# C++ Project Documentation

Barry O'Connor [N0813926]

SOFT10101: Computer Science Programming Project

# 1 Aims

I like to keep up to date with my finances and regularly use an Excel spreadsheet. I use this several times a week to keep track of my spending and it's become a bit of a chore to maintain as I need to go to my bank, manually enter the data and make sure that the calculations in each cell are correct.

I have several categories of expenditure, for example food, and tracking budgets for each as well as an overall monthly budget is difficult. My bank has also started offering CSV format downloads of statements, so I feel like it would be possible to simply import a file once in a while and take a lot of the manual work out of the process.

I wanted to take this opportunity to create something I could use as a replacement for this system, which I feel I don't want to use anymore. Creating something I would use would give an incentive to produce a good quality project and I can continue to tweak the application after the project finishes, really making it work well for my needs. This also means that I want the project to be modular and easily changed. I want to be able to add new features and tweaks as I use the application.

# 2 Requirements Analysis

## 2.1 User Interface Requirements

I find Excel fiddly to enter data into. I keep having to create rows when I have more data and this causes issues with the calculations I have implemented in the workbook. This involves a lot of adjusting and I would like this application to be a more familiar and focused interface. Something I can navigate easily, tailed to what I need and very comfortable to use. I find it easier to fill in a form and edit things that way, so from the start the main requirement for this project was that it be a Windows GUI.

This would enable me to create windows and forms and use already existing controls to make the process of inputting data as painfree as possible. This would also allow me to design the interface with asthetics in mind and make something I know I would enjoy using regularly.

## 2.1 Storage Requirements

The application will need to store increasing amounts of financial information so it makes sense to use a database for this project. In my Python project, I used SQLite which was great but lacked the security of the bigger database systems, however the portability of a single file database is fantastic for projects so I have reused it in this project. I will retrofit the project to SQL Server over the summer but for demo purposes, SQLite is fine.

I will however look into encryption of the data, whether that is done on a database level or by the application when storing and retrieving data. Financial information is obviously confidential and security needs to be factored in.

## 2.1 Functional Requirements

The broad aims of the application can be broken down into smaller, more defined requirements which will shape the functionality of the application. These basic requirements must be met for the

application to work for me and be useful to me. Additional features are suggested with the aims of further developing the application as a long term project.

The Minimum Requirements for the application are:

- Account Based:
  - An account for each user. This allows for privacy and some element of security regarding very private data.
  - A login screen so that the user can access their account.
  - A registration form to allow for the creation of new accounts (I have some friends and family who would like to use the application).
  - An account page where a user can update their personal details and some form of password recovery system to allow for an easy way to recover lost passwords..

- Import/Export
  - I would like to be able to import and export from the system. My bank allows for CSV downloads of transactions and this would add a great deal of value to the application for me.
  - The transactions should be listed in a clear way, with a quick and easy way to add, edit and delete transactions using forms.

  - It would be useful to be able to perform actions like being able to find all transactions in a given date range or to be able to sort by the type of transaction.
  - It would also be useful to have built in transaction types which meet both my needs in terms of things like Food, Travel, Bills but also those that match the entries in my bank statement – POS, BACS etc so that I don't have to change data I get from my bank.
- Interface
  - The main interface needs to be useful without being cluttered. I need to be able to quickly add, edit and delete individual transactions without too many screens.
  - I want to be able to view a lot of transactions at once so using a traditional form is not an option, I will look at the common controls and see if there is a datagrid or something similar to allow me to view and sort the list.
  - There must be a total visible at all times, which reflects changes in the list as I go. It would be helpful if this was updated alongside any date range functionality so I could see my totals for a given period.
  - As a minor consideration and time dependant, I would like to use controls for easy date input as formatting dates can be a painful experience with many transactions to enter.
- Code Updates
  - The interface and underlying code needs to be updatable. If I feel I don't like a certain feature or something new that I want to add comes to mind, I want the underlying framework to allow for those changes to be made fairly quickly.

# 3 Design

## 3.1 Database Diagram

The database will mainly consist of the following two tables, one for Users and one for the list of transactions. The transaction table will be heavily influenced by the layout used by my bank so will

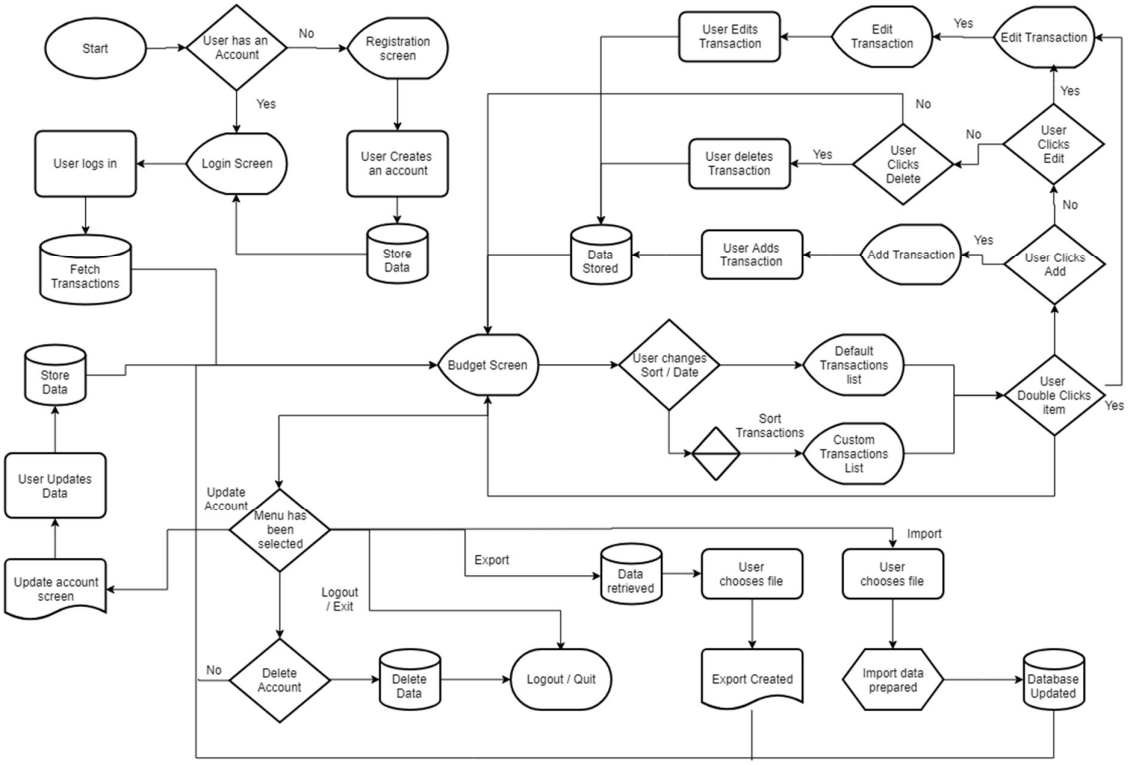
likely reflect the CSV supplied by them. The tables have a one to many relationship, with users.userID being used as a foreign key in the tblTransactions.user_id field.

| TransactionsTable | | | | |
|---|---|---|---|---|
| **Fieldname** | **Type** | **Description** | **Null** | **Info** |
| transID | integer | Unique record ID | NO | PK Unique Auto Increment |
| userID | text | user_id (from users table) | NO | |
| transDate | text | Date in format “yyyy-mm-dd” | NO | |
| transType | real | POS BACS FOOD BILL etc | NO | |
| transDescription | real | Description as appears on my statements | NO | |
| transAmount | real | The value of the transaction | NO | |

| users Table | | | | |
|---|---|---|---|---|
| **Fieldname** | **Type** | **Type** | **Null** | **Info** |
| userID | integer | Unique record ID | NO | PK Unique Auto Increment |
| userFullName | Text | The User’s full name | | |
| userUsername | Text | username | NO | |
| password | Text | User password | NO | |
| userEmail | Text | User email address | NO | |
| userSecretQuestion | Text | A security question | NO | |
| userSecretAnswer | Text | Answer to security question | NO | |

## 3.2 Program Flow Chart

A program flow chart for the application is included on the following page to give an overview of where the.

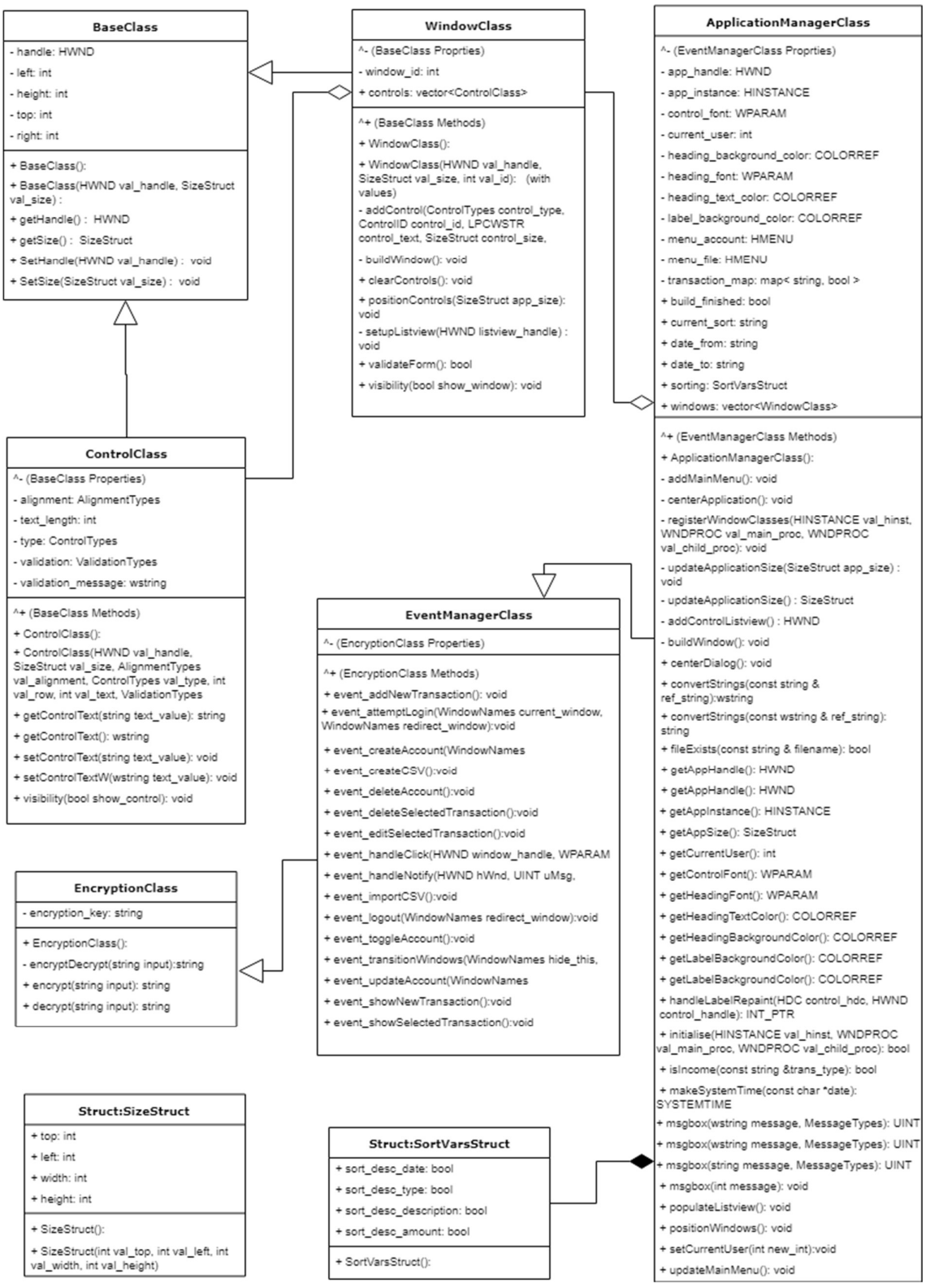



## 3.3 Classes

While developing this application, I struggled keeping track of all the controls using the standard Win32 system which relies heavily on handles to controls and windows. Storing these variables globally became an issue very quickly as did differences in different types of string variables. To help ease these issues and to help futureproof the application, I created several wrapper classes to contain the controls and windows and make the job of calling or referencing various parts of the application easier for myself.

The program began with an Application, Window and Control class. Since Win32 classes controls as windows, this soon changed to define a Base class which would be inherited by Window and Control. On top of this, I implemented an Event Handler class and a class to contain the encryption. These are inherited by the Application class to keep everything together, while allowing me to separate out functionality – all events are handled by the Event Handler, so I can quickly work out which file the code I want is in, which helps with the goal of making the code easy to maintain.

Using classes has been a real benefit allowing me to code vectors to store new controls and windows, allowing me to be able to push_back() new controls and have default constructors build the items. I've also been able to implement features like alignment types and validation types, so I can set up controls to self-validate as part of a function when a form is submitted. The following diagram outlines the classes used by the application.


BaseClass
- handle: HWND
- left: int
- height: int
- top: int
- right: int
+ BaseClass():
+ BaseClass(HWND val_handle, SizeStruct val_size) :
+ getHandle() : HWND
+ getSize() : SizeStruct
+ SetHandle(HWND val_handle) : void
+ SetSize(SizeStruct val_size) : void
WindowClass
^- (BaseClass Proprties)
- window_id: int
+ controls: vector<ControlClass>
^+ (BaseClass Methods)
+ WindowClass():
+ WindowClass(HWND val_handle, SizeStruct val_size, int val_id): (with values)
- addControl(ControlTypes control_type, ControlID control_id, LPCWSTR control_text, SizeStruct control_size,
- buildWindow(): void
+ clearControls(): void
+ positionControls(SizeStruct app_size): void
- setupListview(HWND listview_handle) : void
+ validateForm(): bool
+ visibility(bool show_window): void
ApplicationManagerClass
^- (EventManagerClass Proprties)
- app_handle: HWND
- app_instance: HINSTANCE
- control_font: WPARAM
- current_user: int
- heading_background_color: COLORREF
- heading_font: WPARAM
- heading_text_color: COLORREF
- label_background_color: COLORREF
- menu_account: HMENU
- menu_file: HMENU
- transaction_map: map< string, bool >
+ build_finished: bool
+ current_sort: string
+ date_from: string
+ date_to: string
+ sorting: SortVarsStruct
+ windows: vector<WindowClass>
^+ (EventManagerClass Methods)
+ ApplicationManagerClass():
- addMainMenu(): void
- centerApplication(): void
- registerWindowClasses(HINSTANCE val_hinst, WNDPROC val_main_proc, WNDPROC val_child_proc): void
- updateApplicationSize(SizeStruct app_size) : void
- updateApplicationSize() : SizeStruct
- addControlListview() : HWND
- buildWindow(): void
+ centerDialog(): void
+ convertStrings(const string & ref_string):wstring
+ convertStrings(const wstring & ref_string): string
+ fileExists(const string & filename): bool
+ getAppHandle(): HWND
+ getAppHandle(): HWND
+ getAppInstance(): HINSTANCE
+ getAppSize(): SizeStruct
+ getCurrentUser(): int
+ getControlFont(): WPARAM
+ getHeadingFont(): WPARAM
+ getHeadingTextColor(): COLORREF
+ getHeadingBackgroundColor(): COLORREF
+ getLabelBackgroundColor(): COLORREF
+ getLabelBackgroundColor(): COLORREF
+ handleLabelRepaint(HDC control_hdc, HWND control_handle): INT_PTR
+ initialise(HINSTANCE val_hinst, WNDPROC val_main_proc, WNDPROC val_child_proc): bool
+ isIncome(const string &trans_type): bool
+ makeSystemTime(const char *date): SYSTEMTIME
+ msgbox(wstring message, MessageTypes): UINT
+ msgbox(wstring message, MessageTypes): UINT
+ msgbox(string message, MessageTypes): UINT
+ msgbox(int message): void
+ populateListview(): void
+ positionWindows(): void
+ setCurrentUser(int new_int):void
+ updateMainMenu(): void
ControlClass
^- (BaseClass Properties)
- alignment: AlignmentTypes
- text_length: int
- type: ControlTypes
- validation: ValidationTypes
- validation_message: wstring
^+ (BaseClass Methods)
+ ControlClass():
+ ControlClass(HWND val_handle, SizeStruct val_size, AlignmentTypes val_alignment, ControlTypes val_type, int val_row, int val_text, ValidationTypes
+ getControlText(string text_value): string
+ getControlText(): wstring
+ setControlText(string text_value): void
+ setControlTextW(wstring text_value): void
+ visibility(bool show_control): void
EventManagerClass
^- (EncryptionClass Properties)
^+ (EncryptionClass Methods)
+ event_addNewTransaction(): void
+ event_attemptLogin(WindowNames current_window, WindowNames redirect_window):void
+ event_createAccount(WindowNames
+ event_createCSV():void
+ event_deleteAccount():void
+ event_deleteSelectedTransaction():void
+ event_editSelectedTransaction():void
+ event_handleClick(HWND window_handle, WPARAM
+ event_handleNotify(HWND hWnd, UINT uMsg,
+ event_importCSV():void
+ event_logout(WindowNames redirect_window):void
+ event_toggleAccount():void
+ event_transitionWindows(WindowNames hide_this,
+ event_updateAccount(WindowNames
+ event_showNewTransaction():void
+ event_showSelectedTransaction():void
EncryptionClass
- encryption_key: string
+ EncryptionClass():
- encryptDecrypt(string input):string
+ encrypt(string input): string
+ decrypt(string input): string
Struct:SizeStruct
+ top: int
+ left: int
+ width: int
+ height: int
+ SizeStruct():
+ SizeStruct(int val_top, int val_left, int val_width, int val_height)
Struct:SortVarsStruct
+ sort_desc_date: bool
+ sort_desc_type: bool
+ sort_desc_description: bool
+ sort_desc_amount: bool
+ SortVarsStruct():

# 4 Testing

During production of the application any expected issues were recorded and any unexpected bugs or issues were included as part of the review process. Where possible these have all been addressed before completion date. The following are the test cases used in testing, which may be followed independently to verify.

## 4.1 Test Cases

| Login Page | | | | |
|---|---|---|---|---|
| **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** |
| Log in with valid data | 1)Open program<br>2)enter username<br>3) enter password<br>4) click login button | Username: default<br>password: password | Log into application successfully | Login Successful |
| Log in with invalid data | 1)Open program<br>2)enter username<br>3) enter password<br>4) click login button | Username: something<br>password: password | Login failure with message | Login failure with message |
| Open Register Screen | 1)Open program<br>4) click create account button | N/A | Redirect to Create an Account | Redirect to Create an Account |
| **Create an Account / Edit Account Pages** | | | | |
| **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** |
| Create an account with random data | 1) enter Full Name<br>2) enter username<br>3) enter password<br>4) enter email<br>4) enter Secret Question<br>5) enter Answer<br>10)Click create account button | Full Name: Peter Parker<br>Username: spiderman<br>password: shhsecret<br>email: sometest@test.com<br>Secret Question: (choose any)<br>Answer: My Answer | New user created, message shows | New user created, message shows |
| Create an account with blank data | 1) click create account button | N/A | Warnings issued | Warning issued |
| Choose an invalid email | 1) enter email<br>2) click create account button | email: peter@parkercom | Warning issued | Warning issued |
| **Welcome/Home Page** | | | | |
| **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** |
| Log out | 1) Click Account Menu<br>2) Click Log Out | N/A | Redirected to Login | Redirected to Login |
| Delete account but cancel | 1) Click Account Menu<br>2) Click Delete Account<br>3) click No | N/A | Nothing (account remains) | Nothing (account remains) |
| Delete account | 1) Click Account Menu<br>2) Click Delete Account<br>3) click Yes | N/A | Redirected to login and account deleted | Redirected to login and account deleted |
| Click Update Account | 1) Click Account Menu<br>2) Click Update Account | N/A | Redirected to Budget | Redirected to Budget |

| | | | | |
|---|---|---|---|---|
| Click Exit | 1) Click File Menu<br>2) Click Exit | N/A | Program quits | Program quits |
| **Budget Page** | | | | |
| **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** |
| Data sorting | 1) click one of the headers in the listview | Any header | Listview changes to sore data | Listview changes to sore data |
| Edit button checks | 1) click edit transaction or delete transaction after selecting a listview item | N/A | Dialog opens with correct data | Dialog opens with correct data |
| Delete / Edit button checks | 1) click edit transaction or delete transaction without selecting a listview item | N/A | Warning issued | Warning issued |
| Delete transaction but cancel | 1) pick any transaction<br>2) click delete transaction<br>3) Click cancel when prompted | N/A | transaction remains | transaction remains |
| Delete a transaction | 1) change date selector<br>2) click on a diary entry<br>3) click delete transaction<br>4) Click OK when prompted | N/A | transaction is deleted | transaction is deleted |
| Update a transaction | 1) pick any transaction<br>2) click edit<br>3) enter updated info<br>3) Click edit when prompted | Date: 08 April 2019<br>Type: select any<br>Description:Hairdresser<br>Amount: 12.00 | transaction is updated | transaction is updated |
| Show the create interface | 1) click on add transaction button | N/A | Blank Create dialog is shown and save button | Blank Create dialog is shown and save button |
| Insert a new transaction | 1) click on add a new transaction button<br>2) enter date<br>3) enter type<br>4) enter description<br>5) enter amount<br>6) click edit transaction | Date: 09 April 2019<br>Type: select any<br>Description:Butcher (sausages)<br>Amount: 6.00 | Item is created | Item is created |
| Backup transactions | 1) click on the Account -> Export menu | N/A | File is created in the current directory | File is created in the current directory |
| Import transactions | 1) click on the Account -> Import menu | N/A | Imports contents of file | Imports contents of file |
| **Validation checks** | | | | |
| Check field not empty | Try and submit the login form with empty values | N/A | Message shown | Message shown |
| Check email | Try and submit Create account with non email | N/A | Message shown | Message shown |
| Check amount | Add a transaction with s non money value | Enter any text | Message shown | Message shown |
| Check date | Try and add an invalid date | Manually type datepicker | autofixed | autofixed |

# 5 Critique

I have coded in several languages before so I wanted to challenge myself with something new in this project. I originally tried to use CLI/CLR but felt that it strayed too much from pure C++ to really fit with a C++ project. In the end I stopped working on that version and began working with Win32 which is a much more “classic” way of doing things.

Initially progress was good and I was enjoying learning the new approach to coding windows programs. I’m familiar with Visual Basic and some other languages, so Windows development feels comfortable to me. However I was not prepared for the difficulty associated with Win32. It operates at such a low level that even the most basic of tasks takes several lines of code. There are also many implementations of strings, characters, arrays of characters and pointers to both to consider. Learning to deal with all of those has been an extremely difficult task and has made progress slower than I would have liked but learning how Windows works at such a low level has been rewarding of itself and I’ve been forced to use classes a lot, helping me to understand more about C++.

This extra level of complexity had a knock on effect and I struggled to get everything done. While I’m happy with the final program overall, there are some minor issues that I feel need a little work.

It’s the first time I’ve tried to use encryption on my own and there have been a few issues with that. It’s just a really basic XOR cipher and a Base64 conversion for good measure. I was wanting to use AES but I couldn’t understand how to create and use the libraries online – I need more time to understand how to compile those and use the compiled libraries. I’m happy with the proof of concept but stronger encryption would be needed moving forward.

Encrypting the import so that it works with the existing system has proved tricky and my code has issues with the imported values being surrounded by placeholders. There have also been issues with the database being unable to sort encrypted date values. All in all I feel like encrypting the communication with the database would be preferable.

Regarding the import, I also discovered that my bank uses, what seems to me, a strange implementation of CSV. The description field has commas in the field, which is tricky but also delimits the fields with quote marks and for some strange reason includes a single apostrophe at the start of only the description field. I need more time to work out how to handle that. An example of this follows:

```
15/04/2019,POS,"'5585 12APR19 C , ALDI 14 779 , BARNSLEY GB",-7.22,-
1392.33,"'O'CONNOR BJ","'557023-86221809"
```

I have begun using the software in earnest for my own data and I’m mostly happy with the results. I find there’s less work involved and I can get more done in a smaller amount of time using the app over Excel. Dates are a breeze with the datepicker controls and I can add or edit information in much less time than before.

Moving forward I do have some suggestions for improvement. I designed the app to run at 800x600 as an exercise for myself to not make the interface too cluttered. I certainly don’t think there’s any issues with the application at that size but I will increase that so I can add some additional items to the screen. I’d also like to look into coloring the listbox items so I can have a really quick overview of the information. This is done via custom paint messages which was a little too indepth for me to fully understand when I started looking at doing it. I also want to catch the DEL keypress on the listview so I can delete using that key instead of pressing a button.

I’d also like to look at making some form of graph page with the ability to set personalised monthly goals. So, for example, I might decide that I want to budget £200 on food but that’s difficult to track unless you do a monthly shop. If you shop once or twice a week, it’s really difficult to work out how much of that budget is left without sitting down with a calculator and working it out. If I had a way of setting monthly goals and maybe a set of graphs showing planned and actual spend on each goal graphically, I think it would be a great addition. I’d be able to tell how much budget I still had left for each goal.