4 – Prototype

Contents

4.1 Desired Aspects for Prototype

As my prototype is to reflect my final system and is to present key stakeholders its possibility I will aim to include as much as I can to show its full potential, but with that being said, as it is a prototype it should hence not be a fully complete system with areas not needed not being present. During this time, I should also be able to experiment with ideas and at the end get an idea what to include and exclude for the final system. To determine suitability for my objectives of my system I will follow a few basic rules to ensuring features that do not need to be included are not present to maximise development time on features I must have included. With that being said I have laid out a few rules to what can be included:

- The objective should be vital to operations of the system
- No feature that is already present should be included again; similar objectives should be omitted unless it impedes operation of the main system
- Overall the system should be able to complete the core aspect of the end product that being: create and store documents for patients

Objectives to be included and omitted

Objective # (Investigation)	Feature	Included / Omitted	Justification
1	**Input/select symptoms into expert system**		This feature will be implemented on the prototype because it will be an important feature on the system, due to it being one of the key features for generating new admissions as the data will need to be retrieved in some manner, disregarding it now in prototype would also result in no reason to also include the basic diagnosis generation. It is important as it will help reduce the need of predetermination of ailment as the symptoms have been recorded and could allow for the patient to feel more comfortable explaining any symptoms they are experiencing as sometimes telling another person may be uncomfortable for them.
2	**Determine suggestion for user**		While I definitely want this aspect to be included on the final version of the system, currently I do not want this feature on the prototype for many reasons. The main reason is that the development for it will take up more of my focus than from other areas and while a basic one could be developed I believe that the main system should be finished and that a feature like this could be really focused on for the final version of the system. When using the prototype if a user wants to create an admission the staff entity is not present in the prototype for the time being all conditions or diagnosis will be pre-set with dummy data.

Objective # (Investigation)	Feature	Included / Omitted	Justification
3	**Generating new patients**		This feature will be developed on the prototype because it is a key feature of the staff entity. This will also allow for the staff entity to be evaluated at a later date as they will contain the majority of features expected to be used in the actual system. However, to test adding the new entity onto the system the need for unique fields is minimal so test data will be used in the majority of fields. One area this will be beneficial for will be the PatientID section as this can allow for the algorithm to determine the ID of the patient to be tested, however as the name of the patient will be used throughout many entities this is where the number at the end of the ID will be important as it will have to make the ID unique by generating a number at the end that has not been used.
4	**Generating a new admission**		This feature will be included onto the system as again it is an essential process for the system as it will be needed for adding entities onto the system. As the system will revolving around patients' admissions it is critical that the ability to create new admissions should be included. This should be available to both the patient and consultant entity as both are required to fully complete the admission.
5	**Booking a new appointment**		I believe this feature and all other related features regarding the booking and using of appointments should be absent from the system, this is because it will not be the main draw to the system that bookings can be created but more of a benefit. As the main focus is to predominantly be for document management I feel that this should be focused on during development of the system, as the need for a booking system can be revaluated at the end of the process and see if it is overall worth the development time.
6	**login**		I intend to now include this feature to the system. This is due to the fact that in order for a desired patient account to be accessed one will need to be retrieved and in order for that to occur an account needs to be chosen by entering their primary key on the system, however as it would be detrimental to fully implement the feature to the system it will only be included to allow the entry of the name. As for the password it will be included at a later date during development.

Objective # (Investigation)	Feature	Included / Omitted	Justification
7	**Display menu options**		It is obvious that this process is crucial for the system as it will be needed to test how the GUI will eventually turn out and if any adjustments need to be made from the original conceptions made in the design. It will also allow for an idea for how it will look and feel in the final system. **However, design and branding will not necessarily be prioritised in the prototype as functionality is the main aim.** As a prototype is to signify how it will look in the final version some features may appear to exist but may not be fully complete.
8	**Add employees**		The ability for the management entity to add employees onto the system is a feature I do not intend to include and will therefore omit from the prototype. This is because it will not have a large impact on the system and was only intended to be used occasionally on the system when new staff are brought onto the system, there is no reason for why this feature would be used on the prototype in respect to entity management as this will be similar to the ability to add patients onto the system.
9	**Archive employees**		This is similar to objective #8, as it is relating to an entity that the system is not centred around. Because of this I intend to omit this feature from the system, this is due to the process being unnecessary with respect to the main aim of the system which is document and admission management. In addition to this the feature set surrounding the management entity will be limited due to the size of the prototype as I do not want to promise a fully finished version for a prototype.
10	**Sort for employees**		The process of sorting will be used throughout the system many times, I feel a larger collection of entities, such as patients or admissions will be more beneficial as it will test the sorting algorithm better as it will work with more objects. Due to the fact only, a handful will be present for the time being it is absolutely pointless to include the feature as it will be present I other areas anyway where the sorting algorithm can be better shown to the user.

Objective # (Investigation)	Feature	Included / Omitted	Justification
11	**Search for employees**		As the user will need to find their desired employee in the action log I will have to add the ability to search for employees on the system. This will allow me to search through long lists and will also allow me to test how the system copes with dealing with retuning attributes of the management staff, while it will be predominantly used for the action log it could also be utilised for the log in system later on so having it work well now will be a major benefit to the system later on during development.
12	**View an employee's transaction log**		I intend to include this feature on the system as it will allow this experimental idea to be tested and see how it will work out in reality. This is important because this process will be used to record every action the employees do on the system. It will also incorporate other processes such as sorting and searching, so it is important to have the feature working early on. While the feature will be for an entity that is not a priority, I feel that this is a unique aspect of my system that will distinguish it from other EHRs so I need it to be fully tested and operational from the onset.
13	**Read/write transaction log from file**		This is another feature I intend to include on the system because it will be used to handle lots of data and allow for a comprehensive testing for how fields of data are wrote to file and subsequentially managed afterwards. However, I still intend for other data to be wrote to file despite other features being used in the prototype once. This is because there will be many files used throughout the system to retrieve data so file writing and reading will be needed a lot.
14	**Staff can sort for patients**		This feature will be omitted due to the fact that the same sorting algorithm will be used thoroughly in other areas such as searching for patient documentation. In addition, it is also used for a section deemed unnecessary to the main operations of the system as it will be for the staff entity to find patients. While it will be important for finding patients in the final version, as the majority of patients will have their fields filled in with dummy data it will be pointless to sort them if they will be majorly the same.

Objective # (Investigation)	Feature	Included / Omitted	Justification
15	**Staff can search for a patient**		Again, this feature will be omitted because the searching algorithm used for searching for patients will be used in other sections of the prototype. It will also be excluded due to the fact the entities it will be searching for will be mostly identical to the rest of the other entities, as the algorithm will return all entities that meet the search query it will be pointless as most of the patients on the system will be shown, resulting in the same issue as earlier.
16	**Staff can View patient details**		This feature will be omitted from the system, this is because the prototyped system will not include any staff features are included and therefore would be an obvious decision to omit this feature. The main reason the entity is to be removed is that it is the middleman between the consultant and patient; to do the tasks too repetitive for the consultant to free up their time but have levelled access of sensitive information like a patient.
17	**Add archived notes from old system**		The ability for the staff entity to add old files from the old system is a feature I do not intend on include and will therefore omit it in the prototype. This is because I will need to test methods for how old documents will be brought over to the new system which will take into other time spent on developing essential aspects of the system. At a later point similar to the booking system if more emphasis is needed on other features this can be included in the final version if need be.
18	**editing bookings**		I will omit this feature from my prototype due to it being unnecessary with the main aim of the system prototype which is to manage the admissions and documentation of the patient and making sure core functionality and processes are fully operational. As generating bookings will be excluded anyway it is clear that this would be omitted. However, while this will be missing from the prototype it will be a key aspect to include for the final version of the system.
19	**View patient bookings for staff**		I intend to exclude this feature as while it is important to show booking information to the user, this objective is to show non confidential booking information to the staff entity. While in the final system I will want this feature included I believe it to be unnecessary to be used in the prototype currently as it will be exactly the same as objective #24 but will show less information.

Objective # (Investigation)	Feature	Included / Omitted	Justification
20	**View patient admissions for staff**		Again, this is a similar situation to objective #19 as the aim of this feature is to show non confidential information regarding the admission to the staff entity. Because of this I will omit this from the prototype because it is similar to objective #21 but will show only non-confidential information this will reduce the wasting of resources on repeated features which could be spent on more pressing features of the prototype.
21	**Patients view their admissions and Demographic information**		This feature is required in the prototype as it is a key aspect of the system to allow the patient to view both the admission and demographic information regarding their account. This needs to be included as it will allow the ability for fields to update and respond to changes regarding user input. Also, to add to this the layout of fields will also allow for feedback from the prototype in case the layout is too cluttered which can be readjusted later on.
22	**Patients can amend their demographic information**		This feature follows up from objective 21and links to it very closely. It allows information in the fields to be changed and manipulated from the consultants/patients input. Overall this feature is critical when using an EHR as some medical information is never constant like address etc. Like previously stated the prototype will allow for testing for how the system reacts to changes regarding patient fields.
23	**Validate information**		This is an objective that I feel that can be omitted from the system simply due to the fact the prototype is needed to test core aspects of the system. While validating information will be critical to preventing incorrect data entry and data inconsistency on the final build of the system currently for prototyping core features and aspects it is completely unneeded and irrelevant. As only I will be entering data so can therefore moderate data entry to what I deem necessary to include. However, to look out for during prototype feedback is to make sure users testing the system do not input any invalid data.
24	**Consultants can View bookings**		On the system the ability to utilise bookings is completely unnecessary during the point of the development of the prototype as this version will be used to create a system in which all the features around the use of documents is flawless the need for the ability to add documents seems minor I comparison because of this, the ability to view appointments will be omitted, while the ability to see the closest appointment for the consultant and patient will be included nothing beyond this will be seen on this build of the system.

25	**Reschedule/ amend bookings**		This is a feature of the system I intend to omit this is on the basis that the booking feature is a non-vital necessity to the systems operations and the inclusion of this would detriment development the system as resources would be wasted for a partially complete function. The importance of this feature will also be validated at a later point to determine whether or not its inclusion is important enough to include in the final version.
26 A	**Sort documents**		This is an aspect of the system that I believe is important to the extent it needs to be included on the system. This is because the number of documents will contain many documents and therefore can test the efficiency of the sorting algorithm. By including this in the prototype this will allow other users when evaluating the system to see how well the sort works. Later on, this will be evaluated to see whether it needs to be improved or kept the same for the final version of the system.
26 B	**search documents**		This is an aspect of the system that I believe is important to the extent it needs to be included on the system. This is because the number of admissions will contain many documents and therefore can test the efficiency of the searching algorithm. By including this in the prototype this will allow other users when evaluating the system to see how well the search works it will also allow the chance to determine if a more powerful algorithm is to be utilised or whether it can be kept the same.
27	**Print documents**		This an aspect of the system I intend on omitting. This is clear because it has no relevance with the main aim being to digitally display information to the patient. While I feel it is an aspect that will be benefited in the final version, currently it has no need to be included. This can be omitted in addition to this as it will reduce time for development of other key aspects of the prototype. On looking into this feature, it seems that the implementation will not take too long but this can still be done later down the line.
28	**Consultant can Search for patients**		This is an aspect of the system I intend on including on the prototype, this is down to the fact that it will make the consultant part of the system more complete and feature full. While the search algorithm is going to be utilised in other areas I feel that this one will be important as it will be used often and will be helpful towards showing areas of the system that otherwise would not get too much attention at this stage of development and as the system will only be having a handful of consultants each entity will be linked to many patients so it would be useful to include this feature.

Objective # (Investigation)	Feature	Included / Omitted	Justification
29	**Consultant can view patient accounts**		This feature will be included on the prototype. This will be because it will a key feature in allowing employees to view their patient's data in general. Also, to add to this it will allow for the reading features to also be tested when extracting information from specific areas in text files. Finally, it will allow myself to evaluate the process of navigating to specific areas, because of this it will be included.
30	**View patient Demographic information**		This will be a feature I intend on including from the prototype of my system, this is due to the fact that I will need to see how I will approach gathering data from files and the process of encrypting and reading that comes with it. To add to this, I am also including Objective 29 with regards to viewing file information. Because of this I feel it would be a waste of time including the ability to find and output the file to then not include the ability to view an individual subsection of that file.
31	**Sort admission**		This is a feature that will be included on the system due to the fact the patient may have many admissions and the ability to sort them will be an important feature to include and could allow for testing and evaluation at the end of the prototype stage of development. It may also be important due the fact the number of admissions could have fields that could be used to allow for options for sorting orders
32	**Edit Prescriptions from the document end of the system**		Again, this is another feature I intend on including this on the system. This is because it will be attached to both the document and admission parts of the system, to allow for both ends of this feature to change is a chance to develop and test how linking aspects of the system together can work and how that eventually the final version might have more components which utilise this feature to reduce the need of going through panels to see a single item, for instance a favourites tab comes to mind.
33	**Add Admission information**		This is another feature I intend on including on the system as it is one of the main aspects of the system, this will allow myself to evaluate how well the current system deals with adding new lines of data to specific places in the text file. In addition to this it will allow myself to also evaluate the ability to add new admissions and see if any tweaks need to be made or if the layout needs to be readjusted. Because of all of this I intend on keeping this aspect within the prototype of the system.

Objective # (Investigation)	Feature	Included / Omitted	Justification
34	**Edit Admission information**		This is another aspect of the system I intend on including, due to the fact it is another main feature of the system. Even more, like other features this will incorporate other key objectives such as 36 and 37, however I do not intend to validate most fields so the development time can be reduced and other features can be focused on. It will also allow for the end user to get an early feeling on how data entry feels and the ability to change the input overtime. However, as this will be the prototype I don't not intend on fully including unique data for every entry and will utilise "dummy data" to reduce this process. Any fields without this will be ones I have used this objective to test this feature.
35	**Add notes**		This is another feature I intend on including on the system, this is the main feature of the system and will therefore have to be included on the prototype. Because it is such an important feature it will allow myself to evaluate the feature afterwards and see how others feel about the ability to add documents onto the system. To add to this as I will include the ability to search and sort documents, I feel to help test these features having the ability to casually add new documents can help test this.
36	**Encrypting data before being written to file**		This is another feature I intend on including on the prototype for the system. This is because I will be using it a lot during the final system to write everything to file, because of this to have the ability to perfect this will be a major advantage and will greatly reduce the effort of deciding the cypher to include in the final build. Even more it will allow myself to experiment while I have the time to see which encryption method suits the system best. While I will need to view the data as unscrambled during development I will make sure the
37	**Decrypting that has been read from file**		Obviously coming from objective 36 I will therefore have to include the ability to decrypt any data for my prototype. This will also allow myself to see how efficient the method of decryption is and will also allow myself to test how effective I can get the function to work before actually including it in the finished product. While in other cases including these features will be a waste of time, as the system is focused on data protection I aim to perfect this aspect.

Objective # (Investigation)	Feature	Included / Omitted	Justification
38	**Using the Jargon library**		This is another aspect I intend on including on the main system. This will allow myself to develop and experiment on a feature I deem to be not a main part of the system. Also, it will allow myself to evaluate its usefulness and determine whether or not if it is needed overall. In addition, it will help determine how well others will receive the feature and can help see if it needed at all. The feature will be important to help see how well the system deals with reading specific files from the large file of definitions.
39	**Adding to the Jargon library**		While I am including objective 38 onto the prototype I feel that the ability to add new definitions onto the system is feature that will not be needed. This will be down to the fact the processes including will be present throughout other parts of the system. In addition to this I feel that the ability to set up new definitions is too much of a minor aspect that it will waste time for development for the more major features.
40	**Search through Bookings**		This is another feature I intend on not including on the prototype. This will be because the ability to view bookings has been already omitted from the system and therefore would be pointless to have it included However as most searches will be done by the same algorithm other areas can be utilised to stress test how the system copes with searching through indexes instead of this one.
41	**Search through demographic information**		Finally, this is the last aspect of the system which will make its way over to the prototype. This again is important to include as it will be needed to make sure that the correct demographic is read from the file for key objectives like 22 which allow the demographic to be edited.

Other features to be excluded

While I went into detail about specific objectives and justified their inclusion or not above, here I want to go into detail about some of the larger features that will be limited or fully excluded on the prototype. This is because I may have justified that an objective did not merit an inclusion on the fact that it would hold no benefit to have on the system, here these sections will give a definitive reason why I made those decisions about certain objectives.

The Staff entity

The decision to exclude such a large entity was not a last-minute effort to reduce workload in the development of the system but was a result of the number of objectives relating to the entity being excluded in the prototype due to the lack of need during this stage of development. This overall culminated in the fact that the staff entity would be pretty much useless in this build of the system, and the most sensible choice would be to exclude the entity in its entirety instead of including the ability to log in and view basic information, all of which are present in the majority of other entities of the system. The main purpose of this entity is to do the more boring work and tasks that would be time consuming for the consultant, such as scanning in pdfs of old documents(a feature that was omitted in the prototype) and while still being a critical aspect of Euxton today, as the build is in its prototype stage of development I mainly want the core aspect about creating new documents and having the ability to read them. Because of this I feel that the staff entity has no place on the prototype as of this moment.

Login system

As currently non-sensitive data will be the only type of data on the system I will omit this feature solely on the basis that there is no need to protect any data at this current instance, while the ability to encrypt data and decrypt it are currently included on the prototype; these are to implement them at an early stage so that once they need to start working they can be. However, that being said, the system currently needs to retrieve information from file and the only way to correctly identify that the data is for the intended patient is to input the desired patient id, because of this the user has to input some credentials in a login system style manner. However, nothing has been done in the way of authentication and therefore the login system actually has not been developed. What actually happens is that the user enters the id of the desired account to log in and then that text field is passed into the method which returns the user's data. In the final system in between pressing the login button and retrieving the account information I will insert the method which will compare the credentials entered by the user against those in the database, once accepted the information will then be retrieved.

File Organisation

As stated in my design document I had planned to include all various types of file organisation such as hashing algorithms and indexed sequential files. However, that has been omitted during the prototype of the system. While it would be perfect to have currently these methods of organisation been omitted due to the fact that they and hold no benefit right now for their use and take up more time in development over a simpler method. Despite this I have predominantly used sequential file organisation throughout the prototype as most of my searches are binary and require the list to be in order, while I could just order the list afterwards using a sort algorithm I feel that this would be best as I am not having to sort the list every time it is read from file. The file that does utilise serial file organisation however are for instances where all the records are needed from the file and can be sorted using a sort box in the form of a JCombo box such as the file containing all the consultant information or the action log for employees.

Having more than 5 admissions

Another feature to be omitted is having a larger catalogue of admissions, while in the final system the patient should theoretically have a limitless number of admissions, currently the patient is limited to having 5 admissions. This is because the admission hot bar only has 6 buttons attached formatted to fit on it. However, after looking into it I think I have a way to work around this. Either do what I normally do for this situation and have a button to navigate and reuse assets to format the button or create a horizontal list and attach a JScrollPane and create an array of buttons and just keep adding it to it whenever a new admission is created.

4.2 User Interface and outputs

As the prototype stage of development has been completed here I want to show the end product of my work and show how it looks through a collection of screen shots of how the panels look and a description of each panel's functionality along with any outputs of the system either anything that is written to file or displayed to the user. While it is still a prototype the features shown are not set in stone as some aspects may change and move around. The last thing to note is that not all features of the system are included so anything that looks incomplete is a subsidiary affect of the prototype development and will be completed in the final version.

Objective designed Screens

Home Screen

Outputs

On loading of the system all consultants are read from file with their corresponding patients. For example, the following information are fields are what are read for a specific consultant:

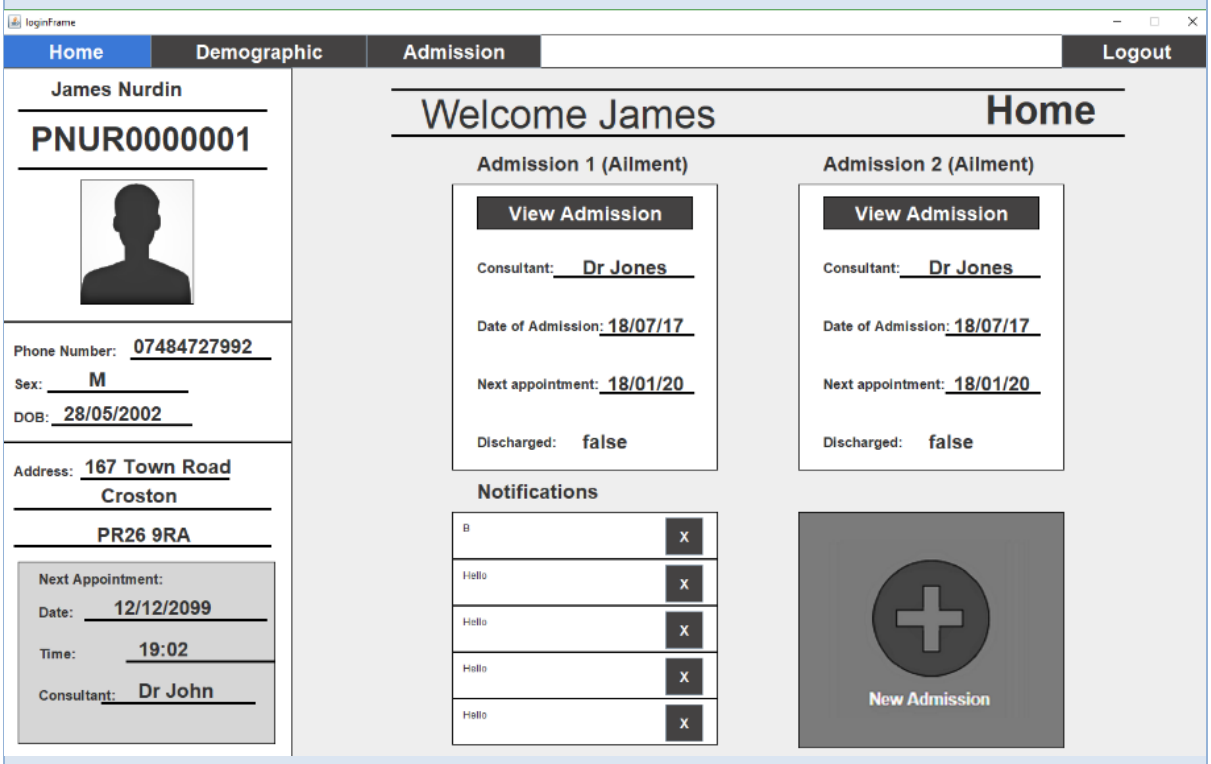
consultantID,employeeID,surName,firstName,wardLocatedIn,expertiese,wage,hoursPerWeek,archived,numberOfPatients,addressHouseNum,addressHouseStreet,town,postcode,contactNum,listOfPatientsAdmissions

Features

The options are twofold they can either login through the 4 standard user types, or they can create an admission and account on the system. The whole point is to be minimalistic and try to give the user the bare essentials as options for the first panel not to overwhelm them.

When they logout they will also be brought to the display in both the graphical and text line instances.

Patient home screen



Output

Updates the array of notifications in this case what is amended would be the notifications :

PNUR0000001,B,Hello,Hello,Hello,Hello,Hello,Hello,Hello,a,

Obviously the example is dummy data but this shows what it will be like in the final system this consists of the which user followed by the notifications data can be added in a serial format as it will be in date order

Features

The first thing the patient sees when they log in is this. They can swap between this panel and the demographic and admission panels (See other screens). They can logout(Common throughout system). Unique to the page they can:

1.Remove notifications in any order from the list

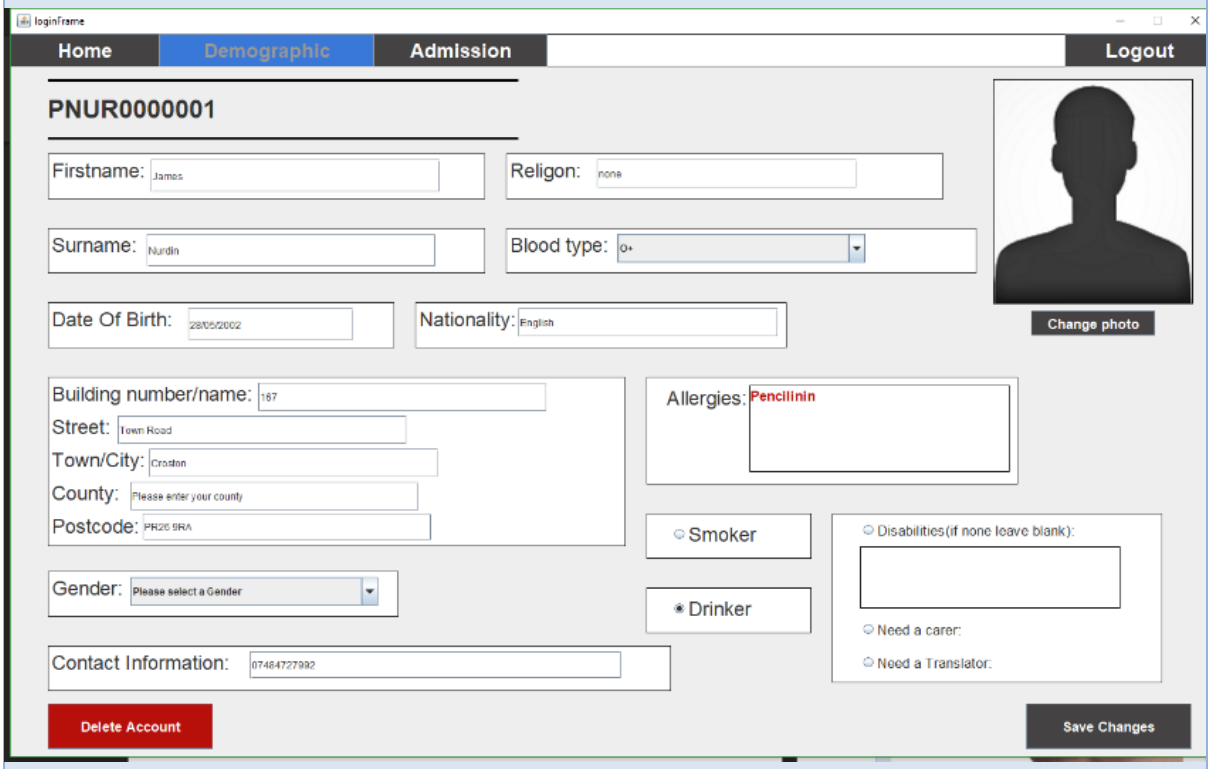
2.Create a new Admission

3.Read their basic demographic information

4.View their most recent admissions quickly(Not yet included just visually added for time being).

Overall the screen is supposed to be inviting to the user when they login, while it displays some amount of information it does not overwhelm them.

Demographic Screen



Output

Writes the following information to file:

PNUR0000001,Nurdin,James,167,Town Road ,Croston,PR269RA,07484727992,English,O+,false,true,5,28/05/2002,none ,Pencilinin,M,,false,false

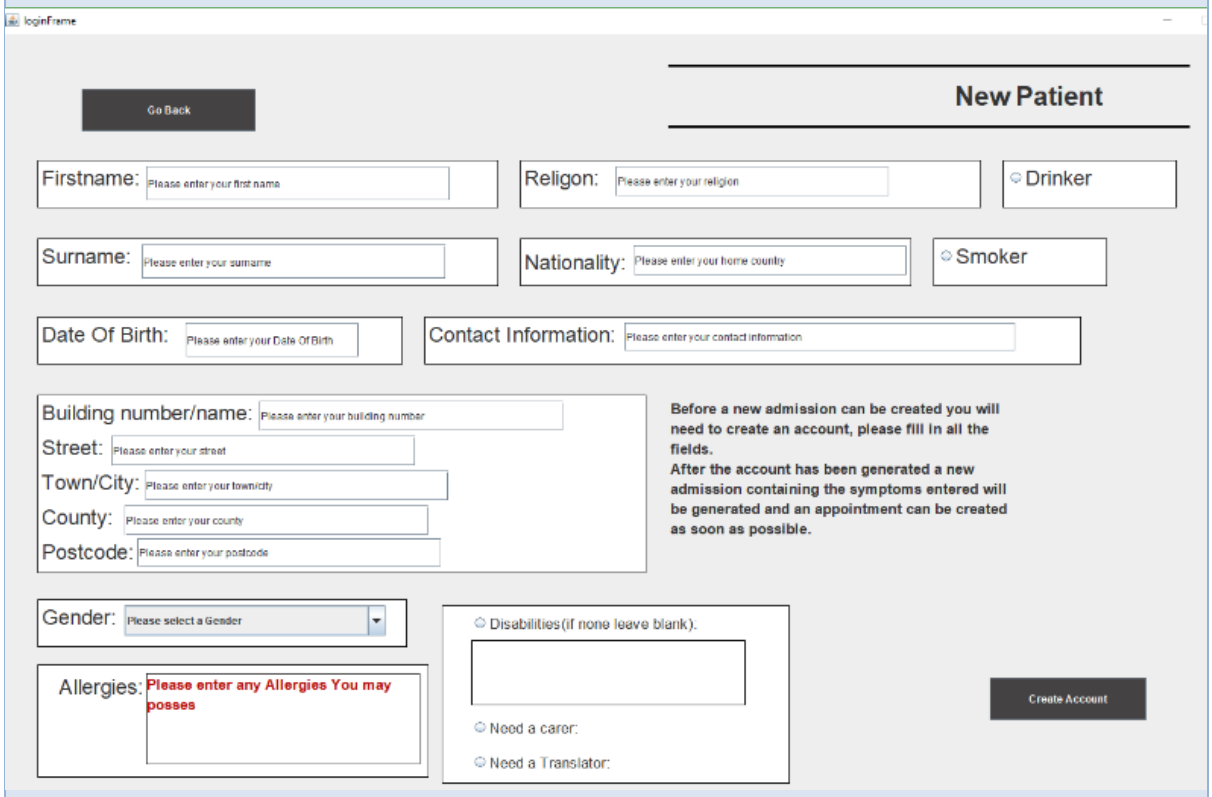
This following information will either be the same or will be new information in areas that are blank a double comma indicates an empty field

Features

The main features of this screen are twofold: view their details and update them.

1. The patient is able to enter into and amend the data in the fields presented to them this could be in the form of a text field or a combo box from editing this they can save the information by pressing the save changes button. This re writes any existing data and will overwrite any old ones by replacing the field with whatever is in the area
2. They can also read the information about them.

New patient screen



Output

Writes the following information to file:

PNUR0000001,Nurdin,James,167,Town Road ,Croston ,PR269RA,07484727992,English,O+,false,true, 5,28/05/2002,none,Pencilinin,M,,false,false

This following information will either be the same or will be new information in areas that are blank a double comma indicates an empty field

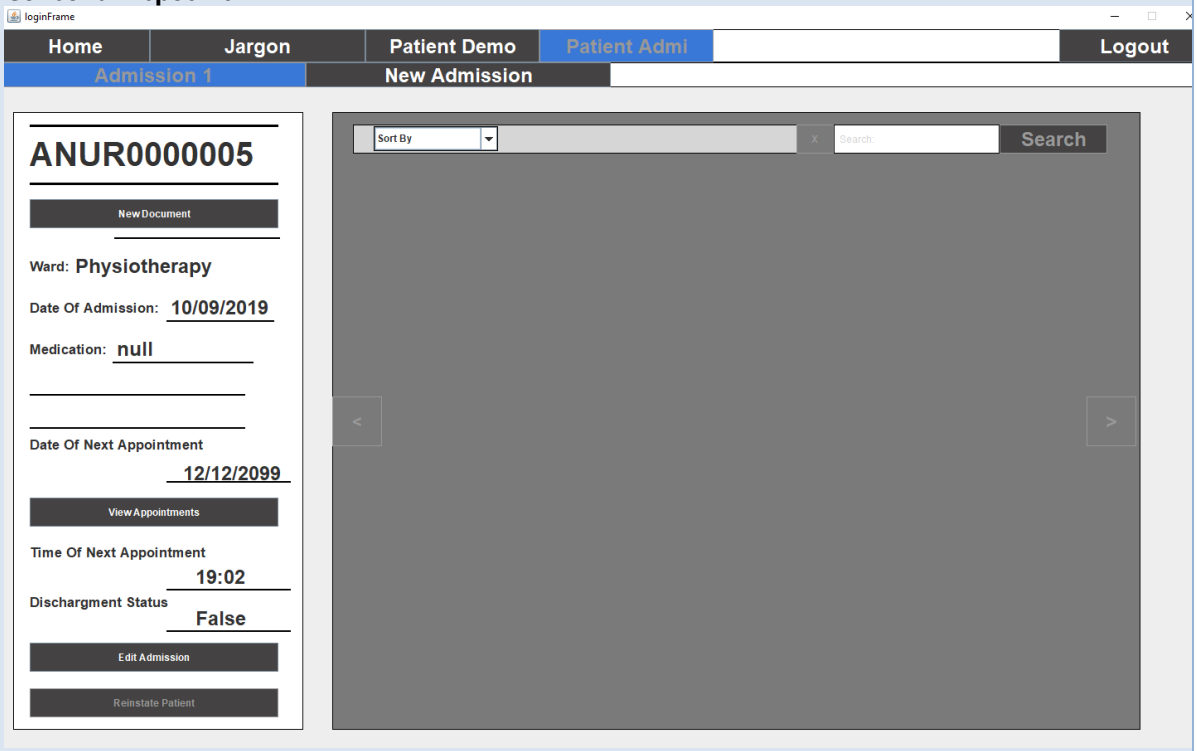
Features

This page is used to allow the user to create an account on the system as a patient, on this page they can do the following:

1. Return back to the prior page the user was just on
2. Create their account using the information they had just entered
3. Enter their information regarding them in general
4. If the field is not appropriate to enter text it can be selected either using a combo box or a radio button

Admission screen

Consultant specific



Output

There are no actual outputs for ths screen except links that move the panel onto the next desired page.

The only following data is retrieved for the panel:

PatientID,wardLocatedin,DateOfAdmissionCreation,Medication,DateOfNextAppointment,TimeOfNextAppointment,DischargmentStatus and the number of admissions

Features

On this screen they can view all the admissions they have on the system here all the information on the contact panel varies due to the type of user inspecting it.

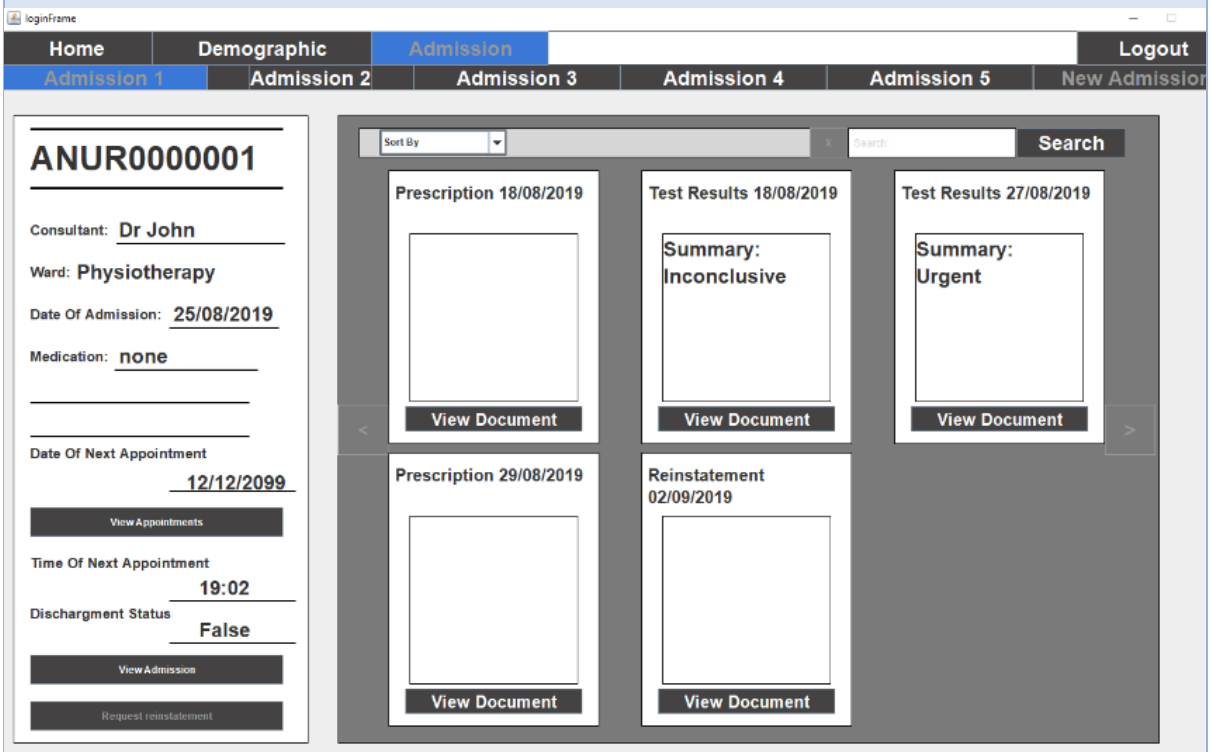
For the consultant they can:

1. Add new documents(Opens new document screen)
2. Edit the admission(loads edit admission screen)
3. Move to the patient's demographic panel
4. Move to the other admissions that are assigned to them
5. Search for a key document
6. Sort the list of documents
7. Clear the currently searched document

Reinstate the admission (if currently discharged

Admission screen

Patient specific



Output

There are no actual outputs for ths screen except links that move the panel onto the next desired page.

The only following data is retrieved for the panel:

PatientID,consultant,wardLocatedin,DateOfAdmissionCreation,Medication,DateOfNextAppointment,TimeOfNextAppointment,DischargmentStatus and the number of admissions

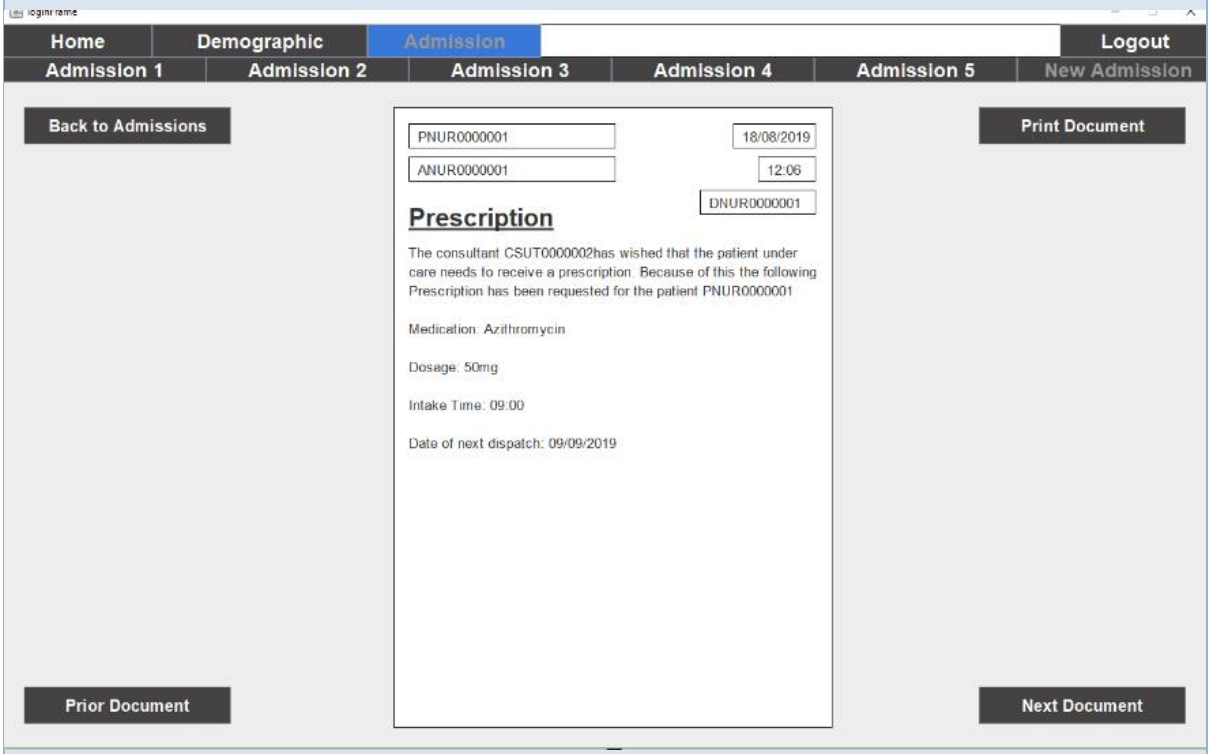
Features

On this screen they can view all the admissions they have on the system here all the information on the contact panel varies due to the type of user inspecting it.

For the patient they can:

1. View the admission information
2. View all their admissions by pressing the second tab bar
3. Add new admissions to the account
4. View individual documents
5. Navigate the page by moving the list of documents forward by 6 documents
6. Search for individual documents
7. Sort documents by key fields

Document Screen



Output

DNUR0000004,29/08/2019 17:28,Prescription,paracetamol,100g,12:02,12/12/2019, , ,
DNUR0000005,02/09/2019 13:36,Reinstatement,,,,,,

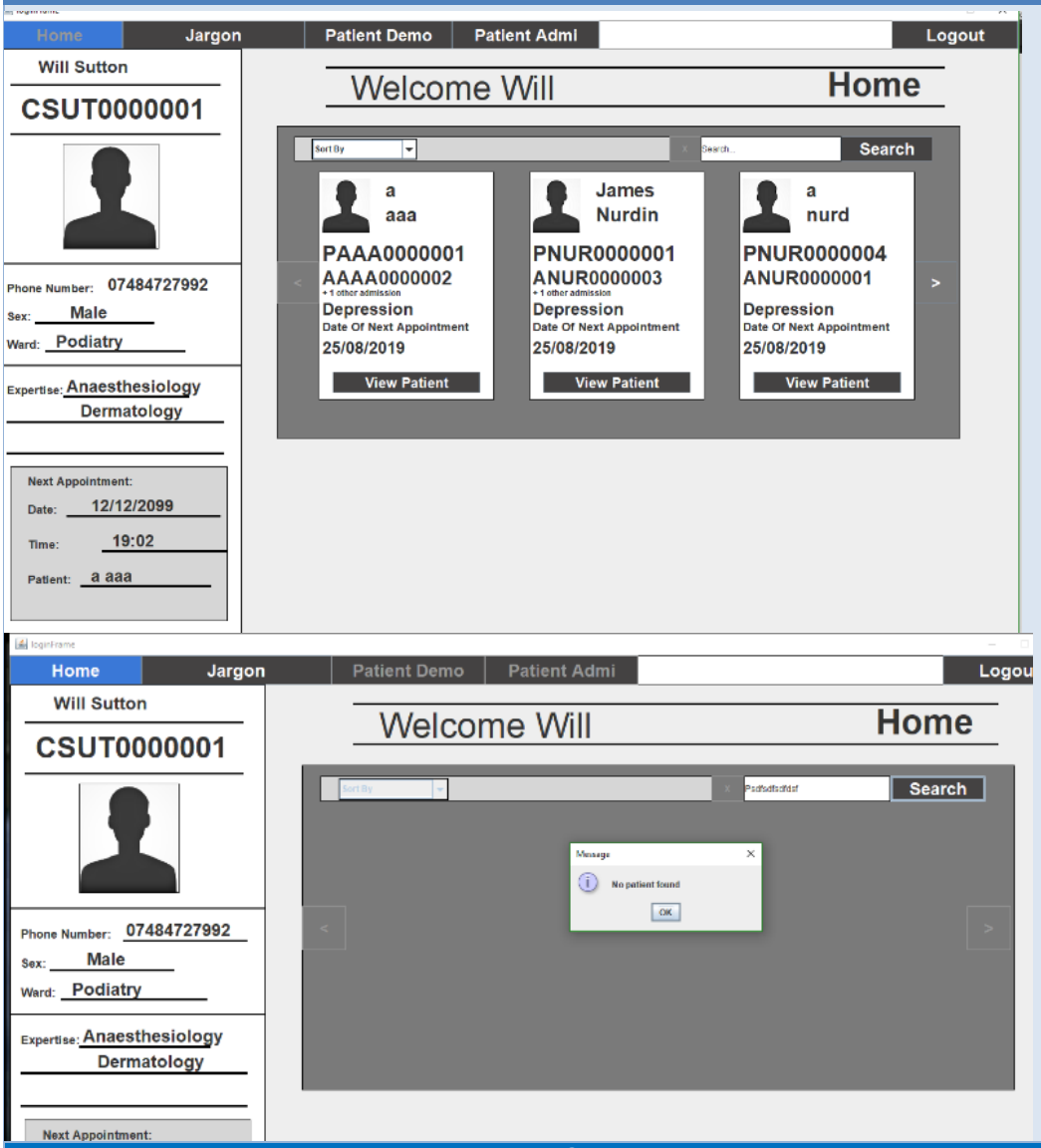
As you can see some documents don't actually need to have any unique data just only the type of document as it is so general there is no need to include personal data, this will be good for allowing documents to be viewed from non-patient or consultant users.

Features

This screen will allow the user to read the current document currently it will do the following:

1. Retrieve the individual document from the file and load all the fields with the correct text
2. They can return back to the current admission homepage either through the back button or admission 1 tab
3. They can go onto the other admissions through the admission tabs
4. The whole design is supposed to resemble a standard document viewing software like Adobe or word. When a document is read from file only the unique data is read and the rest is concatenated to save on file sizes as they do not need to be add

Consultant home screen



Output

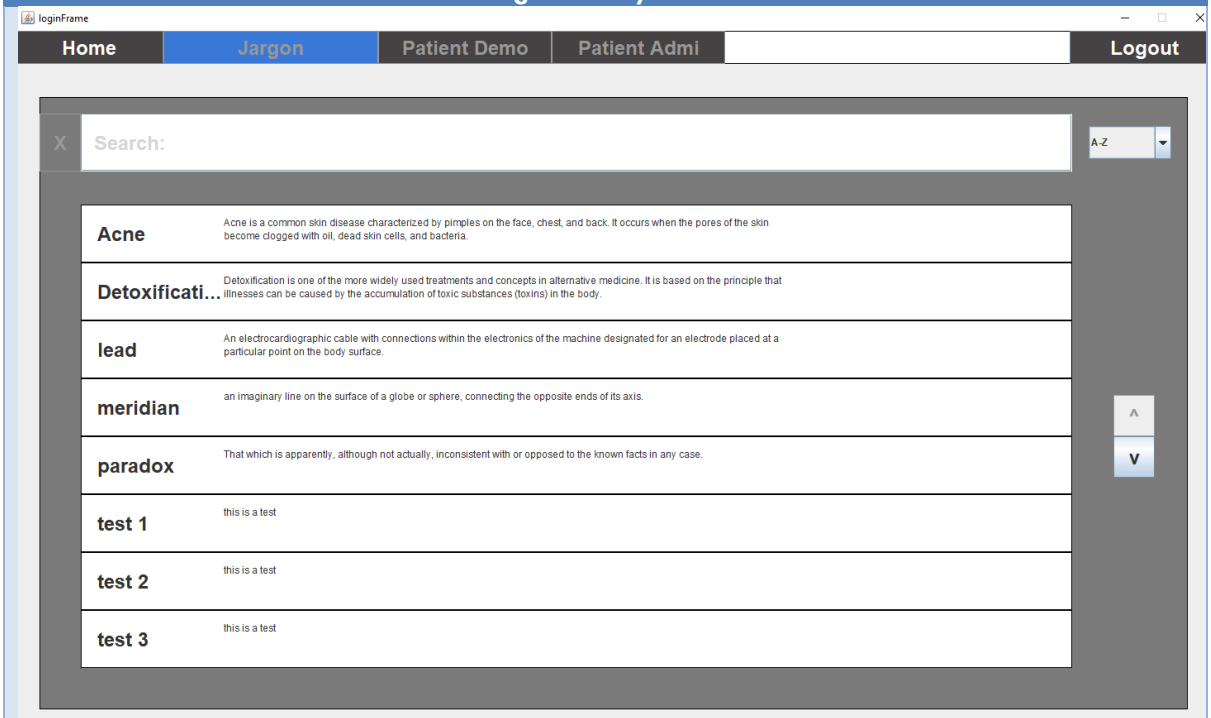
The only output of the screen Is a popup informing the user that their was no patient found, or a pateint card showing the patient returned from the searching of the patient

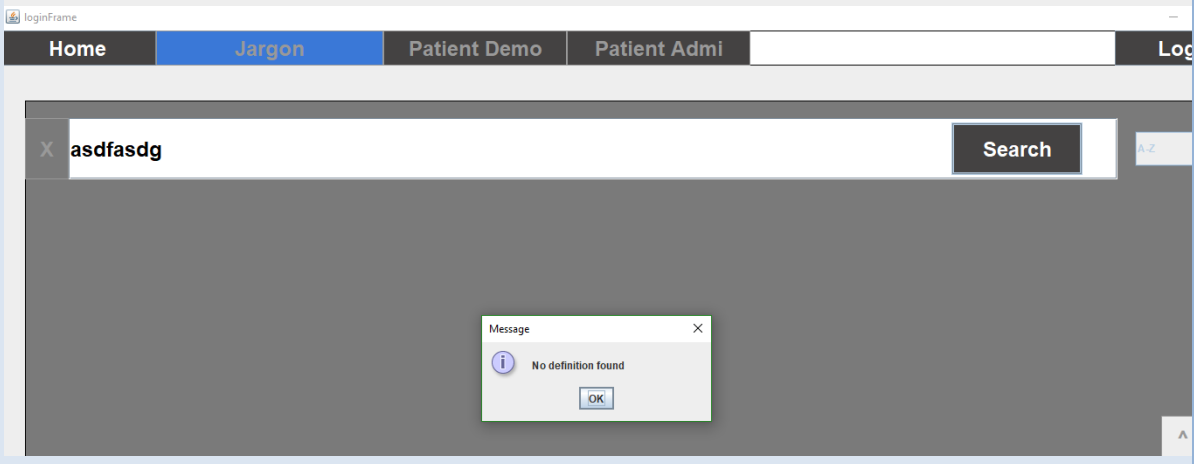
Features

This is the main home page for the consultant they can from this point in development can do the following:

1. Search for individual patients using their patient ids
2. Go to the jargon library screen
3. Logout
4. See their basic information from their demographic on the contact bar
5. Manually navigate the list by using the buttons forward and back
6. Finally, as of now they can receive a pop-up window informing them that they have entered an invalid id or that it does not exist on the system

Jargon library screen





Output

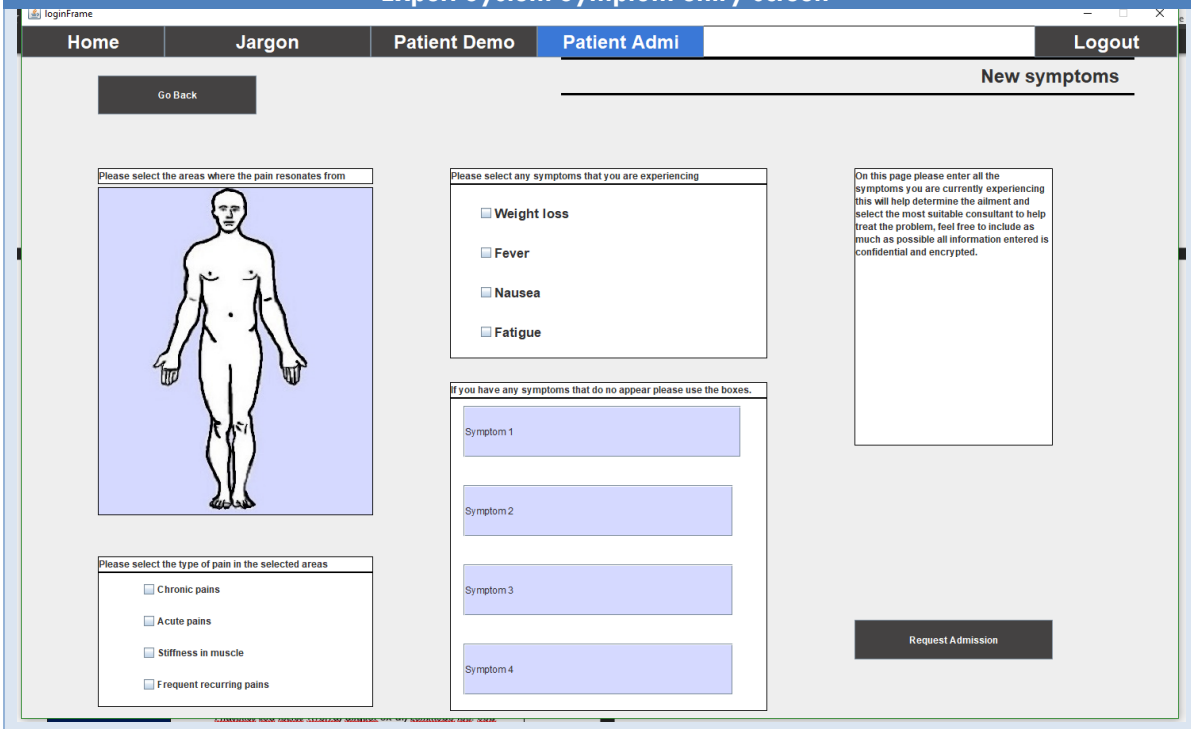
The popup of this screen is an indication that no definition was found, else the only output is the searched word along with ts defenition of the word.

Features

The main aspect of this system is to allow the user to search for words that they currently do not know the meaning of from this, the consultant can:

1. Enter a word and see if it is in the file
   1.5 if it is they will receive the definition
   1.6 if not they will get an output showing them that the word could not be found
2. They can manually navigate the page by pressing the up and down buttons which move the list up and down by 8 indexes.
3. Finally, they can sort the document into two orders ascending and descending alphabetical order.

Expert System Symptom entry screen



Output

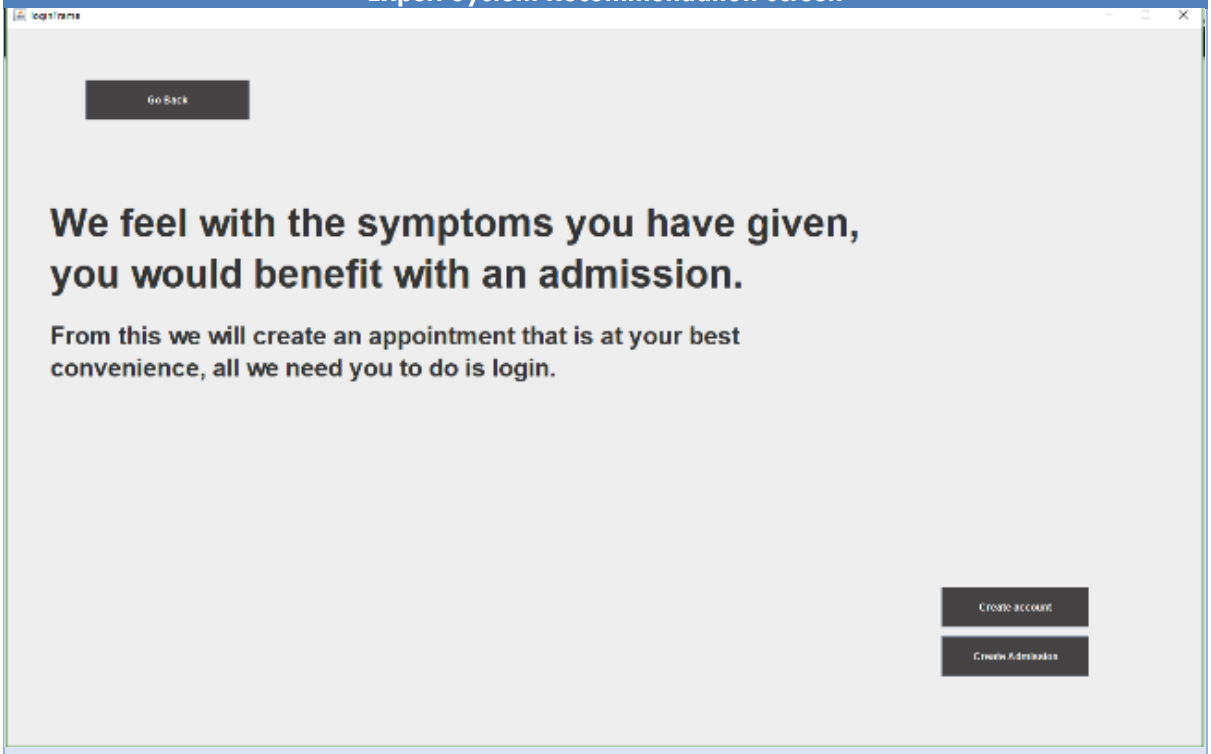
The output of this screen is a list of all selecteditems along with all unselected items eg:

Areas selected:
Chest Area
Head Area
==========
Symptom entered
Symptom 1
Symptom 2
Symptom 3
Symptom 4
==========
Types of pain selected
Acute pains

Features

This screen is to allow the patient to enter all their information regarding the ailment and can do the following:
Select the parts of the body the pain resonates from
Select the type of pain that occurs
Select any common symptoms that occur
Manually type any symptoms that occur
Finally, once they have finished this section they can press a button which will eventually generate an admission.

Expert System Recommendation screen



Output

Either the creation of an account and or the creation of an admission for the user, these both occur through the next pages provided by the system, if they are logged in the admissio is created if not they have to log in first

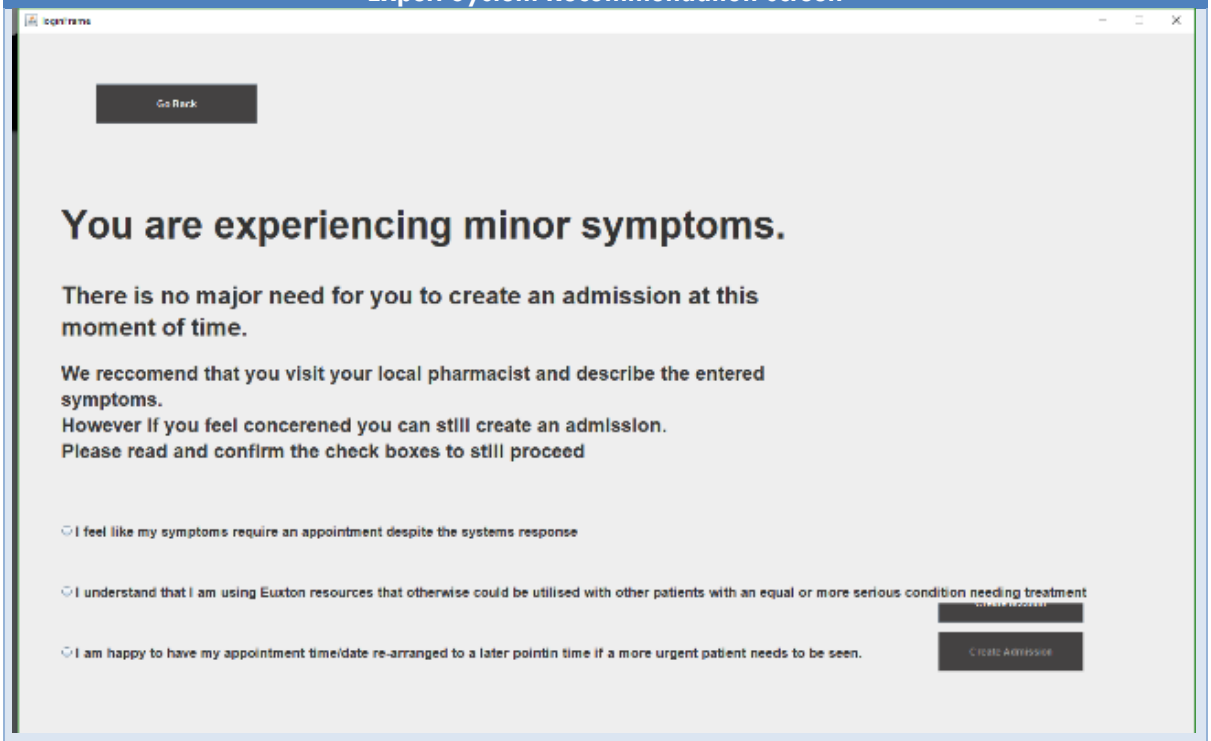
Features

This screen only has a few features assigned to It as the majority of the code for this screen is in the backend Where the system will eventually predetermine the severity and maybe even a diagnosis to the ailment.

If the admission is accepted the screen will

1. Allow the user to create an account and then create the admission
2. If already logged in create an admission.

Expert System Recommendation screen



Output

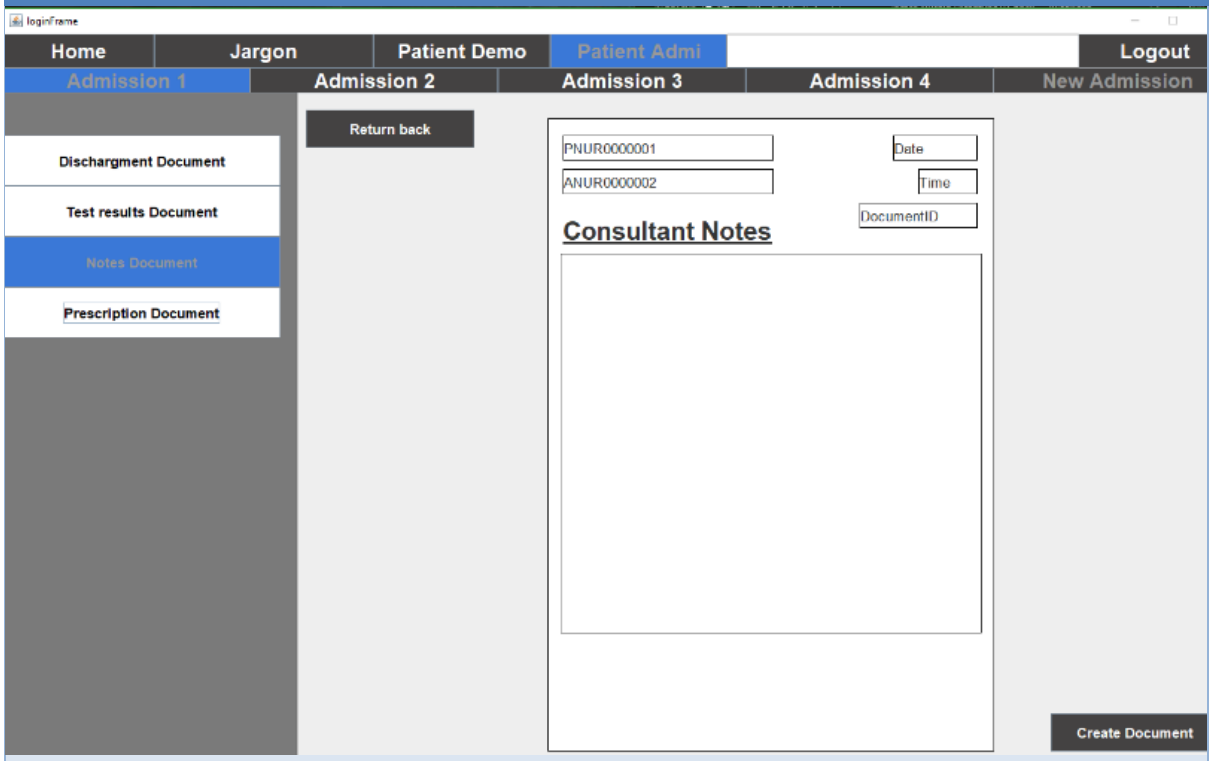
Either the creation of an account and or the creation of an admission for the user, these both occur through the next pages provided by the system, if they are logged in the admissio is created if not they have to log in first

Features

If the admission is not recognised as worthy in the eyes of the system it will

1. Force the user to agree to a set of conditions in order to create an admission
2. Allow the user to create an account and then create the admission
3. If already logged in create an admission.

New document screen



Output

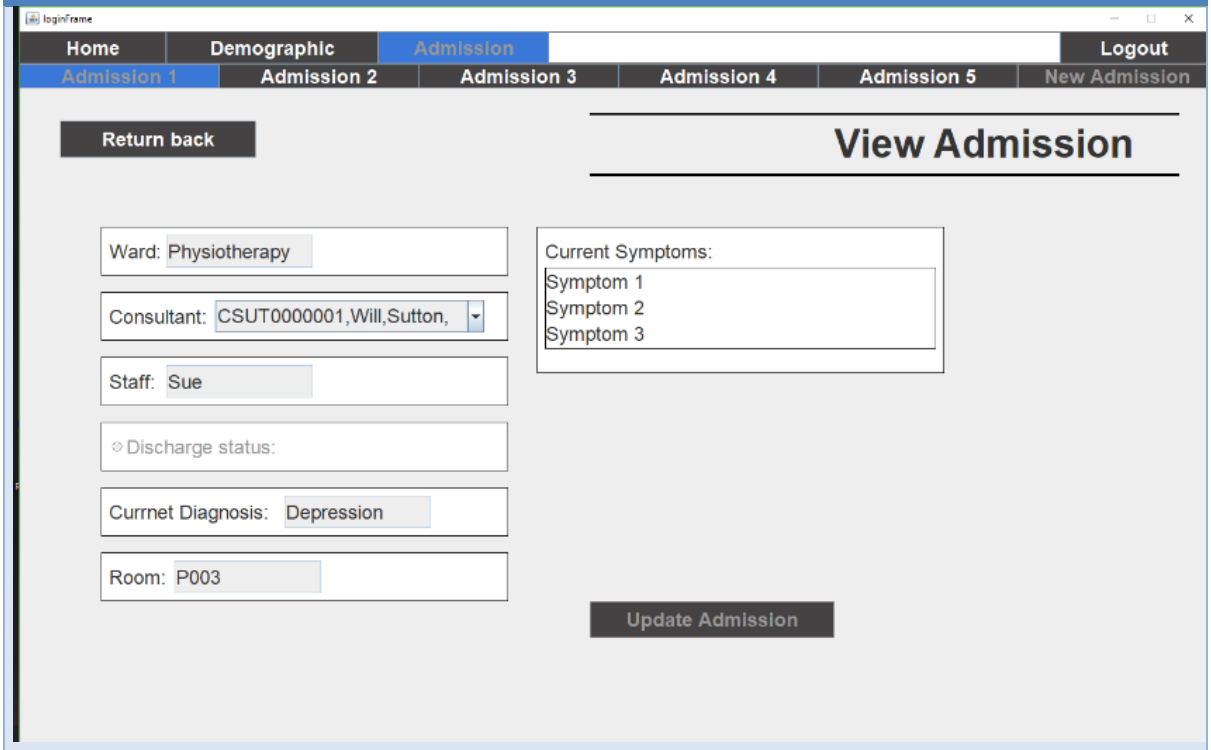
Once the document is created the output is twofold. The document is added to the patients admission homepage allowing them to view it and the following gneral information is added to the documentation file. documentID,dateOfDocumentCreation,docType,medicationName,medicationDosage,medicationIntakeTime,medicationDateOfNextDispatch,notes,testResults,summary

Features

On this panel the consultant can create documents for the currently selected patient, and can:

1. Select the current type of document either (dischargement , notes, test results, prescription)
2. The consultant can enter text into the respective text fields and areas
3. Create the document which will write the document to the correct location for it to be retrieved.

View admission screen



Output

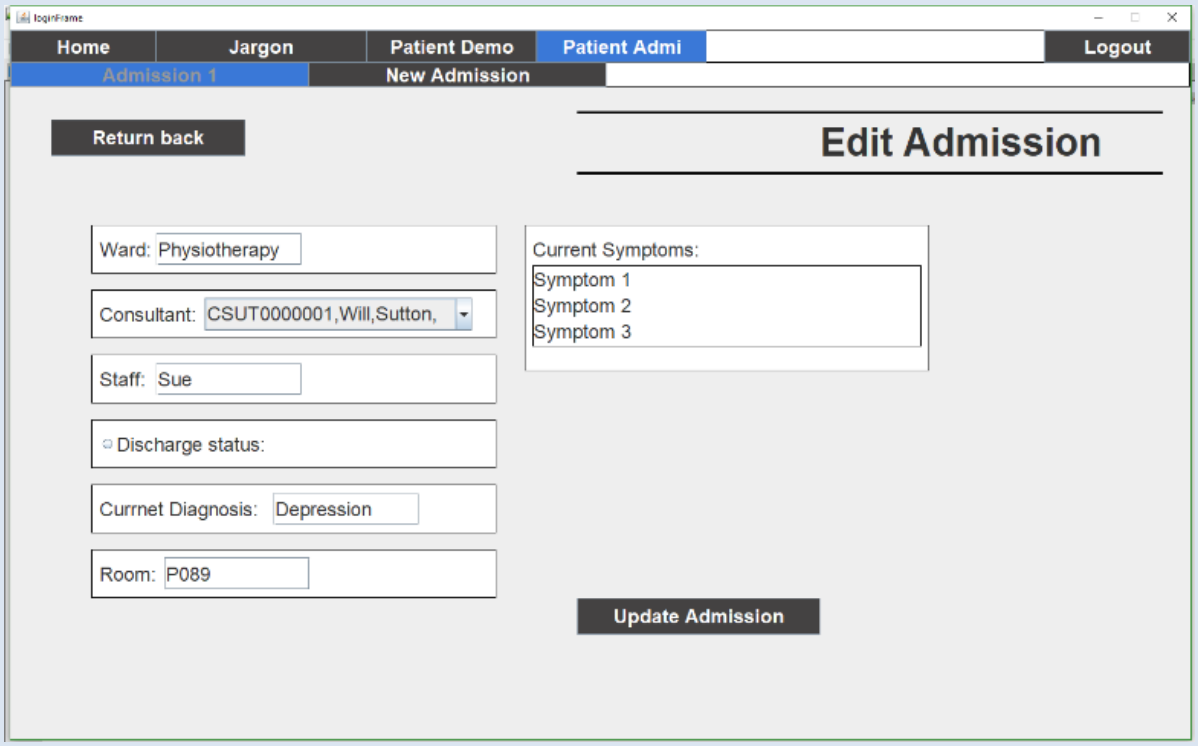
As this is on the patient side of the system they are unable to amend any of the information presented to them because of this there are no outputs to the user except the abulity to mve onto the other panels and view the inforamtion infront of them.

Features

On this page can only be accessed by the patient and prevents any fields from being edited by the user however are actually the same components as the other panel just disabled from editing and have the update admission button invisible. They can:

1. The patient can view the information

Edit admission screen



Output

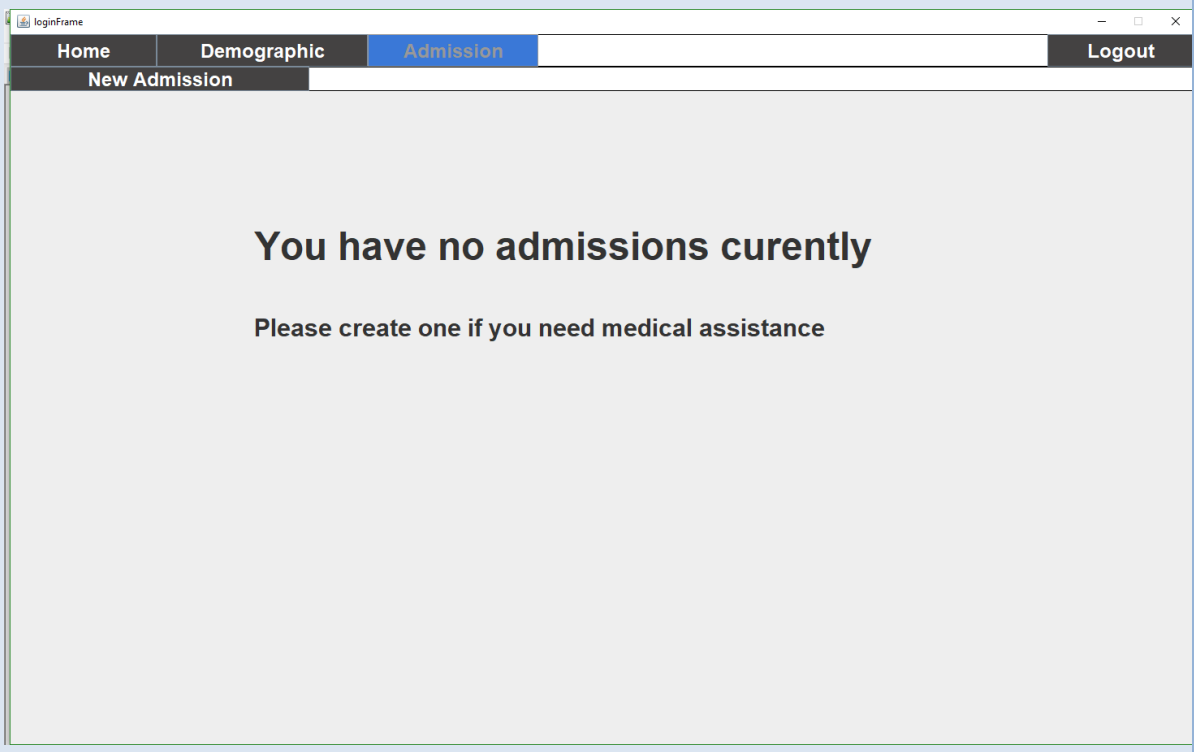
Unlike the prior panel where the patient would be unable to amend the admission, as this panel resides on the consultatn side of the system it can be edited and therefore the followind is an example of what data can be outputted to file:

ANUR0000005,Physiotherapy,John,12/12/2099 19:2,true,0,null,P003,Depression,Symptom 1|Symptom 2|Symptom 3|,Sue,10/09/2019,STOM0000001,CSUT0000002

Features

1. Amend the fields and subsequentially update the admission
2. Return back to the consultant's admission page
3. If the consultant chooses to change the admission the following will happen
   - Remove the admission from the current admission
   - Add the admission to the new consultant
   - Add the admission to the correct array

Edit admission screen



Output

As there is little to do on this screen there is little output besides the features described

Features

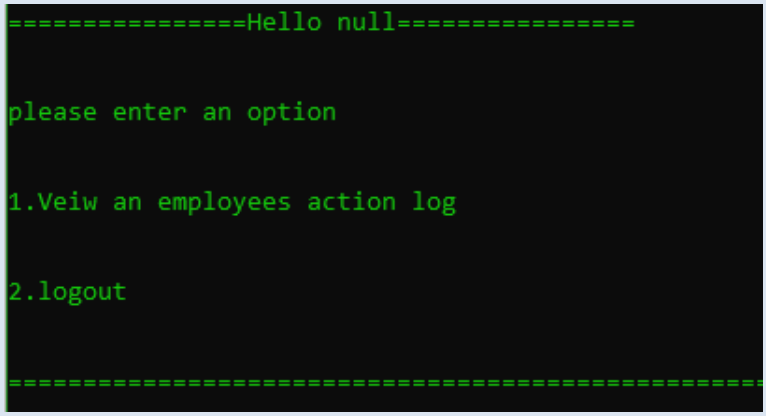
The is a screen to only inform the patient that they have no admissions attached to the account and in order to fully utilise the page they need to create an account. This page as of now will only show when the admission count in the patient demographic file equals one and not account for any discharged documents.

The patient can:

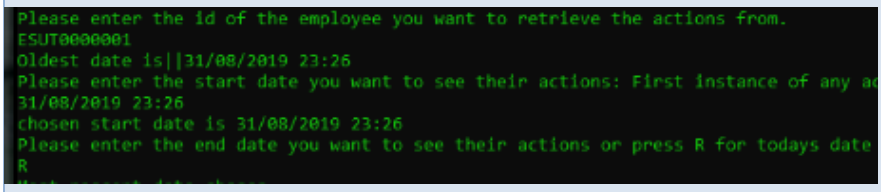
1. Create an admission
2. Go to the homepage
3. Go to the demographic page
4. Logout

Management home screen

The option menu



Option 1



Output

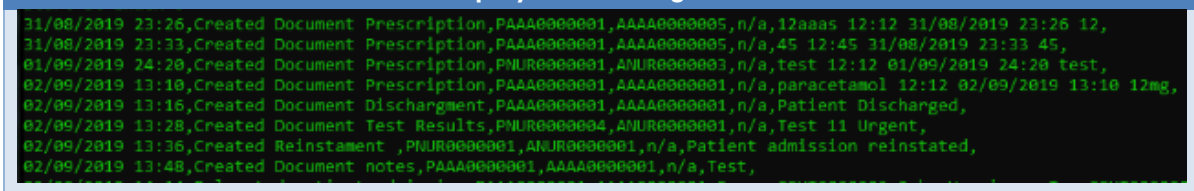
The output of these two "Screens" are the option menu and the questions asked to the user to return the correct employees' information. Besides this there are no other outputs to receive. The following questions are:

Please enter the id of the employee you want to retrieve the actions from.
Please enter the start date you want to see their actions: First instance of any activity is [Oldest Date]
Please enter the end date you want to see their actions or press R for today's date

Features

When the management login the user will have 2 options either logout or view the action log. As this is the prototype there of course a finite number of options but hopefully overtime during the development of the final system more can be implemented to add to the functionality of the entity like the ability to generate Employee entities and so on.
For option two the following input is retrieved:
EmployeeID
Start and end date

Employee action log Screen



Output

The output for this feature is a loop that iterates through every action between two paramters a start date and an end date, while it iterates through it outputs the following data to the user: **employeeID,dateOfDocumentCreation,currentAction,patientID,admissionID, oldData,newData.**

It does this until the end of the list is reached and then returns the user to the home “Screen”

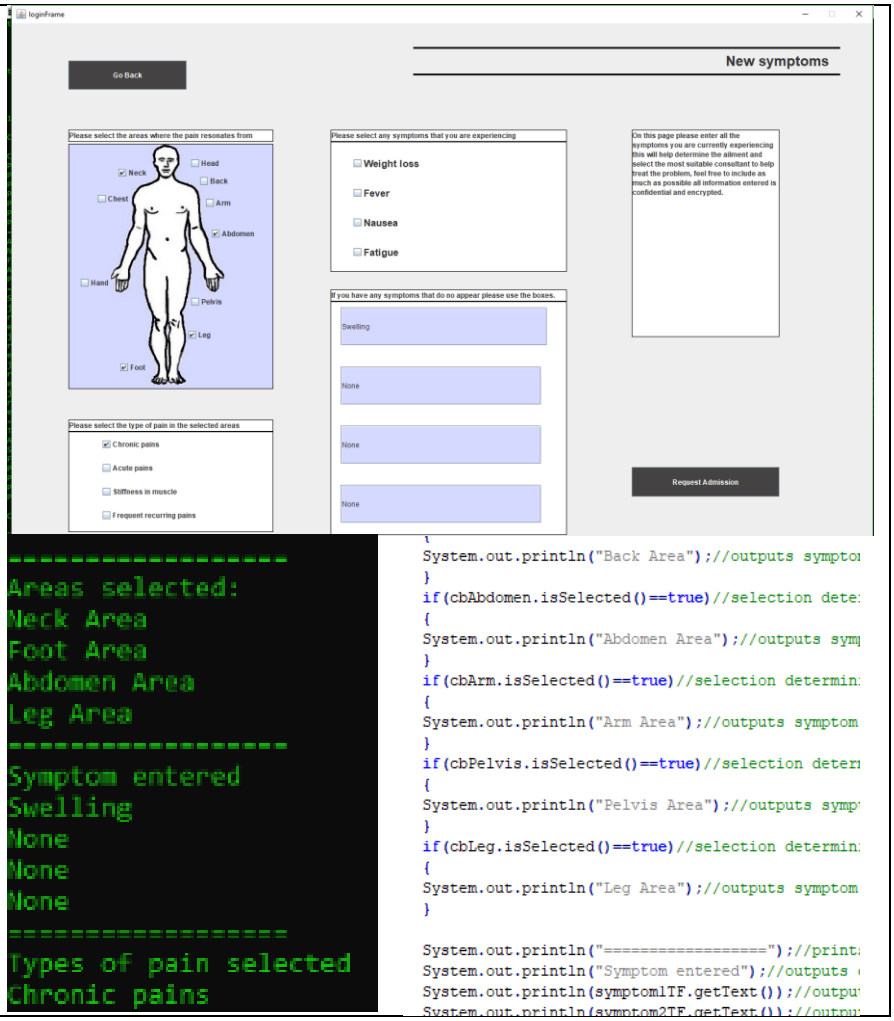
Features

On this “Screen” there is one main feature the actions preformed by the employee using the data from option one the system will read the action file and output all the actions the employee has done over the time entered by the user once the entirety has been printed the option menu will be loaded again.

1.3 Functioning System (with Data)

Now that all the pages and outputs have been shown, I would like to now go into detail about every objective I have set out to include, in order to do this, I have indexed every objective against the number specified in the investigation. In addition to prove that everything works fully I have tried to show as much proof as possible, where I could I took screen shots of processes along with source code if it needed showing to the user its function but as some of the source code calls upon other methods, these methods have not been included otherwise the entire source code would eventually be placed into the system.

Feature	**Entering symptoms onto the system [1]**
Description	If the user wants to create an admission they can enter the symptoms onto this panel here it will be stored and will create an admission once the create admission button has been pressed after the next screen
Output & Storage Requirements	The eventual output will be to run the next objective and correctly generate a correct diagnosis. Currently it will automatically accept the admission request and will allow for the admission to be generated for the user. A series of attributes for the time being are used to store the data.
Data Used	None, all data is entered by the user except for items in radio boxes for example areas the pain resonates from.

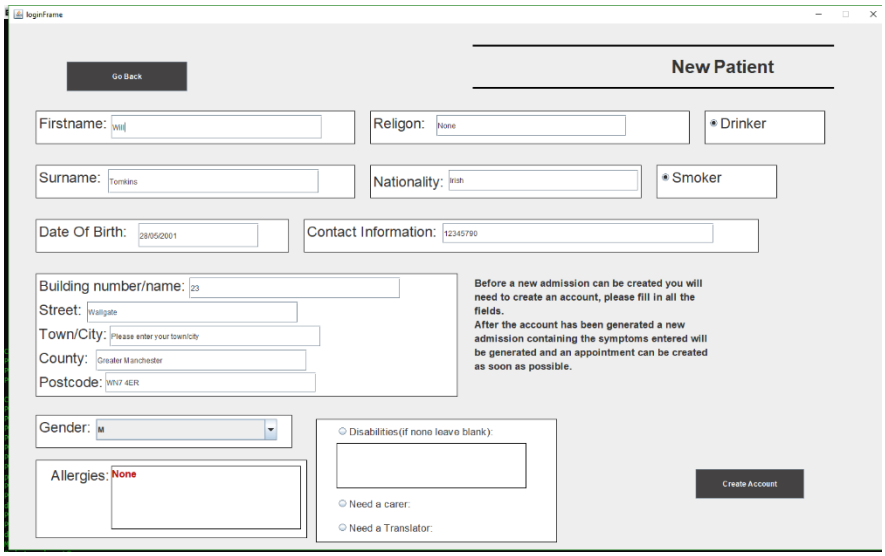
Evidence	

Feature	**Creating new patients on the system [3]**
Description	If the user wants to become a patient all they have to do is correctly enter the fields on the screen then press create account this will load the home screen and will allow them to user the system as a patient
Output & Storage Requirements	Output will be the writing of the information to file, and as this varies between users the size and therefore storage requirements differ between users. For storing the information, a record will be used as the data structure to hold and temporary store the data.
Data Used	In this example it is: patientID,surName,firstName,addressHouseNum, ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, ,numberOfAdmissions,dob,religion,allergies, gender,disability,carer,translator

Evidence

```
Patient newPatient = new Patient();//creates new instance of this object
newPatient.firstName = demographicFNameTextFeild.getText();//retireives attribute from c
newPatient.surName = demographicSNTextFeild.getText();//retireives attribute from compon
try
{
    newPatient.dob = ft.parse(demographicDOBTextFeild.getText());//retireives attribute
}
catch(Exception exc)//if any errrors are found they are caught here
{

}
newPatient.addressHouseNum = Integer.parseInt(demographicBuildingNUMTextFeild.getText())
newPatient.addressHouseStreet = demographicStreetTextFeild.getText();//retireives attrib
newPatient.postcode = demographicPostcode.getText();//retireives attribute from componen
newPatient.town = demographicTownTextFeild.getText();//retireives attribute from compone
newPatient.gender = ((String) demographicGender.getSelectedItem()).charAt(0);//retireive
newPatient.contactNum = demographicContactinfo.getText();//retireives attribute from com
newPatient.religon = demographicReligonTextFeild.getText();//retireives attribute from c
newPatient.bloodType ="";//retireives attribute from component
newPatient.nationality = demographicNotioanlityTextFeild.getText();//retireives attribut
newPatient.allergies = demographicAllergies.getText();//retireives attribute from compon
newPatient.smoker = demographicSmokerPromptLbl.isSelected();//retireives attribute from
newPatient.drinker = demographicDrinkerPromptLbl.isSelected();//retireives attribute fro
newPatient.disability = demographicDisablitiesTextFeild.getText();//retireives attribute
newPatient.carer = demographicCarerPromptLbl.isSelected();//retireives attribute from co
newPatient.translator = demographicTranslatorPromptLbl.isSelected();//retireives attribu
newPatient.numberOfAdmissions = 0;//retireives attribute from component
PatientList pl = new PatientList();//retireives attribute from component
currentPatient = pl.createNewPatient(newPatient);//calls using the new instance the crea
    notifications =currentPatient.getNotifications(currentPatient.patientID);////retireve
numberOfNotifications = notifications.length;//finds the number of notifications the use
currentPatient.updateNotificationsPatient(notifications);//updates the patients notifica
loginChoice = 0;
createPatientHomepagePanelGUI(currentPatient);//calls method which displays panel for th
```

```
patient.patientID = createUniqueID(patient.surName,"Patient_Demographic.txt","P");//creates a unique
String[] stringedPatientList =rffReturnFullFile("Patient_Demographic.txt");//returns entire file in t
String[] stringedNotifications = rffReturnFullFile("Notifications.txt");//returns entire file in the
int patientPosition = findIndexOfPatient(stringedPatientList,patient.patientID);//finds the correct p
//notification file
String[] list2 = Arrays.copyOf(stringedNotifications, stringedNotifications.length+1);//sets all the
stringedNotifications = list2;//updates the previous array to equal the newly declared one

for(int index = stringedNotifications.length-1;index>patientPosition;index--)//runs a for loop for th
{

    stringedNotifications[index] =stringedNotifications[index-1];//sets the index to move up aN INDEX
}
String patientNotification = (patient.patientID+",");//declares first part of notification by adding
stringedNotifications[patientPosition] = patientNotification;//adds new notification to the array
wtfNewPatient(stringedNotifications,"Notifications.txt");//writes the array to file
//demographic file
String[] list3 = Arrays.copyOf(stringedPatientList, stringedPatientList.length+1);//sets all the data
stringedPatientList = list3;//updates the previous array to equal the newly declared one
for(int index = stringedPatientList.length-1;index>patientPosition;index--)//runs a for loop for that
{
    stringedPatientList[index] =stringedPatientList[index-1];//sets the index to move up aN INDEX
}
String patientDemo = (patient.patientID+","+patient.surName+","+patient.firstName+","+patient.address
stringedPatientList[patientPosition] = patientDemo;//adds new information to the array
wtfNewPatient(stringedPatientList,"Patient_Demographic.txt");//writes entire array to file
try
{
    String file = patient.patientID+"_Admissions.txt";//creates a new filename for the user
    FileWriter fwr = new FileWriter(file);//declare a new file writer that will ammend not not write
    fwr.close();//closes the file
```

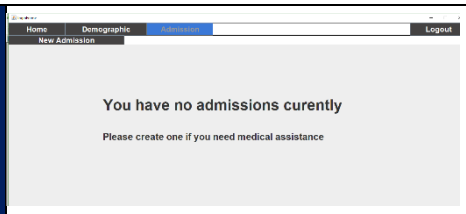


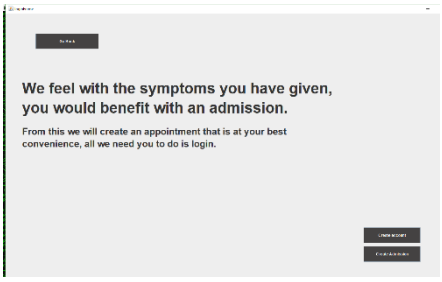

```
PTOM0000001,Tomkins,Will,23,Wallgate,Please enter your town/city,WN7 4ER,12345790,Irish,,true,true,0,28/05/2001,None,None,M,,false,false
```

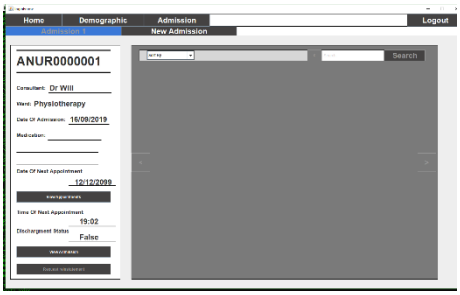
	The first image is the code which reads the attributes from the jcomponents and assigns that to a new instance of the patient object. The second one correctly creates a primary key for the patient and then inserts it into the file where it belongs. The image below that is that of the GUI where the attributes are entered and finally the last is the output data

Feature	**Creating admissions on the system [4]**
Description	In order for the full use of the system it is preferred that the user had an admission in order to do so they need to create one or have them created for them; this is the process for this feature.
Output & Storage Requirements	The output of this is the generation of a new admission and have that attached to both the patient and corresponding employee. However, as the user can enter text through the text fields they can have a more dynamic range of field length rather than a static predetermined value. Like other key entities they will utilise a record.
Data Used	admissionID,ward,consultantName, dateOfNextAppointmentA,active,numberOfDocuments ,medication,room,currentDiagnosis,stringedSymptoms, staffName, dateAdmissionCreated,admissionsStaffID admissionsConsultantID.

Evidence





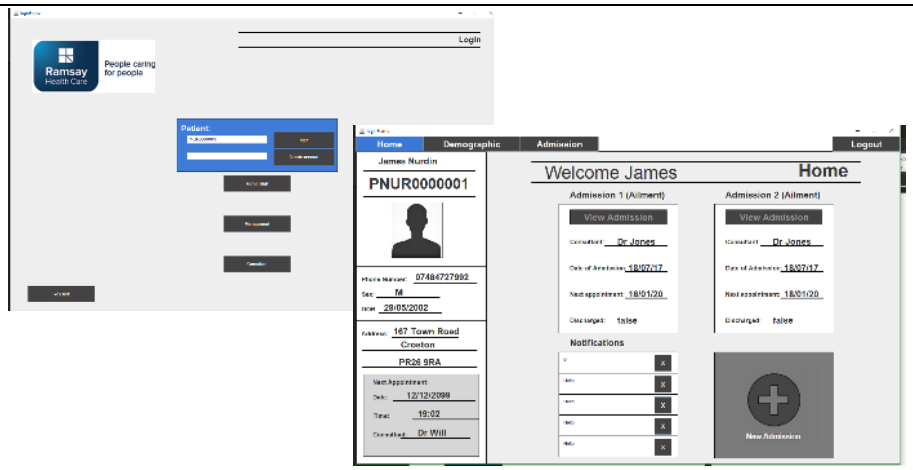


The first image is the page a patient who has no admissions may see, the next one is the screen that is created after the objectives 1 and 2 have been executed. The next image is the final image of the admission homepage now showing the newly created admission attached to the patient. The text line is the admission in the consultant file indicating the admission has been attached to the consultants account.

```
00008¿ANUR0000001¿ PNUR0000011¿ANUR0000001¿
```

The code is showing what is ran to create the admission. The first few lines are where the dummy data is attached to a new instance of admission. After this, the admission is added to the consultants file. Next we see the number of admissions increment for the patient. Finally, to sum up the rest of the code, the system now searches through and performs file management now including the admission to the array of admissions and then writing the new list to file. Finally, the new homepage is loaded.

```
    admissionListCopy=currentPatient.retrieveAdmissions(currentPatient);
    if(loginChoice==3)
    {
        admissionListCopy=tempAdmissionList = al.retrieveAdmissionList(currentPatient);//retrives
        System.out.println("Size of array is "+admissionListCopy.length);
    }
    Admission tempAdmission = new Admission();//new instance of admission is created
    tempAdmission.admissionID = tempAdmission.createUniqueID(currentPatient.surName, currentPatie
    //System.out.println("A ID "+tempAdmission.admissionID);
    tempAdmission.ward ="Physiotherapy";//sets attribute for object
    tempAdmission.room="P003";//sets attribute for object
    tempAdmission.active=true;//sets attribute for object
    tempAdmission.numberOfDocuments =0;//sets attribute for object
    tempAdmission.staffName="Sue";//sets attribute for object
    tempAdmission.consultantName="Will";//sets attribute for object
    tempAdmission.currentDiagnosis="Depression";//sets attribute for object
    tempAdmission.listOfSymptoms[0] = symptom1TF.getText();//sets attribute for object from compo
    tempAdmission.listOfSymptoms[1] = symptom2TF.getText();//sets attribute for object from compo
    tempAdmission.listOfSymptoms[2] = symptom3TF.getText();//sets attribute for object from compo
    tempAdmission.listOfSymptoms[3] = symptom4TF.getText();//sets attribute for object from compo
    tempAdmission.dateAdmissionCreated = todaysDate;//sets attribute for object
    try
    {
        tempAdmission.dateOfNextAppointmentA = ftTimeInc.parse("12/12/2099 19:02");//sets attribu
    }
    catch(Exception exc)
    {
    }
    tempAdmission.admissionsStaffID="STCM0000001";//sets attribute for object
    tempAdmission.admissionsConsultantID = "CSUT0000001";//sets attribute for object
    Consultant tempConsultant = new Consultant();//creates  a new instance for consultant
    //System.out.println(tempAdmission.admissionsConsultantID);
    //System.out.println(currentPatient.patientID);
    //System.out.println(tempAdmission.admissionID);
    tempConsultant.addpatientToConsultantRecord(tempAdmission.admissionsConsultantID,currentPatie
    currentPatient.numberOfAdmissions = currentPatient.numberOfAdmissions + 1;//num of admissions
    numberOfAdmissions++;//num of admissions increments
    Admission[] list4 = Arrays.copyOf(admissionListCopy, admissionListCopy.length+1);//sets all t
    admissionListCopy = list4;//updates the previous array to equal the newly declared one
    if(admissionList==admissionListCopy)
    {
        admissionListCopy[numberOfAdmissions-1] = tempAdmission;//an array of a smaller length is
    }
    if(admissionList!=admissionListCopy)
    {
        admissionListCopy[numberOfAdmissions-1] = tempAdmission;//an array of a smaller length is
    }
    currentAdmission = tempAdmission;//current admission size is reduced
    numberOfDocuments = currentAdmission.numberOfDocuments;//number of documents is found
    singleDefinition = new Document[numberOfDocuments];
    listOfdocumentsGUI = currentAdmission.listOfDocuments;//this arary is saved to the GUI array
    PatientList pl = new PatientList();//new instance of patient list is declared
    pl.updatePatientDemo(currentPatient);//the pateint demographic is updated
try
{
    String file = currentPatient.patientID+"_Admissions.txt";//filename for admission
    FileWriter fwr = new FileWriter(file);//declare a new file writer that will ammen
    int iterationCondition=0;
    if(loginChoice==0)
    {iterationCondition=numberOfAdmissions;
    }
    if(loginChoice==3)
    {iterationCondition=numberOfAdmissions+1;
    }
    for(int count = 0;count<numberOfAdmissions;count++)
    {Admission tempAdmissionwtfa = new Admission();//creates a new instance of admiss
        tempAdmissionwtfa = admissionListCopy[count];//updates current admisison
        String stringedSymptoms = (tempAdmissionwtfa.listOfSymptoms[0]+"|"+tempAdmiss
        String stringedAdmission = (tempAdmissionwtfa.admissionID+","+tempAdmissionwt
        System.out.println("--------------------------"+stringedAdmission);
        fwr.write(stringedAdmission);//writes to file current line and admission
        fwr.write("\r\n");
    }
    fwr.close();//closes the fil
}
catch(Exception exc)
{System.out.println("Error 2");
}
    if(loginChoice ==0)
    {
        admissionList=admissionListCopy;
        admissionList=currentPatient.retrieveAdmissions(currentPatient);//retieves al
        int lengthAdmission = admissionList.length;//finds how many admissions exist
        //System.out.println("number of admissions "+ lengthAdmission);
        for(int i= 0;i<lengthAdmission;i++)//runs for length of the number of admisis
        {
            admissionList[i].retrieveDocuments(currentPatient,admissionList[i]);//ret
        }
    }
    if(loginChoice!=0)
    {admissionList[finalIndexOfAdmission+1] =currentAdmission;
    }
    findSoonestAppointment();//finds the soonest appointment due to the admissions cr
    if(numberOfAdmissions ==1)//selection determining if there are now this many admi
    {setActiveAdmissionTopPanelBttn(topBarAdmission1Bttn);//sets active top bar to th
    }
    if(numberOfAdmissions ==2)//selection determining if there are now this many admi
    {setActiveAdmissionTopPanelBttn(topBarAdmission2Bttn);//sets active top bar to th
    }
    if(numberOfAdmissions ==3)//selection determining if there are now this many admi
    {setActiveAdmissionTopPanelBttn(topBarAdmission3Bttn);//sets active top bar to th
    }
    if(numberOfAdmissions ==4)//selection determining if there are now this many admi
    {setActiveAdmissionTopPanelBttn(topBarAdmission4Bttn);//sets active top bar to th
    }
    if(numberOfAdmissions ==5)//selection determining if there are now this many admi
    {setActiveAdmissionTopPanelBttn(topBarAdmission5Bttn);//sets active top bar to th
    }
    createadmissionHomepagePanelGUI();//updates admission homepage by calling method
```

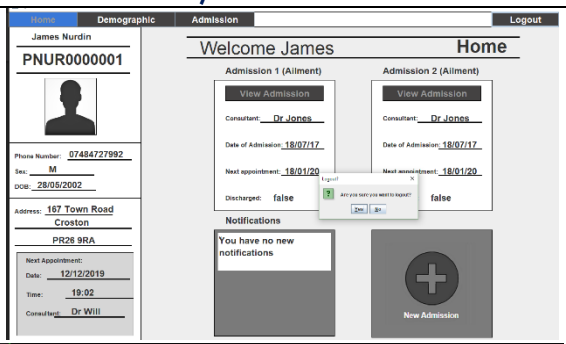
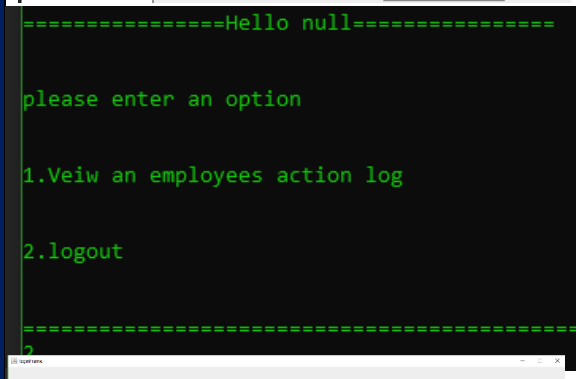
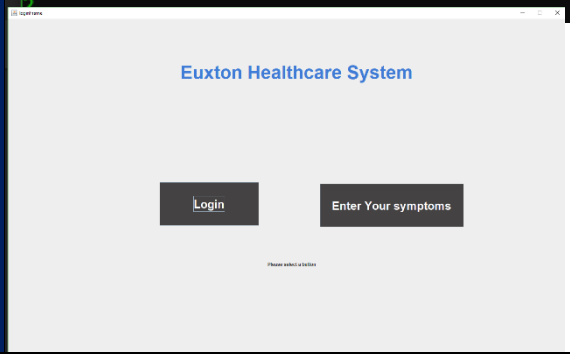
Feature	Login System [6]
Description	The function of this system is to retrieve all the information regarding the individual from the correct location in the file. After this the system will output the homepage screen for that user.
Output & Storage Requirements	The output is the generation of the homepage with all their information in the correct fields. The storage requirements vary from the number of characters for text fields but should normally be only around 150 characters depending on how descriptive the user wants to be. For storage a record will be used to hold the information regarding the desired entity.
Data Used	For the patient: patientID,surName,firstName,addressHouseNum ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, numberOfAdmissons,dob,religion,allergies, gender,disability,carer,translator For the Consultant: consultantID,EmployeeID,surName,firstName, wardLocatedIn,expertise,wage,hoursPerWeek, archived,numberOfPatients,addressHouseNum, addressHouseStreet,town,postcode,contactNum listOfPatientsAdmissions
Evidence	 These are screenshots of the login page and the subsequent screen after loading the information. Below are the two methods used to retrieve the consultant or patient information.

```
public void setUpConsultantObj()
{
    currentConsultant = new Consultant();//declares new instance of consultant
    currentConsultant = currentConsultant.retrieveConsultantInfo(username);//retrieves the correct consultant information
    if(currentConsultant.numberOfPatients>0)//selection determining if the consultant has at least one patient
    {
    numberOfConsultantPatients = currentConsultant.listOfPatientsAdmissions.length;//finds out how many patients the consul
    }
    else
    {
        numberOfConsultantPatients = 0;//if the consultant has no patients it is set to 0
    }
    //System.out.println(numberOfConsultantPatients + " number of patients");
    al = new AdmissionList();//new instance of admisison list is created
    listOfCsPatients = new Patient[numberOfConsultantPatients];//declares for the list of the consultants patients
    for(int count = 0;count<numberOfConsultantPatients;count++)//runs a for loop for the number of patients he has
    {
        String id = currentConsultant.listOfPatientsAdmissions[count];//id is set

        String consultantPateintID = id.substring(0,11);//id is found by splitting string
        //System.out.println("Test 2 id of desired patient " +consultantPateintID);
        Patient tempPatient = new Patient();//new instance of patient is declared

        listOfCsPatients[count] = tempPatient.retrievePatientInfo(consultantPateintID);//create our temp patient in the arr
        System.out.println("Test 3.5 number of admissions " +listOfCsPatients[count].numberOfAdmissions);

        String stringedAdmissionConsultant = id.substring(12,(id.length()));//finds the concatenated admission ids


        String[] admissionIDsConsultant =  tempPatient.unConcatenateStringAdmission(stringedAdmissionConsultant,"&");//arra
        int lengthOfConcatenatedAdmission = admissionIDsConsultant.length;//number of desired admissions from that one pati
        tempAdmissionList = al.retrieveAdmissionList(listOfCsPatients[count]);//retrives all the admissions both desired ar
        for(int admissionArrayLength = 0;admissionArrayLength< tempAdmissionList.length;admissionArrayLength++)//runs for t
        {
            for(int innerCount = 0; innerCount<lengthOfConcatenatedAdmission;innerCount++)//runs for the number of times th
            {

                if(tempAdmissionList[admissionArrayLength].admissionID.equals(admissionIDsConsultant[innerCount]))//selecti
                {
                    listofAdmissions[count][innerCount] = tempAdmissionList[admissionArrayLength];//using the index found f
```

```
public void setUpPatientObj()
{
    date = false;//sets that no date for the patient has been set
    //creates a new patient
    currentPatient = new Patient();
    currentPatient = currentPatient.retrievePatientInfo(username);
    //System.out.println("PatientID "+ currentPatient.patientID);
    //retireves their notfications
    notifications =currentPatient.getNotifications(currentPatient.patientID);
    numberOfNotifications = notifications.length;
    currentPatient.updateNotificationsPatient(notifications);
    //finds number of admissions
    numberOfAdmissions = currentPatient.numberOfAdmissions;
    //System.out.println(numberOfAdmissions);
    admissionList=currentPatient.retrieveAdmissions(currentPatient);
    admissionListCopy=currentPatient.retrieveAdmissions(currentPatient);
    int lengthAdmission = admissionList.length;//finds the length of the admission
    //System.out.println("number of admissions "+ lengthAdmission);
    for(int i= 0;i<lengthAdmission;i++)
    {
        admissionList[i].retrieveDocuments(currentPatient,admissionList[i]);//decl
    }
    findSoonestAppointment();//calls method which finds the soonest appointment
}
```

For the first few lines the system will run the code which reads the lines of the consultant's info and assigns them to the new instance of the consultant. What happens next is that the list of admissions is retrieved and assigns them to the list of patients.

In this screen shot the code a new instance is declared the information for the patient is retrieved and assigned to the object. Next the system reads the notifications and assigns them to the list of admissions, with the length being found also. Next the admissions are then found from file and then for each admission the list of documents is also read and attached to the entity.

Feature	Logging out [6]
Description	Once the user has finished with their account on the system they can logout if they need to either change accounts or want to leave the current user type. This will return them back to the start screen
Output & Storage Requirements	The output will return the user back to the initial screen the user sees. The current entity record will be overwritten and replaced with the new user when they next log in.
Data Used	None(any fields that are used current user will be overwritten)
Evidence	   The first two images are the options to log out wither a popup window or a cmd line option for the user. The third screen is the start screen which is created on loading it after logging out.

	The first few lines of code show the declaration of the pop-up window. The two selection statements show whether if the user wants to logout or not. If they do it sets the current panel as the start one the rest set attributes to allow other panels to reset attributes for their fields

```
if(e.getSource()==logoutbttn)
{
    String logoutConifimration = "Are you sure you want to logout?";//confirmati
    String popUpWindowLogout = "Logout?";//text for frame
    int logoutConfirmationanswer = logoutConfirm.showConfirmDialog(null, logoutC
    if (logoutConfirmationanswer == 0)//user wishes to logout
    {
        emptyArray(startPanel);
        System.out.println("startPanel");
        setActiveTopPanelBttn(homebttn);//active top bar is set to the button ju
        currentPanel = startPanel;//current panel is correctly updated
        loaded[6][0]= false;//the variable is set as true to prevent the compone
        loaded[6][2]= false;//the variable is set as true to prevent the compone
    }
    else//wish to stay
    {
        logoutConfirm.setVisible(false);//popup is hidden from sight
    }
    logoutConfirm.setVisible(false);//popup is hidden from sight regardless of c
}
```

Feature	**Searching for Employees [11]**
Description	When the management entity wants to find an employee, they will have to enter their id, which will cause the system to search the file for that particular employee returning their information
Output & Storage Requirements	The output usually is the index in the array the item is located at. The storage requirement is the size of the array of employees and any data read from file, the only file management operations is the process of reading the file.
Data Used	The data used would be the array of items to search through the primary key also being our search value and our index in which the item is located.

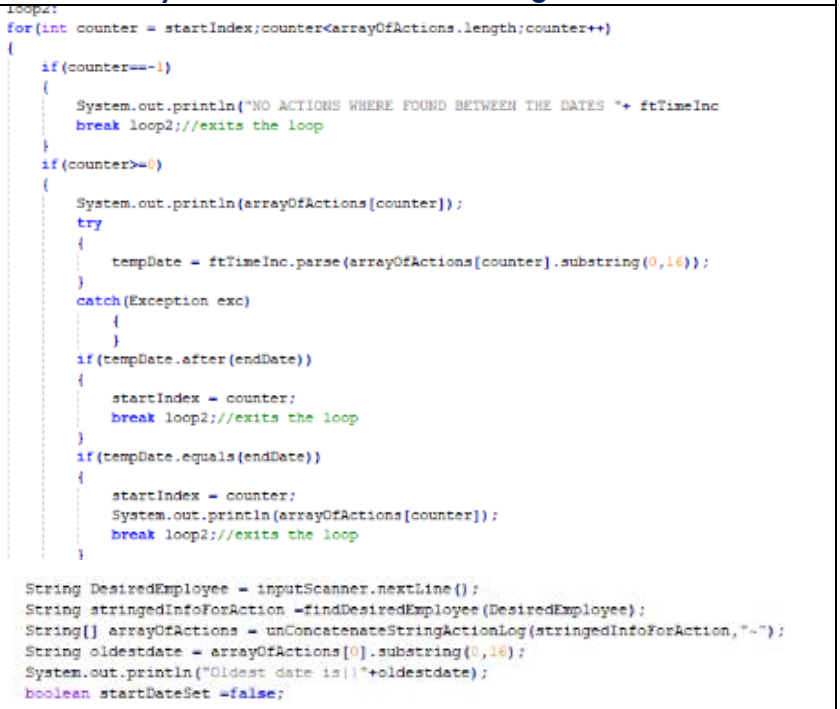
Evidence

```
public int findIndexOfPatient(String[] array, String desiredPatient)
{
    String currentID;//initalises the id used to find the desired value
    int position = -1;//initaly sets the value to an index that will alloways is out of bounds
    for(int count = 0;count<array.length;count++)//for loop that runs through the entirty of the array
    {
        currentID = array[count];//assigns the current id to the attribute
        if(count== 0)//selection determining if the value is at the start of the array
        {
            /*
            An int value: 0 if the string is equal to the other string, ignoring case differences.
            < 0 if the string is lexicographically less than the other string
            > 0 if the string is lexicographically greater than the other string (more characters)
            */
            currentID = array[count].substring(0,11);//substrings string to get primary key
            if(desiredPatient.compareToIgnoreCase(array[0])<0)//selection determining if the desired location is at the first
            {
                position = 0;//assigns position to first index
            }
        }
        if((count > 0)&&(count<array.length))//if the desired location is between the start and end
        {
            currentID = array[count].substring(0,11);//substrings string to get primary key
            if((desiredPatient.compareToIgnoreCase(array[count])<0)&&(desiredPatient.compareToIgnoreCase(array[count-1])>0))/
            {
                position = count;//asigns the correct index the item should be located in
            }
        }
        if(count== array.length-1)//selection determining if the location is at the last index of the array
        {
            currentID = array[count].substring(0,11);//substrings string to get primary key
            if(desiredPatient.compareToIgnoreCase(currentID)>0)//making sure that the position is above 0
            {
                position = array.length;//assigns index to be the next value above the length(last index + 1)
            }
        }
        currentID = array[count].substring(0,11);//substrings string to get primary key
        if(desiredPatient.compareToIgnoreCase(array[0])==0)
        {
            position = count;//asigns the correct index the item should be located in
        }

    }
    return(position);//returns index
```

In the code we can see the generalised binary searched used throughout the system, this has been developed in such a manner that it can be used with pretty much any array and value . The second image shows how the search method actually plays a part in retrieving the information with it then passing along the string employee. Finally, we can see the real application in cmd with the search working and the employee being found in the array.

```
}
public String findDesiredEmployee(String employeeID)
{
    String[] readInfo = rffReturnFullFile("Employee_Actionlog.txt");//retrieves the employees from file
    int indexOfEmployee = findIndexOfPatient(readInfo,employeeID);//finds the index of the employee
    String foundEmployee = readInfo[indexOfEmployee];//saves the index of the correct consultant from the array
    System.out.println("EMPLOYEE FOUND");//informs the user that the employe was found on the system
    return(foundEmployee);//returns the employee
```

```
Please enter the id of the employee you want to retrieve the actions from.
ESUT0000002
EMPLOYEE FOUND
```

Feature	Viewing an employee's action log [12]
Description	If the management entity want to view their staff's activities on the system all they have to do is enter the following information Employee id, start date and end date. Then the system will print all the activities they have done within the time frame.
Output & Storage Requirements	The output will be the actions including the patient affected the specific admission along with any new or old data that may have been created. For storage all that happens is that the information is read from file and outputted. The information however is held on a file, with a line per employee basis.
Data Used	Just an array of the actions nothing else.
Evidence	 Here is the code the top one will run from the start date to the end of the array or until the loop is broken(Then end date is reached), if the search algorithm prior returned a rouge value it is outputted that there is no data and the loop is broken if not it iterates through every line checking if the desired end has been reached. The second one retrieves the data from file and then correctly indexes them into an array.

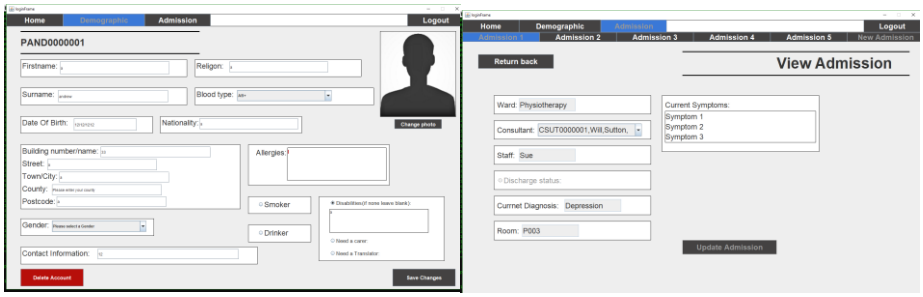
```
loop2:
for(int counter = startIndex;counter<arrayOfActions.length;counter++)
{
    if(counter==-1)
    {
        System.out.println("NO ACTIONS WHERE FOUND BETWEEN THE DATES "+ ftTimeInc
        break loop2;//exits the loop
    }
    if(counter>=0)
    {
        System.out.println(arrayOfActions[counter]);
        try
        {
            tempDate = ftTimeInc.parse(arrayOfActions[counter].substring(0,16));
        }
        catch(Exception exc)
            {
            }
        if(tempDate.after(endDate))
        {
            startIndex = counter;
            break loop2;//exits the loop
        }
        if(tempDate.equals(endDate))
        {
            startIndex = counter;
            System.out.println(arrayOfActions[counter]);
            break loop2;//exits the loop
        }
```

```
String DesiredEmployee = inputScanner.nextLine();
String stringedInfoForAction =findDesiredEmployee(DesiredEmployee);
String[] arrayOfActions = unConcatenateStringActionLog(stringedInfoForAction,"~");
String oldestdate = arrayOfActions[0].substring(0,16);
System.out.println("Oldest date is||"+oldestdate);
boolean startDateSet =false;
```

Feature	Writing actions to file. [13B]
Description	When an employee performs an action regarding the patient, it will be recorded in a log to help monitor and protect data integrity and ensure that if anything happens all actions can be traced to the original user.
Output & Storage Requirements	The output is the action of writing the information to the file in a serial manner as it is not amended just added by date. For storage requirements a file is used to hold any information regarding the patient and each line is isolated per employee.
Data Used	EmployeeID,dateOfDocumentCreation,patientID, admissionID,oldData(any data previously held in the fields),newData(The data the employee has entered)
Evidence	

```
iblic void writeActionToFile(Consultant consultant,Document document,Admission admission,Patient patient,int action)

   String[] readInfo = rffReturnFullFile("Employee_Actionlog.txt");//retireves the file where the actions are stored
   int locationOfEmployee=findIndexOfEmployee(readInfo,consultant.employeeID);//finds the index of the correct employee
   if(action == 0)//selection determining if the current action was of this type
   {currentAction="Created Document Prescription";//declares the action of the consultant
       oldData ="n/a";//concatenates the information together
       newData =document.medicationName+" "+document.medicationIntakeTime+" "+ftTimeInc.format(document.medicationDateOf
   }
   if(action == 1)//selection determining if the current action was of this type
   {
   if(action == 2)//selection determining if the current action was of this type
   {
   if(action == 3)//selection determining if the current action was of this type
   {
   if(action == 4)//selection determining if the current action was of this type
   {
   if(action == 5)//selection determining if the current action was of this type
   {
   if(action == 7)//selection determining if the current action was of this type
   {
       currentAction="Relocated patient admission";//declares the action of the consultant
       oldData ="From: "+consultant.consultantID;//concatenates the information together
       newData ="To: "+consultant.newConsultantID;//concatenates the information together
   }
   if(action == 8)//selection determining if the current action was of this type
   {
   try{if(consultant.employeeID.equals(readInfo[locationOfEmployee].substring(0,11))==false)//trys to see if the desired
       {String[] list4 = Arrays.copyOf(readInfo, readInfo.length+1);//sets all the data the same but adds on an extra in
           readInfo = list4;//updates the previous array to equal the newly declared one
           for(int count = readInfo.length-1;locationOfEmployee<count;count--)//runs a for loop that moves every item ab
           {
               readInfo[count+1]= readInfo[count];//moves items forward by one
           }
           readInfo[locationOfEmployee]=consultant.employeeID+"~"+ftTimeInc.format(document.dateOfDocumentCreation)+","+
           //System.out.println("NEW PATIENT ADDED");//informs the user a new patient has been added
       }else if(consultant.employeeID.equals(readInfo[locationOfEmployee].substring(0,11))==true)//checks to see if the
       {readInfo[locationOfEmployee]=readInfo[locationOfEmployee]+ftTimeInc.format(document.dateOfDocumentCreation)+","+
       }
       //System.out.println(readInfo[locationOfEmployee]);
   }
   catch(Exception exc)
   {if(locationOfEmployee==readInfo.length)//if the line of the desired employee is at the end of the array(out of bound
   {
       String[] list4 = Arrays.copyOf(readInfo, readInfo.length+1);//sets all the data the same but adds on an extra inc
           readInfo = list4;//updates the previous array to equal the newly declared one
       readInfo[locationOfEmployee]=consultant.employeeID+"~"+ftTimeInc.format(document.dateOfDocumentCreation)+","+curr
   }
   }try
   {FileWriter fw = new FileWriter("Employee_Actionlog.txt");//declare a new file writer that will ammend not not write
   for(int count = 0;count<readInfo.length;count++)//runs a for loop for the number of consultants on the system that ex
   {fw.write(readInfo[count]);//writes consultant to line
   fw.write("\r\n");//adds in a space to the file
   }
   fw.close();//closes the file
   }catch(Exception exc)
```

	In the code the system retrieves the employee's actions then, the system determines what action has actually been performed by the employee and correctly sets the new and old data attributes then the array of notifications is increased by one. After this the array of notifications will be concatenated together and then added back to the correct location, and then finally writing them to file.

Feature	**Patients can view their admissions and Demographic information [21]**
Description	Once the system has initialised and all components have been generated the patient will then be able to view their admission information and demographic information whenever they want.
Output & Storage Requirements	The output in each case would be to display the panel for the respective page, this would include loading in all the text fields and other components required of that page. AS there would be a varying amount of information between the users the number of bytes would vary however in all cases they are held on the same line
Data Used	For the admission: admissionID,ward,consultantName, dateOfNextAppointmentA,active,numberOfDocuments ,medication,room,currentDiagnosis,stringedSymptoms, staffName, dateAdmissionCreated,admissionsStaffID admissionsConsultantID. For the demographic: patientID,surName,firstName,addressHouseNum, ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, ,numberOfAdmissions,dob,religion,allergies, gender,disability,carer,translator
Evidence	 Here are the screens where the patients can view their information regarding the demographic and admission respectively

```
public void createEditEditAdmissionPanel()
{
    public void createEditAdmissionPageMainPart(JPanel panel)
    {
        panel.add(editAdmissionTitle);//the component is added to the panel
        panel.add(backToAdmissionConBttn);//the component is added to the panel
        editAdmissionTitle.setText("Edit Admission");//formatting component to include text
        saveChangesToAdmisionBttn.setEnabled(true);//the component is enabled to interact with t
        dischargeLblPrompt.setEnabled(true);//the component is enabled to interact with the user
        archiveAdmissionBttn.setEnabled(false);//the component is disabled
        wardAdmissionTextField.setEditable(true);//the text area is made so that the user can am
        StaffsNameTextField.setEditable(true);//the text area is made so that the user can ammen
        roomTextField.setEditable(true);//the text area is made so that the user can ammend the
        currentDiagnosisTextField.setEditable(true);//the text area is made so that the user can
        listOfCurrnetSymptomsTextPane.setEditable(true);//the text area is made so that the user
        //System.out.print("current admission is "+currentAdmission.active);
        if(currentAdmission.active == false)//selection determining if the current admission has
        {
        if(loginChoice!=3)//selection determining if the user is not a consultant
        {
            editAdmissionTitle.setText("View Admission");//formatting component to include text
            saveChangesToAdmisionBttn.setEnabled(false);//the component is disabled
            dischargeLblPrompt.setEnabled(false);//the component is disabled
            archiveAdmissionBttn.setEnabled(true);//the component is enabled to interact with th
            wardAdmissionTextField.setEditable(false);//the text will be unable to be edited by
            StaffsNameTextField.setEditable(false);//the text will be unable to be edited by the
            roomTextField.setEditable(false);//the text will be unable to be edited by the user
            currentDiagnosisTextField.setEditable(false);//the text will be unable to be edited
            listOfCurrnetSymptomsTextPane.setEditable(false);//the text will be unable to be edi
        }
        if(currentAdmission.active==true)//selection determining if the admission is active
        {
        if(currentAdmission.active==false)//selection determining if the admission is not active
        {
        wardAdmissionTextField.setText(currentAdmission.ward);//formatting component to include
        StaffsNameTextField.setText(currentAdmission.staffName);//formatting component to includ
        roomTextField.setText(currentAdmission.room);//formatting component to include text
        currentDiagnosisTextField.setText(currentAdmission.currentDiagnosis);//formatting compon
        listOfCurrnetSymptomsTextPane.setText(currentAdmission.listOfSymptoms[0]+"\n"+currentAdm
    }
        public void createEditAdmissionPageMainPartGeneral()
```

All the code does here is disable the ability to amend any fields in the document besides this the text is assigned to the component, that's it.

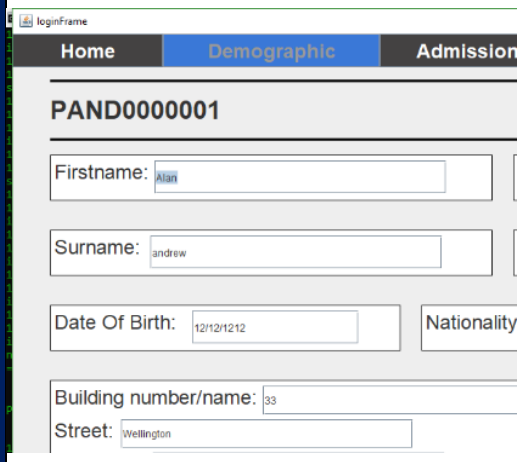
```
demographicFNameTextFeild.setText(currentPatient.firstName);//text is added to the compone
demographicSNTextFeild.setText(currentPatient.surName);//text is added to the component
demographicDOBTextFeild.setText(ft.format(currentPatient.dob));//text is added to the comp
demographicBuildingNUMTextFeild.setText(currentPatient.addressHouseNum+"");//text is added
demographicStreetTextFeild.setText(currentPatient.addressHouseStreet);//text is added to t
demographicTownTextFeild.setText(currentPatient.town);//text is added to the component
demographicPostcode.setText(currentPatient.postcode);//text is added to the component
System.out.println(currentPatient.gender);//text is added to the component
demographicGender.setSelectedItem(currentPatient.gender);//text is added to the component
demographicContactinfo.setText(currentPatient.contactNum);//text is added to the component
demographicAllergies.setText(currentPatient.allergies);//text is added to the component
demographicNotioanlityTextFeild.setText(currentPatient.nationality);//text is added to the
demographicBloodType.setSelectedItem(currentPatient.bloodType);//text is added to the comp
demographicReligonTextFeild.setText(currentPatient.religon);//text is added to the compone
demographicDisablitiesTextFeild.setText(currentPatient.disability);//text is added to the
```

The exact same code is used for the admission page just that it is for the admission information and that the attributes are for the admission.

Feature	Patients can amend their demographic information [22]
Description	If a patient feels that the information they entered when creating the account where invalid or may no longer correctly represent the information the patient can go into the demographic page of the system and amend any field they wish, this is then written to file.
Output & Storage Requirements	The output is the writing of the new information to file and the updating of the information regarding the demographic of the patient. The storage requirements are a record to hold the new information regarding the entity and a file to store it after they system closes. The information is only used on one line.
Data Used	patientID,surName,firstName,addressHouseNum, ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, ,numberOfAdmissions,dob,religion,allergies, gender,disability,carer,translator
Evidence	

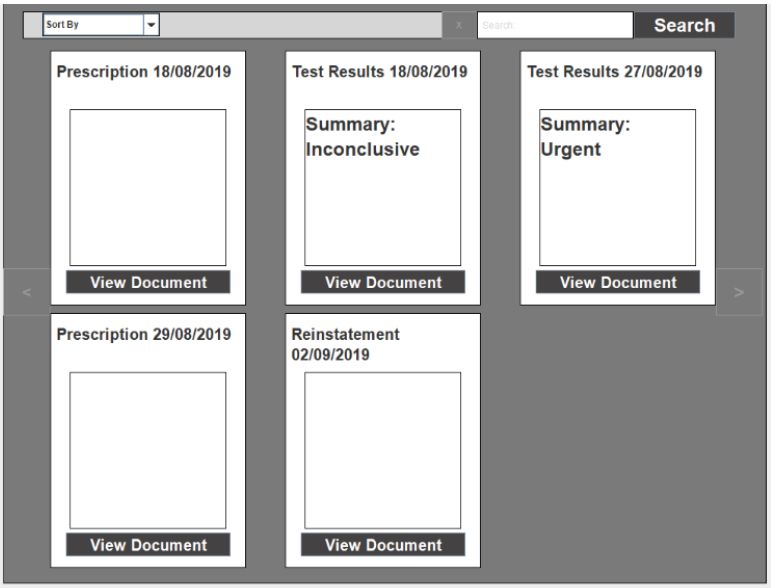
```
if(e.getSource()==saveDemographicChanges)
    {
        currentPatient.firstName = demographicFNameTextFeild.getText();//retrieves information from comp
        currentPatient.surName = demographicSNTextFeild.getText();//retrieves information from component
        try
        {
            currentPatient.dob = ft.parse(demographicDOBTextFeild.getText());//retrieves information fro
        }
        catch(Exception exc)//if any errrors are found they are caught here
        {
        }
        currentPatient.addressHouseNum = Integer.parseInt(demographicBuildingNUMTextFeild.getText());//r
        currentPatient.addressHouseStreet = demographicStreetTextFeild.getText();//retrieves informatio
        currentPatient.postcode = demographicPostcode.getText();//retrieves information from component
        currentPatient.town = demographicTownTextFeild.getText();//retrieves information from component
        currentPatient.gender = ((String) demographicGender.getSelectedItem()).charAt(0);//retrieves inf
        currentPatient.contactNum = demographicContactInfo.getText();//retrieves information from compo
        currentPatient.religon = demographicReligonTextFeild.getText();//retrieves information from com
        currentPatient.bloodType =(String) demographicBloodType.getSelectedItem();//retrieves informati
        currentPatient.nationality = demographicNotionlityTextFeild.getText();//retrieves information f
        currentPatient.allergies = demographicAllergies.getText();//retrieves information from component
        currentPatient.smoker = demographicSmokerPromptLbl.isSelected();//retrieves information from com
        currentPatient.drinker = demographicDrinkerPromptLbl.isSelected();//retrieves information from c
        currentPatient.disability = demographicDisablitiesTextFeild.getText();//retrieves information fr
        currentPatient.carer = demographicCarerPromptLbl.isSelected();//retrieves information from compo
        currentPatient.translator = demographicTranslatorPromptLbl.isSelected();//retrieves information
        PatientList pl = new PatientList();//creates new instance of patient list
        pl.updatePatientDemo(currentPatient);//calls the update patient demograpgic method passing throu
    }
```

```
public void updatePatientDemo (Patient patient)
{
    String[] stringedPatientList =rffReturnFullFile("Patient_Demographic.txt");//returns en
    String stringPatientID = patient.patientID;//using the patient passed through the id is
    int patientPosition = findPositionOfString(stringedPatientList,stringPatientID);//finds
    String patientDemo = (patient.patientID+","+patient.surName+","+patient.firstName+","+p
    stringedPatientList[patientPosition] = patientDemo;//sets the desired object from the a
    int length = stringedPatientList.length;//findsa out how many characters the string has
    wtf(stringedPatientList,"Patient_Demographic.txt",length);//writes to file the patient
}
```

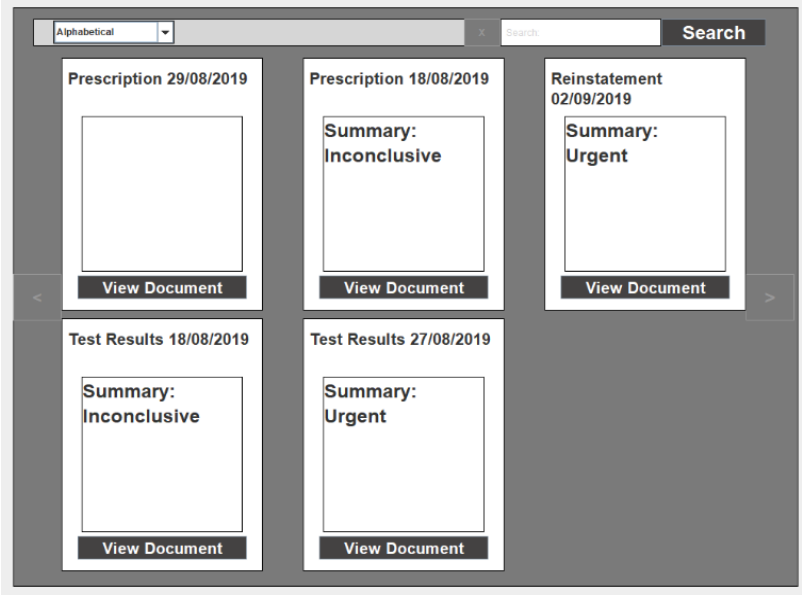
	For this example, I have changed the first name along with the street address of the user From a and a to Alan and Wellington. What happens is that the details are assigned to the current patient which allows them to be kept and not have to reread them from file but are then written to the line of the current patient where they are located in file overwriting any prior information held there. Then the entire contents of the file are rewritten so the only change where the desired fields. This can be seen by the text line beneath.  PAND0000001,andrew,Alan,33,Wellington,a,a,12,a,AB+

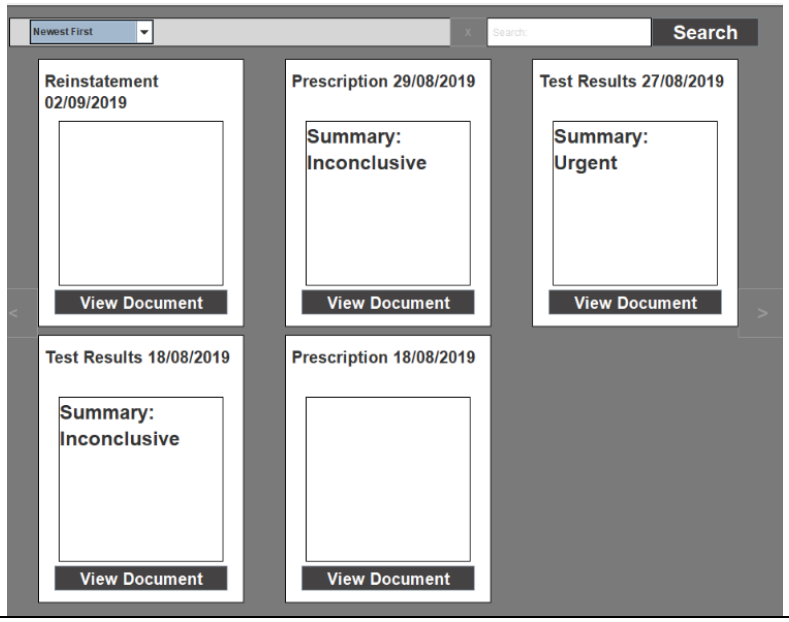
Feature	**Sorting Documents [26 A]**
Description	When viewing the documents on the system, the user may wish that the order in which they can be viewed is to be changed to a new order that may be more suitable for their situation. The options being: Newest first, Alphabetical and Oldest first.
Output & Storage Requirements	The output is one of three options them being the order in which the array of documents are held in. For storage requirements all that is needed is the array that holds the documents and the documents themselves. AS the array is theoretically dynamic a determined size does not exist.
Data Used	The sort order, the array of documents

Evidence



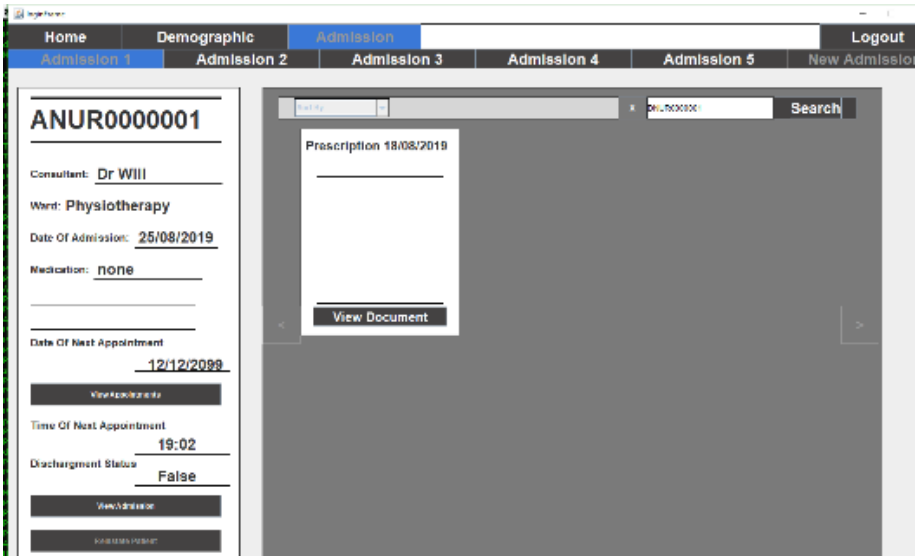
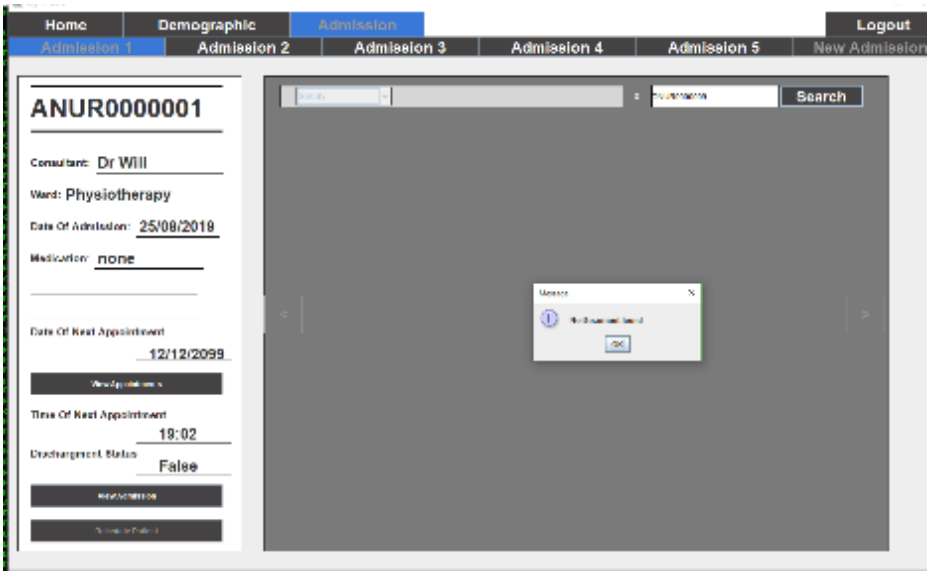
This is the





standard order that is used when initially loaded however there are also other options available to them as mentioned. The last image is the code where there are multiple different sorts depending on what was selected. The methods are called not from an action listener but from a focus listener

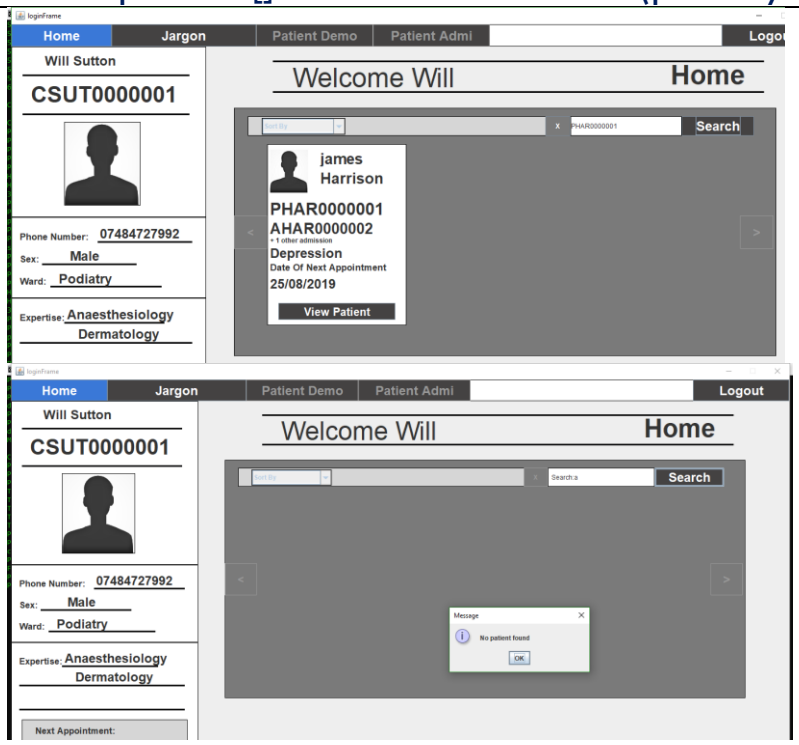
```
public Document[] sortlistofdocumentsArray(int order, Document[] array)
    {
        Document[] newOrderedArray = new Document[1];//declares the array the new values will be stored in
        int sizeOfArray = array.length;//finds length of array
        if(order ==0)//selection determining if they want it in ascending order
        {
            for (int i = 0; i < sizeOfArray; i++) //for loop that will iterate through every value
            {
                for (int j = i + 1; j < sizeOfArray; j++) //for loop that will iterate through every value
                {
                    if(array[i].docType.compareTo(array[j].docType)>0) //selection determining if the value goes af
                    {
                        newOrderedArray[0] = array[i];//temp value to store array
                        array[i] = array[j];//swap occurs
                        array[j] = newOrderedArray[0];//temp value is then inserted back into correct spot
                    }
                }
            }
            for (int i = 0; i < sizeOfArray; i++) //for loop that will iterate through every value
            {
                System.out.print(array[i].documentID);
            }
        }
        else if(order ==1)//selection determining if they want it in descending order
        {

        if(order ==2)//selection determining if they want it in ascending order
        {
        else if(order ==3)//selection determining if they want it in descending order
        {
        System.out.println("========================");
        for (int i = 0; i < sizeOfArray; i++) //for loop that will iterate through every value
            {
        return(array);//new list is returned
```

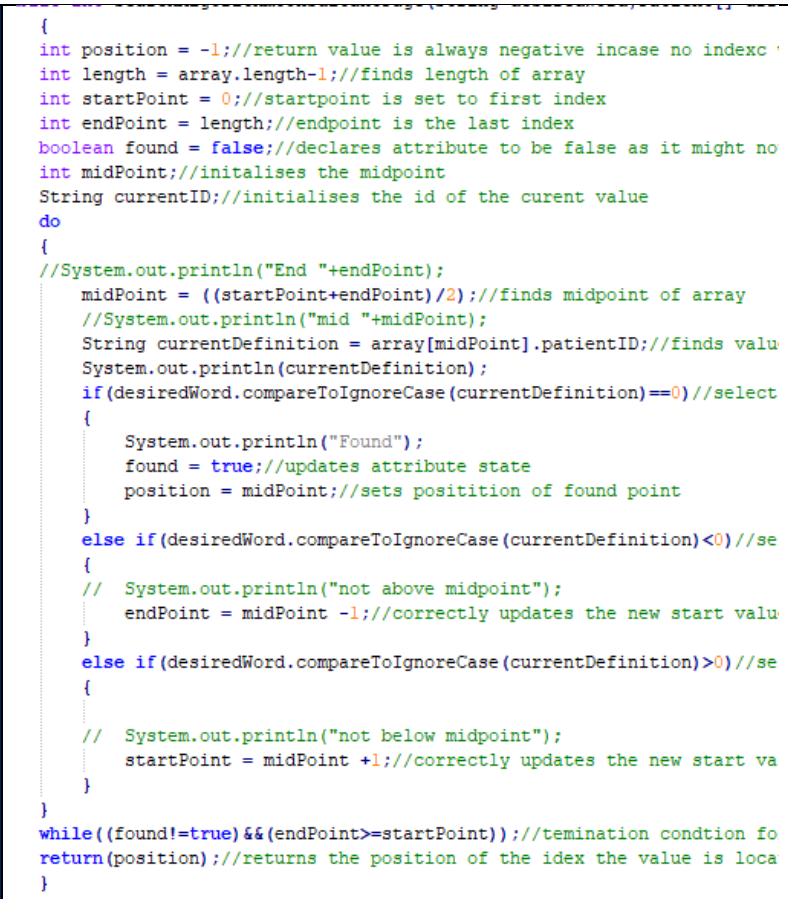
Feature	Searching Documents [26 B]
Description	When viewing the documents, the user may decide the best option to get to the document they want is to search for it using the primary key. This feature allows them to do exactly that.
Output & Storage Requirements	The output will be twofold either the document they were looking for or a popup informing them that the search value did not find a corresponding value. For storage requirements only two things are required the array of items and the search value nothing else is used.
Data Used	List of documents[] and the search value(primary key)
Evidence	 

Here are the two outputs either the desired item or an error output.

```
blic int searchAlgorithmAdmissionPage(String desiredWord,Document[] array)
  {
  int position = -1;//return value is always negative incase no indexc was fou
  int length = array.length-1;//finds length of array
  int startPoint = 0;//startpoint is set to first index
  int endPoint = length;//endpoint is the last index
  boolean found = false;//declares attribute to be false as it might not be fo
  int midPoint;//initalises the midpoint
  String currentID;//initialises the id of the curent value
  do
  {
  //System.out.println("End "+endPoint);
      midPoint = ((startPoint+endPoint)/2);//finds midpoint of array
      //System.out.println("mid "+midPoint);
      String currentDefinition = array[midPoint].documentID;//finds value at m
      System.out.println(currentDefinition);
      if(desiredWord.compareToIgnoreCase(currentDefinition)==0)//selection det
      {
          System.out.println("Found");
          found = true;//updates attribute state
          position = midPoint;//sets posititon of found point
      }
      else if(desiredWord.compareToIgnoreCase(currentDefinition)<0)//selection
      {
      //  System.out.println("not above midpoint");
          endPoint = midPoint -1;//correctly updates the new start value respe
      }
      else if(desiredWord.compareToIgnoreCase(currentDefinition)>0)//selection
      {

      //  System.out.println("not below midpoint");
          startPoint = midPoint +1;//correctly updates the new start value res
      }
  }
  while((found!=true)&&(endPoint>=startPoint));//temination condtion for the l
  return(position);//returns the position of the idex the value is located at
```

Here is the algorithm, it uses a basic binary search to find the index of the desired document.

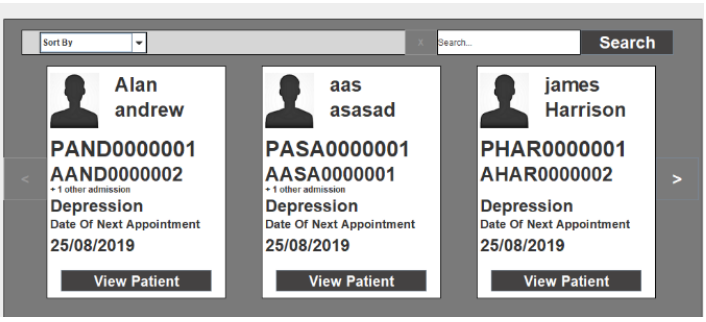
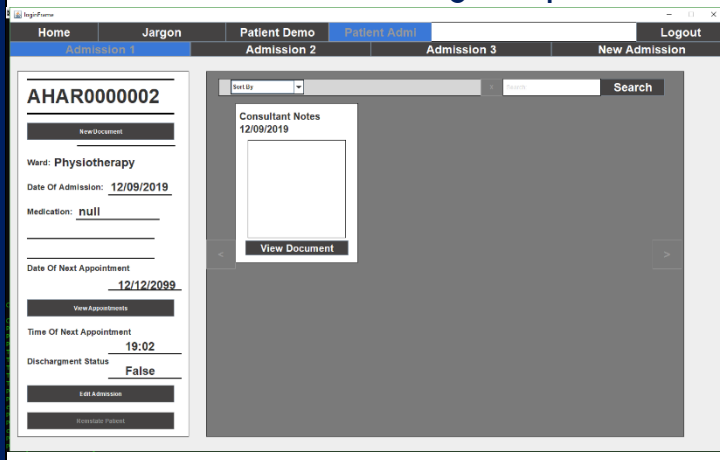
Feature	Searching for patients [28]
Description	When viewing the patients, the consultant may decide the best option to get to the patient they want is to search for it using the primary key. This feature allows them to do exactly that.
Output & Storage Requirements	The output will be twofold either the patient they were looking for or a popup informing them that the search value did not find a corresponding value. For storage requirements only two things are required the array of items and the search value nothing else is used.
Data Used	List of patients[] and the search value(primary key)
Evidence	 Here are the two outputs either the desired item or an error output.

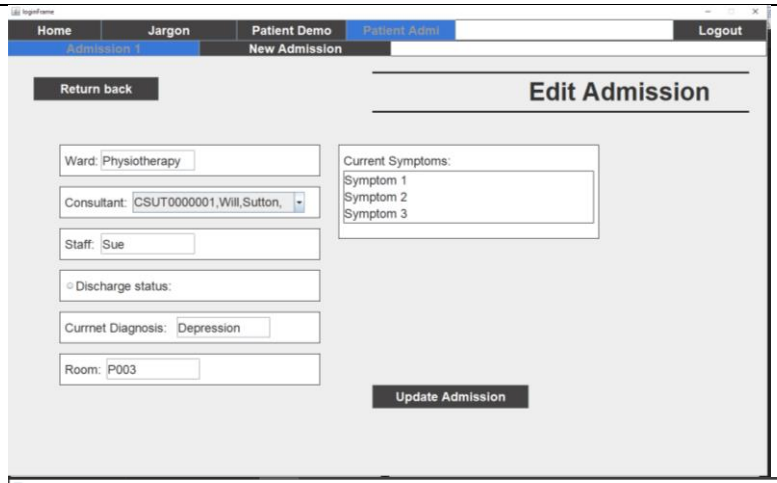


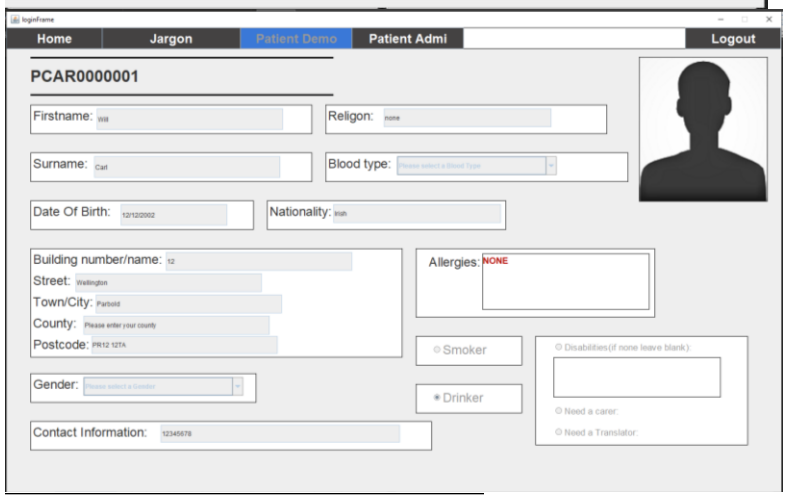
```
{
int position = -1;//return value is always negative incase no indexc
int length = array.length-1;//finds length of array
int startPoint = 0;//startpoint is set to first index
int endPoint = length;//endpoint is the last index
boolean found = false;//declares attribute to be false as it might no
int midPoint;//initalises the midpoint
String currentID;//initialises the id of the curent value
do
{
//System.out.println("End "+endPoint);
    midPoint = ((startPoint+endPoint)/2);//finds midpoint of array
    //System.out.println("mid "+midPoint);
    String currentDefinition = array[midPoint].patientID;//finds valu
    System.out.println(currentDefinition);
    if(desiredWord.compareToIgnoreCase(currentDefinition)==0)//select
    {
        System.out.println("Found");
        found = true;//updates attribute state
        position = midPoint;//sets positition of found point
    }
    else if(desiredWord.compareToIgnoreCase(currentDefinition)<0)//se
    {
    //  System.out.println("not above midpoint");
        endPoint = midPoint -1;//correctly updates the new start valu
    }
    else if(desiredWord.compareToIgnoreCase(currentDefinition)>0)//se
    {

    //  System.out.println("not below midpoint");
        startPoint = midPoint +1;//correctly updates the new start va
    }
}
while((found!=true)&&(endPoint>=startPoint));//temination condtion fo
return(position);//returns the position of the idex the value is loca
}
```

Here is the algorithm, it uses a basic binary search to find the index of the desired patient..

Feature	**The consultant can view patient files [29]**
Description	In order for the consultant to actually add documents to the patient admissions, they will need to gain access and find the admission in the first place. Because of this they will need to be able to view the majority of the patients file, this would be a combination of the admissions and demographic information respectively
Output & Storage Requirements	The output would be the assigning of data to text fields and creating of cards such as those to the admissions. For temporary storage these have been assigned to one entity which uses hierarchical inheritance which allows other entities such as a list of admissions to be attached to the object. When the system is not on however all data is permanently held in a text file.

Data Used	For the demographic: patientID,surName,firstName,addressHouseNum ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, numberOfAdmissons,dob,religion,allergies, gender,disability,carer,translator For the admissions: admissionID,ward,consultantName, dateOfNextAppointmentA,active,numberOfDocuments ,medication,room,currentDiagnosis,stringedSymptoms, staffName, dateAdmissionCreated,admissionsStaffID admissionsConsultantID.
Evidence	 Here are screenshots of the consultant home screen and then the resultant screen of viewing the patients file. In the  second image the button next to the currently selected one will bring up the demographic panel where they can view the information. On the contact bar the consultant can also press the edit admission button allowing them to view and alter the admission information.



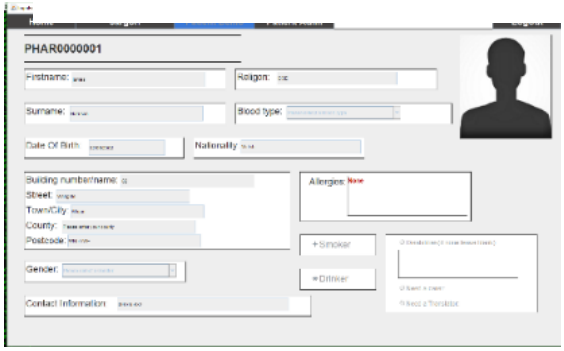



```
if(e.getSource()==consultantPatientDemographic)
{
    consultantPatientDemographicPanel();//creates des:
}
if(e.getSource()==consultantPatientAdmission)
{
    createConsultantadmissionHomepagePanelGUI();//crea
    createTopbarAdmission(consultantAdmissionPanel);//,
}
```

```
if(e.getSource()==viewPatient1bttn)
{
consultantPatient = listOfCsPatients[0+currentPageIndexConsultant];
currentPatient = consultantPatient;
currentPatientConsultantIndex =0+currentPageIndexConsultant;
admissionList=listofAdmissions[0+currentPageIndexConsultant];
currentAdmission = listofAdmissions[0+currentPageIndexConsultant][0];
int lengthAdmission = 0;
for(int i= 0;admissionList[i]!=null;i++)
{
    lengthAdmission++;
    admissionList[i].retrieveDocuments(consultantPatient,admissionList[i]);
}
numberOfAdmissions = lengthAdmission;

listOfdocumentsGUI = currentAdmission.listOfDocuments;
numberOfDocuments =currentAdmission.numberOfDocuments;
singleDefinition = new Document[numberOfDocuments];
createButtonTopBarAdmission();
setActiveAdmissionTopPanelBttn(topBarAdmission1Bttn);
createConsultantadmissionHomepagePanelGUI();//calls method which displays pane
setActiveTopPanelBttn(consultantPatientAdmission);
consultantPatientAdmission.setEnabled(true);//the component is enabled to inte
consultantPatientDemographic.setEnabled(true);//the component is enabled to ir
}
```

	In the bottom screenshot, it shows the code for how a patient is retrieved, what happens is the current patient is set to whatever index the patient resides in on the array. Then their corresponding list of admissions are then saved to the global list. Using this array for each admission the list of documents are then retrieved respective to the admission they are attached to. Finally, the rest of the lines are updating any text field that would have any attributes that belong to any older account that was accessed. Then the homepage is ran.

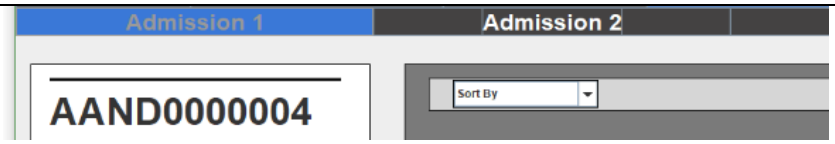
Feature	Consultants can view patient demographic information [30]
Description	This feature was designed to work in hand with the prior objective, as when the consultant retrieves the desired patient, the system will then assign the attributes attached to the entity to the text fields and then allow the consultant to only view them
Output & Storage Requirements	The output will be the fact that the consultant can view their patient's demographic information and the assigning of attributes to the correct JComponent. For storage requirements the record data structure is used to hold all the different data types assigned to the entity.
Data Used	patientID,surName,firstName,addressHouseNum ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, numberOfAdmissons,dob,religion,allergies, gender,disability,carer,translator

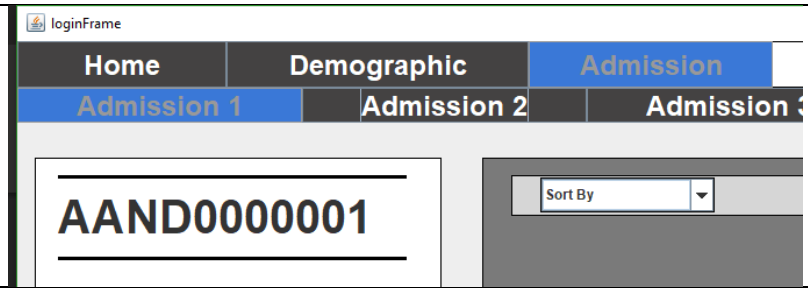
Evidence	((loginChoice==3)||(loginChoice==1))//selection determining if the user is the ma
	
	demographicDrinkerPromptLbl.setEnabled(false);//the component is disabled
	demographicSmokerPromptLbl.setEnabled(false);//the component is disabled
	demographicDisablitiesPromptLbl.setEnabled(false);//the component is disabled
	demographicCarerPromptLbl.setEnabled(false);//the component is disabled
	demographicTranslatorPromptLbl.setEnabled(false);//the component is disabled
	demographicFNameTextFeild.setEditable(false);//the text will be unable to be ed
	demographicSNTextFeild.setEditable(false);//the text will be unable to be edite
	demographicDOBTextFeild.setEditable(false);//the text will be unable to be edit
	demographicBuildingNUMTextFeild.setEditable(false);//the text will be unable to
	demographicStreetTextFeild.setEditable(false);//the text will be unable to be e
	demographicTownTextFeild.setEditable(false);//the text will be unable to be edi
	demographicCountyTextFeild.setEditable(false);//the text will be unable to be e
	demographicPostcode.setEditable(false);//the text will be unable to be edited b
	demographicGender.setEnabled(false);//the component is disabled
	demographicContactinfo.setEditable(false);//the text will be unable to be edite
	demographicAllergies.setEditable(false);//the text will be unable to be edited
	demographicNotioanlityTextFeild.setEditable(false);//the text will be unable to
	demographicBloodType.setEnabled(false);//the component is disabled
	demographicReligonTextFeild.setEditable(false);//the text will be unable to be
	demographicDisablitiesTextFeild.setEditable(false);//the text will be unable to
	
	 
	
	This top screenshot is used to prevent the consultant from entering and altering any information regarding the patient. The next image is the demographic page,

```
demographicFNameTextFeild.setText(currentPatient.firstName);//text is added to the comp
demographicSNTextFeild.setText(currentPatient.surName);//text is added to the component
demographicDOBTextFeild.setText(ft.format(currentPatient.dob));//text is added to the c
demographicBuildingNUMTextFeild.setText(currentPatient.addressHouseNum+"");//text is ad
demographicStreetTextFeild.setText(currentPatient.addressHouseStreet);//text is added t
demographicTownTextFeild.setText(currentPatient.town);//text is added to the component
demographicPostcode.setText(currentPatient.postcode);//text is added to the component
System.out.println(currentPatient.gender);//text is added to the component
demographicGender.setSelectedItem(currentPatient.gender);//text is added to the compone
demographicContactinfo.setText(currentPatient.contactNum);//text is added to the compon
demographicAllergies.setText(currentPatient.allergies);//text is added to the component
demographicNotioanlityTextFeild.setText(currentPatient.nationality);//text is added to
demographicBloodType.setSelectedItem(currentPatient.bloodType);//text is added to the c
demographicReligonTextFeild.setText(currentPatient.religon);//text is added to the comp
demographicDisablitiesTextFeild.setText(currentPatient.disability);//text is added to t
```

where the actual data is showed to the user, by boxing up the information and separating the information it creates a tidier look to the page. Finally, the last image is the code which actually assigns the information to the components

Feature	Sort admission [31]
Description	When the admission list is retrieved from file an insertion sort used to order the admission list from the serial order used in the file into the order of oldest first.
Output & Storage Requirements	The output varies for a patient the order is in oldest first due to the serial nature they were written to file. The same can be said for the consultant despite the fact not all of the patient's admissions will be for them resulting in gaps being created. For storage they are held in an array.
Data Used	A list of admissions.

Evidence	 As you can see the first admission button is that of the 4 admission, this happens due to the order they were accessed last. However, for the patient
	`for (int i = 0; i < admissionList.length; i++) //for loop that will iterate through e`
	`{`
	`for (int j = i + 1; j < admissionList.length; j++) //for loop that will iterate t`
	`{`
	`if(admissionList[i].admissionID.compareTo(admissionList[j].admissionID)>0) //s`
	`{`
	`Admission newOrderedArray = admissionList[i];//temp value to store array`
	`admissionList[i] = admissionList[j];//swap occurs`
	`admissionList[j] = newOrderedArray;//temp value is then inserted back into`
	`}`
	`}`
	`}`
	it is order from oldest first:
	All it is a standard insertion sort which finds the correct location to insert the admission, while the sort itself isn't anything special the selection condition determines the order here we can change the order in which items would be ordered through the primary key order(date or id) or even if it has be ascending or descending

Feature	**Edit Prescriptions from the document end of the system[32]**
Description	In order for a new prescription or in this case to edit one, the consultant will create a new document for the prescription and enter the 4 main fields of the document. Once the document has been created. Next time the patient views the documents for that admission, they will see that a new prescription document has been created.
Output & Storage Requirements	The output is the generation of a new document specifically a prescription. For temporary storage it is stored as a document object which is then stored as an item in the list of documents on the admission object which is then stored in an array of admissions being an attribute of the patient. For permanent storage it is held on its own document for admissions of that type, I will eventually change it so that one patient has one document.
Data Used	dateOfDocument,dateOfDocumentCreation ,documentID,medicationName, medicationDosage,medicationIntakeTime, medicationDateOfNextDispatch

Evidence

PAND0000001 | 17/09/2019

AAND0000004 | 13:39

DAND0000001

Prescription

The consultant CSUT0000001has wished that the patient under care needs to receive a prescription. Because of this the following Prescription has been requested for the patient PAND0000001

Medication: Paracetamol

Dosage: 0.5g

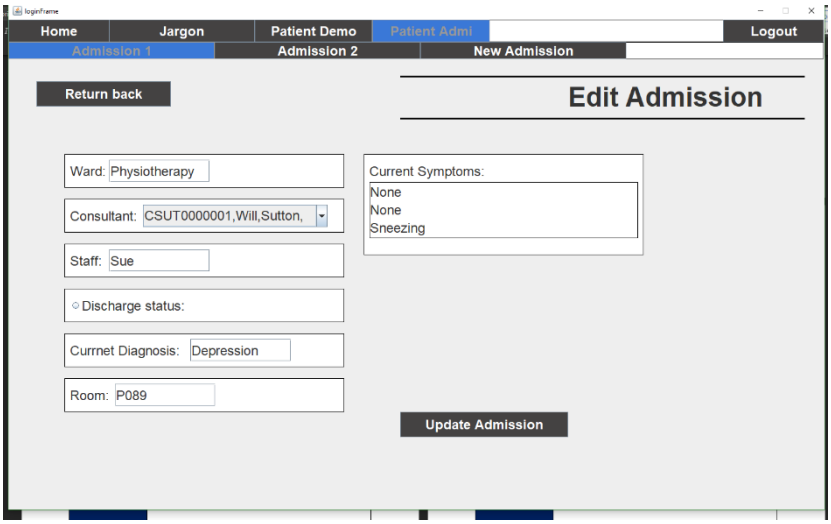
Intake Time: 12:00

Date of next dispatch: 17/09/2019

What we can see here is the prescription document that has just been created as you can see all the attributes have been declared and this can be seen from the patient. When they log on they will see that they have a new document and then view the prescription for them to realise that it has changed from last time. The code shows how the specific type of document has been created, with assigning the correct attributes and the type of document itself

```
if(currentDocument.docType =="Prescription")//selection determining if the current docum
{
    actionPerformed = 0;//sets the action to this value

    tempDocument.medicationName =medicationDocumentTA.getText();//retrievs text from com
    tempDocument.medicationDosage = dosageDocumentTA.getText();//retrievs text from comp
    tempDocument.medicationIntakeTime =intakeTimeDocumentTA.getText();//retrievs text fr
    try{
    tempDocument.medicationDateOfNextDispatch = ftTimeInc.parse(DONDDocumentTA.getText()
    }
    catch(Exception exc)
    {
    }
    stringedDocument = tempDocument.documentID+","+ftTimeInc.format(actionCreated)+","+"
}
currentAdmission.numberOfDocuments++;//increments the number of docuemtns the admission
manager.writeActionToFile(currentConsultant,tempDocument,currentAdmission,currentPatient
try{
```

Feature	Add/Edit Admission information [33/34]
Description	When the consultant feels like that there are any missing fields to the data they can add onto the admission the field containing any blank data. Once the consultant presses save it will then be added. The system works with documents by assigning the object with every attribute possible however to prevent null pointer exceptions the type of document determines what actual fields are actually used so adding data would look like this: ANUR0000001,,12/09/2019 to ANUR0000001,Prescription,12/09/2019 as no old data exists the gap between admission id and document creation is empty, with editing it would look like this: ANUR0000001,Notes,12/09/2019 to ANUR0000001,Prescription,12/09/2019. As you can see no gap exists.
Output & Storage Requirements	The output would be the writing and then reading of the file containing the new admission information that being edited or brand-new data. For storage requirements as it revolves around an entity an object is used . And to store data while the system is not in use a text file is used with every line being a new admission
Data Used	admissionID,ward,consultantName, dateOfNextAppointmentA,active, numberOfDocuments ,medication,room,currentDiagnosis, stringedSymptoms,staffName, dateAdmissionCreated,admissionsStaffID admissionsConsultantID

Evidence(explained)	The top image is the screen shot of the admission page, where the adding/editing of the admission occurs. Below is the code for amending the admission. The first large section is used for discharging the admission, the  code here just is the same as the creation of a normal document, what actually happens is that a discharging document is created here. Next we see the updating of the actual fields themselves using the data from the components. Finally, we see a special feature which allows the consultant to change the consultant attached to the admission. By removing the current admission and the Item in the file and adding it to the new one

Evidence

```
    catch(Exception exc)
    {
        System.out.println("read error");//informs of error
    }
}
currentAdmission.ward = wardAdmissionTextField.getText();//retrieves attribute from component
currentAdmission.staffName = StaffsNameTextField.getText();//retrieves attribute from component
currentAdmission.room = roomTextField.getText();//retrieves attribute from component
currentAdmission.currentDiagnosis = currentDiagnosisTextField.getText();//retrieves attribute from componer
//System.out.println(oldAdmissionInfo.room);
String symptomLine = "";//declares an empty string
String[] symptomsStringed  = listOfCurrnetSymptomsTextPane.getText().split("\\n");//splits line up
Admission newAdmissionInfo = currentAdmission;//declares the new admission to the newly updated admission
for(int count =0;count<currentAdmission.listOfSymptoms.length;count++)//runs a for loop that iterates throu
{
    currentAdmission.listOfSymptoms[count] = symptomsStringed[count];//assigns attribute from list that was
}
listOfCurrnetSymptomsTextPane.setText(currentAdmission.listOfSymptoms[0]+"\n"+currentAdmission.listOfSympto
    if(staffNameTextField.getSelectedItem()!=currentConsultant.consultantID +","+currentConsultant.firstNam
    {
        actionPerformed = 7;
        Document tempDocument = new Document();//creates new instance of this object
        ConsultantList cl = new ConsultantList();//creates new instance of this object
        String newConsultantBasicInfo = (String) staffNameTextField.getSelectedItem();//retrieves attribute
        String[] splitNewConsultantBasicInfo =  cl.unConcatenateStringConsultanttxt(newConsultantBasicInfo,
        int numberOfPatients = currentConsultant.numberOfPatients;//finds out how mant patients the current
        currentConsultant.numberOfPatients = currentConsultant.removepatientToConsultantRecord(currentAdmis
        currentAdmission.admissionsConsultantID = splitNewConsultantBasicInfo[0];//assigns attribute from a
        currentAdmission.consultantName = splitNewConsultantBasicInfo[1];//assigns attribute from array
            String tempConsultantID = (String) staffNameTextField.getSelectedItem();//retrieves attribute f
            currentConsultant.newConsultantID = tempConsultantID.substring(0,11);//substribngs the string t
        for(int count = 0;count<listOfAllConsultants.length;count++)//runs a for loop for the number of con
        {if((tempConsultantID.substring(0,11) == listOfAllConsultants[count].consultantID))//checks to see
            {locationOfReplacementConsultant = count;//sets postition of the desired consultant }}
        listOfAllConsultants[locationOfReplacementConsultant] = currentConsultant.updatePatientAdmissionToC
        manager.writeActionToFile(currentConsultant,tempDocument,currentAdmission,currentPatient,actionPerf
        for(int index = 0;index<numberOfAdmissions;index++)
        {System.out.println(listofAdmissions[currentPatientConsultantIndex][index]);    }
        for(int index = numberOfAdmissions;index>indexofadmission;index--)
        {listofAdmissions[currentPatientConsultantIndex][index-1] =listofAdmissions[currentPatientConsultan
        numberOfAdmissions--;
        for(int index = numberOfAdmissions;index>indexofadmission;index--){
            listofAdmissions[currentPatientConsultantIndex][index-1] =listofAdmissions[currentPatientConsul
        }admissionList =listofAdmissions[currentPatientConsultantIndex];}
    //System.out.println(oldAdmissionInfo.room);
    actionPerformed = 6;//sets the action performed by the consultant
    Document tempDocument = new Document();//creates new instance of this object
    manager.writeActionToFile7(currentConsultant,tempDocument,oldAdmissionInfo,currentPatient,actionPerfor
    al.updatePatientAdmission(currentPatient,currentAdmission);//updates the patient information
    refreshConsultantInfo();//retireives the new updates information for that consultant
    loaded[6][2] = false;//the variable is set as true to prevent the components from being reran
    createTopbarAdmission(consultantAdmissionPanel);//the top multiuse bar is then created, this is isolate
    createConsultantHomepagePanelGUI(currentConsultant);//creates the correct panel
    createButtonTopBarAdmission();
```

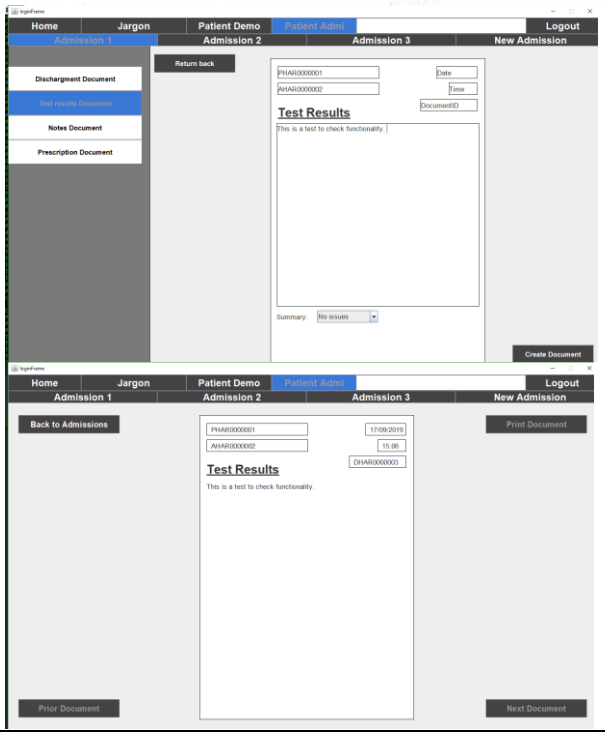
```
f(e.getSource()==saveChangesToAdmisionBttn)

   AdmissionList al = new AdmissionList();//creates new instance of this object
   oldAdmissionInfo=currentConsultant.copyAttributesAdmission(currentAdmission);//a

f(dischargeLblPrompt.isSelected()==true)//selection determining if the consultant w

   currentAdmission.active=false;//discharges the admission
   String stringedDocument="";//declares an empty string
   Document tempDocument = new Document();//creates new instance of this object
   actionPerformed = 1;//sets the action as dischraging the patient
   tempDocument.documentID = tempDocument.createUniqueID(currentPatient.surName,cur
   stringedDocument = tempDocument.documentID+","+ftTimeInc.format(new Date())+","+
   currentAdmission.numberOfDocuments++;//increments the number of documents the pa
   try{

      FileReader fra = new FileReader(currentPatient.patientID+"_"+currentAdmissio
      BufferedReader bra = new BufferedReader(fra);//declares a new bufferedreader
      String lineOfDataa = bra.readLine();//saves the line read from file to the v
      int line = 0;
      while(lineOfDataa !=null)//a while loop that runs untill the variable read i
      {
         line++;//increments line
         lineOfDataa = bra.readLine();//this then reads from file again to check
      }
      //System.out.println("Step 1");
         String[] arrayofDocuments = new String[line+1];//creates a new array wit
         FileReader fr = new FileReader(currentPatient.patientID+"_"+currentAdmis
      BufferedReader br = new BufferedReader(fr);//declares a new bufferedreader c
      String lineOfData = br.readLine();//saves the line read from file to the var
      //writes every line of file before desired locatio to the array
      for(int count=0;count<arrayofDocuments.length-1;count++)//declares a of loop
      {
         arrayofDocuments[count] = lineOfData;//saves the line read from file to

         //System.out.println(arrayofDocuments[count]);
         lineOfData = br.readLine();//this then reads from file again to check if
      }
      //System.out.println("Step 2");
         arrayofDocuments[arrayofDocuments.length-1] = stringedDocument;
         //adds the new admission
         //System.out.println(arrayofDocuments[arrayofDocuments.length-1]+"HEll")
         //adds rest to array
         FileWriter fw = new FileWriter(currentPatient.patientID+"_"+currentAdmis
      for(int counter = 0;counter<arrayofDocuments.length;counter++)//declares a o
      {
         fw.write(arrayofDocuments[counter]);//writes the line to file
         fw.write("\r\n");//adds in a space to the file
      }
      //System.out.println("Step 3");
```

Feature	Generating new documents [35]
Description	The function of this feature is to allow the consultant to create documents, besides having the ability to view existing documents this is the most important feature of the system and so it needs to work flawlessly.
Output & Storage Requirements	The output is then writing the document to file. For actual use the system stores this data as an instance of document and then is saved to the list of documents attached to the admission entity. Currently it is stored in a file with other documents of the admission. However, for the final system it is likely to be moved to a single file.
Data Used	dateOfDocument,dateOfDocumentCreation, documentID,hospital,notes,testResults medicationName,medicationDosage, medicationIntakeTime,summary, medicationDateOfNextDispatch
Evidence explained	In the screenshots we can see how the documents are created. Initially the document is created as a new instance and then all the general information is created . In the next screenshot the data of the document is actually written to file in the correct location, and then increments the number of documents the admission has. Finally, the list of documents for the admission is retrieved and then the user is returned back to the homepage with the new document.

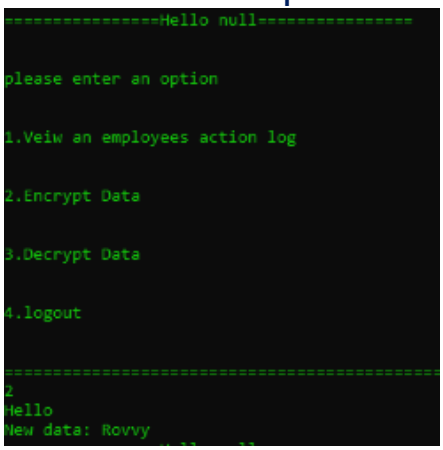
Evidence

```
if(e.getSource()==createNewDocumentButton)
{
    listOfdocumentsGUI = sortlistofdocumentsArray(2,listOfdocumentsGUI);//sorts
    currentAdmission.listOfDocuments=listOfdocumentsGUI;
    if(reinstatePatient==true)//selection determining if the patient has been r
    {currentAdmission.active=true;//sets current admisison as active agian
    }Document tempDocument = new Document();//creates a new instance of documen
    tempDocument.documentID = tempDocument.createUniqueID(currentPatient.surNam
    String stringedDocument="";//declares empty variable
    Date actionCreated = new Date();//creates new instance of this object
    tempDocument.dateOfDocumentCreation = actionCreated;//assigns the current d
    if(currentDocument.docType =="Consultant Notes")//selection determining if
    {
```

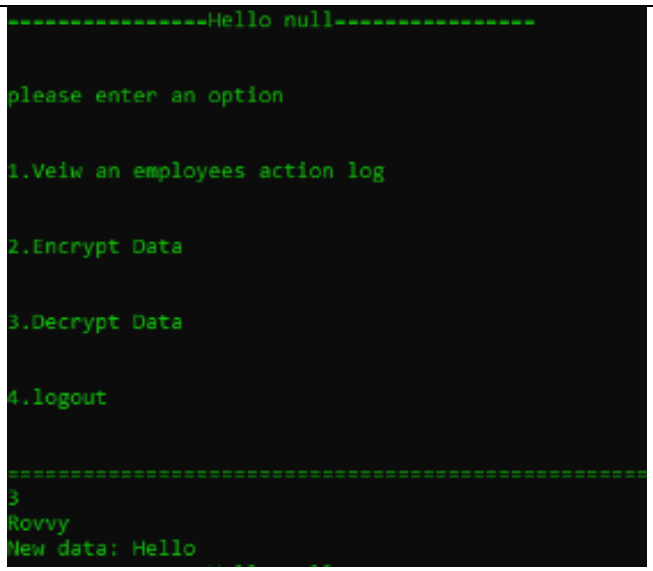
```
manager.writeActionToFile(currentConsultant,tempDocument,currentAdmission,currentPatient,actionPerformed);//writes act
try{FileReader fra = new FileReader(currentPatient.patientID+"_"+currentAdmission.admissionID+"_Documentation.txt");//
    BufferedReader bra = new BufferedReader(fra);//declares a new bufferedreader called br using fr as a parameter to
    String lineOfDataa = bra.readLine();//saves the line read from file to the variable
    int line = 0;
    while(lineOfDataa !=null)//a while loop that runs untill the variable read is not empty(The end of the array is no
    (line++;//increments variable
        lineOfDataa = bra.readLine();//this then reads from file again to check if not empty
    }String[] arrayofDocuments = new String[line+1];//declares array with a size +1
    FileReader fr = new FileReader(currentPatient.patientID+"_"+currentAdmission.admissionID+"_Documentation.txt");//d
    BufferedReader br = new BufferedReader(fr);//declares a new bufferedreader called br using fr as a parameter to be
    String lineOfData = br.readLine();//saves the line read from file to the variable
    for(int count=0;count<arrayofDocuments.length-1;count++)//for loop that runs for the number of documtns of the ol
    {
        arrayofDocuments[count] = lineOfData;//assigns new line to latest line read

        lineOfData = br.readLine();//this then reads from file again to check if not empty
    }
        arrayofDocuments[arrayofDocuments.length-1] = stringedDocument;//adds newest document back to the  array
        FileWriter fw = new FileWriter(currentPatient.patientID+"_"+currentAdmission.admissionID+"_Documentation.txt")
    for(int counter = 0;counter<arrayofDocuments.length;counter++)//for loop that runs for the number of documtns of
    {
        fw.write(arrayofDocuments[counter]);//adds document to file
        fw.write("\r\n");//adds in a space to the file
    }
    fw.close();//closes the file
    AdmissionList al = new AdmissionList();//creates new instance of object
    al.updatePatientAdmission(currentPatient,currentAdmission);//updates patient information
refreshConsultantInfo();
updatePatient();
currentAdmission.retrieveDocuments(consultantPatient,currentAdmission);
listOfdocumentsGUI = currentAdmission.listOfDocuments;
numberOfDocuments =currentAdmission.numberOfDocuments;
singleDefinition = new Document[numberOfDocuments];
createButtonTopBarAdmission();
setActiveAdmissionTopPanelBttn(topBarAdmissionlBttn);
createConsultantadmissionHomepagePanelGUI();//creates the correct panel
createContactBarAdmissionGeneral();//creates the correct panel
```



Feature	Encrypting data [36]
Description	The purpose of this feature is to scramble data that is passed through the system. This is to allow security when reading it from file and only the people who know the encryption cypher can have access to the data. Currently this is used for a test of application just to check it works so I can still the contents of the file while I am developing the system.
Output & Storage Requirements	The output is the scrambled version of the string that was entered. As of now it is just the character that is +10 in the ascii values.
Data Used	No data is retrieved all data supplied is by the user. For the actual use of the feature it will be whatever is written to file.
Evidence	In the first image we see the method in action outputting the data returned from the method. The next one is the option to run the method. Finally, the last one is the cypher itself which just moves the ascii value by 10, to go the other way instead we subtract 10 to get the original character




```
public void encryptData()
{
    String stringToBeEncrypted = inputScanner.nextLine();
    System.out.println("New data: "+encrypt(stringToBeEncrypted));

public String encrypt(String string)//declares the encrypt
{
    String encryptedValue="";//emptys out any previous
    int length = string.length();//sets the length of the
    for (int i = 0; i<length;i++)//runs a for loop for ev
    {
        char letter = string.charAt(i);//finds the single
        int letterASCII = (int)letter+10;//finds the asci
        char newletter = (char)letterASCII;//finds the ne
        encryptedValue = encryptedValue + newletter;//add
    }
    return encryptedValue;//returns the value
```

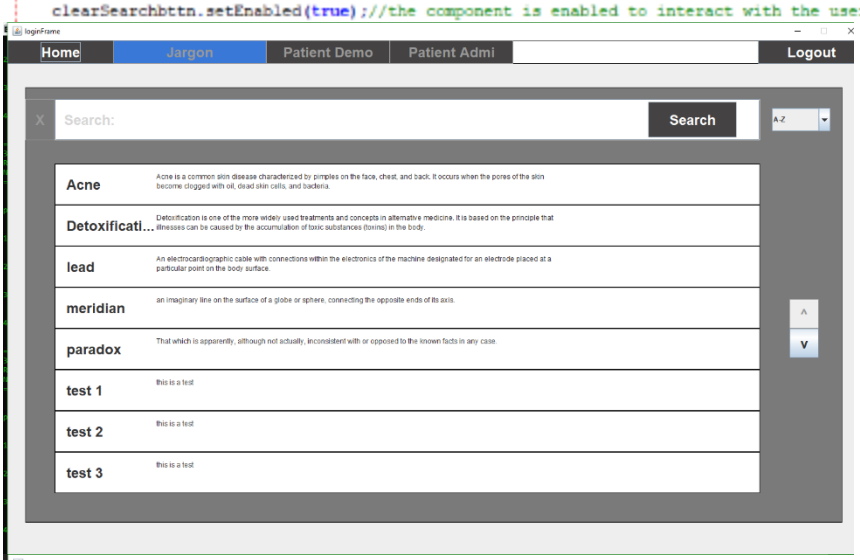
Feature	Decrypting data [37]
Description	The purpose of this feature is to unscramble data that is passed through the system. This is to allow security when reading it from file and only the people who know the encryption cypher can have access to the data. Currently this is used for a test of application just to check it works so I can still the contents of the file while I am developing the system.
Output & Storage Requirements	The output is the unscrambled version of the string that was entered. As of now it is just the character that is -10 in the ascii values
Data Used	No data is retrieved all data supplied is by the user. For the actual use of the feature it will be whatever is read from file.
Evidence	 In the first image we see the method in action outputting the data returned from the method. The next one is the option to run the method. Finally, the last one is the cypher itself which just moves the ascii value by

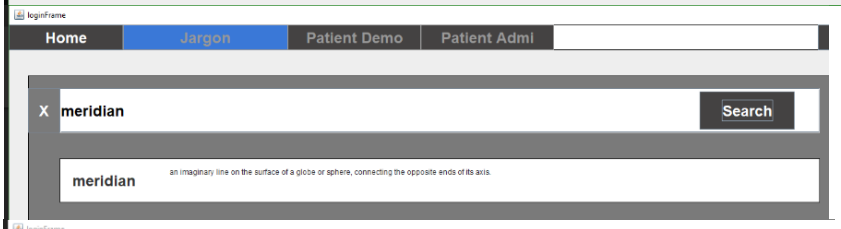
```
public void decryptData()
{
    String stringToBeDecrypted = inputScanner.nextLine();
    System.out.println("New data: "+decrypt(stringToBeDecrypted));

}
public String decrypt(String string)//runs the decrypt procedure
{
    String newString="";//empties the variable
    int length = string.length();//finds the length of the string
    for (int i = 0; i<length;i++)//runs a for loop for the length
    {
        char letter = string.charAt(i);//finds an individual char
        int letterASCII = (int)letter-10;//-10, which will find t
        char newletter = (char)letterASCII;//converts the ascii v
        newString = newString + newletter;//concatines so that at
    }
    return newString;//returns the new decrypted balance
}
```

	10 and concatenates it to an empty string which is then returned back to the user

Feature	Using the jargon library [38]
Description	This feature works by having the user enter into the field the word they want a description for, it will then return one of two items either an output saying it was not found or the description
Output & Storage Requirements	The output is twofold, either the item is found(the definition will be returned, or the item was not present in the array and an output informing them of this is shown.) For storage requirements a 2D array is used to hold the word along with the definition, with a row size of 2. And a dynamic size for the number of columns as the size will vary over time as more and more definitions get added.
Data Used	A 2D array containing the following: [word][Definition]
Evidence	How it works is that using the array of values the list is initially in a sorted state and then used in a binary search which returns the index of which the definition is located if it is not there it returns a one. Then using the first card the contents of the array of definitions at the index are attached to the components allowing them to be outputted to the user. The second is a screenshot of the jargon library, with all the definitions. The last two are of the search system one has a result in the database and the last one is an invalid query.

```
if(e.getSource()==searchLibraryBttn)
{
    sortBoxEmployeeAdmissionPanel.setEnabled(false);//the component is disabled
    int indexOfDefinition = searchAlgorithm(searchBarJargon.getText(),listOfDefinitions);//
    if(indexOfDefinition>=0)//selection determining if there at least one
    {
        String[] singleDefinition = listOfDefinitions[indexOfDefinition];//assigns the lis
        formatJargonCards(singleDefinition);//formats cards
        upButton.setEnabled(false);//the component is disabled
    downButton.setEnabled(false);//the component is disabled
    }
    else{
    jargonHideDef1();//hides definition card
    jargonHideDef2();//hides definition card
    jargonHideDef3();//hides definition card
    jargonHideDef4();//hides definition card
    jargonHideDef5();//hides definition card
    jargonHideDef6();//hides definition card
    jargonHideDef7();//hides definition card
    jargonHideDef8();//hides definition card
    upButton.setEnabled(false);//the component is disabled
    downButton.setEnabled(false);//the component is disabled
    JOptionPane.showMessageDialog(null, "No definition found");

    }
    clearSearchbttn.setEnabled(true);//the component is enabled to interact with the user
```

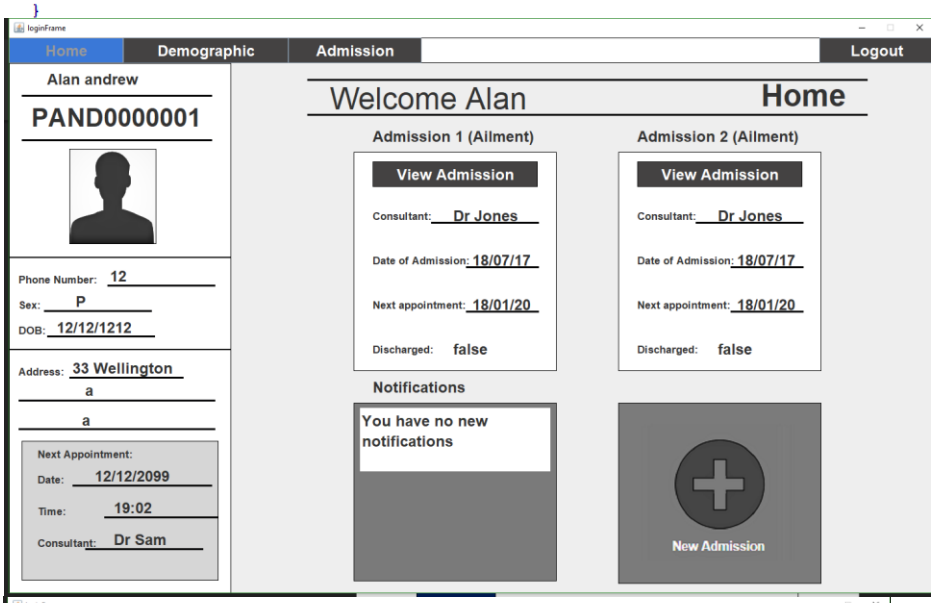


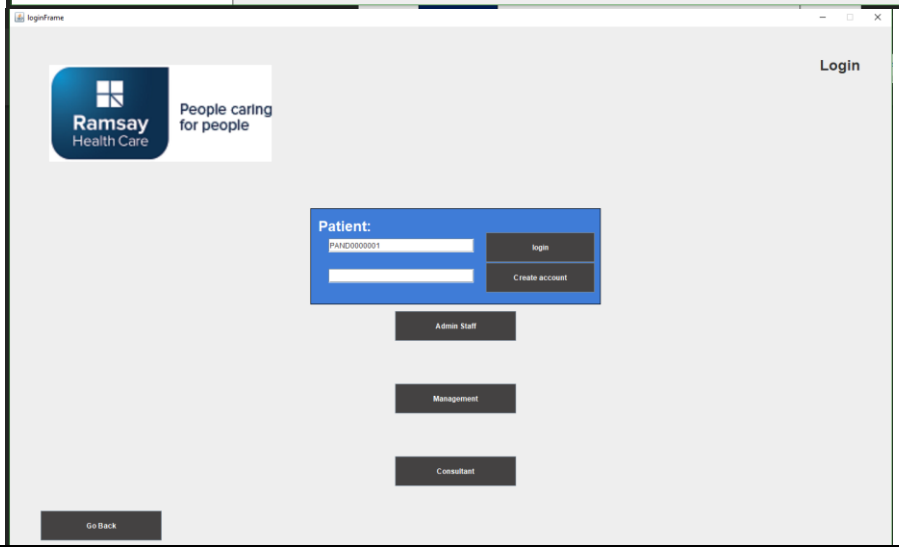


Feature	Searching through demographic information [41]
Description	When the initial loading of the patient, the system needs to return all the information regarding the correct patient. It works as the primary key is always the first 11 characters of the string allowing for the substring command to return the id no matter what value is entered. The list of demographic information is searched using a search which returns the index of the desired patient. Once found the information is retrieved and set to an instance of patient and loads the rest of their information.
Output & Storage Requirements	The output is the returning of the patient after pressing login. The storage is a record that is used to hold an object of patient which holds multiple instances of its child classes through the use of array data structures.
Data Used	The data used would be : patientID,surName,firstName,addressHouseNum ,addressHouseStreet,town,postcode,contactNum, nationality,bloodType,smoker,drinker, numberOfAdmissons,dob,religion,allergies, gender,disability,carer,translator

Evidence

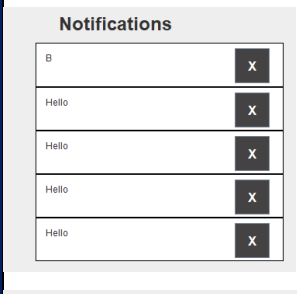
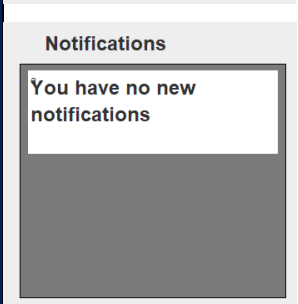
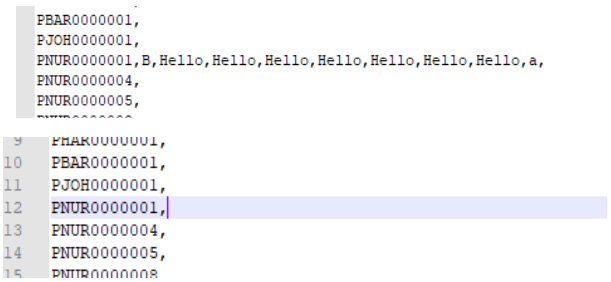
```
public void setUpPatientObj()
{
    date = false;//sets that no date for the patient has been set
    //creates a new patient
    currentPatient = new Patient();
    currentPatient = currentPatient.retrievePatientInfo(username);
    //System.out.println("PatientID "+ currentPatient.patientID);
    //retireves their notfications
    notifications =currentPatient.getNotifications(currentPatient.patientID);
    numberOfNotifications = notifications.length;
    currentPatient.updateNotificationsPatient(notifications);
    //finds number of admissions
    numberOfAdmissions = currentPatient.numberOfAdmissions;
    //System.out.println(numberOfAdmissions);
    admissionList=currentPatient.retrieveAdmissions(currentPatient);
    for (int i = 0; i < admissionList.length; i++)  //for loop that will iterate through every
            {
                for (int j = i + 1; j < admissionList.length; j++)  //for loop that will iterat
                {
                    if(admissionList[i].admissionID.compareTo(admissionList[j].admissionID)>0)
                    {
                        Admission newOrderedArray = admissionList[i];//temp value to store arra
                        admissionList[i] = admissionList[j];//swap occurs
                        admissionList[j] = newOrderedArray;//temp value is then inserted back i
                    }
                }
            }

    admissionListCopy=currentPatient.retrieveAdmissions(currentPatient);
    int lengthAdmission = admissionList.length;//finds the length of the admission
    //System.out.println("number of admissions "+ lengthAdmission);
    for(int i= 0;i<lengthAdmission;i++)
    {
        admissionList[i].retrieveDocuments(currentPatient,admissionList[i]);//declares the meth
    }
    findSoonestAppointment();//calls method which finds the soonest appointment
}
```





GUI Specific Features

These are a collection of features that while not specified as objectives on the design, but where instead an addition that where included on the development of the prototype to overall improve the system, while hopefully not hindering the actual development of other features. Because of this these features have no expectations to be met besides the fact that it looked correct and performed the correct action when the component was pressed.

Feature	**Viewing and removing notifications [-]**
Description	Once the system retrieves the information and logs in the user the patient will be brought to their home screen when this is loading the list of notifications are loaded. Here the patient can view then delete any notifications that they no longer wish to see. What happens is that the array of notifications removes the content and then moves every value up by one, also it decrements the length by one
Output & Storage Requirements	This varies as every notification differs in length but the array utilises the copyArray method theoretically allowing for a dynamic array. However, the data is limited to a single line in file with data being separated by the comma.
Data Used	In the evidence below this is the data used: PNUR0000001,B,Hello,Hello,Hello,Hello,Hello,Hello,Hello,a,
Evidence	   Here we can see an example of the notification system; the patient will view the notifications and then subsequentially delete all of them by pressing the X on the box. This will call the method below which removes the information below and will decrement the number of notifications after, once this has been done the system will visually update and remove the contents from file once the user logs out.

```
        {
    int iterations = numberOfNotifications - position-1;//the number of
        if(iterations == 0)//if no more swaps can occur this is ran(the
        {
            notifications[position] = null;//the contents are emptied
            numberOfNotifications--;//the number of notfications is dec
        }
        else
        {
            do//a do until loop is declared to iterate until the condit
            {
                notifications[position] = notifications[position +1];//
                iterations --;//the number of iterations required decri
                position++;// the position of the index incriments as r

            }while(iterations!=0);//termination condition looking until
            notifications[position] = null;//the last position of the a
            numberOfNotifications--;//the number of iterations left dec
        }
        updateNotifications();//the gui aspect of this task now occurs
```

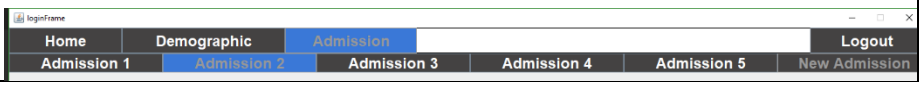
Feature	**Tab quick access bar(including admission tab bar) [-]**
Description	As I wanted to create my own tab bar it has allowed me to easily generate buttons which move the user between panels. The code is exactly the same between both tab bar except the admission one formats the size and location of the buttons relative to how many there are.
Output & Storage Requirements	The only output is to move the user onto a new panel, there are no storage requirements as there is no need to store any information in file
Data Used	none
Evidence	The first image is of the tab bar which is used when the user selects the admission. The current admission is set to the index of the button pressed then the button is set as selected to not allow it to be repressed, in some cases this is not always set then all the information

```
if(e.getSource()==topBarAdmission1Bttn)
{
    currentAdmission = admissionList[0];//assigns correct a
    setActiveAdmissionTopPanelBttn(topBarAdmission1Bttn);//
    listOfdocumentsGUI = currentAdmission.listOfDocuments;//
    numberOfDocuments =currentAdmission.numberOfDocuments;//
    singleDefinition = new Document[numberOfDocuments];
    outputDocumentCards(currentAdmission);//the cards for t
    createadmissionHomepagePanelGUI();//calls method which
    indexofadmission = 0;
}
```

```
public void setActiveTopPanelBttn(JButton button)
{
    homebttn.setEnabled(true);//the label is set to enabled to allow
    demographicbttn.setEnabled(true);//the label is set to enabled t
    admissionbttn.setEnabled(true);//the label is set to enabled to
    newPatientbttn.setEnabled(true);//the label is set to enabled to
    jargonLibrarybttn.setEnabled(true);//the label is set to enabled
    consultantPatientAdmission.setEnabled(true);//the label is set t
    consultantPatientDemographic.setEnabled(true);//the label is set
    homebttn.setBackground(darkButtonGrey);//the background colour o
```

```
public void setNonActiveTopPanel()//when the user l
{
    homebttn.setEnabled(true);//the label is set to
    demographicbttn.setEnabled(true);//the label is
    admissionbttn.setEnabled(true);//the label is s
    newPatientbttn.setEnabled(true);//the label is
    jargonLibrarybttn.setEnabled(true);//the label
    homebttn.setBackground(darkButtonGrey);//the ba
    admissionbttn.setBackground(darkButtonGrey);//tl
    demographicbttn.setBackground(darkButtonGrey);//
    newPatientbttn.setBackground(darkButtonGrey);//
    jargonLibrarybttn.setBackground(darkButtonGrey)
}
```

regarding the admission is retrieved.
In the second example we can see the code for active bar which sets all the buttons to the unselected colour and then using the parameter sets the selected colour to that button.
Also, in the third code screen shot it shows the exact same code as the second one but does not set an active button for instance in the document panels this is called as the user may wish to go back to their original admission.
Finally, the last image is a screen shot of both tab bars with the blue button being the one that has been selected.



4.4 Self-Evaluation

4.41 Functioning of the Prototype

Looking back on the prototype it has made me realise the difficulties with large software development in a limited time scale, while obviously this system is a prototype it should not be viewed as a final product, however that being said, it should reflect the core concept of the system by possessing functionality of the final system and that of which was planned to be included earlier on. **As I feel I left myself enough time to develop my EHR. In my opinion, I believe my system does satisfy the overall purpose and criteria of a prototype due to it achieving the main functionality I had intended when initially concepting the idea at the very beginning**. With regards to general functionality I believe that it has a wide-ranging set of features allowing for a fairly close representation to the final and desired system. While the code itself still does have a handful of bugs currently present, besides the fact the staff entity at the moment has no functionality, each other main user has the majority of features planned during the design stage and are fully working. When viewing the most important aspect, document viewing and management **I believe the process of creating and viewing documents was a great idea to do**, however in practise a lot more than I intended has come up to ensure that the feature works without issues arising. When it comes to the rest of the features of the prototype I think they too also show their usefulness towards the end result of the system, by doing it in a way that seems to benefit the main functionality of the system rather than overshadow it. For instance, having the ability to swap admissions between consultants while not really essential to view documents, eventually it will be a really helpful feature if the issue where to ever arise and merit the features use. While I did go into detail on the last section about specific functions, **here I want to say that as a collective the functionality holds up to a more or less complete system excluding any missing entities and minor bugs.**

4.42 Good Features

Overall I would refer to the prototype as a success at the same time not trying to sound conceited. Like I said before it does everything I wanted it to as it shows to current stakeholders the actual potential of my system, and also provides a wide range of features to demonstrate the eventual functionality of the system. Where I believe the system comes up stronger would be in the **visual display and overall navigation between screens and panels**. With that being said however, this does not mean my backend is a detriment to the system as naturally that is most susceptible to issues as it does not have to follow the standard conventions of declaring visual components and having the ability to use powerful prewritten functions such as set enabled. Anyway, I believe this to be the case as for my system as it will have to be used by patients, I felt also it was imperative that the system right from the start looked complete as it would help me also get an idea on what also was needed to be done during development. To go into further detail, I would like to credit navigation on part to the stack data structure while it was a last addition the ability to move through the panels has been a feature I have become proud of. Finally on the mater of GUI creation I believe while somewhat unconventional my panel generation system is another great aspect of the system due to it only ever initialising the panels that are desired and only needing to be generated once, while simple enough for panels have no updating data what I believe to be better about mine is due to the fact any fields or components that need updating will include the newest data once the panel is reloaded. Looking beyond visuals one section that I felt was strong was my **use of the object-oriented paradigm**. Because of this it has really helped with the backend due to the systems natural reliance on main entities. The record data structure has allowed myself to become more efficient when declaring methods trying to generalise where I can so it can be used in the parent class, however this isn't always the case as sometimes I had to overwrite a method later on due to the specific requirements of that process. Also, this aspect has also allowed myself to use inheritance really well, so that I only have to initialise attributes like name once at the top of the inheritance hierarchy, a challenge while doing this was to assign objects of the

child class to the parent class for instance when creating a document for a patient as they can have more than one to get around this the use of a list attribute worked really well allowing multiple objects to be stored despite the child class being the object we are holding information about. Other current good aspects I believe to be is **response time and processing speed**, while this is somewhat credited to the hardware the program was developed on, I believe that performance will not take a nosedive once the changeover between my personal laptop and the standard hardware provided by the collage occurs, This is due to the modular approach to designing processes for buttons and other areas of my system. Finally, the last few good aspects I would like to talk about are specific features I think work really well with each other, which is the ability **to amend information easily and the ability to record down the action at that precise moment of time**.

4.43 Shortcomings of the system

The system itself is not finalised and because of this there are limitations with its performance. While the system is bound to occasionally throw an exception or error, here I wanted to discuss the capabilities of the system and the areas that need to be focused on during the actual development of the software. One key area to address would be file organisation and management, although the writing and reading methods seem to be fully usable; to the extent of not changing them much in the final system it's the **actual location data is being written to that needs to be overhauled drastically**. This is clearly aimed towards the document and admission files that are created on a per admission and per patient basis respectively. This on reflection is a really inefficient approach to data handling and should have been realised earlier on when the idea was concepted during design. Other faults to the prototype would seem to be the occasional use of a linear search when a binary one was intended; however, this is an easy fix to implement. Another key issue I faced during the prototype was **generation of the primary key**, this was due to the generation of the numeric value rather than the surname as the system would occasionally fail to increment the number and as a result either large gaps where produced between values or that the value wouldn't even increment. In addition to this another shortcoming of the system also seems to be the current sensitivity towards errors and exceptions being thrown, what I mean by this is **its volatile nature** when something occurs that shouldn't causing a large chain of issues, while this could stem from a small mistake from the initial stages of development it could also be due to some logical errors causing actions that shouldn't occur take place. While some of these issues still need to be addressed it is for a later point in time.

Shortcomings with development - Improper use of time

Whilst I did give myself plenty of time to produce the prototype some portions of said time I believe to be misused of sorts. For example, around 16/08/19 when the first aspects of the backend where in development a few hours went into the researching, developing and then final disregarding of a random-access file system for data storage. During this time, I had hoped that I could utilise the java Random Access File writing methods to write my data to file, however due to my lack of inexperience and frustration in counting bytes **the idea was swiftly removed**. Another example was even earlier on around 07/08/19 when I looked into using the graphics class to draw lines so I could create lines for my system. In total roughly 4 hours went into a failed idea that never made it past testing in which that time could have been spent working on something else. However as this a prototype I **believe that these errors in judgment where not so negative as initially thought as they have allowed myself to explore multiple opportunities to tackle a task and have actually made me think of even better ideas than before**, for example the idea to use JLabels with the foreground set black and sizes set to that of a line have been a much easier approach to that of the one I initially thought of and also have the benefit of having the methods available to all swing components.

Actions for next time

The final thing I will address in the prototype is what I believe could have gone better for next time, while it is not a critique as such it is just a collection of actions I could make to help improve the development process. Initially during development, I spent a noteworthy time trying to introduce and create aspects that where partially irrelevant for instance the notification system. While it would have been introduced at some point this feature is not a vital necessity to the main core process of my system and could have been left out for some time at least to allow more time for more important features. This was due to the fact I felt that every screen needed to look finished before I moved on to another, this is why the patients home screen looks more complete than it actually is, as after this point I realised that I would fall behind in work if I continued this trend. **Another approach to development would have been my commenting**, due to the natural size of my system I left the commenting of the source code to a later point and did it all at once, while I could still understand all aspects it did take time to comment all classes and methods. Finally, like previously addressed the final change I would have made is **scrap the initially planned file storing and replaced it there and then** however I will now have to do it in my post prototype and plan it carefully how it will be done.

4.44 Suggestions for Improvement

Like any piece of software there is no end to development of the system as there is always room for improvement. With that being said I have collected a list of aspects I would like to see change, along with any potential inclusions and bug fixes. These were ideas that were not included on the prototype due to either being too ambitious and would be detrimental to the rest of the development of the prototype or at the current moment would be pointless to include. As these ideas are an accumulation of both suggestions for improvement and fixes I have colour coded them respectively

- Include search for bookings
- Multiple paged documents
- For booking GUI add title that informs the user where they are
- Scalable windows
- for symptom check boxes: remove arm, add shoulder, add hip
- centre text in jtextpanes (in symptoms area)
- Allow both the admin staff and patient to create new accounts:
    - Admin- to continue over old ones from previous system
    - Patient- new users
- Address variable fields isolate and separate the number of the building and the street make name an optional choice but either one is required
- User homepage realign welcome and home labels to fit better
- When initialising the notification array size will equal the number of notifications they currently have
- In contact bar admission id will be displayed for more clarity
- Include a date picker
- Text fields have bold font for all text
- Contact information/postcode have individual text fields for every character
- Urgency/hierarchy for admissions more important
- Have a horizontal jlist when more than 5 admission with a scrolpane holding it
- Format text fields to automatically fit in their location
- Remove in admission contact bar medication and replace with symptoms
- For contact bar, Add current diagnosis
- Add notice when patient has no documents that there are no documents
- Add county to patient attributes
- In User turn findIndexOfPatient from linear to a binary search
- Add closet appointments for patient cards in consultant home, use the code for the contact bar one if need be using the inner loop.
- Possibility to amend documents
- Possibility to create feature that allows management to add new types of documents onto the system
- Fix bug that causes first patient for consultant to replace every other patient, when a newer patient is added to the list(**NEEDS FIXING**) seems to occur when adding the last admission over to the other patient, I reckon could resonate from an issue a series of two for loops occur with the second one not functioning properly.
- Fix visual issue when consultant end is used to access an admission then a patient with no admissions is access from the patient end has the prior admission contact card
- Upgrade pop up window for jargon library from message to ability to request new definitions ~~ similar to notification system for consultant

- Improve search for jargon to iterate with a binary search for all the characters entered but then go linearly from the start of the section afterwards adding them into a list e.g. endo entered but endo endothermic endorsement performs a linear search until finds part that **contains** the search criteria then copies over all of the rest until no longer contains aspect
- For action log add labels to every index so that e.g. old data would have "From" before being printed concatenated to it
- When editing demographic set and receive text for allergies pane correctly currently writing to new line when there are more than one line in the patient demographics
- Reduce documents needed by having the following structure for every patient:

AAAA0000001

DAAA0000001 …

DAAA0000002 …

AAAA0000002

DAAA0000001 …

- Add ability to force an update to demographic information=> add attribute to every demographic for the patients dateOFLAstUpdate if number > 90 days force the demographic screen to be loaded and check information is up to date
- If keeping idea of having a prescription on the admission home screen: create a new list of prescriptions using a stack and always display the most recent prescription
- Force the user to have at least 1 or more symptoms before creating an admission
- Include search that checks if user has the same information as another user when account being created.
- Patient homepage link together the two admission parts to the corresponding admission.

Comparing current effectiveness and evaluation of the prototype

While not entirely necessary I wanted to evaluate the current objectives that I had designed back in the investigation against the current versions of the system and compare how well they currently work against how I intended them to in the initial conception. In addition to this I have also discussed about current effectiveness about each objective. While all the objectives I set out to do where included some criterion have been left out due to either a change of plan during development or the fact that it would impractical to include on this version. To help visualise what actually was included on the prototype I have colour coded each criterion one of three colours. Green to signify that it has been met, Yellow to indicate it is included but may not be fully finished, and finally red to show that it is not included for whatever reason

Objective	Success Criteria	Has the Criteria been met? / Current effectiveness
1 **Input symptoms into expert system**	To count as a success when the user has entered all their symptoms, they will then submit the information onto the system no errors should occur. This should be for the patient and new users who will become patients. More than one symptom should be able to be selected.	This first criteria I believe to be met, while nothing is done with the information as of yet, the information the user is extracted and outputted showing what has been selected. Again, this criterion has been met because as the ability to create new admissions is available to both types of users, we can say the ability to select symptoms is also present for both and prove that the criteria has been met. While forcing the user to pick more than one symptom has not been included, the feature will now be intended to be included.
3 **Generating new patients on system**	Detect if the new account is pre-existing, this would be that the user is trying to create an account with similar credentials as a current patient. Generate a new patient account onto the system containing all the initially required information.	This criterion has not been met due to the basis that multiple people may share information that is similar or could be exact and therefore would be counterintuitive as the whole reason the primary key generator has been created is to allow this feature to be included rather than using both the user name and surname followed by an integer. However, despite the last criteria not being met, this next one can definitely be said to have been met as in order for any information to be generated and tested an account is needed and the only way to create one is through the account generation system.

4 **Generating a new admission on the system**	Adding the admission to the account should generate a corresponding consultant. The consultant assigned to the patient should see that a new admission has been created. The admission should be added to the patients account containing all the relevant information.	I believe I have achieved this criterion well due to the fact once a corresponding admission has been generated the correct consultant also receives the new admission respectively. The second criteria hasn't been met due to the fact that a notification system is not yet in place for this user. While through inspection of the patient's admission the consultant may realise that a new admission has been created, they will not receive any new alert that said process has occurred. Finally, I can confidently say that this aim has been met. In order to view any admission in the first place the information first needs to be read from file and the only way to set the data is to write it to file, which proves that the admission must have been added to the account.
6 **Login users**	An error message should be generated if not correct. if the password is wrong that should be notified id the username is wrong then that should be notified. If the credentials have been entered correctly the corresponding user should be allowed access onto the respective part of the system.	I believe the main idea of the criteria has been met, however determining individual credentials, I believe has not been satisfied. This being down to the fact the login system was only partially implemented and that the full version was not yet fully developed. This last criterion has been met. This is because that in order to gain access to the systems features the user eventually needs to log in, once the user does login they are returned to the respective part of their system, that being the homepage.
7 **Display menu options**	The buttons should be noticeable and clear to what they do.	Personally, as this is inevitably down to preference, I think that I have met the criteria as all buttons on the system are concise and are noticeable, while the location and formatting are subjected to change I think this has been met.

11 **Search for employees**	When finished the system should return the correct information needed for management. If an account or entry can't be found a notice should appear that no items that match the query were found.	This criterion has been met as I believe that it is a vital requirement of a search, without this criterion the prototype would not benefit from the objective in any way and would have been ultimately omitted from the system. However, while the prior condition was a necessity, this criterion is the opposite. Because of this it has not been met and will be included at a later point in time.
12 **View an employee's transaction log**	As it could be very large the user should be able to section off parts of the log and view the specified area. After entering the times, a list should appear containing all the actions performed by the employee along with information relating to it.	I believe that I have met this goal, due to the fact that it is not bounded by the date of the actions, besides the initial action. When the user enters the bounds between two dates they can enter any date from the instance of the first action all the way to the current moment of time. The best aspect being that any date can be entered not just dates that an action has been performed on. To merit the use of the function, I have also included an output method which returns all the actions the employee has done along with any necessary information. Because it works I can optimistically say the objective has been a success.
13A **A Read transaction log from file**	The date, time and patient should be included with a description of what was done for each specific action should all be copied over from file to view. This should also be done where the running of the system is not disrupted or slowed.	This criterion I believe is a success due to the fact that in order to retrieve the patient the information has to be read from file. I believe that this criteria has been satisfied despite the fact that it has to utilise a standard function of the java package, this is due to the fact that it needs to only read the file rarely and saves a copy of it to an array, reducing the need to call it.

Objective	Success Criteria	Has the Criteria been met? / Current effectiveness
13B **B write transaction log to file**	This should be done for every action the employee performs. The date, time and patient should be included with a description of what was done.	I am certain that the criteria has been met this is because that whenever an employee performs an action the method which also performs the action updates an attribute indicating what action has been performed, when the method which writes the action to file, it uses this attribute to correctly determine the action. I am pleased to say that the current effectiveness of this objective is good and because of this I can say the criteria has been met.
21 **Have patients view their Admissions and Demographic information**	The patient should see all the information held about them. No information should be missing from the page. All the information in the patients file should be brought up.	The current effectiveness of this feature is perfect, the feature does exactly what I want it to achieve, it allows the patient to read all the fields the system holds about them. Because of this I can say the criteria has been met as whatever information the patient enters either when creating or updating the file is immediately available to be viewed once entered. However, one of the success criteria says that no information should be missing, this is misleading ad needs to be clarified it is referring to fields the patient has already entered non required fields do not need to fill with information.

Objective	Success Criteria	Has the Criteria been met? / Current effectiveness
22 **Amend demographic information**	This should once finished update the entire system about what has happened. If not occurred for three months, it should be enforced on the user to make sure it is up to date. Only basic fields must be kept and not removed i.e. name The patient should be able to enter new information onto their file where necessary and available.	I believe that the current effectiveness of this feature is in a state where this works but could use some attention in the final version. This will be for any areas that I may have missed out when updating the patient, the current method is to redeclare a patient if any changes have been made so any attribute that needs to be updates can be. The second requirement has not yet been introduced this is because it is unnecessary and while it will be possible to alter dates to achieve the 3-month time period it would have just been a feature that would hardly get used in this version. Some aspects of this last requirement have been met, for instance while the first name and surname can be changed the username is still unchanged. However, as no processing is done to the data as of yet there was no benefit at that moment of time fully include it.
26A **Sort documents**	When the selected sort has been chosen the system should place the documents in the admission in that order	I believe that this works well, with current effectiveness being suitable for the current situation. For the number of documents that exist a binary search is more than a preferable choice for the system. Because of this I can also say that the criteria for the objective have been met.
26B **search documents**	When finished the system should return the correct information needed for staff If an account or entry can't be found a notice should appear that such item was not found	This objective I can say was mostly complete. For the first criteria I can say that it has been met due to it correctly retuning the respective data relating to the search criteria For the second one I can also say that it has been met due to the fact that if an invalid document primary key has been entered a pop up appears informing them nothing was found. With overall effectiveness with the objective I am pleased with it using a binary search.

28 **Consultant can Search for patients**	It should correctly return all the correct patients from the search query If no one is returned a message should appear informing them of so	Similar to objective 26B I am pleased with it using a binary search, with current effectiveness it does not need to change and is unlikely to be altered later on. For the first criteria I would say that it is partially met due to the fact that the return value from the search is the location of the item in the array. In the final version it will use a hybrid search (similar to the search used in finding the correct primary key) where once the initial value is found the rest are also outputted until the end of the items. For the last criteria I can say that this has been as it works well. If no items can be found a return value of -1 is passed through, when trying to find the index it will see that this is out of bounds and will instead output the indication.
29 **Consultant can view patient files**	It should not be cluttered, or hard to find relevant information. Everything should be easily identifiable. All their patients' content should be able to be accessed from this file.	I feel that this objective is currently effective on the system and achieves the desired processes and outcome. I believe all fields and buttons are clearly labelled with either the process or data it holds. While the term file is more representational of the two tabs and the collection of data as a whole, all information concerning the patient is always viewable to the consultant from this section of the system.
30 **View patient Demographic information**	All the information should appear as intended nothing should be missing or obscured from sight. No missing components of the GUI should be present.	While this set of criteria is slightly subjective I believe that these criteria have been met due to the fact that navigation seems to be effortless in my case and that no components are obstructed from sight, with the second criteria I believe that during development I would encounter many issues with displaying the components, however as that has hardly occurred I believe that this criteria have been met successfully due to there being satisfactory effectiveness with this objective.

31 **Sort admission**	When they are being sorted, they should be in the correct order. It should return the admissions in the correct order as desired.	Again, this objective was written when I had only a partial idea of how the system would be created because of this, I can say that the sort method is flawless for this set of data due to it correctly displaying the list in the intended order. As the sort method is modular. With regards to the second statement again this is really a result of the method working as intended and as I have achieved that I can say that this objective has been met.
32 **Edit Prescriptions**	when selected to be updated, when loaded it should fill in all currently filled fields with the old information to allow for slight adjustments like dosage without removing all the data.	Due to the change of how the prescription system changed during development unfortunately the circumstances are that this is yet and possibly unlikely to be fulfilled in the prototype and even the final system. However, this ability to create prescriptions can be referred over to Objective 35 with the ability to create new prescriptions being fully functional.
33 **Add Admission information**	The information added should be correct and no errors should be included The consultant should be able to add a wide range of information to the admission like documents to general information The information added should be saved to the action log	While this first criteria will not be able to be achieved due to the lack of validation this will be a key feature that will be included at a later date and will be a primary focus overall during development especially during the early testing stages of the software. While currently the number of fields is limited to 7 fields these are editable, with others such as document count or current prescription also having the ability to be introduced at a later point in time. Finally to show current effectiveness I can say that these last two have been met, when the user enters data it is recorded to the action log and that any important change e.g. current consultant is changed it is immediately updated and the effects can be seem, continuing on with the consultant example the admission is moved to the correct consultant

34 **Edit Admission information**	The patient should be notified in some way if a change is made to the admission The information should be saved to the action log	The backend of the system is yet to be finalised however all file management aspects and GUI features are all in place, it is simply the process of calling a write method to add the action to the correct location of the patient to their notification file. While the first one is yet to be finished this last criterion has been met and I can happily say is fully effective with its process.
35 **Add notes**	The input should be validated The new set should correctly be added to the correct location The patient should be able to view the new document once added	Like before, validation is an aspect that is not yet included on the system and so it is unfair to suggest that this objective should be said to be incomplete as the other criteria has been met due to the fact the consultant can easily create any of the four documents available to them. With it being fairly easy to create a document to any admission the consultant has access to.
36 **Encrypting data before being written to file**	When this occurs, it should correctly encrypt the data that has passed through it. When it enters the data should be correctly changed to the right data type and then encrypted and should then pass on the data to wherever it needs to be sent.	While it's not being used currently for any data the function is available to the management user as a proof of functionality and show that one is present in the prototype as I currently need to see all the data in file to check no errors are occurring. However, as it has been declared in the user class it is available through the entirety of the system and because of these features I can say that the criteria have been met and that the feature is effective.
37 **Decrypting that has been read from file**	When read from file it should decrypted using the same cypher into the correct data type	Like objective 36 I can optimistically say that this feature has been achieved and can say it to be effective for its purpose this due to it being a proof of concept for the current moment of time. However, as I might be wanting to change the cypher at a later point of time I can do this as no data is scrambled at the time of writing.

Objective	Success Criteria	Has the Criteria been met? / Current effectiveness
38 **Using the Jargon library**	When the user accesses the library, they should be able to enter a specific word onto the system and see for the definition If no definition is present it should be automatically put forward for it to be added	This feature has been developed completely and is fully function because of this it is currently effective on the system and achieves everything I intended it to do. The next step for this is to allow the consultant to add and edit definitions on the system without having to go onto the file for this.
41 **Search through demographic information**	When ran it should go through the file and return the patient that correctly matches the search query as only one is intended no others should be returned.	Again as this feature is important to the running of the system it was clear from the outset that the feature was going to be finished and complete for the prototype while the search can always be improved for the current effectiveness I am pleased with the feature and because of this I can say that the objective has been met.