## Aim

The aim of this project was to enhance our programming capabilities in C sharp. This would help in solving real world problems.

## Objectives

The project requires us to program a two-floor office building with an elevator to simulate it. The elevator is made up of a room with a control panel that includes buttons and a display window. There are also request buttons outside each floor.

The elevator doors should open when it arrives at a floor and closes when it departs.

## Task Description

The project requires the program to have a GUI (Graphical user Interface) containing two corresponding request buttons on each floor. A control panel with two buttons and a display window, the display window shows on what floor the elevator is on. Also a log button to show the elevator log.

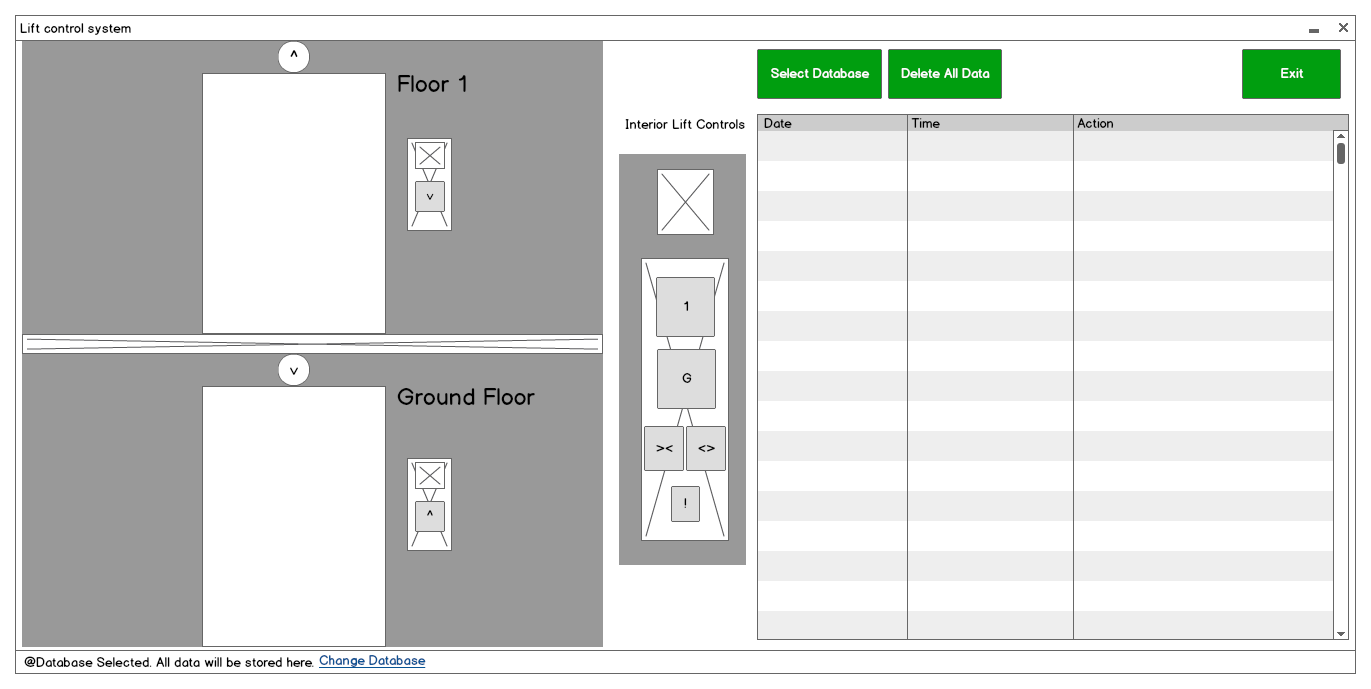
The elevator should move to the corresponding floor when the request button or the floor number button for the floor is pressed.

The program should implement an animation system to animate the events i.e. moving the lift and the doors.

A log is required to store all actions including the time information in a MS Access database.

## Design

The project started with a concept design as shown below:



The graphical user interface design was kept simple. Although during the actual programming some components were omitted or changed.

## Program

The program is broken down and explained as follows:

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Speech;

using System.Speech.Synthesis;

using System.Data.OleDb;

The above code adds all the references to the libraries used in the program.

using System.Speech;

using System.Speech.Synthesis;

This code is used to use the library that enables us to us text to speech functions and add voice to our program.

using System.Data.OleDb;

This code is used to connect the program to a database.

//variables

int y\_up = 63;

int y\_down = 376;

int x\_door\_left\_close = 74;

int x\_door\_left\_open = 12;

int x\_door\_right\_close = 139;

int x\_door\_right\_open = 200;

bool go\_up = false;

bool go\_down = false;

bool arrived\_G = false;

bool arrived\_1 = false;

//Database Variables and instantiations

private bool filled;

public DataSet ds = new DataSet();

//object

SpeechSynthesizer reader = new SpeechSynthesizer();

This part of the code instantiates all the variables and objects used in the program.

public Form1()

{

InitializeComponent();

}

This function calls the initializeComponent(); function that builds up the whole user interface.

private void timer\_lift\_down\_Tick(object sender, EventArgs e) //Moves the lift Down

{

if (picture\_lift.Top <= y\_down)

{

picture\_lift.Top += 1;

}

else

{

timer\_lift\_down.Enabled = false;

btn\_up.Enabled = true;

btn\_1.Enabled = true;

btn\_close.Enabled = true;

btn\_open.Enabled = true;

btn\_down.BackColor = Color.White;

btn\_G.BackColor = Color.White;

door\_open\_down();

arrived\_G = true;

picture\_lift.Image = global::ElevatorControl.Properties.Resources.Inside\_of\_the\_lift;

display\_panel.Image = global::ElevatorControl.Properties.Resources.G;

display\_top.Image = global::ElevatorControl.Properties.Resources.G;

display\_bottom.Image = global::ElevatorControl.Properties.Resources.G;

}

}

This function is called every time the timer interval runs out and moves the lift down. At every interval an if statement check if the current location of the picture box has reached its final location and moves it by 1 pixel down. When the picture box reaches its final co ordinates, the timer is disabled and all buttons are enabled again.

The private void timer\_lift\_up\_Tick(object sender, EventArgs e) //Moves the lift Up

Function works in the same way, but moves the lift up.

private void timer\_door\_open\_down\_Tick(object sender, EventArgs e) //Opens the Ground floor doors

{

if (door\_left\_down.Left >= x\_door\_left\_open && door\_right\_down.Left <= x\_door\_right\_open)

{

door\_left\_down.Left -= 1;

door\_right\_down.Left += 1;

}

else

{

timer\_door\_open\_down.Enabled = false;

}

}

This function is called at every interval of the timer\_door\_open\_down and it opens the doors at the ground floor.

private void timer\_door\_close\_down\_Tick(object sender, EventArgs e)

private void timer\_door\_open\_up\_Tick(object sender, EventArgs e)

private void timer\_door\_close\_up\_Tick(object sender, EventArgs e)

private void door\_close\_down() //Activates the timer for closing Ground floor doors

{

reader.Speak("doors closing");

insertdata("Doors Closing @Ground Floor");

timer\_door\_close\_down.Enabled = true;

timer\_door\_open\_down.Enabled = false;

}

This function enables the timer to close the door for the ground floor. It also calls another function, insertdata("Doors Closing @Ground Floor");

This assigns a string as a function parameter.

private void door\_open\_down()

private void door\_close\_up()

private void door\_open\_up()

The above functions work in the same way.

private void going\_up() //Activates the timer for moving the lift up

{

go\_up = true;

door\_close\_down();

btn\_G.Enabled = false;

btn\_down.Enabled = false;

btn\_close.Enabled = false;

btn\_open.Enabled = false;

arrived\_G = false;

insertdata("Lift Going Up");

}

This function calls the door\_close\_down() method and disables other buttons

private void going\_down()

This function also works in the same manner although it calls the door\_close\_up() method

private void btn\_down\_Click(object sender, EventArgs e)

{

btn\_up.BackColor = Color.Red;

going\_up();

}

This function is an event handler when the button btn\_down\_click is pressed. It changes the backcolor to red and calls the going\_up() method

private void btn\_1\_Click(object sender, EventArgs e)

is the same.

private void btn\_up\_Click(object sender, EventArgs e)

private void btn\_G\_Click(object sender, EventArgs e)

work in the same way but they call going\_down() method instead.

private void btn\_alarm\_Click(object sender, EventArgs e)

{

btn\_alarm.BackColor = Color.Green;

reader.Speak("Emergency Stop. Please exit carefully.");

insertdata("Emergency Stop!");

timer\_lift\_down.Enabled = false;

timer\_lift\_up.Enabled = false;

timer\_door\_open\_down.Enabled = true;

timer\_door\_open\_up.Enabled = true;

display\_panel.Image = global::ElevatorControl.Properties.Resources.alarmbellbutton;

display\_top.Image = global::ElevatorControl.Properties.Resources.alarmbellbutton;

display\_bottom.Image = global::ElevatorControl.Properties.Resources.alarmbellbutton;

}

This function Stops the lift movement and opens all the doors on both floors. It is an added function not required by the project.

private void btn\_displaylog\_Click(object sender, EventArgs e)

{

try

{

string dbconnection = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=ElevatorDatabase.accdb;";

string dbcommand = "Select \* from Actions;";

OleDbConnection conn = new OleDbConnection(dbconnection);

OleDbCommand comm = new OleDbCommand(dbcommand, conn);

OleDbDataAdapter adapter = new OleDbDataAdapter(comm);

//cnn.Open();

conn.Open();

//MessageBox.Show("Connection Open ! ");

while (filled == false)

{

adapter.Fill(ds);

filled = true;

}

//cnn.Close();

conn.Close();

}

catch (Exception)

{

MessageBox.Show("Can not open connection ! ");

string message = "Error in connection to datasource";

string caption = "Error";

MessageBoxButtons buttons = MessageBoxButtons.OK;

DialogResult result;

result = MessageBox.Show(message, caption, buttons);

}

database\_listbox.Items.Clear();

foreach (DataRow row in ds.Tables[0].Rows)

{

database\_listbox.Items.Add(row["Date"] + "\t\t" + row["Time"] + "\t\t" + row["Action"]);

}

}

This function connects to the MS Access database and displays all the data into the database\_listbox

private void insertdata(string action)

{

string dbconnection = "Provider=Microsoft.ACE.OLEDB.12.0;" + @"data source = ElevatorDatabase.accdb;";

string dbcommand = "insert into [Actions] ([Date],[Time],[Action]) values (@date, @time, @action)";

string date = DateTime.Now.ToShortDateString();

string time = DateTime.Now.ToLongTimeString();

database\_listbox.Items.Add(date + "\t\t" + time + "\t\t" + action);

OleDbConnection conn\_db = new OleDbConnection(dbconnection);

OleDbCommand comm\_insert = new OleDbCommand(dbcommand, conn\_db);

OleDbDataAdapter adapter\_insert = new OleDbDataAdapter(comm\_insert);

comm\_insert.Parameters.AddWithValue("@date", date);

comm\_insert.Parameters.AddWithValue("@time", time);

comm\_insert.Parameters.AddWithValue("@action", action);

conn\_db.Open();

comm\_insert.ExecuteNonQuery();

conn\_db.Close();

}

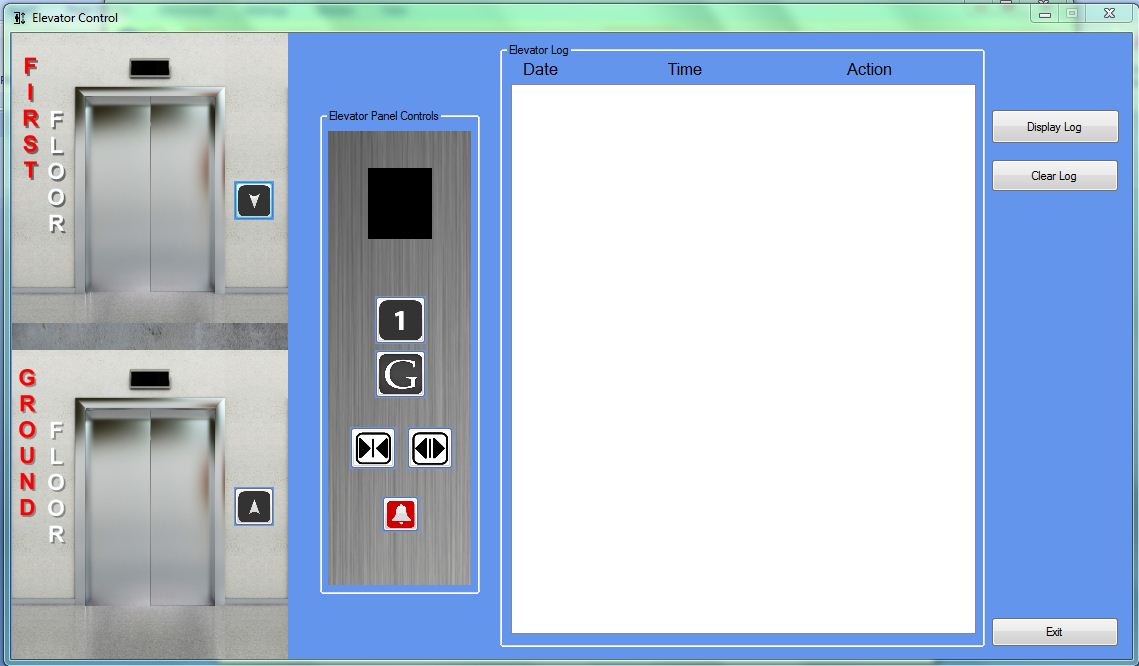
This function inserts data into the database and saves it.

The Form1.Designer.cs holds all the code for the User interface generated by the form designer.

## Testing

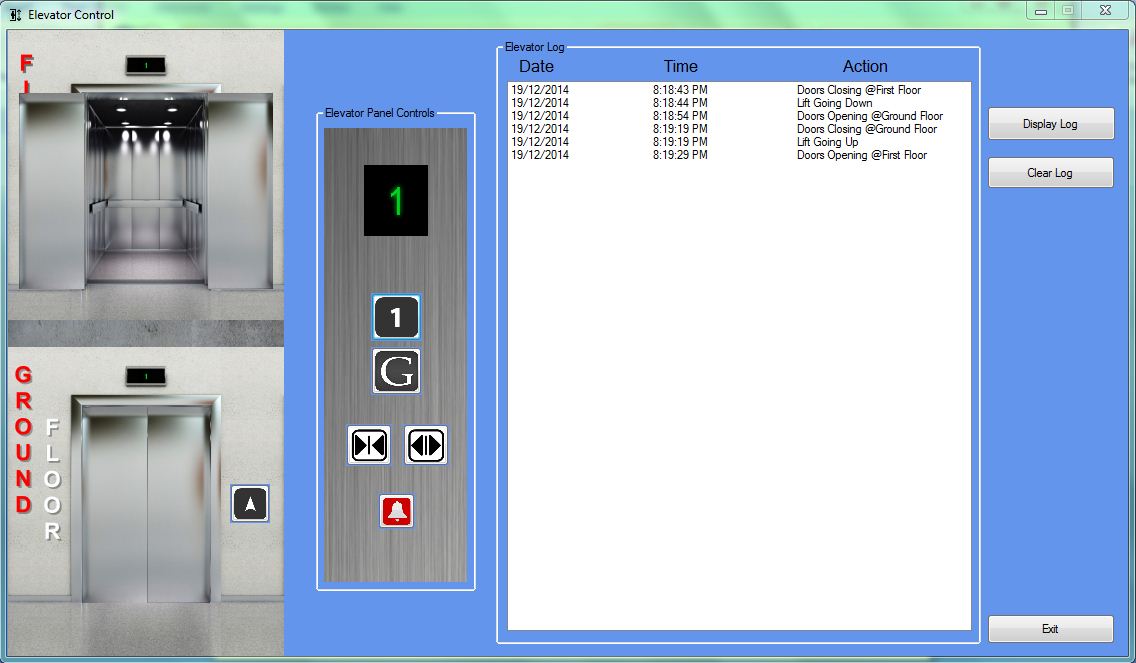
The finished program was run with the following operations:

When the program is executed it is displayed as this.

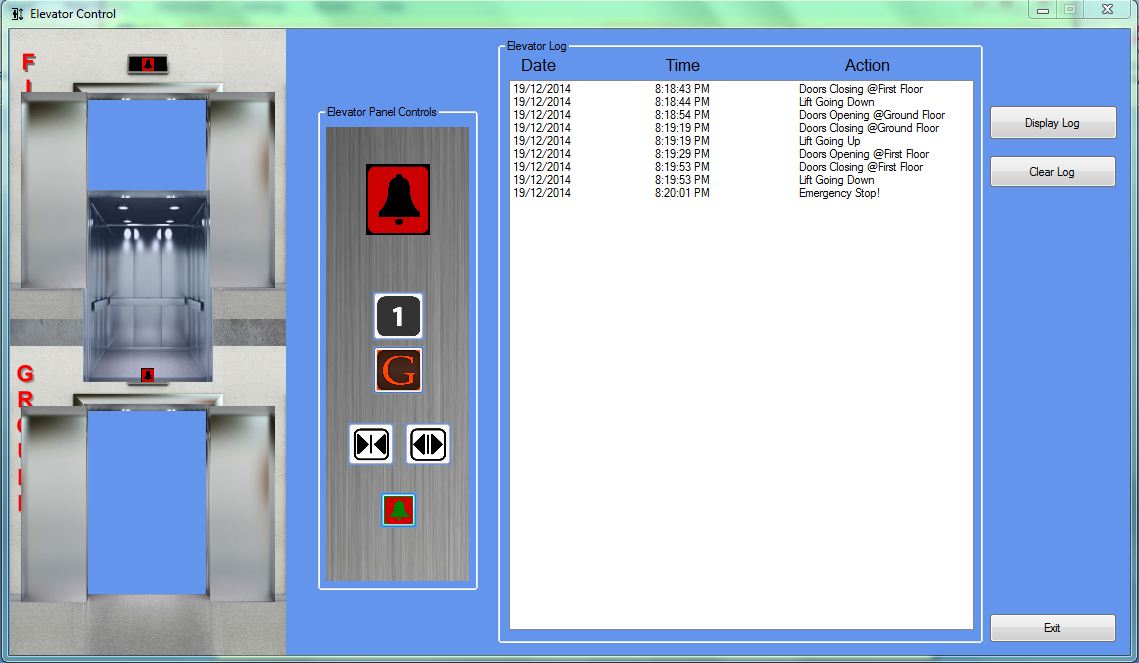




The Ground floor button was pressed from inside the lift on the control panel. The listbox then displayed data that was added to the database.



The first floor button was pressed to go to the first floor. Respective data was collected and stored in the database and viewed in the listbox.



The emergency button stops the lift movement and opens all the doors. A narrated voice is also accompanied.

