Modern Bank

SYSTEM DESIGN DOCUMENT

MODERN BANK ATM KIOSK SYSTEM

JULY 2020

Scenario/context

A bank has been developing a simple login system allowing their customer to log into the system and check their current balance. This system however is outdated and does not provide much response to the user's current errors or invalid entries. A project has been commissioned to further this application to better inform the user.

Task 2: Design a piece of software using event driven programming solution that identifies the components, data/file structures, makes effective use of the Integrated Development Environment and incorporates onscreen help for users (e.g. Error messages, end user guidance). The design components can include any of the following:

- public
- private
- protected
- text boxes
- combo boxes
- list boxes

check boxes

- radio buttons
- listeners
- exceptions
- projects
- classes
- variables
- strings
- arrays
- images.

Evidence to be handed in: Electronically submitted in Zip format

Table of Contents

Table of Figures

Overview

Modern Bank, an Irish Banking Organisation, have been in development of a simple login system that will allow their customers to log into the system and check their current balance. The system that they have developed however is outdated as it does not provide much response to the user's current errors or invalid entries. Modern Bank is looking for a more updated ATM Kiosk Style application that will display these errors and invalid entries.

In this document we will run through the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts and detailed design that relates to the creation of the Modern Bank ATM Kiosk System. The aim of this system is to create software that will allow the customer to login to the system and check their current balance, to change their pin, view and edit banking details, view their transaction history, add a payee, transfer money between accounts/payees and display errors or invalid entries when a customer is accessing the Modern Bank ATM Kiosk System.

1.0 Introduction

1.1 Purpose and Scope

The purpose and scope of this project involves the creation of an ATM Kiosk System on behalf of Modern Bank. We will produce a piece of software that will allow customers of Modern Bank to be able to login to the system check their current balance, to change their pin, view and edit banking details, view their transaction history, add a payee, transfer money between accounts/payees and display errors or invalid entries when a customer is accessing the Modern Bank ATM Kiosk System.

The main purpose is to improve Modern Bank's current ATM Kiosk System. The system that they have developed already is outdated as it does not provide much response to the end-user's current errors or invalid entries. They are looking for a more updated ATM Kiosk Style application that will display these errors and invalid entries which we are hoping to create.

The primary user of this system is to be current customers of Modern Bank. These users will already have their account credentials – such as their Account Number and Account PIN, which will serve as a username and a password to login to the system. This ATM Kiosk will not allow new customer sign-ups, as the customers must already have an account with the bank in order to access their information.

This ATM Kiosk does not have the infrastructure for cash withdrawals or deposits. It also does not allow for the user to insert their card to access the system. The purpose of the system is to allow Modern Bank users to login with their account credentials as discussed above.

The system itself will be simple and clean. There will be a timer in effect within the system, that will set off after a certain time, the user will have the choice to either continue or the session will end. After this has occurred, the user will gain additional time, but again, another timer will be in effect, and when the time on the second timer has ended, the session will timeout, and the user will have to login again, if required. This is required within the system for security purposes, as this system is dealing with a customer's confidential information, and allows for money to be transferred, it is essential to have security practices in place. This is especially important if a user forgets to log out of the system, which could leave their sensitive information exposed to another user that may use the system next.

After the customer has logged in to the system, they will be presented with the main system screen. This screen will display all of the features and options of the system. There will also be a help section available on each screen of the system – to aid the customer and give further guidance for the end user.

A customer will be able to change their current PIN. They will not be required to enter their Account Number again, as they have already accessed the system, but for security purposes, it will be require for the customer to enter and confirm their current PIN again. Once this has been confirmed and valid, the system will allow the user to create a new pin. The system will also require the end user to confirm the PIN again. Once this is confirmed, the user will be able to change the PIN successfully.

A customer has the option to check their current balance.   Each option from the main menu will have a button that will allow the customer to go back to the main menu. After a customer has viewed their balance, they are able to select other options within the system.

The system will also allow the customer to view their personal details. This screen within the system will display the customer's First Name, Last Name, Date of Birth, Email Address, Phone Number, Address, City, County and Postcode. When the screen is loaded, all of this information will be read only, so that the customer is able to view their details. If they wish to update or change their information, they will have the option to select a button to edit their details. When they have finished updating their information they are able to save their changes. This screen will also feature a checkbox that will allow the user to view their banking details – their Account Number, Balance and PIN – when the box is checked. This information is not able to be edited, and it is only viewable when the checkbox is checked, to ensure that the user can close it if required, for security purposes, so that their sensitive information is not left on screen.

The next option on the main screen will be to add a payee. The system will allow the customer to input the Account Information of another account that they would like to transfer funds to in the future. This information will be stored in a database, which will then be accessible as a choice of an account to transfer to in the transfer screen.

The transfer screen will ask the customer to select an account to transfer from -   this is their own account, but they may have multiple accounts, but in most cases it will likely be the single account. The page loads with their current account number and balance on the page so that the customer is aware of the account that they are transferring funds from, and the amount of funds they have available. They can then select a payee – either to a different account of theirs that is not selected (if applicable) or a payee that they have added, which would have been added in the Add a Payee section of the Kiosk. They can then use the on-screen keyboard to select how much they would like to transfer. They will not be able to transfer more cash than they have available in their account balance.

The final option on main the screen will include the customer's transaction history. This will display the history of any transactions from the customer. This information will include Debit (Out), Credit (In), Date of Transaction, and a description of the transaction.

The main screen will load with a welcome message with the first name of the customer that has logged in. Each page will also feature a close button so that the customer is able to exit the program if required. There will also be an exit button on the main screen that will exit the program.

The system will be created using C# Language while the database for this project will be created using SQLite and the system used to manage this database will be DB Browser.

1.2 Project Executive Summary

Modern Bank have developed a simple login system that will allow their customers to log into the system and check their current balance. The system that they have currently developed however is outdated as it does not provide much response to the user's current errors or invalid entries. Modern Bank is looking for a more updated ATM Kiosk Style system that will allow their customers to login and display errors and invalid entries as they occur.

Modern Bank

Modern Bank, an Irish Banking Organisation, is requesting the development of a Graphical User Interface application to be built using an Object-Oriented and Event Driven Programming Approach (OOED – Object Oriented Event Driven). This application will replace their current system which is outdated.

The project requirements involve an Object-Oriented Event-Driven GUI program that is to be an ATM Style Kiosk application. The requirements and methods that Modern Bank would like to be featured within the scope of this application includes:

1. Login System – Using the Account Number and Account Pin to act as a username and password to the system.
2. View Current Balance – Allowing the customer to see their current balance on screen
3. View and Edit Personal Details – Allow the customer to see their current details, and make amendments where required.
4. View Banking Details – Allowing the customer to view private banking details, such as Account Number, Account PIN and Balance.
5. Transfer – Ability to transfer money from one account to another
6. Transaction History – Allow the user to view their transaction history, display their in/out funds with a description
7. Timer – A timer that will log the user out after a certain amount of time
8. Events – At least 3 different event types to be implemented within the program, e.g. Timer, onMouseOver, onHover, onCheck, etc.
9. Database to store customer's account number, account pin, balance, address, name, etc. Multiple tables within the database.
10. ATM Connector Class – A class for all SQL query's within the system

1.2 Deliverables

This section will provide the assets being produced by the project and the key features.

Completed Software using C#

1. BankData.db Database File
2. Help page using HTML integrated within the system

Documentation

3. Object Oriented Programming Report Documentation
4. Test Log Report Documentation

Functionality

1. The system will begin with a screen that acts like a screensaver. It will require the user to click anywhere to load the system. This will open the Login screen.
2. The user will be prompted to Login. The customer must use their credentials already created via the bank. They will be able to access the system using their Account Number and Account PIN. They must use the onscreen keypad in order to login.
3. After a successful login, the ATM Kiosk system will bring the customer to the main screen which will feature the different options and functionality of the system. This page will also feature a welcome message with the user's first name.
4. The customer will have the ability to click a button to view their balance. This will open a new screen that will display the current funds in their bank account.
5. The customer will have the ability to change their current PIN. Their current PIN must first be confirmed, and then the system will allow the user to input a new pin which also requires a confirmation and to be entered twice.
6. The customer will have the ability to view and edit their personal details. They will also be able to view their banking details.
7. The user will have the ability to Add a Payee, that can be used later to transfer funds.
8. The customer will have the option to transfer funds from one account to another.
9. The customer will have the option to view their transaction history.
10. The customer will have the option to exit the program.
11. A timer will be present within the system that will enable a session to timeout.

1.3 Justification

The justification for this project came from the internal issues within Modern Bank to do with their current system which has been identified by the company to be outdated. They have explained that their current system does not handle errors or invalid entries the way that they would like and are looking for a new system that will incorporate this error-handling. As this system is for a bank, security places an integral part within the system, and for this reasoning error-handling is essential as the ATM kiosk system will deal with sensitive and personal information.

1.4 Design Constraints

1.4.1 System Constraints

There are a few constraints affecting the development of this software, the main constraints being the physical hardware. As this ATM Kiosk will not feature a card reader or allow the ability to withdraw or deposit cash, it places certain constraints within the system. The system will only allow the customer to transfer funds from one account to another on the screen – it does not deal with any physical cash, and all functionality is done through the system itself.

The database system could be updated within the future. The current system used to store data is used with free software, which can cause some design constraints as it is more restrictive than perhaps enterprise databases, as it is only entry-level. At the moment, this database is not able to handle dates in a way that the developers find satisfactory which has made the use of SQLite a constraint on the project.

Security within the system was also a big design constraint – as we are dealing with confidential information it is a big factor to ensure that the system is GDPR compliant and also secure.

1.4.2 Assumptions

There are a number of assumptions made within this project. For example, the main assumption is that customer already has an account – an Account Number and a PIN – made previously to their use of this system. This system does not allow the customer to create a new account. Another assumption is that the customer would have their account number and pin on hand, as they are unable to access the system without it.

Another assumption is that the system that this program is going to be run on is using the Windows Operating System – Windows 7 and above. This is required as the application runs C# and the .NET Framework.

1.4.3 Exclusions

There a few exclusions within this project - this ATM Kiosk will not feature a card reader or allow the ability to withdraw or deposit cash. The system will only allow the customer to transfer funds from one account to another on the screen – it does not deal with any physical cash, and all functionality is done through the system itself. In the future this is an area that Modern Bank could look to upgrade and include within the system as a separate project.

Other exclusions are the lack of multi-language support. The system only allows for the English language. As the company is based in Ireland, this shouldn’t be an issue, but in the future as the company expands, this could be an area that the client looks to expand, perhaps on future projects.

1.5 Future Contingencies

There are contingencies in the future that may impact the project and design of the system that can also have an impact on the direction of the development.

1.5.1 General Data Protection Regulation (GDPR)

At the moment, ***General Data Protection Regulation (GDPR)*** is a hot topic as laws regarding GDPR compliance are constantly evolving. It is important to stay on top of it, and to be aware of the changes as it can have massive affects to how a system can store its data. GDPR can cause the direction of the development of the system to change if laws are modified or updated. On May 25 2018, the European Union imposed obligations and regulations regarding the capturing and targeting of data to anyone living within the EU. This had a massive impact on systems and companies not only in the European Union, but all around the world – as visitors from the EU would need their data captured differently, in accordance with the new regulations – as compared to visitors outside of the EU.

This was a very strict law that was created with the aim of tighter online privacy and security for the people living in this union. There are harsh penalties and fines for those who violate the privacy and security standards – so it is something that a company must be mindful of, and then integrate within a system. It is equally as important to keep up to date in regards to the current GDPR laws, and to implement these changes within a system that is live.

Essentially, it does have an impact in the development of a system, as the system cannot go live unless it is GDPR compliant, or the company will face fines.

1.5.2 Data Encryption

Another potential aspect that could distract the flow of software development is implementing data encryption within the system. As the Modern Bank ATM Kiosk will deal with sensitive and confidential information from customers, it is vital to keep this information secure. This can be aided by the use of Data Encryption. The last thing any company would want is for a data leak, so it is essential to implement this encryption correctly.

Data Encryption would translate and encrypt data into code that only people with restricted access with the use of a secret key known as a decryption key, or password can use in order to read this information. It is recommended that Modern Bank integrates encryption as a data security method in order to secure the data of their passengers.

1.5.3 Additional Users

In the future the system could look to allow new customers to create accounts using the system. At the moment, only current customers with the bank can login to the system, but this is something that could be looked at as further functionality within the future.

2.0 System Overview

The project requirements involve the creation of a C#, .NET Framework application using an Object-Oriented Event-Driven programming approach that is to be used as an ATM Kiosk for Modern Bank.

The requirements and methods that Modern Bank would like to be featured within the scope of this application includes:

1. Login System – Using the Account Number and Account Pin to act as a username and password to the system.
2. View Current Balance – Allowing the customer to see their current balance on screen
3. View and Edit Personal Details – Allow the customer to see their current details, and make amendments where required.
4. View Banking Details – Allowing the customer to view private banking details, such as Account Number, Account PIN and Balance.
5. Transfer – Ability to transfer money from one account to another
6. Transaction History – Allow the user to view their transaction history, display their in/out funds with a description
7. Timer – A timer that will log the user out after a certain amount of time
8. Events – At least 3 different event types to be implemented within the program,
   e.g. Timer, onMouseOver, onHover, onCheck, etc.
9. Database to store customer's account number, account pin, balance, address, name, etc. Multiple tables within the database.
10. ATM Connector Class – A class for all SQL query's within the system

In order to achieve this, we have designed a context diagram for the system. In this diagram, we have identified the requirements that we feel are needed to be integrated within the system. This includes the creation of tables within a database, which can be seen in the context diagram below.

2.1 Modern Bank ATM Kiosk Context Level Diagram

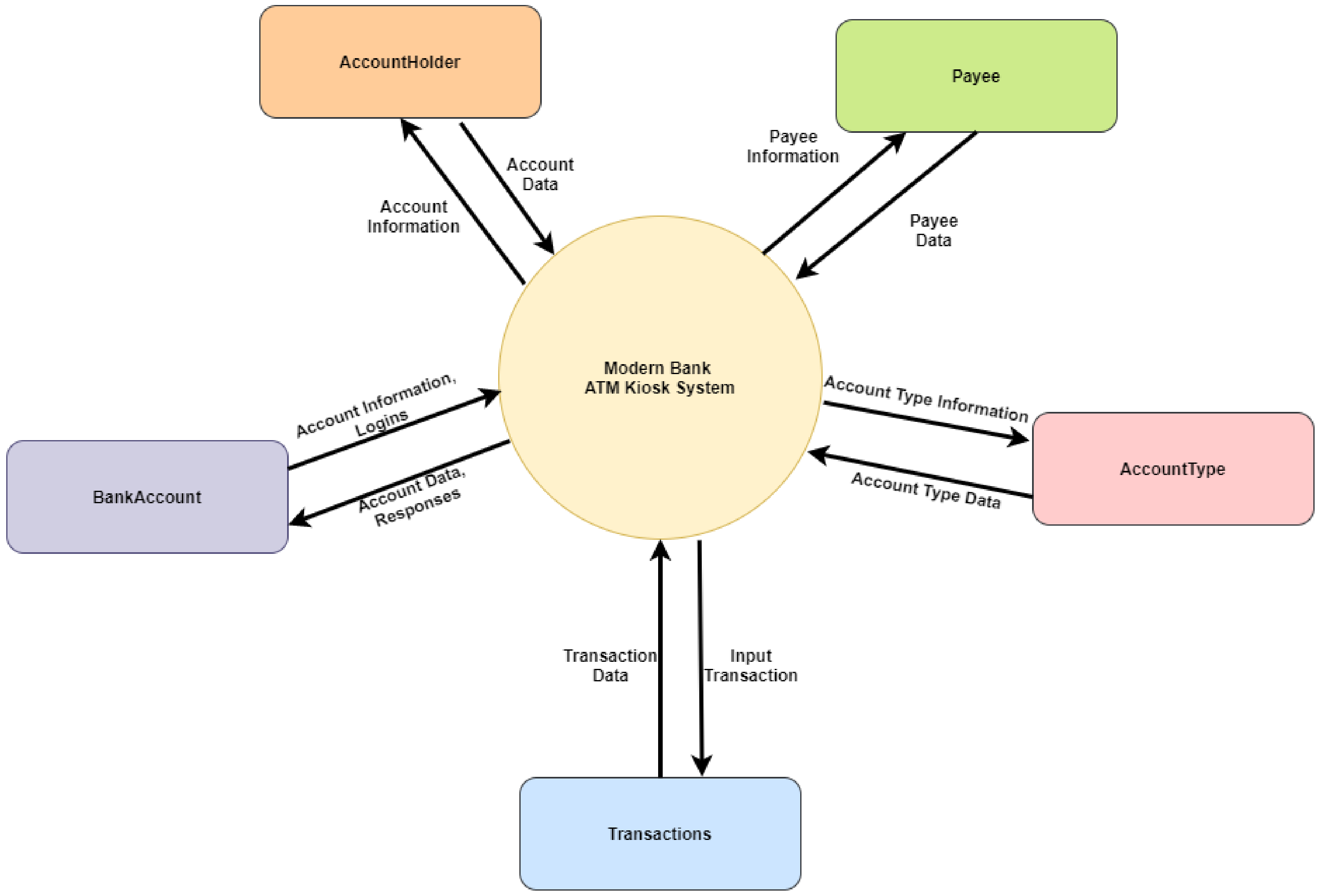


Figure 1: Context Level Diagram

2.2 Use Case Diagram

The following use case diagram depicts a representation of the Modern Bank ATM Kiosk System. It graphically displays the relationship between the Bank Account Holder (Customer) and the ATM kiosk system. The only user of this system is the end-user, a customer of Modern Bank.

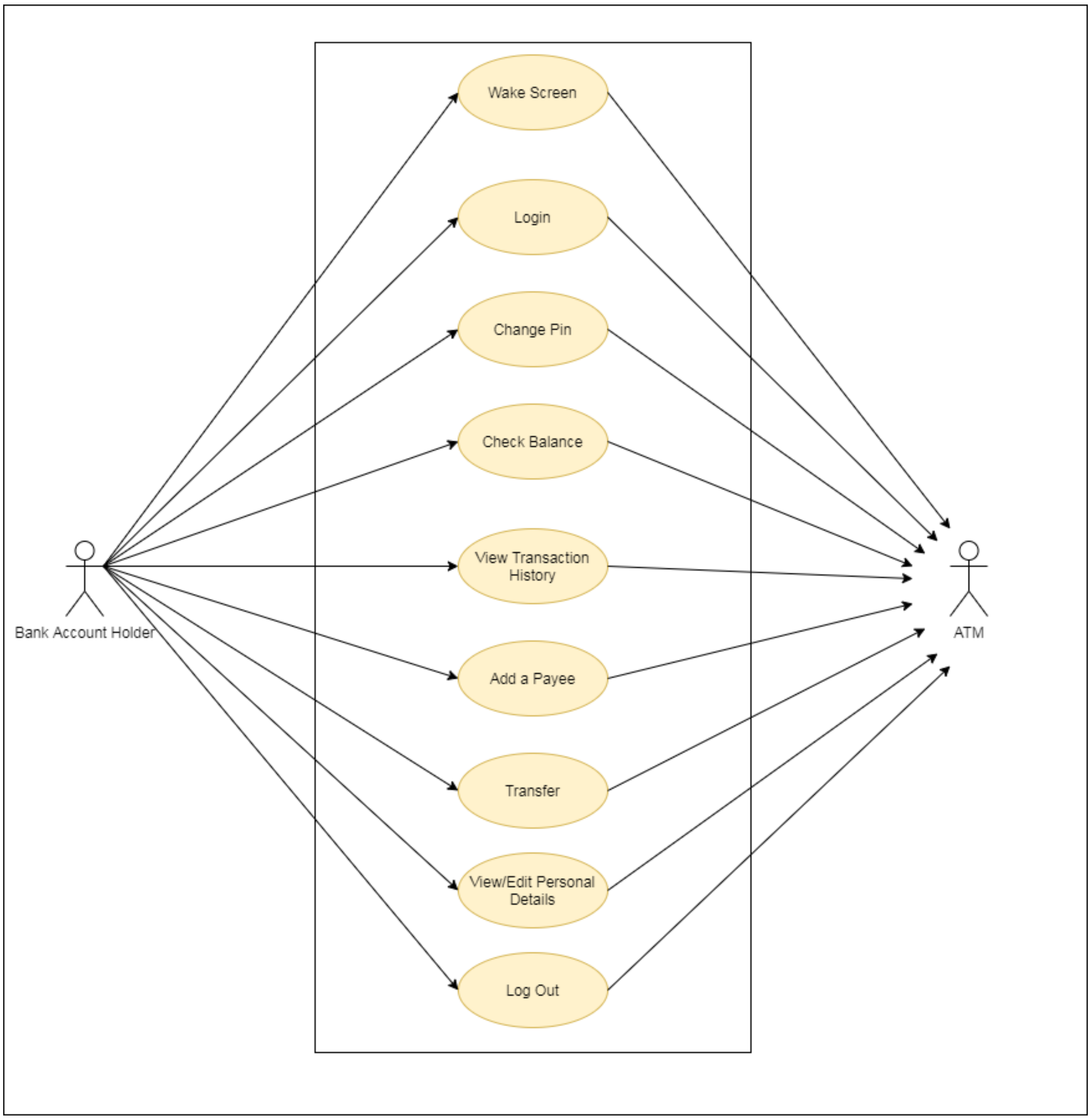


Figure 2:Use Case Diagram

Next we will go through the data flow diagrams that represent the functions of the system.

2.3 Login Level 0 DFD

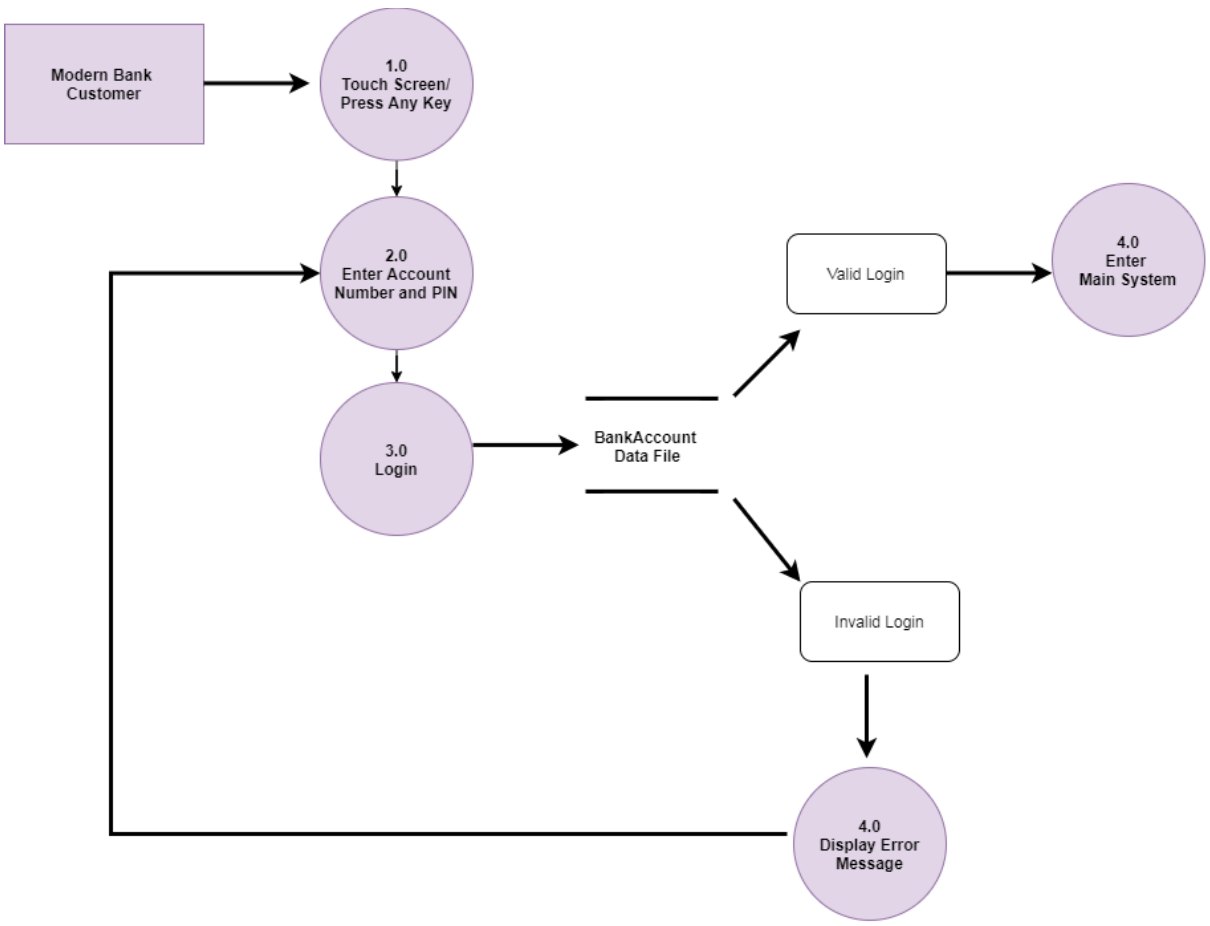


Figure 3: Login DFD 0

2.4 Main Menu Level 0 DFD

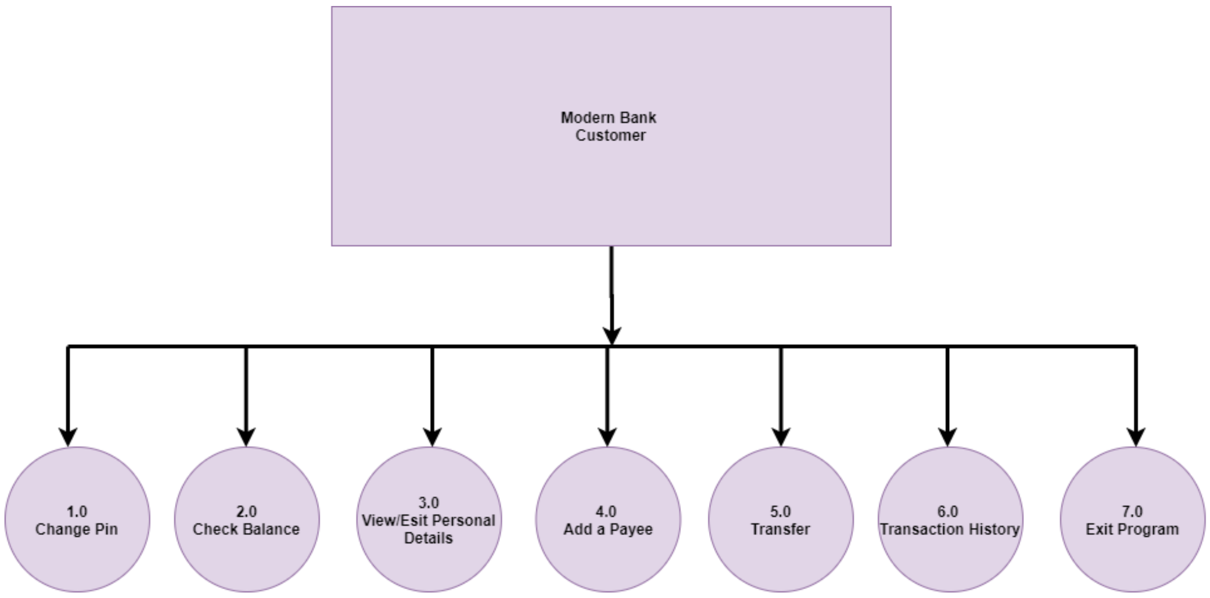


Figure 4: Main Menu DFD 0

2.5 Change Pin Level 0 DFD

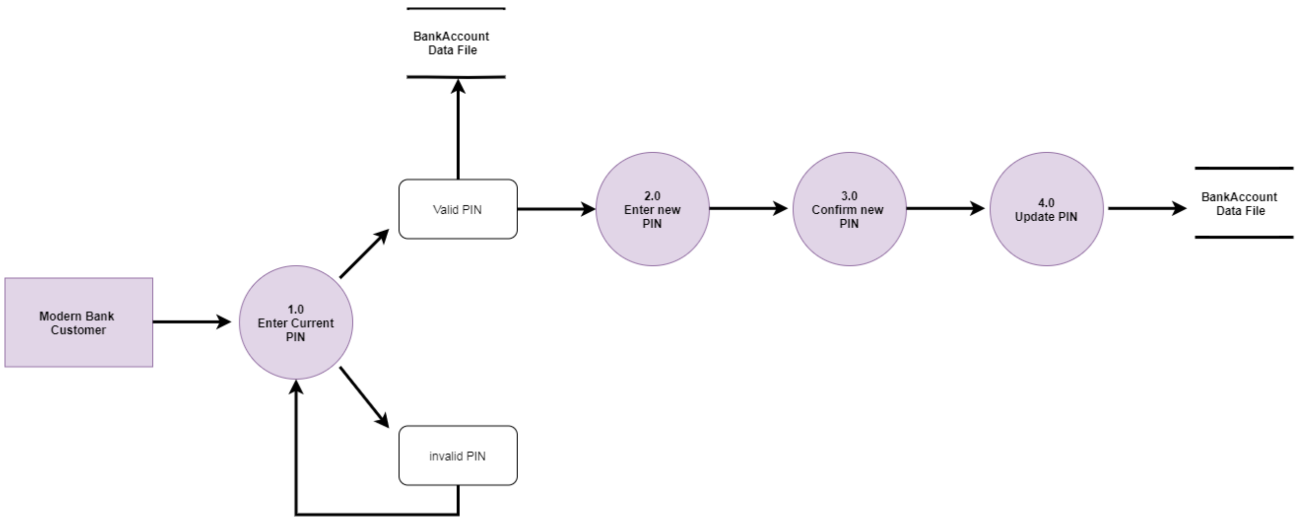


Figure 5: Change Pin DFD 0

2.6 Check Balance Level 0 DFD

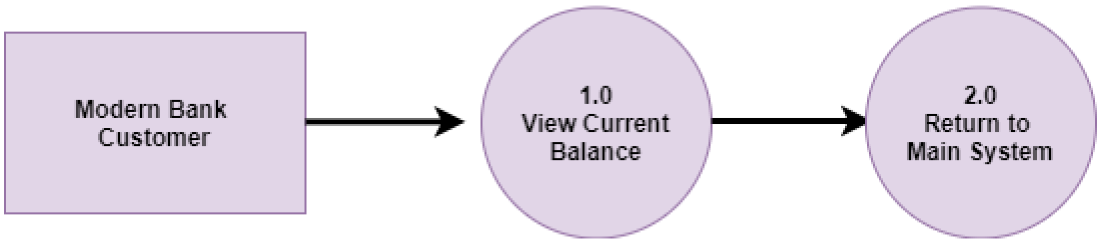


Figure 6:Check Balance DFD 0

2.7 Edit Personal Details/View Banking Details Level 0 DFD

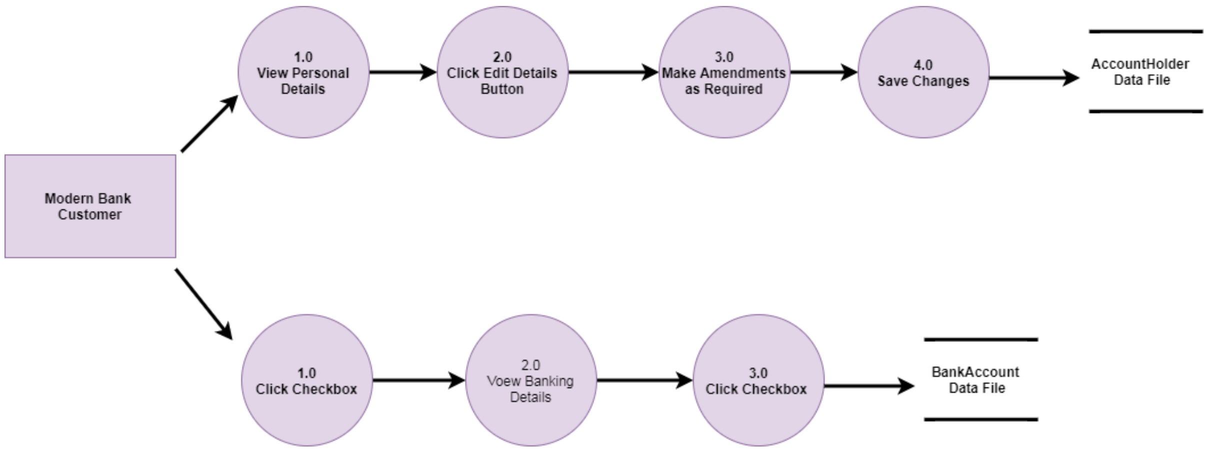


Figure 7: View/Edit Details DFD 0

2.8 Transfer Level 0 DFD

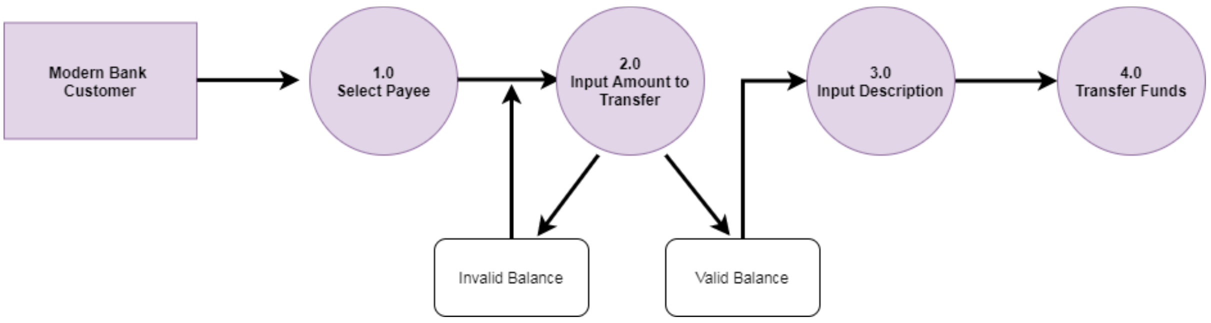


Figure 8: Transfers DFD 0

2.8 Transaction History Level 0 DFD

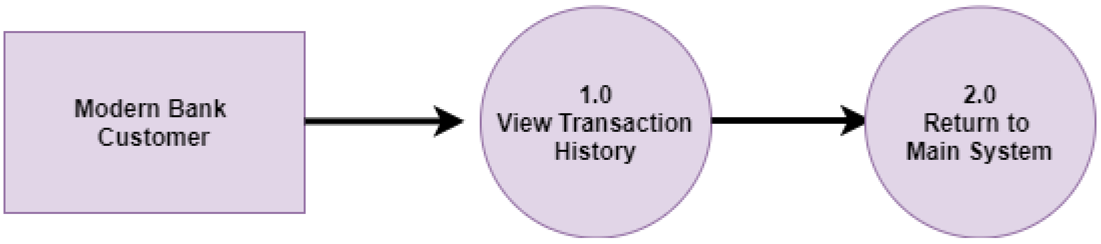


Figure 9: Transaction History DFD 0

2.9 Sequence Diagram

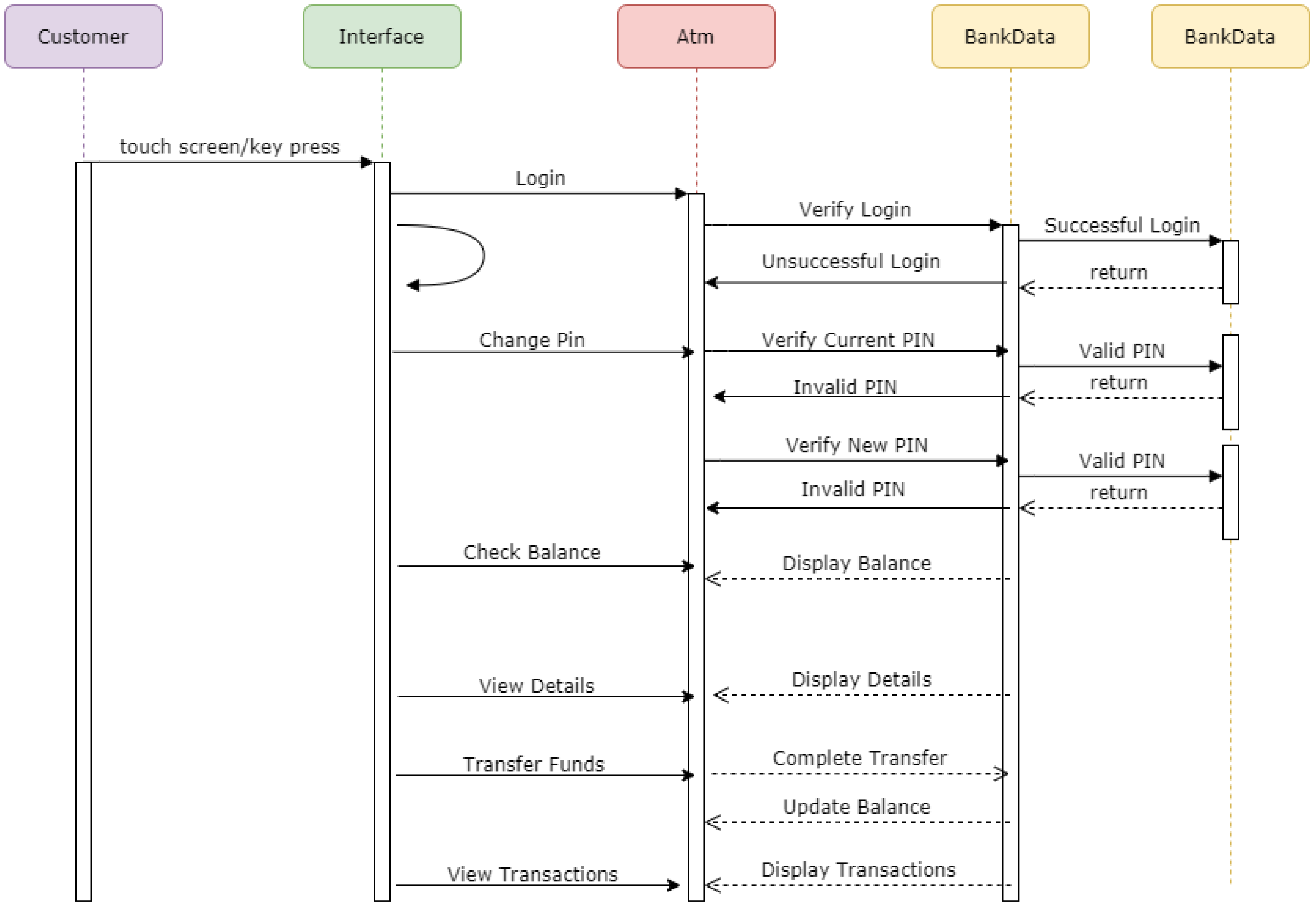


Figure 10: Sequence Diagram

3.0 System Requirements

Project Description

The project requirements involve an Object-Oriented Event-Driven GUI program that is to be an ATM Style Kiosk application. The requirements and methods that Modern Bank would like to be featured within the scope of this application includes:

1. Login System – Using the Account Number and Account Pin to act as a username and password to the system.
2. View Current Balance – Allowing the customer to see their current balance on screen
3. View and Edit Personal Details – Allow the customer to see their current details, and make amendments where required.
4. View Banking Details – Allowing the customer to view private banking details, such as Account Number, Account PIN and Balance.
5. Transfer – Ability to transfer money from one account to another
6. Transaction History – Allow the user to view their transaction history, display their in/out funds with a description
7. Timer – A timer that will log the user out after a certain amount of time
8. Events – At least 3 different event types to be implemented within the program,
   e.g. Timer, onMouseOver, onHover, onCheck, etc.
9. Database to store customer's account number, account pin, balance, address, name, etc. Multiple tables within the database.
10. ATM Connector Class – A class for all SQL query's within the system

4.0 System Architecture

In order for the customers of Modern Bank to be able to use this ATM Kiosk system, it is important that they are presented with all of the required hardware and software requirements.

4.1 System Hardware Requirements

The <u>Minimum Hardware Requirements</u> to run the ***Modern Bank ATM Kiosk*** includes:

Keypad/Keyboard
Kiosk - Computer **Touchscreen Display Monitor**

1. ***Keyboard*** – A keyboard is required for this system. The program features an onscreen keyboard for the login and numeric inputs, however, for the input of text, a physical keyboard will be required.

2. ***Computer*** – A computer is required in order to run this system. Not only is it required to meet the hardware requirements, but also the software requirements. We will speak further about the *minimum system requirements* in the subsequent section to gain further insight.

3. ***Display Monitor*** – In order for the user to use the application, it is necessary to have a display monitor connected to the computer. Without the display monitor, the user cannot use the application as they cannot see or run the system and it's graphical user interface. It is important to note that this display monitor must be touchscreen in order to allow the customer to use the system as intended – as input is required with the use of an onscreen keyboard.

4.2 System Software Requirements

This program will be created using the C# Programming Language. The application will be developed using Microsoft Visual Studio Community. The application will be using .NET framework to develop the system. This booking system will feature the use SQLite for its database, and this will be run using the DB Browser application. The ***Modern Bank ATM Kiosk System*** can only be run on PC's using Windows. It is not available on Mac, mobile phones or tablets. As the system will be developed using Windows 7, the minimum requirements in order to run this system can be seen below.

The Minimum Software Requirements to run the ***Modern Bank ATM Kiosk System*** includes:

1. .NET Framework
2. Windows 7 Operating System or Later
3. 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor*
4. 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit)
5. 16 GB available hard disk space (32-bit) or 20 GB (64-bit)
6. DirectX 9 graphics device with WDDM 1.0 or higher driver

This is the ideal system specifications that are required to run this program correctly.

If your system fails to meet the system requirements, there may be performance issues, or the program may be incompatible.

4.1 System Creation Tools

Programming Language	C#
Framework	.NET Framework
Software Tools	1. **Visual Studio Community** – used for the creation of the Booking System program and the Graphical User Interface 2. **DB Browser** – used to populate and view database tables. Management of the database used in the system
Database	**SQLite** – Entry Level Free database software
Packages Used	**NuGet**– SQLite Packages and Entity Framework installed on Visual Studio required to connect Database to the System

4.2 Packages and Program Installations

In order to create this program, it is essential to follow the technical documentation and specifications to ensure it runs and works as expected.

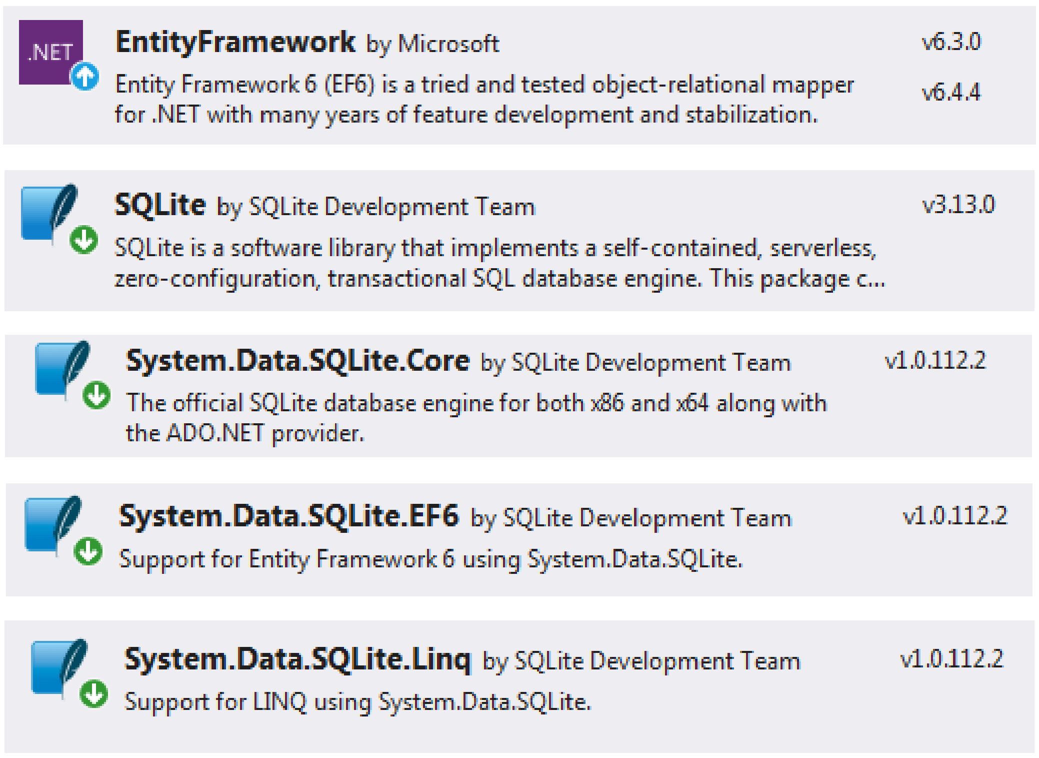
1. Download Visual Studio Community to create C# Form Applications

https://visualstudio.microsoft.com/vs/

2. Download DB Browser to browse the database that we will be using

https://sqlitebrowser.org/dl/

3. Download the following NuGet Packages on Visual Studio:



EntityFramework by Microsoft v6.3.0 v6.4.4
Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years of feature development and stabilization.

SQLite by SQLite Development Team v3.13.0
SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. This package c...

System.Data.SQLite.Core by SQLite Development Team v1.0.112.2
The official SQLite database engine for both x86 and x64 along with the ADO.NET provider.

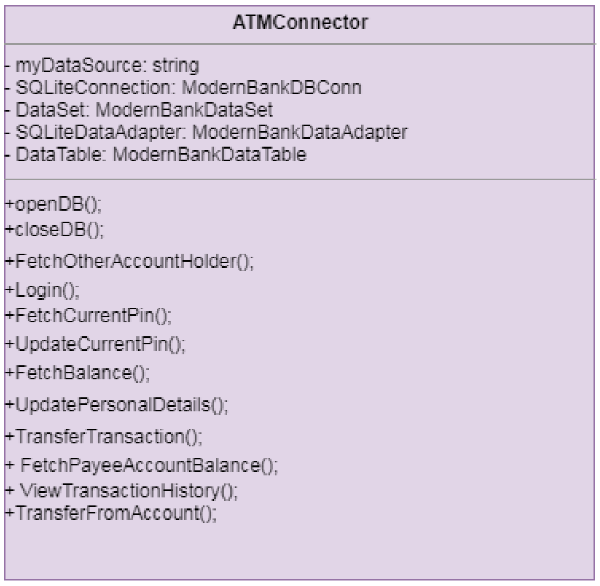
System.Data.SQLite.EF6 by SQLite Development Team v1.0.112.2
Support for Entity Framework 6 using System.Data.SQLite.

System.Data.SQLite.Linq by SQLite Development Team v1.0.112.2
Support for LINQ using System.Data.SQLite.

4. Download the BankData.db database file and create a folder in `C:\\Data\\BankData.db.`

5. Create a Database Connection using SQL in C#

In order to access data from an external database, BankData.db, we will need to establish a connection. We will create this as it's own class, titled, ***ATMConnector***. We will store all of the functions relating to the connection between the database and the system here.


ATMConnector
- myDataSource: string
- SQLiteConnection: ModernBankDBConn
- DataSet: ModernBankDataSet
- SQLiteDataAdapter: ModernBankDataAdapter
- DataTable: ModernBankDataTable
+openDB();
+closeDB();
+FetchOtherAccountHolder();
+Login();
+FetchCurrentPin();
+UpdateCurrentPin();
+FetchBalance();
+UpdatePersonalDetails();
+TransferTransaction();
+ FetchPayeeAccountBalance();
+ ViewTransactionHistory();
+TransferFromAccount();

The connection will manage a connection to the database.

The command will execute a query command on the database.

The data adapter will exchange data between the data set and the database.

5.0 Design

5.1 Class Diagram

This diagram is a visual representation of the classes that are to be used within the Modern Bank ATM Kiosk System.

5.1.1 Class Diagram

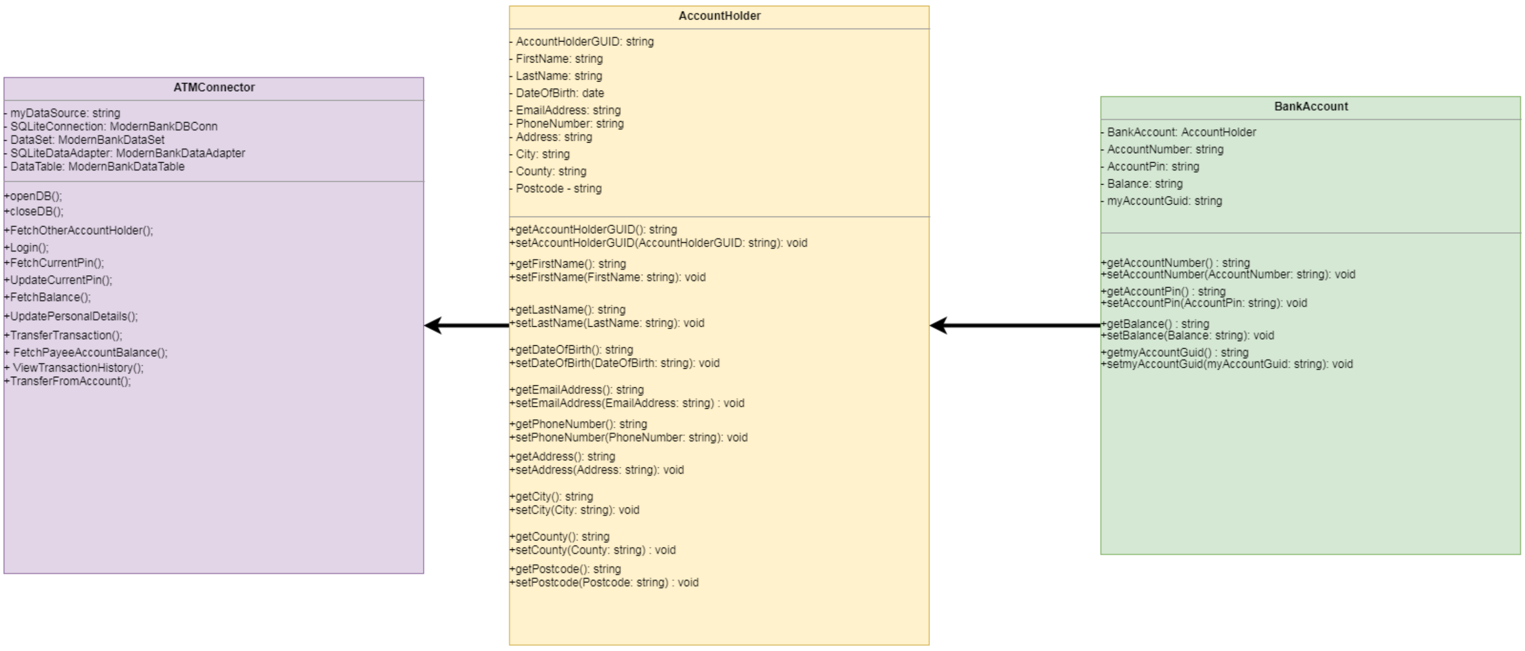

ATMConnector
- myDataSource: string
- SQLiteConnection: ModernBankDBConn
- DataSet: ModernBankDataSet
- SQLiteDataAdapter: ModernBankDataAdapter
- DataTable: ModernBankDataTable
+openDB();
+closeDB();
+FetchOtherAccountHolder();
+Login();
+FetchCurrentPin();
+UpdateCurrentPin();
+FetchBalance();
+UpdatePersonalDetails();
+TransferTransaction();
+ FetchPayeeAccountBalance();
+ ViewTransactionHistory();
+TransferFromAccount();
AccountHolder
- AccountHolderGUID: string
- FirstName: string
- LastName: string
- DateOfBirth: date
- EmailAddress: string
- PhoneNumber: string
- Address: string
- City: string
- County: string
- Postcode - string
+getAccountHolderGUID(): string
+setAccountHolderGUID(AccountHolderGUID: string): void
+getFirstName(): string
+setFirstName(FirstName: string): void
+getLastName(): string
+setLastName(LastName: string): void
+getDateOfBirth(): string
+setDateOfBirth(DateOfBirth: string): void
+getEmailAddress(): string
+setEmailAddress(EmailAddress: string) : void
+getPhoneNumber(): string
+setPhoneNumber(PhoneNumber: string): void
+getAddress(): string
+setAddress(Address: string): void
+getCity(): string
+setCity(City: string): void
+getCounty(): string
+setCounty(County: string) : void
+getPostcode(): string
+setPostcode(Postcode: string) : void
BankAccount
- BankAccount: AccountHolder
- AccountNumber: string
- AccountPin: string
- Balance: string
- myAccountGuid: string
+getAccountNumber() : string
+setAccountNumber(AccountNumber: string): void
+getAccountPin() : string
+setAccountPin(AccountPin: string): void
+getBalance() : string
+setBalance(Balance: string): void
+getmyAccountGuid() : string
+setmyAccountGuid(myAccountGuid: string): void

Figure 11: Class Diagram

5.1.2 ATM Connector Class Diagram

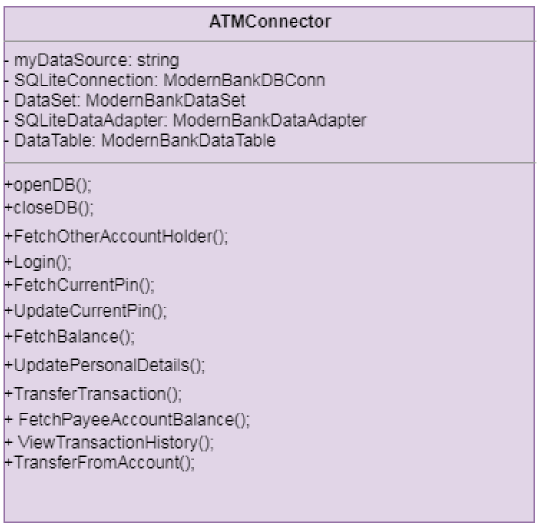

ATMConnector
- myDataSource: string
- SQLiteConnection: ModernBankDBConn
- DataSet: ModernBankDataSet
- SQLiteDataAdapter: ModernBankDataAdapter
- DataTable: ModernBankDataTable
+openDB();
+closeDB();
+FetchOtherAccountHolder();
+Login();
+FetchCurrentPin();
+UpdateCurrentPin();
+FetchBalance();
+UpdatePersonalDetails();
+TransferTransaction();
+ FetchPayeeAccountBalance();
+ ViewTransactionHistory();
+TransferFromAccount();

Figure 12: ATM Connector Class Diagram

5.2 Database Design

As previously mentioned in this document, the management system used for the “BankData.db” database is DB Browser. We have found through research that this is the most ideal program to browse and create our data on, as it is easy to use and user-friendly. The main positive of this management system is that the software is free, compared to other programs that exceed our budget. In the future, we would like to upgrade this management system if possible, to an enterprise system that can handle larger data.

DB Browser for SQLite can be downloaded at: https://sqlitebrowser.org/

We are using SQLite for the connection between the database and the ATM system, with the installation of additional NuGet packages on Visual Studio.

5.2.1 Database Management System Files

As we have created a database to store our data, we need to define a data dictionary for each of the tables within the database. The use of the data dictionary is to provide a more detailed account about each of the contents within the database. This includes the names of the fields, the type of data it represents, the size of this data, the format, an example of this data and a description of the data.

There will be five tables used within this database at present, which include:

1. **AccountHolder** – this table will store the data associated with the account holder. This includes the personal details of the customers, such as their name, address, contact information and connecting guids to other tables within the database.

2. **AccountType** – this table will store the different account types that are available within the system. These account types will link to the bank account.

3. **BankAccount** – this table stores the bank account details related to a customer. It connects both the account holder and account type to store and link information. It will store the account holder’s account number, pin and balance.

4. **Payee** – this table stores the guids associated between the customer’s bank account and the payee’s account.

5. **Transactions** – this table stores information relating to transactions. It will show information relating to payments in and out of the system in terms of credit and debit. It will also store the date of the transaction and description of the transactions.

Modern Bank ATM Kiosk Data Dictionary

Table: AccountHolder – Data Dictionary

Field Name	Field type	Field Size	Format	Example	Description
Uid	Bigint	20		1 …	This is the unique ID for the AccountHolder table
Guid	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is the general ID for the AccountHolder table
FirstName	Varchar	255		Thomas	This is the first name of the AccountHolder
LastName	Varchar	255		Swift	This is the last name of the AccountHolder
DOB	Date	12	99/99/9999	05/10/1995	This is the Date of Birth of the AccountHolder
EmailAddress	Varchar	255	xxx@xxxx.ie	modernbanking@info.com	This is the email address of the AccountHolder

PhoneNumber	**Varchar**	**255**		**+353 85 / +22 /**	**This is the phone number of the AccountHolder**
Address	**Varchar**	**255**		**20 Woodvale Avenue, Ballyknock Road**	**This is the address of the AccountHolder**
City	**Varchar**	**255**		**Knocklyon**	**This is the city of the AccountHolder**
County	**Varchar**	**255**		**Dublin**	**This is the county of the AccountHolder**
Postcode	**Varchar**	**255**	**XXXXXXX**	**D16Y688**	**This is the postcode of the AccountHolder**
Deleted	**Tinyint**	**1**		**Default 0**	**This is to delete an AccountHolder**

Table: BankAccount – Data Dictionary

Field Name	Field type	Field Size	Format	Example	Description
Uid	Bigint	20		1 …	This is the unique ID for the BankAccount
Guid	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is the general ID for the BankAccount
AccountNumber	Varchar	10	XXXXXXXXXX	0123456789	This is the AccountNumber for the BankAccount
AccountPin	Varchar	4	XXXX	1234	This is the AccountPin for the BankAccount
AccountHolderGUID	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is the general ID for the AccountHolderGUID
Balance	Numeric	10,2		20.00	This is the account balance
AccountTypeGUID	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is the general ID for the AccountTypeGUID
Deleted	Tinyint	1		Default 0	This is to delete an Account

Table: AccountType – Data Dictionary

Uid	**Bigint**	**20**		**1 …**	**This is the unique ID for the AccountType**
Guid	**Varchar**	**255**		**1234abc-5678-de9-0fgh-123ijk45678lm9**	**This is the general ID for the AccountType**
Description	**Varchar**	**255**		**Thomas Swift**	**This is the description/name of the type of account**
Deleted	**Tinyint**	**1**		**Default 0A**	**This is to delete an Account Type**

Table: Transactions – Data Dictionary

Field Name	Field type	Field Size	Format	Example	Description
Uid	Bigint	20		1 …	This is the unique ID for the Transactions
Guid	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is the general ID for the Transactions
BankAccountGUID	Varchar	255		Dublin Airport	This is the BankAccountGUID
Debit	Varchar	255		London Stansted	This is to be used as an indicator of money going OUT
Credit	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is to be used as an indicator of money coming IN
DateOfTransaction	Numeric	2,2	00.00	10.15	This is the date the transaction occurred.
PayeeAccountGUID	Numeric	2.1		5.5	This is the Payee's Account GUID
Description	Varchar	6	CY0000	CY1234	This provides information in relation to the transaction

Table: Payee – Data Dictionary

Field Name	Field type	Field Size	Format	Example	Description
Uid	Bigint	20		1 …	This is the unique ID for the Payee
Guid	Varchar	255		1234abc-5678-de9-0fgh-123ijk45678lm9	This is the general ID for the Payee
BankAccountGUID	Varchar	10	EI0000	EI1234	This is the BankAccountGuid
PayeeAccountGUID	Varchar	255		Boeing 747 Airbus 380	This is the PayeeAccountGUID
Deleted	Tinyint	1		Default 0	This is to delete a Payee

6.0 Human-Machine Interface

In regards to human-machine interface, we have produced prototype models of what we envision the ***Modern Bank ATM Kiosk system*** will look like. The goal we have for the graphical user interface is a simple white form, with the Modern Bank logo, with a purple/lavender theme to go through the ATM that is easy to read and matches the logo of the bank. Our aim was to keep the program simple and functional, while remaining user-friendly.

Note: The following images are only prototypes of the program, and may *differ to the final end product.*

6.1 Inputs

Welcome Screen

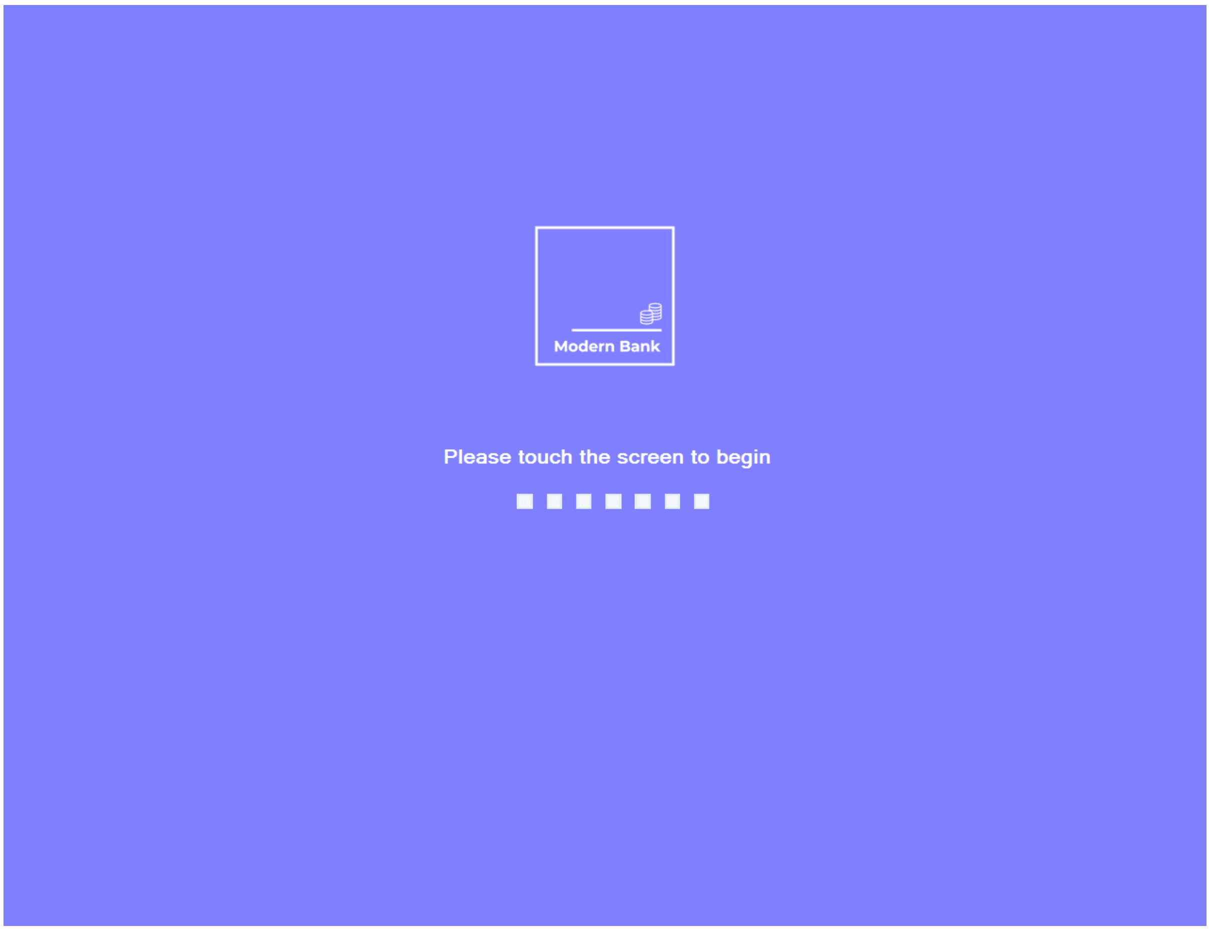


Figure 13: Welcome Screen

This is the first screen that the customer will see when approaching the system. It is used as a screensaver, prompting a click from the customer to begin the program and load the login screen to enter the system.

The design of this page is simple, it shows the Modern Bank logo and instructs the end-user what to do in order to enter the system. This page also features a loader gif.

Login Screen

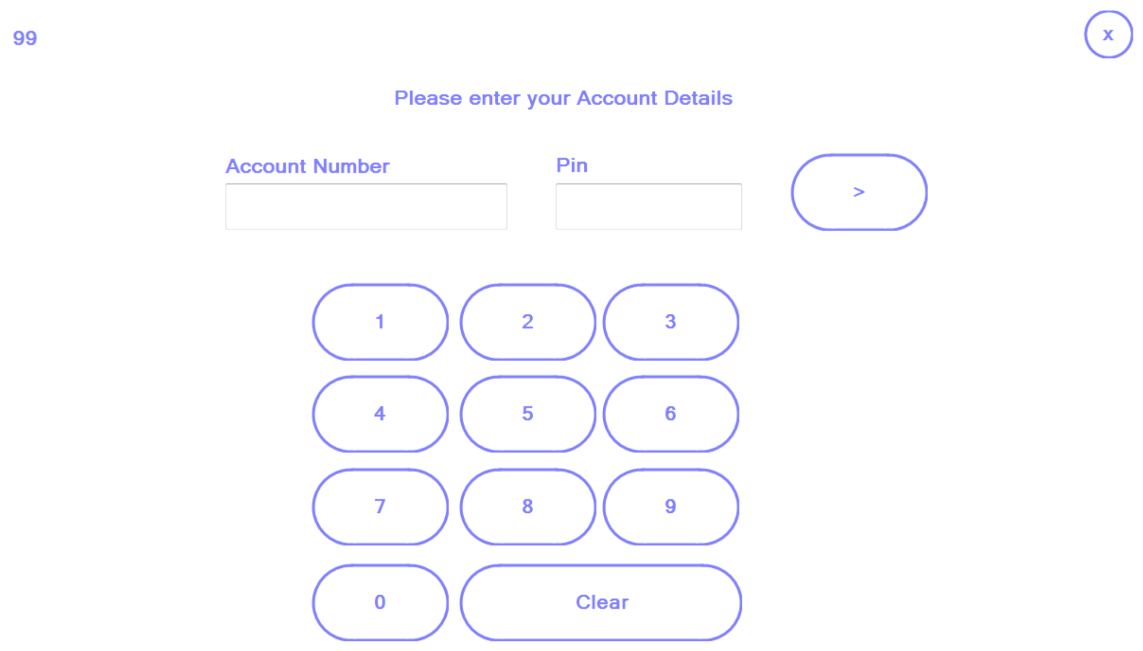


Figure 14: Login Screen

This screen will allow the customer to login using their Account Number and PIN. These login details would have already been created prior to the arrival at the ATM kiosk.

The login will only allow for numeric characters, and the customer must use the keyboard on screen.

There will be error-handling to ensure that the login is valid, and to ensure that the customer cannot access the system without valid credentials and all information completed.

The system will also feature a timer that will be present on the screen.

Main Screen

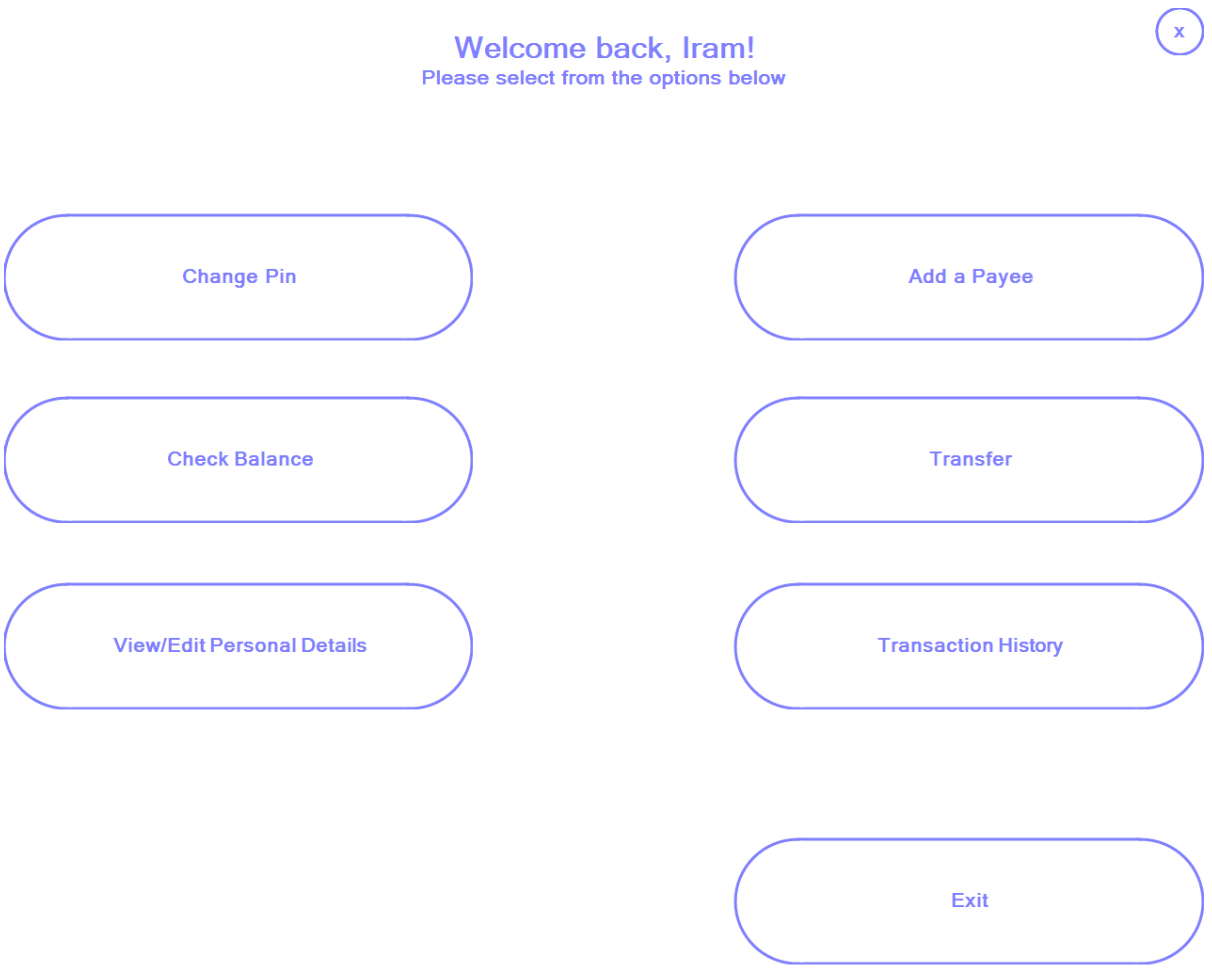


Figure 15: Main Screen

This screen will display the many options and functionality within the program. The customer will be able to select from the many options available on the screen.

All of the forms will also be redirected to this main page with a back button present on each page.

This form will display a welcome message with the first name of the customer pulled from the database based on the account login.

Change Pin

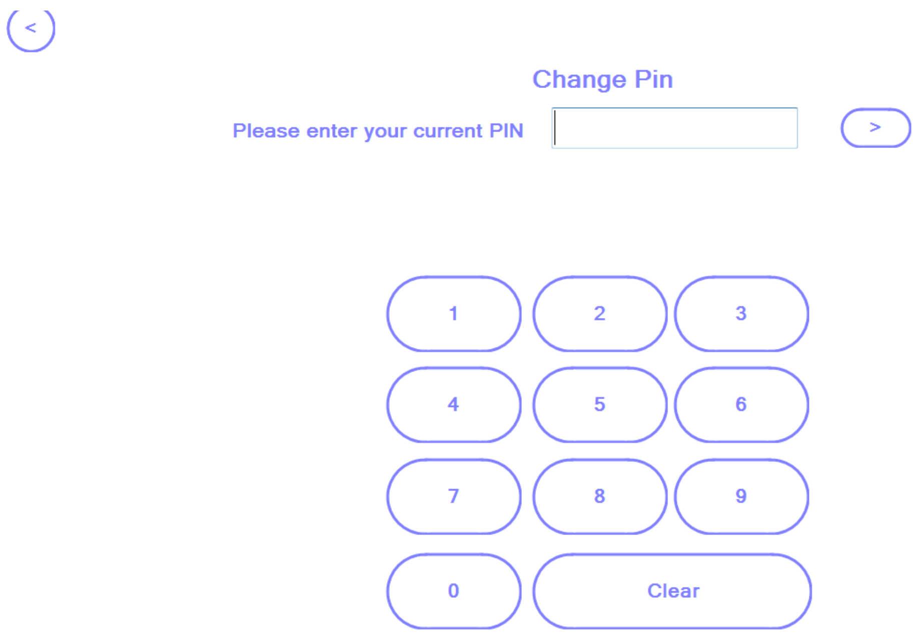


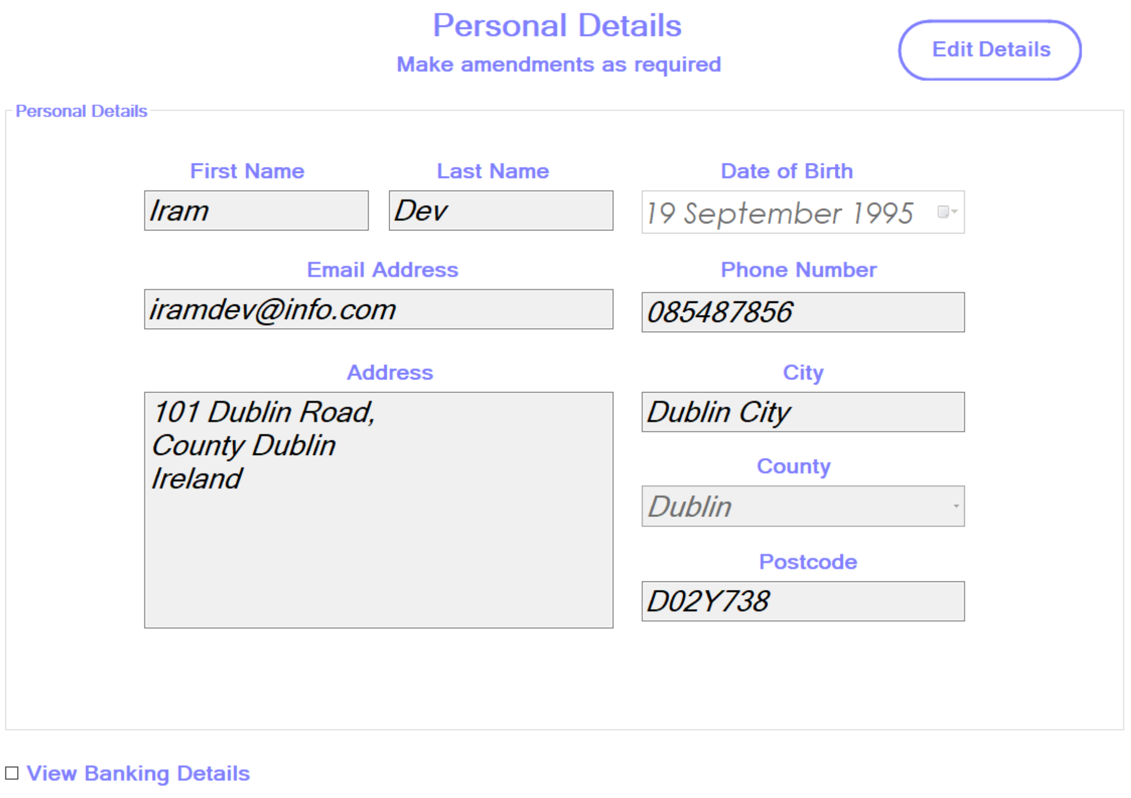
Figure 16: Change Pin

This screen within the program relies on the input from the user.

It requires the end-user to input their current PIN. Once this has been confirmed, the user will then be able to create a new PIN. This PIN will need to be inputted twice, and both of the PIN's inputted must match and be the exact same in order to change the PIN. Once this has been confirmed, the user can save the changes and a message will appear on the green to inform the user of the changes.

Once these changes have been confirmed, the program will automatically redirect the user back to the main screen.

View/Edit Personal Details



First Name: Iram	Last Name: Dev	Date of Birth: 19 September 1995
Email Address: iramdev@info.com		Phone Number: 085487856
Address: 101 Dublin Road, County Dublin Ireland		City: Dublin City
		County: Dublin
		Postcode: D02Y738

☐ View Banking Details

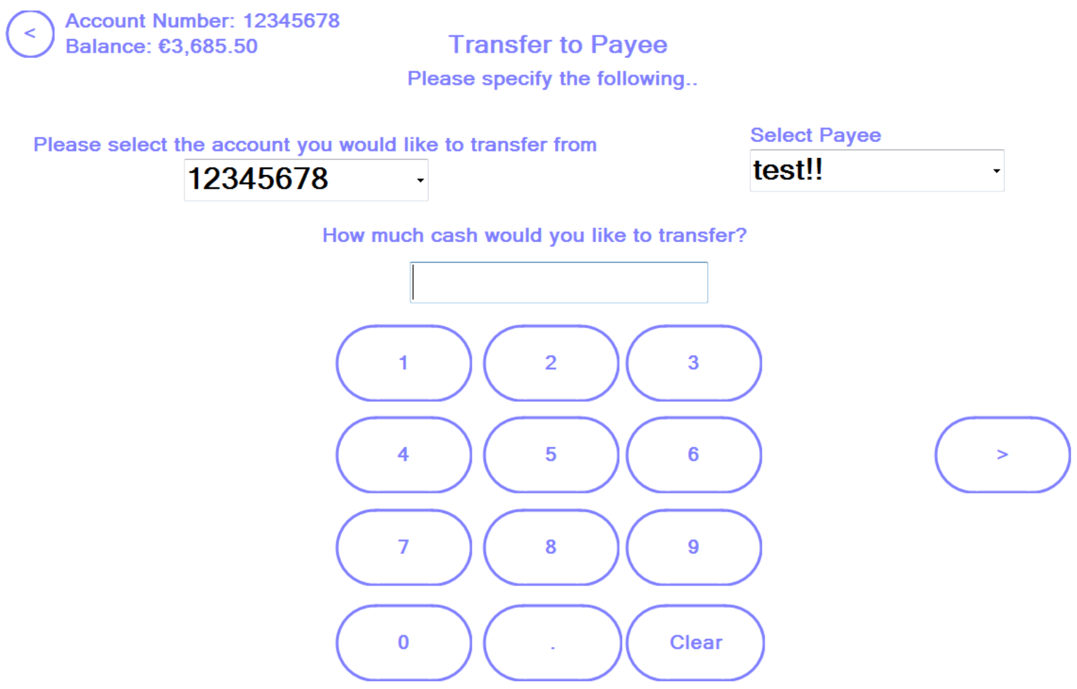
Figure 17: View/Edit Personal Details

This form serves as both an input and output to the system.

The system will load the customer's personal details when viewing this page which serves as an output of the system. This information will be read-only. The user will be able to click a button to edit their details, which will allow input from the user to make changes. A message will appear on screen when the user saves their changes.

This form will also display banking details such as the account number, pin and balance when the checkbox is checked at the discretion of the end-user.

Transfer


Account Number: 12345678
Balance: €3,685.50

Transfer to Payee

Please specify the following..

Please select the account you would like to transfer from: 12345678

Select Payee: test!!

How much cash would you like to transfer?

Figure 18: Transfer

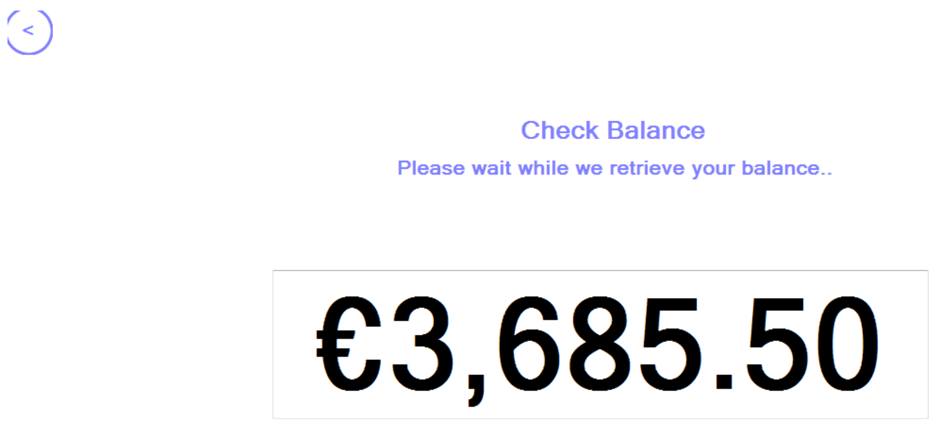
This form will allow the customer to transfer money from one account to another. The user may have multiple bank accounts which they can choose from in a combobox. They can decide to transfer payments to their other bank accounts, if applicable, otherwise, they will be able to transfer funds to any payee's that they have already added. If they do not have any payees added, they will need to return to the main screen and add a payee.

The customer will then need to choose how much they would like to transfer.

For ease of access, the form will display the customer's account number, in the case that they may have multiple accounts, to ensure that they know what account they are transferring from, and their balance so they know what funds are available in their account. The user will not be able to transfer funds that they do not have.

6.2 Outputs

Check Balance



Please wait while we retrieve your balance..

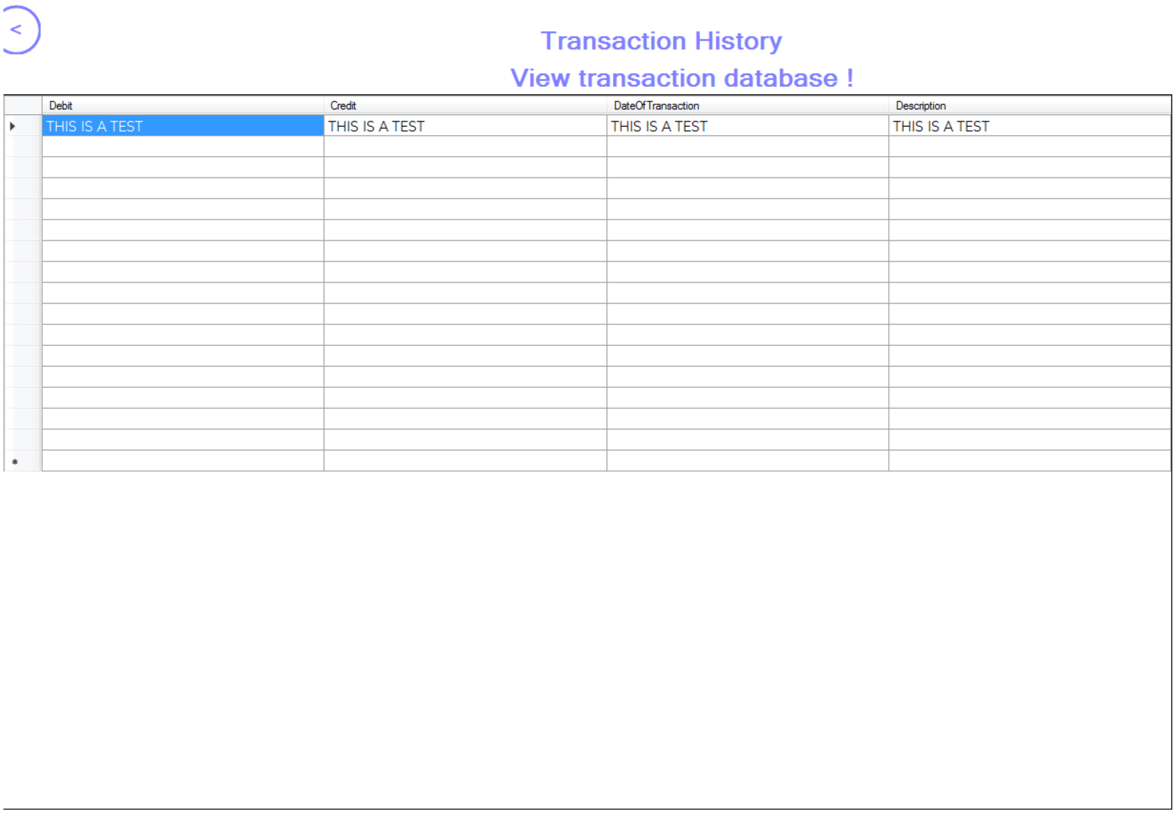
€3,685.50

Figure 19: Balance

This screen will begin with a loader, to inform the user that their balance is being retrieved.

After 3 seconds the customer’s account balance will be displayed on screen. This is an output as it is pulled from the database.

Transactions



Transaction History
View transaction database !

Debit	Credit	DateOfTransaction	Description
THIS IS A TEST	THIS IS A TEST	THIS IS A TEST	THIS IS A TEST

Figure 20: Transactions

This form when loaded will display all of the transactions based on the bank account that is logged in. This data will be loaded into a database that will allow the user to view their transactions.

The transactions will display Debit as OUT, Credit as IN (funds), a date of the transaction and a description of the transaction that has occurred.

This form is an output as it relies on gathering data stored within the database.

7.0 Design Implementation

Welcome screen

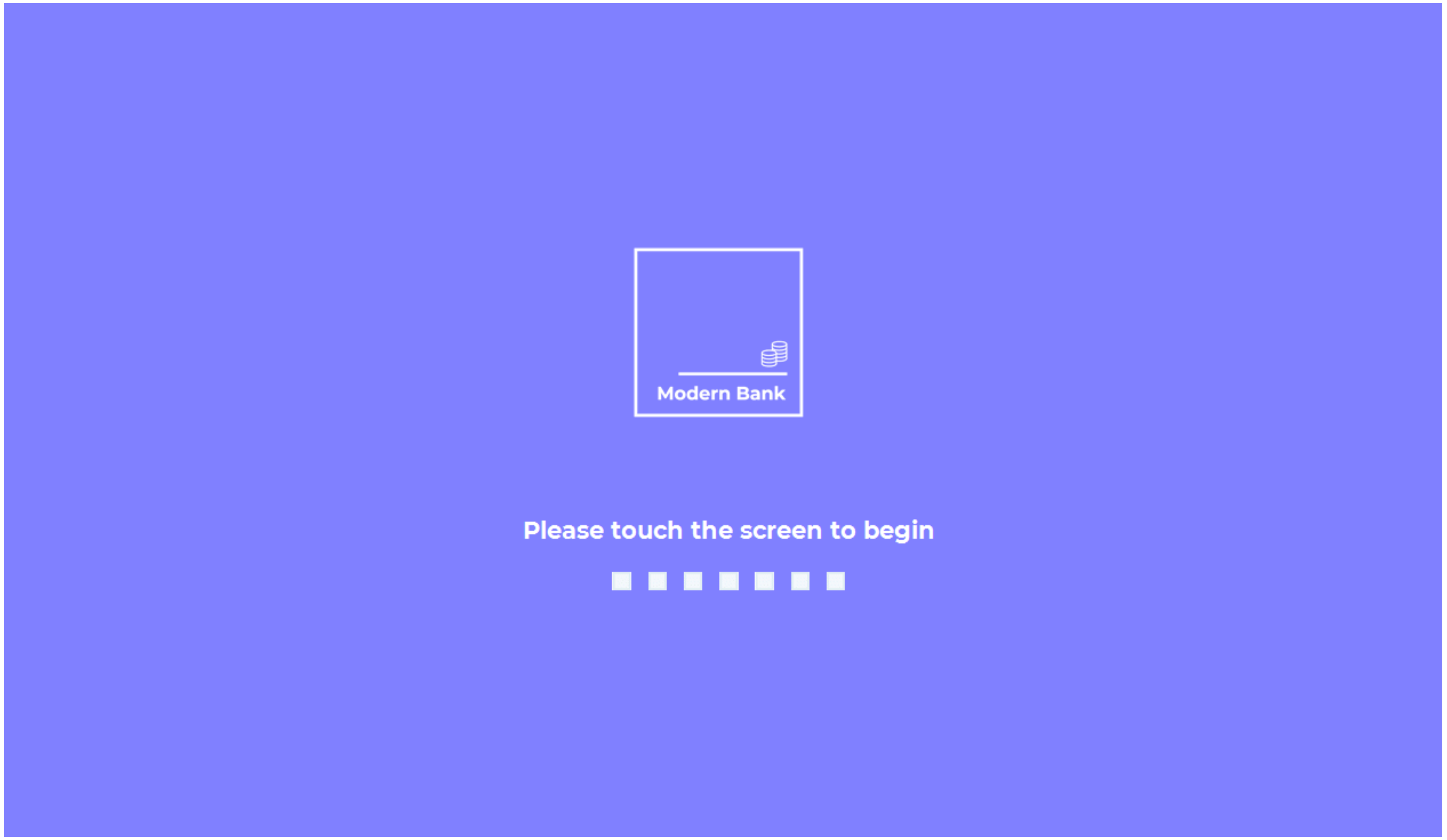


Figure 21: Welcome Screen

Process	Details	Parameter & Additional Information
Purpose	This is the first screen that the customer will see when approaching the system. It is used as a screensaver, prompting a click from the customer to begin the program and load the login screen to enter the system. The design of this page is simple, it shows the Modern Bank logo and instructs the end-user what to do in order to enter the system. This page also features a loader gif.	User must touch screen/press any key in order to continue.
Form Called From	System Load	This is the opening screen of the Modern Bank ATM Kiosk System
Form Controls	Panel x 1	This panel stores all of the components on this form
	pnlWelcome	
	Labels x 1	This label displays the text on the screen. Informs the user to touch the screen in order to wake the system
	lblMessage	
	Picturebox x 2	1) This picturebox provides a gif that acts as a pre-loader for the page. 2) This picturebox contains the Modern Bank logo
	1) pbxLoader, 2) pbxLogo	
Form Behaviour	The form informs the user to touch the screen in order to begin the program.	

Login screen

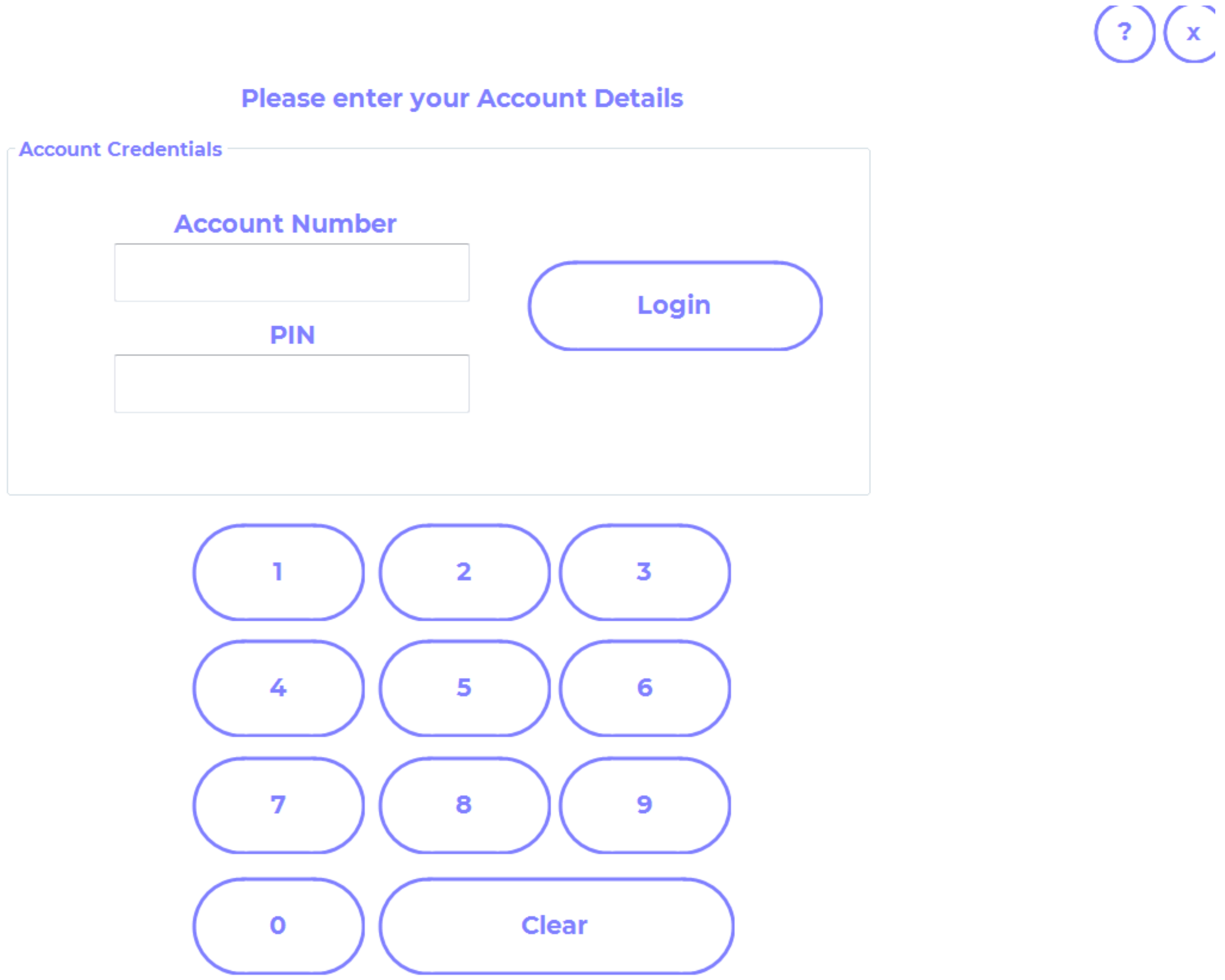


Figure 22: Login Screen

Process	Details	Parameter & Additional Information
Purpose	This screen will allow the customer to login using their Account Number and PIN. These login details would have already been created prior to the arrival at the ATM kiosk. The login will only allow for numeric characters, and the customer must use the keyboard on screen. There will be error-handling to ensure that the login is valid, and to ensure that the customer cannot access the system without valid credentials and all information completed. The system will also feature a timer that will be present on the screen.	The username and password must be valid in order for the user to enter the system.
Form Called From	frmWelcome	This form is called from the Welcome screen which acts as a screensaver until touch has been input. This form requires for the user to login in order to access the system.
Data calls on Form	Login();	"SELECT * from BankAccount LEFT JOIN AccountHolder ON AccountHolder.Guid = " + "BankAccount.AccountHolderGUID WHERE AccountNumber='" + LocalAccount + "' AND AccountPin='" + LocalAccountPin + "'LIMIT 1;";
Form Controls	Label x 3	1) Acts as a title for the Account Number textbox 2) Acts as a title for the Account Pin textbox 3) Acts as a title for the form
	1) lblAccountNo, 2) lblPin 3)lblAccountDetails	
	Panel x2	1) This panel is used to store the components relating to the onscreen keypad 2) This panel stores all of the components within the form
	1) pnlOnScreenKeypad 2) pnlLogin	

	Textbox x 2 1) txtAccountNo, 2) txtAccountPin	1) This textbox allows the user to input their account number 2) This textbox allows the user to input their account pin
	Button x14 1-10)btn0-btn9 11) btnClear 12) btnClose 13) btnLogin 14) btnHelp	1-10) These buttons are used for the onscreen keypad, for numbers 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. 11) This button clears the contents of the textboxes 12) This button exits the program 13) This button allows the user to login (Verification check) 14) This button opens a help system
	Timer x2 1) LoginTimer 2) SessionTimeout	1) This timer runs for 60 seconds. It will reset if the user interacts with any button on the page, or clicks the form 2) This session will trigger after the LoginTimer has elapsed 60 seconds. It will display a message when the time has elapsed that will tell the user their session has timed out, and they will need to login again.
	GroupBox x1 grpLogin	This groupbox is used to group the login components together visually
Form Behaviour	The form allows for a validation check against the contents inputted from the txtAccountNo and txtAccountPin to ensure that they match the details within the database. If the login credentials are correct, the user will be able to login and proceed to the next form.	If the login is valid, the user will be able to enter the program. If the login is incorrect, the user will be presented with a message on the screen to let them know their details are incorrect. If the fields are left blank and the user tries to login, a message will appear on the screen informing them that they cannot login with empty fields
Form Objects & Class	AccountHolder Class ATMConnector Class	Refer to Class Diagram ATMConnector.Login()

Main Screen

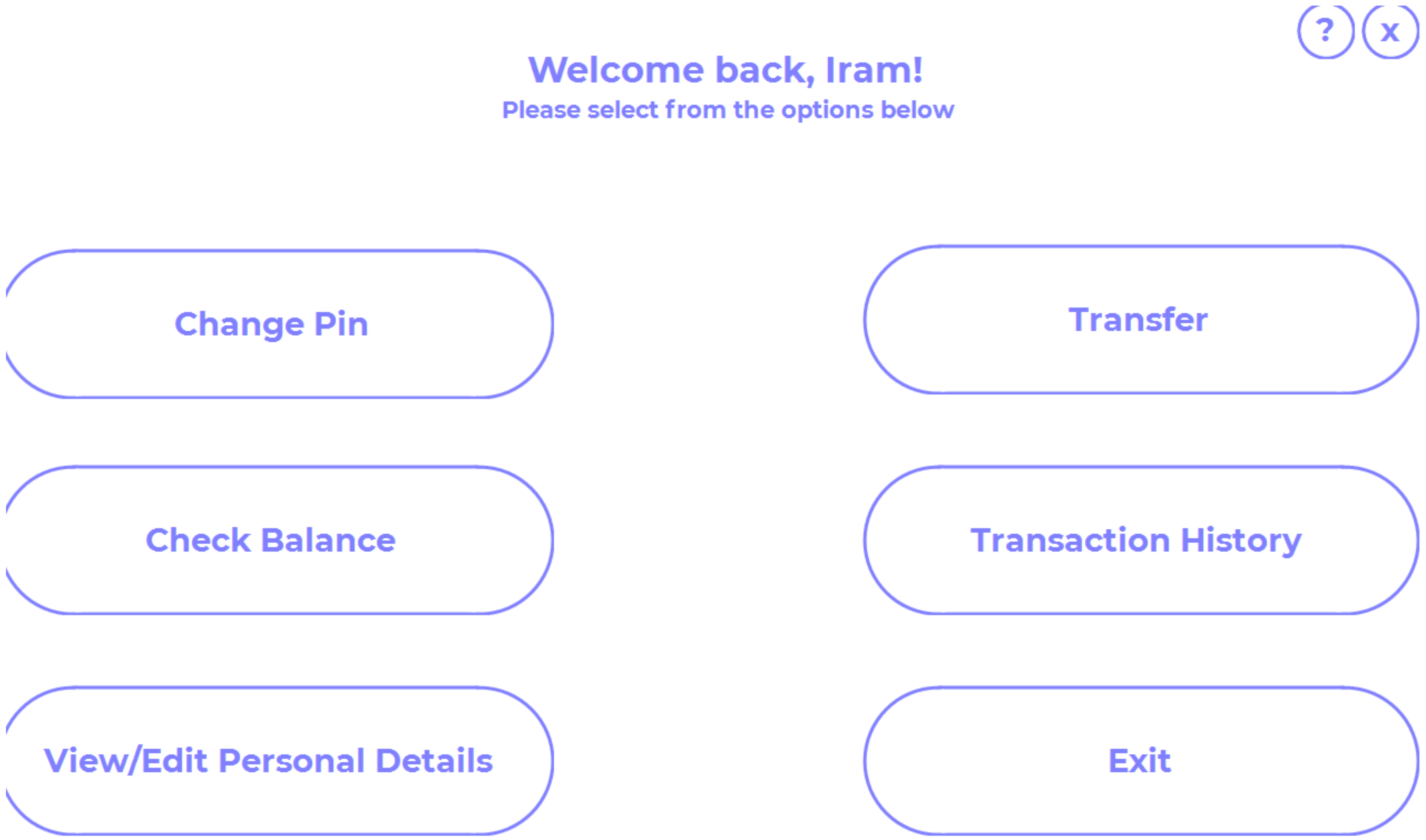


Figure 23: Main Screen

Process	Details	Parameter & Additional Information
Purpose	This screen will display the many options and functionality within the program. The customer will be able to select from the many options available on the screen. All of the forms will also be redirected to this main page with a back button present on each page. This form will display a welcome message with the first name of the customer pulled from the database based on the account login.	The form will display the name of the Account Holder logged in and a welcome message.
Form Called From	frmLogin	This form is called from the Login screen. After the user has confirmed their login successfully, the user can choose from the menu of options.
Form Controls	Button x8 1) btnChangePin 2) btnCheckBalance 3) btnTransfer 4) btnEditPersonalDetails 5) btnCancel 6) btnClose 7) btnTransactionHistory 8) btnHelp	1) Opens Change Pin form 2) Opens Check Balance form 3) Opens Transfer form 4) Opens Edit Personal Details form 5) Closes program 6) Closes program 7) Opens Transaction History 8) Opens Help System
	Label x 2 1) lblAccountDetails 2) lblWelcome	1) This label welcomes the user and displays the users name on screen 2) This label instructs the user to select from the options

	Timer x 2	1) This timer runs for 60 seconds. It will reset if the user interacts with any button on the page, or clicks the form 2) This session will trigger after the MenuTimer has elapsed 60 seconds. It will display a message when the time has elapsed that will tell the user their session has timed out, and they will need to login again.
	1) MenuTimer 2) SessionTimeout	
	Button x14	1-10) These buttons are used for the onscreen keypad, for numbers 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. 11) This button clears the contents of the textboxes 12) This button exits the program 13) This button allows the user to login (Verification check) 14) This button opens a help system
	1-10)btn0-btn9 11) btnClear 12) btnClose 13) btnLogin 14) btnHelp	
	Tooltip x1	Customised tooltip that displays information when hovered upon on a purple background with white font that is distinctive across the systems theme
	ModernBankToolTip7	
Form Behaviour	This form serves as a main menu that has many options displayed as buttons that will redirect the user to other forms as selected.	The user can click any button to go to any of the forms as seen in the main menu
Form Objects & Class	AccountHolder Class ATMConnector Class	Refer to Class Diagram

Change pin

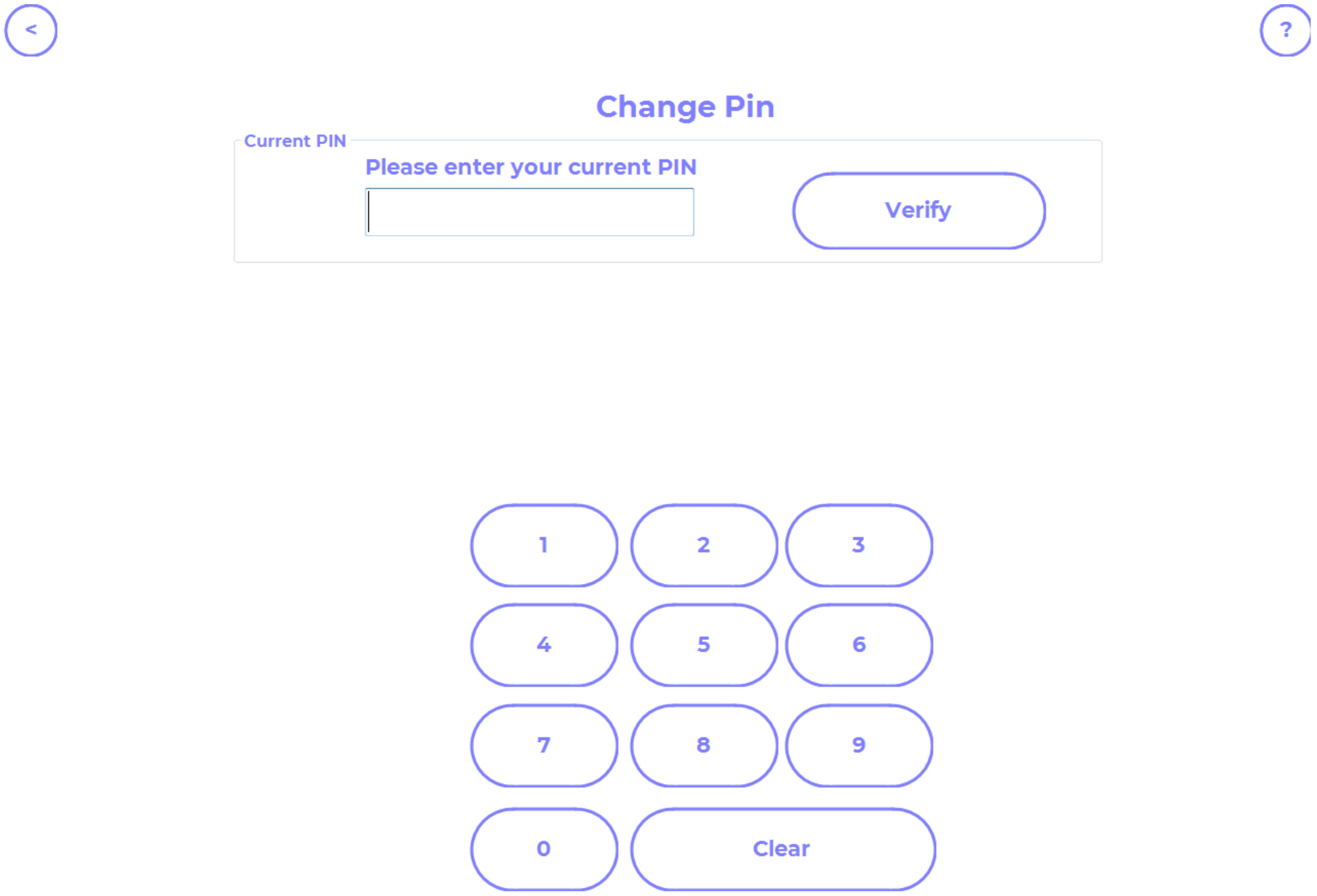


Figure 24: Change Pin

Process	Details	Parameter & Additional Information
Purpose	This screen within the program relies on the input from the user. It requires the end-user to input their current PIN. Once this has been confirmed, the user will then be able to create a new PIN. This PIN will need to be inputted twice, and both of the PIN's inputted must match and be the exact same in order to change the PIN. Once this has been confirmed, the user can save the changes and a message will appear on the green to inform the user of the changes. Once these changes have been confirmed, the program will automatically redirect the user back to the main screen.	The user must enter their current PIN as a confirmation. They can then enter a new PIN that they would like to change their current PIN to. It is then required to confirm this PIN. Both of these new PIN's must match in order to successfully change the PIN.
Form Called From	frmMainSystem	This form is called from the Main Menu. This form handles the process of changing the account holder's current PIN for the account that is signed in.
Data calls on Form	myATMconnector.UpdateCurrentPin(txtNewPin.Text, myLoggedinBankAcount);	@"UPDATE BankAccount SET AccountPin='"+ NEWPin + "' WHERE Guid ='" + myLoggedBankUser._AccountGUID + "'";
Form Controls	Button x14	1-10) These buttons are used for the onscreen keypad, for numbers 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. 11) This button clears the contents of the textboxes 12) This button brings the user back to the main menu 13) This button allows the user to change their PIN (Provided verification beforehand) 14) This button opens the help system 15) This button serves as a verification to ensure the current pin that is entered is correct 16-18) This button is set as a label for an invalid pin (x) 19-20) This button is set as a label for an valid pin (tick)
	1-10)btn0-btn9 11) btnClear 12) btnBackToMain 13) btnPinChangeNow 14) btnHelp 15) btnCheckCurrentPin 16)lblInvalid 17) lblInvalidPin1 18)lblInvalidPin2 19) lblValidPin1 20) lblValidPin2	
	Label x 4	1) This label serves as a title for the form

	1) lblChangePin 2) lblEnterNewPin 3)lblConfirmNewPin 4)lblCurrentPin	2-4) These labs are used to display the information required in the textbox
	Timer x 2	1) This timer runs for 60 seconds. It will reset if the user interacts with any button on the page, or clicks the form 2) This session will trigger after the ChangePinTimer has elapsed 60 seconds. It will display a message when the time has elapsed that will tell the user their session has timed out, and they will need to login again.
	1) ChangePinTimer 2) SessionTimeout	
	Panel x 2	1) This panel is used to store the components relating to the onscreen keypad 2) This panel stores all of the components within the form
	1) pnlOnScreenKeypad 2) pnlChangePin	
	Groupbox x2	1) This groupbox contains the components relating to confirm the current pin 2) This groupbox contains the components relating to the new pin validation
	1) grpChangePin 2) grpNewPIN	
	Tooltip x1	Customised tooltip that displays information when hovered upon on a purple background with white font that is distinctive across the systems theme
	ModernBankToolTip2	
Form Behaviour	This form allows the user to create a new PIN. Requires current PIN validation.	The user can verify and create a new PIN. Must be 4 numbers only. No text or special characters are allowed.
Form Objects & Class	AccountHolder Class BankAccountClass ATMConnector Class	Refer to Class Diagram ATMConnector.UpdateCurrentPin ()

Check Balance

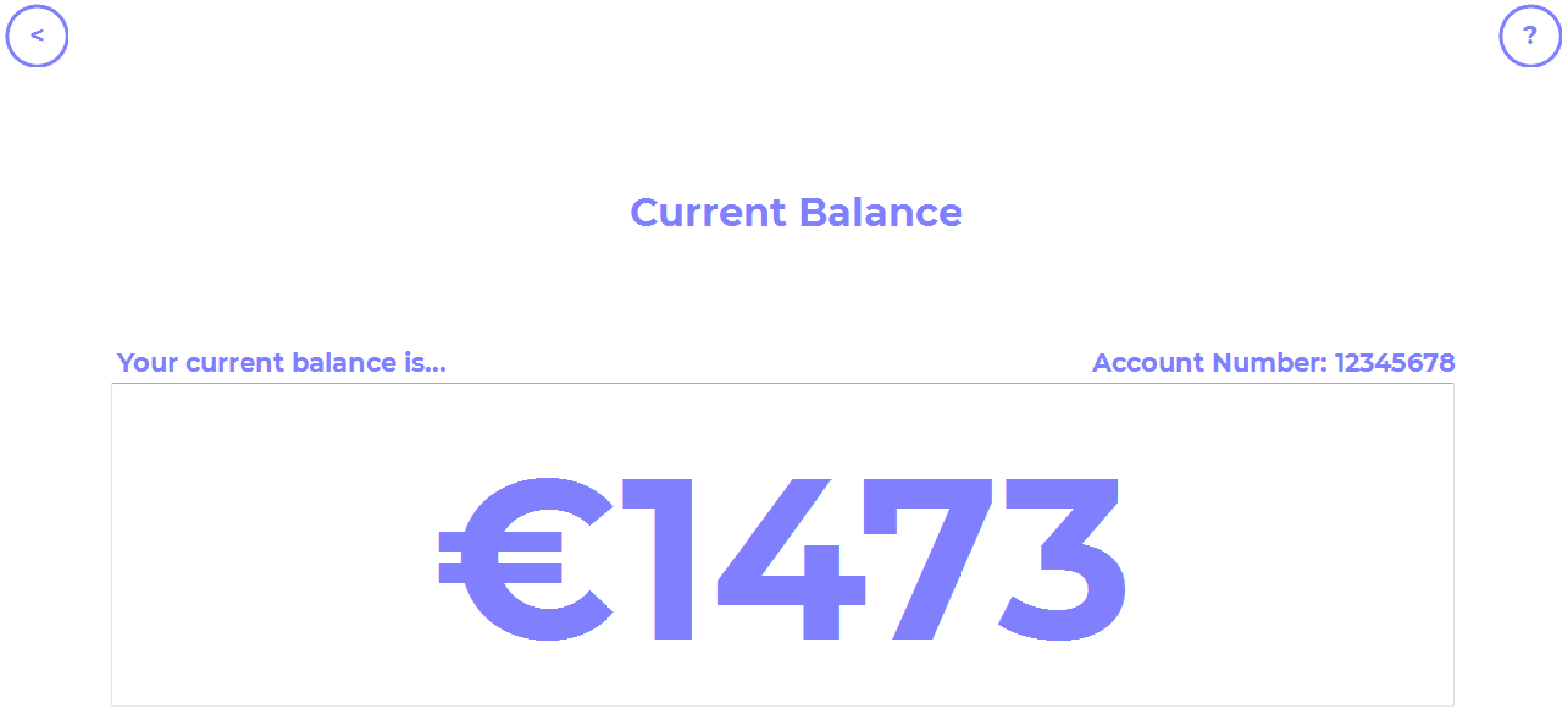
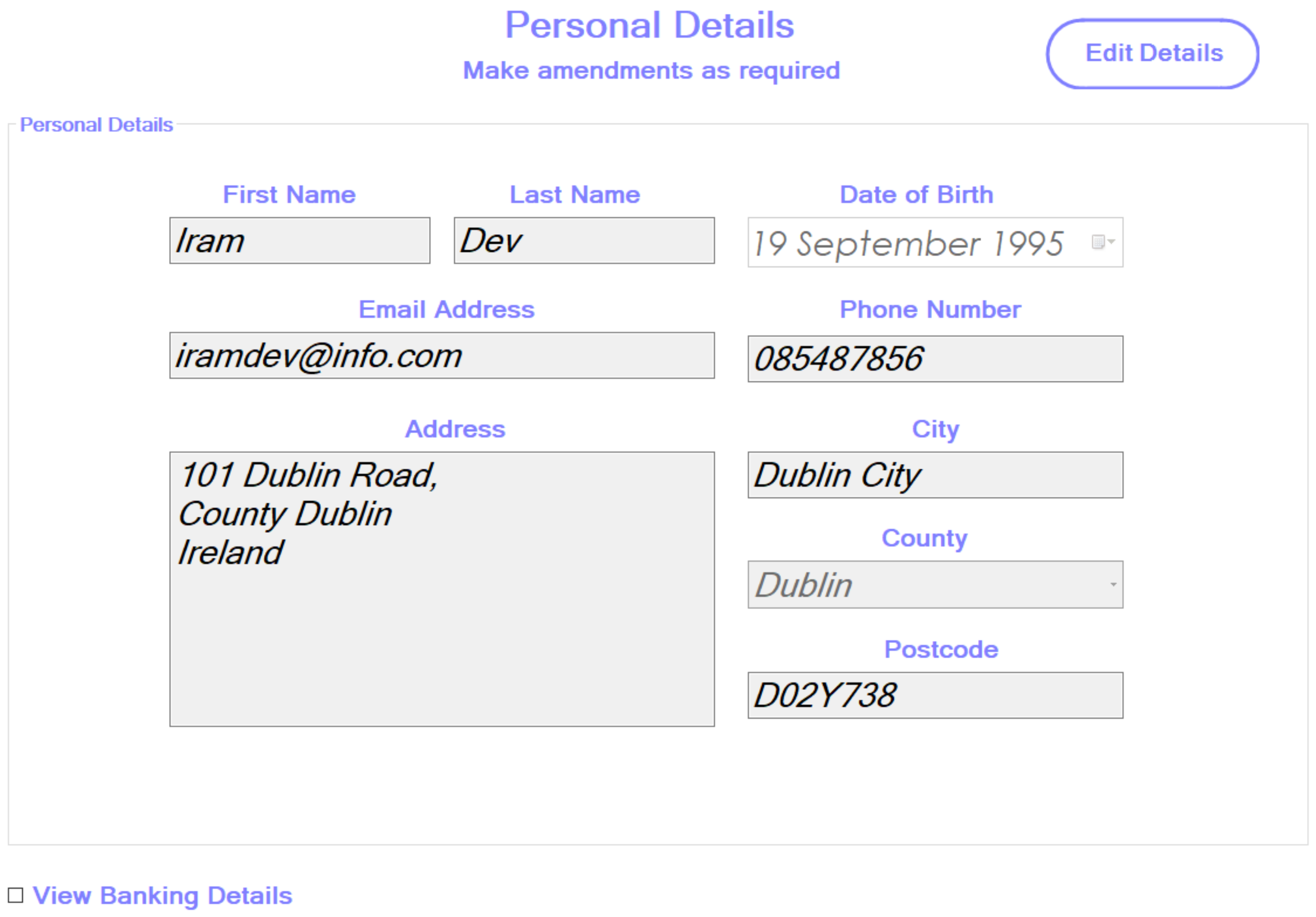


Figure 25: Current Balance

Process	Details	Parameter & Additional Information
Purpose	This screen will begin with a loader, to inform the user that their balance is being retrieved. After 3 seconds the customer’s account balance will be displayed on screen. This is an output as it is pulled from the database.	User can view balance and account number.
Form Called From	frmMainSystem_Load	This form is called from the Main Menu.
Data calls on Form	myATMconnector.FetchBalance(myLoggedinBankAcount);	"SELECT Balance from BankAccount WHERE Guid='" + myLoggedBankUser._AccountGUID + "'";
Form Controls	Panel x 1	This panel stores all of the components on this form
	pnlWelcome	
	Labels x 1	This label displays the text on the screen. Informs the user to touch the screen in order to wake the system
	lblMessage	
	Picturebox x 2	1) This picturebox provides a gif that acts as a pre-loader for the page. 2) This picturebox contains the Modern Bank logo
	1) pbxLoader, 2) pbxLogo	
Form Behaviour	The form displays the balance of the account holder.	Account Balance and Account Number are displayed
Form Objects & Class	AccountHolder Class BankAccountClass ATMConnector Class	Refer to Class Diagram ATMConnector.FetchBalance ()

View/Edit Personal Details


Personal Details

Make amendments as required

Edit Details

Personal Details

First Name	Last Name	Date of Birth
Iram	Dev	19 September 1995

Email Address	Phone Number
iramdev@info.com	085487856

Address

101 Dublin Road,
County Dublin
Ireland

City	Dublin City
County	Dublin
Postcode	D02Y738

☐ View Banking Details

Figure 26: View/Edit Personal Details

Process	Details	Parameter & Additional Information
Purpose	This form serves as both an input and output to the system. The system will load the customer's personal details when viewing this page which serves as an output of the system. This information will be read-only. The user will be able to click a button to edit their details, which will allow input from the user to make changes. A message will appear on screen when the user saves their changes. This form will also display banking details such as the account number, pin and balance when the checkbox is checked at the discretion of the end-user.	Objects created are used here to store and save information relating to the account holder's information. The user will be able to view and edit their personal details
Form Called From	frmMainSystem	This form is called from the Main Menu. This form allows the user to view both their personal and banking details. Only personal details can be modified.
Data calls on Form	myATMconnector.UpdatePersonalDetails(myLoggedinBankAcount);	@"UPDATE AccountHolder SET FirstName='" + myLoggedBankUser.firstname + "', LastName= '" + myLoggedBankUser.lastname + "', DateOfBirth ='" + myLoggedBankUser.dateofbirth + "' , EmailAddress = '" + myLoggedBankUser.emailaddress + "', PhoneNumber ='" + myLoggedBankUser.phonenumber + "', Address = '" + myLoggedBankUser.address + "', City ='" + myLoggedBankUser.city + "', County ='" + myLoggedBankUser.county + "', Postcode ='" + myLoggedBankUser.postcode + "' WHERE Guid='" + myLoggedBankUser._AccountHolderGuid + "';";
Form Controls	Label x14	All of these labels are used as titles to give an indication of which form that they are representing. E.g. lblFirstName = First Name
	1) lblAccountDetails 2) lblMakeAmendments 3) lblFirstName 4) lblPhoneNumber 5) lblEmailAddress 6) lblAddress 7) lblLastName 8) lblDateOfBirth 9) lblPostcode 10) lblCounty 11) lblCity 12) lblAccountNumber 13) lblBalance 14) lblPin	
	Groupbox x2	1) This groupbox is used to group all of the components relating to personal details of the account holder 2) This groupbox is used to group all of the components relating to the banking details of the account holder
	1) grpPersonalDetails 2)grpBankingDetails	

	Timer x 2	1) This timer runs for 60 seconds. It will reset if the user interacts with any button on the page, or clicks the form 2) This session will trigger after the DetailsTimer has elapsed 60 seconds. It will display a message when the time has elapsed that will tell the user their session has timed out, and they will need to login again.
	1)DetailsTimer 2) SessionTimeout	
	Button x4	1) This button is used to save changes made to the user's personal details 2) This button redirects the user back to the main menu 3) This button will allow the user to edit their personal details – from read-only to modifiable 4) This button opens the help system
	1) btnSaveChanges 2) btnBackToMain 3) btnEditDetails 4) btnHelp	
	Textbox x8	All of these textboxes display the information of the user's personal details.
	1) txtFirstName 2) txtLastName 3) txtPhoneNumber 4) txtEmailAddress 5) txtAddress 6) txtCity 7) txtCounty 8) txtPostcode	
	Checkbox x1	This checkbox will allow the user to fill it in order to display the user's banking details.
	cbxBankDetails	
	DateTimePicker x1	This date time picker allows for the user to view their date of birth, or to update it.
	dtpDateOfBirth	
	Tooltip x1	Customised tooltip that displays information when hovered upon on a purple background with white font that is distinctive across the systems theme
	ModernBankToolTip2	
Form Behaviour	This form allows the user to view both their banking and personal details. They can only edit and update their personal details.	User can edit their name, email, phone, address
Form Objects & Class	AccountHolder Class BankAccountClass ATMConnector Class	Refer to Class Diagram ATMConnector.UpdatePersonalDetails ()

Transfer

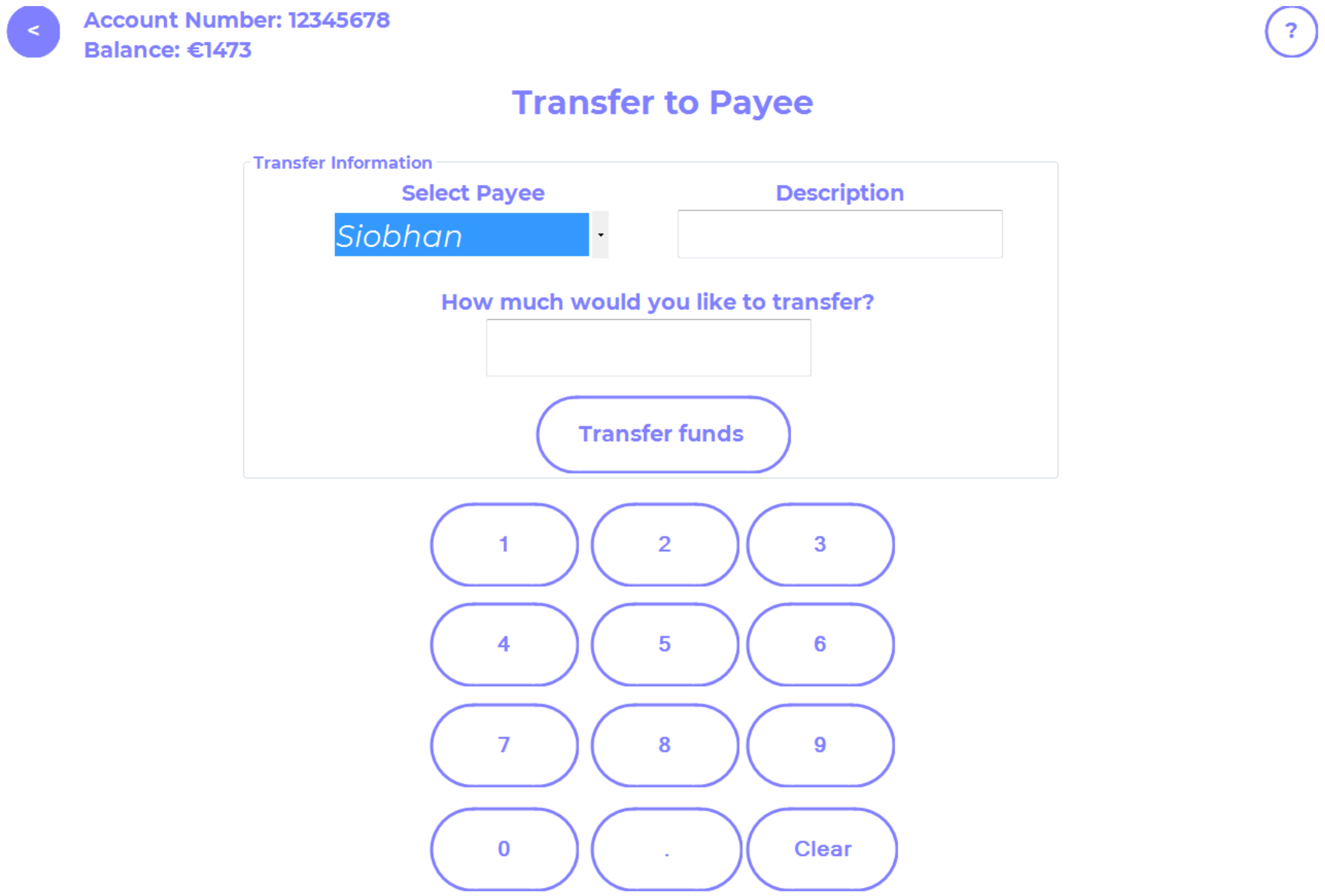


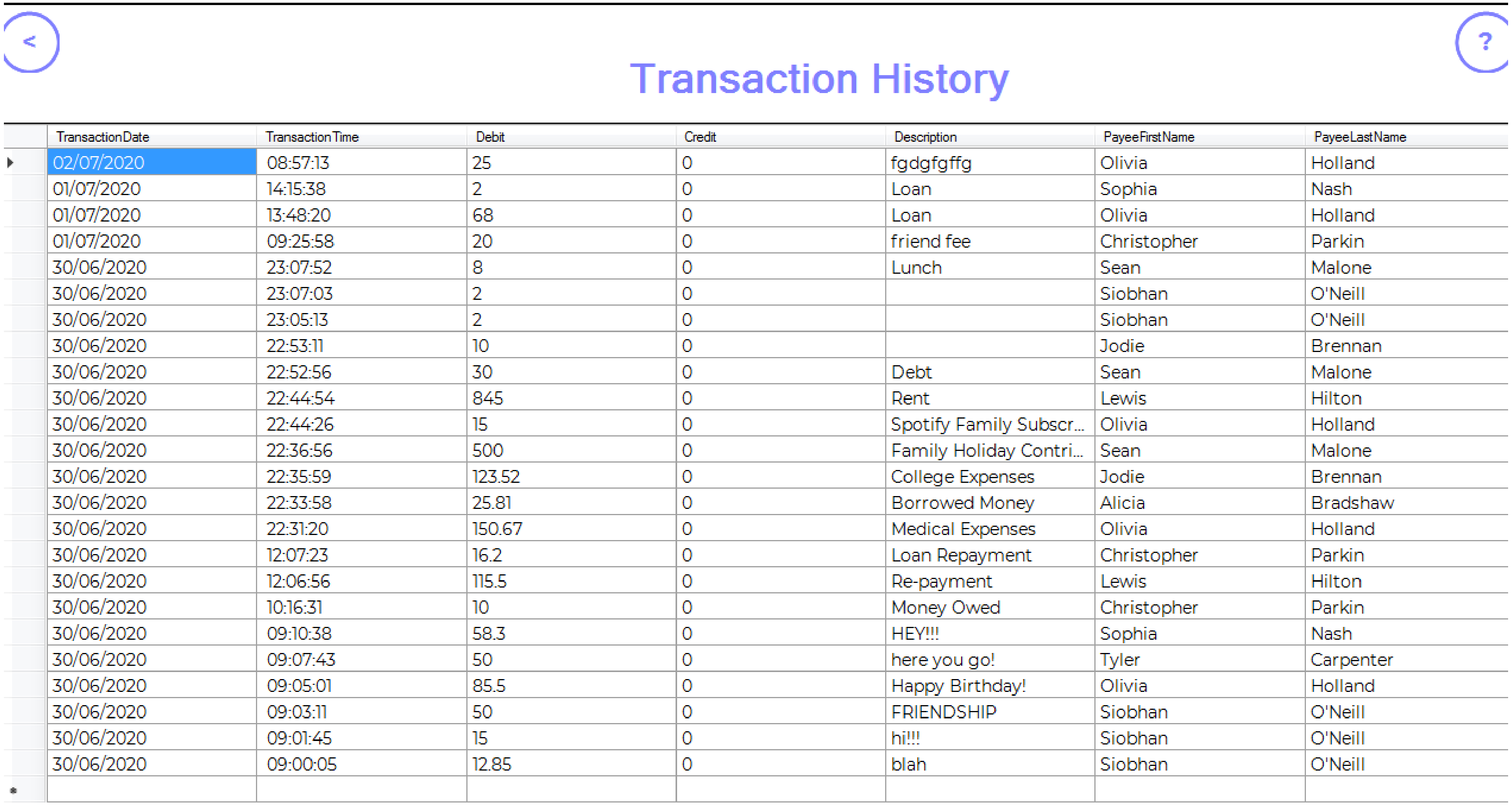
Figure 27: Transfer

Process	Details	Parameter & Additional Information
Purpose	This form will allow the customer to transfer money from one account to another. The user may have multiple bank accounts which they can choose from in a combobox. They can decide to transfer payments to their other bank accounts, if applicable, otherwise, they will be able to transfer funds to any payee's that they have already added. If they do not have any payees added, they will need to return to the main screen and add a payee. The customer will then need to choose how much they would like to transfer. For ease of access, the form will display the customer's account number, in the case that they may have multiple accounts, to ensure that they know what account they are transferring from, and their balance so they know what funds are available in their account. The user will not be able to transfer funds that they do not have.	Payees information is stored in a combobox – pulled from the database. Description and Amount – User input
Form Called From	frmMainSystem	This form is called from the Main Menu. This form allows the user to make transfers to payees.
Data calls on Form	myATMconnector.TransferTransaction(PayeeAccount.SelectedValue.ToString(), Convert.ToDouble(txtCashInput.Text), txtDescription.Text, myLoggedinBankAcount);	1) @"INSERT INTO BankTransactions (Guid,BankAccountGUID,Debit,Credit,DateOfTransaction,PayeeAccountGUID,Description) VALUES ('" + TransactionGuid.ToString() + "','" + myLoggedBankUser._AccountGUID + "','" + DebitBalance.ToString() + "','" + CreditBalance.ToString() + "','" + DateTime.Now.ToString() + "','" + TransferToAccountGuid + "','" + TransferDescription + "')"; 2) @"UPDATE BankAccount SET Balance = '" + UpdatedBalance.ToString() + "'" + " WHERE Guid='" + myLoggedBankUser._AccountGUID + "';"; 3) @"INSERT INTO BankTransactions (Guid,BankAccountGUID,Debit,Credit,DateOfTransaction,PayeeAccountGUID,Description) VALUES ('" + TransactionGuid.ToString() + "','"

		+ TransferToAccountGuid + "','" + DebitBalance.ToString() + "','" + CreditBalance.ToString() + "','" + DateTime.Now.ToString() + "','" + "" + "','" //CREDIT + TransferDescription + "')"; 4) @"UPDATE BankAccount SET Balance = '" + PayeeAccountBalance.ToString() + "'" + " WHERE Guid='" + TransferToAccountGuid + "';";
Form Controls	Label x7	All of these labels are used as titles to give an indication of which form that they are representing. They are used as titles for the form
	1) lblTransferPayee 2) lblAmount 3) lblSelectPayee 4) lblBalance 5) lblAccountNumber 6) lblAccountNumber 7) lblDescription 8	
	Button x17	1-10) These buttons are used for the onscreen keypad, for numbers 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. 11) This button clears the contents of the textboxes 12) This button brings the user back to the main menu 13) This button allows the user to transfer funds (Depending on verification) 14) This button inserts a dot to the textbox input 15) This button opens the help system 16) This button represents a valid image 17) This button represents an image for invalid input
	1-10)btn0-btn9 11) btnClear 12)btnBackToMain 13) btnTransferFunds 14)btnDot 15) btnHelp 16) btnlblValid 17) lblInvalid	
	Timer x 2	1) This timer runs for 60 seconds. It will reset if the user interacts with any button on the page, or clicks the form 2) This session will trigger after the the TransferTimer has elapsed 60 seconds. It will display a message when the time has elapsed that will tell the user their session has timed out, and they will need to login again.
	1)TransferTimer 2) SessionTimeout	
	Panel x2	1) This panel contains all of the components relating to the onscreen keypad 2) This panel contains all of the components relating to the transfer

	1) pnlOnScreenKeypad 2) pnlTransfer	
	Textbox x2	1) This textbox is used for the input from the user – to enter the amount of cash from their account they would like to transfer 2) This textbox allows for the input from the user – to enter a description for the transfer they are completing
	1) txtCashInput 2) txtDescription	
	Combobox x1	This combobox is filled with a selection of payees that the user can select from to transfer funds to.
	PayeeAccount	
	Groupbox x1	This groupbox contains all of the contents relating to the transfer function
	grpTransfer	
	Tooltip x1	Customised tooltip that displays information when hovered upon on a purple background with white font that is distinctive across the systems theme
	ModernBankToolTip2	
Form Behaviour	This form allows the user to make transfers to payees. The user must select a payee, enter a description and a valid amount they would like to transfer.	AccountPayee, Description, Amount to send
Form Objects & Class	AccountHolder Class BankAccountClass ATMConnector Class	Refer to Class Diagram ATMConnector. TransferTransaction ()

Transactions



Transaction History

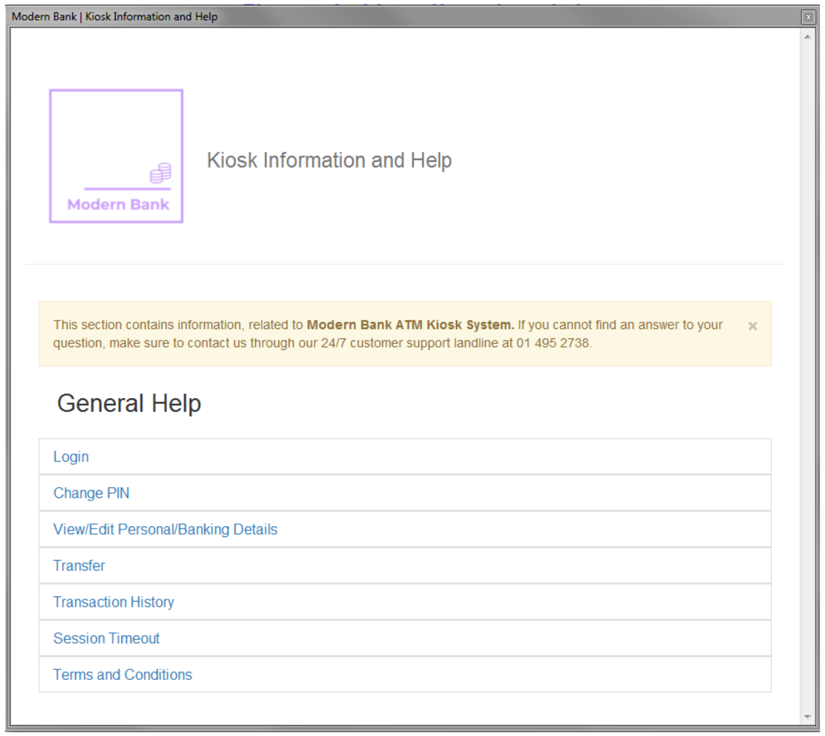
TransactionDate	TransactionTime	Debit	Credit	Description	PayeeFirstName	PayeeLastName
02/07/2020	08:57:13	25	0	fgdgfgffg	Olivia	Holland
01/07/2020	14:15:38	2	0	Loan	Sophia	Nash
01/07/2020	13:48:20	68	0	Loan	Olivia	Holland
01/07/2020	09:25:58	20	0	friend fee	Christopher	Parkin
30/06/2020	23:07:52	8	0	Lunch	Sean	Malone
30/06/2020	23:07:03	2	0		Siobhan	O'Neill
30/06/2020	23:05:13	2	0		Siobhan	O'Neill
30/06/2020	22:53:11	10	0		Jodie	Brennan
30/06/2020	22:52:56	30	0	Debt	Sean	Malone
30/06/2020	22:44:54	845	0	Rent	Lewis	Hilton
30/06/2020	22:44:26	15	0	Spotify Family Subscr...	Olivia	Holland
30/06/2020	22:36:56	500	0	Family Holiday Contri...	Sean	Malone
30/06/2020	22:35:59	123.52	0	College Expenses	Jodie	Brennan
30/06/2020	22:33:58	25.81	0	Borrowed Money	Alicia	Bradshaw
30/06/2020	22:31:20	150.67	0	Medical Expenses	Olivia	Holland
30/06/2020	12:07:23	16.2	0	Loan Repayment	Christopher	Parkin
30/06/2020	12:06:56	115.5	0	Re-payment	Lewis	Hilton
30/06/2020	10:16:31	10	0	Money Owed	Christopher	Parkin
30/06/2020	09:10:38	58.3	0	HEY!!!	Sophia	Nash
30/06/2020	09:07:43	50	0	here you go!	Tyler	Carpenter
30/06/2020	09:05:01	85.5	0	Happy Birthday!	Olivia	Holland
30/06/2020	09:03:11	50	0	FRIENDSHIP	Siobhan	O'Neill
30/06/2020	09:01:45	15	0	hi!!!	Siobhan	O'Neill
30/06/2020	09:00:05	12.85	0	blah	Siobhan	O'Neill

Figure 28: Transaction History

Process	Details	Parameter & Additional Information
Purpose	This form when loaded will display all of the transactions based on the bank account that is logged in. This data will be loaded into a database that will allow the user to view their transactions. The transactions will display Debit as OUT, Credit as IN (funds), a date of the transaction and a description of the transaction that has occurred. This form is an output as it relies on gathering data stored within the database.	User can view transactions, from recent descending.
Form Called From	frmMainSystem_Load	This form is called from the Main Menu. It allows the user to view their transactions.
Data calls on Form	myATMconnector.ViewTransactionHistory(myLoggedinBankAcount);	"SELECT substr(DateOfTransaction,0,11) AS TransactionDate,substr(DateOfTransaction,11,14) AS TransactionTime, Debit, Credit, Description,AccountHolder.FirstName AS PayeeFirstName, AccountHolder.LastName AS PayeeLastName FROM BankTransactions LEFT JOIN BankAccount ON BankAccount.Guid = PayeeAccountGUID LEFT JOIN AccountHolder ON AccountHolder.Guid = BankAccount.AccountHolderGUID WHERE BankAccountGUID = '" + myLoggedBankUser._AccountGUID + "' ORDER BY BankTransactions.uid DESC ;";
Form Controls	Datagridview x 1	This datagridviewer displays the contents of the transactions table within the database – that displays the user's transaction history in a grid
	dgvTransactions	
	Label x 1	This label is used to label the Transactions title
	lblAccountDetails	

	Button x2 1) btnBackToMain 2)btnHelp	1) This button is used to redirect the user back to the main menu 2) This button is used to open the help system
	Tooltip x1 ModernBankTooltip5	Customised tooltip that displays information when hovered upon on a purple background with white font that is distinctive across the systems theme
	Timer x 2 1) TransactionTimer 2) SessionTimeout	1) This timer runs for 60 seconds. It will reset if the user interacts with any button on the page, or clicks the form 2) This session will trigger after the TransactionTimer has elapsed 60 seconds. It will display a message when the time has elapsed that will tell the user their session has timed out, and they will need to login again.
Form Behaviour	The form displays the user's transaction history	Displays transaction history – date, time, debit, credit, description, payee name
Form Objects & Class	AccountHolder Class BankAccountClass ATMConnector Class	Refer to Class Diagram ATMConnector.FetchBalance ()

Help System



Modern Bank | Kiosk Information and Help

Modern Bank

Kiosk Information and Help

This section contains information, related to **Modern Bank ATM Kiosk System.** If you cannot find an answer to your question, make sure to contact us through our 24/7 customer support landline at 01 495 2738.

General Help

- Login
- Change PIN
- View/Edit Personal/Banking Details
- Transfer
- Transaction History
- Session Timeout
- Terms and Conditions

Figure 29: Help System

Process	Details
Purpose	This Help screen will be available to view from each screen in the system. It will pop out and can be closed by clicking the x button, without exiting the program itself. It contains information regarding each page in the system. It has been created with HTML and the form simply contains a web browser element from the design components with the data source.

8.0 System Integrity Controls

Within this project, there has been an emphasis for high system security within the program. This is especially for a project such as an ATM, as this type of system deals with highly sensitive and confidential information. This includes sensitive information such as an account holders personal details, such as their name, date of birth, email address, phone number and address. Not only this, but these details stored in a database, which contain even more sensitive information, such as the account holder's bank account details – their account number, pin and their balance. Information like this is especially sensitive, so it is so important to have high security for this type of information.

In the future we would highly recommend to consider encryption within the system and the database, to ensure high security of this information. This is highly recommended to do so before the system goes live, as this data must be GDPR compliant.

At the moment, ***General Data Protection Regulation (GDPR)*** is a hot topic as laws regarding GDPR compliance are constantly evolving. It is important to stay on top of it, and to be aware of the changes as it can have massive affects to how a system can store its data. GDPR can cause the direction of the development of the system to change if laws are modified or updated. On May 25 2018, the European Union imposed obligations and regulations regarding the capturing and targeting of data to anyone living within the EU. This had a massive impact on systems and companies not only in the European Union, but all around the world – as visitors from the EU would need their data captured differently, in accordance with the new regulations – as compared to visitors outside of the EU.

This was a very strict law that was created with the aim of tighter online privacy and security for the people living in this union. There are harsh penalties and fines for those who violate the privacy and security standards – so it is something that a company must be mindful of, and then integrate within a system. It is equally as important to keep up to date in regards to the current GDPR laws, and to implement these changes within a system that is live.

Essentially, it does have an impact in the development of a system, as the system cannot go live unless it is GDPR compliant, or the company will face fines.