Neeraj Patel 2020 (N0920737)

N0920737@my.ntu.ac.uk

This report contains the detailed specification, design and implementation and testing that is required for my proposed project, the Banking System.

Soft10101: computer science programming 2020-2021

Banking System Project

# Project Specification:

## Project Definition:

The project that I chose to implement is the Banking Management System. The Banking Management System is a C++ program which will be used for maintaining accounts for a Bank. The program will allow for the creation of new accounts, and the management of existing accounts. The program will allow users to input their Credit/Debit Card identification, and pin to gain access into their account. The details will then be validated with the stored information to see if the correct details were entered. Once gaining access into the account, the user is able to complete the various functionalities available within the Banking Management System. The Banking Application and the contents of the file must be secure, and throughout the program there will be error handling and input handling to prevent incorrect entry.

Key features of the Banking Management Application:

* Allowing the user to create an account within the application.
* Performing transactions on the account per the users requirements.
* Checking the CC/Debit card and pin with the stored information.
* Modifying the details on the accounts.
* Managing different types of accounts, such as ISA’s, Mortgages and Loans, with different functions or classes for each type.
* Visual interface: Terminal Based

## The functional requirements include:

|  |  |
| --- | --- |
| Functional Requirement Number | Functional Requirement Description |
| FR1 | Allowing the user to create or access their account |
| FR2 | Checking that the details input by the user are accurate and match what is stored. |
| FR3 | The user can withdrawal from the account |
| FR4 | The user can deposit into their account |
| FR5 | The user can modify the account as per their requirements: For example: Changing Address. |
| FR6 | The user can create and view different types of accounts that are created, For Example: ISA Savings, Mortgages and Loans. |
| FR7 | The user can view the recent transactions |
| FR8 | The changes in application must be saved to the stored file. |
| FR9 | The security of the Credit Card information files. |
| FR10 | Validation throughout, errors in input must be checked. |

# Design and implementation:

## Program flow:

Start

Get input for Main Menu

MENU:

1 TO LOG IN

2 TO CREATE

3 FOR STAFF

Create Account:

int getData()

void cleanData()

void displayData()

void storeInfo()

void AccountMake()

Log In:

bool CustomerLogin()

Get input for Banking Menu

Banking Menu:

1 to Check Balance

2 to Deposit

3 to Withdraw

4 to Transfer

Withdraw Amount

Deposit Amount

Update file and balance

Transfer Amount

Display Balance

3

2

Staff Login:

bool StaffLogin()

1

2& 3

1

4

## Class Diagram:

While creating my program, I decided to split the project into three classes, and created the class diagram’s below to help me to implement them successfully into my program.

<<friend>>

User

Private attributes:

string fname

string lname

string fullname

string address

string postcode

string passportnum

string typeofaccount

double balance

int pin

Public Operations/Methods:

void displayData()

int storeInfo()

int getData()

int cleanData()

void getName()

void getAddress()

void getPassport()

void getAccountType

Private attributes:

string pinAttempt

string pin

string cardAttempt

string cardNumber

Private Operations/Methods:

void pinGen()

void ccGen()

bool CustomerLogin()

bool StaffLogin()

Public Operations/Methods:

void AccountMake()

LoginSystem

Private attributes:

Double CurrentAccRate

Double SavingsAccRate

Double ISAAccRate

string balance

Private Operations/Methods:

void verifyDetails()

void updateDetails(int append balance)

void accBalance()

void accDeposit()

void accWithdraw()

void accTransfer()

void InterestCalc()

void updateRec()

void menu()

Public Operations/Methods:

void adminMenu()

void atmMenu()

Bank

## File structure:

The file structure for the Banking Management System involves separate .txt text files. When an account is created, and text file is created and is named with the credit card number to allow for reading and writing, and validation of details. The text file contains the credit card number, pin, and balance on separate lines (as seen below).

Graphical user interface, application, Word

Description automatically generated

Image 1: Example of Account

To store the customers identification details, a separate file is created. This is called userinfo.csv, and this allows for the Bank Staff to access the details of the Bank’s customers.Table

Description automatically generated

Image 2: Example of CSV file storing accounts.

# Testing:

|  |  |  |
| --- | --- | --- |
| Action | Test Method | Outcome |
| User input throughout all the menu’s | 1. Throughout the Menu’s, you can test whether the correct outcome is received when inputting a correct input.  2. Throughout the Menu’s, you can test whether, you are given the option to re-enter an option when selecting a wrong input to menu. | 1. Successful entry will result in the successful function being called.  2. The user is prompted for the input again. |
| Log into an account | 1. Use the correct details to login to an account  2. Using incorrect details to login to an account | 1. Successful entry into the ATM, as details are validated.  2. Prompt the user to input valid details. |
| Creating an account | 1. Throughout the different stages of the creation of the account, the user input’s the correct details, for example when prompted for a United Kingdom postcode, the user will enter a 6-8 digit postcode.  2. Throughout the creation stage, the user inputs, the incorrect details. | 1. Successful entry will result in a new account being created.  2. Incorrect Entry, means that the user will be promoted to renter valid details. |
| Account facilities | 1.The user should test each of the account facilities, and make sure that they are working correctly when the user inputs the details correctly.  2. The user tests the account facilities by entering incorrect information, such as withdrawing more money, that amount in balance. To test that the functionalities are working correctly. | 1. The user will be able to use all of the functionalities if input is correct.  2. The user will be prompted to enter the correct details. |

# FUNCTIONAL REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| Functional Requirement Number | Functional Requirement Description | Done |
| FR1 | Allowing the user to create or access their account | YES |
| FR2 | Checking that the details input by the user are accurate and match what is stored. | YES |
| FR3 | The user can withdrawal from the account | YES |
| FR4 | The user can deposit into their account | YES |
| FR5 | The user can modify the account as per their requirements: For example: Changing Address. | YES |
| FR6 | The user can create and view different types of accounts that are created, For Example: ISA Savings, Mortgages and Loans. | YES |
| FR7 | The user can view the recent transactions | YES |
| FR8 | The changes in application must be saved to the stored file. | YES |
| FR9 | The security of the Credit Card information files. | YES |
| FR10 | Validation throughout, errors in input must be checked. | YES |

# DEMO

The recordings for the Demo are attached to the file as required.