing’ is then set to False, which revisits the WHILE loop now that the code is existent. This returns to

section [2] to fetch the tuple, and as the code is now existent the IF statement is skipped and we drop out on the while loop.

[5] The tuple is then validated into a sting before the original SQL statement is re-declared to add the invalid combination using the 2 IDs, and then executed. This entire process is repeated due to the FOR loop around the entire function.

7.1.3. Populating a QTable (Section: 10.22, Lines: 37-44)

```
[1]
for row in range(len(self.pdata)):
            [2]
            for col in range(len(self.pdata[row])):
                [3]
                if col == 0:
self.outputTable.setItem(row,col,QTableWidgetItem(str(self.pdata[row][col])
))
                else:
self.outputTable.setItem(row,col,QTableWidgetItem(self.pdata[row][col]))
```

Although a short algorithm I took me a while to implement a population method for my table widget.

[1] The outer loop uses a stepper variable, ‘row’, to iterate though each row in the table.

[2] The inner loop also uses a stepper variable, ‘col’, to iterate through each column in the selected row. This allows the fields for each record to be populated.

[3] The final stage of the algorithm populates the table, the IF statement decides if the column is the ID column as the ID has to be converted to a string in order to be accepted by the table.

7.2. Self-created Algorithms

7.2.1. Compiling a list of Codes (Section: 10.1, Lines: 108-123)

```
[1]
if invalid != "":
            letterlist = []
            [2]
            for count in range(len(invalid)):
                if invalid[count] != ',':
                    letterlist.append(invalid[count])
            [3]
            invalid = []
            for count in range(0,len(letterlist),5):
                try:
                    currentcode =
letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
3]+letterlist[count+4]
                    invalid.append(currentcode)
                    self.valid = True
                except:
                    self.valid = False
                    self.errormsg = "Error- Invalid:\nIncompatible Input"
```

[1] The first IF statement decides whether the variable is empty or not, if not it is entered and an empty array, ‘letterlist’, is instantiated.

[2] This FOR loop iterates through each letter in the ‘invalid’ string, the IF statement then appends the c to the ‘letterlist’ if the character is not a coma.

[3] A new list is then instantiated called ‘invalid’. The FOR loop consist of error handling in case the ‘letterlist’ is empty. Letters are taken from the ‘letterlist’ in groups of 5 and compiled to make a code, which is then added to the ‘invalid’ code list.

7.2.2. Compiling a Consultant's Last Name (Section: 10.3, Lines: 131-141)

```
[1]
con = self.conComboBox.currentText()
letterList = []
step = 0
      [2]
        for item in con:
            if item == ' ':
                step += 1
            if step == 2 and item != ' ':
                letterList.append(item)
      [3]
        conlname = ''
        for item in letterList:
            conlname += item
```

[1] The first line in this algorithm retrieves the consultant's compiled name from the combo box. An empty array is then created and the 'step' variable is set to 0.

[2] This FOR loop steps through each character in the consultant string. Each time loop encounters a space the 'step' variable is incremented. This allows the loop to select the surname using the IF statements, and append each letter in the surname to the 'letterList'.

[3] An empty string is created which will eventually hold the entire consultants surname, 'conlname'. The FOR loop then adds each character in the 'letterList' to this string, generating the name.

7.2.3. Validating Data (Section: 10.13, Lines: 12-18)

```
[1]
errormsg = "Error- Invalid:\n"
validdata = True
if title != None:
            valid = validate_data(title,"Mr|Mrs|Ms|Miss|Dr|Prof|Sir")
            [2]
            if valid == False:
                validdata = False
                errormsg += "Title\n"
```

[1] Before validating a group of data an error message string is created in order to add any invalid pieces of data to. A general valid statement is set to true so that it can be flipped to false if a single piece of data is invalid. If the data is not blank then the IF statement is entered and the validation method is called, returning true of false.

[2] The second IF statement is entered if the return from validating the data was ‘False’. If so then ‘validdata’ is declared ‘False’ and the data name is added to the error message string.

This process would be repeated for each piece of data in a set of data from an input form. These groups can be seen in the ‘gui_validation’ module in section 10.13. of this document as functions.

7.2.4. Generating Invoice HTML (Section: 10.15, Lines: 47-70)

```
[1]
codeBullets = ""
for item in self._patientCodes:
        codeBullets += "<li>{0}</li>".format(item)
[2]
html = """<html>
                <body>
                <p>{0} {1} {2}<br>{3}<br>{4}<br>{5}<br>{6}</p>
                <p>Dear {0} {1} {2},<br>This invoice is from {7} {8} {9}
for the following procedures which occured on {10}:</p>
                <ul>
                {15}
                </ul>
                <p>Suregon Price- £{11}<br>Anaesthetist Price-
£{12}<br>Total Price- £{13}</p>
                <p>Sent from Cambrige Medical Practice on the {14}</p>
                </body>
</html>""".format(self._patientTitle,self._patientFirstName,self._patientLa
stName,self._patientAddressLine1,self._patientAddressLine2,self._patientTow
n,self._patientPostCode,self._consultantTitle,self._consultantFirstName,sel
f._consultantLastName,self._opDate,self._surgeonPrice,self._anaesthetistPri
ce,(self._surgeonPrice+self._anaesthetistPrice),self._date,codeBullets)
[3]
htmlfile = open("Patient_Invoice.html","w")
htmlfile.write(html)
htmlfile.close()
```

[1] To begin with the bullet points for the html need to be created; therefor an empty string is created. The FOR loop then iterates through all of the codes related to the patient and inserts a html list object into the string.

[2] The html structure is then generated and populated using the .format method.

[3] Finally a html file is opened, then html is stored in this file, then the link to the file is terminated. This allows the html file to be opened and displayed in a browser in another function.

7.2.5. Log In Details (Section: 10.17, Lines: 214-224)

```
[1]
self.get_user_details(self.userLineEdit.text())
        if self.data == []:
            QMessageBox.about(self, "Error", "User does not exist.")
        else:
            [2]
            if self.passLineEdit.text() == self.data[0][2]:
                QMessageBox.about(self, "System", "{0} Access
Granted.".format(self.data[0][3]))
                self.logWindow.close()
                self.access = self.data[0][3]
            else:
                QMessageBox.about(self, "Error", "Incorrect Password.")
```

[1] The data is retrieved from the controller using the username from the username line edit. If the data is return as blank the first IF statement will detect this and display and error message to the user. If the data is not blank the ELSE statement is entered (see [2]).

[2] Inside this ELSE statement there is another pair of statements. The first tries to match the input password with the stored password, if matching then the user is granted access to the system which is stored in the ‘self.access’ variable. If they do not match the ELSE statement displays an error message.

7.2.6. Generating a Report (Section: 10.19, Lines: 60-80)

```
[1]
if self.search == "con":
            title = "Patient Report - {0}".format(self.opData[0][3])
        else:
            title = "Patient Report - {0}".format(self.opData[0][2])
[2]
data = ""
for count in range(len(self.opData)):
            currentData = "<p>{0}<br>{1} - {2}<br>Suregon Price-
£{3}<br>Anaesthetist Price- £{4}<br>Total Price-
£{5}</p>".format(self.opData[count][1],self.opData[count][3],self.opData[co
unt][2],self.prices[count][0],self.prices[count][1],self.prices[count][1]+s
elf.prices[count][0])
            data += currentData
       [3]
         html = """<html>
                  <body>
                  <p>{0}<br><br>{1}
                  <p>Produced by- {2}, Date- {3}
                  </body>
                  </html>""".format(title,data,UN,date)
         htmlfile = open("Operation_Report.html","w")
         htmlfile.write(html)
         htmlfile.close()
```

[1] The first set of statements decides the title of the report by checking if the search criteria is by consultant or date.

[2] Secondly a blank data string is instantiated, before a FOR loop which cycles through each patient who will be in the report. The data regarding the patient in each iteration is the formatted into html and appended to the ‘data’ string.

[3] Once the ‘data’ string contains all the patient information in html format, it is combined with the previously established title to make the html for the report. This html is then written to an opened file before the connection to the file is closed again.

8. Settings

No settings needed to be changed on my client's computers as no additional modules are required to run the system. Therefore only Python and PyQt were installed as the modules used are all built in to these.

However if anyone was to continue with the maintenance of the system and develop the encryption module to a usable state, they would need to install PyCrypto[1].

[1] http://www.voidspace.org.uk/python/modules.shtml#pycrypto

9. Acknowledgements

Although I was able to create the html in python I had no knowledge of how to open and display the html file in a browser. Therefore I would like to acknowledge where I sourced the code from: http://pythonconquerstheuniverse.wordpress.com/2010/10/16/how-to-open-a-web-browser-from-python/

The creation of required the use of eight images which were sources from various websites. These are shown in the table below to acknowledge the origin of these images.

Image Label	Image	Source Link
Operations		http://www.iconarchive.com/show/pretty-office-4-icons-by-custom-icon-design/report-icon.html
Code		http://www.iconfinder.com/icondetails/54479/128/code_folder_golden_icon

User		http://www.softicons.com/free-icons/web-icons/user-icons-by-2shi/user-1-icon
Patient		http://www.iconarchive.com/show/medical-icons-by-devcom/pregnant-icon.html
Consultant		http://iconbug.com/detail/icon/424/man-in-suit/
Add		http://www.iconfinder.com/icondetails/18499/32/add_user_icon
Search		http://dryicons.com/icon/shine-icon-set/world-search/

Calculator		http://www.iconarchive.com/show/openphone-icons-by-walrick/Calculator-icon.html

10. Code Listing Appendix

The highlight code in sections 10.15. and 10.19. was sources from the web as during implementation I lacked to knowledge of display html files. The source is explained in the first paragraph of section 9. where the origin of the code is labelled with a [1]

Sections 10.33 and 10.34 contain modules which are only used by the CLI system, which can be found at the bottom of all of the controller modules.

Section 10.32 is the incomplete encryption module which is not called by any other modules as it does not function properly and I had no time to complete it.

10.1. add_code_gui.py

```
1   from PyQt4.QtGui import *
2   from PyQt4.QtCore import *
3   import sys
4   import code_controller
5   import gui_validation
6
7   class addCodeWindow(QMainWindow):
8       def __init__(self):
9           super().__init__()
10          self.createUI()
11
12      def createUI(self):
13          self.addView = QWidget()
14          self.addView.setLayout(self.addCodeGrid())
15          self.setCentralWidget(self.addView)
16          self.setWindowTitle("Code - Add Code")
17          self.setWindowIcon(QIcon('images/CMP_Logo.png'))
18
19      def addCodeGrid(self):
20          self.addCodeLayout = QGridLayout()
21          self.nameLabel = QLabel("Code Name")
22          self.inconpatibleLabel = QLabel("Incompatible Codes")
23
24          self.nameLineEdit = QLineEdit()
25          self.inconpatibleLineEdit = QLineEdit()
26          self.addButton = QPushButton("Add Code")
27
28          self.priceLayout = QGridLayout()
29          self.surgeonLabel = QLabel("Surgeon Price:")
30          self.anaesthetistLabel = QLabel("Anaesthetist Price:")
31          self.bupaApLineEdit = QLineEdit()
32          self.wpaApLineEdit = QLineEdit()
33          self.pppApLineEdit = QLineEdit()
34          self.avivaApLineEdit = QLineEdit()
35          self.pruApLineEdit = QLineEdit()
36          self.cignaApLineEdit = QLineEdit()
37          self.simplyApLineEdit = QLineEdit()
38          self.bupaSpLineEdit = QLineEdit()
39          self.wpaSpLineEdit = QLineEdit()
40          self.pppSpLineEdit = QLineEdit()
41          self.avivaSpLineEdit = QLineEdit()
42          self.pruSpLineEdit = QLineEdit()
43          self.cignaSpLineEdit = QLineEdit()
44          self.simplySpLineEdit = QLineEdit()
45
46          self.priceLayout.addWidget(self.surgeonLabel,0,0)
```

```
47            self.priceLayout.addWidget(self.anaesthetistLabel,0,1)
48            self.priceLayout.addWidget(self.bupaSpLineEdit,1,0)
49            self.priceLayout.addWidget(self.wpaSpLineEdit,2,0)
50            self.priceLayout.addWidget(self.pppSpLineEdit,3,0)
51            self.priceLayout.addWidget(self.avivaSpLineEdit,4,0)
52            self.priceLayout.addWidget(self.pruSpLineEdit,5,0)
53            self.priceLayout.addWidget(self.cignaSpLineEdit,6,0)
54            self.priceLayout.addWidget(self.simplySpLineEdit,7,0)
55            self.priceLayout.addWidget(self.bupaApLineEdit,1,1)
56            self.priceLayout.addWidget(self.wpaApLineEdit,2,1)
57            self.priceLayout.addWidget(self.pppApLineEdit,3,1)
58            self.priceLayout.addWidget(self.avivaApLineEdit,4,1)
59            self.priceLayout.addWidget(self.pruApLineEdit,5,1)
60            self.priceLayout.addWidget(self.cignaApLineEdit,6,1)
61            self.priceLayout.addWidget(self.simplyApLineEdit,7,1)
62
63            self.insuranceLayout = QVBoxLayout()
64            self.insuranceLabel = QLabel("Insurance Company:",self)
65            self.bupaLabel = QLabel("BUPA",self)
66            self.wpaLabel = QLabel("WPA",self)
67            self.pppLabel = QLabel("PPP",self)
68            self.avivaLabel = QLabel("AVIVA",self)
69            self.pruLabel = QLabel("Pru Health",self)
70            self.cignaLabel = QLabel("Cigna",self)
71            self.simplyLabel = QLabel("Simply Health",self)
72
73            self.insuranceLayout.addWidget(self.insuranceLabel)
74            self.insuranceLayout.addWidget(self.bupaLabel)
75            self.insuranceLayout.addWidget(self.wpaLabel)
76            self.insuranceLayout.addWidget(self.pppLabel)
77            self.insuranceLayout.addWidget(self.avivaLabel)
78            self.insuranceLayout.addWidget(self.pruLabel)
79            self.insuranceLayout.addWidget(self.cignaLabel)
80            self.insuranceLayout.addWidget(self.simplyLabel)
81
82            self.addCodeLayout.addWidget(self.nameLabel,0,0)
83            self.addCodeLayout.addWidget(self.inconpatibleLabel,1,0)
84            self.addCodeLayout.addWidget(self.nameLineEdit,0,1)
85            self.addCodeLayout.addWidget(self.inconpatibleLineEdit,1,1)
86            self.addCodeLayout.addLayout(self.insuranceLayout,2,0)
87            self.addCodeLayout.addLayout(self.priceLayout,2,1)
88            self.addCodeLayout.addWidget(self.addButton,3,1)
89
90            self.addButton.clicked.connect(self.add)
91            return self.addCodeLayout
92
93        def add(self):
94            code = self.nameLineEdit.text()
95            costs =
96    [self.bupaSpLineEdit.text(),self.bupaApLineEdit.text(),self.wpaSpLineEdit.t
97    ext(),self.wpaApLineEdit.text(),self.pppSpLineEdit.text(),self.pppApLineEdi
98    t.text(),self.avivaApLineEdit.text(),self.avivaApLineEdit.text(),self.pruSp
99    LineEdit.text(),self.pruApLineEdit.text(),self.cignaSpLineEdit.text(),self.
100   cignaApLineEdit.text(),self.simplySpLineEdit.text(),self.simplyApLineEdit.t
101   ext()]
102           for count in range(14):
103               if costs[count] == "":
104                   costs[count] = 0
105               else:
106                   costs[count] = int(costs[count])
107           invalid = self.inconpatibleLineEdit.text()
```

```
108             if invalid != "":
109                 letterlist = []
110                 for count in range(len(invalid)):
111                     if invalid[count] != ',':
112                         letterlist.append(invalid[count])
113                 invalid = []
114                 for count in range(0,len(letterlist),5):
115                     try:
116                         currentcode =
117     letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
118     3]+letterlist[count+4]
119                         invalid.append(currentcode)
120                         self.valid = True
121                     except:
122                         self.valid = False
123                         self.errormsg = "Error- Invalid:\nIncompatible Input"
124             else:
125                 self.valid = True
126             if self.valid != False:
127                 self.valid, self.errormsg =
128     gui_validation.validate_code(code,costs,invalid)
129             if self.valid == False:
130                 QMessageBox.about(self, "Error", self.errormsg)
131             else:
132                 self.tempController = code_controller.Code_Controller()
133                 data = self.tempController.code_details(code)
134                 if data != None:
135                       QMessageBox.about(self, "Error", "Error- Code already
136     exists.")
137                 else:
138                       self.tempController.add_code(code,costs,invalid)
139                       QMessageBox.about(self, "System", "Code successfully
140     added.")
141                       self.close()
142
143     if __name__ == "__main__":
144         application = QApplication(sys.argv)
145         addCodeWindow = addCodeWindow()
146         addCodeWindow.show()
147         addCodeWindow.raise_()
148         application.exec_()
```

10.2. add_consultant_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import consultant_controller
import gui_validation

class addConsultantWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.addView = QWidget()
        self.addView.setLayout(self.addConsultantGrid())
        self.setCentralWidget(self.addView)
        self.setWindowTitle("Consultant - Add Consultant")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def addConsultantGrid(self):
        self.addConsultantLayout = QGridLayout()
        self.titleLabel = QLabel('Title')
        self.fnameLabel = QLabel('First Name')
        self.lnameLabel = QLabel('Last Name')
        self.emailLabel = QLabel('Email')
        self.fnameLineEdit = QLineEdit()
        self.lnameLineEdit = QLineEdit()
        self.emailLineEdit = QLineEdit()
        self.titleComboBox = QComboBox()
        self.titles = ["-","Mr","Mrs","Ms","Miss","Dr","Prof","Sir"]
        for item in self.titles:
            self.titleComboBox.addItem(item)
        self.addButton = QPushButton('Add Consultant')
        self.addConsultantLayout.addWidget(self.titleLabel,0,0)
        self.addConsultantLayout.addWidget(self.fnameLabel,1,0)
        self.addConsultantLayout.addWidget(self.lnameLabel,2,0)
        self.addConsultantLayout.addWidget(self.emailLabel,3,0)
        self.addConsultantLayout.addWidget(self.fnameLineEdit,1,1)
        self.addConsultantLayout.addWidget(self.lnameLineEdit,2,1)
        self.addConsultantLayout.addWidget(self.emailLineEdit,3,1)
        self.addConsultantLayout.addWidget(self.titleComboBox,0,1)
        self.addConsultantLayout.addWidget(self.addButton,4,1)
        self.addButton.clicked.connect(self.add)
        return self.addConsultantLayout

    def add(self):
        self.valid, self.errormsg =
gui_validation.validate_consultant(self.titleComboBox.currentText(),self.fn
ameLineEdit.text(),self.lnameLineEdit.text(),self.emailLineEdit.text())
        if self.valid == False:
            QMessageBox.about(self, "Error", self.errormsg)
        else:
            tempController = consultant_controller.Consultant_Controller()

tempController.add_consultant(self.titleComboBox.currentText(),self.fnameLi
neEdit.text(),self.lnameLineEdit.text(),self.emailLineEdit.text())
            QMessageBox.about(self, "System", "Consultant successfully
added.")
            self.close()

if __name__ == "__main__":
```

```
61      application = QApplication(sys.argv)
62      addConsultantWindow = addConsultantWindow()
63      addConsultantWindow.show()
64      addConsultantWindow.raise_()
65      application.exec_()
```

10.3. add_patient_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import patient_controller
import consultant_controller
import gui_validation

class addPatientWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.addView = QWidget()
        self.addView.setLayout(self.addPatientGrid())
        self.setCentralWidget(self.addView)
        self.setWindowTitle("Patient - Add Patient")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def addPatientGrid(self):
        self.addPatientLayout = QGridLayout()
        self.titleLabel = QLabel("Title")
        self.fnameLabel = QLabel("First Name")
        self.lnameLabel = QLabel("Surname")
        self.ad1Label = QLabel("Address Line 1")
        self.ad2Label = QLabel("Address Line 2")
        self.townLabel = QLabel("Town")
        self.pcLabel = QLabel("Post Code")
        self.dobLabel = QLabel("Date of Birth")
        self.genderLabel = QLabel("Gender")
        self.emailLabel = QLabel("Email")
        self.telLabel = QLabel("Telephone Number")
        self.icLabel = QLabel("Insurance")
        self.codesLabel = QLabel("Codes")
        self.dooLabel = QLabel("Date of Operation")
        self.conLabel = QLabel("Consultant")

        self.titleComboBox = QComboBox()
        self.titles = ["-","Mr","Mrs","Ms","Miss","Dr","Prof","Sir"]
        for item in self.titles:
            self.titleComboBox.addItem(item)

        self.genderComboBox = QComboBox()
        self.genders = ["-","Male","Female"]
        for item in self.genders:
            self.genderComboBox.addItem(item)

        self.icComboBox = QComboBox()
        self.ics = ["-
","BUPA","WPA","PPP","AVIVA","PruHealth","Cigna","SimplyHealth"]
        for item in self.ics:
            self.icComboBox.addItem(item)

        self.fnameLineEdit = QLineEdit()
        self.lnameLineEdit = QLineEdit()
        self.ad1LineEdit = QLineEdit()
        self.ad2LineEdit = QLineEdit()
        self.townLineEdit = QLineEdit()
        self.pcLineEdit = QLineEdit()
        self.dobLineEdit = QLineEdit()
```

```
61            self.emailLineEdit = QLineEdit()
62            self.telLineEdit = QLineEdit()
63            self.codesLineEdit = QLineEdit()
64            self.dooLineEdit = QLineEdit()
65
66            self.addButton = QPushButton("Add Patient")
67            self.createConsultantComboBox()
68
69            self.addPatientLayout.addWidget(self.titleLabel,0,0)
70            self.addPatientLayout.addWidget(self.fnameLabel,1,0)
71            self.addPatientLayout.addWidget(self.lnameLabel,2,0)
72            self.addPatientLayout.addWidget(self.ad1Label,3,0)
73            self.addPatientLayout.addWidget(self.ad2Label,4,0)
74            self.addPatientLayout.addWidget(self.townLabel,5,0)
75            self.addPatientLayout.addWidget(self.pcLabel,6,0)
76            self.addPatientLayout.addWidget(self.dobLabel,7,0)
77            self.addPatientLayout.addWidget(self.genderLabel,0,2)
78            self.addPatientLayout.addWidget(self.emailLabel,1,2)
79            self.addPatientLayout.addWidget(self.telLabel,2,2)
80            self.addPatientLayout.addWidget(self.titleComboBox,0,1)
81            self.addPatientLayout.addWidget(self.fnameLineEdit,1,1)
82            self.addPatientLayout.addWidget(self.lnameLineEdit,2,1)
83            self.addPatientLayout.addWidget(self.ad1LineEdit,3,1)
84            self.addPatientLayout.addWidget(self.ad2LineEdit,4,1)
85            self.addPatientLayout.addWidget(self.townLineEdit,5,1)
86            self.addPatientLayout.addWidget(self.pcLineEdit,6,1)
87            self.addPatientLayout.addWidget(self.dobLineEdit,7,1)
88            self.addPatientLayout.addWidget(self.genderComboBox,0,3)
89            self.addPatientLayout.addWidget(self.emailLineEdit,1,3)
90            self.addPatientLayout.addWidget(self.telLineEdit,2,3)
91
92            self.addPatientLayout.addWidget(self.icLabel,3,2)
93            self.addPatientLayout.addWidget(self.icComboBox,3,3)
94            self.addPatientLayout.addWidget(self.codesLabel,4,2)
95            self.addPatientLayout.addWidget(self.dooLabel,5,2)
96            self.addPatientLayout.addWidget(self.conLabel,6,2)
97            self.addPatientLayout.addWidget(self.conComboBox,6,3)
98            self.addPatientLayout.addWidget(self.codesLineEdit,4,3)
99            self.addPatientLayout.addWidget(self.dooLineEdit,5,3)
100
101           self.addPatientLayout.addWidget(self.addButton,7,3)
102           self.addButton.clicked.connect(self.add)
103           return self.addPatientLayout
104
105       def createConsultantComboBox(self):
106           tempController = consultant_controller.Consultant_Controller()
107           self.conData = tempController.all_details()
108           self.conComboBox = QComboBox()
109           self.conComboBox.addItem('-')
110           for item in self.conData:
111               con = item[1] + ' ' + item[2] + ' ' + item[3]
112               self.conComboBox.addItem(con)
113
114       def add(self):
115           tempController = patient_controller.Patient_Controller()
116           letterlist = []
117           for count in range(len(self.codesLineEdit.text())):
118               if self.codesLineEdit.text()[count] != ',':
119                   letterlist.append(self.codesLineEdit.text()[count])
120           self.codes = []
121           self.valid = True
```

```
        for count in range(0,len(letterlist),5):
            try:
                currentcode =
letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
3]+letterlist[count+4]
                self.codes.append(currentcode)
            except:
                self.valid = False
                self.errormsg = "Error- Invalid:\nCode Input"
        con = self.conComboBox.currentText()
        letterList = []
        step = 0
        for item in con:
            if item == ' ':
                step += 1
            if step == 2 and item != ' ':
                letterList.append(item)
        conlname = ''
        for item in letterList:
            conlname += item
        if self.valid != False:
            self.valid, self.errormsg =
gui_validation.validate_patient(self.titleComboBox.currentText(),self.fname
LineEdit.text(),self.lnameLineEdit.text(),self.ad1LineEdit.text(),self.ad2L
ineEdit.text(),self.townLineEdit.text(),self.pcLineEdit.text(),self.dobLine
Edit.text(),self.genderComboBox.currentText(),self.telLineEdit.text(),self.
emailLineEdit.text(),self.dooLineEdit.text(),conlname,self.icComboBox.curre
ntText(),self.codes)
        if self.valid == False:
            QMessageBox.about(self, "Error", self.errormsg)
        else:
            tempController = patient_controller.Patient_Controller()

tempController.add_patient(self.titleComboBox.currentText(),self.fnameLineE
dit.text(),self.lnameLineEdit.text(),self.ad1LineEdit.text(),self.ad2LineEd
it.text(),self.townLineEdit.text(),self.pcLineEdit.text(),self.dobLineEdit.
text(),self.genderComboBox.currentText(),self.telLineEdit.text(),self.email
LineEdit.text(),self.dooLineEdit.text(),conlname,self.icComboBox.currentTex
t(),self.codes)
            QMessageBox.about(self, "System", "Patient successfully
added.")
            self.close()

if __name__ == "__main__":
    application = QApplication(sys.argv)
    addPatientWindow = addPatientWindow()
    addPatientWindow.show()
    addPatientWindow.raise_()
    application.exec_()
```

10.4. add_user_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import user_controller
import gui_validation

class addUserWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.addView = QWidget()
        self.addView.setLayout(self.addUserGrid())
        self.setCentralWidget(self.addView)
        self.setWindowTitle("User - Add User")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def addUserGrid(self):
        self.addUserLayout = QGridLayout()
        self.nameLabel = QLabel('Username')
        self.passLabel = QLabel('Password')
        self.emailLabel = QLabel('Email')
        self.accessLabel = QLabel('Access')
        self.nameLineEdit = QLineEdit()
        self.passLineEdit = QLineEdit()
        self.emailLineEdit = QLineEdit()
        self.accessComboBox = QComboBox()
        self.access = ["-","User","Admin"]
        for item in self.access:
            self.accessComboBox.addItem(item)
        self.addButton = QPushButton('Add User')
        self.addUserLayout.addWidget(self.nameLabel,0,0)
        self.addUserLayout.addWidget(self.passLabel,1,0)
        self.addUserLayout.addWidget(self.emailLabel,2,0)
        self.addUserLayout.addWidget(self.accessLabel,3,0)
        self.addUserLayout.addWidget(self.nameLineEdit,0,1)
        self.addUserLayout.addWidget(self.passLineEdit,1,1)
        self.addUserLayout.addWidget(self.emailLineEdit,2,1)
        self.addUserLayout.addWidget(self.accessComboBox,3,1)
        self.addUserLayout.addWidget(self.addButton,4,1)
        self.addButton.clicked.connect(self.add)
        return self.addUserLayout

    def add(self):
        self.valid, self.errormsg =
gui_validation.validate_user(self.nameLineEdit.text(),self.passLineEdit.tex
t(),self.accessComboBox.currentText(),self.emailLineEdit.text())
        if self.valid == False:
            QMessageBox.about(self, "Error", self.errormsg)
        else:
            tempController = user_controller.User_Controller()
            tempController.add_user(self.nameLineEdit.text(),self.passLineE
            dit.text(),self.accessComboBox.currentText(),self.emailLineEdit
            .text())
            QMessageBox.about(self, "System", "User successfully added.")
            self.close()

if __name__ == "__main__":
    application = QApplication(sys.argv)
```

```
61          addUserWindow = addUserWindow()
62          addUserWindow.show()
63          addUserWindow.raise_()
64          application.exec_()
```

10.5. code_calculator.py

```
import code_controller

#BUPA, WPA, PruHealth, Cigna, SimplyHealth
def calculation_method1(costs):
    if len(costs) == 1:
        sp = costs[0][0]
        ap = costs[0][1]
    elif len(costs) == 2:
        sp = (costs[0][0])*1.25
        ap = (costs[0][1])*1.25
    else:
        sp = (costs[0][0])*1.4
        ap = (costs[0][1])*1.4
    return sp, ap

#AVIVA
def calculation_method2(costs):
    if len(costs) == 1:
        sp = costs[0][0]
        ap = costs[0][1]
    elif len(costs) == 2:
        sp = (costs[0][0])+(costs[1][0]*0.5)
        ap = (costs[0][1])+(costs[1][1]*0.5)
    else:
        sp = (costs[0][0])+(costs[1][0]*0.5)+(costs[2][0]*0.25)
        ap = (costs[0][1])+(costs[1][1]*0.5)+(costs[2][1]*0.25)
    return sp, ap

#PPP
def calculation_method3(costs):
    if len(costs) == 1:
        sp = costs[0][0]
        ap = costs[0][1]
    else:
        sp = (costs[0][0])+(costs[1][0]*0.5)
        ap = (costs[0][1])+(costs[1][1]*0.5)
    return sp, ap

def get_code_costs(controller,codes,IC):
    un_costs, exist = controller.get_code_costs(codes,IC)
    if exist != False:
        if len(codes) == 1:
            or_costs = [un_costs[0][0]]
            or_codes = [codes[0]]
        elif len(codes) == 2:
            or_costs = [un_costs[0][0],un_costs[1][0]]
            or_codes = [codes[0],codes[1]]
        else:
            or_costs = [un_costs[0][0],un_costs[1][0],un_costs[2][0]]
            or_codes = [codes[0],codes[1],codes[2]]
        for count in range(len(or_costs)-1):
            for count in range(len(or_costs)-1):
                if or_costs[count][0] < or_costs[count+1][0]:
                    temp = or_costs[count]
                    or_costs[count] = or_costs[count+1]
                    or_costs[count+1] = temp
                    temp = or_codes[count]
                    or_codes[count] = or_codes[count+1]
                    or_codes[count+1] = temp
                else:
```

```
                    or_costs[count] = or_costs[count]
                    or_costs[count+1] = or_costs[count+1]
                    or_codes[count] = or_codes[count]
                    or_codes[count+1] = or_codes[count+1]
        return or_costs, or_codes
    else:
        return None, None


def check_compatibility(controller,codes,cost):
    comp = controller.get_compatibilty(codes)
    invalid = []
    for count in range(len(comp)):
        subinvalid = []
        for item in comp[count]:
            subinvalid.append(item[0][0])
        invalid.append(subinvalid)
    incompatible = []
    for count in range(len(codes)):
        for step in range(len(invalid)):
            if codes[count] in invalid[step]:
                incompatible.append([count,step])
    for item in incompatible:
        if len(cost) != 1:
            if item[0]>item[1]:
                del cost[item[0]]
            else:
                del cost[item[1]]
    return cost

def main(codes,IC):
    tempController = code_controller.Code_Controller()
    cost, codes = get_code_costs(tempController,codes,IC)
    if cost != None:
        costs = check_compatibility(tempController,codes,cost)
        if IC in ["BUPA", "WPA", "PruHealth", "Cigna", "SimplyHealth"]:
            sp, ap = calculation_method1(costs)
        if IC == "AVIVA":
            sp, ap = calculation_method2(costs)
        if IC == "PPP":
            sp, ap = calculation_method3(costs)
        return sp, ap
    else:
        print("One of the codes does not exist.")
        return None,None

if __name__ == "__main__":
    codes = ["A1234","T1234"]
    IC = "BUPA"
    main(codes,IC)
```

10.6. code_calculator_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import code_calculator
import gui_validation

class codeCalculatorWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.calculatorView = QWidget()
        self.calculatorView.setLayout(self.calculatorGrid())
        self.setCentralWidget(self.calculatorView)
        self.setWindowTitle("Code - Calculator")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def calculatorGrid(self):
        self.calculatorLayout = QVBoxLayout()
        self.calculateGroupBox = QGroupBox("Calculate...",self)
        self.calculateLayout = QGridLayout()
        self.code1Label = QLabel("Code 1:",self)
        self.code2Label = QLabel("Code 2:",self)
        self.code3Label = QLabel("Code 3:",self)
        self.icLabel = QLabel("Insurance:",self)
        self.code1LineEdit = QLineEdit()
        self.code2LineEdit = QLineEdit()
        self.code3LineEdit = QLineEdit()
        self.icComboBox = QComboBox()
        self.ics = ["-
","BUPA","WPA","PPP","AVIVA","PruHealth","Cigna","SimplyHealth"]
        for item in self.ics:
            self.icComboBox.addItem(item)
        self.calculateButton = QPushButton("Calculate")
        self.calculateLayout.addWidget(self.code1Label,0,0)
        self.calculateLayout.addWidget(self.code2Label,1,0)
        self.calculateLayout.addWidget(self.code3Label,2,0)
        self.calculateLayout.addWidget(self.code1LineEdit,0,1)
        self.calculateLayout.addWidget(self.code2LineEdit,1,1)
        self.calculateLayout.addWidget(self.code3LineEdit,2,1)
        self.calculateLayout.addWidget(self.icLabel,3,0)
        self.calculateLayout.addWidget(self.icComboBox,3,1)
        self.calculateLayout.addWidget(self.calculateButton,4,1)
        self.calculateGroupBox.setLayout(self.calculateLayout)

        self.outputGroupBox = QGroupBox("Results...",self)
        self.outputLayout = QGridLayout()
        self.surgeonLabel = QLabel("Surgeon Price:",self)
        self.anaesthetistLabel = QLabel("Anaesthetist Price:",self)
        self.totalLabel = QLabel("Total Price:",self)
        self.surgeonLineEdit = QLineEdit()
        self.anaesthetistLineEdit = QLineEdit()
        self.totalLineEdit = QLineEdit()
        self.outputLayout.addWidget(self.surgeonLabel,0,0)
        self.outputLayout.addWidget(self.anaesthetistLabel,1,0)
        self.outputLayout.addWidget(self.totalLabel,2,0)
        self.outputLayout.addWidget(self.surgeonLineEdit,0,1)
        self.outputLayout.addWidget(self.anaesthetistLineEdit,1,1)
        self.outputLayout.addWidget(self.totalLineEdit,2,1)
```

```
61          self.outputGroupBox.setLayout(self.outputLayout)
62
63
64          self.calculatorLayout.addWidget(self.calculateGroupBox)
65          self.calculatorLayout.addWidget(self.outputGroupBox)
66          self.calculateButton.clicked.connect(self.calculate)
67          return self.calculatorLayout
68
69      def calculate(self):
70          codes = []
71          if self.code1LineEdit.text() != '':
72              codes.append(self.code1LineEdit.text())
73          if self.code2LineEdit.text() != '':
74              codes.append(self.code2LineEdit.text())
75          if self.code3LineEdit.text() != '':
76              codes.append(self.code3LineEdit.text())
77          ic = self.icComboBox.currentText()
78          valid, errormsg = gui_validation.validate_calculator(codes,ic)
79          if valid != False:
80              try:
81                  sp,ap = code_calculator.main(codes,ic)
82                  self.surgeonLineEdit.setText(str(sp))
83                  self.anaesthetistLineEdit.setText(str(ap))
84                  self.totalLineEdit.setText(str(sp+ap))
85              except:
86                  QMessageBox.about(self, "Error", "One or more of the codes
87  are invalid.")
88          else:
89              QMessageBox.about(self, "Error", errormsg)
90
91  if __name__ == "__main__":
92      application = QApplication(sys.argv)
93      codeCalculatorWindow = codeCalculatorWindow()
94      codeCalculatorWindow.show()
95      codeCalculatorWindow.raise_()
96      application.exec_()
```

10.7. code_controller.py

```
from db_controller import *

class Code_Controller(db_controller):
    """Creates controller for Codes"""
    def __init__(self):
        super().__init__()

    def add_code(self,code,costs,invalid):
        sql = """insert into code (Code) values ('{0}')""".format(code)
        self.query(sql)
        sql = """select CodeID from code where Code = '{0}'""".format(code)
        IDtuple = self.select_query(sql)
        IDstr = IDtuple[0]
        ID = int(str(IDstr)[1:-2])
        if invalid != None:
            self.add_invalid_combination(ID,invalid)
        self.add_code_cost(ID,costs)

    def add_invalid_combination(self,ID,invalid):
        for item in invalid:
            sql = """select CodeID from code where Code =
'{0}'""".format(item)
            existing = False
            while existing != True:
                invalidIDtuple = self.select_query(sql)
                existing = True
                if invalidIDtuple == []:
                    sql2 = """insert into code
                              (Code)
                             values
                              ('{0}')""".format(item)
                    self.query(sql2)
                    sql2 = """select CodeID from code where Code =
'{0}'""".format(item)
                    newIDtuple = self.select_query(sql2)
                    newID = newIDtuple[0][0]
                    self.add_code_cost(newID,[0,0,0,0,0,0,0,0,0,0,0,0,0,0])
                    existing = False
            invalidIDstr = str(invalidIDtuple[0])
            invalidID = int(invalidIDstr[1])
            sql = """insert into invalidCombination
                      (CodeID,InvalidCode)
                     values
                      ('{0}','{1}')""".format(ID,invalidID)
            self.query(sql)

    def add_code_cost(self,ID,costs):
        for count in range(0,13,2):
            IC = (count+2)//2
            sql = """insert into codeCost

(InsuranceCompanyID,CodeID,SurgeonPrice,AnaesthetistPrice)
                     values

('{0}','{1}','{2}','{3}')""".format(IC,ID,costs[count],costs[count+1])
            self.query(sql)

    def delete_code(self,ID):
        sql = """delete from code
```

```
                    where CodeID = '{0}'""".format(ID)
            self.query(sql)

        def code_details(self,code):
            sql = """select CodeID from code where Code = '{0}'""".format(code)
            IDtuple = self.select_query(sql)
            if IDtuple != []:
                IDstr = IDtuple[0]
                ID = int(str(IDstr)[1:-2])
                invalid = self.invalid_combination_details(ID)
                costs = self.code_cost_details(ID)
                return ID,invalid, costs

        def invalid_combination_details(self,ID):
            sql = """select InvalidCode from invalidCombination where CodeID =
'{0}'""".format(ID)
            IDs = self.select_query(sql)
            invalidCodes = []
            for item in IDs:
                invalidCodes.append(item[0])
            codes = []
            for item in invalidCodes:
                sql = """select Code from code where CodeID =
'{0}'""".format(item)
                codes.append(self.select_query(sql))
            return codes

        def code_cost_details(self,ID):
            IC = [1,2,3,4,5,6,7]
            costs = []
            for count in range(7):
                sql = """select SurgeonPrice, AnaesthetistPrice from codeCost
where CodeID = '{0}' and InsuranceCompanyID = '{1}'""".format(ID,IC[count])
                cost = self.select_query(sql)
                costs.append(cost)
            return costs

        def amend_code(self,ID,code,icList,invalid):
            if code != None:
                sql = "update code set Code = '{0}' where CodeID =
'{1}'".format(code,ID)
                self.query(sql)
            if icList !=
[None,None,None,None,None,None,None,None,None,None,None,None,None,None]:
                for count in range(0,13,2):
                    IC = (count+2)//2
                    if icList[count] != None:
                        sql = "update codeCost set SurgeonPrice = '{0}' where
CodeID = '{1}' and InsuranceCompanyID = '{2}'".format(icList[count],ID,IC)
                        self.query(sql)
                    if icList[count+1] != None:
                        sql = "update codeCost set AnaesthetistPrice = '{0}'
where CodeID = '{1}' and InsuranceCompanyID =
'{2}'".format(icList[count+1],ID,IC)
                        self.query(sql)
            if invalid != None:
                sql = "delete from invalidCombination where CodeID =
'{0}'".format(ID)
                self.query(sql)
                for item in invalid:
```

```
                sql = """select CodeID from code where Code =
'{0}'""".format(item)
                existing = False
                while existing != True:
                    invalidIDtuple = self.select_query(sql)
                    existing = True
                    if invalidIDtuple == []:
                        sql2 = """insert into code
                                  (Code)
                                  values
                                  ('{0}')""".format(item)
                        self.query(sql2)
                        sql2 = """select CodeID from code where Code =
'{0}'""".format(item)
                        newIDtuple = self.select_query(sql2)
                        newID = newIDtuple[0][0]

self.add_code_cost(newID,[0,0,0,0,0,0,0,0,0,0,0,0,0,0])
                        existing = False
                invalidIDstr = str(invalidIDtuple[0])
                invalidID = int(invalidIDstr[1])
                sql = """insert into invalidCombination
                         (CodeID,InvalidCode)
                         values
                         ('{0}','{1}')""".format(ID,invalidID)
                self.query(sql)

    def get_code_costs(self,codes,IC):
        costList = []
        for item in codes:
            sql = """select CodeID from code where Code =
'{0}'""".format(item)
            IDtuple = self.select_query(sql)
            if IDtuple == []:
                return "Code- {0} does not exist.".format(item),False
            IDstr = IDtuple[0]
            ID = int(str(IDstr)[1:-2])
            sql = """select InsuranceCompanyID from insuranceCompany where
InsuranceCompanyName = '{0}'""".format(IC)
            ictuple = self.select_query(sql)
            icstr = ictuple[0]
            icID = int(str(icstr)[1:-2])
            sql = """select SurgeonPrice, AnaesthetistPrice from codeCost
where CodeID = '{0}' and InsuranceCompanyID = '{1}'""".format(ID,icID)
            cost = self.select_query(sql)
            if cost == []:
                return "Code- {0} does not have attached
costs.".format(item),False
            costList.append(cost)
        return costList,True

    def get_compatibilty(self,codes):
        incompatible = []
        for item in codes:
            sql = """select CodeID from code where Code =
'{0}'""".format(item)
            IDtuple = self.select_query(sql)
            IDstr = IDtuple[0]
            ID = int(str(IDstr)[1:-2])
            sql = """select InvalidCode from invalidCombination where
CodeID = '{0}'""".format(ID)
```

```
181                 IDtuple = self.select_query(sql)
182                 if IDtuple != []:
183                     subinvalid = []
184                     for item in IDtuple:
185                         ID = item[0]
186                         sql = """select Code from code where CodeID =
187 '{0}'""".format(ID)
188                         invalid = self.select_query(sql)
189                         subinvalid.append(invalid)
190                     incompatible.append(subinvalid)
191                 else:
192                     incompatible.append([])
193             return incompatible
194 
195 def display_code_menu():
196     print("Code Menu\n")
197     print("1) Add Code")
198     print("2) Delete Code")
199     print("3) Code Details")
200     print("4) Amend Code")
201     print("5) Calculator")
202     print("0) Exit\n")
203 
204 def get_valid_menu_option():
205     valid = False
206     while valid == False:
207         try:
208             choice = int(input("Menu Option: "))
209             if choice in [0,1,2,3,4,5]:
210                 valid = True
211             else:
212                 print("Error - Invalid")
213                 valid = False
214         except:
215             print("Error - Invalid")
216             valid = False
217     return choice
218 
219 def add_code(controller):
220     code = input("Code Name:")
221     invalid = input("Incompatible Codes: ")
222     if invalid != None:
223         letterlist = []
224         for count in range(len(invalid)):
225             if invalid[count] != ',':
226                 letterlist.append(invalid[count])
227         invalid = []
228         for count in range(0,len(letterlist),5):
229             currentcode =
230 letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
231 3]+letterlist[count+4]
232             invalid.append(currentcode)
233     BUPA_SP = int(input("BUPA Surgeon Price: £"))
234     BUPA_AP = int(input("BUPA Anaesthetist Price: £"))
235     WPA_SP = int(input("WPA Surgeon Price: £"))
236     WPA_AP = int(input("WPA Anaesthetist Price: £"))
237     PPP_SP = int(input("PPP Surgeon Price: £"))
238     PPP_AP = int(input("PPP Anaesthetist Price: £"))
239     AVIVA_SP = int(input("AVIVA Surgeon Price: £"))
240     AVIVA_AP = int(input("AVIVA Anaesthetist Price: £"))
241     PruHealth_SP = int(input("PruHealth Surgeon Price: £"))
```

```
242         PruHealth_AP = int(input("PruHealth Anaesthetist Price: £"))
243         Cigna_SP = int(input("Cigna Surgeon Price: £"))
244         Cigna_AP = int(input("Cigna Anaesthetist Price: £"))
245         SimplyHealth_SP = int(input("SimplyHealth Surgeon Price: £"))
246         SimplyHealth_AP = int(input("SimplyHealth Anaesthetist Price: £"))
247         costs =
248     [BUPA_SP,BUPA_AP,WPA_SP,WPA_AP,PPP_SP,PPP_AP,AVIVA_SP,AVIVA_AP,PruHealth_SP
249     ,PruHealth_AP,Cigna_SP,Cigna_AP,SimplyHealth_SP,SimplyHealth_AP]
250         code,invalid,costs = validation.validate_code(code,invalid,costs)
251         controller.add_code(code,costs,invalid)
252 
253     def delete_code(controller):
254         codeID = int(input("Code ID: "))
255         controller.delete_code(codeID)
256 
257     def code_details(controller):
258         code = input("Code: ")
259         invalid, costs = controller.code_details(code)
260         if costs != [[], [], [], [], [], [], []]:
261             print("BUPA Surgeon Price: £{0}".format(costs[0][0][0]))
262             print("BUPA Anaesthetist Price: £{0}".format(costs[0][0][1]))
263             print("WPA Surgeon Price: £{0}".format(costs[1][0][0]))
264             print("WPA Anaesthetist Price: £{0}".format(costs[1][0][1]))
265             print("PPP Surgeon Price: £{0}".format(costs[2][0][0]))
266             print("PPP Anaesthetist Price: £{0}".format(costs[2][0][1]))
267             print("AVIVA Surgeon Price: £{0}".format(costs[3][0][0]))
268             print("AVIVA Anaesthetist Price: £{0}".format(costs[3][0][1]))
269             print("PruHealth Surgeon Price: £{0}".format(costs[4][0][0]))
270             print("PruHealth Anaesthetist Price: £{0}".format(costs[4][0][1]))
271             print("Cigna Surgeon Price: £{0}".format(costs[5][0][0]))
272             print("Cigna Anaesthetist Price: £{0}".format(costs[5][0][1]))
273             print("SimplyHealth Surgeon Price: £{0}".format(costs[6][0][0]))
274             print("SimplyHealth Anaesthetist Price:
275     £{0}".format(costs[6][0][1]))
276         else:
277             print("No costs.")
278         if invalid != []:
279             invalidCodes = []
280             for item in invalid:
281                 invalidCodes.append(item[0][0])
282             print("Invalid Codes: {0}".format(invalidCodes))
283         else:
284             print("No invalid codes.")
285 
286     def ammend_code(controller):
287         ID = input("Code ID: ")
288         print("Update-")
289         code = input("Code: ")
290         if code == "":
291             code = None
292         BUPA_SP = input("BUPA Surgeon Price: £")
293         if BUPA_SP == "":
294             BUPA_SP = None
295         else:
296             BUPA_SP = int(BUPA_SP)
297         BUPA_AP = input("BUPA Anaesthetist Price: £")
298         if BUPA_AP == "":
299             BUPA_AP = None
300         else:
301             BUPA_AP = int(BUPA_AP)
302         WPA_SP = input("WPA Surgeon Price: £")
```

```
303         if WPA_SP == "":
304             WPA_SP = None
305         else:
306             WPA_SP = int(WPA_SP)
307         WPA_AP = input("WPA Anaesthetist Price: £")
308         if WPA_AP == "":
309             WPA_AP = None
310         else:
311             WPA_AP = int(WPA_AP)
312         PPP_SP = input("PPP Surgeon Price: £")
313         if PPP_SP == "":
314             PPP_SP = None
315         else:
316             PPP_SP = int(PPP_SP)
317         PPP_AP = input("PPP Anaesthetist Price: £")
318         if PPP_AP == "":
319             PPP_AP = None
320         else:
321             PPP_AP = int(PPP_AP)
322         AVIVA_SP = input("AVIVA Surgeon Price: £")
323         if AVIVA_SP == "":
324             AVIVA_SP = None
325         else:
326             AVIVA_SP = int(AVIVA_SP)
327         AVIVA_AP = input("AVIVA Anaesthetist Price: £")
328         if AVIVA_AP == "":
329             AVIVA_AP = None
330         else:
331             AVIVA_AP = int(AVIVA_AP)
332         PruHealth_SP = input("PruHealth Surgeon Price: £")
333         if PruHealth_SP == "":
334             PruHealth_SP = None
335         else:
336             PruHealth_SP = int(PruHealth_SP)
337         PruHealth_AP = input("PruHealth Anaesthetist Price: £")
338         if PruHealth_AP == "":
339             PruHealth_AP = None
340         else:
341             PruHealth_AP = int(PruHealth_AP)
342         Cigna_SP = input("Cigna Surgeon Price: £")
343         if Cigna_SP == "":
344             Cigna_SP = None
345         else:
346             Cigna_SP = int(Cigna_SP)
347         Cigna_AP = input("Cigna Anaesthetist Price: £")
348         if Cigna_AP == "":
349             Cigna_AP = None
350         else:
351             Cigna_AP = int(Cigna_AP)
352         SimplyHealth_SP = input("SimplyHealth Surgeon Price: £")
353         if SimplyHealth_SP == "":
354             SimplyHealth_SP = None
355         else:
356             SimplyHealth_SP = int(SimplyHealth_SP)
357         SimplyHealth_AP = input("SimplyHealth Anaesthetist Price: £")
358         if SimplyHealth_AP == "":
359             SimplyHealth_AP = None
360         else:
361             SimplyHealth_AP = int(SimplyHealth_AP)
362         invalid = input("Invalid Codes: ")
363         if invalid == "":
```

```
            invalid = None
        else:
            letterlist = []
            for count in range(len(invalid)):
                if invalid[count] != ',':
                    letterlist.append(invalid[count])
            invalid = []
            for count in range(0,len(letterlist),5):
                currentcode =
letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
3]+letterlist[count+4]
                invalid.append(currentcode)

controller.amend_code(ID,code,BUPA_SP,BUPA_AP,WPA_SP,WPA_AP,PPP_SP,PPP_AP,A
VIVA_SP,AVIVA_AP,PruHealth_SP,PruHealth_AP,Cigna_SP,Cigna_AP,SimplyHealth_S
P,SimplyHealth_AP,invalid)

def main():
    codeController = Code_Controller()
    choice = -1
    while choice != 0:
        display_code_menu()
        choice = get_valid_menu_option()
        if choice == 1:
            add_code(codeController)
        elif choice == 2:
            delete_code(codeController)
        elif choice == 3:
            code_details(codeController)
        elif choice == 4:
            ammend_code(codeController)
        elif choice == 5:
            code_calculator(codeController)
        else:
            print("Exit.")

if __name__ == "__main__":
    main()
```

10.8. code_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import code_calculator_gui
import add_code_gui
import output_code_gui
import code_controller
import gui_validation

class codeWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.setLayout(self.codeGrid())
        return self

    def codeGrid(self):
        self.codeLayout = QGridLayout()
        self.codeImage = QLabel()
        self.codeImage.setPixmap(QPixmap("images/Codes.png"))
        self.codeImage.setFixedWidth(self.codeImage.sizeHint().width())
        self.codeImage.setFixedHeight(self.codeImage.sizeHint().height())
        self.addButton = QPushButton()
        self.addButton.setStyleSheet("background-image:
url(images/Add.jpg);")
        self.addButton.setMinimumHeight(204)
        self.addButton.setMinimumWidth(204)
        self.addLabel = QLabel("Add Code",self)
        self.findButton = QPushButton()
        self.findButton.setStyleSheet("background-image:
url(images/Calculator.png);")
        self.findButton.setMinimumHeight(204)
        self.findButton.setMinimumWidth(204)
        self.findLabel = QLabel("Code Calculator",self)

        self.findGroupBox = QGroupBox('Find Code...',self)
        self.searchLayout = QVBoxLayout()
        self.searchLineEdit = QLineEdit()
        self.searchButton = QPushButton('Search')
        self.searchLayout.addWidget(self.searchLineEdit)
        self.searchLayout.addWidget(self.searchButton)
        self.findGroupBox.setLayout(self.searchLayout)
        self.codeLayout.addWidget(self.addButton,0,0)
        self.codeLayout.addWidget(self.addLabel,1,0)
        self.codeLayout.addWidget(self.codeImage,0,1)
        self.codeLayout.addWidget(self.findButton,0,2)
        self.codeLayout.addWidget(self.findLabel,1,2)
        self.codeLayout.addWidget(self.findGroupBox,1,1)
        self.addButton.clicked.connect(self.AddTest)
        self.findButton.clicked.connect(self.FindTest)
        self.searchButton.clicked.connect(self.search)
        return self.codeLayout

    def search(self):
        valid, errormsg =
gui_validation.validate_code_search(self.searchLineEdit.text())
        if valid == True:
```

```
60              existing = self.existingSearch(self.searchLineEdit.text())
61              if existing == True:
62                  self.outputWindow =
63  output_code_gui.outputCodeWindow(self.searchLineEdit.text())
64                  self.outputWindow.show()
65                  self.outputWindow.raise_()
66          else:
67              QMessageBox.about(self, "Error", errormsg)
68
69      def existingSearch(self,code):
70          self.tempController = code_controller.Code_Controller()
71          data = self.tempController.code_details(code)
72          if data == None:
73              QMessageBox.about(self, "Error", "Error- Code not found.")
74              return False
75          else:
76              return True
77
78      def AddTest(self):
79          self.addWindow = add_code_gui.addCodeWindow()
80          self.addWindow.show()
81          self.addWindow.raise_()
82
83      def FindTest(self):
84          self.calculatorWindow = code_calculator_gui.codeCalculatorWindow()
85          self.calculatorWindow.show()
86          self.calculatorWindow.raise_()
87
88  if __name__ == "__main__":
89      application = QApplication(sys.argv)
90      codeWindow = codeWindow()
91      codeWindow.show()
92      codeWindow.raise_()
93      application.exec_()
```

10.9. consultant_controller.py

```
from db_controller import *

class Consultant_Controller(db_controller):
    """Creates controller for Consultants"""
    def __init__(self):
        super().__init__()

    def add_consultant(self,title,fname,lname,email):
        sql = """insert into consultant

(ConsultantTitle,ConsultantFirstName,ConsultantLastName,ConsultantEmail)
                values

('{0}','{1}','{2}','{3}')""".format(title,fname,lname,email)
        self.query(sql)

    def delete_consultant(self,ID):
        sql = """delete from consultant
                where ConsultantID = '{0}'""".format(ID)
        self.query(sql)

    def consultant_details(self,ID=None,fname=None,lname=None):
        sql = None
        if ID != None:
            sql = """select * from consultant where ConsultantID =
'{0}'""".format(ID)
        if fname != None:
            sql = """select * from consultant where ConsultantFirstName =
'{0}'""".format(fname)
        if lname != None:
           sql = """select * from consultant where ConsultantLastName =
'{0}'""".format(lname)
        elif fname != None and lname != None:
            sql = """select * from consultant where ConsultantFirstName =
'{0}' and ConsultantLastName = '{1}'""".format(fname,lname)
        if sql != None:
            return self.select_query(sql)
        else:
            return None

    def all_details(self):
        sql = "select * from consultant"
        return self.select_query(sql)

    def amend_consultant(self,ID,title,fname,lname,email):
        updates = []
        if title != None:
            updates.append(("ConsultantTitle",title))
        if fname != None:
            updates.append(("ConsultantFirstName",fname))
        if lname != None:
            updates.append(("ConsultantLastName",lname))
        if email != None:
            updates.append(("ConsultantEmail",email))
        sql = "update consultant set "
        for item in updates:
            sql += "{0}='{1}', ".format(item[0],item[1])
        sql = sql[:-2]
```

```
59          sql += " where ConsultantID='{0}'".format(ID)
60          self.query(sql)
61
62  def display_consultant_menu():
63      print("Consultant Menu\n")
64      print("1) Add Consultant")
65      print("2) Delete Consultant")
66      print("3) Consultant Details")
67      print("4) Amend Consultant")
68      print("0) Exit\n")
69
70  def get_valid_menu_option():
71      valid = False
72      while valid == False:
73          try:
74              choice = int(input("Menu Option: "))
75              if choice in [0,1,2,3,4]:
76                  valid = True
77              else:
78                  print("Error - Invalid")
79                  valid = False
80          except:
81              print("Error - Invalid")
82              valid = False
83      return choice
84
85  def add_consultant(controller):
86      title = input("Title: ")
87      fname = input("First Name: ")
88      lname = input("Last Name: ")
89      email = input("Email: ")
90      title,fname,lname,email =
91  validation.validate_consultant(title,fname,lname,email)
92      controller.add_consultant(title,fname,lname,email)
93
94  def delete_consultant(controller):
95      ID = input("Consultant ID: ")
96      controller.delete_consultant(ID)
97
98  def consultant_details(controller):
99      print("Search By-")
100     ID = input("ID: ")
101     if ID == "":
102         ID = None
103     fname = input("First Name: ")
104     if fname == "":
105         fname = None
106     lname = input("Last Name: ")
107     if lname == "":
108         lname = None
109     details = controller.consultant_details(ID=ID,fname=fname,lname=lname)
110     print(details)
111
112 def amend_consultant(controller):
113     ID = input("Consultant ID: ")
114     print("Update-")
115     title = input("Title: ")
116     if title == "":
117         title = None
118     fname = input("First Name: ")
119     if fname == "":
```

```
120             fname = None
121         lname = input("Last Name: ")
122         if lname == "":
123             lname = None
124         email = input("Email: ")
125         if email == "":
126             email = None
127         controller.amend_consultant(ID,title,fname,lname,email)
128 
129 
130     def main():
131         consultantController = Consultant_Controller()
132         choice = -1
133         while choice != 0:
134             display_consultant_menu()
135             choice = get_valid_menu_option()
136             if choice == 1:
137                 add_consultant(consultantController)
138             elif choice == 2:
139                 delete_consultant(consultantController)
140             elif choice == 3:
141                 consultant_details(consultantController)
142             elif choice == 4:
143                 amend_consultant(consultantController)
144             else:
145                 print("Exit.")
146 
147     if __name__ == "__main__":
148         main()
```

10.10. consultant_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import add_consultant_gui
import search_consultant_gui

class consultantWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.setLayout(self.consultantGrid())
        return self

    def consultantGrid(self):
        self.consultantLayout = QGridLayout()
        self.consultantImage = QLabel()
        self.consultantImage.setPixmap(QPixmap("images/Consultants.jpg"))

self.consultantImage.setFixedWidth(self.consultantImage.sizeHint().width())

self.consultantImage.setFixedHeight(self.consultantImage.sizeHint().height(
))
        self.addButton = QPushButton()
        self.addButton.setStyleSheet("background-image:
url(images/Add.jpg);")
        self.addButton.setMinimumHeight(204)
        self.addButton.setMinimumWidth(204)
        self.addLabel = QLabel("Add Consultant",self)
        self.findButton = QPushButton()
        self.findButton.setStyleSheet("background-image:
url(images/Search.jpg);")
        self.findButton.setMinimumHeight(204)
        self.findButton.setMinimumWidth(204)
        self.findLabel = QLabel("Find Consultant",self)
        self.consultantLayout.addWidget(self.addButton,0,0)
        self.consultantLayout.addWidget(self.addLabel,1,0)
        self.consultantLayout.addWidget(self.consultantImage,0,1)
        self.consultantLayout.addWidget(self.findButton,0,2)
        self.consultantLayout.addWidget(self.findLabel,1,2)
        self.addButton.clicked.connect(self.AddTest)
        self.findButton.clicked.connect(self.FindTest)
        return self.consultantLayout

    def AddTest(self):
        self.addWindow = add_consultant_gui.addConsultantWindow()
        self.addWindow.show()
        self.addWindow.raise_()

    def FindTest(self):
        self.searchWindow = search_consultant_gui.searchConsultantWindow()
        self.searchWindow.show()
        self.searchWindow.raise_()

if __name__ == "__main__":
    application = QApplication(sys.argv)
    conWindow = consultantWindow()
    conWindow.show()
```

```
60          conWindow.raise_()
61          application.exec_()
```

10.11. database_creation.py

```
import sqlite3

def create_invoice_entity(db,cursor):
    sql = """create table invoice (
             InvoiceID integer,
             InvoiceDate date,
             PatientID integer,
             primary key (InvoiceID),
             foreign key (PatientID)
             references patient(PatientID))"""
    cursor.execute(sql)
    db.commit()

def create_insuranceCompany_entity(db,cursor):
    sql = """create table insuranceCompany (
             InsuranceCompanyID integer,
             InsuranceCompanyName integer,
             primary key (InsuranceCompanyID))"""
    cursor.execute(sql)
    db.commit()

def create_patient_entity(db,cursor):
    sql = """create table patient (
             PatientID integer,
             Title text,
             FirstName text,
             LastName text,
             AddressLine1 text,
             AddressLine2 text,
             Town text,
             PostCode text,
             DateOfBirth date,
             Gender text,
             TelephoneNumber text,
             Email text,
             DateOfOperation date,
             ConsultantID integer,
             InsuranceCompanyID integer,
             primary key (PatientID),
             foreign key (ConsultantID)
             references consultant(ConsultantID)
             foreign key (InsuranceCompanyID)
             references insuranceCompany(InsuranceCompanyID))"""
    cursor.execute(sql)
    db.commit()

def create_consultant_entity(db,cursor):
    sql = """create table consultant (
             ConsultantID integer,
             ConsultantTitle text,
             ConsultantFirstName text,
             ConsultantLastName text,
             ConsultantEmail text,
             primary key (ConsultantID))"""
    cursor.execute(sql)
    db.commit()

def create_codeCombinations_entity(db,cursor):
    sql = """create table codeCombinations (
             CodeCombinationID integer,
```

```
61                  PatientID integer,
62                  CodeID integer,
63                  primary key (CodeCombinationID)
64                  foreign key (PatientID)
65                  references patient(PatientID)
66                  foreign key (CodeID)
67                  references code(CodeID))"""
68      cursor.execute(sql)
69      db.commit()
70
71  def create_codeCost_entity(db,cursor):
72      sql = """create table codeCost (
73                  CostID integer,
74                  InsuranceCompanyID integer,
75                  CodeID integer,
76                  SurgeonPrice integer,
77                  AnaesthetistPrice integer,
78                  primary key (CostID)
79                  foreign key (InsuranceCompanyID)
80                  references insuranceCompany(InsuranceCompanyID)
81                  foreign key (CodeID)
82                  references code(CodeID)
83                  on delete cascade)"""
84      cursor.execute(sql)
85      db.commit()
86
87  def create_code_entity(db,cursor):
88      sql = """create table code (
89                  CodeID integer,
90                  Code text,
91                  primary key (CodeID))"""
92      cursor.execute(sql)
93      db.commit()
94
95  def create_user_entity(db,cursor):
96      sql = """create table user (
97                  UserID integer,
98                  Username text,
99                  Password text,
100                 UserAccess text,
101                 UserEmail text,
102                 primary key (UserID))"""
103     cursor.execute(sql)
104     db.commit()
105
106 def create_invalidCombination_entity(db,cursor):
107     sql = """create table invalidCombination (
108                 IncompatibleID integer,
109                 CodeID integer,
110                 InvalidCode integer,
111                 primary key (IncompatibleID)
112                 foreign key (CodeID)
113                 references code(CodeID)
114                 on delete cascade
115                 foreign key (InvalidCode)
116                 references code(CodeID)
117                 on delete cascade)"""
118     cursor.execute(sql)
119     db.commit()
120
121 if __name__ == "__main__":
```

```
122         db = sqlite3.connect("CMP_Database.db")
123         cursor = db.cursor()
124         create_invoice_entity(db,cursor)
125         create_insuranceCompany_entity(db,cursor)
126         create_patient_entity(db,cursor)
127         create_consultant_entity(db,cursor)
128         create_codeCombinations_entity(db,cursor)
129         create_codeCost_entity(db,cursor)
130         create_code_entity(db,cursor)
131         create_user_entity(db,cursor)
132         create_invalidCombination_entity(db,cursor)
```

10.12. db_controller.py

```
1   import sqlite3
2
3   class db_controller:
4       def __init__(self):
5           self.dbtext = "CMP_Database.db"
6
7       def query(self,sql):
8           self.db = sqlite3.connect(self.dbtext)
9           self.cursor = self.db.cursor()
10          self.cursor.execute("PRAGMA foreign_keys = ON")
11          self.cursor.execute(sql)
12          self.db.commit()
13          self.cursor.close()
14
15      def select_query(self,sql):
16          self.db = sqlite3.connect(self.dbtext)
17          self.cursor = self.db.cursor()
18          self.cursor.execute("PRAGMA foreign_keys = ON")
19          self.cursor.execute(sql)
20          results = self.cursor.fetchall()
21          self.cursor.close()
22          return results
```

10.13. gui_validation.py

```
import re

def validate_data(data,pattern):
        if re.match(pattern,data):
            return True
        else:
            return False

def
validate_patient(title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,email,D
oO,conLname,icName,codes):
    errormsg = "Error- Invalid:\n"
    validdata = True
    if title != None:
            valid = validate_data(title,"Mr|Mrs|Ms|Miss|Dr|Prof|Sir")
            if valid == False:
                validdata = False
                errormsg += "Title\n"
    if fname != None:
            valid = validate_data(fname,"^[A-Z][a-z]")
            if valid == False:
                validdata = False
                errormsg += "First Name\n"
    if lname != None:
            valid = validate_data(lname,"^[A-Z][a-z]")
            if valid == False:
                validdata = False
                errormsg += "Last Name\n"
    if ad1 != None:
            valid = validate_data(ad1,"\d+\s(^[A-Z][a-z])|((^[A-Z][a-
z])\s)+")
            if valid == False:
                validdata = False
                errormsg += "Address Line 1\n"
    if ad2 != None:
            valid = validate_data(ad2,"^[A-Z][a-z]")
            if valid == False:
                validdata = False
                errormsg += "Address Line 2\n"
    if town != None:
            valid = validate_data(town,"^[A-Z][a-z]")
            if valid == False:
                validdata = False
                errormsg += "Town\n"
    if pc != None:
            valid = validate_data(pc,"([A-Z][A-Z]\d\s\d[A-Z][A-Z])|([A-
Z][A-Z]\d\d\s\d[A-Z][A-Z])") # test
            if valid == False:
                validdata = False
                errormsg += "Post Code\n"
    if DoB != None:
            valid = validate_data(DoB,"\d{4}-\d{2}-\d{2}")
            if valid == False:
                validdata = False
                errormsg += "Date of Birth\n"
    if gender != None:
            valid = validate_data(gender,"Male|Female")
            if valid == False:
                validdata = False
                errormsg += "Gender\n"
```

```
61      if telNo != None:
62              valid = validate_data(telNo,"\d{5}(-|\s)\d{6}")
63              if valid == False:
64                  validdata = False
65                  errormsg += "Telephone Number\n"
66      if email != None:
67              valid = validate_data(email,"[a-zA-Z\d-]+@[a-zA-Z\d-
68  ]+\.(com|co\.uk)")
69              if valid == False:
70                  validdata = False
71                  errormsg += "Email\n"
72      if DoO != None:
73              valid = validate_data(DoO,"\d{4}-\d{2}-\d{2}")
74              if valid == False:
75                  validdata = False
76                  errormsg += "Date of Operation\n"
77      if conLname != None:
78              valid = validate_data(conLname,"^[A-Z][a-z]")
79              if valid == False:
80                  validdata = False
81                  errormsg += "Consultant\n"
82      if icName != None:
83              valid =
84  validate_data(icName,"BUPA|WPA|PPP|AVIVA|PruHealth|Cigna|SimplyHealth")
85              if valid == False:
86                  validdata = False
87                  errormsg += "Insurance Company\n"
88      if codes != None:
89              for item in codes:
90                  valid = validate_data(item,"[A-Z]\d{4}")
91                  if valid == False:
92                          validdata = False
93                          errormsg += "Code: {0}\n".format(item)
94      return validdata,errormsg
95
96  def validate_patient_search(search,data):
97          if search == "ID" and data != 0:
98                  valid = validate_data(data[0],"\d")
99                  if valid == False:
100                         return valid, "Error- Invalid:\nID"
101                 return True, "Error- Invalid:\n"
102         if search == "ID" and data <= 0:
103                 return False, "Error- Invalid:\nID"
104         if search == "name":
105                 errormsg = "Error- Invalid:\n"
106                 validdata = True
107                 for item in data:
108                         valid = validate_data(item,"^[A-Z][a-z]")
109                         if valid == False:
110                                 validdata = False
111                                 errormsg += "Name: {0}\n".format(item)
112                 return validdata, errormsg
113
114 def validate_consultant(title,fname,lname,email):
115     errormsg = "Error- Invalid:\n"
116     validdata = True
117     if title != None:
118             valid = validate_data(title,"Mr|Mrs|Ms|Miss|Dr|Prof|Sir")
119             if valid == False:
120                 validdata = False
121                 errormsg += "Title\n"
```

```
122         if fname != None:
123                 valid = validate_data(fname,"^[A-Z][a-z]")
124                 if valid == False:
125                     validdata = False
126                     errormsg += "First Name\n"
127         if lname != None:
128                 valid = validate_data(lname,"^[A-Z][a-z]")
129                 if valid == False:
130                     validdata = False
131                     errormsg += "Last Name\n"
132         if email != None:
133                 valid = validate_data(email,"[a-zA-Z\d-]+@[a-zA-Z\d-
134     ]+\.(com|co\.uk)")
135                 if valid == False:
136                     validdata = False
137                     errormsg += "Email\n"
138         return validdata,errormsg
139 
140     def validate_consultant_search(search,data):
141             if search == "ID" and data != 0:
142                     valid = validate_data(data[0],"\d")
143                     if valid == False:
144                             return valid, "Error- Invalid:\nID"
145                     return True, "Error- Invalid:\n"
146             if search == "ID" and data <= 0:
147                     return False, "Error- Invalid:\nID"
148             if search == "name":
149                     errormsg = "Error- Invalid:\n"
150                     validdata = True
151                     for item in data:
152                             valid = validate_data(item,"^[A-Z][a-z]")
153                             if valid == False:
154                                     validdata = False
155                                     errormsg += "Name: {0}\n".format(item)
156                     return validdata, errormsg
157 
158     def validate_user(UN,PW,AC,email):
159         errormsg = "Error- Invalid:\n"
160         validdata = True
161         if UN != None:
162                 valid = validate_data(UN,"[a-zA-Z\d]+")
163                 if valid == False:
164                     validdata = False
165                     errormsg += "Username\n"
166         if PW != None:
167                 valid = validate_data(PW,"[a-zA-Z\d]+")
168                 if valid == False:
169                     validdata = False
170                     errormsg += "Password\n"
171         if AC != None:
172                 valid = validate_data(AC,"User|Admin")
173                 if valid == False:
174                     validdata = False
175                     errormsg += "Access\n"
176         if email != None:
177                 valid = validate_data(email,"[a-zA-Z\d-]+@[a-zA-Z\d-
178     ]+\.(com|co\.uk)")
179                 if valid == False:
180                     validdata = False
181                     errormsg += "Email\n"
182         return validdata,errormsg
```

```
183
184 def validate_user_search(search,data):
185         errormsg = "Error- Invalid:\n"
186         validdata = True
187         if search == "ID" and data != 0:
188                 valid = validate_data(data,"\d")
189                 if valid == False:
190                         return valid, "Error- Invalid:\nID"
191                 return True, "Error- Invalid:\n"
192         if search == "ID" and data <= 0:
193                 return False, "Error- Invalid:\nID"
194         if search == "UN":
195                 valid = validate_data(data,"[a-zA-Z\d]+")
196                 if valid == False:
197                         return valid, "Error- Invalid:\nUsername"
198                 return True, "Error- Invalid:\n"
199
200 def validate_code(code,costs,invalid):
201         errormsg = "Error- Invalid:\n"
202         validdata = True
203         if code != None:
204                 valid = validate_data(code,"[A-Z]\d{4}")
205                 if valid == False:
206                     validdata = False
207                     errormsg += "Code Name\n"
208         if costs != []:
209                 for item in costs:
210                         if item != None:
211                             valid = validate_data(str(item),"\d")
212                             if valid == False:
213                                 validdata = False
214                                 if item != None:
215                                         errormsg += "Cost:
216 {0}\n".format(item)
217         if invalid != []:
218                 for item in invalid:
219                         valid = validate_data(item,"[A-Z]\d{4}")
220                         if valid == False:
221                                 validdata = False
222                                 errormsg += "Incompatible Code:
223 {0}\n".format(item)
224         return validdata,errormsg
225
226 def validate_code_search(data):
227         valid = validate_data(data,"[A-Z]\d{4}")
228         if valid == False:
229                 return valid, "Error- Invalid:\nCode"
230         else:
231                 return True, "Error- Invalid:\n"
232
233 def validate_operation_search(search,data):
234         errormsg = "Error- Invalid:\n"
235         validdata = True
236         if search == "con" and data != ['','']:
237                 for item in data:
238                         valid = validate_data(item,"^[A-Z][a-z]")
239                         if valid == False:
240                                 validdata = False
241                                 errormsg += "Name: {0}\n".format(item)
242                 return validdata, errormsg
243         if search == "con" and data == ['','']:
```

```
                return False, "Error- Invalid:\nName"
        if search == "date":
                valid = validate_data(data[0],"\d{4}-\d{2}-\d{2}")
                if valid == False:
                        return valid, "Error- Invalid:\nDate"
                else:
                        return True, "Error- Invalid:\n"
        if search == "both":
                valid = validate_data(data[0],"\d{4}-\d{2}-\d{2}")
                if valid == False:
                        errormsg += "Date\n"
                        validdata = False
                if data[1] != ['','']:
                        for item in data[1]:
                                valid = validate_data(item,"^[A-Z][a-z]")
                                if valid == False:
                                        validdata = False
                                        errormsg += "Name:
{0}\n".format(item)
                if data[1] == ['','']:
                        validdata = False
                        errormsg += "Name\n"
                return validdata, errormsg

def validate_calculator(codes,ic):
        errormsg = "Error- Invalid:\n"
        validdata = True
        for item in codes:
                valid = validate_data(item,"[A-Z]\d{4}")
                if valid == False:
                        validdata = False
                        errormsg += "Code: {0}\n".format(item)
        valid =
validate_data(ic,"BUPA|WPA|PPP|AVIVA|PruHealth|Cigna|SimplyHealth")
        if valid == False:
                validdata = False
                errormsg += "Insurance Company\n"
        return validdata, errormsg


if __name__ == "__main__":
    pass
```

10.14. insurance_company_controller.py

```
from db_controller import *

class Insurance_Company_Controller(db_controller):
    """Creates controller for Insurance Companies"""
    def __init__(self):
        super().__init__()

    def add_Insurance_Company(self,name):
        sql = """insert into insuranceCompany (InsuranceCompanyName) values
('{0}')""".format(name)
        self.query(sql)

    def delete_Insurance_Company(self,ID):
        sql = """delete from insuranceCompany
                 where InsuranceCompanyID = '{0}'""".format(ID)
        self.query(sql)

    def insurance_company_details(self,ID):
        sql = """select InsuranceCompanyName from insuranceCompany where
InsuranceCompanyID = '{0}'""".format(ID)
        ic = self.select_query(sql)
        return ic[0][0]


def display_insurance_company_menu():
    print("Insurance Company Menu\n")
    print("1) Add Insurance Company")
    print("2) Delete Insurance Company")
    print("3) Insurance Company Details")
    print("0) Exit\n")

def get_valid_menu_option():
    valid = False
    while valid == False:
        try:
            choice = int(input("Menu Option: "))
            if choice in [0,1,2,3]:
                valid = True
            else:
                print("Error - Invalid")
                valid = False
        except:
            print("Error - Invalid")
            valid = False
    return choice

def add_insurance_company(controller):
    IC = input("New Insurance Company: ")
    controller.add_Insurance_Company(IC)

def delete_insurance_company(controller):
    IC = input("Insurance Company ID: ")
    controller.delete_Insurance_Company(IC)

def insurance_company_details(controller):
    ID = input("Insurance ID: ")
    IC = controller.insurance_company_details(ID)
    print(IC)

```

```
60  def main():
61      insuranceCompanyController = Insurance_Company_Controller()
62      choice = -1
63      while choice != 0:
64          display_insurance_company_menu()
65          choice = get_valid_menu_option()
66          if choice == 1:
67              add_insurance_company(insuranceCompanyController)
68          elif choice == 2:
69              delete_insurance_company(insuranceCompanyController)
70          elif choice == 3:
71              insurance_company_details(insuranceCompanyController)
72          else:
73              print("Exit.")
74
75  if __name__ == "__main__":
76      main()
```

10.15. invoice_class.py

```
import datetime
import invoice_controller
import code_calculator
import webbrowser

class Invoice():
    def __init__(self,details):
        self._patientID = details[0]
        self._patientTitle = details[1]
        self._patientFirstName = details[2]
        self._patientLastName = details[3]
        self._patientAddressLine1 = details[4]
        self._patientAddressLine2 = details[5]
        self._patientTown = details [6]
        self._patientPostCode = details[7]
        self._patientCodes = None
        self._patientInsurance = details[16]
        self._opDate = details[12]
        self._consultantTitle = details[13]
        self._consultantFirstName = details[14]
        self._consultantLastName = details[15]
        self._surgeonPrice = None
        self._anaesthetistPrice = None
        self._date = None

    def get_prices(self):
        sp,ap =
code_calculator.main(self._patientCodes,self._patientInsurance)
        if sp != None:
            self._surgeonPrice,self._anaesthetistPrice = sp,ap
        else:
            return "Code Error."

    def get_date(self):
        date = datetime.datetime.now()
        strdate = str(date)
        date = strdate[0:10]
        self._date = date

    def format_codes(self,codes):
        codelist = []
        for item in codes:
            codelist.append(item[1:6])
        self._patientCodes = codelist

    def generate(self):
        codeBullets = ""
        for item in self._patientCodes:
            codeBullets += "<li>{0}</li>".format(item)
        html = """<html>
                <body>
                <p>{0} {1} {2}<br>{3}<br>{4}<br>{5}<br>{6}</p>
                <p>Dear {0} {1} {2},<br>This invoice is from {7} {8} {9}
for the following procedures which occured on {10}:</p>
                <ul>
                {15}
                </ul>
```

```
                <p>Suregon Price- £{11}<br>Anaesthetist Price-
£{12}<br>Total Price- £{13}</p>
                <p>Sent from Cambrige Medical Practice on the {14}</p>
                </body>

</html>""".format(self._patientTitle,self._patientFirstName,self._patientLa
stName,self._patientAddressLine1,self._patientAddressLine2,self._patientTow
n,self._patientPostCode,self._consultantTitle,self._consultantFirstName,sel
f._consultantLastName,self._opDate,self._surgeonPrice,self._anaesthetistPri
ce,(self._surgeonPrice+self._anaesthetistPrice),self._date,codeBullets)
        htmlfile = open("Patient_Invoice.html","w")
        htmlfile.write(html)
        htmlfile.close()

    def output(self):
        new = 2
        url = "Patient_Invoice.html"
        webbrowser.open(url,new=new)

    def store(self):
        tempController = invoice_controller.Invoice_Controller()
        tempController.add_invoice(self._date,self._patientID)

def main(details):
    invoice = Invoice(details)
    invoice.format_codes(details[17])
    error = invoice.get_prices()
    if error != None:
        print(error)
    else:
        invoice.get_date()
        invoice.generate()
        invoice.output()
        invoice.store()

if __name__ == "__main__":
    main([1, 'Mrs', 'Jack', 'Cox', '24', '234', 'Camb', 'CB1', '1994-12-
12', 'Male', '09876-283746', 'lhasdf@asda.com', '1993-12-12', 'Mr', 'God',
'Jesus', 'PPP', ["'A1234'", "'T1234'"]])
```

10.16. invoice_controller.py

```
from db_controller import *

class Invoice_Controller(db_controller):
    """Creates controller for Invoices"""
    def __init__(self):
        super().__init__()

    def add_invoice(self,date,pID):
        sql = """insert into invoice
                 (InvoiceDate,PatientID)
                values
                ('{0}','{1}')""".format(date,pID)
        self.query(sql)
```

10.17. main_gui.py

```
from db_controller import *

class Invoice_Controller(db_controller):
    """Creates controller for Invoices"""from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys

import consultant_gui, add_consultant_gui, search_consultant_gui
import user_gui, add_user_gui, search_user_gui, user_controller
import patient_gui, add_patient_gui, search_patient_gui
import code_gui, add_code_gui, code_calculator_gui
import operation_gui

class mainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()
        self.access = None

    def createUI(self):
        #give window title
        self.setWindowTitle("CMP Window")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))
        #create menubar
        self.menuBar = QMenuBar()
        self.systemMenu = self.menuBar.addMenu("System")
        self.mainMenu = self.systemMenu.addAction("Main Menu")
        self.logMenu = self.systemMenu.addMenu("Log In/Out")
        self.logInMenu = self.logMenu.addAction("Log In")
        self.logOutMenu = self.logMenu.addAction("Log Out")
        self.exitMenu = self.systemMenu.addAction("Exit")

        self.operationMenu = self.menuBar.addMenu("Operation")
        self.searchOperationMenu = self.operationMenu.addAction("Find
Operation")

        self.codeMenu = self.menuBar.addMenu("Code")
        self.mainCode = self.codeMenu.addAction("Code Main")
        self.addCodeAction = self.codeMenu.addAction("Add Code")
        self.codeCalculator = self.codeMenu.addAction("Code Calculator")

        self.userMenu = self.menuBar.addMenu("User")
        self.mainUser = self.userMenu.addAction("User Main")
        self.addUserAction = self.userMenu.addAction("Add User")
        self.findUserAction = self.userMenu.addAction("Find User")

        self.consultantMenu = self.menuBar.addMenu("Consultant")
        self.mainConsultant = self.consultantMenu.addAction("Consultant
Main")
        self.addConsultantAction = self.consultantMenu.addAction("Add
Consultant")
        self.findConsultantAction = self.consultantMenu.addAction("Find
Consultant")

        self.patientMenu = self.menuBar.addMenu("Patient")
        self.mainPatient = self.patientMenu.addAction("Patient Main")
        self.addPatientAction = self.patientMenu.addAction("Add Patient")
        self.findPatientAction = self.patientMenu.addAction("Find Patient")
        #set the menu widget
```

```
60            self.setMenuWidget(self.menuBar)
61            #conections
62            self.mainMenu.triggered.connect(self.Main)
63            self.logInMenu.triggered.connect(self.LogIn)
64            self.logOutMenu.triggered.connect(self.LogOut)
65            self.exitMenu.triggered.connect(self.Exit)
66
67            self.operationMenu.triggered.connect(self.Operation)
68
69            self.mainCode.triggered.connect(self.Code)
70            self.addCodeAction.triggered.connect(self.addCode)
71            self.codeCalculator.triggered.connect(self.Calculator)
72
73            self.mainUser.triggered.connect(self.User)
74            self.addUserAction.triggered.connect(self.addUser)
75            self.findUserAction.triggered.connect(self.findUser)
76
77            self.mainConsultant.triggered.connect(self.Consultant)
78            self.addConsultantAction.triggered.connect(self.addConsultant)
79            self.findConsultantAction.triggered.connect(self.findConsultant)
80
81            self.mainPatient.triggered.connect(self.Patient)
82            self.addPatientAction.triggered.connect(self.addPatient)
83            self.findPatientAction.triggered.connect(self.findPatient)
84            #create window components
85            self.mainViewWidget = QWidget() # creates central widget
86            #create layout
87            self.mainViewWidget.setLayout(self.startGrid())
88            #sets background colour
89            pal = self.palette()
90            role = self.backgroundRole()
91            pal.setColor(role, QColor('white'))
92            self.setPalette(pal)
93            self.setCentralWidget(self.mainViewWidget)
94
95
96        def startGrid(self):
97            self.mainLayout = QGridLayout()
98            #buttons
99            self.OperationsButton = QPushButton()
100           self.OperationsButton.setStyleSheet("background-image:
101   url(images/Operations.png);")
102           self.OperationsButton.setMinimumHeight(204)
103           self.OperationsButton.setMinimumWidth(204)
104           self.CodesButton = QPushButton()
105           self.CodesButton.setStyleSheet("background-image:
106   url(images/Codes.png);")
107           self.CodesButton.setMinimumHeight(204)
108           self.CodesButton.setMinimumWidth(204)
109           self.UsersButton = QPushButton()
110           self.UsersButton.setStyleSheet("background-image:
111   url(images/Users.png);")
112           self.UsersButton.setMinimumHeight(204)
113           self.UsersButton.setMinimumWidth(204)
114           self.ConsultantsButton = QPushButton()
115           self.ConsultantsButton.setStyleSheet("background-image:
116   url(images/Consultants.jpg);")
117           self.ConsultantsButton.setMinimumHeight(204)
118           self.ConsultantsButton.setMinimumWidth(204)
119           self.PatientsButton = QPushButton()
```

```
120             self.PatientsButton.setStyleSheet("background-image:
121     url(images/Patients.jpg);")
122             self.PatientsButton.setMinimumHeight(204)
123             self.PatientsButton.setMinimumWidth(204)
124             #labels
125             self.TitleLabel = QLabel("Cambridge Medical Practice",self)
126             self.OperationsLabel = QLabel("Operations",self)
127             self.CodesLabel = QLabel("Codes",self)
128             self.UsersLabel = QLabel("Users",self)
129             self.ConsultantsLabel = QLabel("Consultants",self)
130             self.PatientsLabel = QLabel("Patients",self)
131             #image
132             self.LogoImage = QLabel()
133             self.LogoImage.setPixmap(QPixmap("images/CMP_Logo.png"))
134             self.LogoImage.setFixedWidth(self.LogoImage.sizeHint().width())
135             self.LogoImage.setFixedHeight(self.LogoImage.sizeHint().height())
136 
137             #add widgets
138             self.mainLayout.addWidget(self.OperationsLabel,1,0)
139             self.mainLayout.addWidget(self.OperationsButton,0,0)
140 
141             self.mainLayout.addWidget(self.CodesLabel,3,0)
142             self.mainLayout.addWidget(self.CodesButton,2,0)
143 
144             self.mainLayout.addWidget(self.LogoImage,0,1)
145             self.mainLayout.addWidget(self.TitleLabel,1,1)
146 
147             self.mainLayout.addWidget(self.UsersLabel,3,1)
148             self.mainLayout.addWidget(self.UsersButton,2,1)
149 
150             self.mainLayout.addWidget(self.ConsultantsLabel,1,2)
151             self.mainLayout.addWidget(self.ConsultantsButton,0,2)
152 
153             self.mainLayout.addWidget(self.PatientsLabel,3,2)
154             self.mainLayout.addWidget(self.PatientsButton,2,2)
155 
156             #connections
157             self.OperationsButton.clicked.connect(self.Operation)
158             self.OperationsButton.setShortcut('Ctrl+1')
159             self.CodesButton.clicked.connect(self.Code)
160             self.CodesButton.setShortcut('Ctrl+2')
161             self.UsersButton.clicked.connect(self.User)
162             self.UsersButton.setShortcut('Ctrl+3')
163             self.ConsultantsButton.clicked.connect(self.Consultant)
164             self.ConsultantsButton.setShortcut('Ctrl+4')
165             self.PatientsButton.clicked.connect(self.Patient)
166             self.PatientsButton.setShortcut('Ctrl+5')
167             return self.mainLayout
168 
169         def Main(self):
170             self.createUI()
171 
172         def LogOut(self):
173             QMessageBox.about(self, "System", "You successfully logged out.")
174             self.access =  None
175 
176         def LogIn(self):
177             if self.access != "Admin" and self.access != "User":
178                 self.logWindow = QMainWindow()
179                 self.logView = QWidget()
180                 self.logView.setLayout(self.logInGrid())
```

```
181             self.logWindow.setCentralWidget(self.logView)
182             self.logWindow.setWindowTitle("Log In")
183             self.logWindow.setWindowIcon(QIcon('images/CMP_Logo.png'))
184             self.logWindow.show()
185             self.logWindow.raise_()
186         else:
187             QMessageBox.about(self, "System", "Already logged in.")
188 
189     def logInGrid(self):
190          self.logInLayout = QGridLayout()
191          self.logoImage = QLabel()
192          self.logoImage.setPixmap(QPixmap("images/CMP_Logo.png"))
193          self.logoImage.setFixedWidth(self.logoImage.sizeHint().width())
194          self.logoImage.setFixedHeight(self.logoImage.sizeHint().height())
195          self.logGroupBox = QGroupBox('Log In...',self)
196          self.inputLayout = QGridLayout()
197          self.userLabel = QLabel("User")
198          self.passLabel = QLabel("Password")
199          self.userLineEdit = QLineEdit()
200          self.passLineEdit = QLineEdit()
201          self.logButton = QPushButton("Log In")
202          self.inputLayout.addWidget(self.userLabel,0,0)
203          self.inputLayout.addWidget(self.passLabel,1,0)
204          self.inputLayout.addWidget(self.userLineEdit,0,1)
205          self.inputLayout.addWidget(self.passLineEdit,1,1)
206          self.logGroupBox.setLayout(self.inputLayout)
207          self.logInLayout.addWidget(self.logoImage)
208          self.logInLayout.addWidget(self.logGroupBox)
209          self.logInLayout.addWidget(self.logButton)
210          self.logButton.clicked.connect(self.log)
211          return self.logInLayout
212 
213     def log(self):
214         self.get_user_details(self.userLineEdit.text())
215         if self.data == []:
216             QMessageBox.about(self, "Error", "User does not exist.")
217         else:
218             if self.passLineEdit.text() == self.data[0][2]:
219                 QMessageBox.about(self, "System", "{0} Access
220 Granted.".format(self.data[0][3]))
221                 self.logWindow.close()
222                 self.access = self.data[0][3]
223             else:
224                 QMessageBox.about(self, "Error", "Incorrect Password.")
225 
226     def get_user_details(self,UN):
227         tempController = user_controller.User_Controller()
228         self.data =  tempController.user_details(UN=UN)
229 
230     def Exit(self):
231         self.close()
232 
233     def Operation(self):
234         if self.access != None:
235             self.operationLayout =
236 operation_gui.operationWindow(self.data[0][1])
237             self.setCentralWidget(self.operationLayout)
238             self.resize(self.sizeHint())
239             self.setWindowTitle("Operation")
240         else:
```

```
241             QMessageBox.about(self, "Restricted", "Please Log In to access
242 this functionality.")
243 
244     def Code(self):
245         if self.access != None:
246             self.codeLayout = code_gui.codeWindow()
247             self.setCentralWidget(self.codeLayout)
248             self.setWindowTitle("Code")
249         else:
250             QMessageBox.about(self, "Restricted", "Please Log In to access
251 this functionality.")
252 
253     def addCode(self):
254         if self.access != None:
255             self.addWindow = add_code_gui.addCodeWindow()
256             self.addWindow.show()
257             self.addWindow.raise_()
258         else:
259             QMessageBox.about(self, "Restricted", "Please Log In to access
260 this functionality.")
261 
262     def Calculator(self):
263         if self.access != None:
264             self.calculatorWindow =
265 code_calculator_gui.codeCalculatorWindow()
266             self.calculatorWindow.show()
267             self.calculatorWindow.raise_()
268         else:
269             QMessageBox.about(self, "Restricted", "Please Log In to access
270 this functionality.")
271 
272     def User(self):
273         if self.access == "Admin":
274             self.userLayout = user_gui.userWindow()
275             self.setCentralWidget(self.userLayout)
276             self.setWindowTitle("User")
277         else:
278             QMessageBox.about(self, "Restricted", "Admin access required to
279 access this functionality.")
280 
281     def addUser(self):
282         if self.access == "Admin":
283             self.addWindow = add_user_gui.addUserWindow()
284             self.addWindow.show()
285             self.addWindow.raise_()
286         else:
287             QMessageBox.about(self, "Restricted", "Admin access required to
288 access this functionality.")
289 
290     def findUser(self):
291         if self.access == "Admin":
292             self.findWindow = search_user_gui.searchUserWindow()
293             self.findWindow.show()
294             self.findWindow.raise_()
295         else:
296             QMessageBox.about(self, "Restricted", "Admin access required to
297 access this functionality.")
298 
299     def Consultant(self):
300         if self.access != None:
301             self.consultantLayout = consultant_gui.consultantWindow()
```

```
302             self.setCentralWidget(self.consultantLayout)
303             self.setWindowTitle("Consultant")
304         else:
305             QMessageBox.about(self, "Restricted", "Please Log In to access
306 this functionality.")
307 
308     def addConsultant(self):
309         if self.access != None:
310             self.addWindow = add_consultant_gui.addConsultantWindow()
311             self.addWindow.show()
312             self.addWindow.raise_()
313         else:
314             QMessageBox.about(self, "Restricted", "Please Log In to access
315 this functionality.")
316 
317     def findConsultant(self):
318         if self.access != None:
319             self.searchWindow =
320 search_consultant_gui.searchConsultantWindow()
321             self.searchWindow.show()
322             self.searchWindow.raise_()
323         else:
324             QMessageBox.about(self, "Restricted", "Please Log In to access
325 this functionality.")
326 
327     def Patient(self):
328         if self.access != None:
329             self.patientLayout = patient_gui.patientWindow()
330             self.setCentralWidget(self.patientLayout)
331             self.setWindowTitle("Patient")
332         else:
333             QMessageBox.about(self, "Restricted", "Please Log In to access
334 this functionality.")
335 
336     def addPatient(self):
337         if self.access != None:
338             self.addWindow = add_patient_gui.addPatientWindow()
339             self.addWindow.show()
340             self.addWindow.raise_()
341         else:
342             QMessageBox.about(self, "Restricted", "Please Log In to access
343 this functionality.")
344 
345     def findPatient(self):
346         if self.access != None:
347             self.findWindow = search_patient_gui.searchPatientWindow()
348             self.findWindow.show()
349             self.findWindow.raise_()
350         else:
351             QMessageBox.about(self, "Restricted", "Please Log In to access
352 this functionality.")
353 
354 if __name__ == "__main__":
355     application = QApplication(sys.argv)
356     mainWindow = mainWindow()
357     mainWindow.show()
358     mainWindow.raise_()
359     application.exec_()
```

10.18. operation_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import output_operation_gui
import consultant_controller
import gui_validation

class operationWindow(QWidget):
    def __init__(self,user):
        super().__init__()
        self.user = user
        self.createUI()

    def createUI(self):
        self.setLayout(self.operationGrid())
        return self

    def operationGrid(self):
        self.operationLayout = QVBoxLayout()
        self.operationImage = QLabel()
        self.operationImage.setPixmap(QPixmap("images/Operations.png"))

self.operationImage.setFixedWidth(self.operationImage.sizeHint().width())

self.operationImage.setFixedHeight(self.operationImage.sizeHint().height())
        self.searchGroupBox = QGroupBox('Search By...',self)
        self.searchLayout = QVBoxLayout()
        self.consultantRadioButton = QRadioButton("Consultant",self)
        self.consultantRadioButton.setChecked(1)
        self.dateRadioButton = QRadioButton("Date",self)
        self.bothRadioButton = QRadioButton("Consultant, Date",self)

        tempController = consultant_controller.Consultant_Controller()
        self.conData = tempController.all_details()
        self.conComboBox = QComboBox()
        self.conComboBox.addItem('-')
        for item in self.conData:
            con = item[1] + ' ' + item[2] + ' ' + item[3]
            self.conComboBox.addItem(con)

        self.dateLineEdit = QLineEdit("Date")
        self.searchButton = QPushButton('Search')
        self.searchLayout.addWidget(self.consultantRadioButton)
        self.searchLayout.addWidget(self.dateRadioButton)
        self.searchLayout.addWidget(self.bothRadioButton)
        self.searchLayout.addWidget(self.conComboBox)
        self.searchLayout.addWidget(self.dateLineEdit)
        self.searchLayout.addWidget(self.searchButton)
        self.searchGroupBox.setLayout(self.searchLayout)
        self.operationLayout.addWidget(self.operationImage)
        self.operationLayout.addWidget(self.searchGroupBox)
        self.searchButton.clicked.connect(self.search)
        return self.operationLayout

    def search(self):
        existing = self.setParameters()
        if existing == True:
            valid, errormsg =
gui_validation.validate_operation_search(self.search,self.data)
```

```
60              if valid == True:
61                  self.operationWindow =
62  output_operation_gui.operationWindow(self.search,self.data,self.user)
63                  self.operationWindow.show()
64                  self.operationWindow.raise_()
65                  self.close()
66              else:
67                  QMessageBox.about(self, "Error", errormsg)
68
69      def setParameters(self):
70          con = self.consultantRadioButton.isChecked()
71          date = self.dateRadioButton.isChecked()
72          both = self.bothRadioButton.isChecked()
73          if con == True:
74              self.search = "con"
75              self.data = self.compileName()
76              return True
77          elif date == True:
78              self.search = "date"
79              self.data = [self.dateLineEdit.text()]
80              return True
81          elif both == True:
82              self.search = "both"
83              name = self.compileName()
84              self.data = [self.dateLineEdit.text(),name]
85              return True
86          else:
87              QMessageBox.about(self, "Error", "Error- No search method
88  selected.")
89              return False
90
91      def compileName(self):
92          letterList = []
93          letterList1 = []
94          step = 0
95          for item in self.conComboBox.currentText():
96              if item == ' ':
97                  step += 1
98              if step == 1 and item != ' ':
99                  letterList.append(item)
100             if step == 2 and item != ' ':
101                 letterList1.append(item)
102         confname = ''
103         for item in letterList:
104             confname += item
105         conlname = ''
106         for item in letterList1:
107             conlname += item
108         return [confname,conlname]
109
110 if __name__ == "__main__":
111     application = QApplication(sys.argv)
112     opWindow = operationWindow("test")
113     opWindow.show()
114     opWindow.raise_()
115     application.exec_()
```

10.19. operations_class.py

```
import patient_controller
import consultant_controller
import code_calculator
import datetime
import webbrowser

class Operations():
    def __init__(self,search,data):
        self.patients = None
        self.consultants = None
        self.prices = None
        self.search = search
        self.data = data
        self.opData = []

    def fetch_patient_data(self):
        tempController = patient_controller.Patient_Controller()
        self.data = tempController.operation_details(self.search,self.data)

    def calculate_prices(self):
        costs = []
        for count in range (len(self.data[-1])):
            IC = self.data[-3][count]
            codes = self.data[-1][count]
            codeList = []
            for item in codes:
                code = item[1:6]
                codeList.append(code)
            sp, ap = code_calculator.main(codeList,IC)
            cost = [sp,ap]
            costs.append(cost)
            self.prices  = costs
        print(self.prices)
        return self.prices

    def format_data(self):
        self.pdata = []
        for count in range(len(self.data)-3):
            self.pdata.append(self.data[count])
        for item in self.pdata[0]:
            patient = item[0:4]
            details = []
            details.append(patient[0])
            name = patient[1] +' '+ patient[2] +' '+ patient[3]
            details.append(name)
            details.append(item[12])
            self.opData.append(details)
        step = 0
        for item in self.data[-2]:
            name = item[0] +' '+ item[1] +' '+ item[2]
            self.opData[step].append(name)
            step += 1
        print(self.opData)
        return self.opData

    def generate(self,UN):
        date = datetime.datetime.now()
        strdate = str(date)
        date = strdate[0:10]
        if self.search == "con":
```

```
61              title = "Patient Report - {0}".format(self.opData[0][3])
62          else:
63              title = "Patient Report - {0}".format(self.opData[0][2])
64          data = ""
65          for count in range(len(self.opData)):
66              currentData = "<p>{0}<br>{1} - {2}<br>Suregon Price-
67  £{3}<br>Anaesthetist Price- £{4}<br>Total Price-
68  £{5}</p>".format(self.opData[count][1],self.opData[count][3],self.opData[co
69  unt][2],self.prices[count][0],self.prices[count][1],self.prices[count][1]+s
70  elf.prices[count][0])
71              data += currentData
72          html = """<html>
73                  <body>
74                  <p>{0}<br><br>{1}
75                  <p>Produced by- {2}, Date- {3}
76                  </body>
77                  </html>""".format(title,data,UN,date)
78          htmlfile = open("Operation_Report.html","w")
79          htmlfile.write(html)
80          htmlfile.close()
81
82      def output(self):
83          new = 2
84          url = "Operation_Report.html"
85          webbrowser.open(url,new=new)
86
87  def main(s,d):
88      op = Operations(s,d)
89      op.fetch_patient_data()
90      d = op.format_data()
91      return d
92
93  def calculate(s,d):
94      op = Operations(s,d)
95      op.fetch_patient_data()
96      p = op.calculate_prices()
97      return p
98
99  def report(s,d,UN):
100     op = Operations(s,d)
101     op.fetch_patient_data()
102     op.calculate_prices()
103     op.format_data()
104     op.generate(UN)
105     op.output()
106
107 if __name__ == "__main__":
108     report("con",["Arny","Sw"],"test")
```

10.20. output_code_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import code_controller
import gui_validation

class outputCodeWindow(QMainWindow):
    def __init__(self,code):
        super().__init__()
        self.createUI(code)

    def createUI(self,code):
        self.fetchData(code)
        self.outputView = QWidget()
        self.outputView.setLayout(self.outputCodeGrid(code))
        self.setCentralWidget(self.outputView)
        self.setWindowTitle("Code - Search Result")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def fetchData(self,code):
        self.tempController = code_controller.Code_Controller()
        self.data = self.tempController.code_details(code)
        self.invalid = ""
        for item in self.data[1]:
            self.invalid += item[0][0]
            self.invalid += ","
        self.invalid = self.invalid[:-1]
        self.cost = []
        for item in self.data[2]:
            if item != []:
                currentCost = []
                currentCost.append(str(item[0][0]))
                currentCost.append(str(item[0][1]))
                self.cost.append(currentCost)
            else:
                self.cost.append("(-, -)")
        self.id = self.data[0]


    def outputCodeGrid(self,code):
        self.addCodeLayout = QGridLayout()
        self.nameLabel = QLabel("Code Name")
        self.inconpatibleLabel = QLabel("Incompatible Codes")
        self.codename = code
        self.nameLineEdit = QLineEdit(self.codename)
        self.inconpatibleLineEdit = QLineEdit(self.invalid)

        self.buttonLayout = QHBoxLayout()
        self.deleteButton = QPushButton("Delete Code")
        self.updateButton = QPushButton("Update Code")
        self.buttonLayout.addWidget(self.updateButton)
        self.buttonLayout.addWidget(self.deleteButton)

        self.priceLayout = QGridLayout()
        self.surgeonLabel = QLabel("Surgeon Price:")
        self.anaesthetistLabel = QLabel("Anaesthetist Price:")
        self.bupaApLineEdit = QLineEdit(self.cost[0][1])
        self.wpaApLineEdit = QLineEdit(self.cost[1][1])
        self.pppApLineEdit = QLineEdit(self.cost[2][1])
        self.avivaApLineEdit = QLineEdit(self.cost[3][1])
```

```
61            self.pruApLineEdit = QLineEdit(self.cost[4][1])
62            self.cignaApLineEdit = QLineEdit(self.cost[5][1])
63            self.simplyApLineEdit = QLineEdit(self.cost[6][1])
64            self.bupaSpLineEdit = QLineEdit(self.cost[0][0])
65            self.wpaSpLineEdit = QLineEdit(self.cost[1][0])
66            self.pppSpLineEdit = QLineEdit(self.cost[2][0])
67            self.avivaSpLineEdit = QLineEdit(self.cost[3][0])
68            self.pruSpLineEdit = QLineEdit(self.cost[4][0])
69            self.cignaSpLineEdit = QLineEdit(self.cost[5][0])
70            self.simplySpLineEdit = QLineEdit(self.cost[6][0])
71
72            self.priceLayout.addWidget(self.surgeonLabel,0,0)
73            self.priceLayout.addWidget(self.anaesthetistLabel,0,1)
74            self.priceLayout.addWidget(self.bupaSpLineEdit,1,0)
75            self.priceLayout.addWidget(self.wpaSpLineEdit,2,0)
76            self.priceLayout.addWidget(self.pppSpLineEdit,3,0)
77            self.priceLayout.addWidget(self.avivaSpLineEdit,4,0)
78            self.priceLayout.addWidget(self.pruSpLineEdit,5,0)
79            self.priceLayout.addWidget(self.cignaSpLineEdit,6,0)
80            self.priceLayout.addWidget(self.simplySpLineEdit,7,0)
81            self.priceLayout.addWidget(self.bupaApLineEdit,1,1)
82            self.priceLayout.addWidget(self.wpaApLineEdit,2,1)
83            self.priceLayout.addWidget(self.pppApLineEdit,3,1)
84            self.priceLayout.addWidget(self.avivaApLineEdit,4,1)
85            self.priceLayout.addWidget(self.pruApLineEdit,5,1)
86            self.priceLayout.addWidget(self.cignaApLineEdit,6,1)
87            self.priceLayout.addWidget(self.simplyApLineEdit,7,1)
88
89            self.insuranceLayout = QVBoxLayout()
90            self.insuranceLabel = QLabel("Insurance Company:",self)
91            self.bupaLabel = QLabel("BUPA",self)
92            self.wpaLabel = QLabel("WPA",self)
93            self.pppLabel = QLabel("PPP",self)
94            self.avivaLabel = QLabel("AVIVA",self)
95            self.pruLabel = QLabel("Pru Health",self)
96            self.cignaLabel = QLabel("Cigna",self)
97            self.simplyLabel = QLabel("Simply Health",self)
98
99            self.insuranceLayout.addWidget(self.insuranceLabel)
100           self.insuranceLayout.addWidget(self.bupaLabel)
101           self.insuranceLayout.addWidget(self.wpaLabel)
102           self.insuranceLayout.addWidget(self.pppLabel)
103           self.insuranceLayout.addWidget(self.avivaLabel)
104           self.insuranceLayout.addWidget(self.pruLabel)
105           self.insuranceLayout.addWidget(self.cignaLabel)
106           self.insuranceLayout.addWidget(self.simplyLabel)
107
108           self.addCodeLayout.addWidget(self.nameLabel,0,0)
109           self.addCodeLayout.addWidget(self.inconpatibleLabel,1,0)
110           self.addCodeLayout.addWidget(self.nameLineEdit,0,1)
111           self.addCodeLayout.addWidget(self.inconpatibleLineEdit,1,1)
112           self.addCodeLayout.addLayout(self.insuranceLayout,2,0)
113           self.addCodeLayout.addLayout(self.priceLayout,2,1)
114           self.addCodeLayout.addLayout(self.buttonLayout,3,1)
115
116           self.updateButton.clicked.connect(self.update)
117           self.deleteButton.clicked.connect(self.delete)
118
119           return self.addCodeLayout
120
121       def update(self):
```

```
122             tempController = code_controller.Code_Controller()
123             if self.codename != self.nameLineEdit.text():
124                 code = self.nameLineEdit.text()
125             else:
126                 code = None
127             costList =
128     [[self.bupaSpLineEdit.text(),self.bupaApLineEdit.text()],[self.wpaSpLineEdi
129     t.text(),self.wpaApLineEdit.text()],[self.pppSpLineEdit.text(),self.pppApLi
130     neEdit.text()],[self.avivaSpLineEdit.text(),self.avivaApLineEdit.text()],[s
131     elf.pruSpLineEdit.text(),self.pruApLineEdit.text()],[self.cignaApLineEdit.t
132     ext(),self.cignaApLineEdit.text()],[self.simplySpLineEdit.text(),self.simpl
133     yApLineEdit.text()]]
134             cost = []
135             for count in range(7):
136                 if costList[count][0] != self.cost[count][0]:
137                     cost.append(int(costList[count][0]))
138                 else:
139                     cost.append(None)
140                 if costList[count][1] != self.cost[count][1]:
141                     cost.append(int(costList[count][1]))
142                 else:
143                     cost.append(None)
144             valid = True
145             if self.inconpatibleLineEdit.text() != "":
146                 letterlist = []
147                 for count in range(len(self.inconpatibleLineEdit.text())):
148                     if self.inconpatibleLineEdit.text()[count] != ',':
149
150     letterlist.append(self.inconpatibleLineEdit.text()[count])
151                 invalid = []
152                 for count in range(0,len(letterlist),5):
153                     try:
154                         currentcode =
155     letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
156     3]+letterlist[count+4]
157                         invalid.append(currentcode)
158                     except:
159                         valid = False
160                         errormsg = "Error- Invalid:\nIncompatible"
161             else:
162                 invalid = None
163             if valid != False:
164                 valid, errormsg =
165     gui_validation.validate_code(code,cost,invalid)
166             if valid == False:
167                 QMessageBox.about(self, "Error", errormsg)
168             if valid != False:
169                 tempController.amend_code(self.data[0],code,cost,invalid)
170                 QMessageBox.about(self, "System", "Code successfully updated.")
171                 self.close()
172
173         def delete(self):
174             tempController = code_controller.Code_Controller()
175             try:
176                 tempController.delete_code(self.id)
177                 QMessageBox.about(self, "System", "Code successfully deleted.")
178                 self.close()
179             except:
180                 QMessageBox.about(self, "Erroor", "Code can not be deleted as
181     it is\nrequired for a Patient's record.")
182
```

```
183
184 if __name__ == "__main__":
185     application = QApplication(sys.argv)
186     outputCodeWindow = outputCodeWindow("A1234")
187     outputCodeWindow.show()
188     outputCodeWindow.raise_()
189     application.exec_()
```

10.21. output_consultant_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import consultant_controller
import gui_validation

class outputConsultantWindow(QMainWindow):
    def __init__(self,search,data):
        super().__init__()
        self.createUI(search,data)

    def createUI(self,search,data):
        self.fetchData(search,data)
        self.outputView = QWidget()
        self.outputView.setLayout(self.outputConsultantGrid())
        self.setCentralWidget(self.outputView)
        self.setWindowTitle("Consultant - Search Result")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def fetchData(self,search,data):
        tempController = consultant_controller.Consultant_Controller()
        if search == 'ID':
            self.data = tempController.consultant_details(ID=data[0])
        else:
            self.data =
tempController.consultant_details(fname=data[0],lname=data[1])

    def outputConsultantGrid(self):
        self.outputConsultantLayout = QGridLayout()
        self.titleLabel = QLabel('Title')
        self.fnameLabel = QLabel('First Name')
        self.lnameLabel = QLabel('Surname')
        self.emailLabel = QLabel('Email')
        self.titleComboBox = QComboBox()
        self.titles = ["-","Mr","Mrs","Ms","Miss","Dr","Prof","Sir"]
        for count in range(len(self.titles)):
            self.titleComboBox.addItem(self.titles[count])
            if self.titles[count] == self.data[0][1]:
                self.titleComboBox.setCurrentIndex(count)
        self.fnameLineEdit = QLineEdit(self.data[0][2])
        self.lnameLineEdit = QLineEdit(self.data[0][3])
        self.emailLineEdit = QLineEdit(self.data[0][4])
        self.updateButton = QPushButton('Update Consultant')
        self.deleteButton = QPushButton('Delete Consultant')
        self.outputConsultantLayout.addWidget(self.titleLabel,0,0)
        self.outputConsultantLayout.addWidget(self.fnameLabel,1,0)
        self.outputConsultantLayout.addWidget(self.lnameLabel,2,0)
        self.outputConsultantLayout.addWidget(self.emailLabel,3,0)
        self.outputConsultantLayout.addWidget(self.fnameLineEdit,1,1)
        self.outputConsultantLayout.addWidget(self.lnameLineEdit,2,1)
        self.outputConsultantLayout.addWidget(self.emailLineEdit,3,1)
        self.outputConsultantLayout.addWidget(self.titleComboBox,0,1)
        self.outputConsultantLayout.addWidget(self.updateButton,4,1)
        self.outputConsultantLayout.addWidget(self.deleteButton,4,0)
        self.updateButton.clicked.connect(self.update)
        self.deleteButton.clicked.connect(self.delete)
        return self.outputConsultantLayout

    def update(self):
```

```
60          if self.data[0][1] != self.titleComboBox.currentText():
61              title = self.titleComboBox.currentText()
62          else:
63              title = None
64          if self.data[0][2] != self.fnameLineEdit.text():
65              fname = self.fnameLineEdit.text()
66          else:
67              fname = None
68          if self.data[0][3] != self.lnameLineEdit.text():
69              lname = self.lnameLineEdit.text()
70          else:
71              lname = None
72          if self.data[0][4] != self.emailLineEdit.text():
73              email = self.emailLineEdit.text()
74          else:
75              email = None
76          update = True
77          if fname == None and lname == None and email == None and title ==
78  None:
79              QMessageBox.about(self, "Error", "Error- No updates detected.")
80              update = False
81          valid, self.errormsg =
82  gui_validation.validate_consultant(title,fname,lname,email)
83          if valid == False:
84              QMessageBox.about(self, "Error", self.errormsg)
85          if update != False and valid == True:
86              tempController = consultant_controller.Consultant_Controller()
87
88  tempController.amend_consultant(self.data[0][0],title,fname,lname,email)
89              QMessageBox.about(self, "System", "Consultant successfully
90  updated.")
91              self.close()
92
93      def delete(self):
94          tempController = consultant_controller.Consultant_Controller()
95          try:
96              tempController.delete_consultant(self.data[0][0])
97              QMessageBox.about(self, "System", "Consultant successfully
98  deleted.")
99              self.close()
100         except:
101             QMessageBox.about(self, "Error", "Consultant can not be
102 deleted\n as they are required for one\n or more Patient records.")
103
104 if __name__ == "__main__":
105     application = QApplication(sys.argv)
106     outputConsultantWindow = outputConsultantWindow("name",["Jack","Cox"])
107     outputConsultantWindow.show()
108     outputConsultantWindow.raise_()
109     application.exec_()
```

10.22. output_operation_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import operations_class

class operationWindow(QMainWindow):
    def __init__(self,search,data,user):
        super().__init__()
        self.user = user
        self.createUI(search,data)

    def createUI(self,search,data):
        self.search = search
        self.data = data
        self.fetchData(self.search,self.data)
        self.outputView = QWidget()
        self.outputView.setLayout(self.outputOperationGrid())
        self.setCentralWidget(self.outputView)
        self.setWindowTitle("Operation - Search Result")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def fetchData(self,search,data):
        if search == "both":
            self.pdata = operations_class.main(search,data[1])
        else:
            self.pdata = operations_class.main(search,data)

    def outputOperationGrid(self):
        self.outputLayout = QVBoxLayout()
        self.outputTable = QTableWidget()
        self.outputTable.setRowCount(len(self.pdata))
        self.outputTable.setColumnCount(6)
        self.outputTable.setHorizontalHeaderLabels(['Patient
ID','Patient','Date of Operation','Consultant','Surgeon
Price','Anaesthetist Price'])

        for row in range(len(self.pdata)):
            for col in range(len(self.pdata[row])):
                if col == 0:

self.outputTable.setItem(row,col,QTableWidgetItem(str(self.pdata[row][col])
))
                else:

self.outputTable.setItem(row,col,QTableWidgetItem(self.pdata[row][col]))

        self.buttonLayout = QHBoxLayout()
        self.calculateButton = QPushButton('Calculate Prices')
        self.reportButton = QPushButton('Print Report')
        self.buttonLayout.addWidget(self.calculateButton)
        self.buttonLayout.addWidget(self.reportButton)
        self.outputLayout.addWidget(self.outputTable)
        self.outputLayout.addLayout(self.buttonLayout)
        self.calculateButton.clicked.connect(self.calculate)
        self.reportButton.clicked.connect(self.report)
        return self.outputLayout

    def calculate(self):
        self.spapdata = operations_class.calculate(self.search,self.data)
        for row in range(len(self.spapdata)):
```

```
61
62  self.outputTable.setItem(row,4,QTableWidgetItem(str(self.spapdata[row][0]))
63  )
64
65  self.outputTable.setItem(row,5,QTableWidgetItem(str(self.spapdata[row][1]))
66  )
67
68      def report(self):
69          QMessageBox.about(self, "System", "Producing report.")
70          self.close()
71          operations_class.report(self.search,self.data,self.user)
72
73  if __name__ == "__main__":
74      application = QApplication(sys.argv)
75      operationWindow = operationWindow("con",["Arny","Sw"],"Name")
76      operationWindow.show()
77      operationWindow.raise_()
78      application.exec_()
```

10.23. output_patient_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import consultant_controller
import patient_controller
import gui_validation
import invoice_class

class outputPatientWindow(QMainWindow):
    def __init__(self,search,data):
        super().__init__()
        self.createUI(search,data)

    def createUI(self,search,data):
        self.fetchData(search,data)
        self.outputView = QWidget()
        self.outputView.setLayout(self.outputPatientGrid())
        self.setCentralWidget(self.outputView)
        self.setWindowTitle("Patient - Output Patient")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def fetchData(self,search,data):
        tempController = patient_controller.Patient_Controller()
        if search == "ID":
            self.data = tempController.patient_details(ID=data[0])
        else:
            self.data =
tempController.patient_details(fname=data[0],lname=data[1])
        self.codes = ""
        for item in self.data[17]:
            self.codes += item[1:6]
            self.codes += ","
        self.codes = self.codes[:-1]

    def outputPatientGrid(self):
        self.outputPatientLayout = QGridLayout()
        self.addPatientLayout = QGridLayout()
        self.titleLabel = QLabel("Title")
        self.fnameLabel = QLabel("First Name")
        self.lnameLabel = QLabel("Surname")
        self.ad1Label = QLabel("Address Line 1")
        self.ad2Label = QLabel("Address Line 2")
        self.townLabel = QLabel("Town")
        self.pcLabel = QLabel("Post Code")
        self.dobLabel = QLabel("Date of Birth")
        self.genderLabel = QLabel("Gender")
        self.emailLabel = QLabel("Email")
        self.telLabel = QLabel("Telephone Number")
        self.icLabel = QLabel("Insurance")
        self.codesLabel = QLabel("Codes")
        self.dooLabel = QLabel("Date of Operation")
        self.conLabel = QLabel("Consultant")

        self.titleComboBox = QComboBox()
        self.titles = ["-","Mr","Mrs","Ms","Miss","Dr","Prof","Sir"]
        for count in range(len(self.titles)):
            self.titleComboBox.addItem(self.titles[count])
            if self.titles[count] == self.data[1]:
                self.titleComboBox.setCurrentIndex(count)

```

```
61            self.genderComboBox = QComboBox()
62            self.genders = ["-","Male","Female"]
63            for count in range(len(self.genders)):
64                self.genderComboBox.addItem(self.genders[count])
65                if self.genders[count] == self.data[9]:
66                    self.genderComboBox.setCurrentIndex(count)
67
68            self.icComboBox = QComboBox()
69            self.ics = ["-
70    ","BUPA","WPA","PPP","AVIVA","PruHealth","Cigna","SimplyHealth"]
71            for count in range(len(self.ics)):
72                self.icComboBox.addItem(self.ics[count])
73                if self.ics[count] == self.data[16]:
74                    self.icComboBox.setCurrentIndex(count)
75
76            self.fnameLineEdit = QLineEdit(self.data[2])
77            self.lnameLineEdit = QLineEdit(self.data[3])
78            self.ad1LineEdit = QLineEdit(self.data[4])
79            self.ad2LineEdit = QLineEdit(self.data[5])
80            self.townLineEdit = QLineEdit(self.data[6])
81            self.pcLineEdit = QLineEdit(self.data[7])
82            self.dobLineEdit = QLineEdit(self.data[8])
83            self.emailLineEdit = QLineEdit(self.data[11])
84            self.telLineEdit = QLineEdit(self.data[10])
85            self.codesLineEdit = QLineEdit(self.codes)
86            self.dooLineEdit = QLineEdit(self.data[12])
87
88            self.updateButton = QPushButton("Update Patient")
89            self.deleteButton = QPushButton("Delete Patient")
90            self.invoiceButton = QPushButton("Invoice")
91            self.createConsultantComboBox()
92
93            self.outputPatientLayout.addWidget(self.titleLabel,0,0)
94            self.outputPatientLayout.addWidget(self.fnameLabel,1,0)
95            self.outputPatientLayout.addWidget(self.lnameLabel,2,0)
96            self.outputPatientLayout.addWidget(self.ad1Label,3,0)
97            self.outputPatientLayout.addWidget(self.ad2Label,4,0)
98            self.outputPatientLayout.addWidget(self.townLabel,5,0)
99            self.outputPatientLayout.addWidget(self.pcLabel,6,0)
100           self.outputPatientLayout.addWidget(self.dobLabel,7,0)
101           self.outputPatientLayout.addWidget(self.genderLabel,0,2)
102           self.outputPatientLayout.addWidget(self.emailLabel,1,2)
103           self.outputPatientLayout.addWidget(self.telLabel,2,2)
104           self.outputPatientLayout.addWidget(self.titleComboBox,0,1)
105           self.outputPatientLayout.addWidget(self.fnameLineEdit,1,1)
106           self.outputPatientLayout.addWidget(self.lnameLineEdit,2,1)
107           self.outputPatientLayout.addWidget(self.ad1LineEdit,3,1)
108           self.outputPatientLayout.addWidget(self.ad2LineEdit,4,1)
109           self.outputPatientLayout.addWidget(self.townLineEdit,5,1)
110           self.outputPatientLayout.addWidget(self.pcLineEdit,6,1)
111           self.outputPatientLayout.addWidget(self.dobLineEdit,7,1)
112           self.outputPatientLayout.addWidget(self.genderComboBox,0,3)
113           self.outputPatientLayout.addWidget(self.emailLineEdit,1,3)
114           self.outputPatientLayout.addWidget(self.telLineEdit,2,3)
115
116           self.outputPatientLayout.addWidget(self.icLabel,3,2)
117           self.outputPatientLayout.addWidget(self.icComboBox,3,3)
118           self.outputPatientLayout.addWidget(self.codesLabel,4,2)
119           self.outputPatientLayout.addWidget(self.dooLabel,5,2)
120           self.outputPatientLayout.addWidget(self.conLabel,6,2)
121           self.outputPatientLayout.addWidget(self.conComboBox,6,3)
```

```
122             self.outputPatientLayout.addWidget(self.codesLineEdit,4,3)
123             self.outputPatientLayout.addWidget(self.dooLineEdit,5,3)
124 
125             self.outputPatientLayout.addWidget(self.updateButton,8,0)
126             self.outputPatientLayout.addWidget(self.deleteButton,8,1)
127             self.outputPatientLayout.addWidget(self.invoiceButton,8,2)
128             self.updateButton.clicked.connect(self.update)
129             self.deleteButton.clicked.connect(self.delete)
130             self.invoiceButton.clicked.connect(self.invoice)
131             return self.outputPatientLayout
132 
133         def createConsultantComboBox(self):
134             tempController = consultant_controller.Consultant_Controller()
135             self.conData = tempController.all_details()
136             self.conComboBox = QComboBox()
137             self.conComboBox.addItem('-')
138             for count in range(len(self.conData)):
139                 con = self.conData[count][1] + ' ' + self.conData[count][2] + '
140 ' + self.conData[count][3]
141                 self.conComboBox.addItem(con)
142                 if self.conData[count][1] == self.data[13] and
143 self.conData[count][2] == self.data[14] and self.conData[count][3] ==
144 self.data[15]:
145                     self.conComboBox.setCurrentIndex(count+1)
146 
147         def invoice(self):
148             QMessageBox.about(self,"System","Generating Invoice")
149             self.close()
150             invoice_class.main(self.data)
151 
152         def update(self):
153             if self.data[1] != self.titleComboBox.currentText():
154                 title = self.titleComboBox.currentText()
155             else:
156                 title = None
157             if self.data[2] != self.fnameLineEdit.text():
158                 fname = self.fnameLineEdit.text()
159             else:
160                 fname = None
161             if self.data[3] != self.lnameLineEdit.text():
162                 lname = self.lnameLineEdit.text()
163             else:
164                 lname = None
165             if self.data[4] != self.ad1LineEdit.text():
166                 ad1 = self.ad1LineEdit.text()
167             else:
168                 ad1 = None
169             if self.data[5] != self.ad2LineEdit.text():
170                 ad2 = self.ad2LineEdit.text()
171             else:
172                 ad2 = None
173             if self.data[6] != self.townLineEdit.text():
174                 town = self.townLineEdit.text()
175             else:
176                 town = None
177             if self.data[7] != self.pcLineEdit.text():
178                 pc = self.pcLineEdit.text()
179             else:
180                 pc = None
181             if self.data[8] != self.dobLineEdit.text():
182                 DoB = self.dobLineEdit.text()
```

```
            else:
                DoB = None
            if self.data[9] != self.genderComboBox.currentText():
                gender = self.genderComboBox.currentText()
            else:
                gender = None
            if self.data[10] != self.telLineEdit.text():
                telNo = self.telLineEdit.text()
            else:
                telNo = None
            if self.data[11] != self.emailLineEdit.text():
                email = self.emailLineEdit.text()
            else:
                email = None
            if self.data[12] != self.dooLineEdit.text():
                DoO = self.dooLineEdit.text()
            else:
                DoO = None
            con = self.conComboBox.currentText()
            letterList = []
            step = 0
            for item in con:
                if item == ' ':
                    step += 1
                if step == 2 and item != ' ':
                    letterList.append(item)
            conlname = ''
            for item in letterList:
                conlname += item
            if self.data[15] != conlname:
                conLname = conlname
            else:
                conLname = None
            if self.data[16] != self.icComboBox.currentText():
                icName = self.icComboBox.currentText()
            else:
                icName = None
            valid = True
            if self.codes != self.codesLineEdit.text():
                codes = self.codesLineEdit.text()
                if codes != "":
                    letterlist = []
                    for count in range(len(codes)):
                        if codes[count] != ',':
                            letterlist.append(codes[count])
                    codes = []
                    for count in range(0,len(letterlist),5):
                        try:
                            currentcode =
letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
3]+letterlist[count+4]
                            codes.append(currentcode)
                        except:
                            valid = False
                            errormsg = "Error- Invalid:\nIncompatible"
            else:
                codes = None
            update = True
            if title == None and fname == None and lname == None and ad1 ==
None and ad2 == None and town == None and pc == None and DoB == None and
```

```
243     gender == None and telNo == None and email == None and DoO == None and
244     conLname == None and icName == None and codes == None:
245                 QMessageBox.about(self, "Error", "Error- No updates detected.")
246                 update = False
247             if valid == False:
248                 QMessageBox.about(self, "Error", errormsg)
249             if valid != False and update != False:
250
251     print(title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,email,DoO,conLname
252     ,icName,codes)
253                 valid, errormsg =
254     gui_validation.validate_patient(title,fname,lname,ad1,ad2,town,pc,DoB,gende
255     r,telNo,email,DoO,conLname,icName,codes)
256                 if valid == False:
257                     QMessageBox.about(self, "Error", errormsg)
258                 else:
259                     tempController = patient_controller.Patient_Controller()
260
261     tempController.amend_patient(self.data[0],title,fname,lname,ad1,ad2,town,pc
262     ,DoB,gender,telNo,email,DoO,conLname,icName,codes)
263                     QMessageBox.about(self, "System", "Patient successfully
264     updated.")
265                     self.close()
266
267         def delete(self):
268             tempController = patient_controller.Patient_Controller()
269             tempController.delete_patient(self.data[0])
270             QMessageBox.about(self, "System", "Patient successfully deleted.")
271             self.close()
272
273
274     if __name__ == "__main__":
275         application = QApplication(sys.argv)
276         outputPatientWindow = outputPatientWindow("ID",[4])
277         outputPatientWindow.show()
278         outputPatientWindow.raise_()
279         application.exec_()
```

10.24. output_user_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import user_controller
import gui_validation

class outputUserWindow(QWidget):
    def __init__(self,search,data):
        super().__init__()
        self.createUI(search,data)

    def createUI(self,search,data):
        self.fetchData(search,data)
        self.setLayout(self.outputCodeGrid())
        self.setWindowTitle("Patient - Output Patient")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))
        return self

    def fetchData(self,search,data):
        tempController = user_controller.User_Controller()
        if search == 'ID':
            self.data = tempController.user_details(ID=data)
        else:
            self.data = tempController.user_details(UN=data)

    def outputCodeGrid(self):
        self.outputUserLayout = QGridLayout()
        self.nameLabel = QLabel('Username')
        self.passLabel = QLabel('Password')
        self.emailLabel = QLabel('Email')
        self.accessLabel = QLabel('Access')
        self.nameLineEdit = QLineEdit(self.data[0][1])
        self.passLineEdit = QLineEdit(self.data[0][2])
        self.emailLineEdit = QLineEdit(self.data[0][4])

        self.accessComboBox = QComboBox()
        self.access = ["-","User","Admin"]
        for count in range(len(self.access)):
            self.accessComboBox.addItem(self.access[count])
            if self.access[count] == self.data[0][3]:
                self.accessComboBox.setCurrentIndex(count)

        self.updateButton = QPushButton('Update User')
        self.deleteButton = QPushButton('Delete User')
        self.outputUserLayout.addWidget(self.nameLabel,0,0)
        self.outputUserLayout.addWidget(self.passLabel,1,0)
        self.outputUserLayout.addWidget(self.emailLabel,2,0)
        self.outputUserLayout.addWidget(self.accessLabel,3,0)
        self.outputUserLayout.addWidget(self.nameLineEdit,0,1)
        self.outputUserLayout.addWidget(self.passLineEdit,1,1)
        self.outputUserLayout.addWidget(self.emailLineEdit,2,1)
        self.outputUserLayout.addWidget(self.accessComboBox,3,1)
        self.outputUserLayout.addWidget(self.updateButton,4,1)
        self.outputUserLayout.addWidget(self.deleteButton,4,0)
        self.deleteButton.clicked.connect(self.delete)
        self.updateButton.clicked.connect(self.update)
        return self.outputUserLayout

    def update(self):
```

```
60            update = True
61            if self.data[0][1] != self.nameLineEdit.text():
62                UN = self.nameLineEdit.text()
63            else:
64                UN = None
65            if self.data[0][2] != self.passLineEdit.text():
66                PW = self.passLineEdit.text()
67            else:
68                PW = None
69            if self.data[0][4] != self.emailLineEdit.text():
70                email = self.emailLineEdit.text()
71            else:
72                email = None
73            if self.data[0][3] != self.accessComboBox.currentText():
74                AC = self.accessComboBox.currentText()
75            else:
76                AC = None
77            if UN == None and PW == None and email == None and AC == None:
78                QMessageBox.about(self, "Error", "Error- No updates detected.")
79                update = False
80            valid, self.errormsg = gui_validation.validate_user(UN,PW,AC,email)
81            if valid == False:
82                QMessageBox.about(self, "Error", self.errormsg)
83            if valid == True and update != False:
84                tempController = user_controller.User_Controller()
85                tempController.amend_user(self.data[0][0],UN,PW,AC,email)
86                QMessageBox.about(self, "System", "User successfully updated.")
87                self.close()
88
89        def delete(self):
90            if self.data[0][1] != "SystemAdmin":
91                tempController = user_controller.User_Controller()
92                tempController.delete_user(self.data[0][0])
93                QMessageBox.about(self, "System", "User successfully deleted.")
94                self.close()
95            else:
96                QMessageBox.about(self, "System", "The system admin\ncan not be
97    deleted.")
98
99    if __name__ == "__main__":
100       application = QApplication(sys.argv)
101       outputUserWindow = outputUserWindow("UN","SystemAdmin")
102       outputUserWindow.show()
103       outputUserWindow.raise_()
104       application.exec_()
```

10.25. patient_controller.py

```
from db_controller import *
import consultant_controller
import insurance_company_controller

class Patient_Controller(db_controller):
    """Creates controller for Patients"""
    def __init__(self):
        super().__init__()

    def
add_patient(self,title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,email,D
oO,conLname,icName,codes):
        sql = """select ConsultantID from consultant where
ConsultantLastName = '{0}'""".format(conLname)
        conIDtuple = self.select_query(sql)
        if conIDtuple == []:
            return "Consultant Does not exist."
        conIDstr = conIDtuple[0]
        conID = int(str(conIDstr)[1:-2])
        sql = """select InsuranceCompanyID from insuranceCompany where
InsuranceCompanyName = '{0}'""".format(icName)
        icIDtuple = self.select_query(sql)
        if icIDtuple == []:
            return "Insurance Company Does not exist."
        icIDstr = icIDtuple[0]
        icID = int(str(icIDstr)[1:-2])
        sql = """insert into patient

(Title,FirstName,LastName,AddressLine1,AddressLine2,Town,PostCode,DateOfBir
th,Gender,TelephoneNumber,Email,DateOfOperation,ConsultantID,InsuranceCompa
nyID)
                values

('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}',
'{12}','{13}')""".format(title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo
,email,DoO,conID,icID)
        self.query(sql)
        sql = """select PatientID from patient where LastName = '{0}' and
TelephoneNumber = '{1}'""".format(lname,telNo)
        pIDtuple = self.select_query(sql)
        pIDstr = pIDtuple[0]
        pID = int(str(pIDstr)[1:-2])
        success = self.create_code_combination(pID,codes)
        return success

    def create_code_combination(self,patientID,codes):
        for item in codes:
            sql = """select CodeID from code where Code =
'{0}'""".format(item)
            codeIDtuple = self.select_query(sql)
            if codeIDtuple == []:
                return "Code- {0} Does not exist.".format(item)
            codeIDstr = codeIDtuple[0]
            codeID = int(str(codeIDstr)[1:-2])
            sql = """insert into codeCombinations
                    (PatientID,CodeID)
                    values
                    ('{0}','{1}')""".format(patientID,codeID)
            self.query(sql)
```

```
60          return "Success"
61
62      def delete_patient(self,ID):
63          self.remove_code_combination(ID)
64          sql = """delete from patient
65                   where PatientID = '{0}'""".format(ID)
66          self.query(sql)
67
68      def remove_code_combination(self,ID):
69          sql = """delete from codeCombinations
70                   where PatientID = '{0}'""".format(ID)
71          self.query(sql)
72
73      def
74  patient_details(self,ID=None,fname=None,lname=None,conid=None,date=None):
75          sql = None
76          if ID != None:
77              sql = """select * from patient where PatientID =
78  '{0}'""".format(ID)
79              codes = self.code_details(ID=ID)
80          elif fname != None and lname != None:
81              sql = """select * from patient where FirstName = '{0}' and
82  LastName = '{1}'""".format(fname,lname)
83              codes = self.code_details(fname=fname,lname=lname)
84          elif fname != None:
85              sql = """select * from patient where FirstName =
86  '{0}'""".format(fname)
87              codes = self.code_details(fname=fname)
88          elif lname != None:
89              sql = """select * from patient where LastName =
90  '{0}'""".format(lname)
91              codes = self.code_details(lname=lname)
92          elif conname != None:
93              sql = """select * from patient where Consultant =
94  '{0}'""".format(conid)
95          elif date != None:
96              sql = """select * from patient where DateOfOperation =
97  '{0}'""".format(date)
98          if sql != None:
99              details = self.select_query(sql)
100             pdetails = details[0][0:13]
101             condetails = self.consultant_details(details[0])
102             icdetails = self.insurance_details(details[0])
103             details = []
104             for item in pdetails:
105                 details.append(item)
106             for item in condetails:
107                 details.append(item)
108             details.append(icdetails)
109             details.append(codes)
110             return details
111         else:
112             return None
113
114     def existing(self,ID=None,fname=None,lname=None):
115         if ID != None:
116             sql = """select * from patient where PatientID =
117 '{0}'""".format(ID)
118         else:
119             sql = """select * from patient where FirstName = '{0}' and
120 LastName = '{1}'""".format(fname,lname)
```

```
121             details = self.select_query(sql)
122             return details
123 
124         def code_details(self,ID=None,fname=None,lname=None):
125             sql = None
126             if fname != None and lname != None:
127                 sql = """select PatientID from  patient where FirstName = '{0}'
128     and LastName = '{1}'""".format(fname,lname)
129                 IDtuple = self.select_query(sql)
130                 IDstr = IDtuple[0]
131                 ID = int(str(IDstr)[1:-2])
132             elif fname != None:
133                 sql = """select PatientID from patient where FirstName =
134     '{0}'""".format(fname)
135                 IDtuple = self.select_query(sql)
136                 IDstr = IDtuple[0]
137                 ID = int(str(IDstr)[1:-2])
138             elif lname != None:
139                 sql = """select PatientID from patient where LastName =
140     '{0}'""".format(lname)
141                 IDtuple = self.select_query(sql)
142                 IDstr = IDtuple[0]
143                 ID = int(str(IDstr)[1:-2])
144             if ID != None:
145                 sql = """select CodeID from codeCombinations where PatientID =
146     '{0}'""".format(ID)
147                 codes = self.select_query(sql)
148                 codeList = []
149                 for item in codes:
150                     ID = int(str(item)[1:-2])
151                     sql = """select Code from code where CodeID =
152     {0}""".format(ID)
153                     codetuple = self.select_query(sql)
154                     codestr = codetuple[0]
155                     codeList.append(str(codestr)[1:-2])
156                 return codeList
157 
158         def consultant_details(self,details):
159             tempController = consultant_controller.Consultant_Controller()
160             con = tempController.consultant_details(ID=details[13])
161             con = con[0]
162             con = con[1:4]
163             return con
164 
165         def insurance_details(self,details):
166             tempController =
167     insurance_company_controller.Insurance_Company_Controller()
168             ic = tempController.insurance_company_details(ID=details[14])
169             return ic
170 
171         def operation_details(self,search,data):
172             tempController = consultant_controller.Consultant_Controller()
173             if search == "con":
174                 con =
175     tempController.consultant_details(fname=data[0],lname=data[1])
176                 conID = con[0][0]
177                 sql = "select * from patient where ConsultantID =
178     '{0}'".format(conID)
179             elif search == "date":
180                 sql = "select * from patient where DateOfOperation =
181     '{0}'".format(data[0])
```

```
182             elif search == "both":
183                 con =
184     tempController.consultant_details(fname=data[0],lname=data[1])
185                 conID = con[0][0]
186                 sql = "select * from patient where DateOfOperation = '{0}' and
187     ConsultantID = '{1}'".format(data[0],conID)
188             pdata = self.select_query(sql)
189             icdata = []
190             condata = []
191             for item in pdata:
192                 tempController =
193     insurance_company_controller.Insurance_Company_Controller()
194                 ic = tempController.insurance_company_details(item[14])
195                 con = self.consultant_details(item)
196                 icdata.append(ic)
197                 condata.append(con)
198             codes = []
199             for item in pdata:
200                 code = self.code_details(ID=item[0])
201                 codes.append(code)
202             data = []
203             data.append(pdata)
204             data.append(icdata)
205             data.append(condata)
206             data.append(codes)
207             return data
208
209         def
210     amend_patient(self,ID,title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,em
211     ail,DoO,conLname,icName,codes):
212             updates = []
213             if title != None:
214                 updates.append(("Title",title))
215             if fname != None:
216                 updates.append(("FirstName",fname))
217             if lname != None:
218                 updates.append(("LastName",lname))
219             if ad1 != None:
220                 updates.append(("AddressLine1",ad1))
221             if ad2 != None:
222                 updates.append(("AddressLine2",ad2))
223             if town != None:
224                 updates.append(("Town",town))
225             if pc != None:
226                 updates.append(("PostCode",pc))
227             if DoB != None:
228                 updates.append(("DateOfBirth",DoB))
229             if gender != None:
230                 updates.append(("Gender",gender))
231             if telNo != None:
232                 updates.append(("TelephoneNumber",telNo))
233             if email != None:
234                 updates.append(("Email",email))
235             if DoO != None:
236                 updates.append(("DateOfOperation",DoO))
237             if updates != []:
238                 sql = "update patient set "
239                 for item in updates:
240                     sql += "'{0}'='{1}', ".format(item[0],item[1])
241                 sql = sql[:-2]
242                 sql += " where PatientID='{0}'".format(ID)
```

```
243                 self.query(sql)
244             if conLname != None:
245                 sql = "select ConsultantID from consultant where
246     ConsultantLastName = '{0}'".format(conLname)
247                 conIDtuple = self.select_query(sql)
248                 conIDstr = conIDtuple[0]
249                 conID = int(str(conIDstr)[1:-2])
250                 sql = "update patient set ConsultantID='{0}' where
251     PatientID='{1}'".format(conID,ID)
252                 self.query(sql)
253             if icName != None:
254                 sql = "select InsuranceCompanyID from insuranceCompany where
255     InsuranceCompanyName = '{0}'".format(icName)
256                 icIDtuple = self.select_query(sql)
257                 icIDstr = icIDtuple[0]
258                 icID = int(str(icIDstr)[1:-2])
259                 sql = "update patient set InsuranceCompanyID='{0}' where
260     PatientID='{1}'".format(icID,ID)
261                 self.query(sql)
262             if codes != None:
263                 self.remove_code_combination(ID)
264                 self.create_code_combination(ID,codes)
265
266     def display_patient_menu():
267         print("Patient Menu\n")
268         print("1) Add Patient")
269         print("2) Delete Patient")
270         print("3) Patient Details")
271         print("4) Amend Patient")
272         print("0) Exit\n")
273
274     def get_valid_menu_option():
275         valid = False
276         while valid == False:
277             try:
278                 choice = int(input("Menu Option: "))
279                 if choice in [0,1,2,3,4]:
280                     valid = True
281                 else:
282                     print("Error - Invalid")
283                     valid = False
284             except:
285                 print("Error - Invalid")
286                 valid = False
287         return choice
288
289     def add_patient(controller):
290         title = input("Title: ")
291         fname = input("First Name: ")
292         lname = input("Last Name: ")
293         ad1 = input("Address Line 1: ")
294         ad2 = input("Address Line 2: ")
295         town = input("Town: ")
296         pc = input("Post Code:  ")
297         DoB = input("Date of Birth: ")
298         gender = input("Gender: ")
299         telNo = input("Telephone Number: ")
300         email = input("Email: ")
301         DoO = input("Date of Op: ")
302         conFname = input("Consultant First Name: ")
303         conLname = input("Consultant Last Name: ")
```

```
304         icName = input("Insurance Company: ")
305         codes = input("Codes: ")
306         if codes != None:
307             letterlist = []
308             for count in range(len(codes)):
309                 if codes[count] != ',':
310                     letterlist.append(codes[count])
311             codes = []
312             for count in range(0,len(letterlist),5):
313                 currentcode =
314     letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
315     3]+letterlist[count+4]
316                 codes.append(currentcode)
317 
318     title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,email,DoO,conFname,conLn
319     ame,icName,codes =
320     validation.validate_patient(title,fname,lname,ad1,ad2,town,pc,DoB,gender,te
321     lNo,email,DoO,conFname,conLname,icName,codes)
322 
323     controller.add_patient(title,fname,lname,ad1,ad2,town,pc,DoB,gender,telNo,e
324     mail,DoO,conLname,icName,codes)
325 
326     def delete_patient(controller):
327         patient = input("patient ID: ")
328         controller.delete_patient(patient)
329 
330     def patient_details(controller):
331         searchBy = input("Name or ID?: ")
332         if searchBy == "Name":
333             fname = input("First Name: ")
334             if fname == "":
335                 fname = None
336             lname = input("Last Name: ")
337             if lname == "":
338                 lname = None
339             details = controller.patient_details(fname,lname)
340         else:
341             ID = input("Patient ID: ")
342             details = controller.patient_details(ID)
343         #format output?
344         print(details)
345 
346     def amend_patient(controller):
347         ID = input("Patient ID: ")
348         print("Update-")
349         title = input("Title: ")
350         if title == "":
351             title = None
352         fname = input("First Name: ")
353         if fname == "":
354             fname = None
355         lname = input("Last Name: ")
356         if lname == "":
357             lname = None
358         ad1 = input("Address Line 1: ")
359         if ad1 == "":
360             ad1 = None
361         ad2 = input("Address Line 2: ")
362         if ad2 == "":
363             ad2 = None
364         town = input("Town: ")
```

```
365         if town == "":
366             town = None
367         pc = input("Post Code:  ")
368         if pc == "":
369             pc = None
370         DoB = input("Date of Birth: ")
371         if DoB == "":
372             DoB = None
373         gender = input("Gender: ")
374         if gender == "":
375             gender = None
376         telNo = input("Telephone Number: ")
377         if telNo == "":
378             telNo = None
379         email = input("Email: ")
380         if email == "":
381             email = None
382         DoO = input("Date of Op: ")
383         if DoO == "":
384             DoO = None
385         conFname = input("Consultant First Name: ")
386         if conFname == "":
387             conFname = None
388         conLname = input("Consultant Last Name: ")
389         if conLname == "":
390             conLname = None
391         icName = input("Insurance Company: ")
392         if icName == "":
393             icName = None
394         codes = input("Codes: ")
395         if codes == "":
396             codes = None
397         if codes != None:
398             letterlist = []
399             for count in range(len(codes)):
400                 if codes[count] != ',':
401                     letterlist.append(codes[count])
402             codes = []
403             for count in range(0,len(letterlist),5):
404                 currentcode =
405     letterlist[count]+letterlist[count+1]+letterlist[count+2]+letterlist[count+
406     3]+letterlist[count+4]
407                 codes.append(currentcode)
408 
409     controller.amend_patient(ID,title,fname,lname,ad1,ad2,town,pc,DoB,gender,te
410     lNo,email,DoO,conLname,icName,codes)
411 
412     def main():
413         patientController = Patient_Controller()
414         choice = -1
415         while choice != 0:
416             display_patient_menu()
417             choice = get_valid_menu_option()
418             if choice == 1:
419                 add_patient(patientController)
420             elif choice == 2:
421                 delete_patient(patientController)
422             elif choice == 3:
423                 patient_details(patientController)
424             elif choice == 4:
425                 amend_patient(patientController)
```

```
426             else:
427                 print("Exit.")
428 
429 if __name__ == "__main__":
430     main()
```

10.26. patient_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import add_patient_gui
import search_patient_gui

class patientWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.setLayout(self.patientGrid())
        return self

    def patientGrid(self):
        self.patientLayout = QGridLayout()
        self.patientImage = QLabel()
        self.patientImage.setPixmap(QPixmap("images/Patients.jpg"))

self.patientImage.setFixedWidth(self.patientImage.sizeHint().width())

self.patientImage.setFixedHeight(self.patientImage.sizeHint().height())
        self.addButton = QPushButton()
        self.addButton.setStyleSheet("background-image:
url(images/Add.jpg);")
        self.addButton.setMinimumHeight(204)
        self.addButton.setMinimumWidth(204)
        self.addLabel = QLabel("Add Patient",self)
        self.findButton = QPushButton()
        self.findButton.setStyleSheet("background-image:
url(images/Search.jpg);")
        self.findButton.setMinimumHeight(204)
        self.findButton.setMinimumWidth(204)
        self.findLabel = QLabel("Find Patient",self)
        self.patientLayout.addWidget(self.addButton,0,0)
        self.patientLayout.addWidget(self.addLabel,1,0)
        self.patientLayout.addWidget(self.patientImage,0,1)
        self.patientLayout.addWidget(self.findButton,0,2)
        self.patientLayout.addWidget(self.findLabel,1,2)
        self.addButton.clicked.connect(self.AddTest)
        self.findButton.clicked.connect(self.FindTest)
        return self.patientLayout

    def AddTest(self):
        self.addWindow = add_patient_gui.addPatientWindow()
        self.addWindow.show()
        self.addWindow.raise_()

    def FindTest(self):
        self.findWindow = search_patient_gui.searchPatientWindow()
        self.findWindow.show()
        self.findWindow.raise_()


if __name__ == "__main__":
    application = QApplication(sys.argv)
    patWindow = patientWindow()
```

```
59          patWindow.show()
60          patWindow.raise_()
61          application.exec_()
```

10.27. search_consultant_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import consultant_controller
import output_consultant_gui
import gui_validation

class searchConsultantWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.searchView = QWidget()
        self.searchView.setLayout(self.searchConsultantGrid())
        self.setCentralWidget(self.searchView)
        self.setWindowTitle("Consultant - Search")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def searchConsultantGrid(self):
        self.consultantLayout = QVBoxLayout()
        self.operationImage = QLabel()
        self.operationImage.setPixmap(QPixmap("images/Search.jpg"))

self.operationImage.setFixedWidth(self.operationImage.sizeHint().width())

self.operationImage.setFixedHeight(self.operationImage.sizeHint().height())
        self.searchGroupBox = QGroupBox('Search By...',self)
        self.searchLayout = QVBoxLayout()
        self.idRadioButton = QRadioButton("Consultant ID",self)
        self.idRadioButton.setChecked(1)
        self.nameRadioButton = QRadioButton("Name",self)
        self.idLineEdit = QLineEdit("Consultant ID")
        self.fnameLineEdit = QLineEdit("First Name")
        self.lnameLineEdit = QLineEdit("Surname")
        self.searchButton = QPushButton("Search")
        self.searchLayout.addWidget(self.idRadioButton)
        self.searchLayout.addWidget(self.nameRadioButton)
        self.searchLayout.addWidget(self.idLineEdit)
        self.searchLayout.addWidget(self.fnameLineEdit)
        self.searchLayout.addWidget(self.lnameLineEdit)
        self.searchLayout.addWidget(self.searchButton)
        self.searchGroupBox.setLayout(self.searchLayout)
        self.consultantLayout.addWidget(self.operationImage)
        self.consultantLayout.addWidget(self.searchGroupBox)
        self.searchButton.clicked.connect(self.search)
        return self.consultantLayout

    def search(self):
        existing = self.setParameters()
        if existing == True:
            valid, errormsg =
gui_validation.validate_consultant_search(self.search,self.data) # dont
like ids
            if valid == True:
                existing = self.existingSearch()
                if existing == True:
                    self.outputWindow =
output_consultant_gui.outputConsultantWindow(self.search,self.data)
                    self.outputWindow.show()
```

```
                    self.outputWindow.raise_()
                    self.close()
            else:
                QMessageBox.about(self, "Error", errormsg)

    def setParameters(self):
        ID = self.idRadioButton.isChecked()
        name = self.nameRadioButton.isChecked()
        if ID == True:
            self.search = "ID"
            self.data = [self.idLineEdit.text()]
            return True
        elif name == True:
            self.search = "name"
            self.data =
[self.fnameLineEdit.text(),self.lnameLineEdit.text()]
            return True
        else:
            QMessageBox.about(self, "Error", "Error- No search method
selected.")
            return False

    def existingSearch(self):
        self.tempController = consultant_controller.Consultant_Controller()
        if self.search == "ID":
            data = self.tempController.consultant_details(ID=self.data[0])
        else:
            data =
self.tempController.consultant_details(fname=self.data[0],lname=self.data[1
])
        if data == []:
            QMessageBox.about(self, "Error", "Error- Consultant not
found.")
            return False
        else:
            return True

if __name__ == "__main__":
    application = QApplication(sys.argv)
    searchConsultantWindow = searchConsultantWindow()
    searchConsultantWindow.show()
    searchConsultantWindow.raise_()
    application.exec_()
```

10.28. search_patient_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import patient_controller
import output_patient_gui
import gui_validation

class searchPatientWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.searchView = QWidget()
        self.searchView.setLayout(self.searchPatientGrid())
        self.setCentralWidget(self.searchView)
        self.setWindowTitle("Patient - Search")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def searchPatientGrid(self):
        self.patientLayout = QVBoxLayout()
        self.operationImage = QLabel()
        self.operationImage.setPixmap(QPixmap("images/Search.jpg"))

self.operationImage.setFixedWidth(self.operationImage.sizeHint().width())

self.operationImage.setFixedHeight(self.operationImage.sizeHint().height())
        self.searchGroupBox = QGroupBox('Search By...',self)
        self.searchLayout = QVBoxLayout()
        self.idRadioButton = QRadioButton("Patient ID",self)
        self.idRadioButton.setChecked(1)
        self.nameRadioButton = QRadioButton("Name",self)
        self.idLineEdit = QLineEdit("Patient ID")
        self.fnameLineEdit = QLineEdit("First Name")
        self.lnameLineEdit = QLineEdit("Surname")
        self.searchButton = QPushButton("Search")
        self.searchLayout.addWidget(self.idRadioButton)
        self.searchLayout.addWidget(self.nameRadioButton)
        self.searchLayout.addWidget(self.idLineEdit)
        self.searchLayout.addWidget(self.fnameLineEdit)
        self.searchLayout.addWidget(self.lnameLineEdit)
        self.searchLayout.addWidget(self.searchButton)
        self.searchGroupBox.setLayout(self.searchLayout)
        self.patientLayout.addWidget(self.operationImage)
        self.patientLayout.addWidget(self.searchGroupBox)
        self.searchButton.clicked.connect(self.search)
        return self.patientLayout

    def search(self):
        existing = self.setParameters()
        if existing == True:
            valid, errormsg =
gui_validation.validate_patient_search(self.search,self.data)
            if valid == True:
                existing = self.existingSearch()
                if existing == True:
                    self.outputWindow =
output_patient_gui.outputPatientWindow(self.search,self.data)
                    self.outputWindow.show()
                    self.outputWindow.raise_()
```

```
                    self.close()
                else:
                    QMessageBox.about(self, "Error", errormsg)

    def setParameters(self):
        ID = self.idRadioButton.isChecked()
        name = self.nameRadioButton.isChecked()
        if ID == True:
            self.search = "ID"
            self.data = [self.idLineEdit.text()]
            return True
        elif name == True:
            self.search = "name"
            self.data =
[self.fnameLineEdit.text(),self.lnameLineEdit.text()]
            return True
        else:
            QMessageBox.about(self, "Error", "Error- No search method
selected.")
            return False

    def existingSearch(self):
        tempController = patient_controller.Patient_Controller()
        if self.search == "ID":
            data = tempController.existing(ID=self.data[0])
        else:
            data =
tempController.existing(fname=self.data[0],lname=self.data[1])
        if data == []:
            QMessageBox.about(self, "Error", "Error- Patient not found.")
            return False
        else:
            return True

if __name__ == "__main__":
    application = QApplication(sys.argv)
    searchPatientWindow = searchPatientWindow()
    searchPatientWindow.show()
    searchPatientWindow.raise_()
    application.exec_()
```

10.29. search_user_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import user_controller
import output_user_gui
import gui_validation

class searchUserWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.searchView = QWidget()
        self.searchView.setLayout(self.searchUserGrid())
        self.setCentralWidget(self.searchView)
        self.setWindowTitle("User - Search")
        self.setWindowIcon(QIcon('images/CMP_Logo.png'))

    def searchUserGrid(self):
        self.userLayout = QVBoxLayout()
        self.operationImage = QLabel()
        self.operationImage.setPixmap(QPixmap("images/Search.jpg"))

self.operationImage.setFixedWidth(self.operationImage.sizeHint().width())

self.operationImage.setFixedHeight(self.operationImage.sizeHint().height())
        self.searchGroupBox = QGroupBox('Search By...',self)
        self.searchLayout = QVBoxLayout()
        self.idRadioButton = QRadioButton("User ID",self)
        self.idRadioButton.setChecked(1)
        self.nameRadioButton = QRadioButton("Username",self)
        self.idLineEdit = QLineEdit("User ID")
        self.nameLineEdit = QLineEdit("Username")
        self.searchButton = QPushButton("Search")
        self.searchLayout.addWidget(self.idRadioButton)
        self.searchLayout.addWidget(self.nameRadioButton)
        self.searchLayout.addWidget(self.idLineEdit)
        self.searchLayout.addWidget(self.nameLineEdit)
        self.searchLayout.addWidget(self.searchButton)
        self.searchGroupBox.setLayout(self.searchLayout)
        self.userLayout.addWidget(self.operationImage)
        self.userLayout.addWidget(self.searchGroupBox)
        self.searchButton.clicked.connect(self.search)
        return self.userLayout

    def search(self):
        existing = self.setParameters()
        if existing == True:
            valid, errormsg =
gui_validation.validate_user_search(self.search,self.data)
            if valid == True:
                existing = self.existingSearch()
                if existing == True:
                    self.outputWindow =
output_user_gui.outputUserWindow(self.search,self.data)
                    self.outputWindow.show()
                    self.outputWindow.raise_()
                    self.close()
            else:
```

```
                QMessageBox.about(self, "Error", errormsg)

    def setParameters(self):
        ID = self.idRadioButton.isChecked()
        UN = self.nameRadioButton.isChecked()
        if ID == True:
            self.search = "ID"
            self.data = self.idLineEdit.text()
            return True
        elif UN == True:
            self.search = "UN"
            self.data = self.nameLineEdit.text()
            return True
        else:
            QMessageBox.about(self, "Error", "Error- No search method
selected.")
            return False

    def existingSearch(self):
        self.tempController = user_controller.User_Controller()
        if self.search == "ID":
            data = self.tempController.user_details(ID=self.data)
        else:
            data = self.tempController.user_details(UN=self.data)
        if data == []:
            QMessageBox.about(self, "Error", "Error- User not found.")
            return False
        else:
            return True

if __name__ == "__main__":
    application = QApplication(sys.argv)
    searchUserWindow = searchUserWindow()
    searchUserWindow.show()
    searchUserWindow.raise_()
    application.exec_()
```

10.30. user_controller.py

```
from db_controller import *

class User_Controller(db_controller):
    """Creates controller for Users"""
    def __init__(self):
        super().__init__()

    def add_user(self,UN,PW,AC,email):
        sql = """insert into user
                 (Username,Password,UserAccess,UserEmail)
                 values
                 ('{0}','{1}','{2}','{3}')""".format(UN,PW,AC,email)
        self.query(sql)

    def delete_user(self,ID):
        sql = """delete from user
                 where UserID = '{0}'""".format(ID)
        self.query(sql)

    def user_details(self,ID=None,UN=None):
        sql = None
        if ID != None:
            sql = """select * from user where UserID = '{0}'""".format(ID)
        elif UN != None:
            sql = """select * from user where Username =
'{0}'""".format(UN)
        if sql != None:
            return self.select_query(sql)
        else:
            return None

    def amend_user(self,ID,UN,PW,AC,email):
        updates = []
        if UN != None:
            updates.append(("Username",UN))
        if PW != None:
            updates.append(("Password",PW))
        if AC != None:
            updates.append(("UserAccess",AC))
        if email != None:
            updates.append(("UserEmail",email))
        sql = "update user set "
        for item in updates:
            sql += "{0}='{1}', ".format(item[0],item[1])
        sql = sql[:-2]
        sql += " where UserID='{0}'".format(ID)
        self.query(sql)

def display_user_menu():
    print("User Menu\n")
    print("1) Add User")
    print("2) Delete User")
    print("3) User Details")
    print("4) Amend User")
    print("0) Exit\n")

def get_valid_menu_option():
    valid = False
    while valid == False:
        try:
```

```
61              choice = int(input("Menu Option: "))
62              if choice in [0,1,2,3,4]:
63                  valid = True
64              else:
65                  print("Error - Invalid")
66                  valid = False
67          except:
68              print("Error - Invalid")
69              valid = False
70      return choice
71
72  def add_user(controller):
73      UN = input("Username: ")
74      PW = input("Password: ")
75      AC = input("Access Level: ")
76      email = input("Email: ")
77      UN,PW,AC,email = validation.validate_user(UN,PW,AC,email)
78      controller.add_user(UN,PW,AC,email)
79
80  def delete_user(controller):
81      user = input("User ID: ")
82      controller.delete_user(user)
83
84  def user_details(controller):
85      searchBy = input("Username or ID?: ")
86      if searchBy == "Username":
87          UN = input("Username: ")
88          details = controller.user_details(UN=UN)
89      else:
90          ID = input("User ID: ")
91          details = controller.user_details(ID=ID)
92      print(details)
93
94
95  def amend_user(controller):
96      ID = input("User ID: ")
97      print("Update-")
98      UN = input("Username: ")
99      if title == "":
100         title = None
101     PW = input("Password: ")
102     AC = input("Access Level: ")
103     email = input("Email: ")
104     controller.amend_user(ID,UN,PW,AC,email)
105
106
107 def main():
108     userController = User_Controller()
109     choice = -1
110     while choice != 0:
111         display_user_menu()
112         choice = get_valid_menu_option()
113         if choice == 1:
114             add_user(userController)
115         elif choice == 2:
116             delete_user(userController)
117         elif choice == 3:
118             user_details(userController)
119         elif choice == 4:
120             amend_user(userController)
121         else:
```

```
122                 print("Exit.")
123
124     if __name__ == "__main__":
125         main()
```

10.31. user_gui.py

```
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
import add_user_gui
import search_user_gui

class userWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.createUI()

    def createUI(self):
        self.setLayout(self.userGrid())
        return self

    def userGrid(self):
        self.userLayout = QGridLayout()
        self.userImage = QLabel()
        self.userImage.setPixmap(QPixmap("images/Users.png"))
        self.userImage.setFixedWidth(self.userImage.sizeHint().width())
        self.userImage.setFixedHeight(self.userImage.sizeHint().height())
        self.addButton = QPushButton()
        self.addButton.setStyleSheet("background-image:
url(images/Add.jpg);")
        self.addButton.setMinimumHeight(204)
        self.addButton.setMinimumWidth(204)
        self.addLabel = QLabel("Add User",self)
        self.findButton = QPushButton()
        self.findButton.setStyleSheet("background-image:
url(images/Search.jpg);")
        self.findButton.setMinimumHeight(204)
        self.findButton.setMinimumWidth(204)
        self.findLabel = QLabel("Find User",self)
        self.userLayout.addWidget(self.addButton,0,0)
        self.userLayout.addWidget(self.addLabel,1,0)
        self.userLayout.addWidget(self.userImage,0,1)
        self.userLayout.addWidget(self.findButton,0,2)
        self.userLayout.addWidget(self.findLabel,1,2)
        self.addButton.clicked.connect(self.AddTest)
        self.findButton.clicked.connect(self.FindTest)
        return self.userLayout

    def AddTest(self):
        self.addWindow = add_user_gui.addUserWindow()
        self.addWindow.show()
        self.addWindow.raise_()

    def FindTest(self):
        self.findWindow = search_user_gui.searchUserWindow()
        self.findWindow.show()
        self.findWindow.raise_()

if __name__ == "__main__":
    application = QApplication(sys.argv)
    userWindow = userWindow()
    userWindow.show()
    userWindow.raise_()
    application.exec_()
```

10.32. encryption.py

```
from Crypto.Cipher import AES

def encrypt_data(data):
    encryptor = AES.new('abcdefghijklmnop', AES.MODE_ECB)
    encrypt_data = encryptor.encrypt(data)
    return encrypt_data

def decrypt_data(data):
    decryptor = AES.new('abcdefghijklmnop', AES.MODE_ECB)
    decrypt_data = decryptor.decrypt(data)
    return decrypt_data

def user(switch,UN,PW,AC,email):
    if UN != None:
        if switch == "encrypt":
            UN = encrypt_data(UN)
        else:
            UN = decrypt_data(UN)
    if PW != None:
        if switch == "encrypt":
            PW = encrypt_data(PW)
        else:
            PW = decrypt_data(PW)
    if AC != None:
        if switch == "encrypt":
            AC = encrypt_data(AC)
        else:
            AC = decrypt_data(AC)
    if email != None:
        if switch == "encrypt":
            email = encrypt_data(email)
        else:
            email = decrypt_data(email)
    return UN,PW,AC,email

if __name__ == "__main__":
    data = "i am a qwert abc"
    encrypt_data = encrypt_data(data)
    print(encrypt_data)
```

10.33. log_in.py

```
1   import user_controller
2
3   def get_user_details(UN):
4       tempController = user_controller.User_Controller()
5       return tempController.user_details(UN=UN)
6
7   def main():
8       print("Please Log In-")
9       valid = False
10      while valid != True:
11          username = input("Username: ")
12          password = input("Password: ")
13          details = get_user_details(username)
14          if details == []:
15              print("Username does not exist.")
16          else:
17              if password == details[0][2]:
18                  print("Access Granted.")
19                  valid = True
20              else:
21                  print("Incorrect Password.")
22      return details[0][3]
23      #return all details?
24
25
26  if __name__ == "__main__":
27      pass
```