|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 1 | Function Test | Form 1 | uploadData | Upload data from the txt file to a list and initialize the data | The code will target the txt file and extract its data into the program and display it | Form 1 screen should update now showing a list off all the children present | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 10 | Function Test | Form 3 | Family Graph | Display a info graphic of some of the data contained in the list | When executed a user is asked enter a year | Form 3 should display showing the percentage of children and their education catagory | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 9 | Function Test | Form 2 | School Times | Display what children are is what education category | When executed a user is asked enter a year | Form 1 should update displaying the children and what their education category is for the year entered | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 8 | Function Test | Form 1 | Children’s Allowance for the year | Display and calculate the children’s allowance for that year | When executed the children’s allowance is calculated and displayed | Form 1 should update displaying just the children allowance for the month | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 7 | Function Test | Form 1 | Children’s Allowance for the Month | Display and calculate the children’s allowance for that month | When executed the children’s allowance is calculated and displayed | Form 1 should update displaying just the children allowance for the year | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 6 | Function Test | Form 2 | Add a new child | Display a new window accepting a new childs details | Should ask for a name, date of birth and likes and add it to the list and also the text file | Should update the list with a new child and update the text file | As expected | Had to figure out an alternative for writing back to the text file. |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 5 | Function Test | Form 1 | List all multiple births | Display any multi births, twins triplets or quads, etc. | Calculate the multi births and display them | Should update the list displaying any or all multi births and the corresponding children | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 4 | Function Test | Form 1 | List children in alphabetical order | Print a list of the children’s nicknames in alphabetical order | Order the list using the children’s names in alphabetical order | Should update the list displaying the list again but now it should be ordered | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 3 | Function Test | Form 1 | List children by age | Print a list of the children’s nicknames ordered by age oldest first | Order the list using the children’s age starting with the oldest | Should update the list displaying the list again but now it should be ordered | As expected | All screens rendered as expected no action required |
|  |  |  |  |  |  |  |  |  |
| **Test No.** | **Test Type** | **Target Window** | **Test Name** | **Purpose of test** | **Test Data or Situation** | **Expected Result** | **Actual Result** | **Outcome and Action** |
| 2 | Function Test | Form 1 | Birthdays | Names of children with birthday falling in the next 7 days | Calculate the next 7 days and compare it to the children’s birthday | Should update the list displaying any children whose birthday falls in 7 days | As expected | All screens rendered as expected no action required |

# William mannix

# Nikita Pantskhava

# Software Development Year 2 Group Assignment 20%

# Documentation

This Software Documentation will provide documentation which will be used to explain the details for how the software was built.

In this documentation there will be narrative and graphical documentation of the software design and testing for the project including Snippets of the main code and testing using test data.

There are 4 sections to this Documentation.

## Test Plans

## GUI design.

## Code.

## Test Results.

## 2 GUI Design

The following section of the documentation aims to explain how we came to the final design of our graphic user interface. It will go through each section of the GUI and what functions and features of the visual studio .NET windows forms designer, images, events where used in order to achieve all the design elements of the GUI on our desktop application.



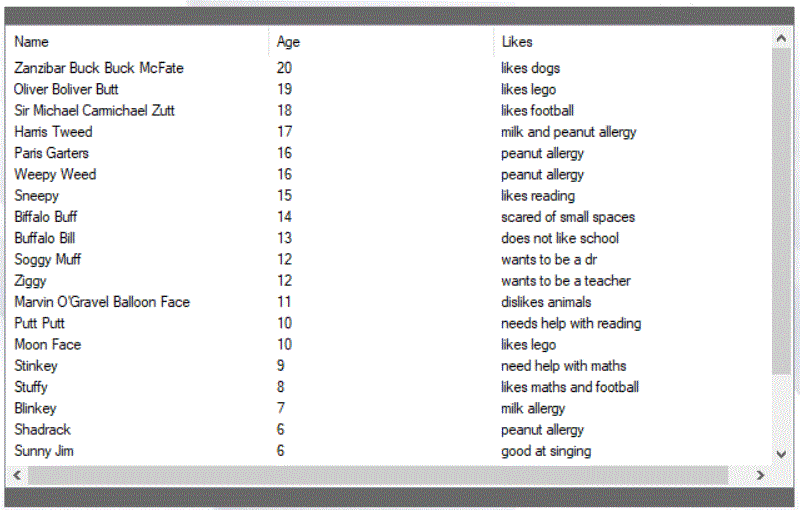
A lot of thought went into designing the GUI. It was important we got the right balance of colours right. The colour combination of the silver abstract background along with a grey side menu with black icons and white font gives the GUI a very eloquent look.

### Menu Side bar



The menu sidebar is achieved by using a grey panel with 10 buttons to navigate the different sections of the form with the text of the buttons used as the labels. A dim grey colour is used. Black picture icons are used to represent each section of the desktop app.

### Listview



An 841 x 456 list view with a grey panel at both the top and bottom and a white back colour is used to display the data.

### Form background



This 1124 x 574 image was used as the forms background image which really gave the form a sleek look.

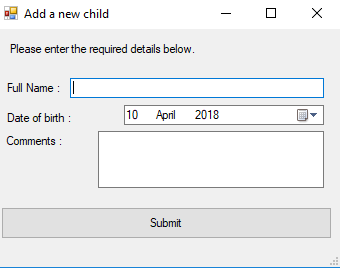
### Side panel

A 924x10 panel was used at the very top of the form to add an extra feature of design.



### User Input

Simple text boxes and a button is used to retrieve data from a user



# Code

This section of the documentation will demonstrate key snippets of code from the main functions within the project.

**Class Person**

The main class within the program is the person class. It has three properties used to describe the name, date of birth and comment of each child.

namespace Project

{

public partial class Form1 : Form

{

List<Person> listPerson = new List<Person>(); //initialize the datat into a list

string \_fullName = @"-\d+/\d+/\d+\D+"; // Extract the Full name from the list using Regex

string \_dateOfBirth = @"(\d+)/(\d+)/(\d+)"; // \d is a digit from 0 - 9, etc.

string \_comments = @"(?<=\d+\w+)";

string line = "";

string insertComments = "";

string tempDOB = "";

double pre, prim, sec, college, finished; //global variables used in the Graph

public string[] Dates;

DateTime currentdate = DateTime.Now; //set up the current Time and Date

public Form1()

{

InitializeComponent();

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

}

private void Form1\_Load(object sender, EventArgs e) /\*====\*/

{

//button10.Enabled = false; // Disable for the time being

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void \_display\_SelectedIndexChanged(object sender, EventArgs e)

{

}

public void uploadData() //Uploaded data from the file and initialize it

{

try

{

using (StreamReader sr = new StreamReader("MaCaveFamily.txt"))

{

string name = "";

while ((line = sr.ReadLine()) != null) // Exclude expression used for the name

{

string[] FullName = Regex.Split(line, \_fullName);

foreach (string match in FullName)

{

if (match != "")

{

name = match;

name = name.Replace("-", " ");

//Person getName = new Person(line, "", "");

//listPerson.Add(getName);

}

}

string[] substringDob = Regex.Split(line, \_dateOfBirth); //Include expression used to find the Date

Match matchDob = Regex.Match(line, \_dateOfBirth);

if (matchDob.Success)

{

tempDOB = matchDob.Value;

}

string[] Comments = Regex.Split(line, \_comments); //Include expression used to find the Comments

foreach (string item in Comments)

{

insertComments = item;

}

insertComments = insertComments.Replace("-", "");

DateTime insertDOB = DateTime.ParseExact(tempDOB, "dd/MM/yyyy", null); //sets the default format for the date

Person newPerson = new Person(name, insertDOB, insertComments);

listPerson.Add(newPerson); //Add a new Person to the list

}

}

}

catch (Exception e)

{

Console.WriteLine("The file could not be read: ");

Console.WriteLine(e.Message);

}

button1.Enabled = false;

}

public void birthdays()

{

\_display.Items.Clear(); //Clear the items in the Display box

DateTime tempDate = new DateTime();

foreach (Person p in listPerson)

{

tempDate = p.getDOB();

int day = Convert.ToInt32(tempDate.Day);

int month = Convert.ToInt32(tempDate.Month);

if (month == currentdate.Month || month == currentdate.Month + 1)

{

if (month == currentdate.Month)

{

int currentDay = currentdate.Day;

if (day - currentDay > 0 && day - currentDay < 8)

{

ListViewItem lvi = new ListViewItem(p.getName()); //Adds a user to the list into the names column

lvi.SubItems.Add(p.getDOB().ToString());

\_display.Items.Add(lvi);

}

}

if (month == currentdate.Month + 1)

{

int currentDay = currentdate.Day;

if (day - currentDay == 7)

{

ListViewItem lvi = new ListViewItem(p.getName());

lvi.SubItems.Add(p.getDOB().ToString());

\_display.Items.Add(lvi);

}

}

}

}

if (\_display.Items.Count < 1) //Checks if the list contains zero(0) items if so then simply output the following message

{

\_display.Columns[0].Text = "";

\_display.Columns[1].Text = "";

\_display.Columns[2].Text = "";

ListViewItem lvi = new ListViewItem("No Birthdays coming up in the next 7 days!");

\_display.Items.Add(lvi);

}

}

public void listByAge()

{

listPerson.Sort(new ListByAge());

foreach (Person p in listPerson)

{

ListViewItem lvi = new ListViewItem(p.getName());

lvi.SubItems.Add(CalculateAge(p.getDOB()).ToString());

lvi.SubItems.Add(p.getComments());

\_display.Items.Add(lvi);

}

}

public void listByABC()

{

listPerson.Sort(new ListBylphaOrder());

foreach (Person p in listPerson)

{

ListViewItem lvi = new ListViewItem(p.getName());

lvi.SubItems.Add(p.getDOB().ToString("dd/MM/yyyy"));

lvi.SubItems.Add(p.getComments());

\_display.Items.Add(lvi); //dislay list into List view

}

}

public void multipleBirths()

{

listPerson.Sort(new ListByAge());

for (int i = 0; i < listPerson.Count; i++)

{

int temp = DuplicateBirth(listPerson[i]);

if (temp == 2)

{

ListViewItem list = new ListViewItem(listPerson[i + 1].getName());

ListViewItem list1 = new ListViewItem(listPerson[i].getName()); //Used a duplicate here to temporarily fix one set of twins displaying

list.SubItems.Add(listPerson[i].DOB.Day + "/" + listPerson[i].DOB.Month + "/" + listPerson[i].DOB.Year);

list1.SubItems.Add(listPerson[i].DOB.Day + "/" + listPerson[i].DOB.Month + "/" + listPerson[i].DOB.Year);

list.SubItems.Add("Twin");

list1.SubItems.Add("Twin");

\_display.Items.Add(list);

\_display.Items.Add(list1);

i++;

}

else if (temp == 3)

{

ListViewItem lvi = new ListViewItem(listPerson[i].getName());

lvi.SubItems.Add(listPerson[i].DOB.Day + "/" + listPerson[i].DOB.Month + "/" + listPerson[i].DOB.Year); //Re Format for the Date of birth just in case

lvi.SubItems.Add("Triplet");

\_display.Items.Add(lvi);

}

else if (temp == 4)

{

ListViewItem lvi = new ListViewItem(listPerson[i].getName());

lvi.SubItems.Add(listPerson[i].DOB.Day + "/" + listPerson[i].DOB.Month + "/" + listPerson[i].DOB.Year);

lvi.SubItems.Add("Quadruplet");

\_display.Items.Add(lvi);

}

}

}

private int DuplicateBirth(Person person) //Check for exact same birthdays

{

int count = 1;

listPerson.Sort(new ListByAge());

for (int i = 0; i < listPerson.Count; i++)

{

if (listPerson[i].DOB == person.DOB)

{

if (listPerson[i].Equals(person))

{

//\* -- \*//

}

else

{

count++;

}

}

}

return count;

}

public void addChild()

{

//Add new child - nickname, date of birth and

//comment should be saved for the new child.The new details

//should be save in the file and program data refreshed.

string name = "";

DateTime DOB = new DateTime();

string likes = "";

bool present = false;

using (AddChild AddChildToList = new AddChild())

{

if (AddChildToList.ShowDialog() == DialogResult.OK) //indicates the return value of the dialog box

{

name = AddChildToList.FullName;

DOB = DateTime.Parse(AddChildToList.DateOfBirth);

likes = AddChildToList.Likes;

foreach (Person p in listPerson)

{

if (p.name == name && p.DOB == DOB)

{

present = true;

MessageBox.Show(name + " Is already a part of the list, please try again!");

}

}

if (!present)

{

Person temp = new Person(name, DOB, likes);

listPerson.Add(temp);

string newChild = Environment.NewLine + name + "-" + DOB.ToString("dd/MM/yyyy") + "-" + likes; //Writes into the text file.

File.AppendAllText("MaCaveFamily.txt", newChild);

MessageBox.Show("You have successfully added " + name + " to the list!");

}

}

}

}

public void nameNextBaby()

{

//Name the next baby - An exciting algorithm that you

//come up with for Mrs.McCave’s next baby name.

string[] FistName = { "Baldy","Chubby","Clean","Dazzling","Drab","Fancy",

"Flabby","Gorgeous","Long",

"Plain","Scruffy","Skinny"};

Random rand = new Random();

string lastname = "";

int indexFirstName = rand.Next(FistName.Length);

if (FistName[indexFirstName] == "Baldy")

{

lastname = "Mcaldy";

}

if (FistName[indexFirstName] == "Chubby")

{

lastname = "O'ruddy";

}

if (FistName[indexFirstName] == "Clean")

{

lastname = "Arlene";

}

if (FistName[indexFirstName] == "Dazzling")

{

lastname = "Darragh";

}

if (FistName[indexFirstName] == "Drab")

{

lastname = "McRab";

}

if (FistName[indexFirstName] == "Fancy")

{

lastname = "Pancy";

}

if (FistName[indexFirstName] == "Flabby")

{

lastname = "O'Toole";

}

if (FistName[indexFirstName] == "Gorgeous")

{

lastname = "George";

}

if (FistName[indexFirstName] == "Plain")

{

lastname = "Jane";

}

if (FistName[indexFirstName] == "Scruffy")

{

lastname = "McGuffy";

}

if (FistName[indexFirstName] == "Skinny")

{

lastname = "Mini";

}

if (FistName[indexFirstName] == "Long")

{

lastname = "John";

}

MessageBox.Show(FistName[indexFirstName] + " " + lastname); //Random name generator based off the random first name selected from the list which

//is matched with a rhyming surname

}

public void allowanceMonth()

{

int final = 0;

listPerson.Sort(new ListByAge());

for (int i = 0; i < listPerson.Count; i++)

{

if (CalculateAge(listPerson[i].DOB) < 18)

{

int multi = DuplicateBirth(listPerson[i]);

if (multi == 1)

{

final += 140;

}

else if (multi == 2)

{

final += 210;

final += 210;

i++;

}

else

{

final += 280;

}

}

}

ListViewItem monthlyAllowence = new ListViewItem("Monthly allowance total");

monthlyAllowence.SubItems.Add(final.ToString());

\_display.Items.Add(monthlyAllowence);

}

public void allowanceYear()

{

int final = 0;

listPerson.Sort(new ListByAge());

for (int i = 0; i < listPerson.Count; i++)

{

if (CalculateAge(listPerson[i].DOB) < 18)

{

int multi = DuplicateBirth(listPerson[i]);

if (multi == 1)

{

DateTime tempP = new DateTime(listPerson[i].DOB.Year, listPerson[i].DOB.Month, listPerson[i].DOB.Day);

int tempFinal = 0;

for (int y = 0; y < 12; y++)

{

if (CalculateAge(tempP.AddMonths(DateTime.Today.Month)) < 18)

{

tempFinal += 140;

tempP = tempP.AddMonths(-1);

}

}

final += tempFinal;

}

else if (multi == 2)

{

Person tempPerson1 = new Person();

tempPerson1 = listPerson[i + 1];

int tempFinal1 = 0;

for (int y = 0; y < 12; y++)

{

if (CalculateAge(tempPerson1.DOB) < 18)

{

tempFinal1 += 210;

tempPerson1.DOB.AddMonths(1);

}

}

final += tempFinal1;

Person tempPerson = new Person();

tempPerson = listPerson[i];

int tempFinal = 0;

for (int y = 0; y < 12; y++)

{

if (CalculateAge(tempPerson.DOB) < 18)

{

tempFinal += 210;

tempPerson.DOB.AddMonths(1);

}

}

final += tempFinal;

i++;

}

else

{

Person tempP = new Person();

tempP = listPerson[i];

int tempFinal = 0;

for (int y = 0; y < 12; y++)

{

if (CalculateAge(tempP.DOB) < 18)

{

tempFinal += 280;

tempP.DOB.AddMonths(1);

}

}

final += tempFinal;

}

}

}

ListViewItem yearlyAllowence = new ListViewItem("Yearly allowence total");

yearlyAllowence.SubItems.Add(final.ToString());

\_display.Items.Add(yearlyAllowence);

}

public void schoolTimes()

{

int year = 0;

bool result = false;

using (Form2 form2 = new Form2())

{

if (form2.ShowDialog() == DialogResult.OK)

{

if (form2.Year == null)

{

year = DateTime.Today.Year;

}

else

{

year = Convert.ToInt32(form2.Year);

}

result = true;

}

}

if (result)

{

DateTime temp = new DateTime(year, 1, 1);

foreach (Person p in listPerson)

{

DateTime tempDate = new DateTime();

if (temp == DateTime.Today)

{

tempDate = (p.DOB.AddDays(temp.Day));

tempDate = (p.DOB.AddMonths(temp.Month));

tempDate = (p.DOB.AddYears(temp.Year));

}

tempDate = (p.DOB.AddYears(DateTime.Today.Year - temp.Year));

ListViewItem lvi = new ListViewItem(p.getName());

lvi.SubItems.Add(CalculateAge(tempDate).ToString());

if (CalculateAge(tempDate) >= 1 && CalculateAge(tempDate) < 5)

{

lvi.SubItems.Add("Pre school");

}

else if (CalculateAge(tempDate) >= 5 && CalculateAge(tempDate) <= 11)

{

lvi.SubItems.Add("Primary School");

}

else if (CalculateAge(tempDate) >= 12 && CalculateAge(tempDate) <= 18)

{

lvi.SubItems.Add("Secondary School");

}

else if (CalculateAge(tempDate) >= 19 && CalculateAge(tempDate) <= 23)

{

lvi.SubItems.Add("College");

}

else

{

lvi.SubItems.Add("Finished");

}

\_display.Items.Add(lvi);

}

}

if(!result)

{

MessageBox.Show("Nothing entered, no Info updated!", "Info",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void infoGraphic()

{

}

public void headingRePop() // Use to re populate the titles of the List box after each reset

{

\_display.Columns[0].Text = "Name";

\_display.Columns[1].Text = "Date of birth";

\_display.Columns[2].Text = "Likes";

}

public static int CalculateAge(DateTime dob) // Calculate the age compared to todays date

{

DateTime currentDate = DateTime.Today;

int currentYear = currentDate.Year;

int DOBYear = dob.Year;

return currentYear - DOBYear;

}

private void button1\_Click(object sender, EventArgs e) // Upload and initialize data

{

uploadData();

foreach (Person P in listPerson)

{

ListViewItem Person = new ListViewItem(P.getName());

Person.SubItems.Add(P.getDOB().ToString("dd/MM/yyyy"));

Person.SubItems.Add(P.getComments());

\_display.Items.Add(Person); // populate list accordingly.

}

}

private void button2\_Click(object sender, EventArgs e) // Birthdays in the next 7 days

{

headingRePop();

birthdays();

}

private void button3\_Click(object sender, EventArgs e) // list names sorted started from the oldest

{

headingRePop();

\_display.Columns[1].Text = "Age";

\_display.Items.Clear();

listByAge();

}

private void button4\_Click(object sender, EventArgs e) // List names by alphabetical order

{

\_display.Items.Clear();

listByABC();

headingRePop();

}

private void button5\_Click(object sender, EventArgs e) // List multi births

{

headingRePop();

\_display.Columns[2].Text = "Multi births";

\_display.Items.Clear();

multipleBirths();

}

private void button6\_Click(object sender, EventArgs e) // Add a new child

{

addChild();

}

private void button7\_Click(object sender, EventArgs e) // Random baby name

{

nameNextBaby();

}

private void button8\_Click(object sender, EventArgs e) // Allowence for the children per month and year

{

\_display.Items.Clear();

headingRePop();

\_display.Columns[1].Text = "Total";

\_display.Columns[2].Text = "";

allowanceMonth();

allowanceYear();

}

private void button9\_Click(object sender, EventArgs e) //School times

{

\_display.Items.Clear();

schoolTimes();

headingRePop();

\_display.Columns[2].Text = "Education";

}

private void button10\_Click(object sender, EventArgs e) //re used code from the School times planner for the graph

{

Form3 form3 = new Form3();

int year = 0;

bool result = false;

using (Form2 form2 = new Form2())

{

if (form2.ShowDialog() == DialogResult.OK)

{

if (form2.Year == null)

{

year = DateTime.Today.Year;

}

else

{

year = Convert.ToInt32(form2.Year);

}

result = true;

}

}

if (result)

{

DateTime temp = new DateTime(year, 1, 1);

foreach (Person p in listPerson)

{

DateTime tempDate = new DateTime();

if (temp == DateTime.Today)

{

tempDate = (p.DOB.AddDays(temp.Day));

tempDate = (p.DOB.AddMonths(temp.Month));

tempDate = (p.DOB.AddYears(temp.Year));

}

tempDate = (p.DOB.AddYears(DateTime.Today.Year - temp.Year));

ListViewItem lvi = new ListViewItem(p.getName());

if (CalculateAge(tempDate) >= 1 && CalculateAge(tempDate) < 5)

{

pre++;

}

else if (CalculateAge(tempDate) >= 5 && CalculateAge(tempDate) <= 11)

{

prim++;

}

else if (CalculateAge(tempDate) >= 12 && CalculateAge(tempDate) <= 18)

{

sec++;

}

else if (CalculateAge(tempDate) >= 19 && CalculateAge(tempDate) <= 23)

{

college++;

}

else

{

finished++;

}

}

}

form3.pre = pre; // Populate the Form 3 chart.

form3.prim = prim;

form3.sec = sec;

form3.college = college;

form3.finished = finished;

form3.ShowDialog();

}

private void quit\_Click(object sender, EventArgs e) // Quits the program

{

this.Close(); // Close ?

}

}

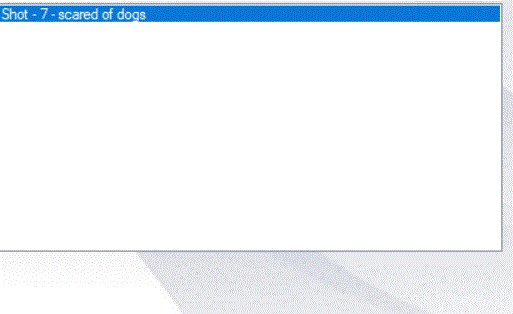
}

## Test results

### Option 1 – List birthdays in the next 7 day’s

Condition - Shot”,10/04/2012,scared of dogs is the only child listed as having a birthday in the next 7 days. = TRUE

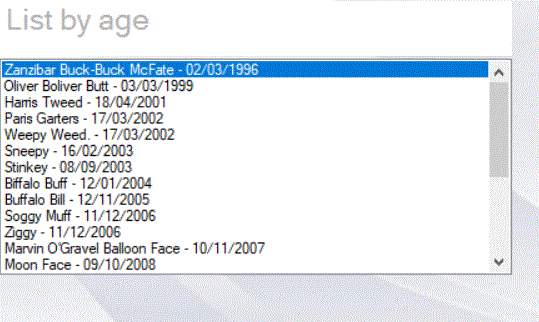
Test Result = Pass

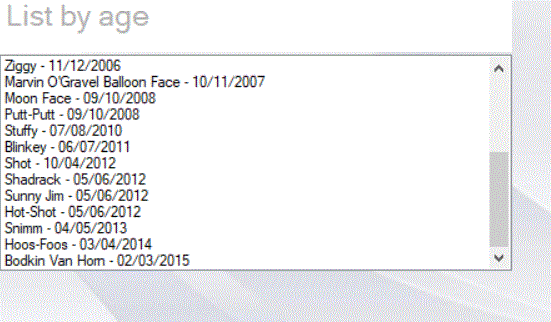


### Option 2 – List Children by age starting with the oldest first

Condition = Zanzibar Buck-Buck McFate is the oldest and Bodkin Van Horn is the youngest. = TRUE

Test Result = Pass



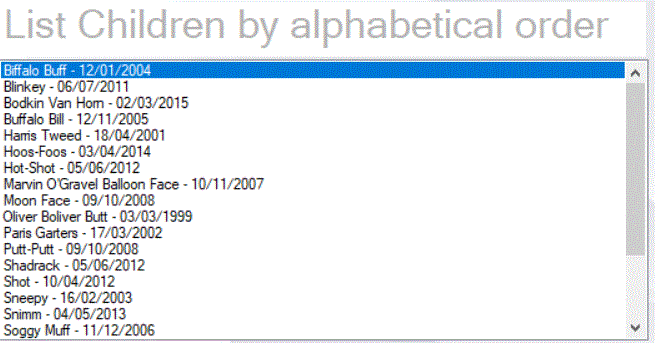


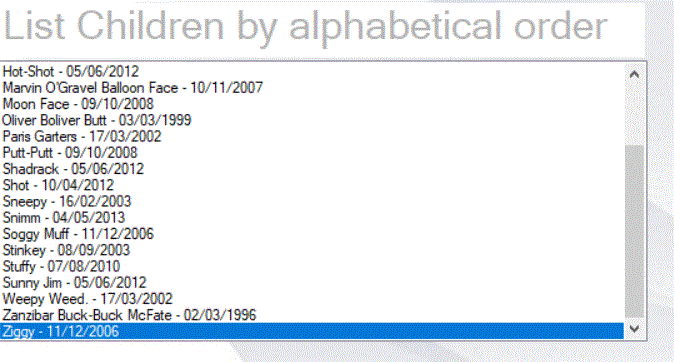
### Option 3 – List Children by ABC

Condition = Biffalo buff is at the top of the alphabetical order list = TRUE

Condition = Ziggy is at the bottom of alphabetical order list. = TRUE

Test Result = Pass



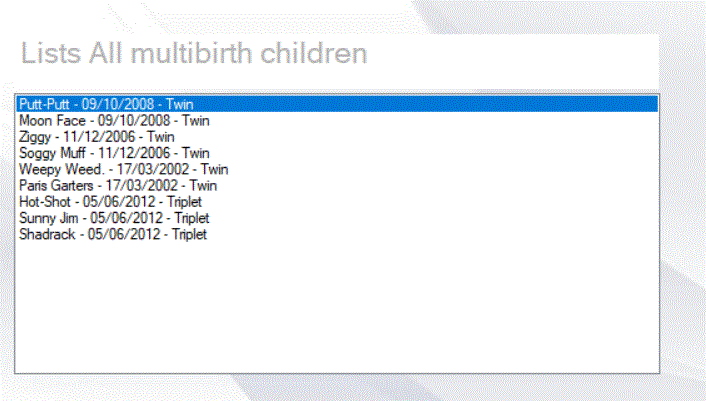


### Option 4 – List Children by Multibirth type

Condition - Putt-Putt,Moon-face,Ziggy,Soggy Muff, Weepy Weed,Paris Garters are listed as twins = TRUE

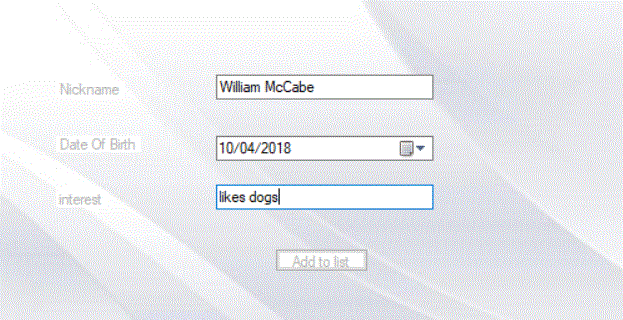
Condition - Hot-Shot,Sunny Jim and Shadrack are listed as triplets. = TRUE

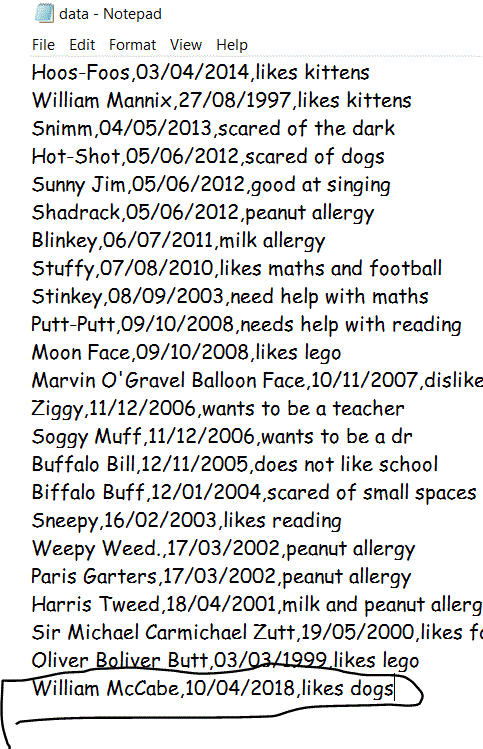
Test Result = Pass



### Option 5 – Add Child

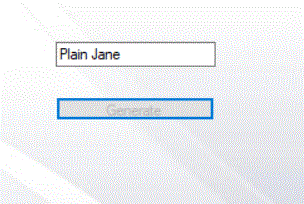
Test Result = Pass





### Option 6 – Nickname Generator

Test Result = Pass



### Option 7 – List Children by Milestone

Test Result = Pass

