Post-prototype:

5.1: Prototype feedback

I decided it would be smart to contact the clinic to see what their opinions on my current prototype was. This would show me if my current prototype was suitable and if I should continue using my time to further develop the prototype into the software development. As a result, I plan to send out a questionnaire to the staff members at the clinic. This would be sent to twenty staff members.

I created a questionnaire that ranged from strongly agree to strongly disagree. I believe that these replies will allow me to get a good input on what people think about my program. I also believe that I should add a comment area at the bottom of the questionnaire, this would allow more complex feedback to the prototype that I have given to the staff members. I believe that this questionnaire will give me suitable feedback on what staff members think about my Prototype and the main strengths and weaknesses of the system.

5.2: Questionnaire

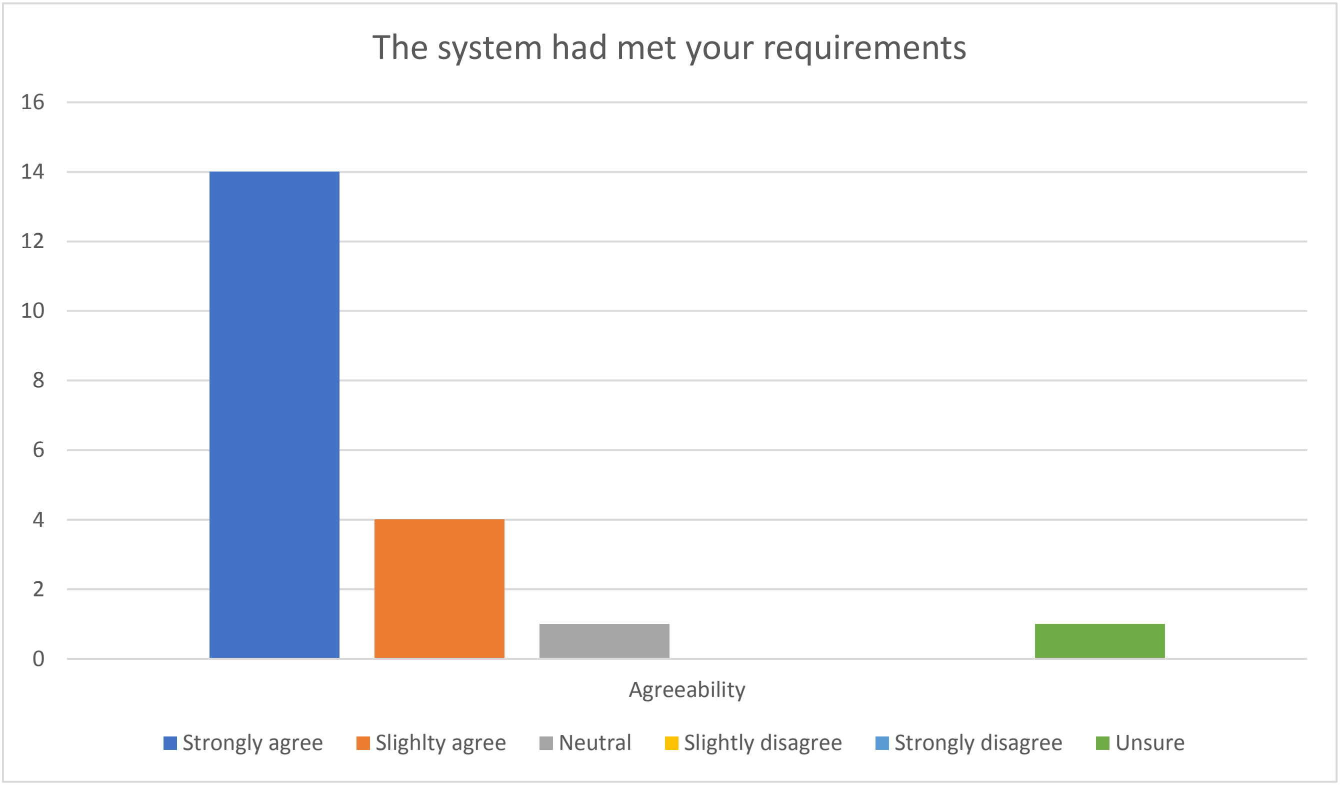
Questions

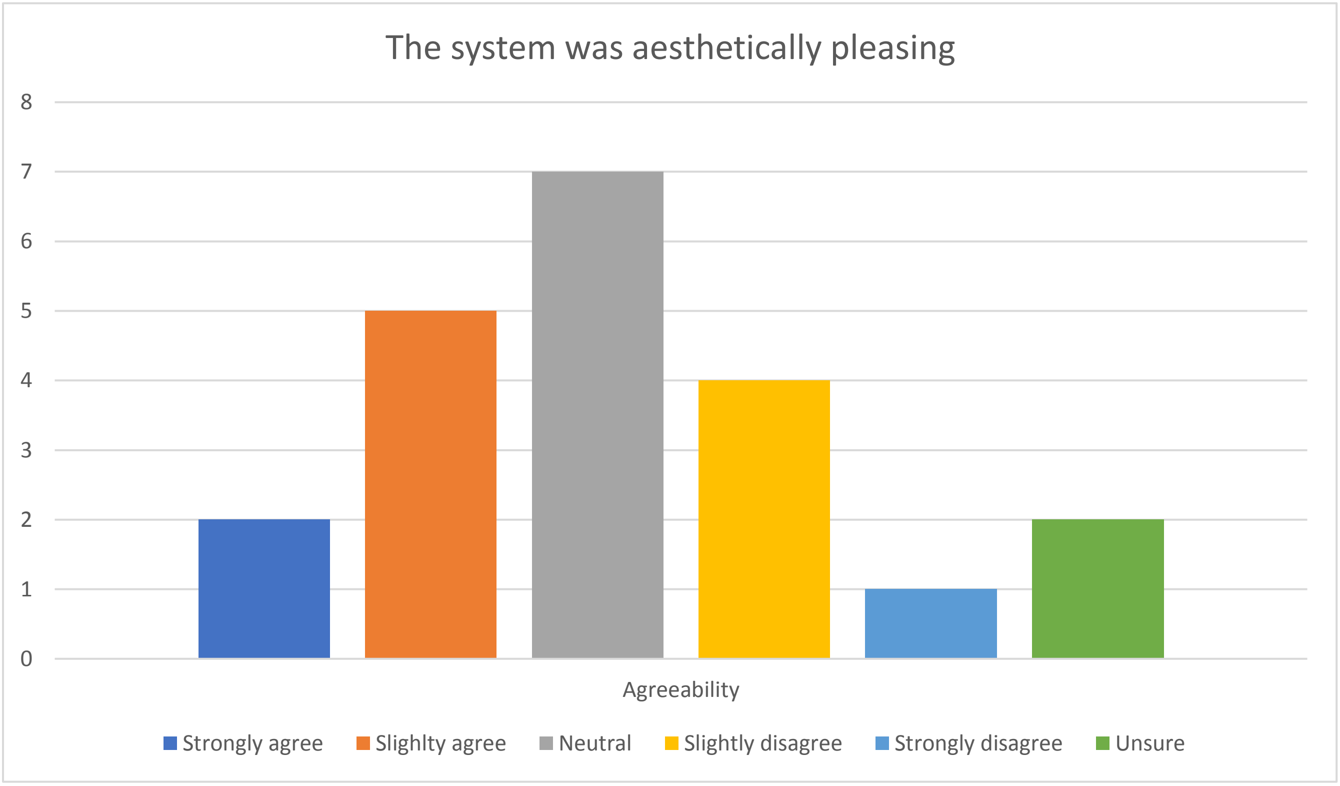
I asked them how much they agreed with the following:

- The system met your requirements
- The system was aesthetically pleasing
- The system was intuitive and easy to use
- The system was functional
- The system had few to no bugs
- The system was secure
- The outputs were effective
- The inputs were effective
- The views system was effective
- The data was laid out cleanly

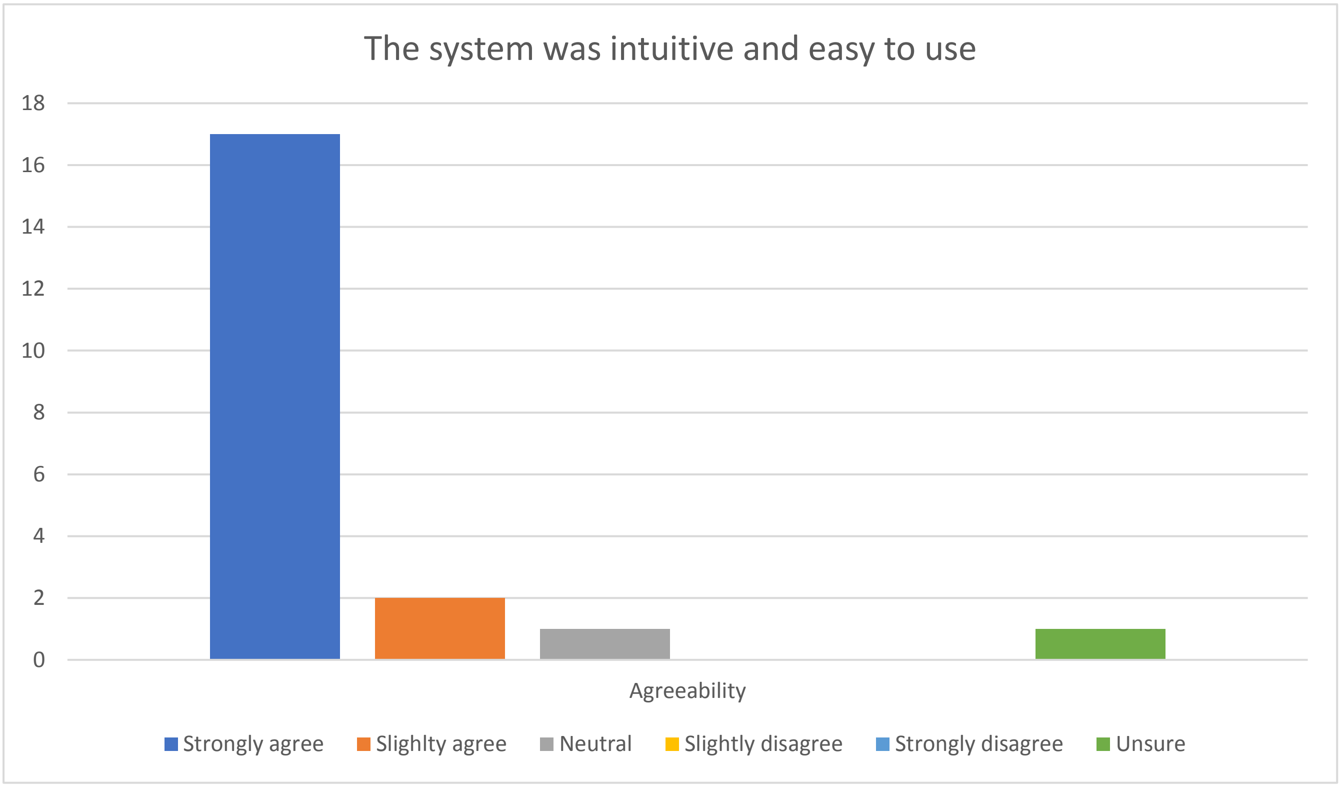
Candidate Number:
3097

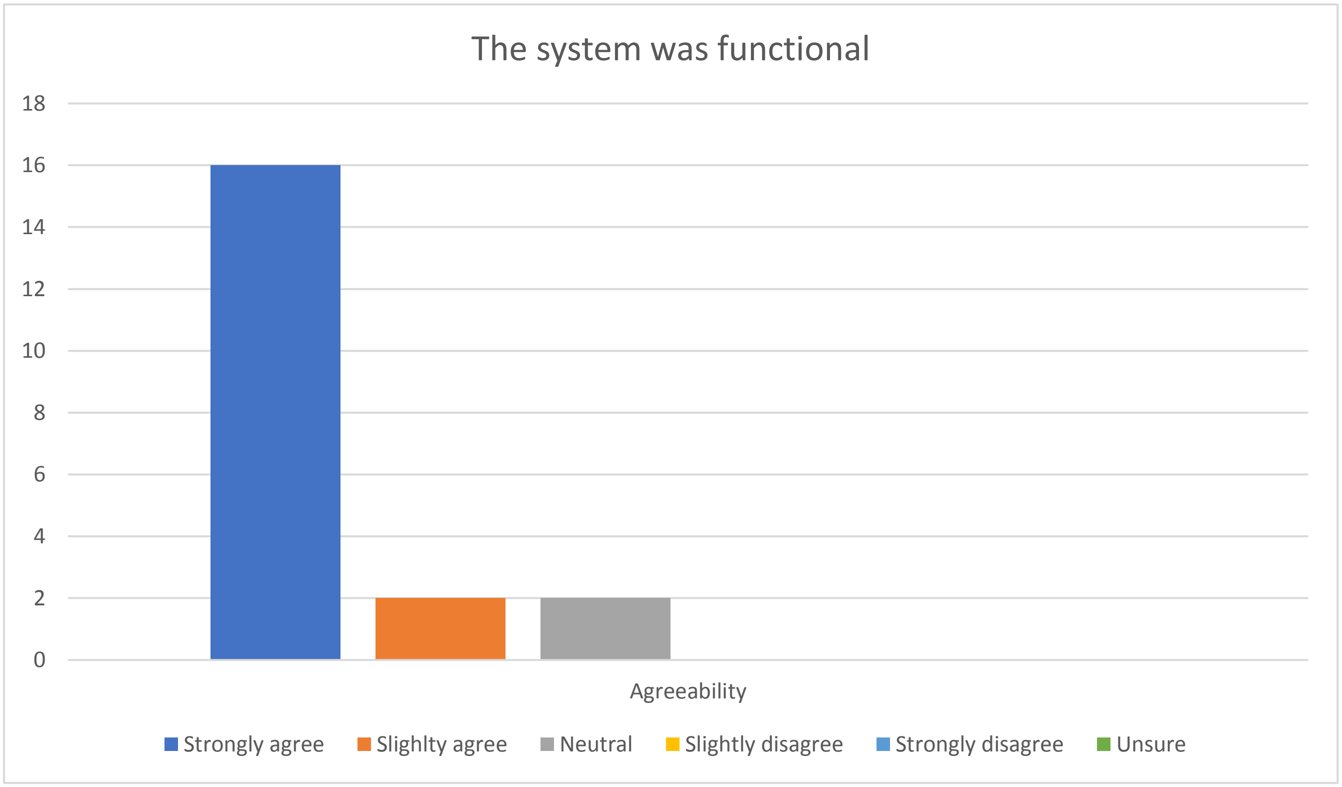
Results

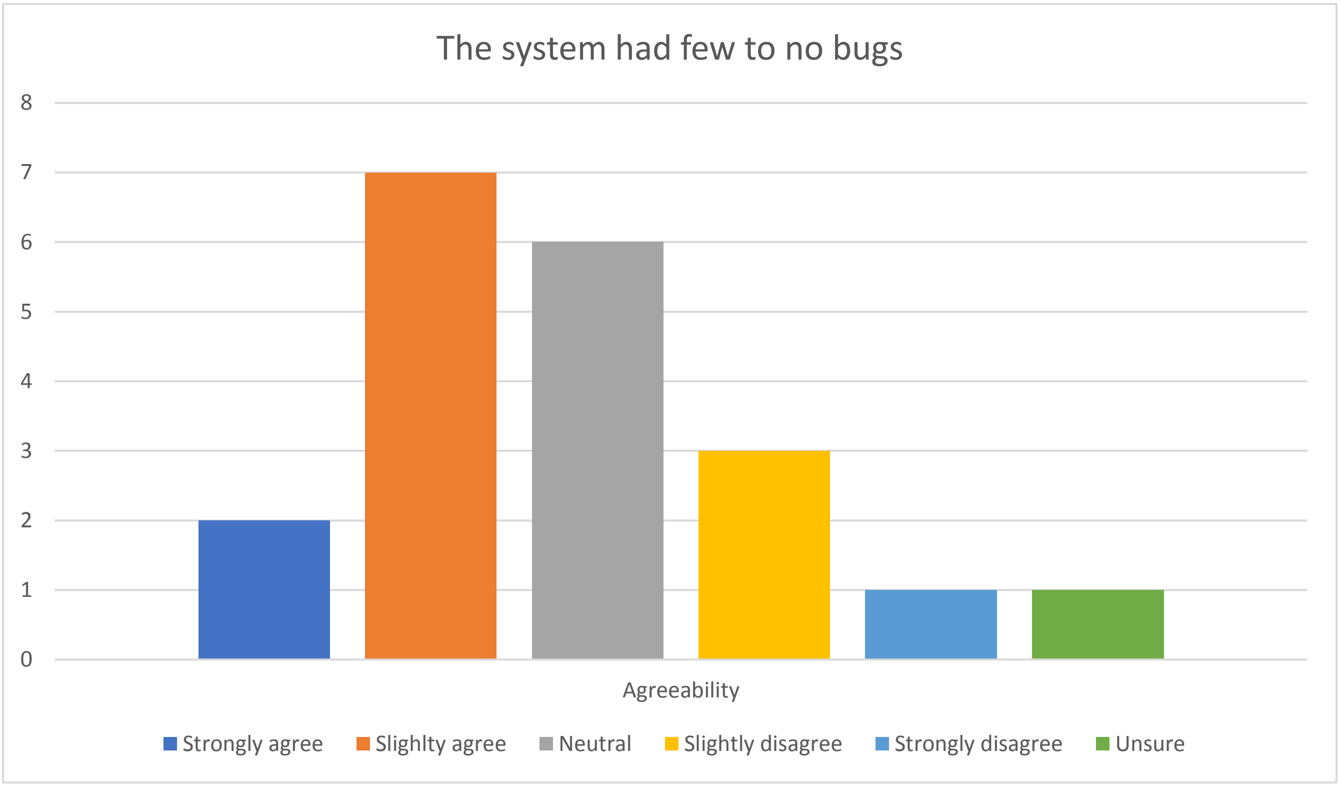


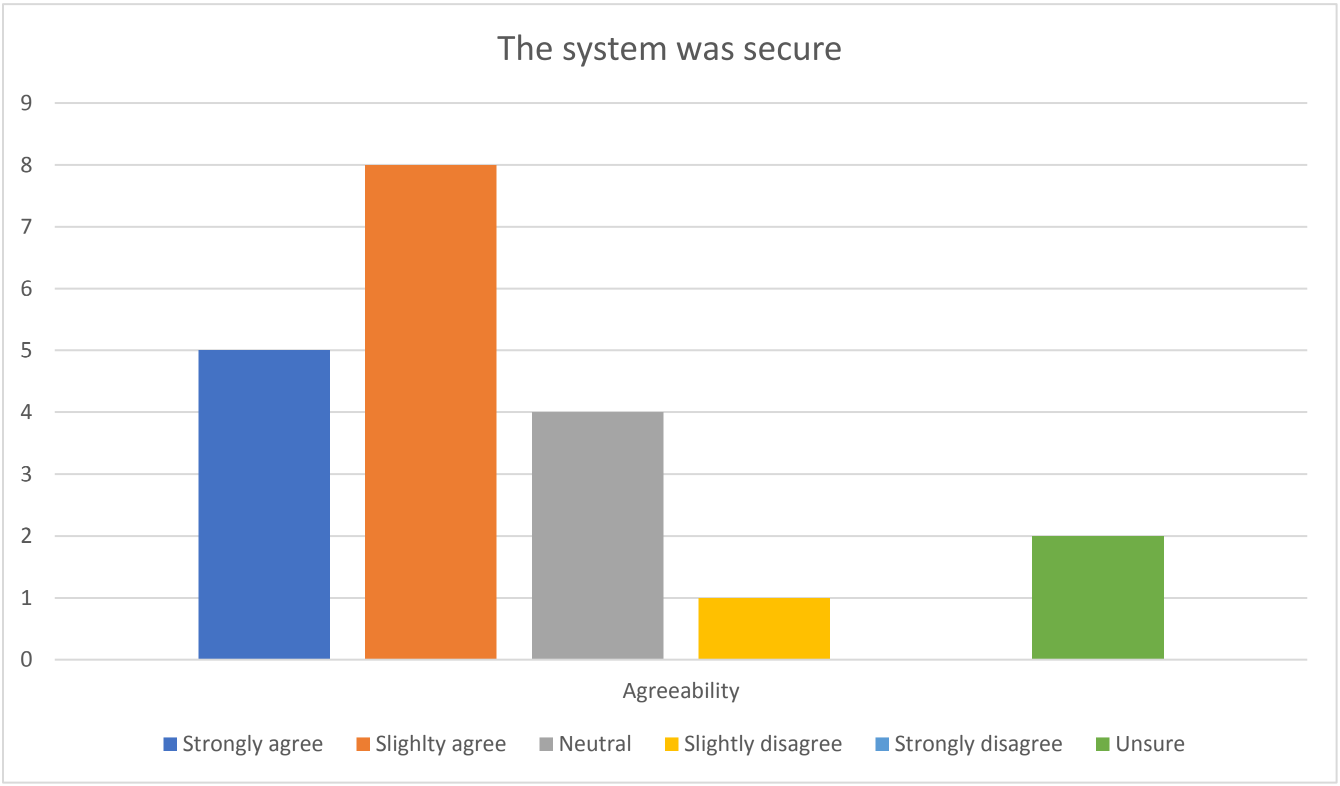


Candidate Number:
3097

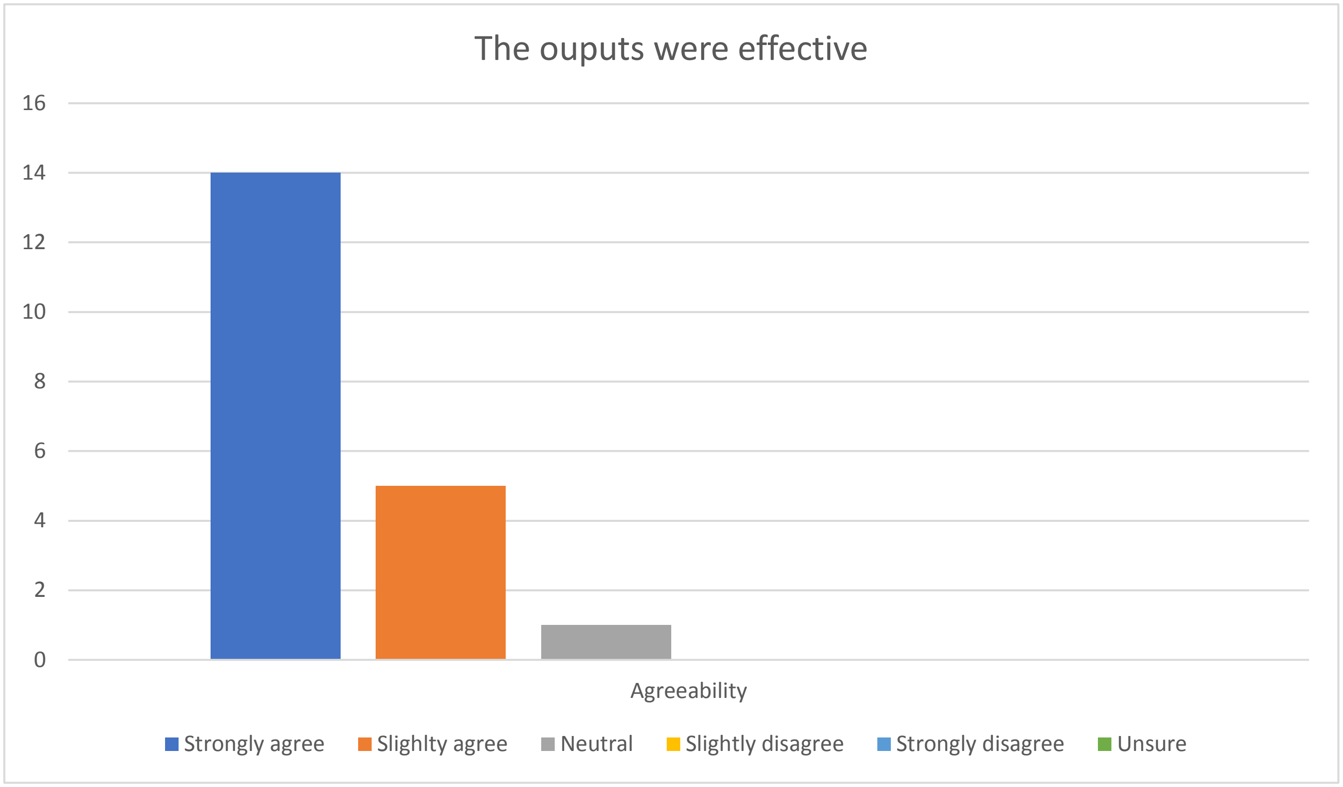


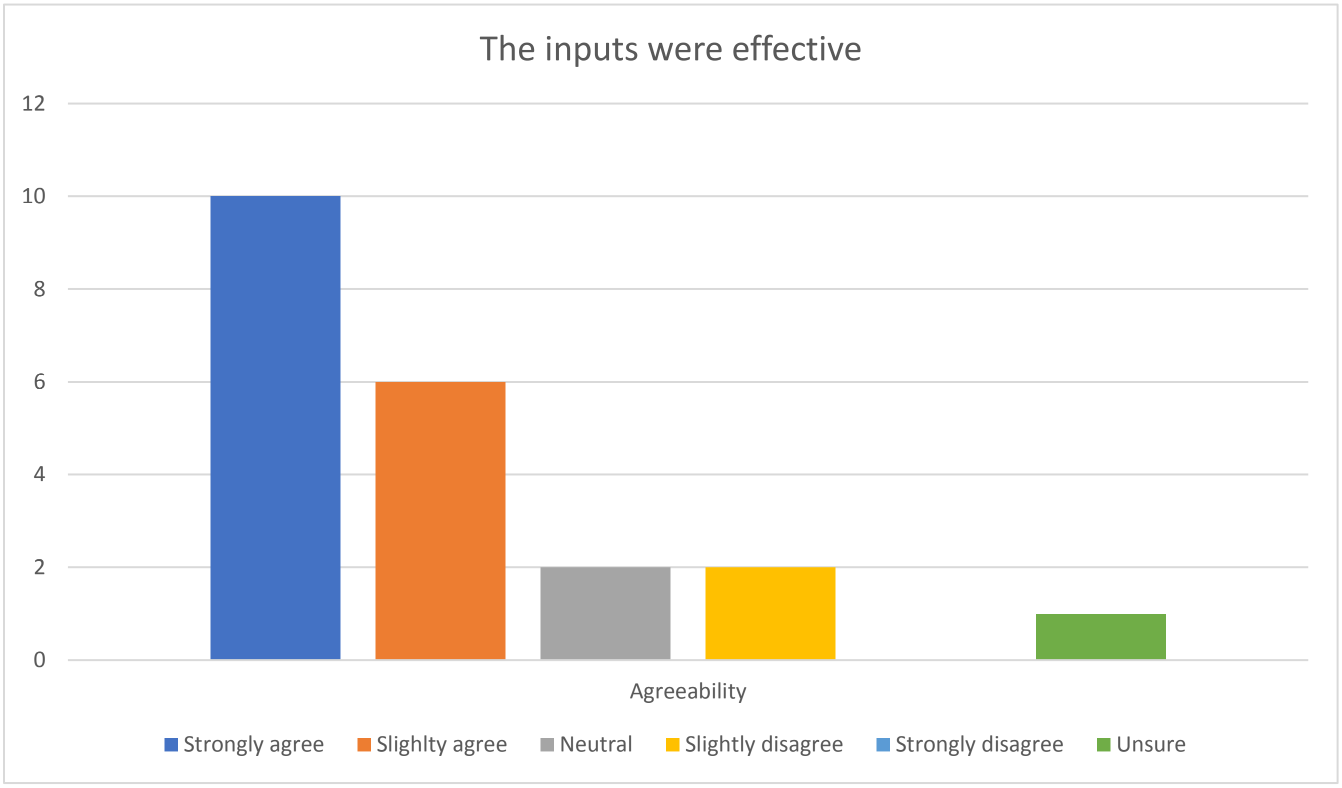



The system had few to no bugs
8
7
6
5
4
3
2
1
0
Agreeability
Strongly agree
Slighlty agree
Neutral
Slightly disagree
Strongly disagree
Unsure

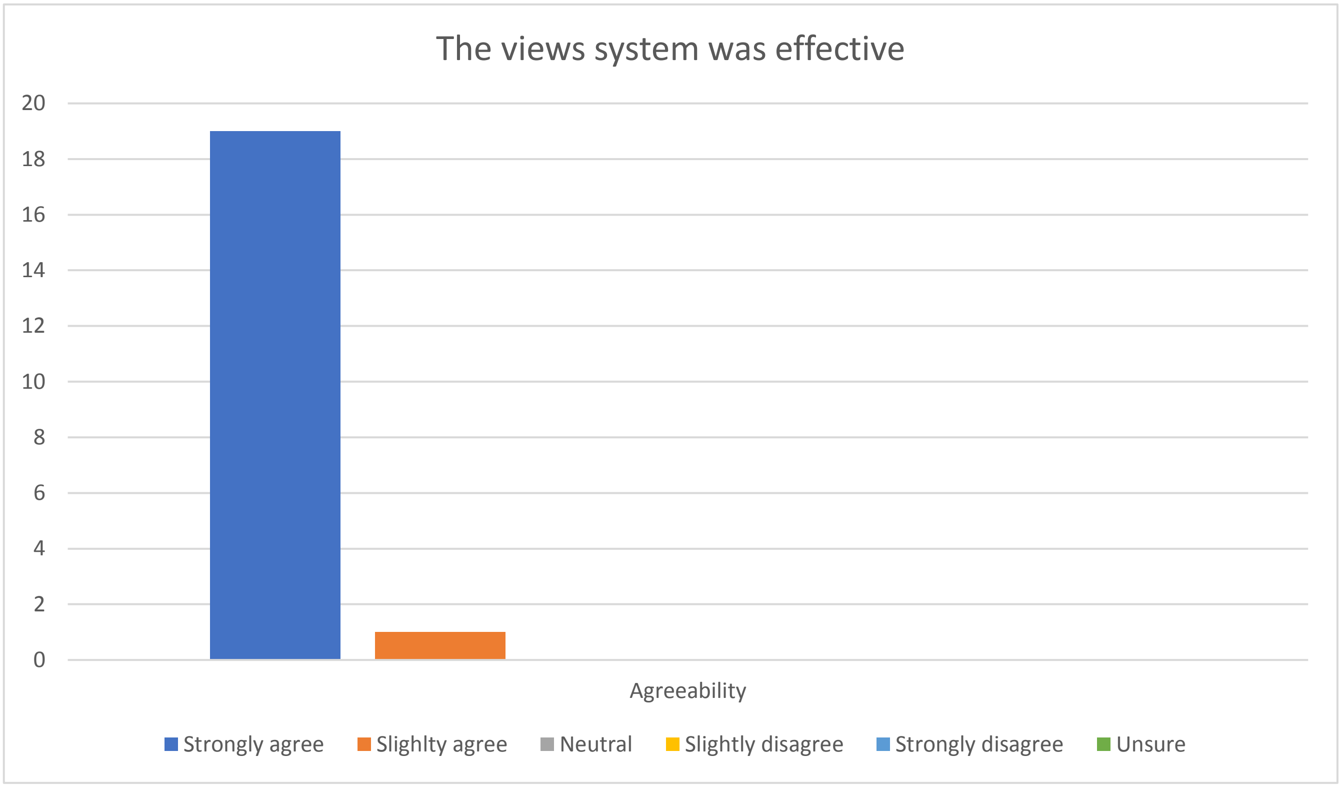

The system was secure
9
8
7
6
5
4
3
2
1
0
Agreeability
Strongly agree
Slighlty agree
Neutral
Slightly disagree
Strongly disagree
Unsure

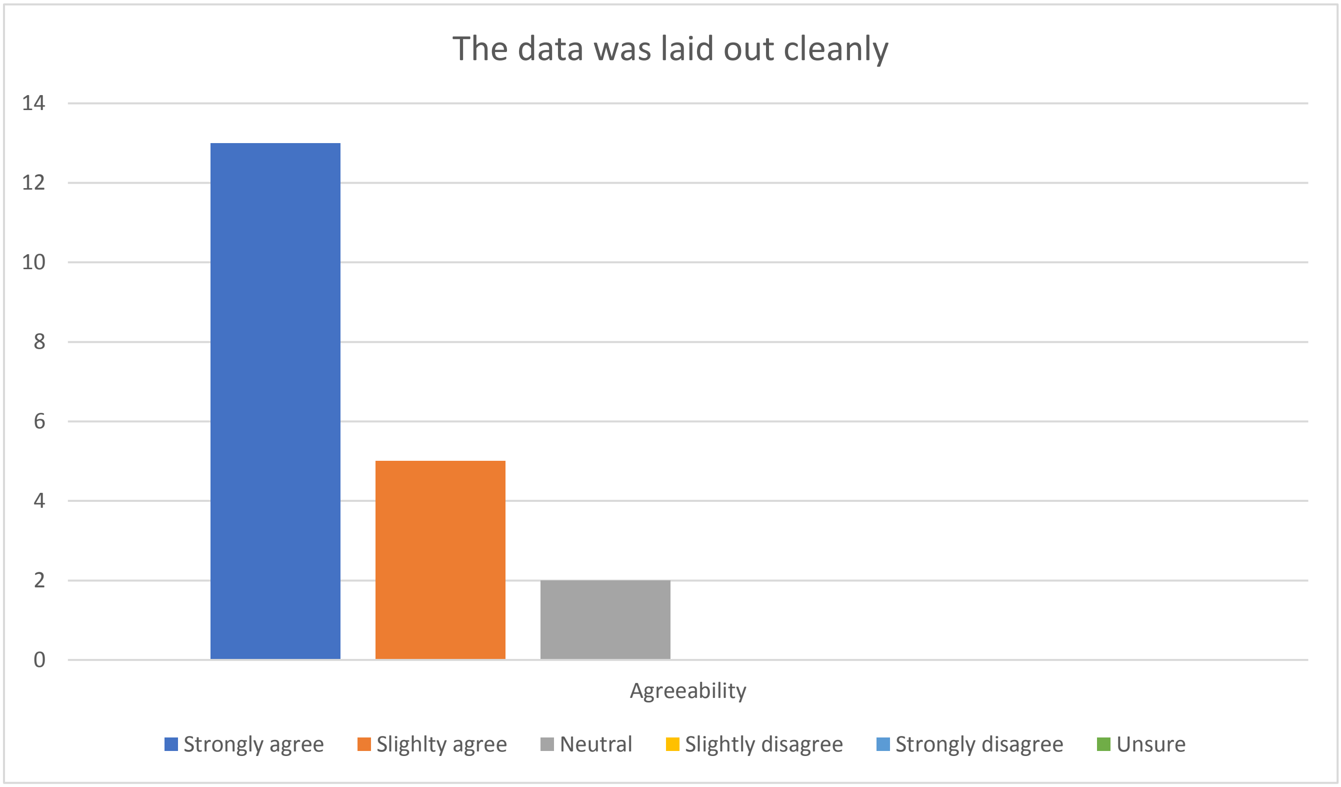
Candidate Number:
3097





Candidate Number:
3097





Comments

At the end of the questionnaire, I added an any extra comments box. This was filled in by 5 users. The comments are:

- 'The program was very intuitive and easy to use, however the login page GUI isn't very aesthetically pleasing'
- 'The program is a massive improvement over our previous system, you should add validation as currently any data can be added in'
- 'The program is very powerful, and I found the view and the ability to easily maneuverer around the program to be very impressive'
- 'You could add more tables for things like finances and the ability to edit records'
- 'A great program, very easy to use'

5.3: Survey analysis

The system met your requirements

The results for this was overwhelmingly positive. Comparing the system to my requirements I have felt all of the ones I aimed to complete for my prototype, however there are still a significant number that I have to finish, this will be completed in the Software development.

The system was aesthetically pleasing

This had mixed opinions. I feel this is for a variety of reasons. I feel like the colouring of the project is bland and I also feel like the format of the login page doesn't look good. I will attempt to centre the login page in my software development.

The system was intuitive and easy to use

This was one of the most positive ratings that I received in the questionnaire, it is clear that the staff members feel like the system is effective and simple to navigate and find out how to use each of the controls.

The system was functional

This also has results that strongly agree, this means that all the functions that I have added to the program work successfully and all features work well. This is an important attribute of my survey and I am happy that the system works well.

The system had few to no bugs

This had a wide range of replies and I agree with the responses. I feel that the problems with the GUI, such as the treeview switching and the widening of the drop boxes in addition to the lack of validation results in a significant number of bugs. I tend to add a large amount of validation in the next iteration of my program.

The system was secure

This had a slight trend to agreeability. I feel like some of the users felt the system was secure due to the views implemented into the program, however the program that I have created does not automatically back records up or encrypt the data that has been added properly.

The outputs were effective

The majority of the users strongly agreed that the outputs were extremely effective, I am happy that I have implemented a range of outputs in my program, with some in the program and some into text files. I will add suitable outputs for the tables I am going to add in the future and add new types of them, such as when a staff member selects a medical record and tries to find out their BMI, an alert will appear stating their BMI.

The inputs were effective

The inputs of the system had a similar, but less positive view than the outputs. I believe this is because they had to type out a form in a popup. I do believe however the inputs are at the best level that they can be, for example I am very happy with how the staff member is able to delete a client by highlighting it.

The views system was effective

The overwhelming majority strongly agreed that the views system was effective. I am very happy with this response. I hope to continue the same techniques that I originally used in my prototype to create my views and I will implement the same function for the remaining tables and features to be added in the software development section.

The data was laid out cleanly

Once again, this had positive views. I am happy with this result, I believe these results are positive due to my use of treeviews which allow the user to easily view all the records, change the width of each column and highlight records that they would like to see. I also feel like my outputs into the text file are of a clean and consistent format.

Comments

The comments have a wide range of comments. The comments tend to be very positive about how intuitive and easy to use the program was however there was some criticism on the lack of validation throughout the program and the GUI on the login page. I hope to take this advice and properly improve my program based on this.

Overall

Overall, I am very pleased with the questionnaire that I have given out and it is clear that the prototype that I have created is effective. I now understand the main issues with my prototype and I aim to fix these issues in the software development section. I am especially happy with the response on how intuitive the program is and how easy it is to use. I will take all this information and use all of it to see which areas I need to reconsider or restructure within the program.

5.4: Areas to be reconsidered or restructured as a result of the feedback

Inputs and outputs

Aim

I aim to add a number of inputs and outputs to the prototype. I feel like all of the inputs and outputs that I have created in the design and investigation section are realistic and should thus all be added in the next stage of the program. The areas I add will be:

Inputs

- The system must be able to create a transaction
- The system must be able to create an appointment
- The system must be able to edit a client
- The system must be able to edit a medical record
- The system must be able to edit a transaction
- The system must be able to edit a staff member
- The system must be able to delete a medical record
- The system must be able to delete a transaction
- The system must be able to cancel an appointment

Outputs

- The system must be able to output a list of all medical records
- The system must be able to output a list of all transactions
- The system must be able to output a list of all appointments
- The system must be able to output the BMI of a selected medical record
- The system must be able to output a list of all results of a search of a medical record table within the program
- The system must be able to output a list of all results of a search of a transaction table within the program
- The system must be able to output a list of all results of a search of an appointment table within the program
- The system must be able to output a list of all results of a search of a medical record table into a text file
- The system must be able to output a list of all results of a search of a transaction table into a text file
- The system must be able to output a list of all results of a search of an appointment table into a text file
- The system must be able to output a list of all transactions on a specific date as well as its total sum into a text file
- The system must be able to output an error when invalid data is inputted

Additional data

Aim

I aim to add three new tables into the program. These will be a medical records table, a transaction table and the appointments table.

Tables added

Appointments:

Field name	Default value	Data type	Field length	Description	Validation
AppointmentID	0	INTEGER	-	Primary key, unique identifier for the table	Automatically created, none required
ClientID	0	INTEGER	-	Foreign key, links to the ClientID table	Automatically created, none required
StaffID	0	TEXT	1	Foreign key, links to the StaffID table	Automatically created, none required
TransactionID	0	TEXT	6	Foreign key, links to the TransactionID table	Automatically created, none required
StartDateAndTime	{YYYY-MM-DD HH:MM:SS}	TEXT	19	The start date and time of the appointment	Format, presence
EndDateAndTime	{YYYY-MM-DD HH:MM:SS}	TEXT	19	The end date and time of the appointment	Format, presence
AppointmentStatus	{Active/ Cancelled}	FLOAT	6	The appointment status	Automatically created, none required

Candidate Number:
3097

Medical records:

Field name	Default value	Data type	Field length	Description	Validation
MedicalRecordID	0	INTEGER	-	Primary key, unique identifier for the table	Automatically created, none required
ClientID	0	INTEGER	-	Foreign key, links to the ClientID table	Automatically created, none required
Sex	{M/F}	TEXT	1	The client's sex	From dropdown menu, so none required
Gender	{Male/Female/ Other}	TEXT	6	The client's gender	From dropdown menu, so none required
BloodType	{A+/A-/B+/B-/ O+/O-/AB+/AB-}	TEXT	3	The client's blood type	From dropdown menu, so none required
Height	0.00	FLOAT	4	The client's height	Digit, value (more than 1 less than 2.5), presence
Mass	000.00	FLOAT	6	The client's mass	Digit, value (more than 30, less than 500), presence

Finances:

Field name	Default value	Data type	Field length	Description	Validation
TransactionID	0	INTEGER	-	Primary key, unique identifier for the table	Automatically created, none required
Difference	{+/- 000.00}	INTEGER	-	The difference of the transaction	Format, presence
DateAndTime	{YYYY-MM-DD HH:MM:SS}	TEXT	19	The date and time of the transaction	Presence, format
TransactionStatus	{Success/Fail}	TEXT	7	The status of the transaction	Automatically created, none required

Required files and data structures

Aim

I will be adding four text files to the program. These text files will be transactions.txt, transactionsbydate.txt, appointments.txt, and medicalrecords.txt. All the tables will remain in the clinic.db file. The entire program will still be in third normalized form, I feel I can follow the requirements that I made in the design stage which is:

3NF

CLIENTS(ClientID, *MedicalRecordID*, Prefix, FirstName, Surname, DOB, Telephone, Address)

STAFF(StaffID, Prefix, FirstName, Surname, DOB, Telephone, Address, Position, Username, Password, LoggedIn)

TRANSACTIONS(TransactionID, Difference, DateAndTime, TransactionStatus)

APPOINTMENTS(AppointmentID, *StaffID*, *TransactionID*, *MedicalRecordID*, StartDateAndTime, EndDateAndTime, AppointmentStatus)

MEDICALRECORDS(MedicalRecordID, *ClientID*, Sex, Gender, Height, Mass)

Additional processes and relationships between the data and the processes

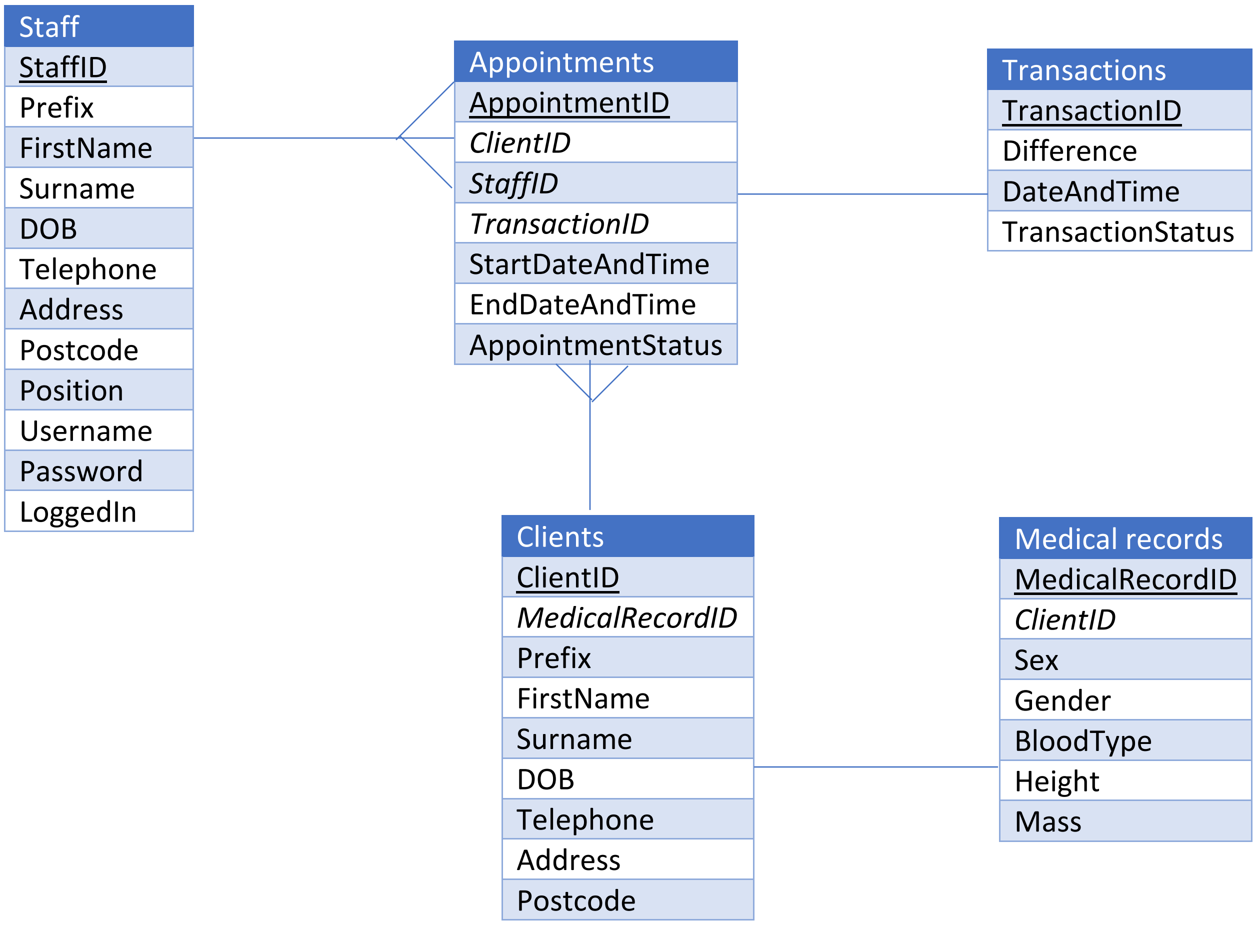
Aim

I aim to look at the new processes I will add to my prototype and see how the relationships between data will change. Currently the relationships between the clients and staff table is non-existent and I intend to change this.

Processes

- The system must be able to store a list of medical records
- The system must be able to store a list of transactions
- The system must be able to store a list of appointments
- The system must be able to validate all data added to the program
- The system must be able to calculate the BMI
- The system must be able to calculate the total sum of transactions in a day

Relationships



Processing routines

Aim

I will add a number of new functionality to the protype to be finally added to the program, this will all be written in pseudocode as I believe pseudo code is an appropriate convention that gives enough detail for a third party to understand it.

Functionality

Editing a record

UPDATE table SET column = column_box WHERE primary key = dropdown_menu

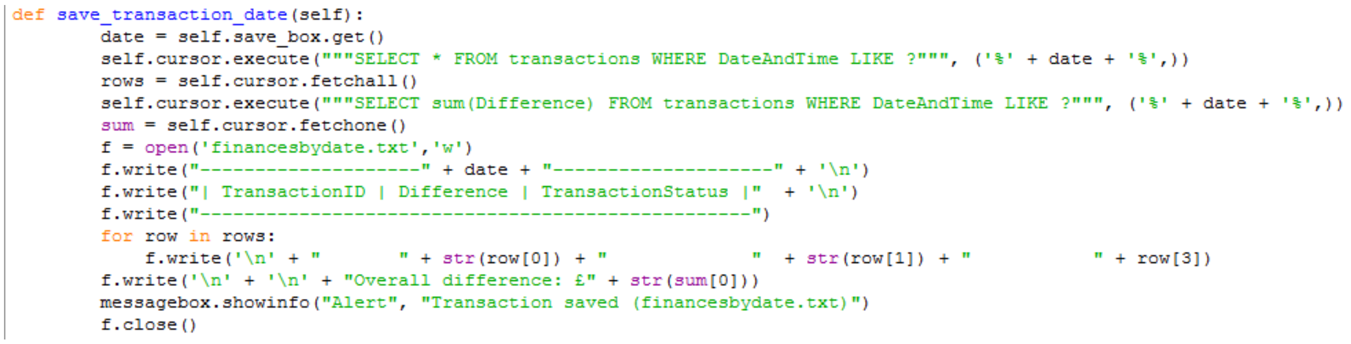
update_treeview

```
clientvariable = self.clientvariable.get()
clientvariable = (ast.literal_eval(clientvariable)[0])  # converts to tuple
self.cursor.execute("""UPDATE clients SET Prefix = ? WHERE ClientID = ?""",
                                (self.variable.get(), clientvariable,))
self.db.commit()
ClientsFrame.update_table(self)
```

Candidate Number:
3097

Saving a transaction as a text file

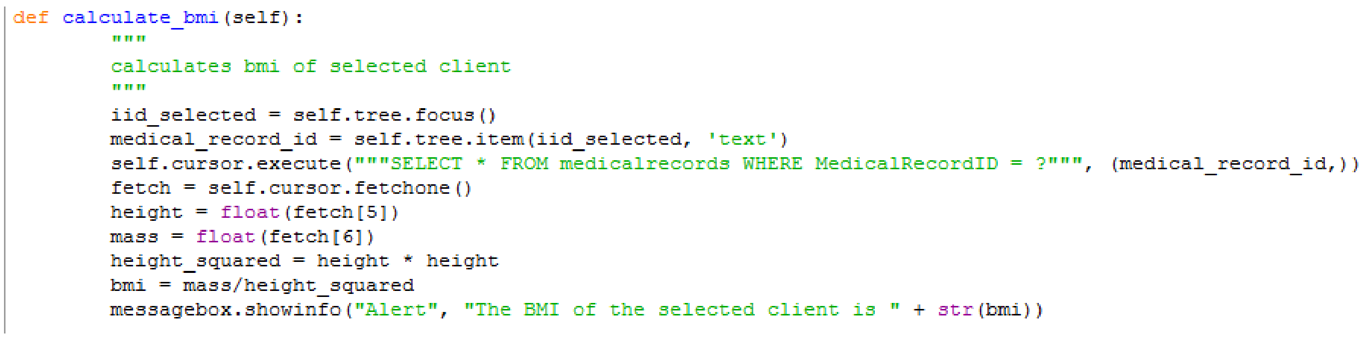
```
SELECT ALL from table WHERE column = input_box

SELECT SUM of value

FOR row IN rows:

    WRITE TO TEXT FILE (row[0] + row[1] … + row[n-1] + row[n])

WRITE SUM of value

OUTPUT record saved
```



```
def save_transaction_date(self):
        date = self.save_box.get()
        self.cursor.execute("""SELECT * FROM transactions WHERE DateAndTime LIKE ?""", ('%' + date + '%',))
        rows = self.cursor.fetchall()
        self.cursor.execute("""SELECT sum(Difference) FROM transactions WHERE DateAndTime LIKE ?""", ('%' + date + '%',))
        sum = self.cursor.fetchone()
        f = open('financesbydate.txt','w')
        f.write("--------------------" + date + "--------------------" + '\n')
        f.write("| TransactionID | Difference | TransactionStatus |"  + '\n')
        f.write("--------------------------------------------------")
        for row in rows:
            f.write('\n' + "       " + str(row[0]) + "             "  + str(row[1]) + "           " + row[3])
        f.write('\n' + '\n' + "Overall difference: £" + str(sum[0]))
        messagebox.showinfo("Alert", "Transaction saved (financesbydate.txt)")
        f.close()
```

Finding the BMI of a medical record

```
INPUT highlighted box

GET primary key FROM highlighted box

SELECT mass, height FROM WHERE primary key = X

BMI = mass/height^2

OUTPUT BMI
```



```
def calculate_bmi(self):
        """
        calculates bmi of selected client
        """
        iid_selected = self.tree.focus()
        medical_record_id = self.tree.item(iid_selected, 'text')
        self.cursor.execute("""SELECT * FROM medicalrecords WHERE MedicalRecordID = ?""", (medical_record_id,))
        fetch = self.cursor.fetchone()
        height = float(fetch[5])
        mass = float(fetch[6])
        height_squared = height * height
        bmi = mass/height_squared
        messagebox.showinfo("Alert", "The BMI of the selected client is " + str(bmi))
```

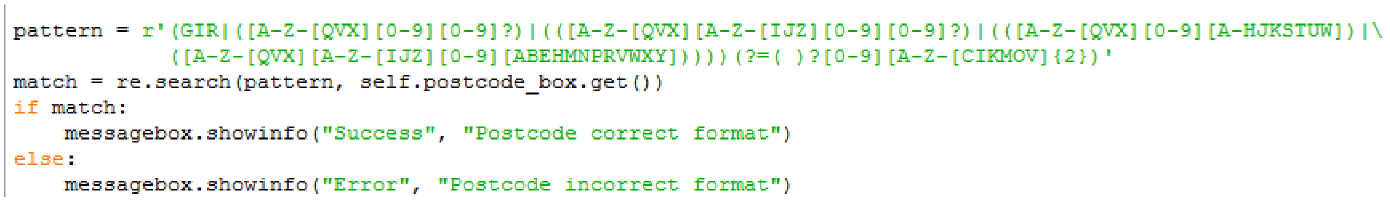
Candidate Number:
3097

Validation

Format check

Postcode format

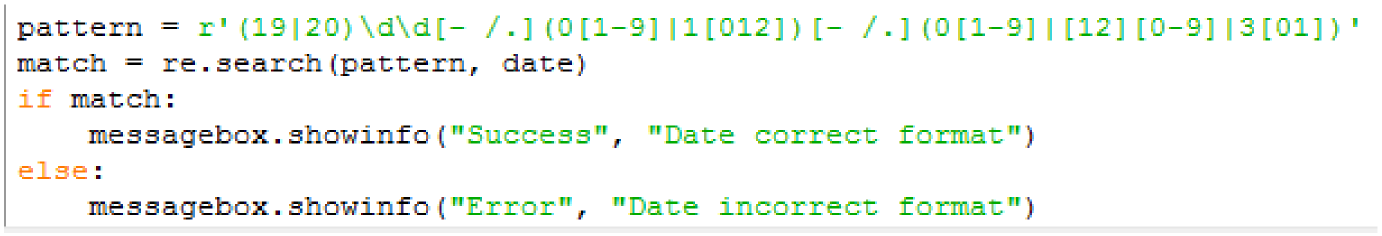
```
INPUT postcode

match = [AA9A 9AA] OR [A9A 9AA] OR [A9 9AA] OR [A99 9AA] OR [AA9 9AA] OR [AA99 9AA]

IF match(postcode):

        return TRUE

ELSE:

        OUTPUT error

        return FALSE

ENDIF
```



```python
pattern = r'(GIR|([A-Z-[QVX][0-9][0-9]?)|(([A-Z-[QVX][A-Z-[IJZ][0-9][0-9]?)|(([A-Z-[QVX][0-9][A-HJKSTUW])|\
            ([A-Z-[QVX][A-Z-[IJZ][0-9][ABEHMNPRVWXY])))))(?=( )?[0-9][A-Z-[CIKMOV]{2})'
match = re.search(pattern, self.postcode_box.get())
if match:
    messagebox.showinfo("Success", "Postcode correct format")
else:
    messagebox.showinfo("Error", "Postcode incorrect format")
```

Date format

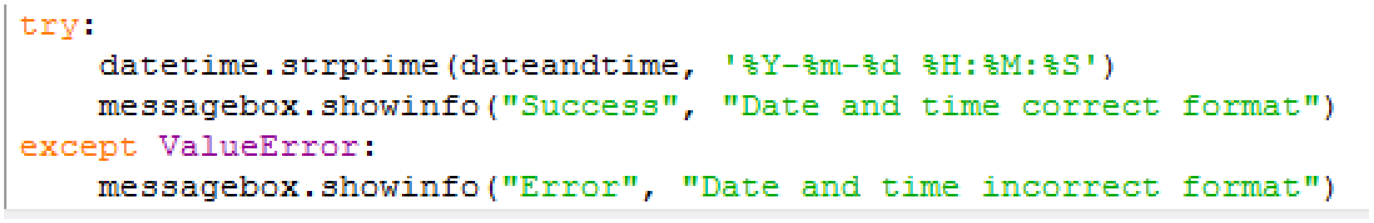
```
INPUT date

match = [YYYY-MM-DD]

IF match(date):

        return TRUE

ELSE:

        OUTPUT error

        return FALSE

ENDIF
```



```python
pattern = r'(19|20)\d\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])'
match = re.search(pattern, date)
if match:
    messagebox.showinfo("Success", "Date correct format")
else:
    messagebox.showinfo("Error", "Date incorrect format")
```

Date and time format

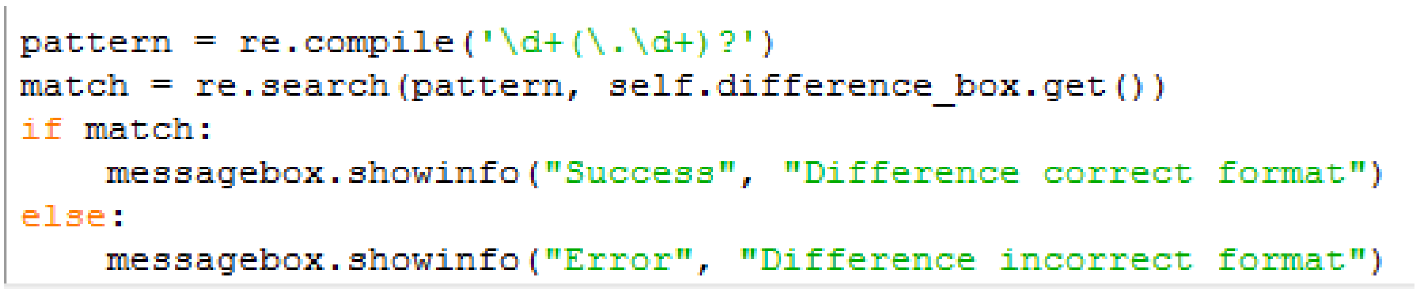
```
INPUT dateandtime

match = [YYYY-MM-DD HH:MM:SS]

IF match(dateandtime):

        return TRUE

ELSE:

        OUTPUT error

        return FALSE

ENDIF
```



```
try:
    datetime.strptime(dateandtime, '%Y-%m-%d %H:%M:%S')
    messagebox.showinfo("Success", "Date and time correct format")
except ValueError:
    messagebox.showinfo("Error", "Date and time incorrect format")
```

Difference format

```
INPUT difference

match = [+/-] + [DD]

IF match(difference):

        return TRUE

ELSE:

        OUTPUT error

        return FALSE

ENDIF
```



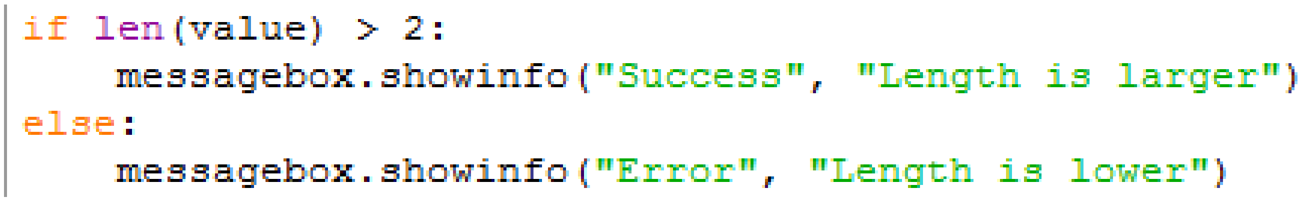
```
pattern = re.compile('\d+(\.\d+)?')
match = re.search(pattern, self.difference_box.get())
if match:
    messagebox.showinfo("Success", "Difference correct format")
else:
    messagebox.showinfo("Error", "Difference incorrect format")
```

Candidate Number:
3097

Length check

More than length check:

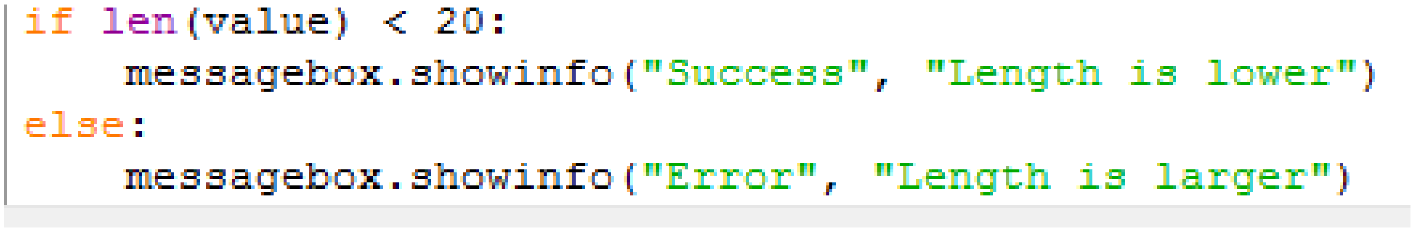
```
INPUT result
IF len(result) < max:
    return TRUE
ELSE:
    OUTPUT error
    return FALSE
ENDIF
```



```
if len(value) > 2:
    messagebox.showinfo("Success", "Length is larger")
else:
    messagebox.showinfo("Error", "Length is lower")
```

Less than length check:

```
INPUT result
IF len(result) > min:
    return TRUE
ELSE:
    OUTPUT error
    return FALSE
ENDIF
```

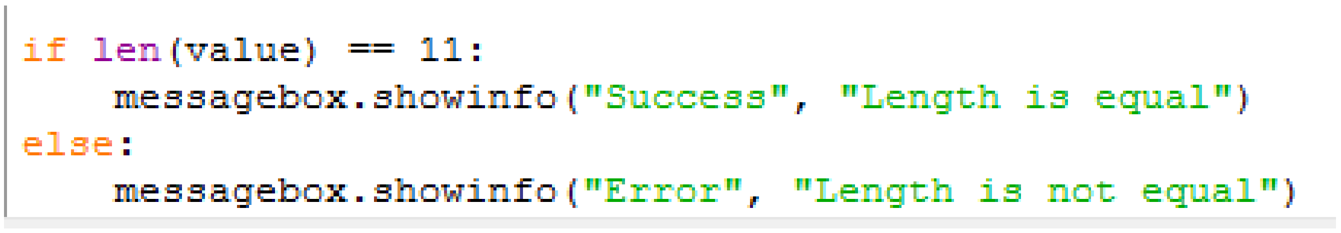


```
if len(value) < 20:
    messagebox.showinfo("Success", "Length is lower")
else:
    messagebox.showinfo("Error", "Length is larger")
```

Candidate Number:
3097

Exact length check:

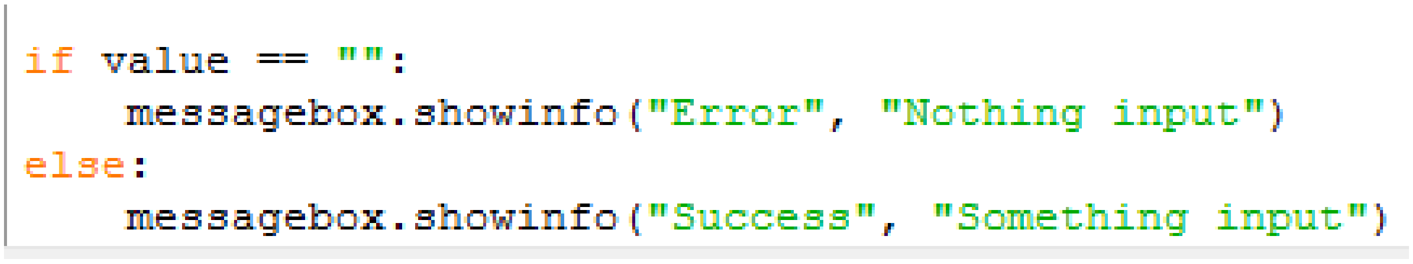
```
INPUT result

IF len(result) = exact:

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```



```python
if len(value) == 11:
    messagebox.showinfo("Success", "Length is equal")
else:
    messagebox.showinfo("Error", "Length is not equal")
```

Presence check

```
INPUT result

IF result = "":

    OUTPUT error

    return FALSE

ELSE:

    return TRUE

ENDIF
```



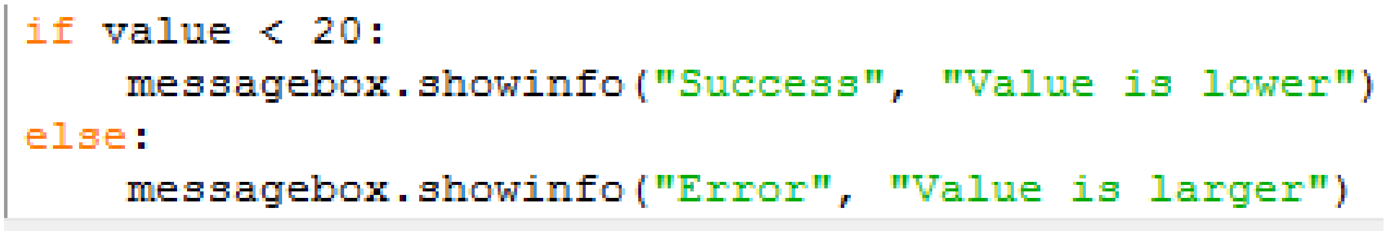
```python
if value == "":
    messagebox.showinfo("Error", "Nothing input")
else:
    messagebox.showinfo("Success", "Something input")
```

Candidate Number:
3097

Range check

Larger than range check:

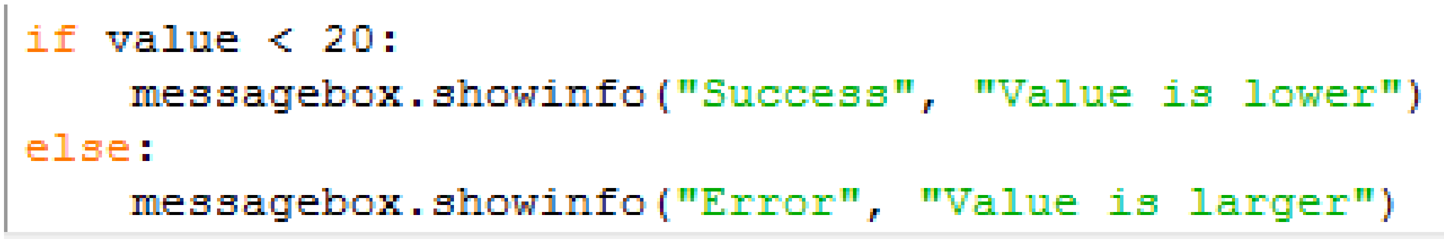
```
INPUT result

IF result < max:

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```



```python
if value < 20:
    messagebox.showinfo("Success", "Value is lower")
else:
    messagebox.showinfo("Error", "Value is larger")
```

Less than range check:

```
INPUT result

IF result > min:

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```



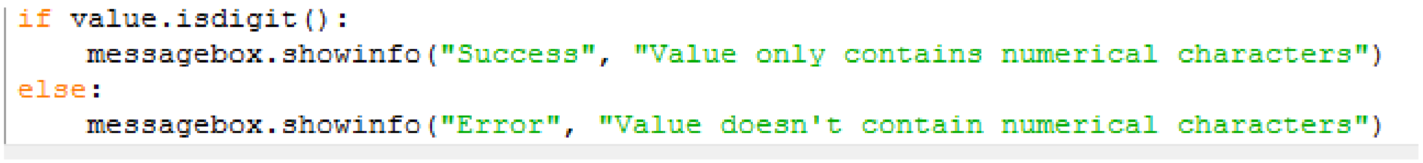
```python
if value < 20:
    messagebox.showinfo("Success", "Value is lower")
else:
    messagebox.showinfo("Error", "Value is larger")
```

Candidate Number:
3097

Type check:

Integer check

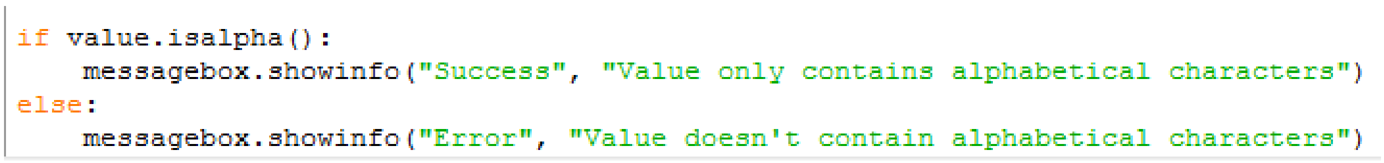
```
INPUT result

IF result.isinteger():

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```



```python
if value.isdigit():
    messagebox.showinfo("Success", "Value only contains numerical characters")
else:
    messagebox.showinfo("Error", "Value doesn't contain numerical characters")
```

Alphabetic character check

```
INPUT result

IF result.isalpha():

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```

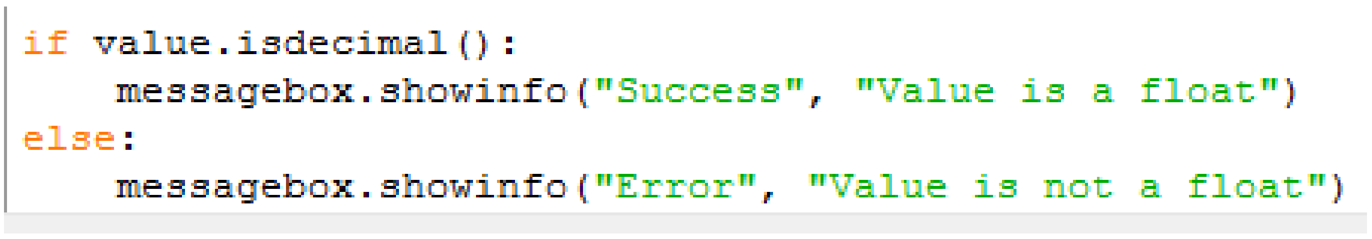


```python
if value.isalpha():
    messagebox.showinfo("Success", "Value only contains alphabetical characters")
else:
    messagebox.showinfo("Error", "Value doesn't contain alphabetical characters")
```

Candidate Number:
3097

Float check

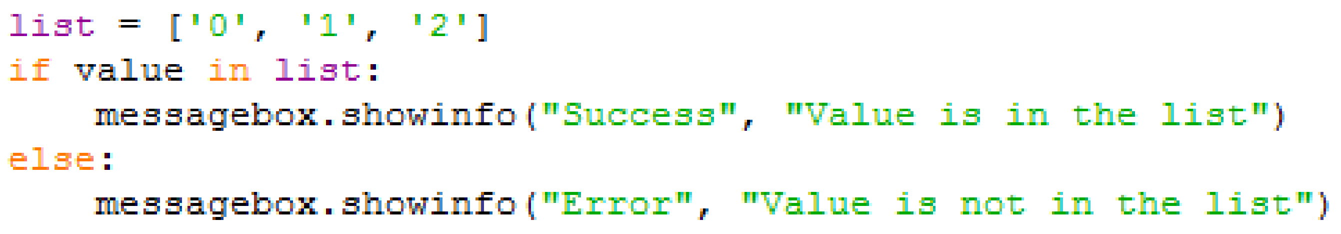
```
INPUT result

IF result.isreal():

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```



```python
if value.isdecimal():
    messagebox.showinfo("Success", "Value is a float")
else:
    messagebox.showinfo("Error", "Value is not a float")
```

Lookup check:

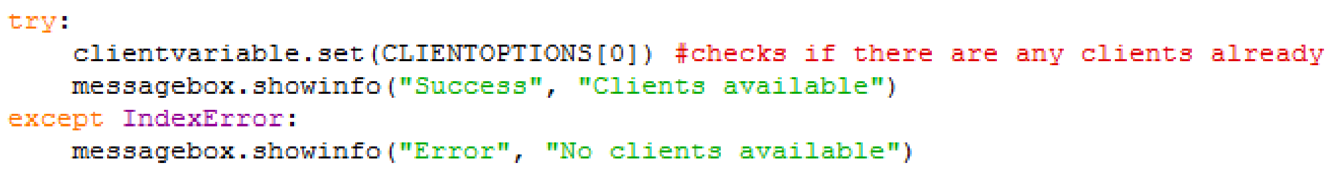
```
INPUT result

list = [0, 1, 2, n-1, n]

IF result IN list:

    return TRUE

ELSE:

    OUTPUT error

    return FALSE

ENDIF
```



```python
list = ['0', '1', '2']
if value in list:
    messagebox.showinfo("Success", "Value is in the list")
else:
    messagebox.showinfo("Error", "Value is not in the list")
```

Editing without any values:

```
SELECT ID FROM table
TRY:
    cursor.fetchone()
EXCEPT IndexError:
    return FALSE
    OUTPUT error
```



```
try:
    clientvariable.set(CLIENTOPTIONS[0]) #checks if there are any clients already
    messagebox.showinfo("Success", "Clients available")
except IndexError:
    messagebox.showinfo("Error", "No clients available")
```

Success criteria

After looking through the success criteria that I have created I believe that no changes should be made to my program. The prototype feedback that I have received was strong and I believe that completing the remaining success criteria is possible without having to discount any of my aims. I won't however add any new aims as I feel like I wouldn't be able to complete any more in the time I have been given for this project.