ASSIGNMENT 1: CONTROL AN ELEVATOR - A C# PROJECT

Unit title and code: Desktop Application Development and Software Engineering
2022-2023
CIS0116-2
Submitted by: Prajjwal Veer Basnet
Submitted to: University of Bedfordshire
University ID – 2028979

Table of Contents:

Table of Contents

Table of Figures:

Introduction:

This assignment focuses on simulating an elevator with its buttons and controls. It also focuses on keeping a log of all the buttons that has been used with its date, time, and the action that it performs.

Task Description:

A company wants to install an elevator in their office building of two floors. The company wants to implement an object-oriented software control application that manages all the controls of the elevator. The elevator should have an elevator car within which there is a control panel to control the elevator car. The control panel also contains a display window that shows the state of the elevator. There are two request buttons for the elevator with one on each floor.

The elevator has doors that opens up when the elevator car arrives at a particular floor. The door must remain closed while the elevator car is moving at all times so that passengers do not injure themselves against the elevator shaft.

The request buttons can be used to request elevator in both the floors. When the elevator is requested from any one of the floors, a built-in light of the button will be on, and the floor door will also open at the same time. The elevator will then move towards the destination floor.

When the elevator reaches its destination floor, it stops and opens the door for the passengers.

Aims and Objectives:

The aim of this assignment is to show the use of Object-Oriented Programming, using C# in .NET Framework. It focuses on how code reusability can be increased through Object-Oriented Programming and on other aspects of Object-Oriented Programming like encapsulation, polymorphism, inheritance, and abstraction. The assignment looks at all different aspects of Object-Oriented Programming and also focuses on improving the robustness as well as multi-threading.

Building an elevator requires different moving parts like control buttons, elevator car, doors, and different kinds of display boxes. All of these can be modelled and be created as a blueprint of different parts of the elevator using Object-Oriented Programming. Many functions in the elevator also have repetition and things like encapsulation helps to overcome this problem. The overall objective of this assignment is to use Object-Oriented techniques to solve a real-life problem and build a software. It also focuses on how the application can be optimized and increase its robustness by using BackgroundWorker and multi-threading.

Project Plan:

Week No.	Tasks
1	➢ Research topic for the use of technologies(Timers, Delegation, Picture Boxes, Object-Oriented Programming Basics, etc.) ➢ Create a GUI with different Picture Boxes and temporary buttons.
2	➢ Work on the function of the timers and sync it with the buttons. ➢ Add request button functions to each floor.
3	➢ Work on creating a control panel for the interior part of the lift. ➢ Create a display to show the status of the elevator.
4	➢ Create a database, table and all the necessary functions as required. ➢ Use the PostgreSQL plugin for C# and connected the database successfully.
5	➢ Create different functions for the database like Inserting Data, selecting to show the data in a table and deleting the data in the database. ➢ Add Validation to different buttons so that the buttons don’t malfunction and process the commands respectively.
6	➢ Finalize the project by adding different background images and optimizing the performance of the elevator using multi-threading.

Implementation:

Use Case Diagram:

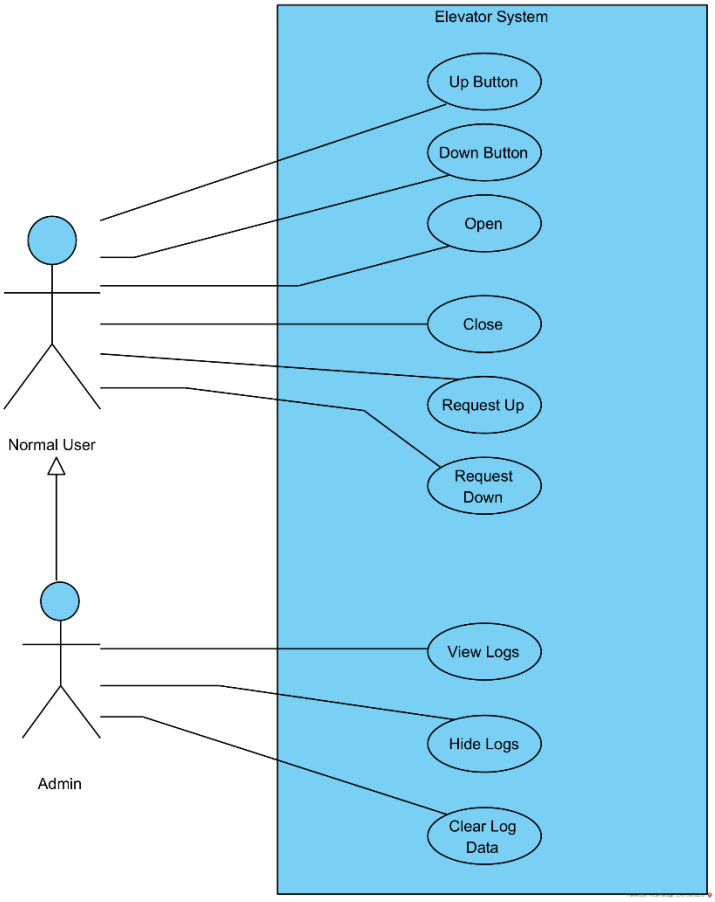


Figure 1: Use Case Diagram of Elevator

An elevator use case diagram was created to see in a general way how the application is supposed to be working. This shows that there will be a normal user who will use all the buttons that are available in the lift system. However, the viewing of logs and clearing logs are not an option available to everyone. This shows a case of generalization where the Admin can do all the functions that a normal user can and more functions like viewing/hiding logs and clearing logs.

Activity Diagram:

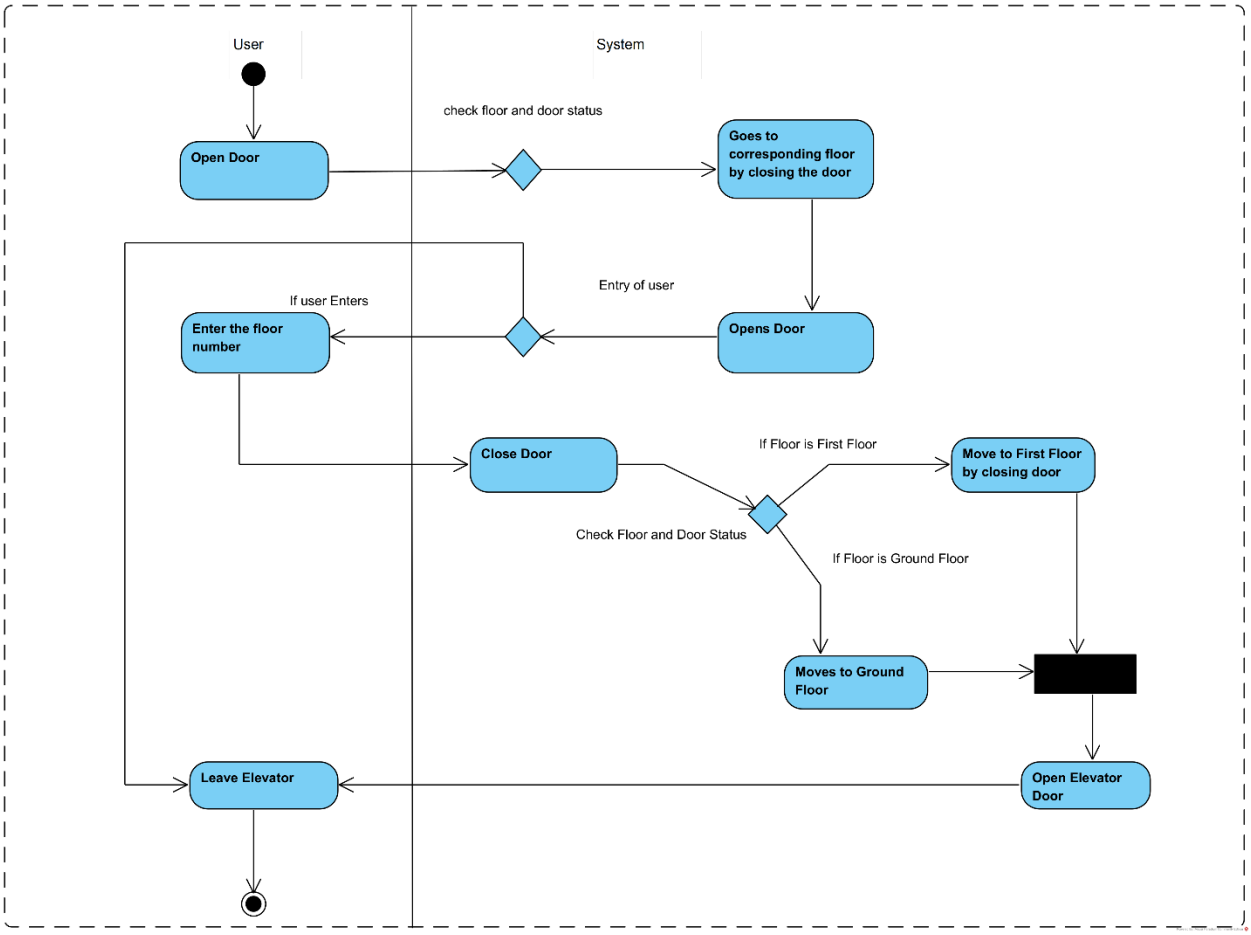


Figure 2: Activity Diagram of Elevator System

The activity diagram sees how a user can complete the actions that can be performed in the elevator. There are various buttons but in general, a normal person will open the door, get inside the elevator, and press the elevator button they want to go. When the elevator reaches the desired destination, the door is automatically opened.

Class Diagram:

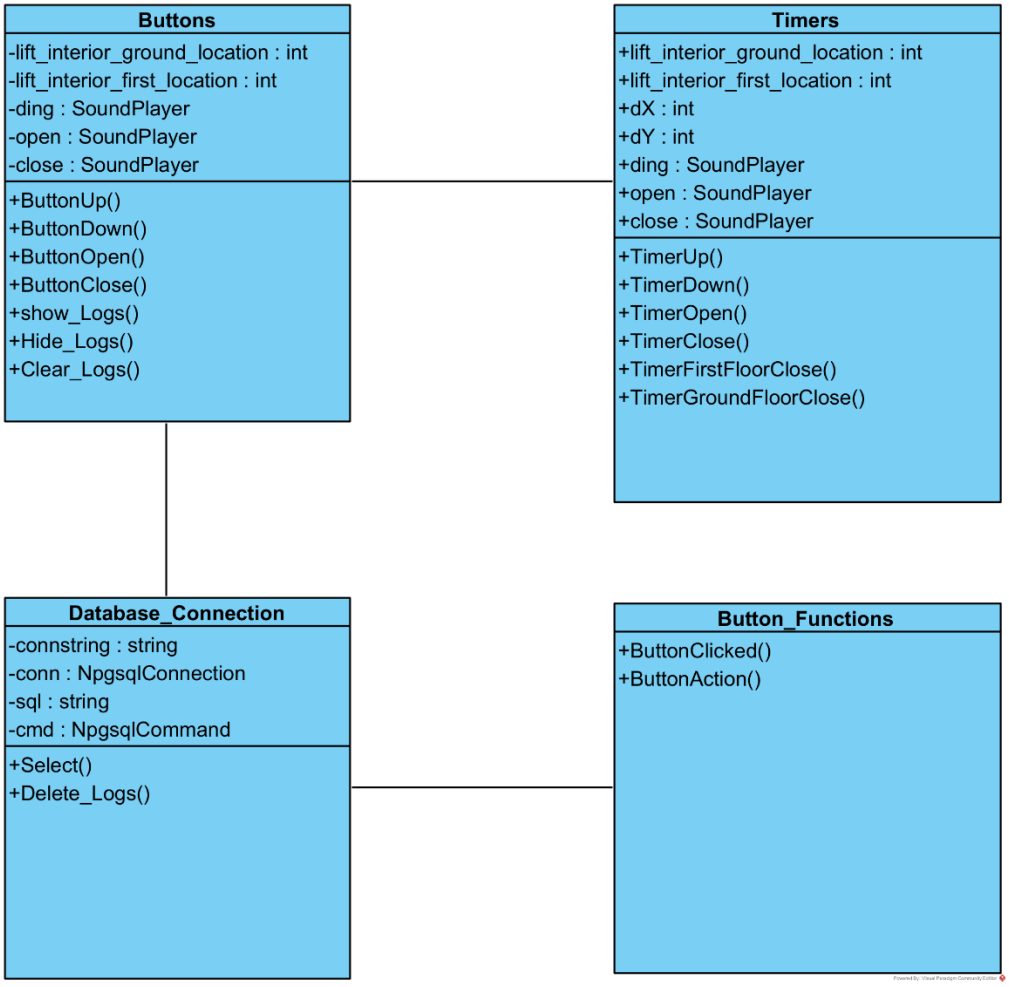


Figure 3: Class Diagram with all methods and attributes

A general class diagram is created for an overall plan layout of how different classes, attributes and methods can be separated and each of them can be called separately as per their need. Four Major Classes are present. The Buttons class manages the functions of all the buttons that are available for users. Timer class makes sure that all the internal commands of timers like moving the picture boxes are moved properly. The database connection class makes sure that whenever selection or deletion of data is required, it connects and executes the query.

Logical Database Design:

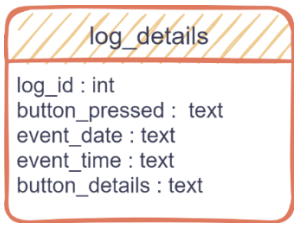
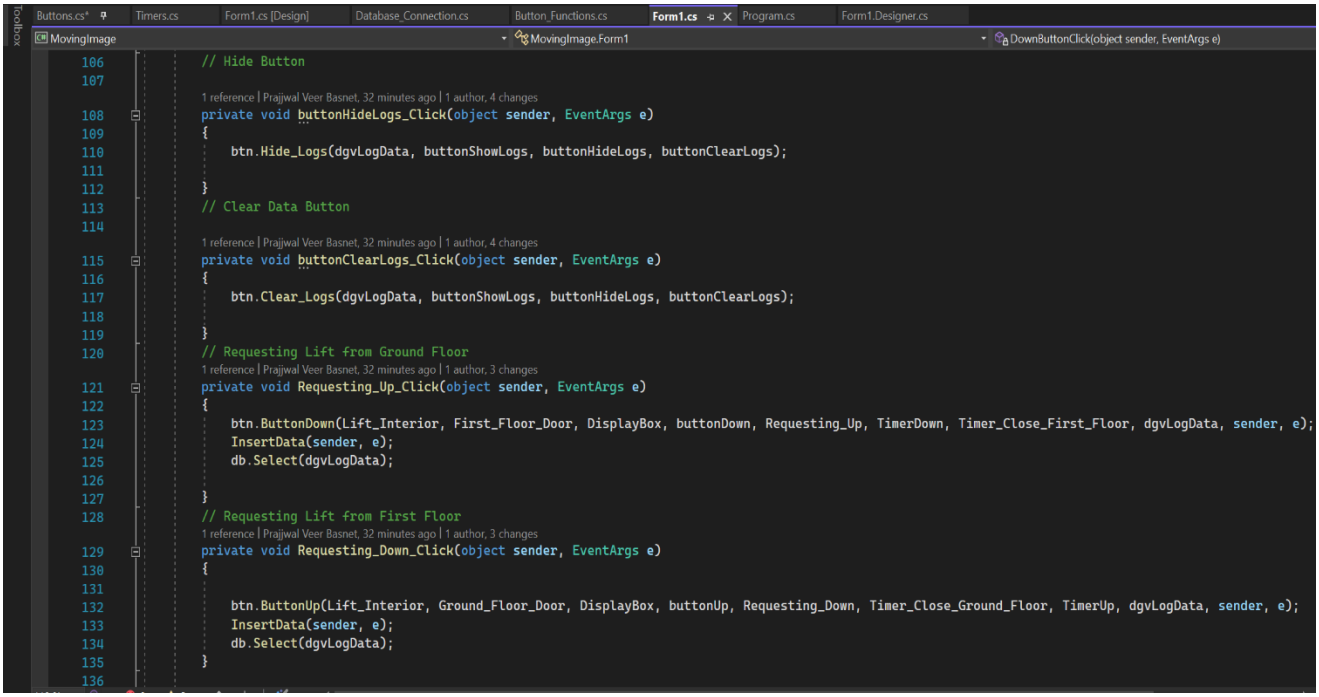


Figure 4: Logical Database Design

Language Used:

C# is used to create this elevator program in .NET Framework. The IDE used was Visual Studio 2022. C# is used due to its support for many different plugins that are used in this program like System.Speech is used for Text to Speech. System.Sound is used to simulate the sounds of the elevator when opening and closing the door.



```
// Hide Button

1 reference | Prajjwal Veer Basnet, 32 minutes ago | 1 author, 4 changes
private void buttonHideLogs_Click(object sender, EventArgs e)
{
    btn.Hide_Logs(dgvLogData, buttonShowLogs, buttonHideLogs, buttonClearLogs);

}
// Clear Data Button

1 reference | Prajjwal Veer Basnet, 32 minutes ago | 1 author, 4 changes
private void buttonClearLogs_Click(object sender, EventArgs e)
{
    btn.Clear_Logs(dgvLogData, buttonShowLogs, buttonHideLogs, buttonClearLogs);

}
// Requesting Lift from Ground Floor
1 reference | Prajjwal Veer Basnet, 32 minutes ago | 1 author, 3 changes
private void Requesting_Up_Click(object sender, EventArgs e)
{
    btn.ButtonDown(Lift_Interior, First_Floor_Door, DisplayBox, buttonDown, Requesting_Up, TimerDown, Timer_Close_First_Floor, dgvLogData, sender, e);
    InsertData(sender, e);
    db.Select(dgvLogData);

}
// Requesting Lift from First Floor
1 reference | Prajjwal Veer Basnet, 32 minutes ago | 1 author, 3 changes
private void Requesting_Down_Click(object sender, EventArgs e)
{

    btn.ButtonUp(Lift_Interior, Ground_Floor_Door, DisplayBox, buttonUp, Requesting_Down, Timer_Close_Ground_Floor, TimerUp, dgvLogData, sender, e);
    InsertData(sender, e);
    db.Select(dgvLogData);
}
```

Figure 5: Form1 Code being called through different classes.

Within C#, Object-Oriented Programming is used. Different classes are used for different parts within the programs. The buttons, timers and database are segregated from each other, and their objects are created to call from the form. The used of inheritance between Forms is also used. Different concepts of encapsulation are also used as access modifiers are specified for all kinds of functions in the program.

Database Used:

PostgreSQL was used for this program's database handling. PostgreSQL is an open-source relational database management system. The reason behind to choose PostgreSQL was due to its extensive support from the community and C# also supports PostgreSQL with the help of a plugin called plpgsql. A local database was created for this project to record all the logs of the elevator.

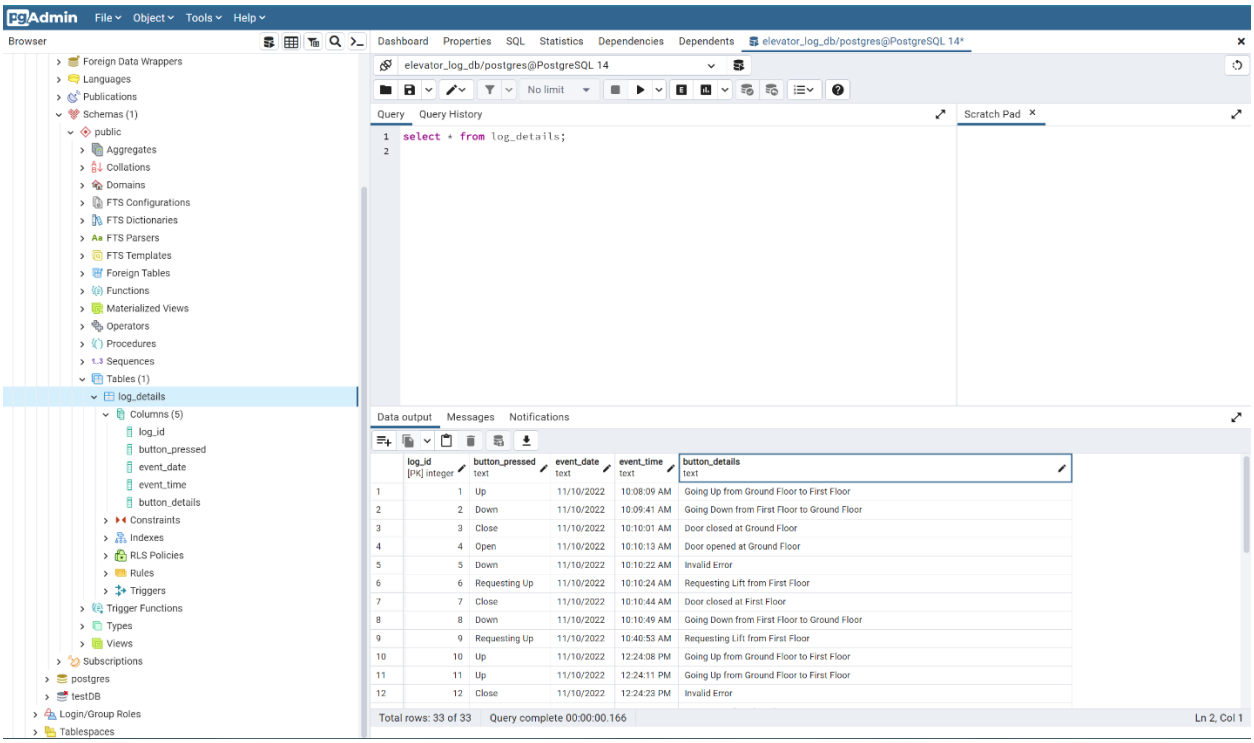


Figure 6: PostgreSQL Local Database

GUI Implementation:

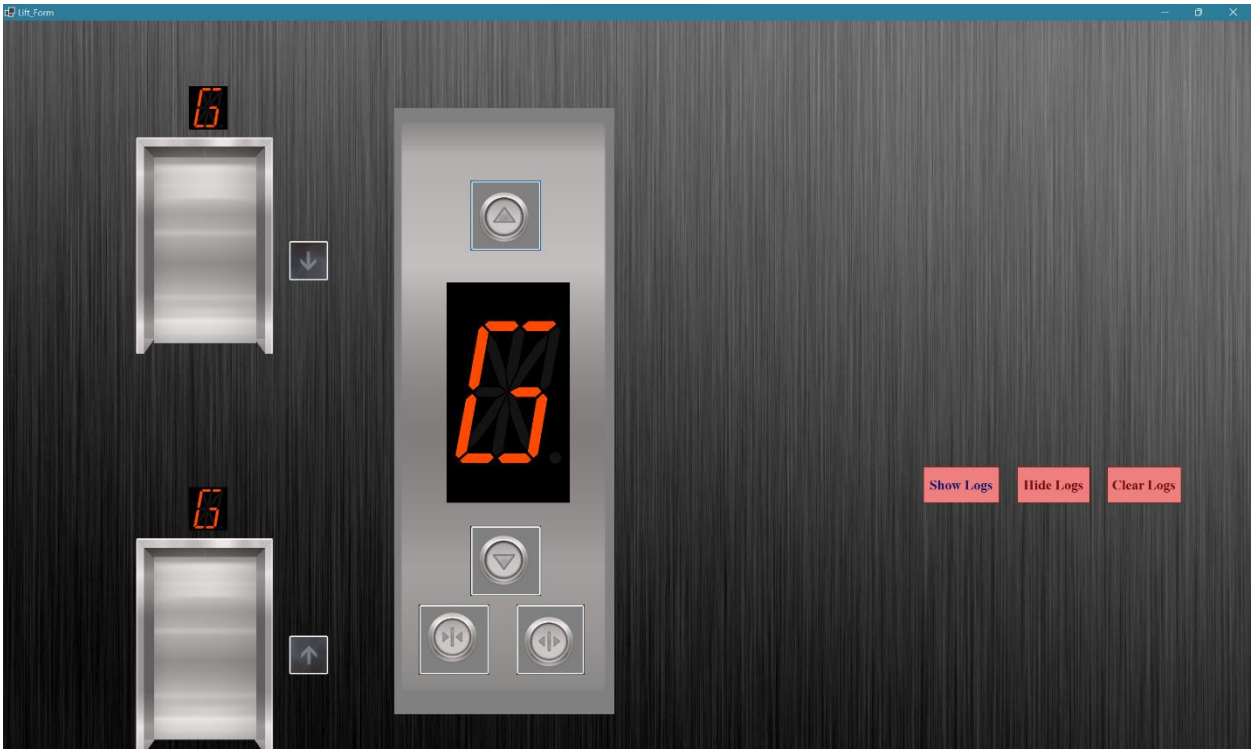


Figure 7: GUI Default Screen

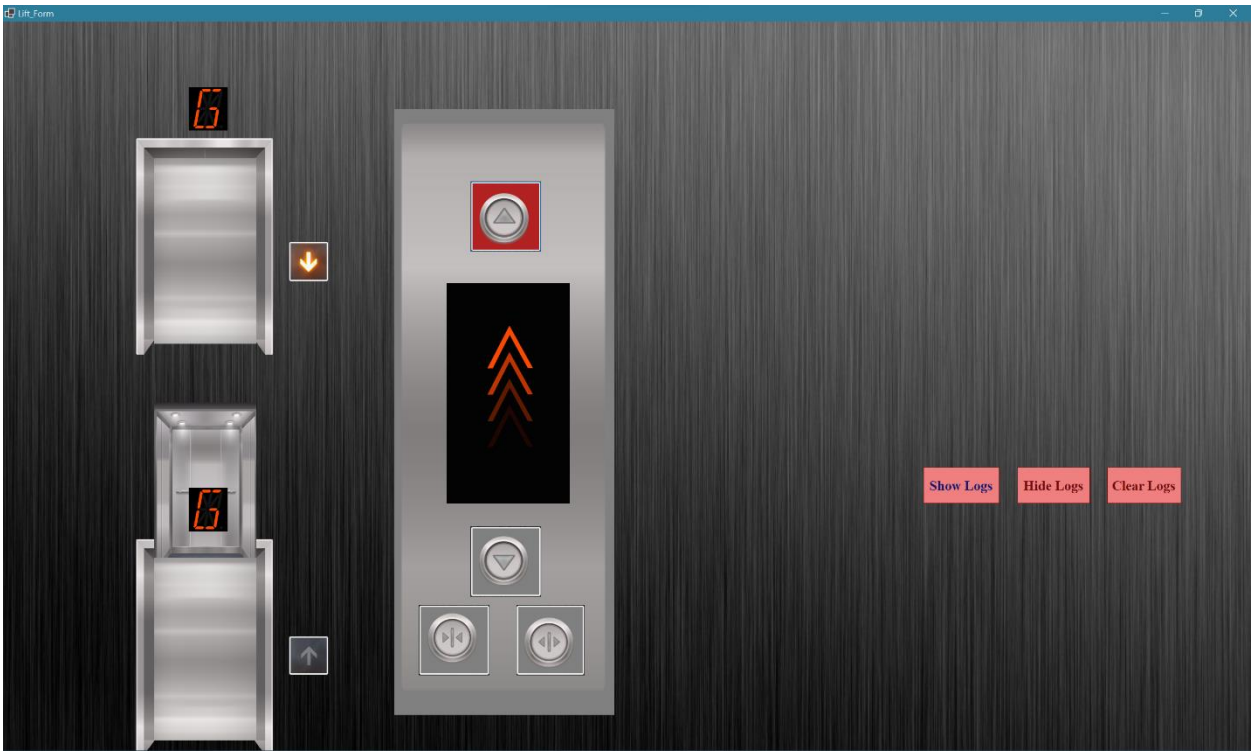


Figure 8: Elevator going up with arrow indication.

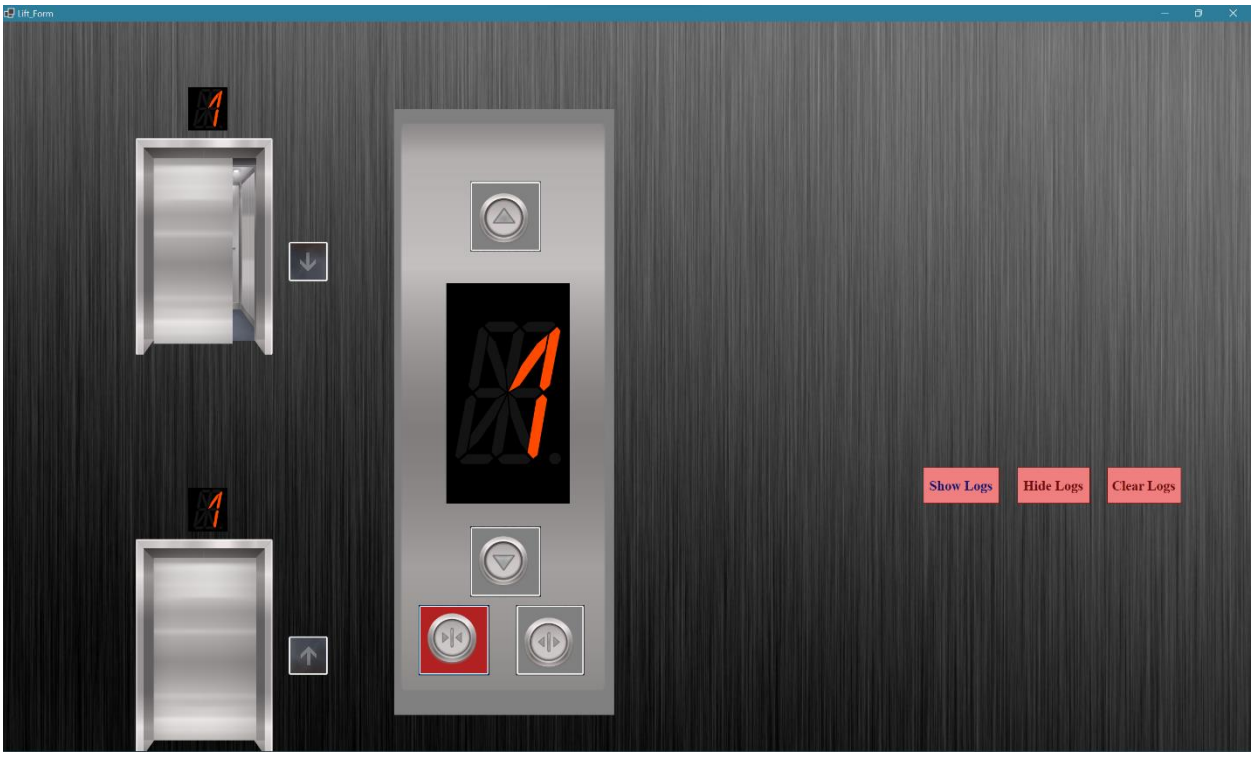


Figure 9: Elevator closing door.

Figure 10: Showing Log Table of the Elevator.

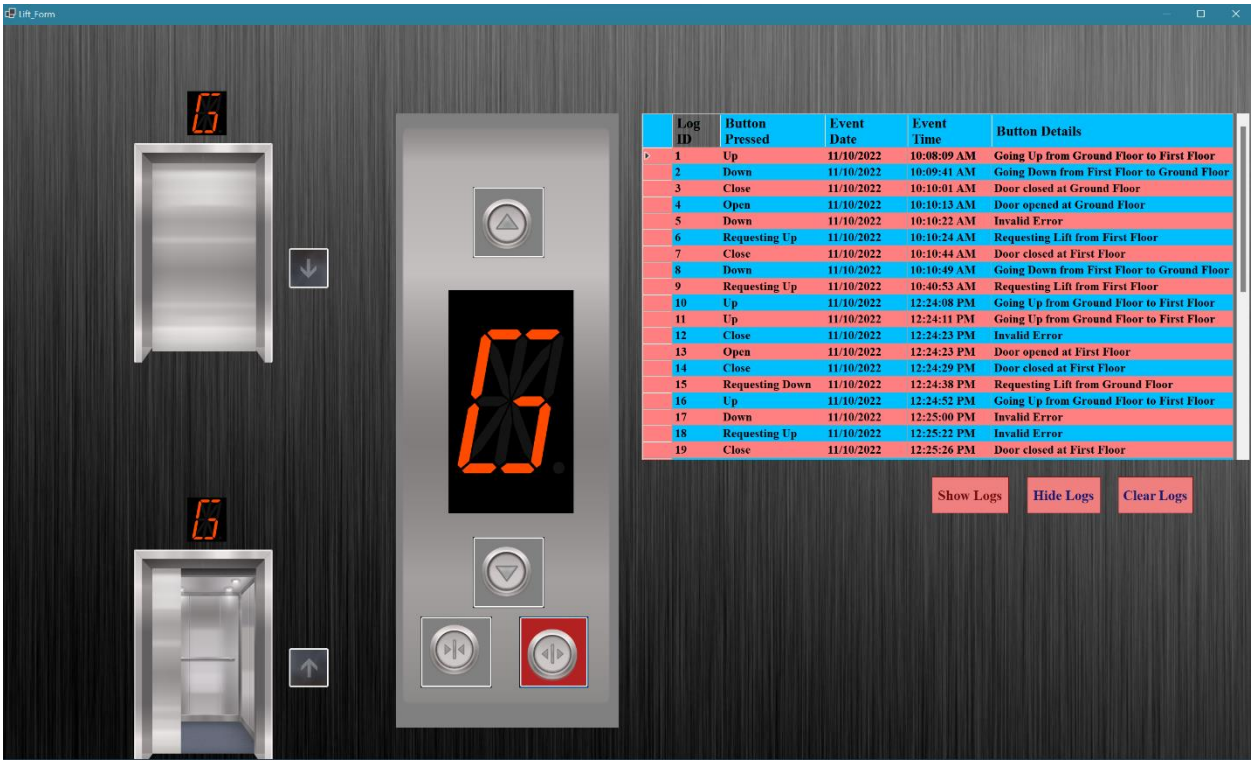


Figure 11: Opening door of elevator.

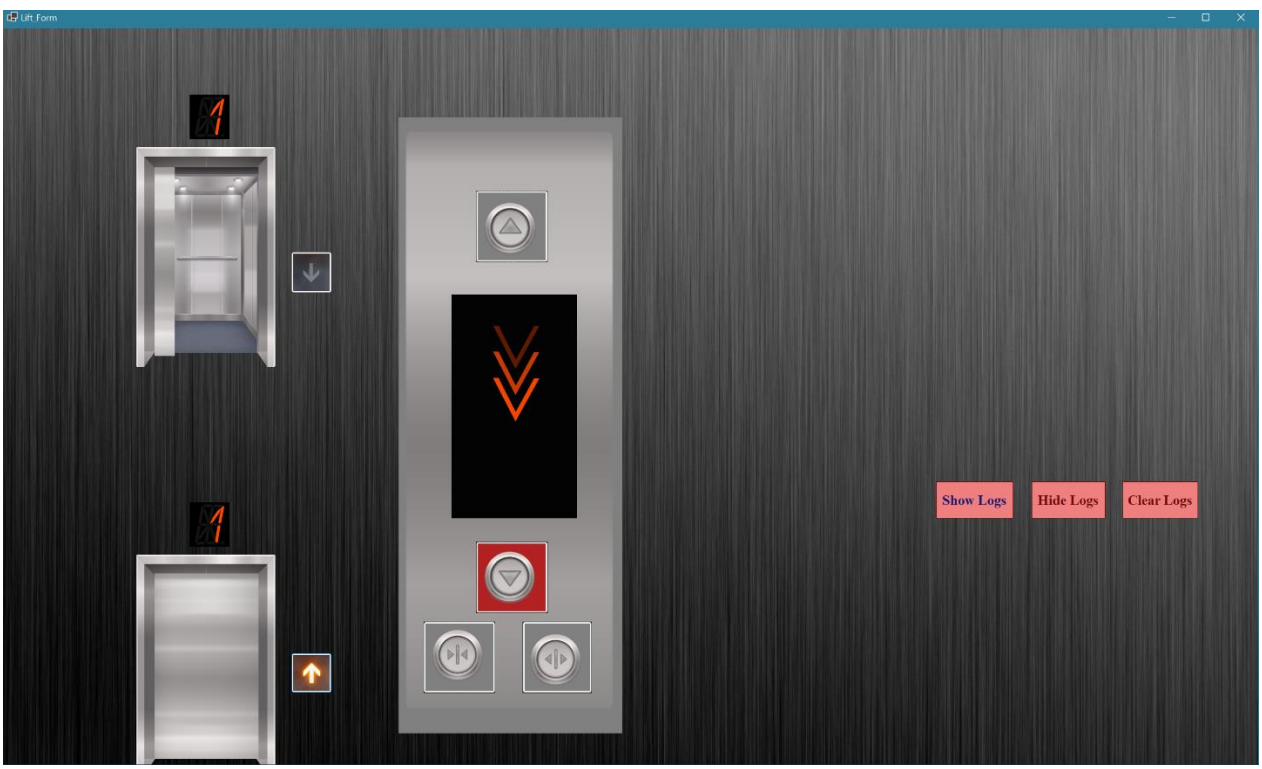


Figure 12: Elevator going down

Testing:

Introduction:

Many different strategies can be followed for testing this software. However, for this particular application methodical strategy will be used.

Scope:

In-Scope: Unit-Testing, Component Testing, UI Testing, Functional Testing, Black Box Testing

Out-Scope: Regression Testing, Alpha Testing, Non-Functional Testing, Performance and Load Testing.

Objectives of the Test:

- Ensuring the application shows desired output when the buttons are pressed.(Happy Path Testing).
- To get an overall idea of how many tasks are completed as mentioned in the task description.
- To identify bugs and issues and fix them before user testing.

Overview:

The test follows waterfall methodology as the software does not require much iteration and the tasks are clearly stated on what must be created. A requirement analysis and research were done on what tasks can be done and the tests were executed.

Bug Triage:

- To define all possible bugs and their proposed solutions.
- To create a priority list on the basis of major and minor functionality.

Resource and Environment Needs:

No Testing were used for the following test. However, there are hardware and software requirements needed.

- Windows 8 and above
- Office 2013 and above
- PostgreSQL
- Visual Studio 2022

Test Plan:

The elevator application is run in the latest Visual Studio 2022. The expected result for this application is that all the buttons' functions as intended with little to no bugs in the application. Different test cases are prepared to assess different situations within the application. This will check all the important functional testing as well as unit testing of various components in the elevator application. A table with the test name, its expected output and the result will also be shown. Any bugs observed during testing phase will have a bug ID and a bug status to denote if the bug has been resolved or not.

Test Cases:

Test Case ID – 1

Case Description- Testing the Up Button
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/07/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the up button in the elevator	The elevator car must move up	Shows expected results	Pass	N/A	No issues
3	Click the up button again in the elevator	The car should not move and provide a message.	Shows expected results	Pass	N/A	No issues

Test Case ID – 2

Case Description- Testing the Down Button
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/07/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the down button in the elevator	The car should not move and provide a message.	Shows expected results	Pass	N/A	No issues
3	Go up and click the down button in the elevator	The elevator car must move down	Shows expected results	Pass	N/A	No issues

Test Case ID – 3

Case Description- Testing the Open Button
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/07/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the open button in the elevator	The door must open on the ground floor.	Shows expected results	Pass	N/A	No issues
3	Go up and click the down button in the elevator	The door must open on the first floor.	Shows expected results	Pass	N/A	No issues

Test Case ID – 4

Case Description- Testing the Close Button
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/07/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the close button in the elevator	The door must close on the ground floor.	Shows expected results	Pass	N/A	No issues
3	Go up and click the open button in the elevator	The door must close on the first floor.	Shows expected results	Pass	N/A	No issues

Test Case ID - 5

Case Description- Request up from first floor
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/08/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the request up button in the elevator	The door must close on the ground floor and go up to the first floor	Shows expected results	Pass	01	Resolved Issue with indicators changing signs.

3	Click the Request up button in the elevator	The elevator car states that it is already at first floor	Shows expected results	Pass	N/A	No issues

Test Case ID - 6

Case Description- Request down from ground floor
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/08/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the request down button in the elevator	The elevator car states that it is already at ground floor	Shows expected results	Pass	02	Resolved Issue with indicators changing signs.
3	Go up and click the request down button in the elevator	The door must close on the first floor and go to the ground floor	Shows expected results	Pass	N/A	No issues

Test Case ID - 7

Case Description- Clicking more than one button
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/08/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed/Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the request down and up button at the same time.	The elevator must wait to complete the second button's function.	The elevator does complete the second button but has problems with syncing Text to Speech	Partial Pass	03	To be fixed
3	Go up and click the request down button in the elevator	The door must close on the first floor and go to the ground floor	Shows expected results	Pass	N/A	No issues

Test Case ID - 8

Case Description- Data entry in log table
Created and Reviewed By – Prajjwal

Tester's Name – Prajjwal

Date – 11/08/2022

Prerequisites : N/A

Test Data – None Required.

Test Case: Pass

Step	Step Details	Expected Results	Actual Results	Pass/Fail/Not Executed/Suspended	Bug	Bug Status
1	Open the Elevator Application	The application should open without hesitation.	Shows expected results	Pass	N/A	No issues
2	Click the request show logs button.	The table must show all the history of the buttons pressed with their details.	Shows expected results.	Pass	N/A	No issues
3	Click hide logs	The table must hide	Shows expected results	Pass	N/A	No issues
4	Click show logs and clear the logs	The logs must clear all the data.	Shows expected results.	Pass	N/A	No issues

Bug Reports:

During UI Testing, some of the bugs were noticed during the program that were not seen during the Functional Testing. They are listed as follows:

- When pressing two buttons at once although the button queues another function, it enters it in the database before the function is executed.
- Data grid view has some problem while showing data due to background images and background color of cells.
- The queuing procedure of the elevator is not well optimized however, the program does not halt at any point.

Conclusion:

The application focuses on implementing the Object-Oriented Programming concept and contains all the basic functions of an elevator. The program features two different floors with up, down, open, and close buttons as well as calling requesting buttons for each floor.

Due to time constraint, programming such an application was a challenge with a lot of features but most of the part was completed successfully in the project. The assignment enlarges our knowledge in OOP and teaches us the basic implementation of OOPs.

References:

- Atlassian (no date) The different types of testing in software, Atlassian. Atlassian. Available at: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing (Accessed: November 11, 2022).
- The Complete Guide to different types of testing (no date) Perfecto by Perforce. Perforce . Available at: https://www.perfecto.io/resources/types-of-testing (Accessed: November 11, 2022).
- Hamilton, D. (2022) How to create junit parameterized tests, Parasoft. Parasoft. Available at: https://www.parasoft.com/blog/how-to-create-junit-parameterized-tests/ (Accessed: November 11, 2022).
- Manager, D.C.S.C. (2021) 15 functional testing types with examples, Applause. Applause App Quality, Inc. Available at: https://www.applause.com/blog/functional-testing-types-examples (Accessed: November 11, 2022).
- Test strategy - javatpoint (no date) www.javatpoint.com. www.javatpoint.com. Available at: https://www.javatpoint.com/test-strategy (Accessed: November 11, 2022).

Marking Matrix with Assessment:

Task Number	Sub-tasks	Possible Marks	Self-assessment (completed Yes/No)	Reference to your testing report	Mark Awarded
Task 1	Complete GUI for Task 1	10	Yes	Created GUI in Windows Forms	10
	Skeleton of event handlers in place for all buttons	10	Yes	Created Buttons in Control panel as well as request buttons for lifts	10
Task 2	All event handlers are functional	10	Yes	All handlers are functional	10
Task 3	Database (DB) is designed and can be connected	5	Yes	DB connects PostgreSQL successfully	5
	Log Information can be retrieved from DB and displayed in the GUI	5	Yes	Data shows up in DataGrid View	5
	When the log button is pressed, log information is sent to and stored in the DB	5	Yes	The log shows the data in the DB as well as shows the data in GUI	5
	Use the disconnected model rather than connected model (Data source is updated via DataAdapters Update() method instead of ExecuteNonQuery() method)	5	Partial Yes	Used PostgreSQL for connection	2.5
	Using relative path instead of absolute path	5	Yes	Used Relative Paths for all Images by uploading in resources	5

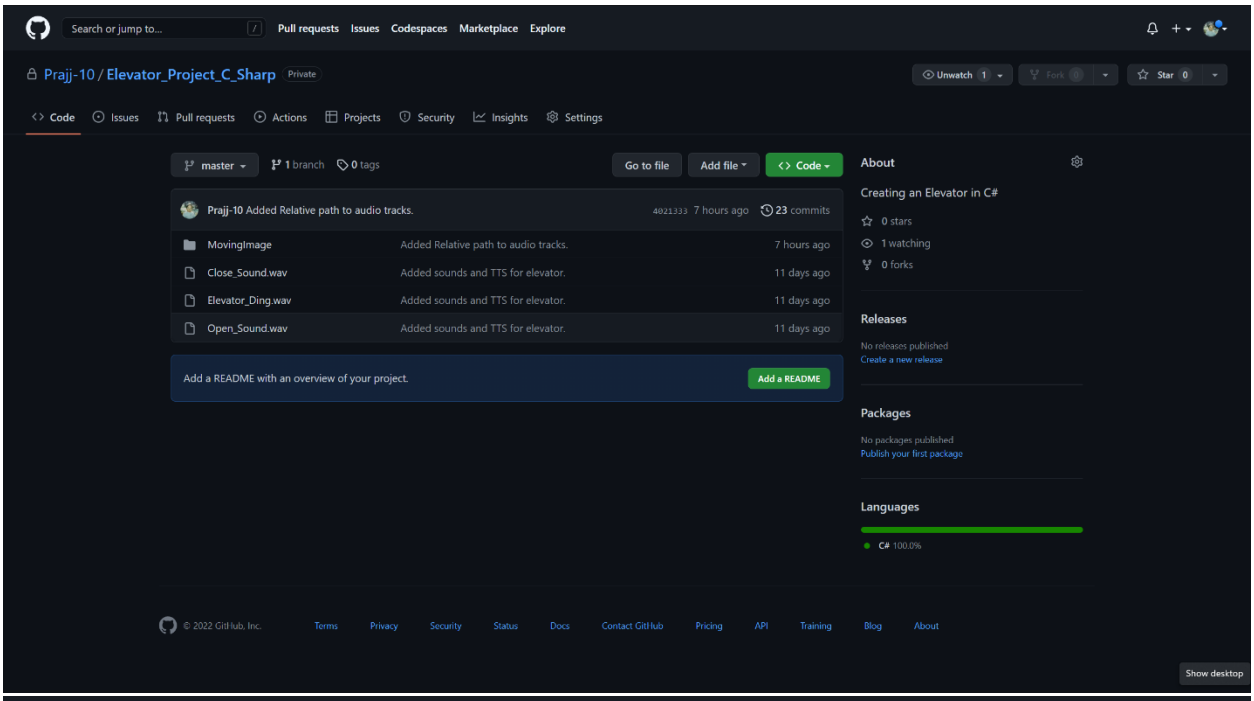
	Avoiding any duplication among the event handlers over the database related functions	5	Yes	No duplication of data among event handlers	5
Task 4	Events described in Task 2 animated using delegation and timer	10	Yes	Lift moves and simulates real lift with delegation and timers	10
Task 5	Eliminating logical errors and handling exceptions with try and catch	5	Yes	Handles all the basic logical errors as well as catches the database errors.	5
	Optimise the efficiency of GUI by implementing multiple tasks concurrently via BackgroundWorker	5	Yes	Used BackgroundWorker	5
	Use state patterns instead of if-else statements to accommodate future changes of the requirement	10	No	Not used	5
Task 6	Testing report	10	Yes		10
Total		100			92.5

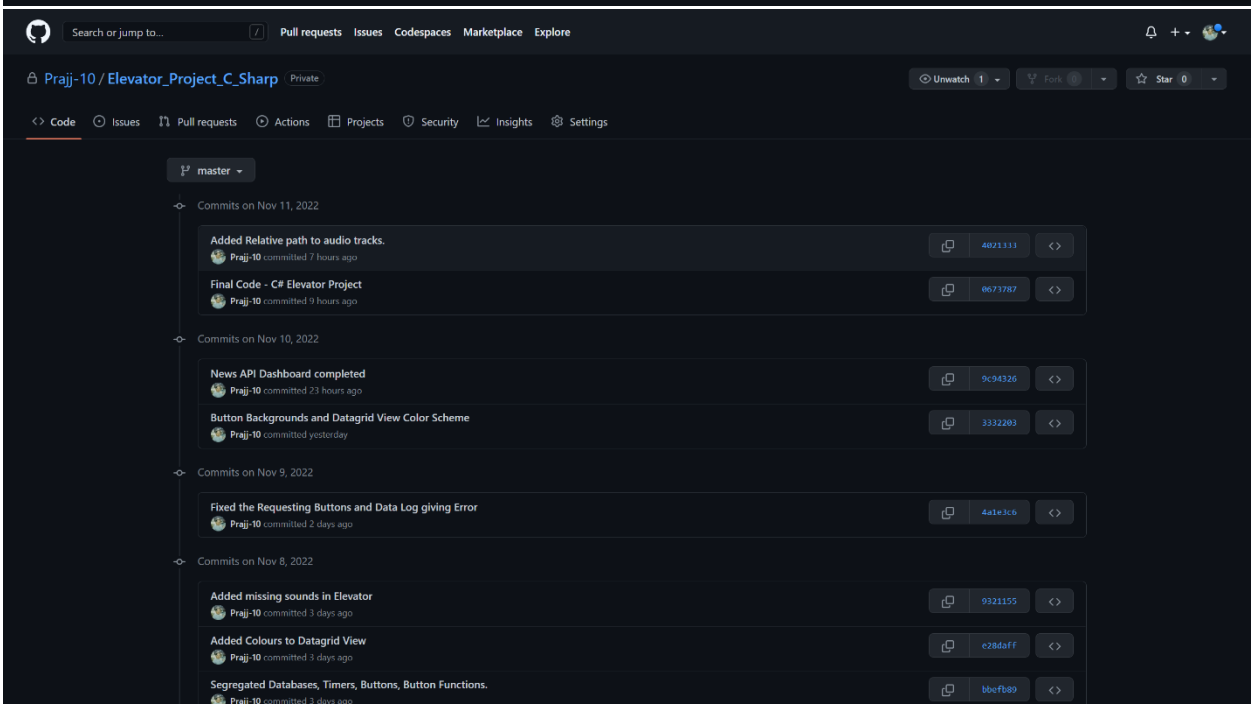
Appendix:

Video of Code and Demonstration of the elevator:

https://mega.nz/file/S6QjgAqb#x1hEr1gD0EvFsQL92qpJY30X610JP7IBZkrLV6Sd7ag

GitHub Commits:





Buttons.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.NetworkInformation;
using System.Speech.Synthesis;
using System.Text;
using System.Threading.Tasks;
using Timer = System.Windows.Forms.Timer;

namespace MovingImage
{
    internal class Buttons
    {
        // Lift Interior Coordinates
        int lift_interior_ground_location = 1098;
        int lift_interior_first_location = 280;

        // Speech Synthesizer

        SpeechSynthesizer synthesizer = new SpeechSynthesizer();

        // Database Connection class object created

        Database_Connection db = new Database_Connection();

        // Default Sounds for Opening, Closing and Ding sounds

        System.Media.SoundPlayer ding = new
System.Media.SoundPlayer(Properties.Resources.Elevator_Ding);
        System.Media.SoundPlayer open = new
System.Media.SoundPlayer(Properties.Resources.Open_Sound);
        System.Media.SoundPlayer close = new
System.Media.SoundPlayer(Properties.Resources.Close_Sound);

        public void ButtonUp(PictureBox Lift_Interior, PictureBox
Ground_Floor_Door, PictureBox DisplayBox, Button buttonUp,Button
Requesting_Down, Timer Timer_Close_Ground_Floor, Timer TimerUp, DataGridView
dgvLogData, object sender, EventArgs e)
        {
            // Synthesizer Settings
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                // Changes the colour of Buttons
                buttonUp.BackColor = Color.Firebrick;
                Requesting_Down.Image = Properties.Resources.Down_Light;
                ding.Play();
                synthesizer.Speak("Going Up.");
                // If the door is not closed
                if (Ground_Floor_Door.Size.Width == 0)
                {
                    Timer_Close_Ground_Floor.Enabled = true;
                    close.Play();
```

```
                db.Select(dgvLogData);
                DisplayBox.Image = Properties.Resources.Arrow_Up;



            }
            else
            {
                // if the door is closed
                TimerUp.Enabled = true;

                db.Select(dgvLogData); ;
                DisplayBox.Image = Properties.Resources.Arrow_Up;


            }
        }
        else
        {
            synthesizer.Speak("You are already at First Floor. ");
        }

    }
    // Button Function for Lift going down
    public void ButtonDown(PictureBox Lift_Interior, PictureBox
First_Floor_Door, PictureBox DisplayBox,Button buttonDown, Button
Requesting_Up, Timer TimerDown, Timer Timer_Close_First_Floor, DataGridView
dgvLogData, object sender, EventArgs e)
    {
        // Synthesizer Settings
        synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

        if (Lift_Interior.Location.Y == lift_interior_first_location)
        {
            // Changes the colour of Buttons
            buttonDown.BackColor = Color.Firebrick;
            Requesting_Up.Image = Properties.Resources.Up_Light;
            ding.Play();
            synthesizer.Speak("Going Down.");

            // If the door is not closed
            if (First_Floor_Door.Size.Width == 0)
            {
                close.Play();
                Timer_Close_First_Floor.Enabled = true;


                db.Select(dgvLogData);
                DisplayBox.Image = Properties.Resources.Arrow_Down;

            }
            else
            {
                // if the door is closed
                TimerDown.Enabled = true;
```

```
                    db.Select(dgvLogData);
                    DisplayBox.Image = Properties.Resources.Arrow_Down;

                }
            }
            else
            {
                synthesizer.Speak("You are already at Ground Floor.");
            }
        }
        // Open Button
        public void ButtonOpen(PictureBox Lift_Interior, PictureBox
Ground_Floor_Door, PictureBox First_Floor_Door, Button buttonOpen, Timer
TimerOpen, DataGridView dgvLogData, object sender, EventArgs e )
        {
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                if (Ground_Floor_Door.Size.Width == 0)
                {
                    synthesizer.SpeakAsync("Door is already open ! ");

                }
                else
                {

                    buttonOpen.BackColor = Color.Firebrick;
                    synthesizer.Speak("Opening Door.");
                    Thread.Sleep(200);
                    open.Play();

                    TimerOpen.Enabled = true;
                    // db.InsertData(sender, e);
                    db.Select(dgvLogData);
                }
            }
            else if (Lift_Interior.Location.Y ==
lift_interior_first_location)
            {
                if (First_Floor_Door.Size.Width == 0)
                {
                    synthesizer.SpeakAsync("Door is already open ! ");

                }
                else
                {

                    buttonOpen.BackColor = Color.Firebrick;
                    synthesizer.Speak("Opening Door.");
                    Thread.Sleep(200);
                    open.Play();

                    TimerOpen.Enabled = true;
                    // db.InsertData(sender, e);
```

```
                    db.Select(dgvLogData);
                }
            }
        }

        // Close Button
        public void ButtonClose(PictureBox Lift_Interior, PictureBox
Ground_Floor_Door, PictureBox First_Floor_Door, Button buttonClose, Timer
TimerClose, DataGridView dgvLogData, object sender, EventArgs e)
        {
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                if (Ground_Floor_Door.Size.Width == 210)
                {
                    synthesizer.Speak("Door is already Closed !");
                }
                else
                {
                    buttonClose.BackColor = Color.Firebrick;
                    synthesizer.Speak("Closing Door.");
                    Thread.Sleep(200);
                    close.Play();

                    TimerClose.Enabled = true;
                    // db.InsertData(sender, e);
                    db.Select(dgvLogData);

                }
            }
            else if (Lift_Interior.Location.Y ==
lift_interior_first_location)
            {
                if (First_Floor_Door.Size.Width == 210)
                {
                    synthesizer.Speak("Door is already Closed !");
                }
                else
                {
                    buttonClose.BackColor = Color.Firebrick;
                    synthesizer.Speak("Closing Door.");
                    Thread.Sleep(200);
                    close.Play();

                    TimerClose.Enabled = true;
                    // db.InsertData(sender, e);
                    db.Select(dgvLogData);

                }
            }
        }

        // Show Logs Button
        public void Show_Logs(DataGridView dgvLogData, Button buttonShowLogs,
Button buttonHideLogs, Button buttonClearLogs)
```

```
        {
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);
            dgvLogData.Visible = true;
            db.Select(dgvLogData);
            buttonShowLogs.Enabled = false;
            buttonHideLogs.Enabled = true;
            buttonClearLogs.Enabled = true;
        }

        // Hiding Logs Button

        public void Hide_Logs(DataGridView dgvLogData, Button buttonShowLogs,
Button buttonHideLogs, Button buttonClearLogs)
        {
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);
            dgvLogData.Visible = false;
            buttonHideLogs.Enabled = false;
            buttonShowLogs.Enabled = true;
            buttonClearLogs.Enabled = false;
        }

        // Clear Logs Button
        public void Clear_Logs(DataGridView dgvLogData, Button
buttonShowLogs, Button buttonHideLogs, Button buttonClearLogs)
        {
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);
            db.Delete_Logs();
            db.Select(dgvLogData);
            buttonClearLogs.Enabled = false;
            buttonHideLogs.Enabled = true;
            buttonShowLogs.Enabled = true;
        }

    }
}
```

Form1.cs

```
using System.Diagnostics.Metrics;
using System.Timers;
```

```
using System;
using System.Speech.Synthesis;
using Npgsql;
using System.Data;
using System.ComponentModel;
using System.Threading;
using System.Reflection.Metadata.Ecma335;
using System.Windows.Forms;

namespace MovingImage
{

    public partial class Form1 : Form
    {

        // Object Declaration of Timers
        Timers tm = new Timers();

        // Object Declaration of Database_Connection
        Database_Connection db = new Database_Connection();

        // Strings created for Database to insert data
        private string connstring = String.Format("Server={0}; Port={1};" +
            "User Id={2}; Password={3}; Database={4};",
            "localhost", 5432, "postgres", "12345", "elevator_log_db");
        private NpgsqlConnection conn;
        private string sql;
        private NpgsqlCommand cmd;

        // Object Declaration for Buttons

        Buttons btn = new Buttons();

        // Object Declaration for Button_Functions

        Button_Functions buttonF = new Button_Functions();

        static System.Windows.Forms.Timer myTimer = new
System.Windows.Forms.Timer();


        public Form1()
        {
            InitializeComponent();

            dgvLogData.Visible = false;
            dgvLogData.DefaultCellStyle.SelectionBackColor =
dgvLogData.DefaultCellStyle.BackColor;
            dgvLogData.DefaultCellStyle.SelectionForeColor =
dgvLogData.DefaultCellStyle.ForeColor;

        }

        // Up Button Click Event
        public void UpButtonClick(object sender, EventArgs e)
        {
```

```
            btn.ButtonUp(Lift_Interior, Ground_Floor_Door, DisplayBox,
buttonUp, Requesting_Down, Timer_Close_Ground_Floor, TimerUp, dgvLogData,
sender, e);
            InsertData(sender, e);
            db.Select(dgvLogData);


        }

        // Down Button Click Event

        private void DownButtonClick(object sender, EventArgs e)
        {

            btn.ButtonDown(Lift_Interior, First_Floor_Door, DisplayBox,
buttonDown, Requesting_Up, TimerDown, Timer_Close_First_Floor, dgvLogData,
sender, e);
            InsertData(sender, e);
            db.Select(dgvLogData);



        }
        // Open Button Click Event

        private void OpenButtonClick(object sender, EventArgs e)
        {

            btn.ButtonOpen(Lift_Interior, Ground_Floor_Door,
First_Floor_Door, buttonOpen, TimerOpen, dgvLogData, sender, e);
            InsertData(sender, e);
            db.Select(dgvLogData);

        }

        // Close Button Click Event

        private void CloseButtonClick(object sender, EventArgs e)
        {

            btn.ButtonClose(Lift_Interior, Ground_Floor_Door,
First_Floor_Door, buttonClose, TimerClose, dgvLogData, sender, e);
            InsertData(sender, e);
            db.Select(dgvLogData);

        }
        private void buttonShowLogs_Click(object sender, EventArgs e)
        {
            btn.Show_Logs(dgvLogData, buttonShowLogs, buttonHideLogs,
buttonClearLogs);

            // To unselect the selected values by default

            dgvLogData.DefaultCellStyle.SelectionBackColor =
dgvLogData.DefaultCellStyle.BackColor;
            dgvLogData.DefaultCellStyle.SelectionForeColor =
dgvLogData.DefaultCellStyle.ForeColor;
```

```
        }
        // Hide Button

        private void buttonHideLogs_Click(object sender, EventArgs e)
        {
            btn.Hide_Logs(dgvLogData, buttonShowLogs, buttonHideLogs,
buttonClearLogs);

        }
        // Clear Data Button

        private void buttonClearLogs_Click(object sender, EventArgs e)
        {
            btn.Clear_Logs(dgvLogData, buttonShowLogs, buttonHideLogs,
buttonClearLogs);

        }
        // Requesting Lift from Ground Floor
        private void Requesting_Up_Click(object sender, EventArgs e)
        {
            btn.ButtonDown(Lift_Interior, First_Floor_Door, DisplayBox,
buttonDown, Requesting_Up, TimerDown, Timer_Close_First_Floor, dgvLogData,
sender, e);
            InsertData(sender, e);
            db.Select(dgvLogData);

        }
        // Requesting Lift from First Floor
        private void Requesting_Down_Click(object sender, EventArgs e)
        {

            btn.ButtonUp(Lift_Interior, Ground_Floor_Door, DisplayBox,
buttonUp, Requesting_Down, Timer_Close_Ground_Floor, TimerUp, dgvLogData,
sender, e);
            InsertData(sender, e);
            db.Select(dgvLogData);
        }

        // Timers

        // TimerUp
        private void TimerUp_Tick(object sender, EventArgs e)
        {
            tm.TimerUp(Lift_Interior, DisplayBox, Requesting_Down,
DisplayBox_Ground_Floor, DisplayBox_First_Floor, TimerUp, TimerOpen,
buttonUp);
        }

        // TimerDown
        private void TimerDown_Tick(object sender, EventArgs e)
        {
            tm.TimerDown(Lift_Interior, DisplayBox, Requesting_Down,
DisplayBox_First_Floor, DisplayBox_Ground_Floor, TimerDown, TimerOpen,
buttonDown);
        }
        // TimerOpen
        private void TimerOpen_Tick(object sender, EventArgs e)
```

```
        {
            tm.TimerOpen(Lift_Interior, Ground_Floor_Door, First_Floor_Door,
TimerOpen, buttonOpen);

        }
        // TimerClose
        private void TimerClose_Tick(object sender, EventArgs e)
        {
            tm.TimerClose(Lift_Interior, Ground_Floor_Door, First_Floor_Door,
TimerClose, buttonClose);
        }
        // Timer for 1st Floor
        private void Timer_Close_1F_Tick(object sender, EventArgs e)
        {
            tm.TimerFirstFloorClose(Lift_Interior, First_Floor_Door,
Timer_Close_First_Floor, TimerDown, buttonDown, Requesting_Up,
DisplayBox_Ground_Floor, DisplayBox_First_Floor);
        }
        // Timer for Ground Floor
        private void Timer_Close_Ground_Floor_Tick(object sender, EventArgs
e)
        {
            tm.TimerGroundFloorClose(Lift_Interior, Ground_Floor_Door,
Timer_Close_Ground_Floor, TimerUp, buttonUp, Requesting_Down);

        }

        // Form 1 loading prerequisites
        private void Form1_Load_1(object sender, EventArgs e)
        {
            // Establishes connection with database anywhere the insert data
is done.
            conn = new NpgsqlConnection(connstring);

            // Height and Width for the Form
            int w = Screen.PrimaryScreen.Bounds.Width;
            int h = Screen.PrimaryScreen.Bounds.Height;
            this.Location = new Point(0, 0);
            this.Size = new Size(w, h);
        }

        // Insert Details to Db when button is clicked
        private void InsertData(object sender, EventArgs e)
        {
            try
            {
                string button = buttonF.ButtonClicked((Button)sender, e);
                string action = buttonF.ButtonAction((Button)sender, e,
Lift_Interior, TimerUp, Timer_Close_Ground_Floor, TimerDown,
Timer_Close_First_Floor, TimerOpen, TimerClose);
                conn.Open();
                sql = String.Format("select * from log_insert('{0}', '{1}',
'{2}', '{3}');", button, DateTime.Now.ToString("MM/dd/yyyy"),
DateTime.Now.ToString("h:mm:ss tt"), action);
                cmd = new NpgsqlCommand(sql, conn);

                cmd.ExecuteNonQuery();
```

```
                conn.Close();

            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Error : " + ex.Message);


            }
        }


    }
}
```

Timers.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Speech.Synthesis;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Timer = System.Windows.Forms.Timer;

namespace MovingImage
{
    internal class Timers
    {
        // Interior Lift Location
        int lift_interior_ground_location = 1098;
        int lift_interior_first_location = 280;

        // Speed of the Lifts

        int dX = 2;
        int dY = 2;
        SpeechSynthesizer synthesizer = new SpeechSynthesizer();

        // Sounds for Lift

        System.Media.SoundPlayer ding = new
System.Media.SoundPlayer(Properties.Resources.Elevator_Ding);
        System.Media.SoundPlayer open = new
System.Media.SoundPlayer(Properties.Resources.Open_Sound);
        System.Media.SoundPlayer close = new
System.Media.SoundPlayer(Properties.Resources.Close_Sound);


        // Timers

        // Timer Up
```

```
        public void TimerUp(PictureBox Lift_Interior, PictureBox DisplayBox,
Button Requesting_Down, PictureBox DisplayBox_Ground_Floor, PictureBox
DisplayBox_First_Floor, Timer TimerUp, Timer TimerOpen, Button buttonUp)
        {
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            // Changing Point

            Lift_Interior.Location = new Point(Lift_Interior.Location.X,
Lift_Interior.Location.Y - (dY));
            TimerUp.Enabled = true;

            if (Lift_Interior.Location.Y == lift_interior_first_location)
            {
                TimerUp.Enabled = false;
                TimerOpen.Enabled = true;
                buttonUp.BackColor = Color.Gray;

                Thread.Sleep(200);

                synthesizer.Speak("You have reached First Floor.");
                open.Play();

                // Changing Necessary Images

                Requesting_Down.Image = Properties.Resources.Down;

                DisplayBox.Image = Properties.Resources.First_Floor;
                DisplayBox.SizeMode = PictureBoxSizeMode.CenterImage;
                DisplayBox_Ground_Floor.Image =
Properties.Resources.First_Floor_Small;
                DisplayBox_Ground_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

                DisplayBox_First_Floor.Image =
Properties.Resources.First_Floor_Small;
                DisplayBox_First_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

            }
        }

        // Timer Down
        public void TimerDown(PictureBox Lift_Interior, PictureBox
DisplayBox, Button Requesting_Up, PictureBox DisplayBox_First_Floor,
PictureBox DisplayBox_Ground_Floor, Timer TimerDown, Timer TimerOpen, Button
buttonDown)
        {

            // Synthesizer Settings

            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            // Changing Coordinates
```

```
            Lift_Interior.Location = new Point(Lift_Interior.Location.X,
Lift_Interior.Location.Y + (dY));
            TimerDown.Enabled = true;
            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                TimerDown.Enabled = false;
                TimerOpen.Enabled = true;
                buttonDown.BackColor = Color.Gray;


                synthesizer.Speak("You have reached Ground Floor.");
                Thread.Sleep(200);
                open.Play();

                // Making necessary changes in Picture Box

                DisplayBox.Image = Properties.Resources.Ground_Floor;
                DisplayBox.SizeMode = PictureBoxSizeMode.CenterImage;
                DisplayBox_First_Floor.Image =
Properties.Resources.First_Floor_Small;
                DisplayBox_First_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

                DisplayBox_Ground_Floor.Image =
Properties.Resources.First_Floor_Small;
                DisplayBox_Ground_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

                Requesting_Up.Image = Properties.Resources.Up;

            }
        }

        // Timer Open

        public void TimerOpen(PictureBox Lift_Interior, PictureBox
Ground_Floor_Door, PictureBox First_Floor_Door, Timer TimerOpen, Button
buttonOpen)
        {
            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                // Changing Size of the door to open the door

                Ground_Floor_Door.Size = new
Size(Ground_Floor_Door.Size.Width - dX, Ground_Floor_Door.Size.Height);

                TimerOpen.Enabled = true;
                if (Ground_Floor_Door.Size.Width == 0)
                {
                    TimerOpen.Enabled = false;
                    buttonOpen.BackColor = Color.Gray;

                }
            }
            if (Lift_Interior.Location.Y == lift_interior_first_location)
            {
                // Changing Size of the door to open the door
```

```
                First_Floor_Door.Size = new Size(First_Floor_Door.Size.Width
- dX, First_Floor_Door.Size.Height);
                TimerOpen.Enabled = true;
                if (First_Floor_Door.Size.Width == 0)
                {
                    TimerOpen.Enabled = false;
                    buttonOpen.BackColor = Color.Gray;
                }
            }
        }

        // Timer Close
        public void TimerClose(PictureBox Lift_Interior, PictureBox
Ground_Floor_Door, PictureBox First_Floor_Door, Timer TimerClose, Button
buttonClose)
        {
            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                // Changing Size of the door to close the door

                Ground_Floor_Door.Size = new
Size(Ground_Floor_Door.Size.Width + dX, Ground_Floor_Door.Size.Height);
                TimerClose.Enabled = true;
                if (Ground_Floor_Door.Size.Width == 210)
                {
                    TimerClose.Enabled = false;
                    buttonClose.BackColor = Color.Gray;


                }
            }
            if (Lift_Interior.Location.Y == lift_interior_first_location)
            {
                // Changing Size of the door to close the door

                First_Floor_Door.Size = new Size(First_Floor_Door.Size.Width
+ dX, First_Floor_Door.Size.Height);
                TimerClose.Enabled = true;
                if (First_Floor_Door.Size.Width == 210)
                {
                    TimerClose.Enabled = false;
                    buttonClose.BackColor = Color.Gray;
                }
            }
        }
        // Timer First Floor
        public void TimerFirstFloorClose(PictureBox Lift_Interior, PictureBox
First_Floor_Door, Timer Timer_Close_First_Floor, Timer TimerDown, Button
buttonDown, Button Requesting_Up, PictureBox DisplayBox_First_Floor,
PictureBox DisplayBox_Ground_Floor)
        {

            // Synthesizer Settings
            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            if (Lift_Interior.Location.Y == lift_interior_first_location)
```

```
            {
                // Closes door if its open ar first floor

                First_Floor_Door.Size = new Size(First_Floor_Door.Size.Width
+ 1, First_Floor_Door.Size.Height);
                Timer_Close_First_Floor.Enabled = true;
                if (First_Floor_Door.Size.Width == 210)
                {
                    Timer_Close_First_Floor.Enabled = false;
                    TimerDown.Enabled = true;
                    buttonDown.BackColor = Color.Gray;
                    Requesting_Up.Image = Properties.Resources.Up;
                    DisplayBox_First_Floor.Image =
Properties.Resources.Ground_Floor_Small;
                    DisplayBox_First_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

                    DisplayBox_Ground_Floor.Image =
Properties.Resources.Ground_Floor_Small;
                    DisplayBox_Ground_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

                }
                if (Lift_Interior.Location.Y ==
lift_interior_ground_location)
                {
                    // Changing Picture Box

                    Requesting_Up.Image = Properties.Resources.Up;
                    DisplayBox_First_Floor.Image =
Properties.Resources.Ground_Floor_Small;
                    DisplayBox_First_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;

                    DisplayBox_Ground_Floor.Image =
Properties.Resources.Ground_Floor_Small;
                    DisplayBox_Ground_Floor.SizeMode =
PictureBoxSizeMode.CenterImage;
                    synthesizer.Speak("You have reached Ground Floor.");
                    open.Play();
                }
            }
        }

        // Timer for Ground Floor Closing Door
        public void TimerGroundFloorClose(PictureBox Lift_Interior,
PictureBox Ground_Floor_Door, Timer Timer_Close_Ground_Floor, Timer TimerUp,
Button buttonUp, Button Requesting_Down)
        {
            // Synthesizer Settings

            synthesizer.SelectVoiceByHints(VoiceGender.Female,
VoiceAge.Teen);

            if (Lift_Interior.Location.Y == lift_interior_ground_location)
            {
                // Closes door if its open at ground floor
```

```
                Ground_Floor_Door.Size = new
Size(Ground_Floor_Door.Size.Width + 1, Ground_Floor_Door.Size.Height);
                Timer_Close_Ground_Floor.Enabled = true;
                if (Ground_Floor_Door.Size.Width == 210)
                {
                    Timer_Close_Ground_Floor.Enabled = false;
                    TimerUp.Enabled = true;
                    buttonUp.BackColor = Color.Gray;
                    Requesting_Down.Image = Properties.Resources.Down;



                }
                if (Lift_Interior.Location.Y == lift_interior_first_location)
                {
                    synthesizer.Speak("You have reached First Floor.");
                    open.Play();
                    Requesting_Down.Image = Properties.Resources.Down;

                }
            }
        }

    }
}
```

Database_Connection.cs

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MovingImage
{
    internal class Database_Connection
    {

        // Database Connection Declaration

        private  NpgsqlConnection conn;
        private string? sql;
        private NpgsqlCommand? cmd;
        private DataTable? dt;

        public string connstring = string.Format("Server={0}; Port={1};" +
            "User Id={2}; Password={3}; Database={4};",
            "localhost", 5432, "postgres", "12345", "elevator_log_db");

        // Select Function
```

```
        public void Select(DataGridView dgvLogData)
        {
            // using try catch to catch exceptions.

            try
            {
                conn = new NpgsqlConnection(connstring);
                conn.Open();
                sql = @"select * from log_select();";
                cmd = new NpgsqlCommand(sql, conn);
                dt = new DataTable();
                dt.Load(cmd.ExecuteReader());
                conn.Close();
                dgvLogData.DataSource = null; // reset datagridview
                dgvLogData.DataSource = dt;

            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Error : " + ex.Message);
            }
        }

        // Deleting Logs

        public void Delete_Logs()
        {
            try
            {
                conn = new NpgsqlConnection(connstring);
                conn.Open();
                sql = @"TRUNCATE log_details restart identity;";

                cmd = new NpgsqlCommand(sql, conn);

                cmd.ExecuteNonQuery();

                conn.Close();

            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Error : " + ex.Message);

            }
        }

    }


}
```

Button_Functions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Timer = System.Windows.Forms.Timer;

namespace MovingImage
{
    internal class Button_Functions
    {
        // Lift Interior Coordinates
        int lift_interior_ground_location = 1098;
        int lift_interior_first_location = 280;
        public string ButtonClicked(object sender, EventArgs e)
        {
            // Brings the value of the clicked button

            Button clicked = (Button)sender;
            // If Statements for returning the right vaue.
            if (clicked.Name == "buttonUp")
            {
                return "Up";
            }
            else if (clicked.Name == "buttonDown")
            {
                return "Down";
            }
            else if (clicked.Name == "buttonOpen")
            {
                return "Open";
            }
            else if (clicked.Name == "buttonClose")
            {
                return "Close";
            }
            else if (clicked.Name == "Requesting_Up")
                return "Requesting Down";
            else
            {
                return "Requesting Up";
            }


        }

        // Returns the action depending on the button clicked.
        public string ButtonAction(object sender2, EventArgs e2, PictureBox
Lift_Interior,Timer TimerUp, Timer Timer_Close_Ground_Floor, Timer TimerDown, Timer
Timer_Close_First_Floor, Timer TimerOpen, Timer TimerClose)
        {
            Button clicked = (Button)sender2;

            switch (clicked.Name)
            {
                case "buttonUp":
                    if (TimerUp.Enabled || Timer_Close_Ground_Floor.Enabled)
                    {
```

```
                        return "Going Up from Ground Floor to First Floor";

                    }
                    else
                    {
                        return "Invalid Error";
                    }

                case "buttonDown":
                    if (TimerDown.Enabled || Timer_Close_First_Floor.Enabled)
                    {
                        return "Going Down from First Floor to Ground Floor";
                    }
                    else
                    {
                        return "Invalid Error";
                    }

                case "buttonOpen":
                    if (TimerOpen.Enabled && Lift_Interior.Location.Y ==
lift_interior_ground_location)
                    {
                        return "Door opened at Ground Floor";
                    }

                    else if (!TimerOpen.Enabled && Lift_Interior.Location.Y ==
lift_interior_first_location)
                    {
                        return "Door opened at First Floor";
                    }
                    else
                    {
                        return "Invalid Error";
                    }
                case "buttonClose":
                    if (TimerClose.Enabled && Lift_Interior.Location.Y ==
lift_interior_ground_location)
                    {
                        return "Door closed at Ground Floor";
                    }
                    else if (TimerClose.Enabled && Lift_Interior.Location.Y ==
lift_interior_first_location)
                    {
                        return "Door closed at First Floor";
                    }
                    else
                    {
                        return "Invalid Error";
                    }
                case "Requesting_Up":
                    if (TimerDown.Enabled || Timer_Close_First_Floor.Enabled)
                    {
                        return "Requesting Lift from Ground Floor";

                    }
                    else
                    {
                        return "Invalid Error";
```

```
                }

            case "Requesting_Down":
                if (TimerUp.Enabled || Timer_Close_Ground_Floor.Enabled )
                {
                    return "Requesting Lift from First Floor";
                }
                else
                {
                    return "Invalid Error";
                }
            default: return "Invalid Error";
        }
    }



    }
}
```

Form1.Designer.cs

```
namespace MovingImage
{
    partial class Form1
    {
        /// <summary>
        ///  Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        ///  Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        ///  Required method for Designer support - do not modify
        ///  the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle1
= new System.Windows.Forms.DataGridViewCellStyle();
```

```
            System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle2
= new System.Windows.Forms.DataGridViewCellStyle();
            System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle3
= new System.Windows.Forms.DataGridViewCellStyle();
            System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle4
= new System.Windows.Forms.DataGridViewCellStyle();
            System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle5
= new System.Windows.Forms.DataGridViewCellStyle();
            this.Lift_Interior = new System.Windows.Forms.PictureBox();
            this.TimerUp = new System.Windows.Forms.Timer(this.components);
            this.TimerDown = new System.Windows.Forms.Timer(this.components);
            this.TimerOpen = new System.Windows.Forms.Timer(this.components);
            this.TimerClose = new
System.Windows.Forms.Timer(this.components);
            this.buttonUp = new System.Windows.Forms.Button();
            this.buttonDown = new System.Windows.Forms.Button();
            this.buttonOpen = new System.Windows.Forms.Button();
            this.buttonClose = new System.Windows.Forms.Button();
            this.Ground_Floor_Door = new System.Windows.Forms.PictureBox();
            this.First_Floor_Door = new System.Windows.Forms.PictureBox();
            this.Timer_Close_First_Floor = new
System.Windows.Forms.Timer(this.components);
            this.Timer_Close_Ground_Floor = new
System.Windows.Forms.Timer(this.components);
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.pictureBox3 = new System.Windows.Forms.PictureBox();
            this.dgvLogData = new System.Windows.Forms.DataGridView();
            this.buttonShowLogs = new System.Windows.Forms.Button();
            this.buttonClearLogs = new System.Windows.Forms.Button();
            this.buttonHideLogs = new System.Windows.Forms.Button();
            this.DisplayBox = new System.Windows.Forms.PictureBox();
            this.DisplayBox_First_Floor = new
System.Windows.Forms.PictureBox();
            this.DisplayBox_Ground_Floor = new
System.Windows.Forms.PictureBox();
            this.Requesting_Up = new System.Windows.Forms.Button();
            this.Requesting_Down = new System.Windows.Forms.Button();
            this.pictureBox4 = new System.Windows.Forms.PictureBox();
            this.Timer_Requesting_Up = new
System.Windows.Forms.Timer(this.components);

((System.ComponentModel.ISupportInitialize)(this.Lift_Interior)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.Ground_Floor_Door)).BeginIni
t();

((System.ComponentModel.ISupportInitialize)(this.First_Floor_Door)).BeginInit
();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.dgvLogData)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.DisplayBox)).BeginInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.DisplayBox_First_Floor)).Beg
inInit();

((System.ComponentModel.ISupportInitialize)(this.DisplayBox_Ground_Floor)).Be
ginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).BeginInit();
            this.SuspendLayout();
            //
            // Lift_Interior
            //
            this.Lift_Interior.BackColor = System.Drawing.Color.Transparent;
            this.Lift_Interior.Image =
global::MovingImage.Properties.Resources.Lift;
            this.Lift_Interior.Location = new System.Drawing.Point(310,
1098);
            this.Lift_Interior.Margin = new System.Windows.Forms.Padding(4);
            this.Lift_Interior.Name = "Lift_Interior";
            this.Lift_Interior.Size = new System.Drawing.Size(210, 374);
            this.Lift_Interior.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.Lift_Interior.TabIndex = 0;
            this.Lift_Interior.TabStop = false;
            //
            // TimerUp
            //
            this.TimerUp.Interval = 4;
            this.TimerUp.Tick += new System.EventHandler(this.TimerUp_Tick);
            //
            // TimerDown
            //
            this.TimerDown.Interval = 4;
            this.TimerDown.Tick += new
System.EventHandler(this.TimerDown_Tick);
            //
            // TimerOpen
            //
            this.TimerOpen.Interval = 4;
            this.TimerOpen.Tick += new
System.EventHandler(this.TimerOpen_Tick);
            //
            // TimerClose
            //
            this.TimerClose.Interval = 4;
            this.TimerClose.Tick += new
System.EventHandler(this.TimerClose_Tick);
            //
            // buttonUp
            //
            this.buttonUp.BackColor = System.Drawing.Color.Gray;
            this.buttonUp.Image =
global::MovingImage.Properties.Resources.Button_up;
            this.buttonUp.Location = new System.Drawing.Point(959, 327);
            this.buttonUp.Margin = new System.Windows.Forms.Padding(2);
            this.buttonUp.Name = "buttonUp";
            this.buttonUp.Size = new System.Drawing.Size(145, 144);
```

```
            this.buttonUp.TabIndex = 1;
            this.buttonUp.UseVisualStyleBackColor = false;
            this.buttonUp.Click += new
System.EventHandler(this.UpButtonClick);
            //
            // buttonDown
            //
            this.buttonDown.BackColor = System.Drawing.Color.Gray;
            this.buttonDown.Image =
global::MovingImage.Properties.Resources.Button_Down;
            this.buttonDown.Location = new System.Drawing.Point(959, 1034);
            this.buttonDown.Margin = new System.Windows.Forms.Padding(2);
            this.buttonDown.Name = "buttonDown";
            this.buttonDown.Size = new System.Drawing.Size(145, 144);
            this.buttonDown.TabIndex = 2;
            this.buttonDown.UseVisualStyleBackColor = false;
            this.buttonDown.Click += new
System.EventHandler(this.DownButtonClick);
            //
            // buttonOpen
            //
            this.buttonOpen.BackColor = System.Drawing.Color.Gray;
            this.buttonOpen.Image =
global::MovingImage.Properties.Resources.Button_Open;
            this.buttonOpen.Location = new System.Drawing.Point(1053, 1195);
            this.buttonOpen.Margin = new System.Windows.Forms.Padding(2);
            this.buttonOpen.Name = "buttonOpen";
            this.buttonOpen.Size = new System.Drawing.Size(140, 144);
            this.buttonOpen.TabIndex = 3;
            this.buttonOpen.UseVisualStyleBackColor = false;
            this.buttonOpen.Click += new
System.EventHandler(this.OpenButtonClick);
            //
            // buttonClose
            //
            this.buttonClose.BackColor = System.Drawing.Color.Gray;
            this.buttonClose.Image =
global::MovingImage.Properties.Resources.Button_Close;
            this.buttonClose.Location = new System.Drawing.Point(853, 1195);
            this.buttonClose.Margin = new System.Windows.Forms.Padding(2);
            this.buttonClose.Name = "buttonClose";
            this.buttonClose.Size = new System.Drawing.Size(145, 144);
            this.buttonClose.TabIndex = 4;
            this.buttonClose.UseVisualStyleBackColor = false;
            this.buttonClose.Click += new
System.EventHandler(this.CloseButtonClick);
            //
            // Ground_Floor_Door
            //
            this.Ground_Floor_Door.BackColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.Ground_Floor_Door.Image =
global::MovingImage.Properties.Resources.Door;
            this.Ground_Floor_Door.Location = new System.Drawing.Point(310,
1098);
            this.Ground_Floor_Door.Margin = new
System.Windows.Forms.Padding(2);
```

```
            this.Ground_Floor_Door.Name = "Ground_Floor_Door";
            this.Ground_Floor_Door.Size = new System.Drawing.Size(210, 374);
            this.Ground_Floor_Door.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.Ground_Floor_Door.TabIndex = 6;
            this.Ground_Floor_Door.TabStop = false;
            //
            // First_Floor_Door
            //
            this.First_Floor_Door.Image =
global::MovingImage.Properties.Resources.Door;
            this.First_Floor_Door.Location = new System.Drawing.Point(310,
280);
            this.First_Floor_Door.Margin = new
System.Windows.Forms.Padding(2);
            this.First_Floor_Door.Name = "First_Floor_Door";
            this.First_Floor_Door.Size = new System.Drawing.Size(210, 380);
            this.First_Floor_Door.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.First_Floor_Door.TabIndex = 7;
            this.First_Floor_Door.TabStop = false;
            //
            // Timer_Close_First_Floor
            //
            this.Timer_Close_First_Floor.Interval = 4;
            this.Timer_Close_First_Floor.Tick += new
System.EventHandler(this.Timer_Close_1F_Tick);
            //
            // Timer_Close_Ground_Floor
            //
            this.Timer_Close_Ground_Floor.Interval = 4;
            this.Timer_Close_Ground_Floor.Tick += new
System.EventHandler(this.Timer_Close_Ground_Floor_Tick);
            //
            // pictureBox1
            //
            this.pictureBox1.BackColor = System.Drawing.Color.Transparent;
            this.pictureBox1.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.None;
            this.pictureBox1.Image =
global::MovingImage.Properties.Resources.Door_Frame;
            this.pictureBox1.Location = new System.Drawing.Point(255, 1049);
            this.pictureBox1.Margin = new System.Windows.Forms.Padding(2);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(300, 458);
            this.pictureBox1.TabIndex = 8;
            this.pictureBox1.TabStop = false;
            //
            // pictureBox3
            //
            this.pictureBox3.BackColor = System.Drawing.Color.Gray;
            this.pictureBox3.Image =
global::MovingImage.Properties.Resources.Button_BG;
            this.pictureBox3.Location = new System.Drawing.Point(803, 179);
            this.pictureBox3.Margin = new System.Windows.Forms.Padding(2);
            this.pictureBox3.Name = "pictureBox3";
            this.pictureBox3.Size = new System.Drawing.Size(451, 1241);
```

```
            this.pictureBox3.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox3.TabIndex = 10;
            this.pictureBox3.TabStop = false;
            //
            // dgvLogData
            //
            this.dgvLogData.AllowUserToAddRows = false;
            this.dgvLogData.AllowUserToDeleteRows = false;
            this.dgvLogData.AllowUserToResizeColumns = false;
            this.dgvLogData.AllowUserToResizeRows = false;
            dataGridViewCellStyle1.BackColor =
System.Drawing.Color.DeepSkyBlue;
            dataGridViewCellStyle1.Font = new System.Drawing.Font("Times New
Roman", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            dataGridViewCellStyle1.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            dataGridViewCellStyle1.SelectionBackColor =
System.Drawing.Color.Transparent;
            dataGridViewCellStyle1.SelectionForeColor =
System.Drawing.Color.White;
            this.dgvLogData.AlternatingRowsDefaultCellStyle =
dataGridViewCellStyle1;
            this.dgvLogData.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.AllCells;
            this.dgvLogData.BackgroundColor =
System.Drawing.SystemColors.ControlDarkDark;
            this.dgvLogData.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            dataGridViewCellStyle2.Alignment =
System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;
            dataGridViewCellStyle2.BackColor =
System.Drawing.Color.DeepSkyBlue;
            dataGridViewCellStyle2.Font = new System.Drawing.Font("Times New
Roman", 14F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            dataGridViewCellStyle2.ForeColor =
System.Drawing.SystemColors.WindowText;
            dataGridViewCellStyle2.SelectionBackColor =
System.Drawing.Color.Transparent;
            dataGridViewCellStyle2.SelectionForeColor =
System.Drawing.SystemColors.HighlightText;
            dataGridViewCellStyle2.WrapMode =
System.Windows.Forms.DataGridViewTriState.True;
            this.dgvLogData.ColumnHeadersDefaultCellStyle =
dataGridViewCellStyle2;
            this.dgvLogData.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            dataGridViewCellStyle3.Alignment =
System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;
            dataGridViewCellStyle3.BackColor =
System.Drawing.SystemColors.Window;
            dataGridViewCellStyle3.Font = new System.Drawing.Font("Segoe UI",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);
            dataGridViewCellStyle3.ForeColor =
System.Drawing.SystemColors.ControlText;
```

```
            dataGridViewCellStyle3.SelectionBackColor =
System.Drawing.Color.Transparent;
            dataGridViewCellStyle3.SelectionForeColor =
System.Drawing.SystemColors.HighlightText;
            dataGridViewCellStyle3.WrapMode =
System.Windows.Forms.DataGridViewTriState.False;
            this.dgvLogData.DefaultCellStyle = dataGridViewCellStyle3;
            this.dgvLogData.EnableHeadersVisualStyles = false;
            this.dgvLogData.Location = new System.Drawing.Point(1301, 179);
            this.dgvLogData.MultiSelect = false;
            this.dgvLogData.Name = "dgvLogData";
            this.dgvLogData.ReadOnly = true;
            dataGridViewCellStyle4.Alignment =
System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;
            dataGridViewCellStyle4.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))),
((int)(((byte)(128)))));
            dataGridViewCellStyle4.Font = new System.Drawing.Font("Times New
Roman", 14F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            dataGridViewCellStyle4.ForeColor =
System.Drawing.SystemColors.WindowText;
            dataGridViewCellStyle4.SelectionBackColor =
System.Drawing.Color.Transparent;
            dataGridViewCellStyle4.SelectionForeColor =
System.Drawing.SystemColors.HighlightText;
            dataGridViewCellStyle4.WrapMode =
System.Windows.Forms.DataGridViewTriState.True;
            this.dgvLogData.RowHeadersDefaultCellStyle =
dataGridViewCellStyle4;
            this.dgvLogData.RowHeadersWidth = 62;
            dataGridViewCellStyle5.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))),
((int)(((byte)(128)))));
            dataGridViewCellStyle5.Font = new System.Drawing.Font("Times New
Roman", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            dataGridViewCellStyle5.ForeColor =
System.Drawing.SystemColors.ActiveCaptionText;
            dataGridViewCellStyle5.SelectionBackColor =
System.Drawing.Color.Transparent;
            this.dgvLogData.RowsDefaultCellStyle = dataGridViewCellStyle5;
            this.dgvLogData.RowTemplate.Height = 33;
            this.dgvLogData.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect;
            this.dgvLogData.Size = new System.Drawing.Size(1116, 700);
            this.dgvLogData.TabIndex = 11;
            //
            // buttonShowLogs
            //
            this.buttonShowLogs.BackColor = System.Drawing.Color.LightCoral;
            this.buttonShowLogs.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
            this.buttonShowLogs.Font = new System.Drawing.Font("Times New
Roman", 14F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
```

```
            this.buttonShowLogs.ForeColor =
System.Drawing.Color.MidnightBlue;
            this.buttonShowLogs.Location = new System.Drawing.Point(1888,
913);
            this.buttonShowLogs.Name = "buttonShowLogs";
            this.buttonShowLogs.Size = new System.Drawing.Size(156, 74);
            this.buttonShowLogs.TabIndex = 12;
            this.buttonShowLogs.Text = "Show Logs";
            this.buttonShowLogs.UseVisualStyleBackColor = false;
            this.buttonShowLogs.Click += new
System.EventHandler(this.buttonShowLogs_Click);
            //
            // buttonClearLogs
            //
            this.buttonClearLogs.BackColor = System.Drawing.Color.LightCoral;
            this.buttonClearLogs.Enabled = false;
            this.buttonClearLogs.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
            this.buttonClearLogs.Font = new System.Drawing.Font("Times New
Roman", 14F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            this.buttonClearLogs.ForeColor =
System.Drawing.Color.MidnightBlue;
            this.buttonClearLogs.Location = new System.Drawing.Point(2265,
913);
            this.buttonClearLogs.Name = "buttonClearLogs";
            this.buttonClearLogs.Size = new System.Drawing.Size(152, 74);
            this.buttonClearLogs.TabIndex = 13;
            this.buttonClearLogs.Text = "Clear Logs";
            this.buttonClearLogs.UseVisualStyleBackColor = false;
            this.buttonClearLogs.Click += new
System.EventHandler(this.buttonClearLogs_Click);
            //
            // buttonHideLogs
            //
            this.buttonHideLogs.BackColor = System.Drawing.Color.LightCoral;
            this.buttonHideLogs.Enabled = false;
            this.buttonHideLogs.FlatStyle =
System.Windows.Forms.FlatStyle.Popup;
            this.buttonHideLogs.Font = new System.Drawing.Font("Times New
Roman", 14F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            this.buttonHideLogs.ForeColor =
System.Drawing.Color.MidnightBlue;
            this.buttonHideLogs.Location = new System.Drawing.Point(2081,
913);
            this.buttonHideLogs.Name = "buttonHideLogs";
            this.buttonHideLogs.Size = new System.Drawing.Size(149, 74);
            this.buttonHideLogs.TabIndex = 14;
            this.buttonHideLogs.Text = "Hide Logs";
            this.buttonHideLogs.UseVisualStyleBackColor = false;
            this.buttonHideLogs.Click += new
System.EventHandler(this.buttonHideLogs_Click);
            //
            // DisplayBox
            //
```

```
            this.DisplayBox.BackColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.DisplayBox.Image =
global::MovingImage.Properties.Resources.Ground_Floor;
            this.DisplayBox.Location = new System.Drawing.Point(910, 536);
            this.DisplayBox.Name = "DisplayBox";
            this.DisplayBox.Size = new System.Drawing.Size(253, 451);
            this.DisplayBox.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.DisplayBox.TabIndex = 15;
            this.DisplayBox.TabStop = false;
            //
            // DisplayBox_First_Floor
            //
            this.DisplayBox_First_Floor.BackColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.DisplayBox_First_Floor.Image =
global::MovingImage.Properties.Resources.Ground_Floor_Small;
            this.DisplayBox_First_Floor.Location = new
System.Drawing.Point(381, 134);
            this.DisplayBox_First_Floor.Name = "DisplayBox_First_Floor";
            this.DisplayBox_First_Floor.Size = new System.Drawing.Size(80,
89);
            this.DisplayBox_First_Floor.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.DisplayBox_First_Floor.TabIndex = 16;
            this.DisplayBox_First_Floor.TabStop = false;
            //
            // DisplayBox_Ground_Floor
            //
            this.DisplayBox_Ground_Floor.BackColor =
System.Drawing.SystemColors.ActiveCaptionText;
            this.DisplayBox_Ground_Floor.Image =
global::MovingImage.Properties.Resources.Ground_Floor_Small;
            this.DisplayBox_Ground_Floor.Location = new
System.Drawing.Point(381, 955);
            this.DisplayBox_Ground_Floor.Name = "DisplayBox_Ground_Floor";
            this.DisplayBox_Ground_Floor.Size = new System.Drawing.Size(80,
89);
            this.DisplayBox_Ground_Floor.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.DisplayBox_Ground_Floor.TabIndex = 17;
            this.DisplayBox_Ground_Floor.TabStop = false;
            //
            // Requesting_Up
            //
            this.Requesting_Up.Image =
global::MovingImage.Properties.Resources.Up;
            this.Requesting_Up.Location = new System.Drawing.Point(586,
1258);
            this.Requesting_Up.Name = "Requesting_Up";
            this.Requesting_Up.Size = new System.Drawing.Size(82, 81);
            this.Requesting_Up.TabIndex = 18;
            this.Requesting_Up.UseVisualStyleBackColor = false;
            this.Requesting_Up.Click += new
System.EventHandler(this.Requesting_Up_Click);
            //
```

```
            // Requesting_Down
            //
            this.Requesting_Down.Image =
global::MovingImage.Properties.Resources.Down;
            this.Requesting_Down.Location = new System.Drawing.Point(586,
450);
            this.Requesting_Down.Name = "Requesting_Down";
            this.Requesting_Down.Size = new System.Drawing.Size(82, 83);
            this.Requesting_Down.TabIndex = 19;
            this.Requesting_Down.UseVisualStyleBackColor = true;
            this.Requesting_Down.Click += new
System.EventHandler(this.Requesting_Down_Click);
            //
            // pictureBox4
            //
            this.pictureBox4.BackColor = System.Drawing.Color.Transparent;
            this.pictureBox4.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.None;
            this.pictureBox4.Image =
global::MovingImage.Properties.Resources.Door_Frame;
            this.pictureBox4.Location = new System.Drawing.Point(255, 228);
            this.pictureBox4.Margin = new System.Windows.Forms.Padding(2);
            this.pictureBox4.Name = "pictureBox4";
            this.pictureBox4.Size = new System.Drawing.Size(300, 458);
            this.pictureBox4.TabIndex = 20;
            this.pictureBox4.TabStop = false;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(10F, 25F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.SystemColors.ControlDark;
            this.BackgroundImage =
global::MovingImage.Properties.Resources.Titanium_Textures;
            this.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Stretch;
            this.ClientSize = new System.Drawing.Size(2529, 1552);
            this.Controls.Add(this.Requesting_Down);
            this.Controls.Add(this.Requesting_Up);
            this.Controls.Add(this.DisplayBox_Ground_Floor);
            this.Controls.Add(this.DisplayBox_First_Floor);
            this.Controls.Add(this.DisplayBox);
            this.Controls.Add(this.buttonHideLogs);
            this.Controls.Add(this.buttonClearLogs);
            this.Controls.Add(this.buttonShowLogs);
            this.Controls.Add(this.dgvLogData);
            this.Controls.Add(this.First_Floor_Door);
            this.Controls.Add(this.Ground_Floor_Door);
            this.Controls.Add(this.Lift_Interior);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.buttonClose);
            this.Controls.Add(this.buttonOpen);
            this.Controls.Add(this.buttonDown);
            this.Controls.Add(this.buttonUp);
            this.Controls.Add(this.pictureBox3);
            this.Controls.Add(this.pictureBox4);
            this.DoubleBuffered = true;
```

```
            this.ForeColor = System.Drawing.SystemColors.ControlText;
            this.Margin = new System.Windows.Forms.Padding(4);
            this.Name = "Form1";
            this.Text = "Lift_Form";
            this.Load += new System.EventHandler(this.Form1_Load_1);

((System.ComponentModel.ISupportInitialize)(this.Lift_Interior)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.Ground_Floor_Door)).EndInit(
);

((System.ComponentModel.ISupportInitialize)(this.First_Floor_Door)).EndInit()
;

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.dgvLogData)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.DisplayBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.DisplayBox_First_Floor)).End
Init();

((System.ComponentModel.ISupportInitialize)(this.DisplayBox_Ground_Floor)).En
dInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private PictureBox Lift_Interior;
        private System.Windows.Forms.Timer TimerUp;
        private System.Windows.Forms.Timer TimerDown;
        private System.Windows.Forms.Timer TimerOpen;
        private System.Windows.Forms.Timer TimerClose;
        private Button buttonUp;
        private Button buttonDown;
        private Button buttonOpen;
        private Button buttonClose;
        private PictureBox Ground_Floor_Door;
        private PictureBox First_Floor_Door;
        private System.Windows.Forms.Timer Timer_Close_First_Floor;
        private System.Windows.Forms.Timer Timer_Close_Ground_Floor;
        private PictureBox pictureBox1;
        private PictureBox pictureBox3;
        private DataGridView dgvLogData;
        private Button buttonShowLogs;
        private Button buttonClearLogs;
        private Button buttonHideLogs;
        private PictureBox DisplayBox;
        private PictureBox DisplayBox_First_Floor;
        private PictureBox DisplayBox_Ground_Floor;
```

```
        private Button Requesting_Up;
        private Button Requesting_Down;
        private PictureBox pictureBox4;
        private System.Windows.Forms.Timer Timer_Requesting_Up;
    }
}
```

Program.cs

```
namespace MovingImage
{
    internal static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high DPI
settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}
```