# 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.

## GUI designs

## Test Plans.

## Code.

## Test Results.

## 1 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.



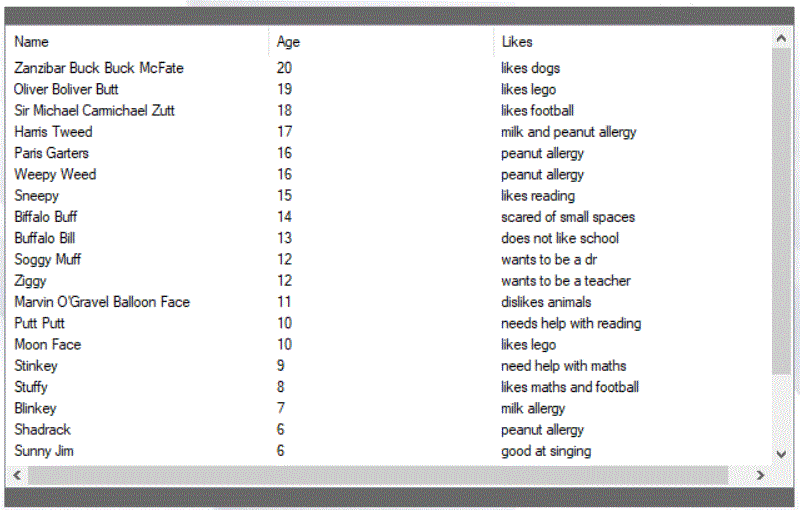
A lot of thought went into designing the GUI. It was important we got the right balance of colours right. The colour combination of the silver abstract background along with a grey side menu with black icons and white font gives the GUI a very eloquent look.

### 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.

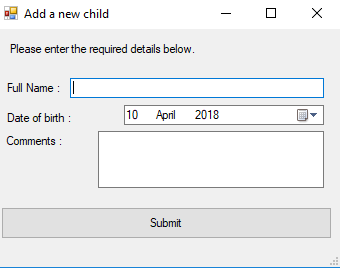
### Side panel

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



# Testing Plan

In the test plan, we plan to test each menu option with a range of test data to cover as many possible scenarios as we can think of to ensure our options are handled to the best they can.

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

#### Test data conditions

##### Test Pass if following condition is true.

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

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

#### Test data conditions

##### Test Passes if following condition is true.

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

##### Option 3 – List children by alphabetical order

#### Test data conditions

##### Test Passes if following condition is true.

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

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

##### Option 4 – List children Multi Births

#### Test data conditions

##### Test Passes if following condition is true.

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

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

##### Option 5– Add new child

#### Test data conditions

##### Test Passes if following condition is true.

Condition = New line of child details gets added to the file.

Condition = You cannot enter the same nickname.

**Option 6 – Generate random Nickname**

**Test data conditions**

##### Test Passes if following condition is true.

Condition = Generates sequence of random Nicknames

**Option 7 – List children Multi Births**

**Test data conditions**

**Test Passes** **if following condition is true**

Condition = Month allowance = 12x 140 + 6 x 210 + 3 x 280 = 3780

Condition = Yearly allowance = 3780 x 12 = 45360

**Option 8 – List children Milestone**

**Test data conditions**

**Test Passes** **if following condition is true**

2003 entered

Condition = Weepy weed – Pre school

Condition = Paris Garters – Pre School

Condition = Harris Tweed – Pre School

Condition = Oliver Boliver Butt – Pre School

Condition = Zanzibar Buck-Buck Mcfate – Primary School

# 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.

class Person

{

=

public string name { get; set; }

public DateTime DOB { get; set; }

public string comments { get; set; }

public Person()

{

this.name = "";

this.DOB = new DateTime();

this.comments = "";

}

public Person(string Name, DateTime DOB, string Likes)

{

this.name = Name;

this.DOB = DOB;

this.comments = Likes;

}

public string getName()

{

return this.name;

}

public DateTime getDOB()

{

return this.DOB;

}

public string getComments()

{

return this.comments;

}

}

**Upload data()**

uploads the data from the Mccave.txt file. Uses Try catch to cath an error if the file doesn’t exist. Extracts the individual data needed for the class from text file using .Split() function and adds the data to class.

public void uploadData()

{

try

{

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

{

string name = "";

while ((line = sr.ReadLine()) != null)

{

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);

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

if (matchDob.Success)

{

tempDOB = matchDob.Value;

}

string[] Comments = Regex.Split(line, \_comments);

foreach (string item in Comments)

{

insertComments = item;

}

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

DateTime insertDOB = DateTime.ParseExact(tempDOB, "dd/MM/yyyy", null);

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

listPerson.Add(newPerson);

}

}

}

catch (Exception e)

{

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

Console.WriteLine(e.Message);

}

**Birthdays()**

This function checks each object in listPerson and checks to see if their Date of birth falls within the next 7 days.

public void birthdays()

{

\_display.Items.Clear();

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());

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);

}

}

**ListByAge()**

This function iterates through each child in listPerson and sorts the children by age, oldest first.

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);

}

**ListByABC()**

This function iterates through each child in listPerson and sorts the children in Alphabetical order.

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);

}

}

**addChild()**

This function takes the input from user, Assigns the user inputted data to a new child object and the new child is added to the listPerson list. A new line containing the new data is amended to the file. An error is caught if the child nickname already exists in the file.

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)

{

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;

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

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

}

}

}

}

**multipleBirths()**

This function iterates through the List of children and compares their date of births. Depending on how much of the same date of birth shows up in the list, The children who’s date of birth’s are the same will be assigned a multibirth depending on how much of their date of birth shows up in the list.

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());

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);

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);

}

}

}

**nameNextBaby()**

This function chooses a random index from the first name string array. Concatenates it with similar sounding last name and it is outputted.

public void nameNextBaby()

{

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

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

string[] FirstName = { "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);

}

**allowanceMonth ()**

This function calculates the total child benefit the McCave family is entitled to for the current month. It checks to see if a child is multibirth and checks to see if a child is under 18. It then calculates the monthly allowance based on the rules.

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);

}

**allowanceYear()**

This function calculates the total child benefit the McCave family is entitled to for the current Year. It checks to see if a child is multibirth and checks to see if a child is under 18. It also checks to see what month a child that is 17’s birthday falls on then calculates the yearly allowance based on the rules

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);

}

**schoolTimes()**

This function checks each child to see what age they are in September of a year inputted by the user. It will then assign each child with a educational milestone. “Preschool”,”Primary school”,” Secondary School” , “College” depending on what age they are in September of the given year.

public void schoolTimes()

{

int year = 0;

using (Form2 form2 = new Form2())

{

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

{

if (form2.Year == null)

{

year = DateTime.Today.Year;

}

else

{

year = Convert.ToInt32(form2.Year);

}

}

}

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);

}

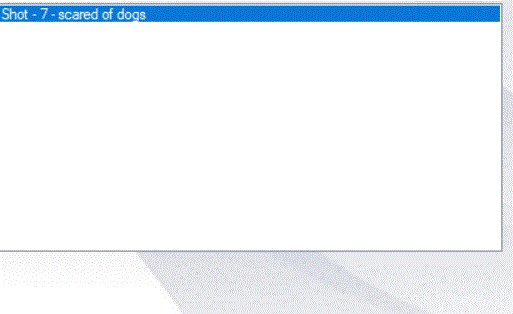
}

## 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

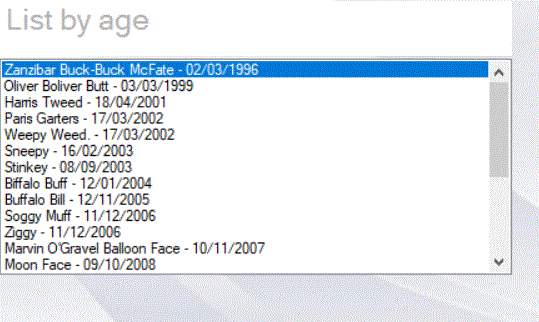
Test Result = Pass

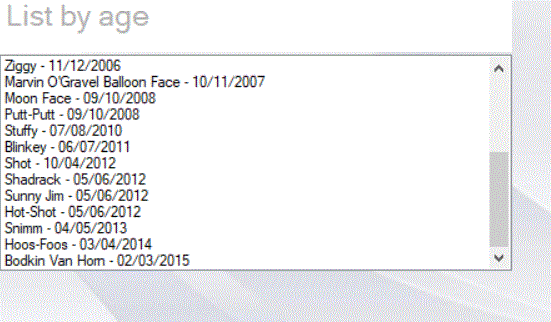


### Option 2 – List Children by age starting with the oldest first

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

Test Result = Pass



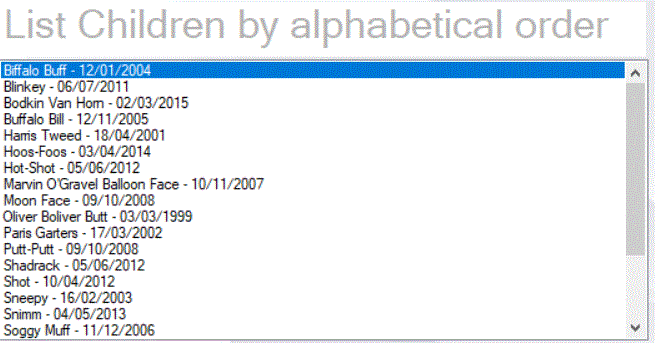


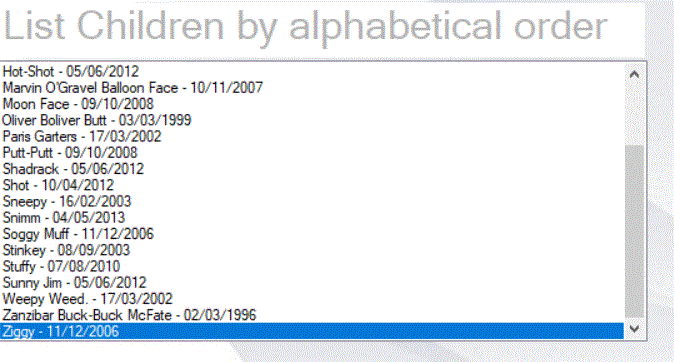
### Option 3 – List Children by ABC

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



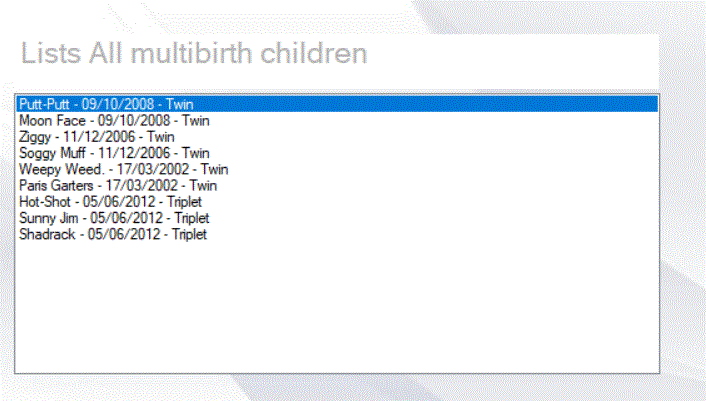


### Option 4 – List Children by Multibirth type

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

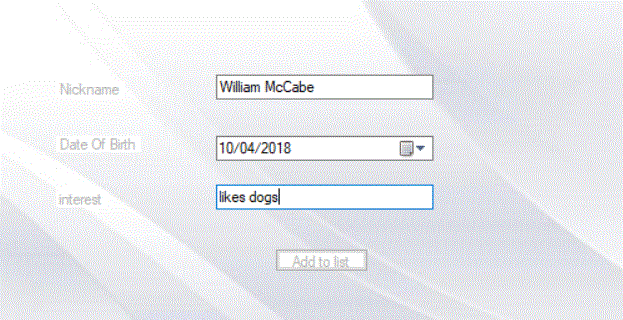
Condition - Hot-Shot,Sunny Jim and Shadrack are listed as triplets. = TRUE

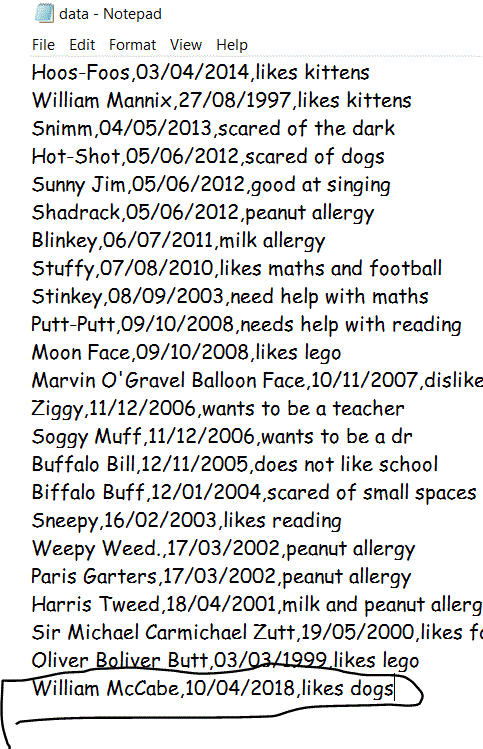
Test Result = Pass



### Option 5 – Add Child

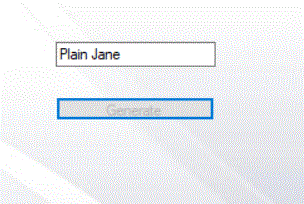
Test Result = Pass





### Option 6 – Nickname Generator

Test Result = Pass



### Option 7 – List Children by Milestone

Test Result = Pass

