# Jack Hoyle: Comp 4 Project

Myler – Customizable Flyer app

GREENHEAD COLLEGE

2012/2013

Authored by: Jack Hoyle

# Contents

# Introduction

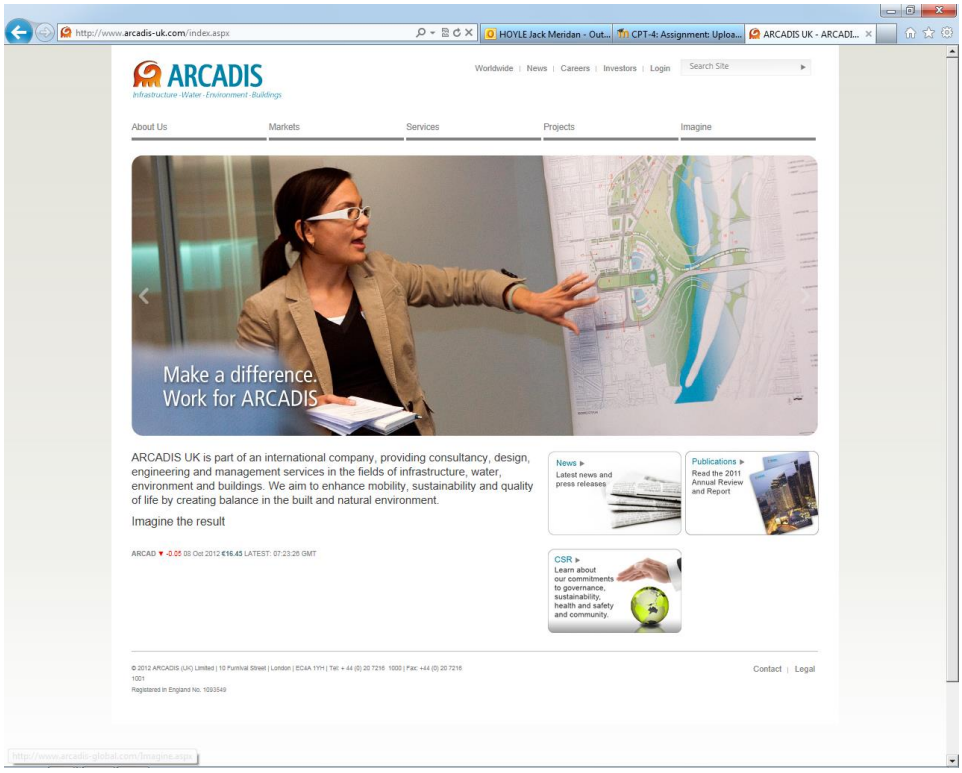
In the business world flyers and business cards are very important things for advertising your services or company however the information gained from one is very limited, for example on a business card there is only a small space and there is a limit to what can be learnt from it, for example the name, contact details and the general services provided by the individual/company, this means that face to face contact is required to provide in depth information which is often crucial to a decision.

To provide an easy to use and accessible solution I am going to create an app for a windows phone and a program for a pc which will give the user the opportunity to create a personalised online flyer which will be accessible by a unique id code, meaning that the user can create a flyer and send the id code to a potential customer and they can view the flyer without the face to face contact and be provided with the extra information like case studies etc.

# Analysis

## Background and Identification of the Problem

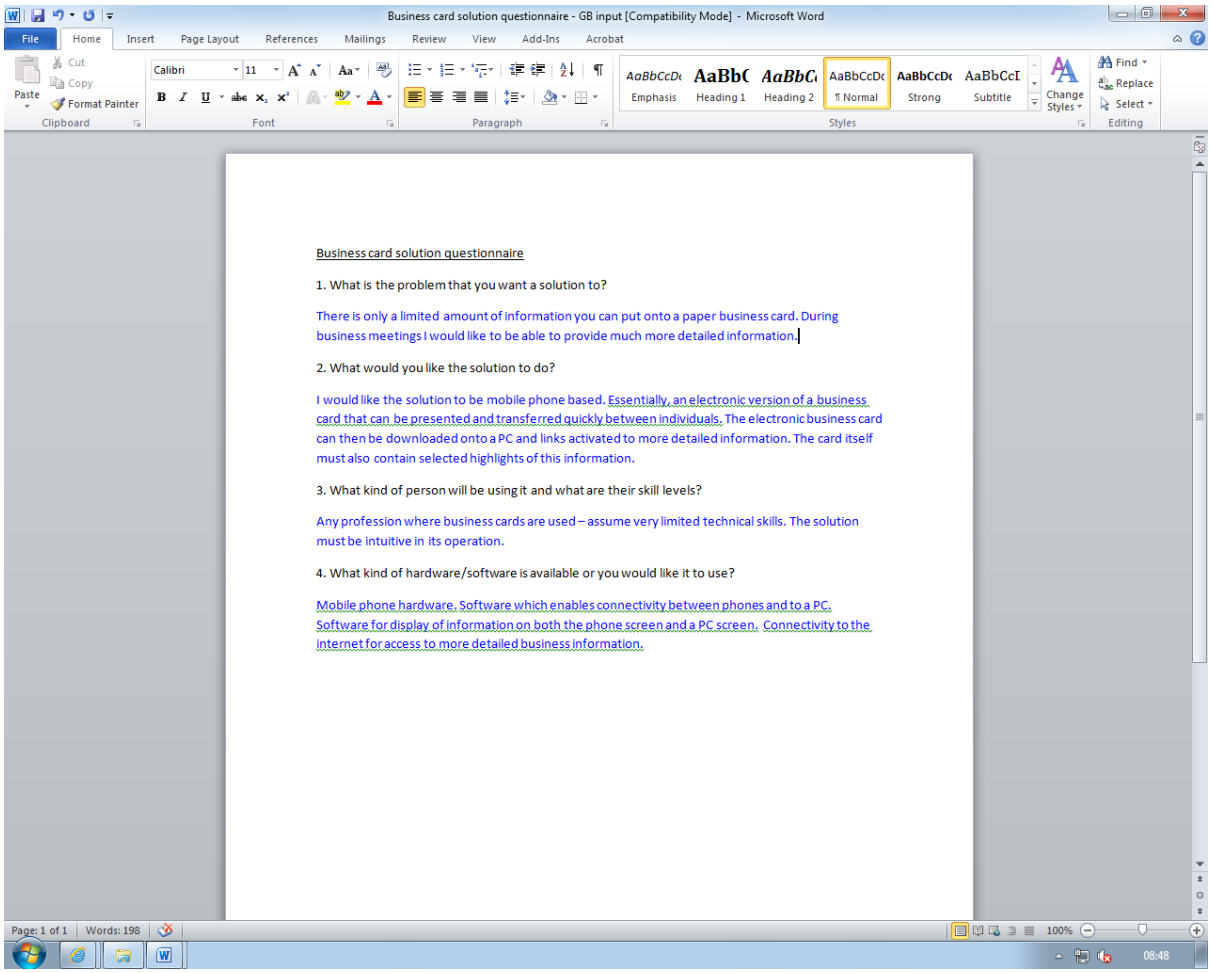
The client for my project is my uncle. He is the executive director of a ARCADIS UK, they provide consultancy, design, engineering and management services in the fields of infrastructure, water, environment and buildings, specialising in project and cost management, environmental consultancy, safety and risk management, contaminated land remediation, strategic consultancy.



Currently when companies go to meetings with potential customers or employees the only external information they receive would be a business card and because they are so small there is literately no information that can provide an educated decision as to whether to employ the person/company requiring time and effort to research the provider of the business card to make the decision.

## The interview

In the interview with my uncle he expressed that he wanted a solution to the limit of information that can be shown on a business card. His concern was that when he meets with potential clients or potential employees that the exchange of information between them isn't sufficient. His idea of a solution would be an online flyer that can be filled indefinitely with information about things which the company/individual have been involved in like case study's and research etc. He also suggested that he wanted it to be portable meaning rather than creating a website to accommodate the flyers, it would be a phone app that they are viewed on meaning that it would be extremely portable.



**Completed form**
Jdu|#Edp irug#^Jdu|1Edp irugC DufdglvOXN1frp '#
**Sent:** 5;#Whswhp ehu#5345#53=46
**To:** KR\OH#Mdfn#Phulgdg
**Attachments:** Exvlqhvv#fdug#vrowdrq#tx£41grf{ +45#NE,

Hows the attached sound?

Gary Bamford - Executive Director, Technical Knowledge and Innovation

ARCADIS | Portland Tower | Portland Street | Manchester | M1 3AH | United Kingdom
T. + 44 (0) 161 934 3300 | M. + 44 (0) 7747 100 302 | F. + 44 (0) 161 934 3301
www.arcadis-uk.com
Follow on Twitter: @grandwizz

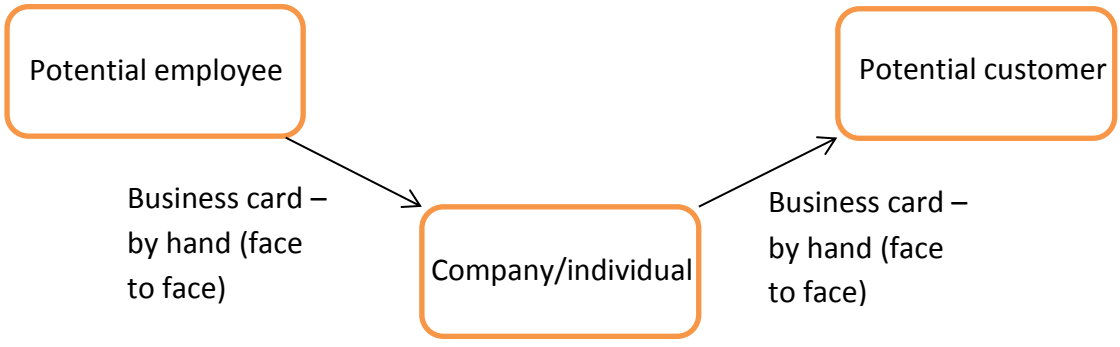
-----Original Message-----
From: HOYLE Jack Meridan [mailto:s16515@greenhead.ac.uk]
Sent: Thu 27/09/2012 11:06
To: Gary Bamford
Subject: computing project

Hi its jack. I've got a questionnaire for my project, could you fill it in and send it back to me? cheers.

## Current system

The current system in place is a physical business card which is handed between users requiring face to face contact which takes up time and money just for someone to consider your services.

## Information flow diagram (old system)



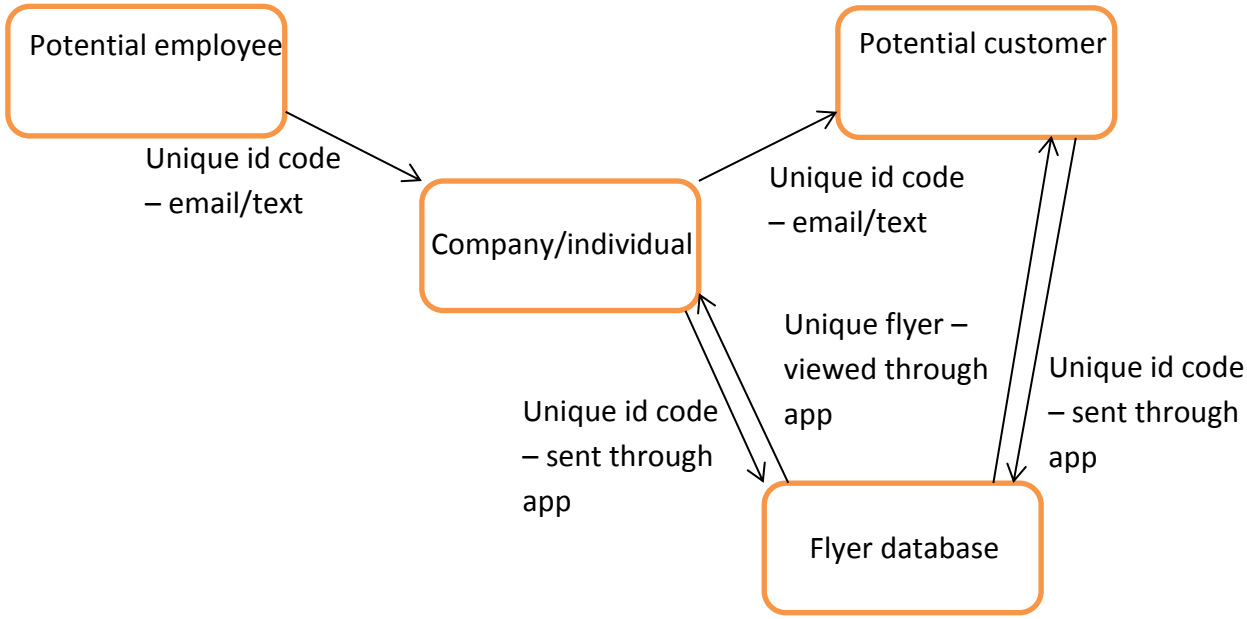
## Prospective users

The prospective users for the app would be anyone or business that requires an exchange of information between the individual/business and the client, meaning that there would be a varying degree of skill levels thus my app will have to be simple to use yet be customisable depending on the user.

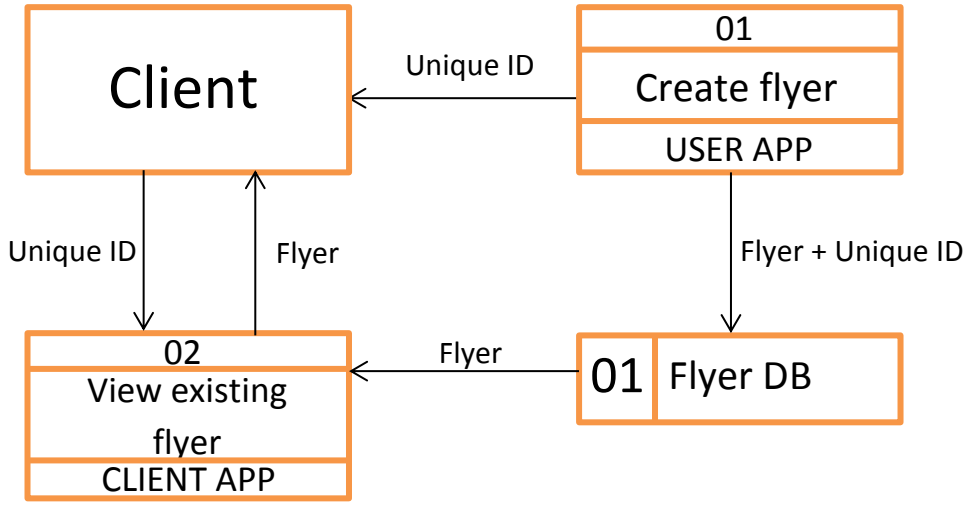
## User needs

- The new system needs to be easy to use.
- The new system should make it easy and quick to view a flyer.
- The new system should allow the information to be added to the flyer quickly and easily.
- The new system needs to look professional.

## Information flow diagram (new system)

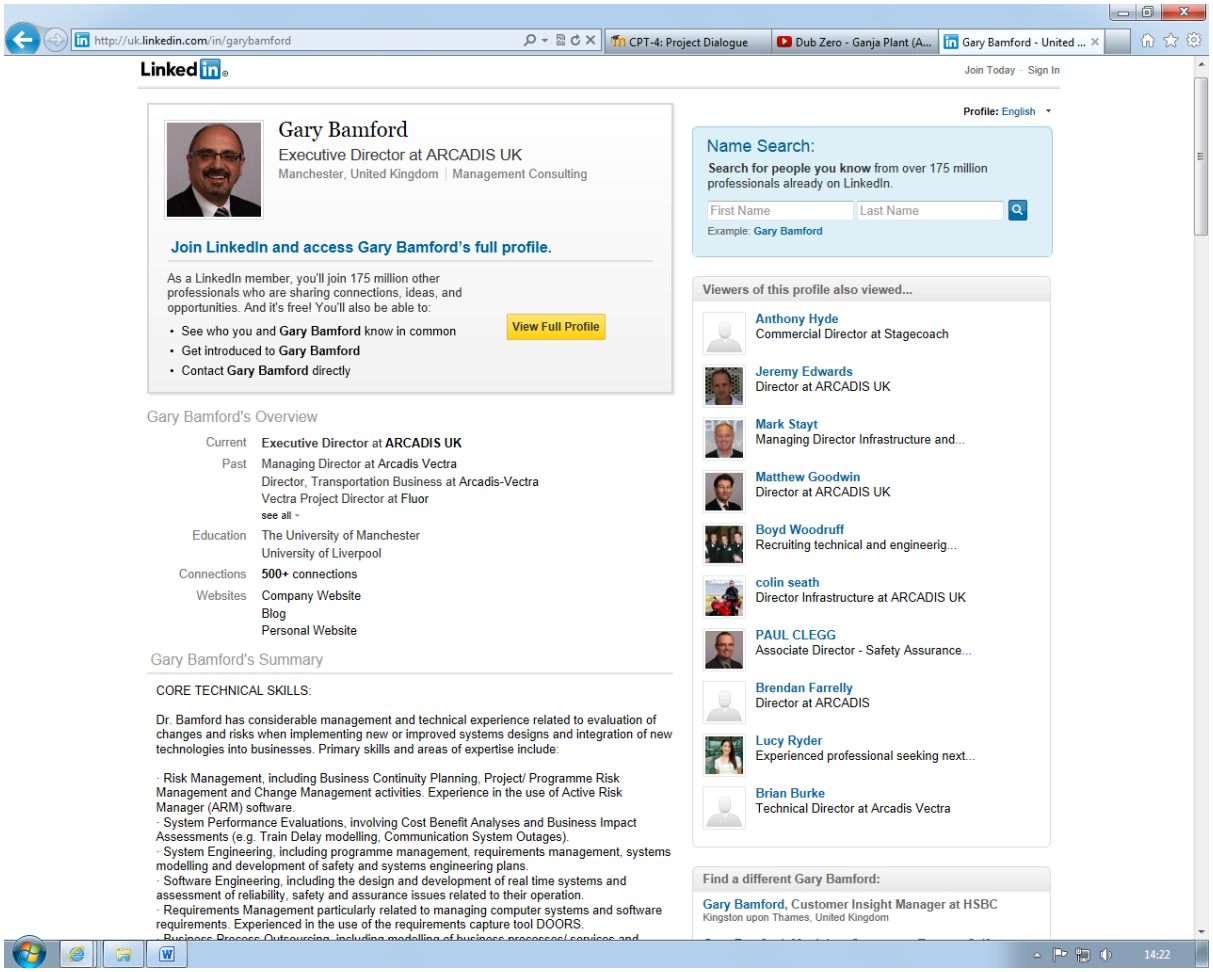



## Data flow diagram (new system)



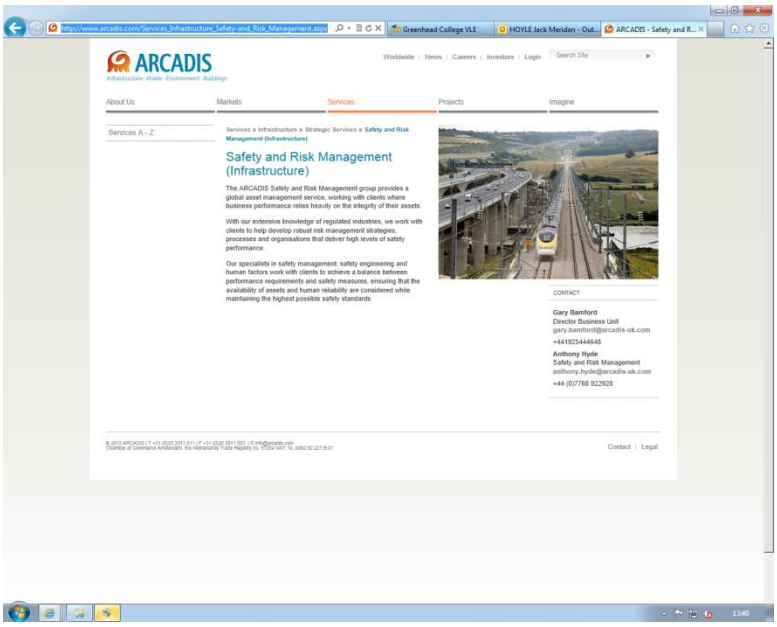
## Investigation of similar systems

From my investigation I have found that there is nothing similar to my project in terms of the interface and online storage. However in terms of the kind of information being put into the flyers the website http://uk.linkedin.com/ is quite similar as it incorporates information that is important at a corporate or business level as it has information relating to research and projects that the individual has been involved in, their skills and publications. Although the amount of information is a lot and is in my opinion too much for the app I am creating but it is probably the closest thing to my idea even though it could be seen as a businessman's social network.



Linked in also have a windows phone app. However in my view it can be seen as pretty much the same as Facebook as it is similar to a social network in its profile centred view. Whereas my solution would provide specific information related to what the flyer is created for not just loads of information. However the general look and feel of the app may be something that I want to replicate.

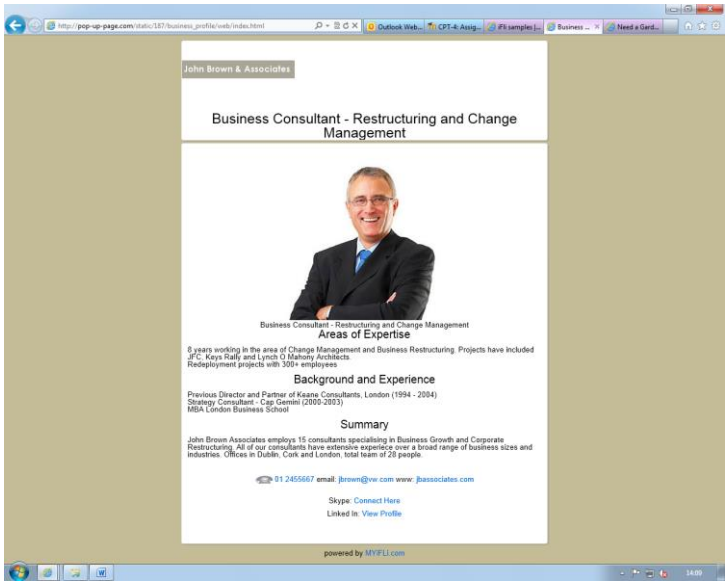

In terms of colour schemes/themes Gary gave me a couple of examples of what he wanted it to look like. One thing he sent me was the webpage for his area of Arcadis-UK and as you can see there is a blue and orange colour scheme as is similar to the logo.



He also gave me a couple of flyers produced by Arcadis-UK they both have the company logo on and have a distinct blue theme to it as is shown in the title bar of the flyer.


Looking at all of these examples I think I am going to go for a grey, blue and orange theme to my app as it will look like the Arcadis-UK logo and the flyers. This will mean that I am using a colour scheme that has been used by a high end company who have probably had some input about it from a professional.

Upon research I have found that there are a few website which provide “mobile flyers” however these are mainly websites that are adapted for mobile phones. As shown below these are usually just thinner and scroll down. However these show an example of similar ideas just on a website version but it is something for me to consider as they are a simpler version persay of what I am creating.



## Objectives for new system

My solution should:

1. Provide an easy to use and accessible interface for users of any skill level by:
    a. Limiting the amount of buttons in the interface as to not confuse people with an unnecessary amount of options.
    b. Providing clear options to the user in buttons.

   Success will be measured by whether the users understand how to navigate through the interface without getting confused.
2. Be able to create, save, edit and view flyers quickly and easily by:
    a. Providing instructions to the user as to how to create, save, edit and view a flyer.
    b. Having a template ready for users to add information and not have to format.
    c. Having clear buttons for saving etc.

   Success will be measured by whether the users can create a flyer without having outside help and/or getting confused.
3. Store the flyers in a database and make it easy to access existing flyers by:
    a. Providing a box on the main interface for the Flyer ID to be added and then viewed.

   Success will be measured by whether the flyers are accessed and loaded quickly.
4. Make the flyers easily customisable in terms of the amount of information added by:
    a. Providing buttons to add paragraphs, pictures and such to the set template.

   Success will be measured by whether each separate flyer is different and don't all look the same.
5. Make it easy to manage their flyers and such by:
    a. Providing a page where their existing flyers can be viewed and monitored

   Success will be measured by whether all the users' flyers are found and shown on the page quickly without any errors.
6. Overall success will be measured by:
    a. A questionnaire being given to a mix of people of different skill levels
    b. Whether or not the users can navigate the interface and create flyers.

## Feasibility of potential solutions

One possible solution for my project could be a website based application showing people with a website or program where you can create the flyers online and edit them from a pc, this would provide people with a larger and most likely clearer interface for the creation of the flyers and with the extra size would mean you could add more options however it would also mean that it could become more confusing to use and it would mean that it is not portable as you would have to be at a pc or carry a laptop around with you. In terms of programming I would most likely use an event driven/object orientated solution using either html for a website based version or a Lazarus based form for a program.

Another possible solution is a phone app, for this the interface would be much smaller and would mean there would be less options for customisability but it would also mean that it would be easier to use and it would be extremely portable as people always carry their phones around with them, so they would always have an access to the app. To actually program this solution I would use visual basics and c#, however I would have to learn how to use this and this also means it will be made for a windows phone. However I could create it for any smart phone but I have chosen a windows phone because they are fairly powerful and the programing side of it is more accessible in my point of view than IOS and android as there is a section for creating them within visual basics and there is a lot of support online if I ever need it.

## Justification of chosen solution

I have chosen to merge the two of my ideas by having the phone app as a delivery system for viewing the flyers and a program on a pc for creating and editing the flyers. This will provide a bridge between portability and customisation. Using a program on a pc for the creation of the flyer will mean that the interface will be much bigger and can hold more customisation options and using a phone app for just the delivery system will mean that you can view them on the move and negates the limit of options within it and if I had the creation on the phone app it would be more difficult to use than pc program. The pc program will also act as a management system for the flyers meaning they will be able to view their flyer and how many people have viewed them and their details.

In conclusion I will be creating a Pascal based form for the flyer creation and management meaning that the flyers will be able to be made quickly, easily and look acceptable without the problem of a small interface had I chosen to make it on the windows phone alone. Using the phone as merely a deliverance system will mean that it will be portable as are business cards but it will mean that the flyer creation will be more accessible because not everyone has a windows phone and many people would not be able to use one fully as they are quite complicated to use but everyone has access to a computer and many more people are able to use them with a certain degree of skill. I will be using a database to store the users details and flyers as well as the information about the client however the client will not be require

to log  in to view the flyers once they have been sent the unique id code only the user will be required to log in to view, edit and create their flyers.

# Design

## Purpose

The purpose of the software is to replace business cards with online flyers as to eliminate the limit of information provided between companies or individuals when exchanging them and provide them with more information about the flyers owner.

## Environment

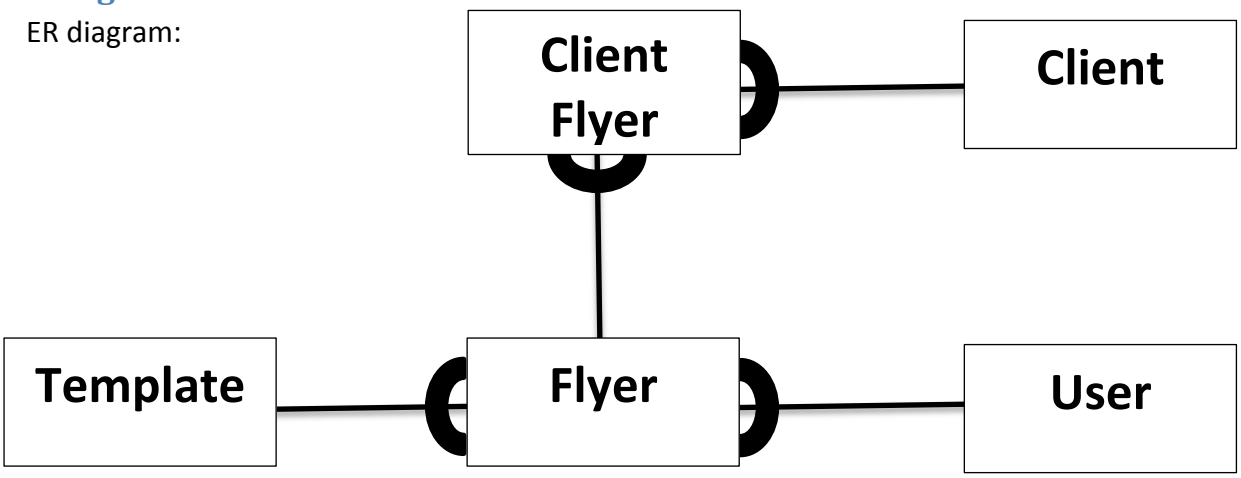
The environment for my solution would range from big companies to small business and individuals therefore it has to be usable and suitable for a professional workplace and lower end environments.

## Users

There will be a wide range of users with varying skill levels from business men to professionals to new companies with skills going from beginner to expert. This means that my solution must be easy to pick up for beginners but also for people who have a lot of experience with computers and such.

## Using a database

ER diagram:




## Storage media

The program will be used on a pc but the flyer and user details will be stored online in the database. The main body of the program will be stored in a folder on the pc with all the shell data within it but the users data will be stored in an online database and will be accessed and used in the program when the users login meaning that when another user log in the will get their data and flyers not the previous users.

## Security

The data will be stored online in a database however to access the main program you will be required to log in meaning that only the user with the correct login details can access their own flyers. I will also not be storing any extremely sensitive data like finance data only things like their name and address.

## Validation

In terms of validation I will only be using it on the inputs for a new account (forename, surname) to not allow integers to be entered everything else that is user entered can be allowed to have integers in them.

## Test strategy

I will test each form to make sure it works as it should and does what it is meant to do. I will apply these tests separately for each form when they have been completed and any changes will be made before moving onto the next form.

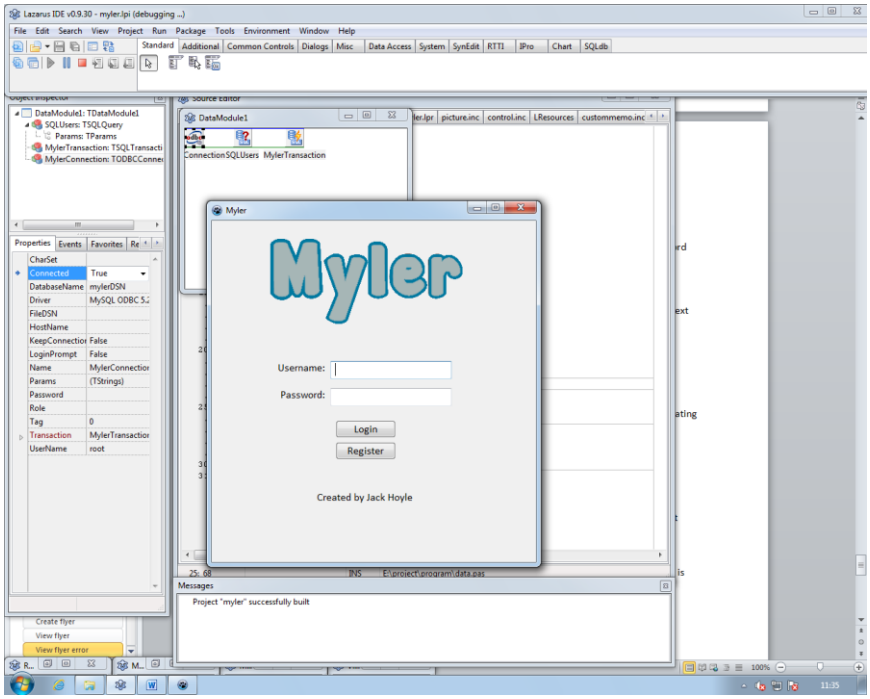
| Test ID | Purpose of test | Data/Actions used | Type of test data | Expected results |
|---|---|---|---|---|
| **Valid login** | To check if the login works when provided with valid inputs | txtUsername<br>txtPassword | normal | To login in with no error message and proceed to main page |
| **Non valid login** | To check if the program recognizes if the data entered is incorrect | txtUsername<br>txtPassword | normal | To show an error message that there is a problem with the login data |
| **Register** | To check if the data entered is added to the database correctly | txtUsername<br>txtPassword<br>txtForname<br>txtSurname<br>txtEmail<br>txtAddress<br>txtBusinessname | normal | The data should be entered into the database with no errors |
| **Main navigation** | To check if the main page navigation works properly | | normal | The buttons will direct the user to the correct places |
| **Main flyers** | To check if the main page shows the users existing flyers | | normal | The main page should show the users existing flyers |
| **Create Flyer** | To check if the form works properly and | | normal | The inputs should be put in correctly and when saved it should |

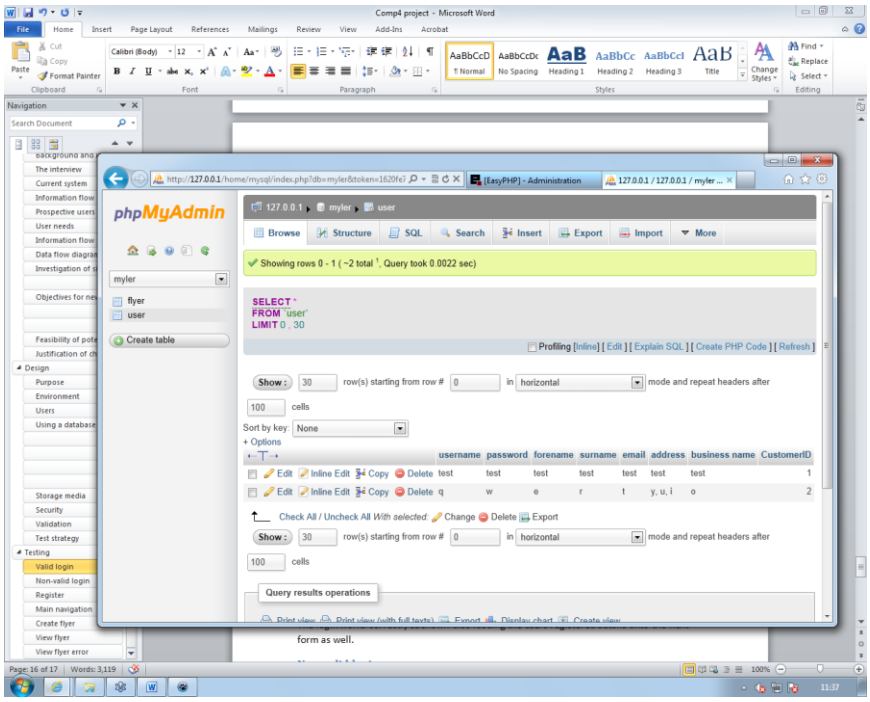
| | | | | |
|---|---|---|---|---|
| | saves properly | | | show on the main page as well |
| **View flyer** | To check if when the flyer name is entered the correct flyer is loaded | Name in the flyer table in the myler database | normal | The correct flyer should be loaded |
| **View flyer error** | To check if the correct error message is shown when an incorrect flyer name is entered | Name in database as stated above | normal | A message saying that no flyer has been found should be shown |
| | | | | |

# Testing

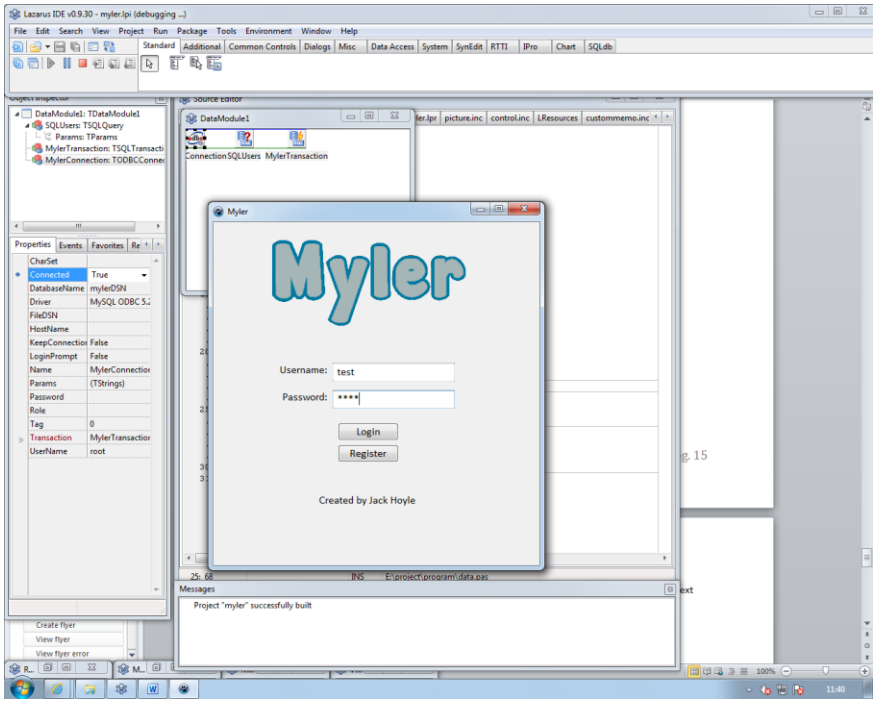
## Valid login

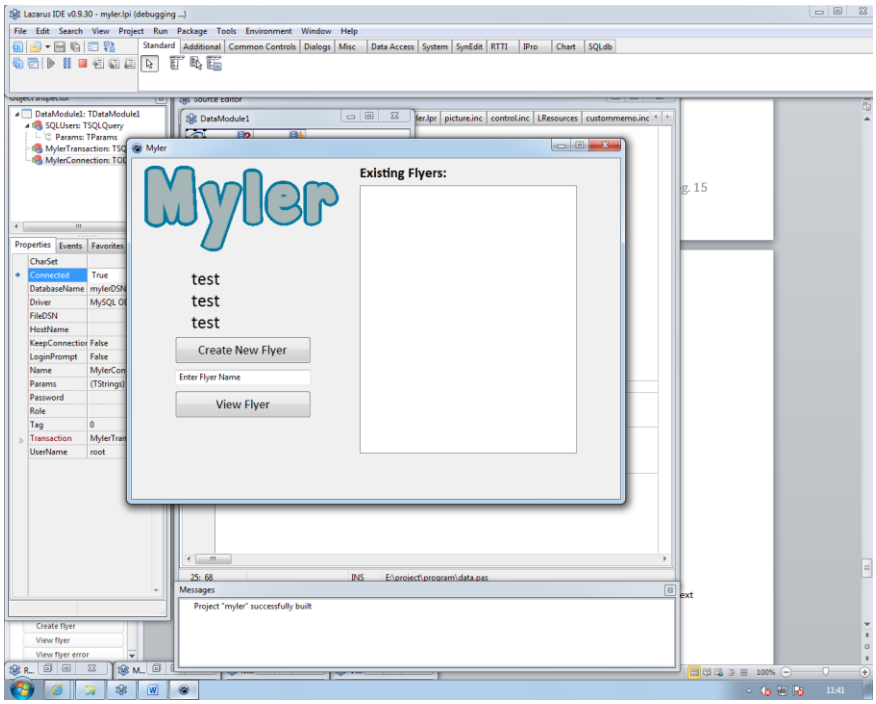
This test is to check if the program logs in correctly when a valid username and password are entered from the user table in the database.




Using the test details from the database I will login using these valid details.

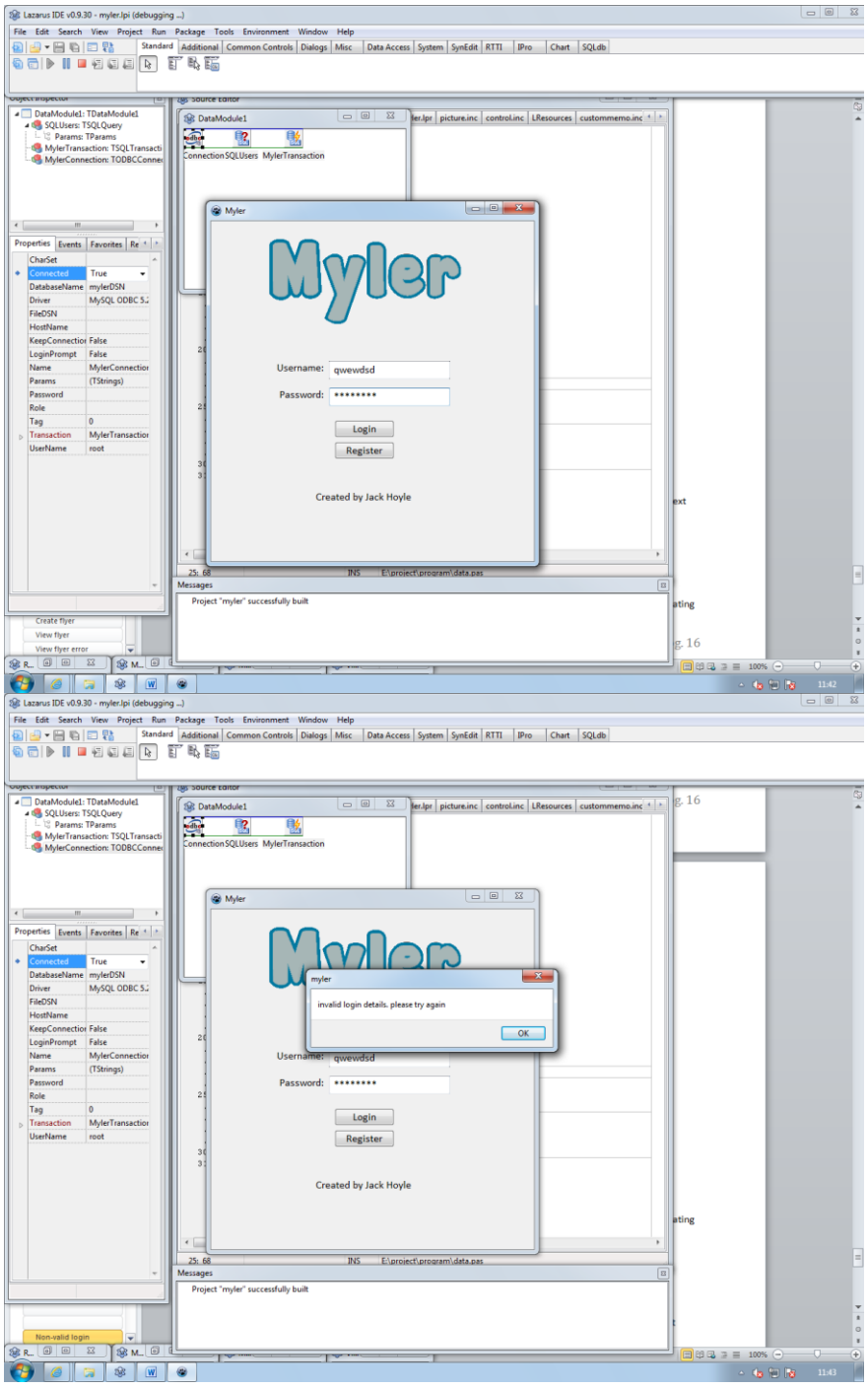



The login works correctly as shown also loading the users registered details onto the next form as well.

## Non-valid login

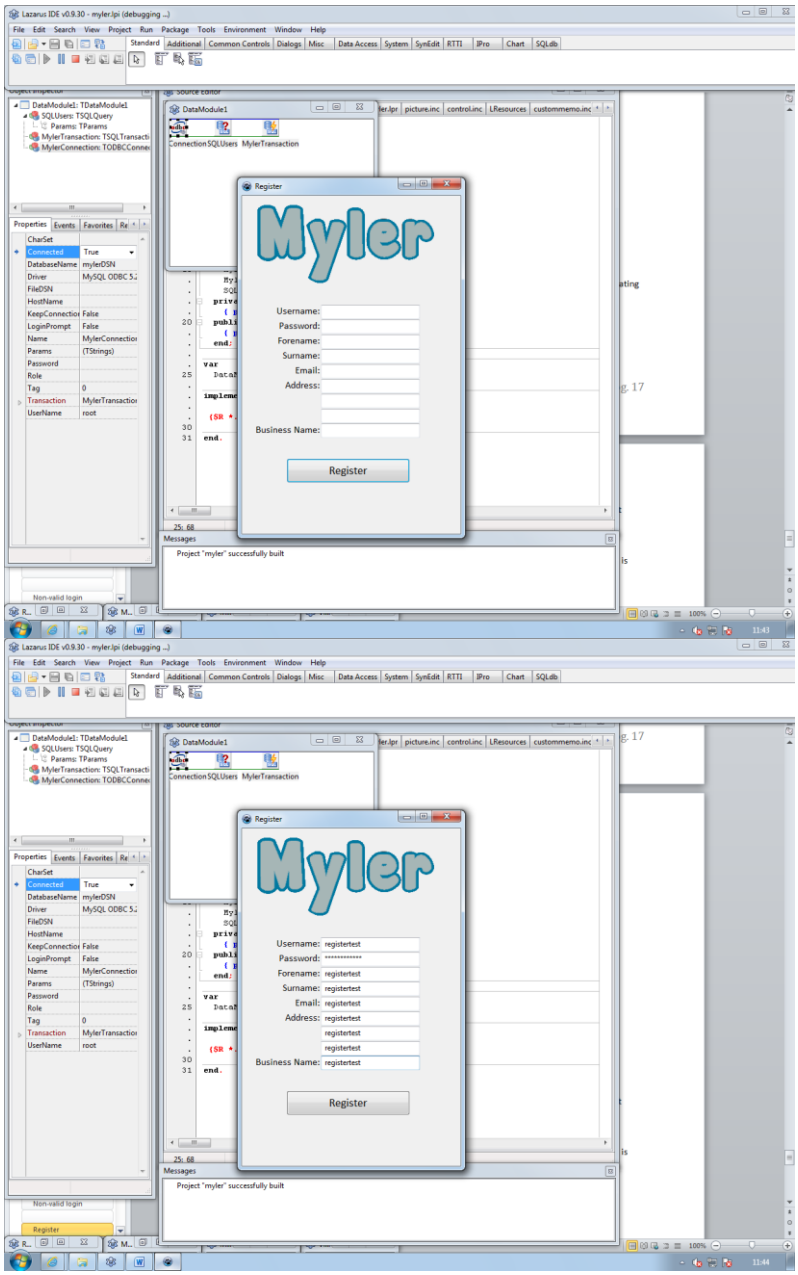
This test is to check if an error message is shown if invalid user details are inputted.

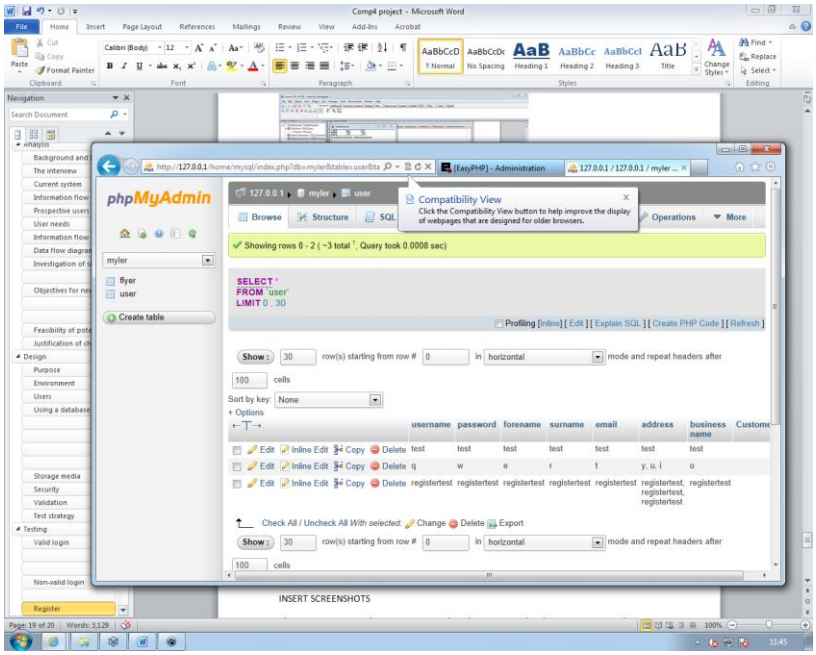



As shown the program shows an error message if incorrect user details are inputted stating “invalid login details. please try again”.

## Register

This test is to check if the users details are added to the database correctly.

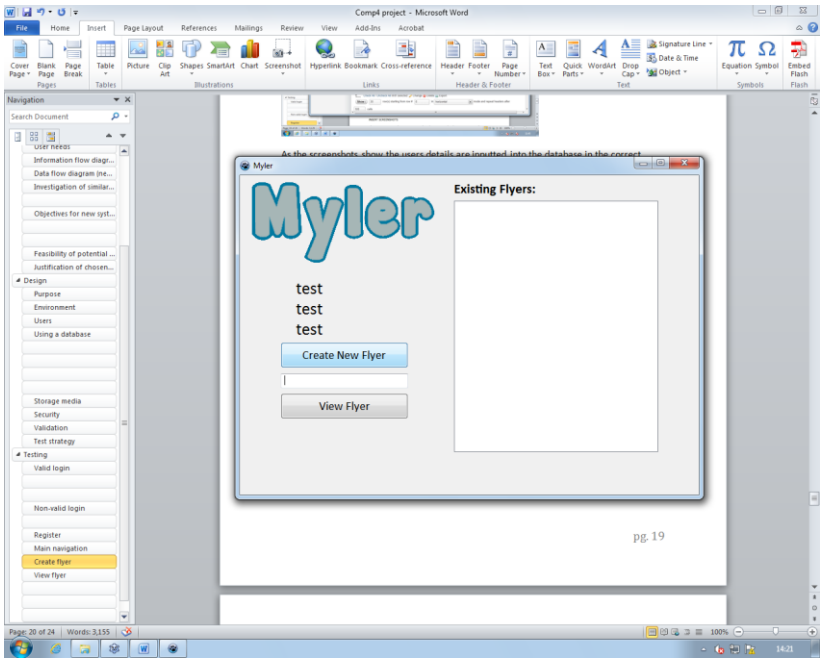




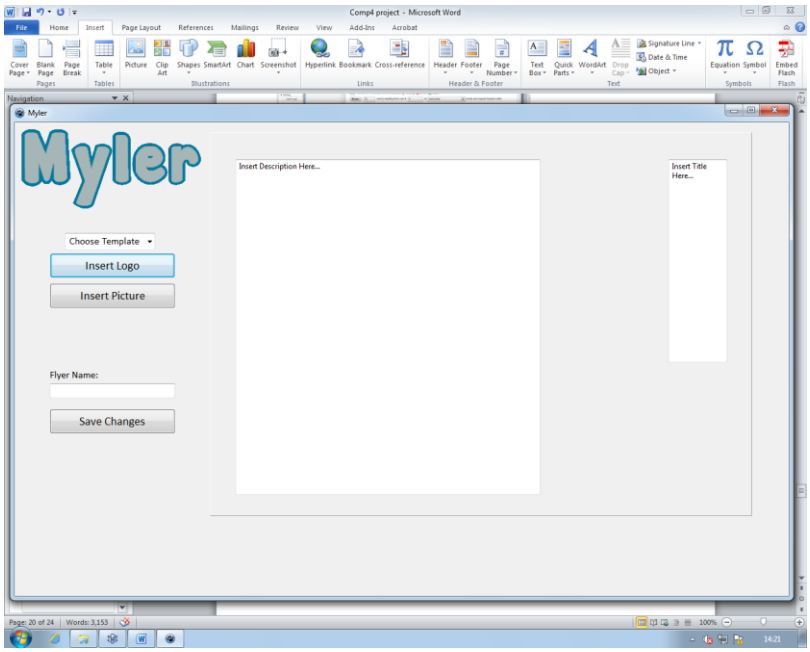

As the screenshots show the users details are inputted into the database in the correct fields within the database.

## Main navigation

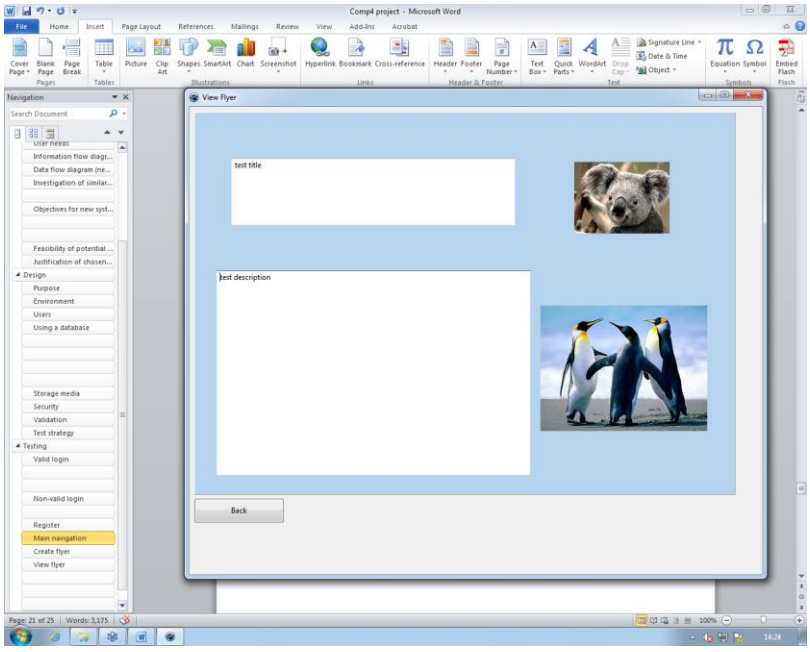
This test is to check if the main page goes to the correct form when the correct button is pressed.



When the create new flyer button is pressed.



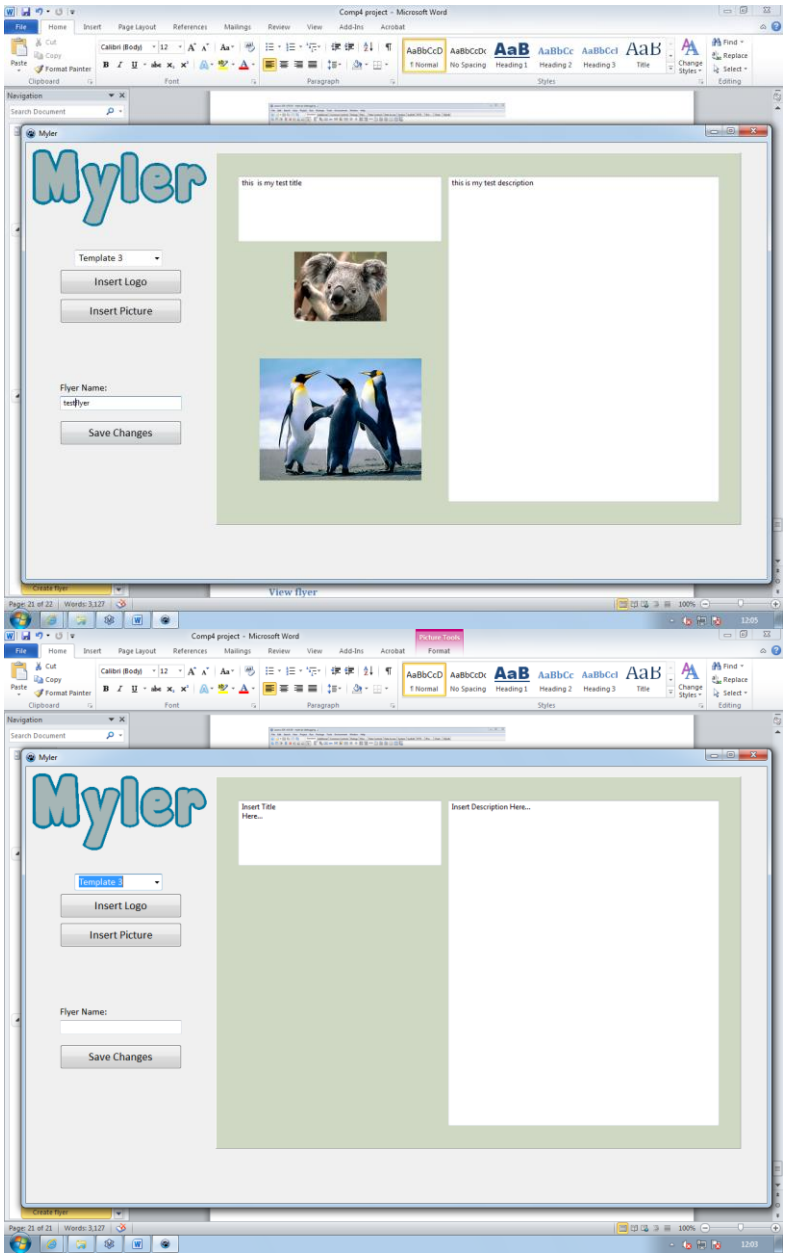

When a valid flyer name is inputted and the view flyer button is pressed.

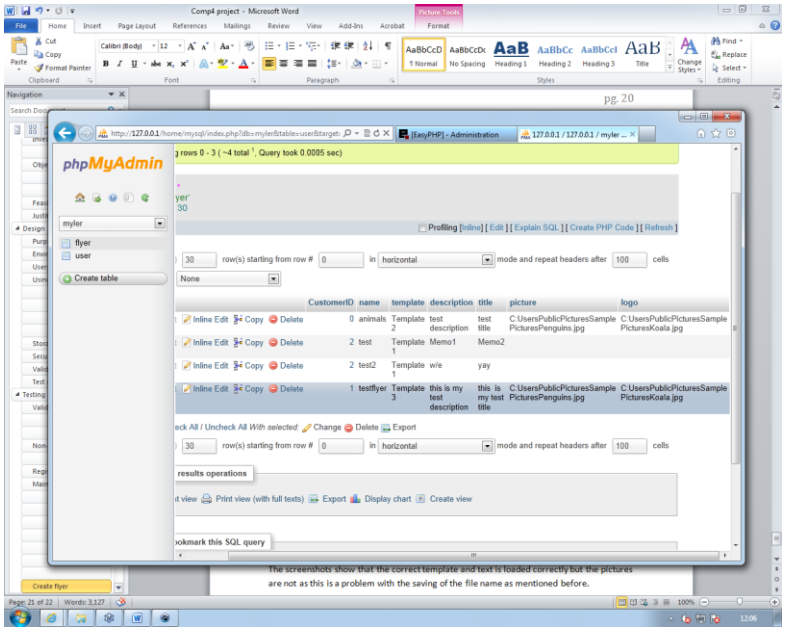



The program goes to the correct pages when the corresponding buttons are pressed.

## Create flyer

This test is to check if the form works correctly when the different inputs are changed to the panel as to simulate the flyer that is being created for example when the drop down list is changed the panel should change to correspond with the correct template.



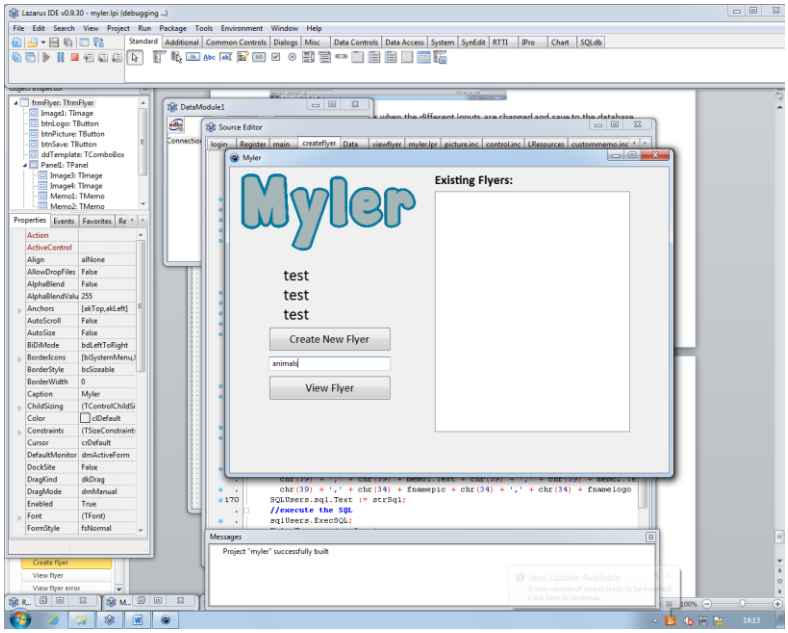



The form correctly changes when the different inputs are changed and save to the database correctly however the filenames (filepath) are saved without the "/" in them this will affect the loading of the pictures as the filenames will be incorrect.

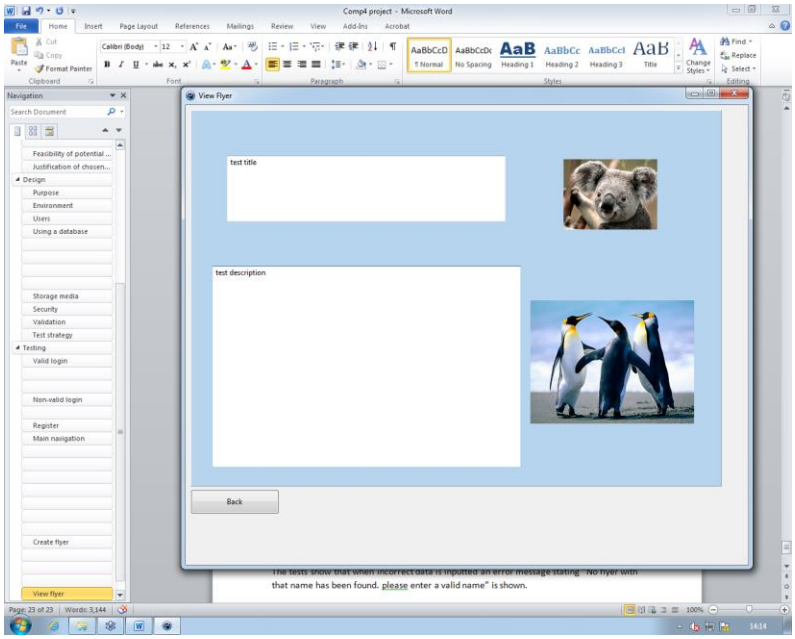
After searching for a solution on the internet extensively I wasn't able to find a solution.

## View flyer

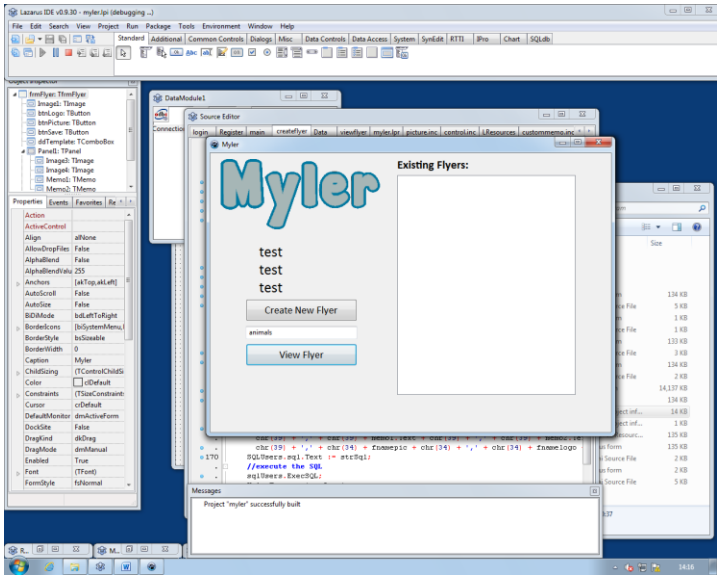
This test is to check if the correct flyer is loaded when the correct flyer name is inputted.




You input the flyer name and click view flyer.



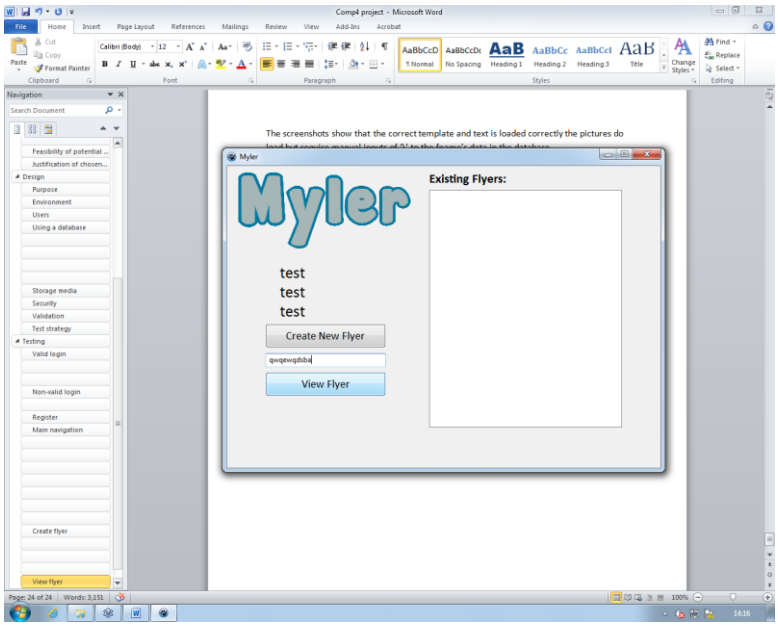

The screenshots show that the correct template and text is loaded correctly the pictures do load but require manual inputs of '\' to the fname's data in the database.



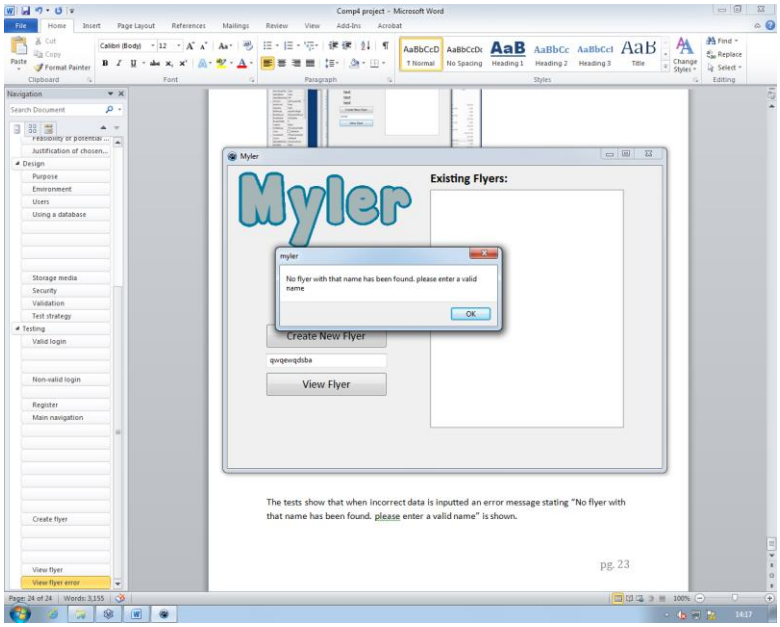

The back button also goes back to the correct form.

## View flyer error

This test is to check if when an incorrect flyer name is inputted an error message is shown.




A non-valid flyer name being inputted.




The tests show that when incorrect data is inputted an error message stating “No flyer with that name has been found. please enter a valid name” is shown.

# System Maintenance

The purpose of the program is to create flyers which contain information upon a specific thing as a way to produce items which have a greater value of information than a business card.

## Installation Details

The program will run from the folder that is given however the odbc drivers are required to be installed before you can access the database.

## Login

```
unit login;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, sqldb, odbcconn, FileUtil, Forms, Controls, Graphics,
  Dialogs, ExtCtrls, StdCtrls, Data;

type

  { TfrmLogin }

  TfrmLogin = class(TForm)
    btnLogin: TButton;
    btnRegister: TButton;
    Label3: TLabel;
    txtUsername: TEdit;
    txtPassword: TEdit;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    procedure fillMain;
    procedure btnLoginClick(Sender: TObject);
    procedure btnRegisterClick(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;
```

```
var
  frmLogin: TfrmLogin;

implementation

{$R *.lfm}

uses Register, main, createflyer;

procedure TfrmLogin.fillMain;
// this procedure accesses the database and loads the users details into frmMain
var
  strSQL: string;
begin
  with DataModule1 do
  begin
    //generate SQL
    strSQL := 'SELECT * FROM user WHERE username=';
    strSQL := strSQL + chr(39) + txtUsername.Text + chr(39);

    //clear any residual SQL
    if sqlUsers.Active then
      sqlUsers.Close;

    //set the SQL text
    sqlUsers.SQL.Text := strSQL;

    //execute the SQL
    sqlUsers.Open;

    //check if a record has been found
    if sqlUsers.RecordCount <> 0 then
    frmMain.lblForename.Caption := sqlUsers.FieldByName('forename').AsString;

    frmMain.lblSurname.Caption := sqlUsers.FieldByName('surname').AsString;
    frmMain.lblEmail.Caption := sqlUsers.FieldByName('Email').AsString;
  end;
end;

procedure TfrmLogin.btnLoginClick(Sender: TObject);
```

```
// this procedure logs the user in
var
  strSQL: string;
begin
  with DataModule1 do
  begin
    //generate SQL
    strSQL := 'SELECT username, password, customerID FROM user WHERE username=';
    strSQL := strSQL + chr(39) + txtUsername.Text + chr(39);

    //clear any residual SQL
    if sqlUsers.Active then
      sqlUsers.Close;

    //set the SQL text
    sqlUsers.SQL.Text := strSQL;

    //execute the SQL
    sqlUsers.Open;

    //check if a record has been found
    if sqlUsers.RecordCount = 0 then
      ShowMessage('invalid login details. please try again')
    else
    begin
      if txtPassword.Text = sqlUsers.FieldByName('password').AsString then
      begin
        frmflyer.customerID:= sqlUsers.FieldByName('customerID').AsInteger;
        frmMain.Show; //shows frmMain
        fillmain; //runs the fillmain procedure to fill the captions on the main form
        frmLogin.hide; //hides current form
      end
      else
        ShowMessage('invalid login details. please try again');
    end;

  end;
end;

procedure TfrmLogin.btnRegisterClick(Sender: TObject);
begin
```

```
  frmRegister.Show; // Shows the FrmRegister
  frmLogin.hide; // Makes FrmLogin invisible
end;

end.
```

This part of the program is for the login where the procedure btnLoginClick is used to acquire the user's login details from the inputs on the form and check if they are valid login details and if so log the user in. The procedure fillMain is used to fill the users details onto the main form so they show up.

## Register

```
unit Register;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,
  StdCtrls, Data;

type

  { TfrmRegister }

  TfrmRegister = class(TForm)
    btnRegisterSubmit: TButton;
    txtUsernameReg: TEdit;
    txtPasswordReg: TEdit;
    txtForename: TEdit;
    txtSurname: TEdit;
    txtEmail: TEdit;
    txtAddress1: TEdit;
    txtAddress2: TEdit;
    txtAddress3: TEdit;
    txtBusiness: TEdit;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
```

```
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    procedure btnRegisterSubmitClick(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  frmRegister: TfrmRegister;

implementation

{$R *.lfm}

{ TfrmRegister }

uses login;

procedure TfrmRegister.btnRegisterSubmitClick(Sender: TObject);
// this procedure adds the details entered to the database
var
  strAddress, strSql: string;
begin
  with DataModule1 do
  begin
    //clear sql
    if sqlUsers.Active then
      sqlUsers.Close;
    //set sql
    strAddress := (txtAddress1.Text) + ', ' + (txtAddress2.Text) + ', ' + (txtAddress3.Text);
    strSql := 'INSERT INTO `user`(`username`, `password`, `forename`, `surname`, `email`,
`address`, `business name`) VALUES('
      + chr(39) + txtUsernameReg.Text + chr(39) + ',' + chr(39) + txtPasswordReg.Text +
      chr(39) + ',' + chr(39) + txtForename.Text + chr(39) + ',' + chr(39) + txtSurname.Text +
      chr(39) + ',' + chr(39) + txtEmail.Text + chr(39) + ',' + chr(39) + strAddress + chr(39) + ',' +
chr(39) +
      txtBusiness.Text + chr(39) + ')';
```

```
    SQLUsers.sql.Text := strSql;
    //execute the SQL
    sqlUsers.ExecSQL;
    MylerTransaction.Commit;
    self.hide; // hides current form
    frmLogin.Show; // goes back to FrmLogin
  end;
end;

end.
```

This part of the program is for the registration of a user's details where the procedure btnRegisterSubmitClick takes the users inputted data and adds it to the database so they can log in to the program.

## Main

```
unit main;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,
  StdCtrls, Data;

type

  { TfrmMain }

  TfrmMain = class(TForm)
    btnCreateNewFlyer: TButton;
    btnView: TButton;
    Edit1: TEdit;
    Image1: TImage;
    Label1: TLabel;
    lblForename: TLabel;
    lblSurname: TLabel;
    lblEmail: TLabel;
    ListBox1: TListBox;
    procedure btnCreateNewFlyerClick(Sender: TObject);
```

```
    procedure btnViewClick(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  frmMain: TfrmMain;

implementation

{$R *.lfm}

{ TfrmMain }
uses createflyer, viewflyer;

procedure TfrmMain.btnCreateNewFlyerClick(Sender: TObject);
begin
  frmFlyer.Show; // Shows the FrmFlyer
  frmMain.hide;  // hides the current form
end;

procedure TfrmMain.btnViewClick(Sender: TObject);
begin
  frmViewFlyer.Show; // Shows the FrmFlyer
  self.hide;  // hides the current form
  frmviewflyer.flyername:=edit1.text;
end;

end.
```

This part of the program is for the main form where the create flyer button takes you to a the create flyer form and the view flyer button takes you to the view flyer page and loads a specific flyer depending on the flyer name inputted.

## Create Flyer

```
unit createflyer;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,
  StdCtrls, Menus, data;

type

  { TfrmFlyer }

  TfrmFlyer = class(TForm)
    btnLogo: TButton;
    btnPicture: TButton;
    btnSave: TButton;
    ddTemplate: TComboBox;
    Edit1: TEdit;
    Image1: TImage;
    Image3: TImage;
    Image4: TImage;
    Label1: TLabel;
    Memo1: TMemo;
    Memo2: TMemo;
    OpenDialog1: TOpenDialog;
    Panel1: TPanel;
    procedure btnLogoClick(Sender: TObject);
    procedure btnPictureClick(Sender: TObject);
    procedure btnSaveClick(Sender: TObject);
    procedure ddTemplateChange(Sender: TObject);

  private
    { private declarations }
  public
    customerID:integer
  end;

var
```

```
  frmFlyer: TfrmFlyer;
  Fnamepic, Fnamelogo:string;

implementation

{$R *.lfm}

{ TfrmFlyer }

uses main;

procedure TfrmFlyer.ddTemplateChange(Sender: TObject);
//this procedure changes the size and positions of all the objects on
//the panel and the panel colour depending on the template selected
begin
  if ddTemplate.text='Template 1' then
  begin
  Panel1.color:=$008AA3E9;
  memo1.Height:=324;
  memo1.Width:=789;
  memo1.Top:=251;
  memo1.Left:=35;
  Image3.Height:=145;
  Image3.Width:=152;
  Image3.Top:=39;
  Image3.Left:=35;
  Image4.Height:=200;
  Image4.Width:=275;
  Image4.Top:=40;
  Image4.Left:=548;
  memo2.Height:=87;
  memo2.Width:=320;
  memo2.Top:=66;
  memo2.Left:=206;
  end;
  if ddTemplate.text='Template 2' then
  begin
  Panel1.color:=$00EED4B6;
  memo1.Height:=327;
  memo1.Width:=502;
  memo1.Top:=253;
```

```
memo1.Left:=35;
Image3.Height:=145;
Image3.Width:=152;
Image3.Top:=80;
Image3.Left:=606;
Image4.Height:=200;
Image4.Width:=275;
Image4.Top:=309;
Image4.Left:=552;
memo2.Height:=107;
memo2.Width:=453;
memo2.Top:=74;
memo2.Left:=59;
end;
if ddTemplate.text='Template 3' then
begin
Panel1.color:=$00C3D8CF;
memo1.Height:=534;
memo1.Width:=445;
memo1.Top:=39;
memo1.Left:=382;
Image3.Height:=145;
Image3.Width:=152;
Image3.Top:=163;
Image3.Left:=129;
Image4.Height:=200;
Image4.Width:=275;
Image4.Top:=338;
Image4.Left:=72;
memo2.Height:=107;
memo2.Width:=334;
memo2.Top:=39;
memo2.Left:=37;
end;
if ddTemplate.text='Template 4' then
begin
Panel1.color:=$007FD6F8;
memo1.Height:=534;
memo1.Width:=486;
memo1.Top:=43;
memo1.Left:=40;
```

```
  Image3.Height:=145;
  Image3.Width:=152;
  Image3.Top:=133;
  Image3.Left:=549;
  Image4.Height:=200;
  Image4.Width:=275;
  Image4.Top:=377;
  Image4.Left:=549;
  memo2.Height:=323;
  memo2.Width:=94;
  memo2.Top:=43;
  memo2.Left:=730;
  end;
end;

procedure TfrmFlyer.btnLogoClick(Sender: TObject);
//this procedure opens a dialog box to load a logo into the postion
//required on the template
begin
 if opendialog1.execute then
 fnamelogo:=opendialog1.filename;
 image3.picture.loadfromfile(Fnamelogo);
end;

procedure TfrmFlyer.btnPictureClick(Sender: TObject);
//this procedure opens a dialog box to load a picture into the postion
//required on the template
begin
 if opendialog1.execute then
 fnamepic:=opendialog1.filename;
 image4.picture.loadfromfile(Fnamepic);
end;

procedure TfrmFlyer.btnSaveClick(Sender: TObject);
//this procedure saves the created flyer to the database
var strsql: string;
begin
 with DataModule1 do
  begin
   //clear sql
   if sqlUsers.Active then
```

```
    sqlUsers.Close;
   //set sql
   strSql := 'INSERT INTO flyer(CustomerID, name, template, description, title, picture, logo)
VALUES('
    + chr(39) +inttostr(customerID)+ chr(39)+',' + chr(39) + edit1.Text + chr(39) + ',' + chr(39)
+ ddtemplate.Text +
    chr(39) + ',' + chr(39) + memo1.Text + chr(39) + ',' + chr(39) + memo2.Text +
    chr(39) + ',' + chr(34) + fnamepic + chr(34) + ',' + chr(34) + fnamelogo + chr(34) + ')';
   SQLUsers.sql.Text := strSql;
   //execute the SQL
   sqlUsers.ExecSQL;
   MylerTransaction.Commit;
   self.hide; // hides current form
   frmmain.Show; // goes back to FrmLogin
  end;
end;

end.
```

This part of the program is for the creation of a new flyer where the different input methods change the end flyer for example the procedure ddTemplateChange changes the layout and clour of the form depending on which template is selected from the dropdown list and the picture and logo buttons open a dialog box to load a file to fill the picture box. When the flyer has been completed and the users has entered a name for it the procedure btnSaveClick adds the flyers details to the database so it can be loaded at a later date however there is a problem at this stage which is that the file paths for the pictures are not added to database correctly as the '\' are left out even though the sql statement is built correctly and includes the '\' it takes them out in the database for no reason.

## View Flyer

```
unit viewflyer;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,
  StdCtrls, data;

type

  { Tfrmviewflyer }

  Tfrmviewflyer = class(TForm)
    Button1: TButton;
    Image3: TImage;
    Image4: TImage;
    Memo1: TMemo;
    Memo2: TMemo;
    Panel1: TPanel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
  flyername: string;
  end;

var
  frmviewflyer: Tfrmviewflyer;

implementation

{$R *.lfm}

{ Tfrmviewflyer }
uses
```

```
  main;

procedure Tfrmviewflyer.FormCreate(Sender: TObject);
// this procedure accesses the database and loads the users flyer
var
  strSQL: string;
begin
  with DataModule1 do
  begin
    //generate SQL
    strSQL := 'SELECT * FROM flyer WHERE name=';
    strSQL := strSQL + chr(39) + frmMain.edit1.text + chr(39);

    //clear any residual SQL
    if sqlUsers.Active then
      sqlUsers.Close;

    //set the SQL text
    sqlUsers.SQL.Text := strSQL;

    //execute the SQL
    sqlUsers.Open;

    //check if a record has been found
  if sqlUsers.RecordCount <> 0 then
     begin
   memo1.text:=sqlUsers.FieldByName('description').AsString;
   memo2.text:=sqlUsers.FieldByName('title').AsString;
//this set of if statements
   if (sqlUsers.FieldByName('Template').AsString = 'Template 1') then
   begin
   Panel1.color:=$008AA3E9;
   memo1.Height:=324;
   memo1.Width:=789;
   memo1.Top:=251;
   memo1.Left:=35;
   Image3.Height:=145;
   Image3.Width:=152;
   Image3.Top:=39;
   Image3.Left:=35;
   Image4.Height:=200;
```

```
Image4.Width:=275;
Image4.Top:=40;
Image4.Left:=548;
memo2.Height:=87;
memo2.Width:=320;
memo2.Top:=66;
memo2.Left:=206;
end;
if sqlUsers.FieldByName('Template').AsString = 'Template 2' then
begin
frmViewFlyer.Panel1.color:=$00EED4B6;
frmViewFlyer.memo1.Height:=327;
frmViewFlyer.memo1.Width:=502;
frmViewFlyer.memo1.Top:=253;
frmViewFlyer.memo1.Left:=35;
frmViewFlyer.Image3.Height:=145;
frmViewFlyer.Image3.Width:=152;
frmViewFlyer.Image3.Top:=80;
frmViewFlyer.Image3.Left:=606;
frmViewFlyer.Image4.Height:=200;
frmViewFlyer.Image4.Width:=275;
frmViewFlyer.Image4.Top:=309;
frmViewFlyer.Image4.Left:=552;
frmViewFlyer.memo2.Height:=107;
frmViewFlyer.memo2.Width:=453;
frmViewFlyer.memo2.Top:=74;
frmViewFlyer.memo2.Left:=59;
end;
if sqlUsers.FieldByName('Template').AsString = 'Template 3' then
begin
frmViewFlyer.Panel1.color:=$00C3D8CF;
frmViewFlyer.memo1.Height:=534;
frmViewFlyer.memo1.Width:=445;
frmViewFlyer.memo1.Top:=39;
frmViewFlyer.memo1.Left:=382;
frmViewFlyer.Image3.Height:=145;
frmViewFlyer.Image3.Width:=152;
frmViewFlyer.Image3.Top:=163;
frmViewFlyer.Image3.Left:=129;
frmViewFlyer.Image4.Height:=200;
frmViewFlyer.Image4.Width:=275;
```

```
  frmViewFlyer.Image4.Top:=338;
  frmViewFlyer.Image4.Left:=72;
  frmViewFlyer.memo2.Height:=107;
  frmViewFlyer.memo2.Width:=334;
  frmViewFlyer.memo2.Top:=39;
  frmViewFlyer.memo2.Left:=37;
  end;
  if sqlUsers.FieldByName('Template').AsString = 'Template 4' then
  begin
  frmViewFlyer.Panel1.color:=$007FD6F8;
  frmViewFlyer.memo1.Height:=534;
  frmViewFlyer.memo1.Width:=486;
  frmViewFlyer.memo1.Top:=43;
  frmViewFlyer.memo1.Left:=40;
  frmViewFlyer.Image3.Height:=145;
  frmViewFlyer.Image3.Width:=152;
  frmViewFlyer.Image3.Top:=133;
  frmViewFlyer.Image3.Left:=549;
  frmViewFlyer.Image4.Height:=200;
  frmViewFlyer.Image4.Width:=275;
  frmViewFlyer.Image4.Top:=377;
  frmViewFlyer.Image4.Left:=549;
  frmViewFlyer.memo2.Height:=323;
  frmViewFlyer.memo2.Width:=94;
  frmViewFlyer.memo2.Top:=43;
  frmViewFlyer.memo2.Left:=730;
  end;
  image3.picture.loadfromfile(sqlUsers.FieldByName('logo').AsString);
  image4.picture.loadfromfile(sqlUsers.FieldByName('picture').AsString);
  end
   else
   ShowMessage('No flyer with that name has been found. please enter a valid name');
 end;
end;
```

```
procedure Tfrmviewflyer.Button1Click(Sender: TObject);
begin
  self.hide;
  frmMain.show;
end;

end.
```

This part of the program is where an existed flyer is loaded when the correct flyer name is added in the main form when the view flyer button is pressed. The procedure FormCreate is used to create the opened flyer with the objects on the panel changing their properties depending on the data saved in the database.

## Myler

```
program myler;

{$mode objfpc}{$H+}

uses {$IFDEF UNIX} {$IFDEF UseCThreads}
  cthreads, {$ENDIF} {$ENDIF}
  Interfaces, // this includes the LCL widgetset
  Forms,
  login,
  Register,
  main,
  createflyer,
  Data, viewflyer;

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TfrmLogin, frmLogin);
  Application.CreateForm(TfrmRegister, frmRegister);
  Application.CreateForm(TfrmMain, frmMain);
  Application.CreateForm(TfrmFlyer, frmFlyer);
  Application.CreateForm(TDataModule1, DataModule1);
  Application.CreateForm(Tfrmviewflyer, frmviewflyer);
  Application.Run;
end.
```