University of Derby
Department of Computing and Engineering

Assignment Specification

Module Code and Title: Programming II (4CC511)	
Assignment No. and Title: Information Tracker Application	
Assessment Tutor: **Patrick Merritt**	**Weighting Towards Module Grade:** 100%
Date Set: 25th January 2021	**Hand-In Deadline Date:** Monday 10th May 2021 at 11:59pm

Conditions for Late Submission

Recognising that deadlines are an integral part of professional workplace practice, the University expects students to meet all agreed deadlines for submission of assessments. However, the University acknowledges that there may be circumstances that prevent students from meeting deadlines. There are now 3 distinct processes in place to deal with differing student circumstances:

1. Assessed Extended Deadline (AED): Students with disabilities or long-term health issues are entitled to a Support Plan.
2. Exceptional Extenuating Circumstances (EEC): The EEC policy applies to situations where serious, unforeseen circumstances prevent the student from completing the assignment on time or to the normal standard. http://www.derby.ac.uk/eec
3. Late Submission: Requests for late submission will be made to the relevant Subject Manager in the School (or Head of Joint Honours for joint honours students) who can authorise an extension of up to a maximum of one week.

Work that is submitted up to one week late without being covered under one or more of the above processes to deal with student circumstances will be awarded a maximum grade of 40%. Work submitted more than one week late will be awarded a grade of 0. You must submit your work; otherwise, a grade of NS (non-submission) will be recorded. Under the University regulations, a non-submission may be considered evidence of non-engagement, which may lead to your being removed from the course.

Level of Collaboration

This is an individual assignment. No collaboration with other students or anyone else is allowed.

Learning Outcomes covered in this Assignment:

1. Design, develop, and test computer applications of moderate complexity, using a professional approach.
2. Demonstrate the ability to implement software that meets the requirements described in a specifications document.

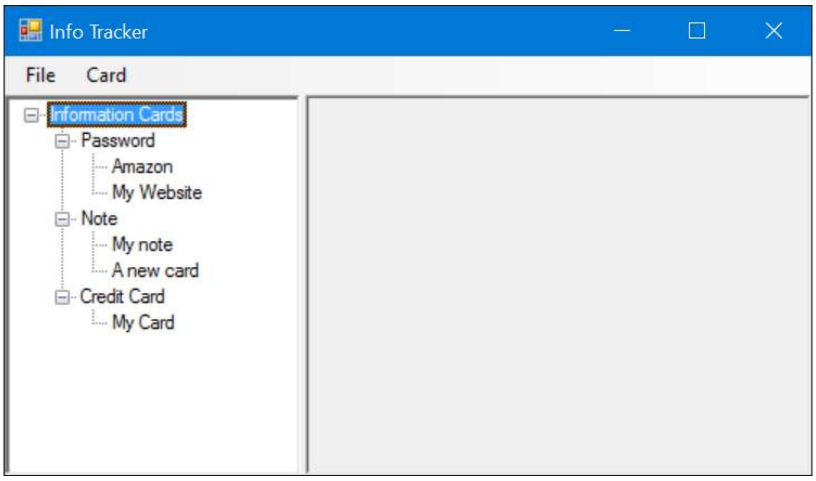
Assessment Criteria

The assessment criteria are documented below. Make sure you read them carefully as well as the submission requirements. Failure to submit the assignment correctly could have a major impact on your grade.

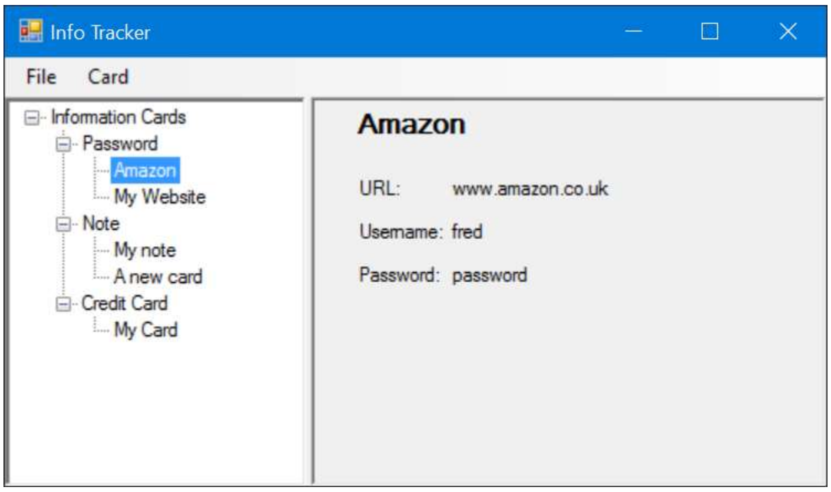
Scenario

You are part of a team working on a new application. The application, called "Info Tracker" is designed to help people keep track of personal information. There are other products like this on the market. The difference in this application is that it is extendable using plug-in modules to support different types of information that people want to record.

At the moment, the interfaces have been designed and the initial version of the main application has been written. One plug-in has been written in order to test the main application. When run, the application looks something like the following (note that you will not see any cards in the list when you run it since no file has been created yet):



In the left-hand pane, there is a tree of categories and, under that category; there are pieces of information (known as Info Cards in the application). In the example above, you can see three categories – Note, Credit Card and Password. If you double-click on one of the Info Cards, you will see the information associated with that card. For example, if you double-click on the Password card "Amazon", you will see the information that the user has stored about their account on Amazon as follows:



Information Cards
- Password
  - Amazon
  - My Website
- Note
  - My note
  - A new card
- Credit Card
  - My Card

Amazon

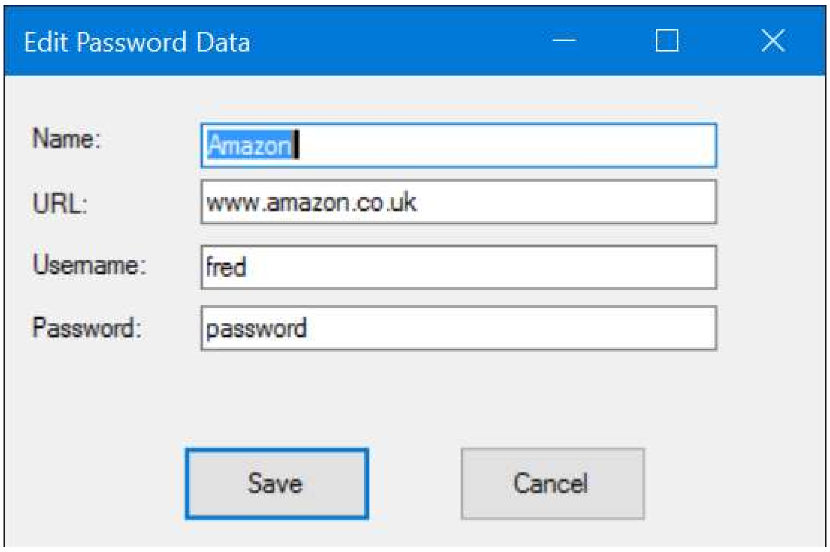
URL: www.amazon.co.uk

Username: fred

Password: password

Info Cards can either be displayed in the right-hand pane, or if the information could be too big to fit, it can be displayed in a separate window (if you look at a Note info card, you will see that it is displayed in a separate window). The decision about where to display the information is left to the developer of the plug-in module that handles that category of Info Card.

If you right-click on a card listed in the tree, a popup menu is displayed that allows you to Display, Edit or Delete the card. If you select Edit, a modal dialog box is displayed that enables you to edit the data associated with that card. For example, if you select Edit on the password card shown above, you would see the following dialog box displayed:



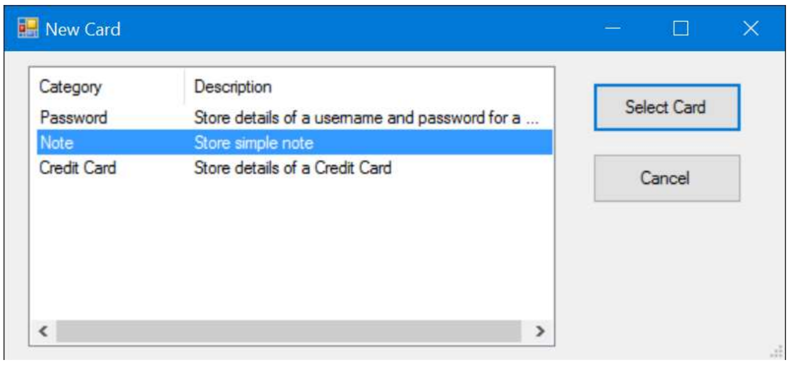
Name: Amazon

URL: www.amazon.co.uk

Username: fred

Password: password

If you right-click on a category in the tree, you can add a new card of that category. Or, if there are currently no cards stored of the type you want, you can select New Card from the Card menu and a dialog box will be displayed listing all of the categories supported, along with a description, which you can choose from. This is seen below:



Category	Description
Password	Store details of a username and password for a ...
Note	Store simple note
Credit Card	Store details of a Credit Card

Tasks

Read the following sections carefully. Although it may seem daunting at first, if you read through everything very carefully and approach this assignment methodically, you will find that there is not a huge amount of work needed. This assignment is intended to assess your understanding of the core concepts covered in this module, not to make you write a lot of code.

A lot of the application has already been written. The main user interface code exists (and is supplied to you) as well as the code that loads and saves Info Cards.

Although the example shown above shows Credit Card as one of the categories supported, this info card type has not yet been implemented. Only Password and Note are currently implemented. For this assignment you are to:

1. Write the plug-in module that will handle the Credit Card info card type.
2. Add an additional info card type of your choice to this plug-in module. For example, you could choose from frequent flyer numbers, bank account details, passport details or any other type of information you can imagine storing.
3. Add an additional card type to the system, this card must be a photo ID card which contains an image. Images may be stored and retrieved by file path. or by conversion to string data. Multiple file types may be supported.

The full specification of the interfaces you need to use and other information that will be helpful to you can be found in the appendix.

Submission

Your assignment must be uploaded to the submission point for this module on Course Resources by the due date and time

Please read the following submission requirements carefully. Failure to submit your assignment exactly as specified will result in a reduction in your grade and may even result in your assignment not being marked.

Your assignment must be submitted as a single zip file containing the following:

1. The full solution folder for your assignment. This will include the code supplied to you as well as the code you have written. Your submission should be written using Visual Studio 2019
2. A document in PDF format that contains the following:
   - A diary of your work on your assignment. Each entry in the diary should include:

    a. The date, start time and end time you worked on the assignment
    b. Details of the work you did during this period of working on the assignment
    c. Details of any problems you encountered and how you solved the problems.

- Screen shots of the application showing the Windows Forms that you have created as part of your plug-in module this assignment being used

The zip file you submit must use your student number as its name. For example, if your student number is 123456789, your zip file must be named 123456789.zip. The reason for this is that in order to download the zip files from Course Resources for marking as a mass download, we need to be able to distinguish your work from that of other students. Files that use any other form of naming convention will have a reduction in the grade of 20%.

Do not use any file format other than .zip.

Assessment Criteria

Your assignment will be marked according to the following criteria:

- How completely it meets the requirements
- The effectiveness and efficiency of the code
- The level of error handling performed in the code
- The style of the code
- The effectiveness of testing carried out and your approach to the assignment (your diary will be used for assessing this).

Grade	Description
80% or greater	As below but in addition: - Photo card implemented with multiple formats supported, all inputs are error checked thoroughly and a full test plan is provided. For a grade of 90% or greater**,** Images are stored and retrieved as a base 64 string.
70 –79%	Credit card, 2nd card and photocard storage function as expected with no problems evident with either the saving, loading, displaying or editing of the cards. The software is functional in line with the specification. Robust error checking is performed throughout the program and is successful in preventing errors in the integrity of the data. Error checking effectively prevents crashes within the program. Development diary documents several issues encountered during development and how these challenges were resolved effectively. Testing has been performed and documented to a high standard. UI design choices are good with no errors. Tab order is appropriate.
60 – 69%	Credit card and 2nd card storage function as expected with no problems evident with either the saving, loading, displaying or editing of the cards. The software is functional in line with the specification. Some attempt at error checking is found within the program; and is mostly successful, however edge case may reveal some weakness in the robustness of the solution. Edge cases identified are documented in the development diary and explained. Testing has been performed and documented to an appropriate standard. UI design choices are acceptable, with only minor errors, tab order is inappropriate.
50 – 59%	Credit card storage functions as expected with no problems evident with either the saving, loading, displaying or editing of the card. The software is functional in line with the specification. Some attempt at error checking is found within the program; and is mostly successful, however edge case may reveal some weakness in the robustness of the solution. Some testing has been performed to an acceptable standard; testing may be of questionable quality in places. UI design choices are poor, tab order is inappropriate.
40 – 49%	Credit card storage functions as expected; however, problems may exist with either the saving, loading, displaying or editing of the card in certain situations as to make usage of the software difficult. Some attempt at error checking is found within the program; however, it may not be fully successful in its implementation. Some testing has been documented in the development diary; however, this is largely ineffective in its approach.

35 – 39%	Credit card storage fails to function as expected, problems exist with either the saving, loading, displaying or editing of the card as such to prevent effective usage of the software. No evidence of error checking is found within the program. No evidence of testing is found in the development diary or the development diary is missing.
< 35%	Little of substance or merit submitted

Additional marks for all grades will be awarded based adherence to appropriate style templates.

Plagiarism/Collaboration Warning

All submissions will be checked to see if there is any evidence of collaboration with other students. If we suspect that you have submitted work that is not your own, we may ask you to attend a meeting to explain your work to us so that we can ensure that it is your own work. Work on this assignment on your own. Do not ask to see the work of other students and do not share your work with other students. The penalties for plagiarism or collaboration are severe.

Appendix

The Project Structure

A complete Visual Studio 2019 solution has been provided for you to work with in the zip file that accompanies this specification. You should unzip this, making sure you maintain the folder structure. Each plug-in module is a separate project that is compiled to form a dynamic-link library (DLL). These are searched for at run-time and loaded dynamically by the main application. The Visual Studio solution includes the following projects:

Assignment	This is the main application. All of the source code for this is provided for you to look at. You should not change any of the source code in this project.
AssignmentInterfaces	This contains the definitions of the interfaces you will need (IInfoCard and IInfoCardFactory) as well as other definitions used in the application. You must not add these to the InfoCards2 project – that project is already setup to refer to the AssignmentInterfaces project.
InfoCards2	This project is the one you must use to work on the assignment. The assignment properties and references are setup correctly to refer to the other projects in the solution where needed and to copy the built files into the correct place. You should add the classes you write for the assignment to this project. Do NOT add any files to any other project in this solution.
InfoCards	This is the class that loads and saves Info Cards. You might find this useful to look at since it will use methods that you write to perform this task.

(InfoCards1 was used to build the plug-in module that handles the Note and Password info cards. The source code for this is not available to you. However, the built version of this project (Infocards1.dll) has been placed in the Assignment folder and will automatically be copied to the correct place when you build the solution. Do not delete this file or change its properties.)

What You Need to Write

You will need to create the following classes in the InfoCards2 project:

- One class that implements the IInfoCardFactory interface
- Three classes that implement the IInfoCard interface. One of these classes will handle the Credit Card info card, another class will handle the additional info card you decide to implement. The third will handle the photo ID.
- For each info card, you will need two Windows Forms classes – one to display the details of the info card and the other to edit the info card.

The Interfaces

These interfaces are already defined in the AssignmentInterfaces project, so you should not add them to the InfoCards2 project. The interfaces are as follows:

IInfoCardFactory:

```
public interface IInfoCardFactory
{
    IInfoCard CreateNewInfoCard(string category);
    IInfoCard CreateInfoCard(string initialDetails);
    string[] CategoriesSupported { get; }
    string GetDescription(string category);
}
```

The following describes what each of the methods and properties need to do:

CreateNewInfoCard	This method is called when the user wants to create a new Info card of the specified category. It returns a reference to an object that implements IInfoCard. For example, to create a new Password Info Card, the method is called with the category set to “Password”
CreateInfoCard	This is called when the file that stores the Info Card data is loaded into memory. The parameter passed to this method is a string that uses the following format: `Category|Rest of details of card` The method should create a new card of the specified category, using the data in the string that follows the category name and | character to initialise the card. A reference to the new info card object should be returned. For examples of the string passed to this method, see the description of the GetDataAsString method in the IInfoCard interface.
CategoriesSupported	Returns a string array containing the categories supported by this class. For example, if the class supports the categories “Credit Card” and “Account”, it should return a string array containing the strings “Credit Card” and “Account”.
GetDescription	Returns a string containing a description for the specified category. The description is used in the New Card dialog box.

IInfoCard:

```
public interface IInfoCard
{
    string Name { get; set; }
    string Category { get; }
    string GetDataAsString();
    void DisplayData(Panel displayPanel);
    void CloseDisplay();
    bool EditData();
}
```

The following describes what each of the methods and properties need to do:

Name	Used to set and get the name of the Info Card. The name is used to identify the Info Card in the tree view
Category	The category of this info card. This should match one of the category values referenced in the factory class. This is used to determine where the card is to be positioned in the tree view

GetDataAsString	Returns the data stored in the info card in a form that can be written to a file. The format used is:
	
	`Category|Name|<other properties separated by |>`
	
	For example, for a password info card, the string returned might be:
	
	`Password|Amazon|www.amazon.com|joebloggs|mypassword`
	
	Note that this string is the string passed to the CreateInfoCard method in the class that implements IInfoCardFactory when the data is loaded from the file again. Apart from the category and the first "|" character, the rest of this information is only used inside the class that handles the info card

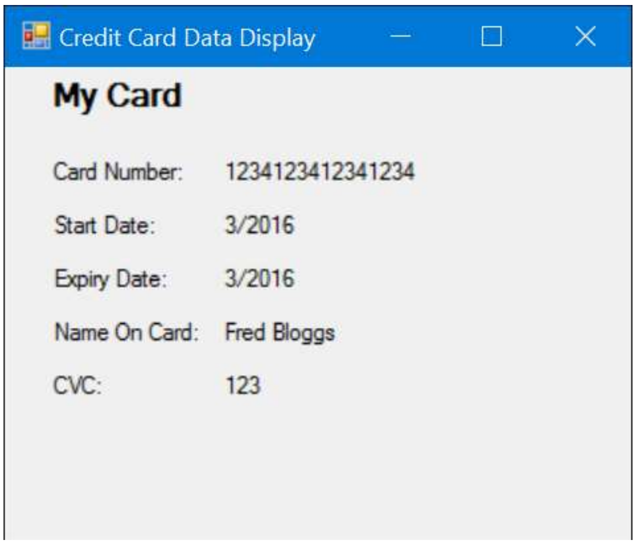
DisplayData	This method displays a windows form that displays the data in the info card. The parameter passed to DisplayData is a reference to the panel to the right of the tree in the main application. If the data in the info card is small enough to fit inside this panel, the form can be displayed on that panel. This is not a requirement, but extra credit will be given if you can figure out how to do this. The form must <u>not</u> be closed by this method. It should only be closed by the CloseDisplay method.
CloseDisplay	This method should close the form displayed by the DisplayData method.

EditData	This method displays a modal dialog box that allows the user to edit the data in the info card. Ideally, it should ensure that all fields in the info card contain data (i.e. there are no blank fields). If there are blanks, it is up to you to ensure that your class handles this sensibly when the data is retrieved using GetDataAsString and when a new info card is created. EditData returns a bool that must be set to true if the data in the info card has been changed, false if the data has not been changed.

Suggestions and Hints

The following are some suggestions and hints that you might find useful:

1. Do not try to implement all the info cards at once. Do the Credit Card info card first and only once you have that working correctly should you attempt the other cards.
2. Do not try to write all of the code at once. Break the development into stages and think carefully about what needs to be implemented at each stage so that it can be tested before proceeding to the next stage. For example, you do not need to have all of the methods in the class that implements IInfoCard written before you can test whether the plug-in module can be loaded by the main application.
3. Study the code in the main application to see how the IInfoCardFactory and IInfoCard methods and properties are used. You can debug the application to see how the methods in the classes are used (although you cannot see how they are implemented). There will be code in the main application that you are not familiar with, but don't let that put you off. You can search through the code to see where the methods are called from.
4. If you are not sure what the credit card info cards forms should look like and what data should be stored, here are some suggestions (you are free to add additional information if you wish). Also feel free to use other controls if you feel they are more appropriate.



My Card

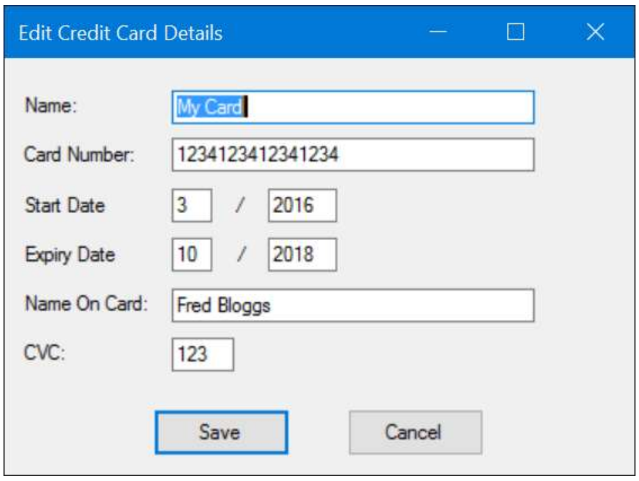
Card Number: 1234123412341234

Start Date: 3/2016

Expiry Date: 3/2016

Name On Card: Fred Bloggs

CVC: 123


Edit Credit Card Details

Name:	My Card		
Card Number:	1234123412341234		
Start Date	3	/	2016
Expiry Date	10	/	2018
Name On Card:	Fred Bloggs		
CVC:	123		