Contents

[Overview of the system 2](#_Toc159929230)

[User interface design 3](#_Toc159929231)

[Login 3](#_Toc159929232)

[Menu 3](#_Toc159929233)

[Patients menu 4](#_Toc159929234)

[View specific patient 5](#_Toc159929235)

[Add patient 5](#_Toc159929236)

[View discharged patients 6](#_Toc159929237)

[Create and manage users 7](#_Toc159929238)

[Module design 7](#_Toc159929239)

[Login 7](#_Toc159929240)

[Create account 7](#_Toc159929241)

[Logout 8](#_Toc159929242)

[View patients 8](#_Toc159929243)

[Add patients 8](#_Toc159929244)

[Remove patients 8](#_Toc159929245)

[Discharge patient 8](#_Toc159929246)

[Manage users 9](#_Toc159929247)

[Saving patients to file 9](#_Toc159929248)

[View discharged patients 9](#_Toc159929249)

[Input, output and processing requirements 9](#_Toc159929250)

[Login 9](#_Toc159929251)

[Output 10](#_Toc159929252)

[Create account 10](#_Toc159929253)

[View patients 10](#_Toc159929254)

[Add patients 11](#_Toc159929255)

[Manage users 11](#_Toc159929256)

[Saving patients to file 12](#_Toc159929257)

[View discharged patients 12](#_Toc159929258)

[Data Structures and files 12](#_Toc159929259)

[Files 12](#_Toc159929260)

[Variables/data structures 13](#_Toc159929261)

[Algorithms 14](#_Toc159929262)

[Login 14](#_Toc159929263)

[Create account 14](#_Toc159929264)

[Logout 15](#_Toc159929265)

[View patients 15](#_Toc159929266)

[Add patients 15](#_Toc159929267)

[Remove patients 15](#_Toc159929268)

[View Discharged patients 15](#_Toc159929269)

[Saving patients 16](#_Toc159929270)

# Overview of the system

#### Processes

* Add user – taking input from an already signed in user to create an account
* Export as file – starting a save dialog for the user to save the data
* Contact GP – displays the email of the patients GP
* Add patient – takes data input from user to add a patient record
* Logout – closes all windows and takes user back to login screen
* Discharge / remove patients – two ways of taking patients out of the main patients data grid and CSV
* View patient detail – opens a form on a specific patient with spaces to edit their details

Design choices in the overall layout of the system were made with the aim to create a layout that would make sense to users without extensive technical experience. All modules and features of the program are organized in a way in which the user is likely to work out where they are with minimal assistance or difficulty. This layout of the program also allows for all modules of the system to be separated by an intuitive menu all fitting into the design and not affecting one another.

# User interface design

### Login

A screenshot of a login form

Description automatically generated

This is the textbox for the username

This is the textbox for the password – the text is represented as dots

This button starts the login subroutine taking the values from the text boxes

The design for the login page consists of a single large button with clearly labelled and readily apparent functions. There is no more information or things on the screen than required. All these things are put together to make the form as user friendly as possible, particularly for the typical users of this program – most of which are lacking technical knowledge and who don’t have time to be trained on a completely new system.

### Menu

A screenshot of a computer

Description automatically generated

Logout button separate from others so not accidentally clicked

A group of buttons corresponding to each major form of the program

Logs user out and returns to the login page

Similarly to the login page, the menu does not provide any more information than is necessary and has a few large, clear buttons with clear functions. These aspects all work together to make a form that is tailored towards the users of this program who will likely lack technical experience and training.

### Patients menu

Overwrites or creates a CSV file in the program files that will automatically be opened on the loading of this form. A different colour so that the users are reminded to use it

A screenshot of a computer

Description automatically generated

A datagrid structure displaying all the patients in the database

Creates a CSV file and opens a save dialog for the file so the user can chose where to save it

Delete patient button away from others and in red as to not be accidentally clicked

The view patients menu has a large data grid table where the patients and their data will be stored. Each column of the data grid has clear headings with units where necessary to unambiguously display all stored patient data to the user. There are also buttons in an easy to access position with clear and understandable labels. This means that the users with limited technical experience are easily able to get a grasp of the system and use it effectively.

### View specific patient

Possible space for picture of patient – may just be left as generic image

All stored information about the patient with a space to change it

A screenshot of a computer

Description automatically generated

Date of birth in a calendar selector so the format is consistent

Makes changes to the data-grid

The menu for specific patients displays all the data about the patient and the option to edit them and nothing else. It is simple and intuitive enough that the likely users of the system will be able to work it out with little instruction or difficulty.

### Add patient

A screenshot of a computer

Description automatically generated

Units given for some values for clarity

Adds patient to data grid

Spaces to input patient data

Date time picker for date of birth (age is calculated from this )

The Add patient menu has space to input all the details about a patient and add them to the data grid. The procedure to do this is easy to workout and all buttons and input fields are clearly labelled and positioned. The date of birth section uses a date time picker so that the users are forced to input data in the correct format, the ethnicity uses a combo box to prevent inconsistencies in the data and the sex input uses radial buttons for the same reason. Overall this form is design to ensure ease of use for the user and useable, effective inputs.

Large simple button with apparent functions

### View discharged patients

A screenshot of a computer

Description automatically generated

Headings for each field with units where required

A datagrid structure displaying all the patients in the database

The view discharged patients form is very similar to the patients form in purpose and functionality. It has 3 simple buttons and a data-grid containing no more information than is required at this point in an effort to make the form as user friendly and easy to learn as possible.

Text boxes for all data that needs to be entered for a new user

Simple headings

### Create and manage users

A screenshot of a computer

Description automatically generated

A datagrid structure displaying all the Users in the database

The Create and manger users is slightly more complicated than the other forms however all information displayed on it is still relevant and necessary and it will be accessed mainly by system admins so the speed to learn it is slightly less critical.

# Module design

### Login

This module is a procedure that is run once the user has entered their details and pressed login. The Input will be subject to a length and presence check and if the inputs pass this, the module will continue, if not, an error will be thrown. Once the system has taken the user inputs (username and password) values will be checked against the CSV file using a binary search algorithm (password inputs will need to be check against stored passwords that are encrypted).

This module will be key part of the login and user authentication system (A major objective of the system) that keeps data on the system secure.

This function will be called on the login menu (the first window shown to the user) once the user has entered their details, this module will begin.

One of the objectives of the system is to have a login system so that each user can be identified, and levels of access can be granted. For this to happen, users must be able to login to their account so function to do this is required.

### Create account

This module will open a new window where the user adds their relevant information and details. Once the user has entered this, the Input will be subject to a length and presence check and if the inputs pass this, the module will continue, if not, an error will be thrown. After this the system will take this data and store it in the CSV file containing the relevant data and the user will then be able to login to their account with the module above using this data.

*\*Note: only already signed in users will be able to create accounts for other people (the system starts with an admin account)*

One of the objectives of the system is to have a login system so that each user can be identified, and levels of access can be granted. For this to happen, users must be able to create accounts so function to do this is required.

The subroutine will be accessible by signed in users on the main menu.

### Logout

As multiple users are going to be using the system, the user will have to have the ability to logout so that others can use the system. This module will be a function triggered by a button on the main menu displayed once the user has logged in.

One of the objectives of the system is to have a login system so that each user can be identified, and levels of access can be granted. For this to happen, users must be able to logout so that others can login.

### View patients

This module consists of the code run/ subroutine that occurs when the user clicks the *View Patients* button on the main menu. It brings up a window featuring a data grid with all relevant patient data and the options to: *View specific patient, add patient, delete patient, go back, discharge patient or contact patient GP.*

The primary system objective is to store and display patient data, this module loading the *View Patients* screen is the way in which the system will be displaying the patient data obtained.

### Add patients

This module is a subroutine called when the user clicks the *Add patient* button on the *View Patients screen*. It displays a popup menu that lets the user enter the details for a new patients (Name, Sex, height, weight, date of birth(used for age), medical history, GP name, GP email, ethnicity and a section for other important information). This data is then written to the patients data grid.

The primary system objective is to store and display patient data, this module loading taking the data input, is the way in which the system will be obtaining patient data.

### Remove patients

This module is a subroutine called when the user clicks the *Remove patient* button on the *View Patients screen*. It displays a popup menu that lets the user select patient(s) to remove from the system.

\*Note, this feature is different to the *Discharge Patient* module as with this one, the patient is just outright removed whereas with *Discharge Patient,* the protocol involving a discharge summary and email to gp is followed

Part of the objective of storing patient data involves removing it where required and when removing patient data it does not always warrant a full email/discharge summary so a *Remove Patient* subroutine is required.

### Discharge patient

The discharge patient module is a feature of the program in which, similarly to *Remove Patients,* a patient is removed from the main patients table and subsequently the patients CSV file. It differs to remove patients as it takes the removed record and adds it to a removed patient CSV where it can be viewed later by the user.

Part of the objective of storing patient data involves removing it where required and most of the time when a patient leaves the hospital it is beneficial for their details to be kept for later use and reference creating the need for a file and table for the patients not currently in treatment.

### Manage users

This module consists of the code run/ subroutine that occurs when the user clicks the *Manage or add users* button on the main menu. It brings up a window featuring a data grid with all relevant user data(taken from Users CSV) and the options to add users. There will be a button to add the user which will start a subroutine that reads data from all the text boxes and adds the user to the data grid and Users CSV file.

One of the objectives of the system is to have a login system so that each user can be identified, and levels of access can be granted. For this to happen, users need accounts to use but shouldn’t be allowed to make them their selves so a mange and add users menu for the already logged in users is required.

### Saving patients to file

This module is a subroutine called by a button on the view patients form. It takes the records from the data grid view on the patients form and loads them into the patients CSV. This subroutine may also be used to save other data grids to their relevant CSVs.

This will be able to happen either through the program and be saved directly into the program files or it can happen through a save dialog meaning the user can save the file where they wish. The use will get a choice.

The primary system objective is to store and display patient data, this module saving patient data is vital given that without it all data would be lost when the program is closed.

If the file already exists it will be deleted. At this point all the data in the file will have been entered into the data-grid that is about to be saved so the old file can be deleted to make the saving more efficient. This also means that if there is no file there a new file is always created without issue.

### View discharged patients

When the discharged patient module adds the discharged patient’s data to the discharged patients CSV the user needs to be able to see these records and mange them. This module consists of the code run/ subroutine that occurs when the user clicks the *View discharged patients* button on the main menu. It loads the data from the discharged patients CSV and displays it to the user giving them the option to view specific patients and contact GPs.

Part of the objective of storing patient data involves removing it where required and most of the time when a patient leaves the hospital it is beneficial for their details to be kept for later use and reference creating the need for a file and table for the patients not currently in treatment.

# Input, output and processing requirements

## Login

Input

* Username
  + as string.
  + It will be subject to a presence check
* Password
  + As a string
  + Subject to a presence check

Process

* Username searched for in the Users CSV file using a binary search algorithm
  + If the user is not found the processing stops
  + If the user is found their password is then read
* The user’s password is compared to the password found associated with the found account

### Output

* If the entered username was not found show a message box saying their username wasn’t recognised
* If the entered password doesn’t match the password of their account show a message box saying their password was wrong
* If either field is blank say they need to fill in all fields before logging in

## Create account

Input

* Full name
  + String
  + It will be subject to a presence check
  + The system will check there is a space somewhere in the name to ensure both the last and first name where entered
* Username
  + string
  + It will be subject to a presence check
* Email address
  + string
  + It will be subject to a presence check
  + The system will check it contains both a ‘.’ and an @ symbol
* Password
  + string
  + It will be subject to a presence check
* Confirm password
  + It will be subject to a presence check

Process

* The program will check the password and confirm password match

Output

* If the password and confirm password do not match the system will show a message box saying the passwords don’t match
* Otherwise, the data will be added to a new line of the Users CSV file

## View patients

Input

* The system reads the patient data from the Patients CSV file
  + (Name, Sex, height, weight, date of birth(used for age), medical history, GP name, GP email, ethnicity and a section for other important information) for each record in the CSV)

Process

* The data is then added to a data-grid view

Output

* The Data-grid view is then shown to the user with all data inside it

## Add patients

Input

* Name
  + string
  + It will be subject to a presence check
  + The system will check there is a space somewhere in the name to ensure both the last and first name where entered
* Sex
  + string
  + Sex is input with two radial buttons so no validation is required
* Height
  + double
  + The system checks that the input is a numerical greater than 0
* Weight
  + double
  + The system checks that the input is a numerical greater than 0
* date of birth
  + date time
  + The form uses a date time picker meaning that all inputs are of the correct format.
* medical history
  + string
  + No validation to be done on this since it can be left blank and could take any value
* GP name
  + string
  + It will be subject to a presence check
  + The system will check there is a space somewhere in the name to ensure both the last and first name where entered
* GP email
  + string
  + It will be subject to a presence check
  + The system will check it contains both a ‘.’ and an @ symbol
* Ethnicity
  + string
  + It will be subject to a presence check
* other important information
  + string
  + No validation to be done on this since it can be left blank and could take any value

Process

* The age will be calculated using the date of birth
  + If this value is below 0 or above 150 the user will be asked to input it again

Output

* The full set of data will be added to the patients data grid as a record

## Manage users

Input

* The system reads the patient data from the Users CSV file
  + Name, username, email, password

Process

* The data is then added to a data-grid view

Output

* The Data-grid view is then shown to the user with all data inside The Data-grid view is then shown to the user with all data inside it

## Saving patients to file

Input

* The system reads all records from the patients data grid view on the patients form

Process

* If a file already exists for the patients CSV it is deleted

Output

* All data from the patients CSV is written to a new CSV

## View discharged patients

Input

* The system reads the patient data from the discharged patients CSV file
  + (Name, Sex, height, weight, date of birth, age, medical history, GP name, GP email, ethnicity and a section for other important information) for each record in the CSV)

Process

* The data is then added to a data-grid view

Output

* The Data-grid view is then shown to the user with all data inside The Data-grid view is then shown to the user with all data inside it

# Data Structures and files

### Files

I have chosen to use CSV files for this system. This is because I am planning to make extensive use of data-grid and the column, row format of data-grids translates well to a CSV format. CSVs are widely used across the medical field and the program is designed to have files which can be taken and used elsewhere. Because of this a CSV file would be the most appropriate.

|  |  |  |
| --- | --- | --- |
| File name | File type | File use |
| Users table.CSV | CSV | Stores information and login for each user (Doctors etc) – Name, email, username, password |
| Patients table.CSV | CSV | Stores all patient data - Name, Sex, height, weight, date of birth(used for age), medical history, GP name, GP email, ethnicity and a section for other important information) |
| Discharged Patients table.CSV | CSV | Stores all data on the patients that have been discharged - Name, Sex, height, weight, date of birth(used for age), medical history, GP name, GP email, ethnicity and a section for other important information) |

All files are independent and there is no relationship between them as there is no redundancy to warrant a relational file system and all files work as intended independently of one another.

### Variables/data structures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Length | Scope | Description | Relevant to |
| Patients () | String | 2D Array for up to 500 Patients | View Patients module | [An array consisting of patient’s last name and first name separated by commas][unique 3-character identifier] | Patient |
| Patient first name | String | 25 (2D Array for up 5000 Patients) | View Patients module | [Patient’s first name][unique 3-character identifier] | Patient |
| Patient second name | String | 25 (2D Array for up 5000 Patients) | View Patients module | [Patient’s second name][unique 3-character identifier] | Patient |
| DOB | Date/ integer | 8 (2D Array for up 500 Patients) | View Patients module | [Patient’s date of birth][unique 3-character identifier] | Patient |
| Weight | Float | 5 (2D Array for up 500 Patients) | View Patients module | [Patient’s weight][unique 3-character identifier] | Patient |
| GP name | String | 50 (2D Array for up 500 Patients) | View Patients module | [First and last name of GP][unique 3-character identifier] | Patient |
| GP Email | String | 50 (2D Array for up 500 Patients) | View Patients module | [Email of Patient’s GP][unique 3-character identifier] | Patient |
| Scheduled Clinic dates | Date/Integer | 12 per item in queue of 100 | View Patients module | A queue containing all scheduled operations | User |
| Scheduled Clinic dates | Date/integer | 12 per item in queue of 100 | View patients module | A queue containing all scheduled clinic appointments | User |
| Ethnicity | String | 35 (2D Array for up 500 Patients) | View Patients module | [Ethnicity of patient ][unique 3-character identifier] | Patient |
| Relevant medical history | String | 1000 (2D Array for up 500 Patients) | View Patients module | [A qualitive account of the patient’s medical history][unique 3-character identifier] | Patient |
| Height | Float | 5 (2D Array for up 500 Patients) | View Patients module | [Patient’s height ][unique 3-character identifier] | Patient |
| Extra notes | String | 700 (2D Array for up 500 Patients) | View Patients module | [Any additional notes and comments for patient ][unique 3-character identifier] | Patient |
| Name | String | 25 (2D Array for up 50 users) | Login/ Create account module | Name of user | User |
| Second name | String | 25 (2D Array for up 50 users) | Login/ Create account module | Last name of user [unique 2-character identifier] | User |
| Role | String | 30 (2D Array for up 50 users) | Login/ Create account module | Role of user in the organisation [unique 2-character identifier] | User |
| Password | String | 20 (2D Array for up 50 users) | Login/ Create account module | User’s login password [unique 2-character identifier] | User |
| Access level | String | 4 (2D Array for up 50 users) | Login/ Create account module | User’s access level [unique 2-character identifier] | User |
| Email Address | String | 50 (2D Array for up 50 users) | Login/ Create account module | User’s email address [unique 2-character identifier] | User |

# Algorithms

### Login

Declare Username, Password

Read Username

Read Password

Read [CSV containing user accounts]

For I in range (length([CSV containing user accounts])

If Username = AccountUsername And Password = AccountPassword then

LoginUser(Username)

Loggedin = True

Endif

Endfor

If NOT(loggedin) then

Output “Wrong username or password”

Endif

### Create account

Declare Username, Password, CheckPassword

Read Username

Read Password

Read CPassword

If Password = CPassword then

Write Username, Password

Else

Output “Passwords do not match”

Endif

### Logout

Close Current form

Open login menu

Current account = Null

### View patients

If PatientsFile.CSV Exists then

For I in range (length(Patients[]))

Read patient Name, Age, Height, Weight, DOB, Relevant medical history, Current GP information, Ethnicity to Datagrid

Endfor

Show Patients.vb

### Add patients

Declare patient Name, Age, Height, Weight, DOB, Relevant medical history, Current GP information, Ethnicity

Read patient Name, Age, Height, Weight, DOB, Relevant medical history, Current GP information, Ethnicity

Add patient Name, Age, Height, Weight, DOB, Relevant medical history, Current GP information, Ethnicity to datagrid

### Remove patients

Declare SelectedPatient

Read Selectedpatient

Remove(selected patient)

### View Discharged patients

If DischargedPatientsFile.CSV Exists then

For I in range (length(Patients[]))

Read patient Name, Age, Height, Weight, DOB, Relevant medical history, Current GP information, Ethnicity to Datagrid

Endfor

Show Patients.vb

### Saving patients

If PatientsFile.CSV Exists then

Delete PatientsFile.CSV

For I in range (length(PatientsDataGrid[]))

Write patient Name, Age, Height, Weight, DOB, Relevant medical history,Current GP information, Ethnicity to Datagrid

Endfor