|  |
| --- |
|  |
| IT PAT 2021- Phase 1  Non-profit organization inventory application |
| |  |  |  | | --- | --- | --- | | Rohan Dayaram | 4/24/21 | Information Technology | |

Contents

[INTRODUCTION 2](#_Toc71025152)

[Background 2](#_Toc71025153)

[The Task 2](#_Toc71025154)

[Main Problem 2](#_Toc71025155)

[My Solution 2](#_Toc71025156)

[Additional problems and solutions 3](#_Toc71025157)

[Problem - The input of data 3](#_Toc71025158)

[Solution 3](#_Toc71025159)

[Problem - Sorting and arranging data 3](#_Toc71025160)

[Solution 3](#_Toc71025161)

[Problem – Access control 3](#_Toc71025162)

[Solution 3](#_Toc71025163)

[Problem – Fading of ink and loss of files 3](#_Toc71025164)

[Solution 3](#_Toc71025165)

[WHAT WILL THE PROGRAM DO? 4](#_Toc71025166)

[HOW WILL THE SYSTEM BE USED? 4](#_Toc71025167)

[WHO WILL USE THE PROGRAM? 4](#_Toc71025168)

[FLOW OF APPLICATION 5](#_Toc71025169)

[DATA STRUCTURES 5](#_Toc71025170)

[DATABASE DESIGN 15](#_Toc71025171)

[ENTITY RELATIONSHIP DIAGRAM 15](#_Toc71025172)

[GUI DESIGN (SCREENSHOTS) 17](#_Toc71025173)

[COMPONENTS USED 22](#_Toc71025174)

# INTRODUCTION

## Background

Inventory management is a vital part of any non-profit organisation. An example of such an organisation is Child Welfare. Details of donations, employees, expenses etc need to be updated constantly. The current system being used is not efficient enough for our world which is rapidly becoming more and more technologically inclined, I believe that my program will optimize the storage and retrieval process of information as it will make use of a database to hold records.

## The Task

My task is to develop an information system to capture accurate records of all the necessary details of donations, employees, income, expenses and other important records electronically. The main objective of my project is to design and develop a user-friendly and visually modern system. This system is proposed to be a database management system which will automate most of the tedious daily tasks for the admin, staff and user.

## Main Problem

A filing system is used to keep track of all records. Many non-profit organisations use physical files to record inventory, employee information, income, expenses and other information. These methods are unreliable as the files can go missing and are at a risk of theft, fire and flood. Also, when recording data, there is a possibility for human error and incorrect data being recorded. Accounts and donations are recorded using books filled manually. This is time-consuming and the risk of human error is high.

## My Solution

I am going to design a user-friendly, modern application to interface with a local database which will remove the need to record information on paper. The pain advantage is that all data and information is stored in the database and can be backed up instantly. The information can be restored with the backup in case the main computer is stolen or damaged. The program will offer password protection with different levels of user access therefor information cannot be altered by unauthorised individuals.

## Additional problems and solutions

### Problem - The input of data

* Handwritten information can be misspelt, misread or miswritten.

### Solution

* My program will request for the data to be entered into the system and will be verified with data validation techniques.

### Problem - Sorting and arranging data

* The information from the files are physically being sorted and arranged.

### Solution

* Data will automatically be sorted and manipulated instantaneously in several ways by the application.

### Problem – Access control

* There is currently no manner of controlling who can access and edit data stored in a filing cabinet.

### Solution

* Data would be protected by the database and only accessible through the application which has user access levels which restricts how users can interact with the data.

### Problem – Fading of ink and loss of files

* Old files can fade, tear and be damaged with water.

### Solution

* Information will be in permanent storage and will be backed up. The user will also be able to view the information clearly and the information will not suffer from fading over time or any type of deterioration.

# WHAT WILL THE PROGRAM DO?

* Provide login for multiple users.
* Allow the administrator to add, edit and delete records.
* Validate data being recorded to prevent invalid data being captured.
* Provide an interface to allow the management of expenses, payment of staff, ordering of stock, creating backups, manage donations, inventory, generate reports.
* Allow users to donate to the organisation.
* Capture donations.
* Alert management of low inventory.

# HOW WILL THE SYSTEM BE USED?

Users will be presented with a login screen when the application opens. Depending on the type of account they log in with, they will have access to certain functions.

Administrators have access to all parts of the application including some special menus for directly editing the database efficiently and checking for abnormalities.

Management will be shown more than the user but less than the administrator, they will have access to the statistics such as the expenses, income, inventory etc. They will also be able to manage finances such as paying expenses, ordering stock and paying staff.

Staff will be able to look up inventory and log donations.

Users will be able to manage their donations and create new donations.

# WHO WILL USE THE PROGRAM?

Administrators -Unrestricted access to the database and application.

Management – Do payments, access finances, create reports and do everything staff and users can do.

Staff – Log donations and access inventory.

Users – Donate money and view previous donations.

# FLOW OF APPLICATION

# DATA STRUCTURES

|  |  |
| --- | --- |
| STRUCTURE | USE/S |
| Database | Primary input/output file for system. Data will be extracted and written to the database for the majority of tasks in the system. Data validation techniques will be used to ensure the Integrity of Data. |
| Text Files | Text Files will be used to store user settings, file locations and Country-codes. |
| Arrays | Arrays will be used when performing aggregate calculations and sorting data using code construct. |
| Classes | Classes will be used to create custom component templates so that objects can be created from those classes. |
| Variables | Variables of a multitude of different data types will be used when processing extracted data. For example, when calculating total sales, a variable of type Real will be used. When checking whether a password has been entered correctly or not, a Boolean variable will be used. |

## CLASS DESCRIPTION AND CLASS DIAGRAM

### **TNaviPanel:**

**Purpose**:  
The NaviPanel class is used to create custom navigation panel objects   
for the [MultiView] component. I could not use regular labels because I needed to change the background colour.

#### **Attributes:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| olabel | TLabel | Label to display the name of the tab. |
| oimage | TImage | Image used to display an icon for the tab. |
| opanel | TRectangle | Panel to define the boarder of the tab label. |
| tabcontrolindex | integer | Store the number that corelates to the tab that the panel is assigned to. (Used to change the tab control to the correct tab.) |
| TabControl | TTabControl | Used to interface with the correct tab control. |
| selected | boolean | Indicates whether the tab is selected. |

#### **Methods:**

{ public methods }  
constructor Create(AOwner, AParent: TFmxObject)  
procedure SetLabel(caption: string)  
function GetLabel: string  
procedure SetImage(image: string)  
function GetTabIndex: integer  
procedure setTabIndex(index: integer)  
procedure setTabControl(AObject: TTabControl)  
procedure resetColor  
procedure setClicked

Methods (Continued):

{ private methods }  
procedure onMouseEnter(Sender: TObject)  
procedure onMouseLeave(Sender: TObject)  
procedure onPanelClick(Sender: TObject)

### **TDashPanel:**

**Purpose**:  
The DashPanel class is used to create custom dashboard panel objects   
for the [Flowpanel] component. I could not use regular labels because I needed to change the background colour, have an image in a consistent relative position and have it automatically position itself.

#### **Attributes:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| olabel1 | TLabel | Label to display the panels title. |
| olabel1 | TLabel | Label to display the panels subtitle. |
| oimage | TImage | Image used to display an icon for the dashboard panel. |
| opanel | TRectangle | Panel to define the boarder of the dashboard panel. |

#### **Methods:**

{ public methods }  
constructor Create(AOwner, AParent: TFmxObject)  
procedure SetLabel1(caption: string)  
procedure SetLabel2(caption: string)  
function GetLabel1: string  
procedure SetImage(image: string)   
procedure resetColor  
procedure setColor(Color: TAlphaColor);

{ private methods }  
procedure onMouseEnter(Sender: TObject)  
procedure onMouseLeave(Sender: TObject)  
procedure onPanelClick(Sender: TObject)

### **TDonationPanel:**

**Purpose**:  
The DonationPanel class is used to create custom Donation panel objects   
for the [Flowpanel] component. I could not use regular labels because I needed to change the background colour, have a set format and size.

#### **Attributes:**

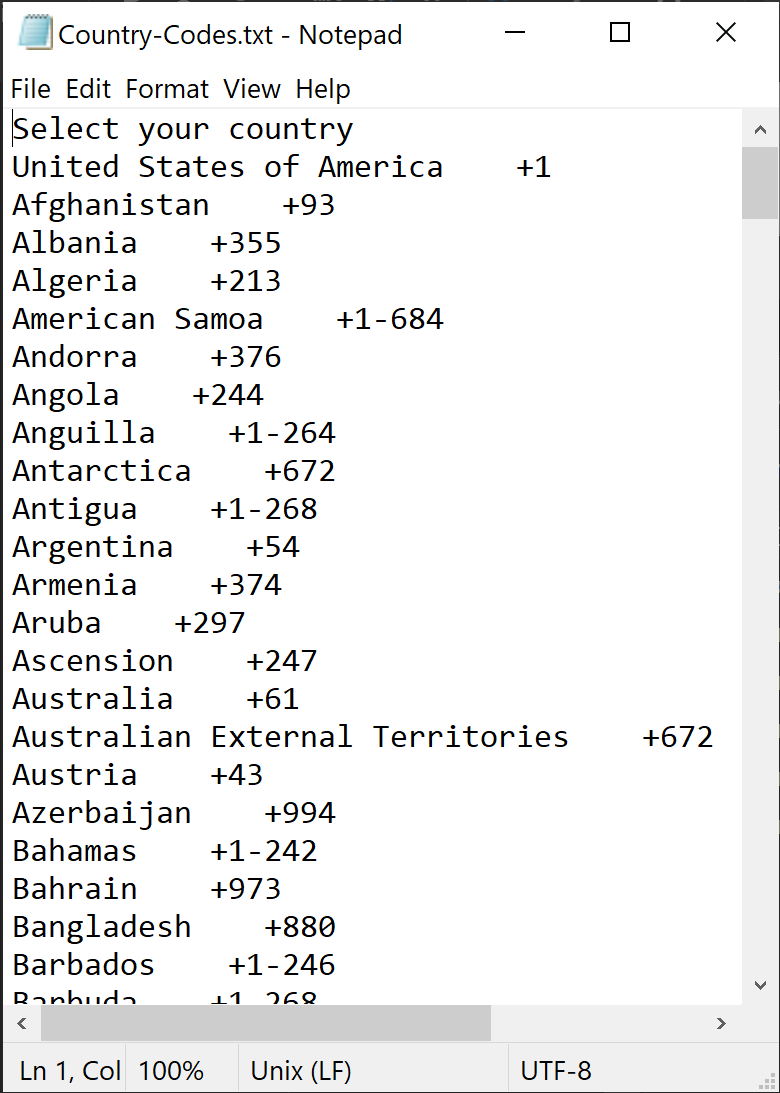
|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| olabelName | TLabel | Label to display the name of the donation. |
| olabelTime | TLabel | Label to display the time of the donation. |
| olabelDesc | TLabel | Label to display the description of the donation. |
| olabelQuantity | TLabel | Label to display the quantity of the donation. |
| opanel | TRectangle | Panel to define the boarder of the donation panel. |
| opanelTime | TRectangle | Panel to define the boarder of the Time section. |
| opanelName | TRectangle | Panel to define the boarder of the Name section. |
| opanelDesc | TRectangle | Panel to define the boarder of the Description section. |
| opanelQuantity | TRectangle | Panel to define the boarder of the Quantity section. |

#### **Methods:**

{ public methods }  
constructor Create(AOwner, AParent: TFmxObject);  
SetLabelName(caption: string);  
procedure SetLabelTime(caption: string);  
procedure SetLabelDesc(caption: string);  
procedure SetLabelQuantity(caption: string);  
procedure resize;

{ private methods }  
procedure setColor(Color: TAlphaColor);

# TEXT FILE AND ARRAY/ADVANCED PROGRAMMING CONCEPTS



Country-Codes.txt stores the country telephone codes. This makes it easier to update the country codes. A new Country-Codes file would need to be downloaded instead of editing the source code of the application.

+ Easier to update

+ Dynamically loaded

+ Reduces source code file size

+ Neater and less tedious code.

- May be interfered with by the user.

Format:

[Country] [whitespace] ‘+’ [code]

CountryCodes:TStringList – Used to store the country codes when loaded from the textfile.

CountryCodesFiltered:TStringList – Used to store the filtered country codes.

TFloatAnimation – Used multiple times to animate objects position and opacity. Complexity: 6/10

TMultiView – Used to save space and give the application a more modern feel when navigating through tab pages. Complexity: 5/10

StyleBook – Used to edit the appearance of existing components for a more modern feel. Complexity: 7/10

TFlowPanel – Used to automatically assign a position value to dynamically generated compnents.Complexity: 2/10

TRectangle – Used as a more basic foundation to replace the TPanel component in situations where more flexibility in components is needed. Makes it possible to edit the shape, color and border of a panel-like component. Complexity: 1/10

Object arrays – Used to store objects (instances of classes) to make it easier to organise and control objects. Complexity 5/10

TVertScrollBox – Used to show more objects in a smaller form. 2/10

# DATA INPUT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Source | Type | Format | Component |
| Country codes | Text file | String | [country][whitespace][code] | Combo box |
| Username | Keyboard | String | [username] | Edit box |
| Password | Keyboard | String | [password] | Edit box |
| Email | Keyboard | String | [email] | Edit box |
| Item Name | Keyboard | String | [item name] | Edit box |
| Item Description | Keyboard | String | [item description] | Edit box |
| Item Quantity | Keyboard | Integer | [number of items] | Edit box |
| Donation Date | Auto-generated (date extracted from computer) | Date | [dd/mm/yyyy] | label |
| Donation Quantity | Keyboard | Integer | [number of items] | Edit box |
| Expense Name | Keyboard | String | [name of expense] | Edit box |
| Expense amount | Keyboard | Real | [amount] | Edit box |
| Payment Date | Keyboard | Date | [dd/mm/yyyy] | Edit box |

# DATA VALIDATION

|  |  |  |
| --- | --- | --- |
| Name | Type | Validation |
| Email | String | Verify valid email |
| Item Quantity | Integer | Verify valid integer |
| Donation Quantity | Integer | Verify valid integer |
| Expense amount | Real | Verify valid real value |
| Payment Date | Date | Verify valid date |
| Donation Date | Date | Verify valid |
| Username and Password | String | No special characters or numbers, only letters no blank or whitespaces. Not empty/null. |

# DATABASE DESIGN

## ENTITY RELATIONSHIP DIAGRAM

# GUI DESIGN (SCREENSHOTS)

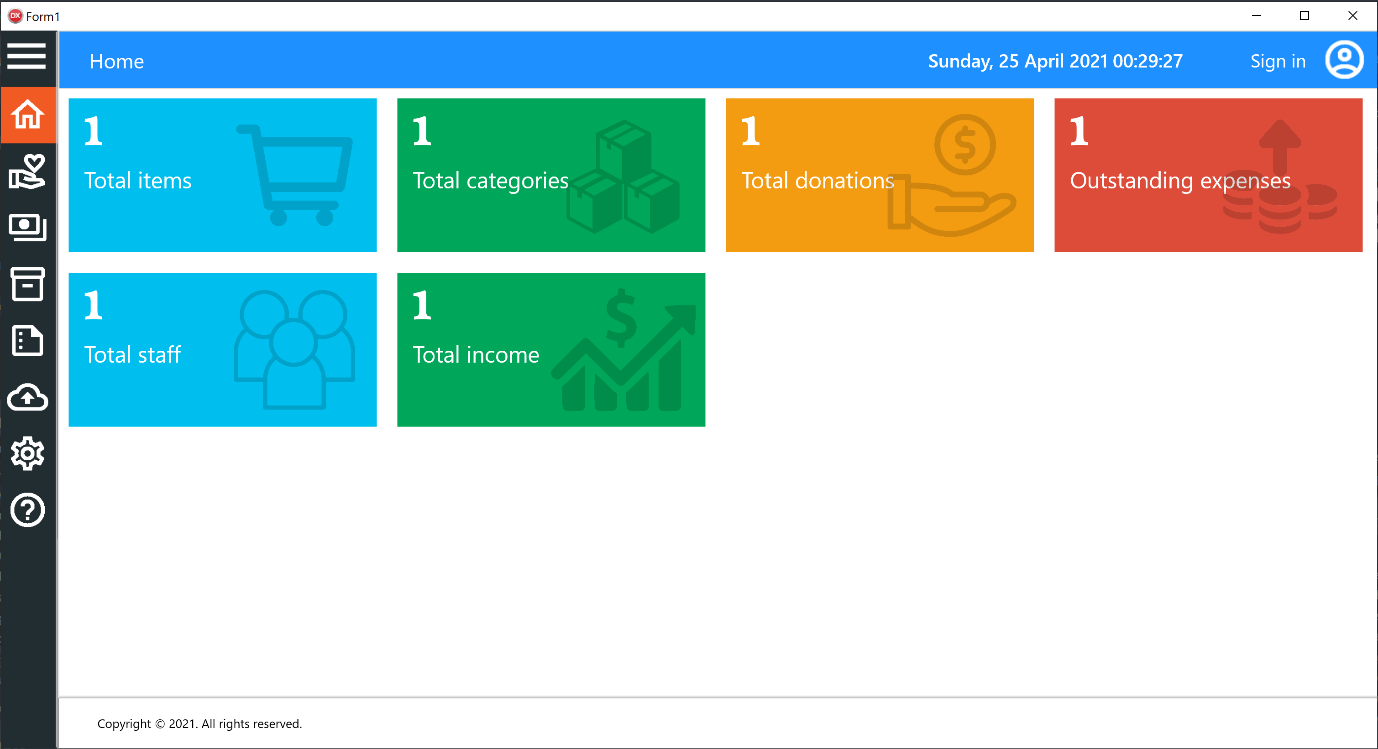


Figure 1: tblHome

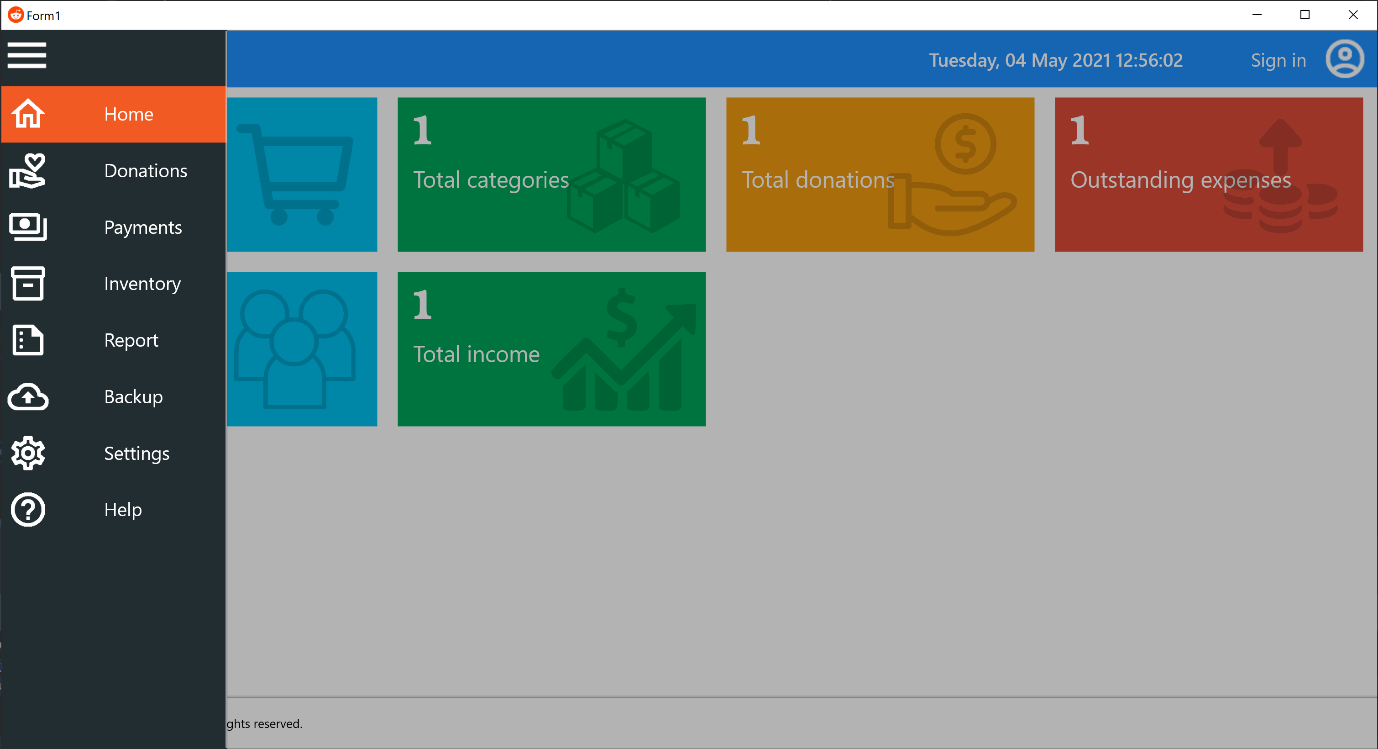


Figure 2: MutliView

# GUI DESIGN (SCREENSHOTS) [Continued]

Image2
tabDonations

Figure 3: tblDonations

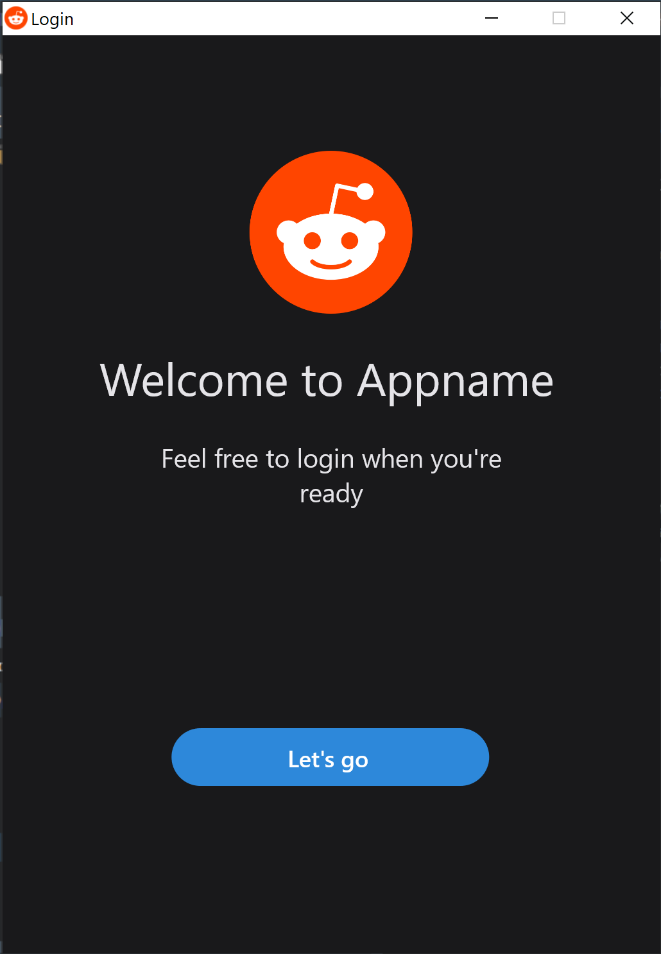


Figure 4: Login screen 1

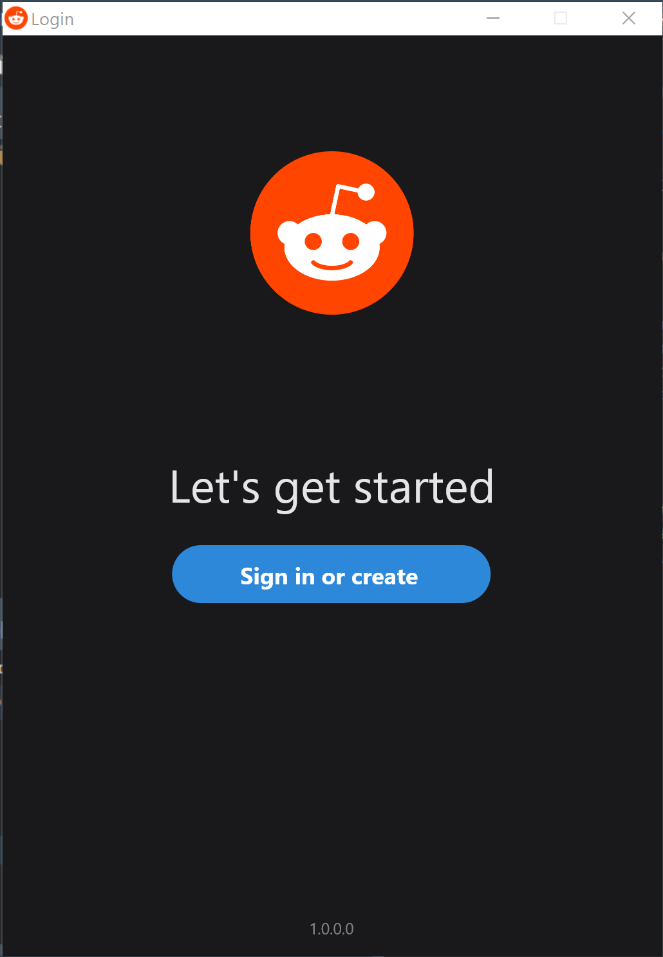


Figure 5: Login Screen 2

# GUI DESIGN (SCREENSHOTS) [Continued]

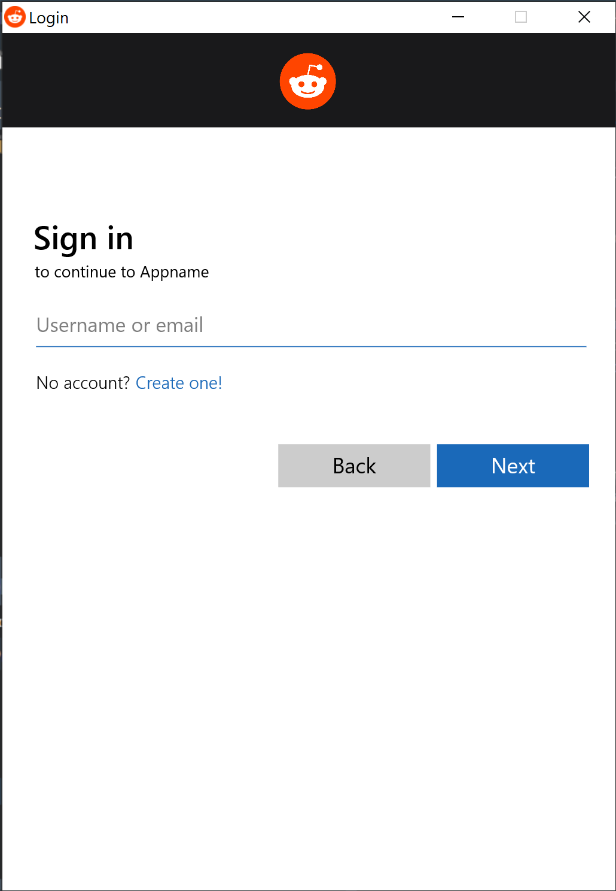


Figure 6: Login screen 3 - Sign in

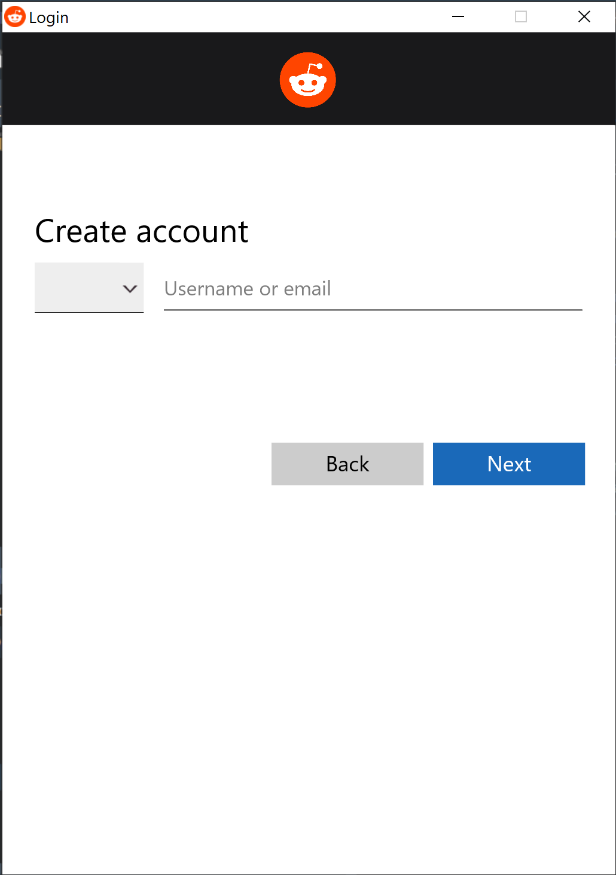


Figure 7: Login screen 4 - Create account



Placeholder icon.

<https://www.reddit.com>

# COMPONENTS USED

* TabControl – Navigating frames.
* MultiView – Modernise the look of the tabcontrol.
* Stylebook – Edit components visually.
* Image – Display icons.
* FlowLayout – Encapsulate dynamically created objects.
* Panel – Organise components.
* Timer – To update components periodically.
* Label – To display text.
* Rectangle – More flexable than a panel.
* TabItem – To be used with TabControl.
* Editbox – Input text.
* Button – Receive button presses.
* Combo box – Select country code [], select number of records to display[Image 2 - tabDonations].
* TImageList – Store the images in the code itself so it will always be together with the executable file.

# DATA PROCESSING

* Help – When the help tab is clicked or any help button, it will show the user helpful information.
* Login – Username is confirmed with the database.  
  if the username does not exist, an appropriate error message is shown. If the editbox is blank, an appropriate error message is shown.  
  If the username exists, the next login screen is shown and the user is asked for their password.  
  The username must not be blank. The username cannot contain spaces.  
  The username must be in the users table in the database.
* Login - Password is confirmed with the database.  
  The users entered password is hashed with base64 and the hash is compared with the hash stored in the database. If it does not match, an appropriate error message is shown. If no password is entered, an appropriate error message is shown.

The password must not be blank. The password is hashed and must be exactly the same as the users password hash in the database  
If the password is correct, the hashes will match and the user will be shown the main form (dashboard). And depending on the users permissions, they are shown more or fewer controls.

* Create new user – Phone number  
  When the create new user button is pressed on the login form, the user is prompted to enter their phone number. The phone number is validated and if valid, the user is shown the create password screen.

The phone number must contain 10 digits. The phone number cannot contain any special characters or letters. A prefix must be chosen. The phone number cannot be in the database already (it must be unique).

* Create new user – Password  
  User is prompted to enter a password for their new account.  
  If the password is valid, the user is passed on to the personal details screen.

The password must not be blank. The password must contain at least 8 characters, 1 uppercase. The password cannot contain spaces.

* Phone number prefixes  
  When the login form is created, the phone number combo box is populated with international phone number prefixes from a text file.
* Phone number combo box – search  
  When a key is pressed while the combo box is dropped down, the search function will start, each key press is logged and added to the search. A string list is created and any prefix containing the search string is added to the temporary string list. While the search is active, the combo box is populated with the temporary list. When the combo box is closed,
* Search – Donations  
  When a user searches through donations, all the existing donation objects are destroyed and the database is searched, objects are recreated for every donation record that meets the search requirements.
* Report  
  When a report is requested, a report is generated based on the requested data between the requested dates. The report shows data on donations, inventory and money etc.
* Dashboard  
  The dashboard panels indicate important information and statistics and make it easily accessible. These values are calculated by the application using data from the database and is displayed on these panels.

# DATA OUTPUT

|  |  |  |
| --- | --- | --- |
| **OUTPUT** | **FORMAT** | **COMPONENT** |
| Total items | Integer | Label |
| Total categories | Integer | Label |
| Total donations | Integer | Label |
| Outstanding expense | Currency | Label |
| Total staff | Integer | Label |
| Total income | Currency | Label |
| Time | String (Converted from datetime) | Label |