Exercise 2: Login System

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 This system however is outdated and does not provide much response to the user's current errors or invalid entries.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.

This exercise consists of two parts; a report and an application.

Exercise 2 a: Report

Prepare a design specification report describing a solution for the above scenario. In your report identify the objects, data and file structures required.

Also include in your specification:

- Clear and concise technical and end user help documentation
- Discussion on how to load test the login system (resources, procedures etc.)

The report is to be between 750 and 1,000 words.

Exercise 2 b: Application

Based on the requirements of the report you developed in Exercise 2 a, implement the Login System in C# using Microsoft Visual Studio Integrated Development Environment.

A quick textual analysis of the scenario above yields the following key words:

- Bank
- Simple
- Login System
- Customer
- Errors and invalid entries
- Inform the user

Adding to the key words:

- Bank
    - Think in terms of a general business as there are no "Bank" specific requirements.
- Simple
    - Use role based permissions. Privileges are granted to the user based upon the user's role within the bank. A user privilege is a right to execute a particular option within the application. See Appendix A
    - Find a balance. The system needs to work for the bank and be generic enough to be easily applied to other systems and organisations
    - It would be nice to incorporate a database into this application. However, at this stage of development, and to keep it simple – use a CSV file to store the login details (read only). For the purposes of this

exercise use Excel or Notepad to write to the CSV file. See Appendix A for the structure of the CSV file

- Login System
    - Login Credentials to this system are to consist of a User ID and Password
    - User ID (label and text box)
    - Password (label and text box)
      See Appendix A for details of the password strength requirements
      A C# class for validating passwords will be supplied to you
    - Login (button)
      Greyed out until User Name and Password have been entered
      Check that the credentials entered are valid and if they are, close the login dialog and open the main application form
    - Register (button)
      Open a dialog to register an existing customer for a new password, customer display the details, and solicit a password and simulate th e saving of the data to the CSV file
    - Forgot Password (button)
      Open a dialog and solicit authentication details from the user
      Authenticate the user and email the user a new password (just simulate the email operation)
    - Help (button)
      Display a help dialog including end user help text outlining how the login system works
- Customer
    - The word customer is very specific, think in terms of end user. See Appendix A for list of end user roles for this system
- Errors and invalid entries
    - Error messages
      Displayed via a message dialog
- Inform the user
    - Display on screen help and “how to” details in a dialog

Notes:

- The focus of this exercise is for you to demonstrate your knowledge of event driven programming.
- A C# class for validating passwords will be supplied to you.
- The password CSV file; **`p_words.csv`** will be supplied to you.
- Cater for a backdoor password. A backdoor password is a master password that grants the software developers access to the system without the need to create an account. For obvious reasons this password must be kept completely secret. The system will be compromised if it falls into the wrong hands. Use "**`fred`**" for both the backdoor **`User Id`** and **`Password`**, and grant full access privileges.
- Create a very basic main application form, just include a menu bar with a File menu that has an Exit option (which exits the application when selected).
- Your application must include a bare minimum of five of the following (you will probably need more than five). Feel free to add to this list:

• Forms	• Project
• Common Dialogs	• Classes
• Dialogs	• Public
• Text Boxes	• Private
• Lables	• Listeners
• Command Buttons	• Exceptions
• Combo boxes	• Variables
• List boxes	• Strings
• Check boxes	• Arrays
• Radio buttons	• Sequential File Handling

- Your application **must**:
  - Compile
  - Run
  - Be fully compliant with the specification
  - Include error message dialogs
  - Include end user help and guidance

Appendix A: Password File Record Specification

Filename: p_words.csv

Sequence	Column	Required	Data Type	Min Chars	Max Chars	Min Value	Max Value	Default Value
1	First Name	Yes	String (Alpha)	3	25			
2	Last Name	Yes	String (Alpha)	3	25			
3	User Id	Yes	String (Alpha Numeric)	8	8			
4	Password	Yes	String (Alpha Numeric)	8	16			
5	Role	Yes	String (Alpha)	3	35			Customer
6	Account Number	Yes, for Role: Customer No for other roles	String (Numeric)	9	9			
7	Email Address	Yes	String (Alpha Numeric)	15	35			
8	Administration Full Access	Yes	Boolean	N/A	N/A	No	Yes	No
9	Administration Report Privileges	Yes	Boolean	N/A	N/A	No	Yes	No
10	Generate Audit Records	Yes	Boolean	N/A	N/A	No	Yes	No
11	View Audit Records	Yes	Boolean	N/A	N/A	No	Yes	No
12	Input Account Payments	Yes	Boolean	N/A	N/A	No	Yes	No
13	Authorise Account Payments	Yes	Boolean	N/A	N/A	No	Yes	No
14	Manage Account	Yes	Boolean	N/A	N/A	No	Yes	No
15	View Account Information	Yes	Boolean	N/A	N/A	No	Yes	No
16	View Account Balances	Yes	Boolean	N/A	N/A	No	Yes	No

Description

This file contains the passwords and related data. It is a **C**omma **S**eparated **V**alues (CSV) format file. It represents a rectangular array of text and numeric values and is an example of a "flat file" file format. The columns/fields are separated or delimited using a comma and each record/row ends with a line break. Each line contains the same number of fields. The file contents and structure may be viewed, edited and updated using Microsoft Excel or Notepad.

Your application must include routines for reading this file. This exercise is supposed to demonstrate your skill at event driven programming, therefore, I suggest reading the file in either the **`formLoad`** or the **`formActivated`** event. The load event occurs before a form is displayed for the first time and the activated event occurs when the form is activated in code or by the user.

Field Requirements

Sequence: 3

Column: User Id

The User ID for bank employees is derived from:

1. The first three characters of the employees first name
2. The first three letters of the employees last name
3. A random 2-digit integer – used to make the user name unique
4. Accented characters are used without the accent
5. Characters other than a – z are ignored

Sequence: 4

Column: Password

The password must meet the following eight strength requirements:

1. Minimum 8 characters
2. Maximum 16 characters
3. Have at least one upper case (A - Z) character
4. Have at least one lower case (a - z) character
5. Have at least one number (0 - 9)
6. Have at least one of the 31 symbols or special characters from this list:

   ```
   ~ ! @ # $ % ^ & * (
   ) _ + - = { } | [ ]
   \ : " ; ' < > ? , . /
   ```

7. Must not contain whitespace such as spaces or tabs
8. Must be free of consecutive identical characters

Any password entered by the user must conform to the requirements above.

A C# class for validating passwords will be supplied to you.

Sequence: 5

Column: Role

Seven user roles have been defined for this system. A role is assigned to each user. A role groups together a set of privileges that a bank employee or customer has, setting out what the user can and can't do. Typically, these roles grant the user permission to access various types of information and varying levels of functionality.

The roles in the bank system are:

- ***Super User (All Payments)***
  Allows users to input and authorise all payment types, and also includes all privileges for Administration, Audit and Account information
- ***Super User***
  Allows access to view all audit and account information, and all administration privileges.
- ***Administrator***
  Gives users all privileges for administration
- ***Auditor***
  Allows users to view audit records and privileges reports
- ***Account Operator***
  Allows users to manage accounts and view account information
- ***Balance Operator***
  Allows users to only view balances for all accounts
- ***Customer***
  Allows customers to view the balances only for their accounts