

Selected Delhi Transportation Routing Bus Scheduling

*Report submitted in partial fulfillment for
the award of degree of*

Master of Science

in

Applied Mathematics

by

**Monu
2k19/MSCMAT/20**

Under the supervision of
Prof. L.N.Das



DEPARTMENT OF APPLIED MATHEMATICS

DELHI TECHNOLOGICAL UNIVERSITY

Acknowledgement

I would to express my sincere thanks and appreciation to my guide, Prof. L.N. Das, for his patient guidance, constant encouragement , endless support during my study.

I would be like to thank the Department of Applied Mathematics of Delhi Technological University (formerly Delhi College of Engineering) for providing a productive scientific research environment.

I would also like to thank the Head of Bawana Depot to give me report how Delhi Transport Corporation work and their useful suggestions.

Candidate's Declaration

I, (Monu), 2K19/MSCMAT/20 of M.Sc (Mathematics), hereby declare that the project Dissertation titled "Selected Delhi Transportation Routing Bus Scheduling" which is submitted by me in Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the Degree of Master of Science, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or similar title or recognition

Certificate

I hereby certify that the Project Dissertation titled "Selected Delhi Transportation Routing Bus Scheduling", which is submitted by Monu(2k19/MSCMAT/20) an postgraduate student of the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree to this University or elsewhere.

Abstract

In this report, we present a model for solving real daily scheduling problem of Delhi Transport Corporation in crew planning and bus scheduling in selected route with considering proper allotment of bus in the different mode such as early,afternoon,evening and late mode. An model is designed for improve the efficiency to minimize the idle time length, which improve the budget of Delhi Transport Corporation to minimize the cost of crew and efficiently allot the crew as well as maximize the earning of Delhi Transport Corporation by joining the inter route of the bus according to waiting ratio of passengers by 0-1 integer programming problem solution model with the help of Computer Assisted Software.At last system database management is to be developed.

Contents

Acknowledgement	iii
Abstract	iv
1 Introduction	1
2 Literature Review	3
3 Bus Scheduling	5
3.1 Problem Description	5
3.2 Definitions and Notations	6
3.3 Model Formation	7
3.4 Expression of Solution	7
4 Crew Scheduling	9
4.1 Problem Description	9
4.1.1 Assumptions:	9
4.2 Model Formation	10
4.2.1 Model 1	10
4.2.2 Modal 2	10
4.3 Expression of Solution	12
5 Computational Results	13
5.1 Code	13
5.1.1 Screen and Code for Login Page	13
5.1.2 Screen and Code for MDI Form	15
5.1.3 Screen and Code for Add Crew	18
5.1.4 Screen and Code for View Crew and Update its Information	20
5.1.5 Screen and Code for Add Buses	24
5.1.6 Screen and Code for View , Update and Delete details of Bus	26
References	30

Chapter 1

Introduction

The effect of planning frameworks in open transportation has been expanding as significant value saving area and are achievable if on the market resources like crews and vehicles are assigned for efficient use. In particular, in Delhi Public Transit operation designing method, it includes 5 fundamental exercises now and then acted in an arrangement such as network course style, schedule advancement, vehicle scheduling, crew arranging, and crew programs.

In the network course allocation stage, the transport stoppage terminals and exchanges area are generally resolved to adjust the number of routed buses in specific terminal and interchange areas. Because of the route for every bus line, there are a limited number of buses, crews, and an indefinite number of passengers are served in a time periodical frequency for each transit. It is decided to fulfill the Public demand and transport management administration requirements.

The Timetables(schedule advancement) area is constructed in such a manner that the bus routing with beginning and finish terminals' should have the list of departure and arrival time of the present as well as the destination. Similarly, in the Vehicle Scheduling Problem, the vehicles are arranged with the routes having periodical every day timetabling schedules, called vehicle block. the timetable is also characterized by the bus route beginning at a depot, and getting back to an equivalent or a special depot with consideration of route journey plans.

The crew roasters(crew programs) plan includes the future (long terms) crew designing, wherever crew roasters generated from crew shifts with constraints such as halting time is minimum and each shifts transit vehicles should be accommodated with a definite number of crews as per the scheme of the roster list.

This stage comprises of the days-off arranging and moves to arrange. The vacation day arranging manages the task of rest days to the group over a planning skyline, though proper arranging of the task for the crew in shifts In this investigation, we have taken 2 basic activities such as Vehicle Scheduling and crew Planning.

Crew planning Stage of the look method is taken into account of Delhi Transportation

Corporation. Crew Planning is short-term crew planning in which daily shifts are formed for the crews. There is a specified number of steps similar to the vehicle blocks operation transition. Whereinto complete the sequence of shift wise operations, tasks are allocated shift wise, keeping in mind that none of the shift overlaps. The crew shifts are build upon a specified set of rules like the most driving cut-off time, minimum operation cost, in order to fulfill a given set of tasks and minimizing the idle time length. A task is outlined as an action performed on one vehicle by limited definite crew members without any intermission. A task may be a d-trip, shaped by partitioning the journeys at a relief point; or a deadhead, that could be driving action between 2 relief points with no travelers in the bus. The relief points are designed locations and times where an amendment of selective crews could perform the task. The journeys that are the beginning and ending location for every task in that trips are pre-planned. A task may be allotted to specific crews reckoning on the preceding task assigned to that crew. Therefore, the traveling time between task differ from any beginning location to any end location resulting in sequence-dependent setup times between the tasks

Bus Scheduling task considers only 1 form of bus type and with a fixed number of seats routing in a selective route within the time slot and from a listed number of buses available in beginning depot.

Chapter 2

Literature Review

Fischetti et al. (1987) considering the fixed occupation plan issue with spread time constraints and show that is NP-hard, and solve by introducing several lower bounds [1]Fischetti et al. (1989) considering the fixed-job schedule problem with operating time constraints and show that is also NP-hard. As these two extraordinary instances of the crew planning an issue with spread and working time limitations is NP-hard[2].

Ugur Eliyi, Deniz T Eliyi, Levent Kandiller,2016 [3]proved that a genuine crew arranging of a public transportation authority, where the objective is to work out the best variety of different kinds of team individuals with a base worth that could be a given arrangement of assignments identifying with working and unfold time impediments such as each driver features an unfurl cutoff time from the start time to the top season of his/her day of work, including the inactive occasions(idle times). For sure, a driver can't surplus the most extreme all-out working cutoff time. The process times of the work allocated to each driver are enclosed in his/her operating time, also in light of the fact that the crew subordinate arrangement times. As our intention is impressed by genuine crew planning, the works will need different sorts of vehicles that need completely distinct crew capabilities. Subsequently, there are distinctive team classifications that upheld the skills expected to utilize bound vehicle sorts incitement qualification limitations inside the issues. We planned a Tactical Fixed Job Scheduling Problem-based two-fold programming model for the issue. Inside the planed program, they consider solely process times of tasks as operating time. To avoid process a further sequence management variable that explodes the model size and successively ruins resolution performance, we have a tendency to develop an Associate in Nursing reiterative valid difference generation theme. Which eliminates task sequences exceeding the entire operating time once arrangement times are enclosed. The presentation of the created model is researched through complete experimentation and furthermore,the mathematical outcomes are supposed.

M. Chen and H. Niu 2012, in their study he present an associate approach for finding the bus crew planning problem that considers early, day, and late duty modes with time shift and works intensity constraints. moreover, the constraint with the smallest amount crew variety of a particular duty[4] e.g., day duty has conjointly been thought of. An

improvement model is developed as a 0-1 integer programming problem to enhance the potency of crew programming at the minimum expense of total idle time of crew for a circle transit line. Correspondingly, a heuristic algorithmic program utilizing the tabu search algorithmic program has been projected to solve the model

Avishai (Avi) Ceder, in chapter 8 and 9 describe the vehicle scheduling problem of different sorts such as fleet size required for a single route, depot constraints vehicle scheduling, Example of an exact solution for multi-route vehicle scheduling, Fleet-size lower bound for fixed schedules, Fleet-size lower bound for variable schedules, and experience with bus scheduling

Chapter 3

Bus Scheduling

3.1 Problem Description

Bus Scheduling in this paper is to arrange the route and bus in Delhi Transport Corporation Bus Service. I take a single type of Bus in this Paper and the fixed-route with starting depot which is Bawana Depot. We assumed in this Paper that from Bawana Depot three routes namely Bawana to Janakpuri, Bawana to Old Delhi, and Bawana to ISBT. The way from Bawana to Mandan Chowk, at where 3 routes share the common path. Therefore transit in the different route. We track the Buses start from Bawana to Janakpuri and their capacity(Capacity of a passenger in Bus) is not more than 25 from Mandan Chowk But Passenger waiting at Mandan Chowk is more to go to Old Delhi. Then the Bus Stop at Madan Chowk and shift the passenger to the next coming bus from Bawana to Janakpuri and start its trip as new trips from Madan chowk to Old Delhi. Similarly when Bus starts from Old Delhi, at the time Bus coming to Mandan chowk at passenger capacity is not more than 25 then the bus drop the passenger at Mandan chowk or shift to another Bus, that comes from either Jankpuri or ISBT to Bawana. Because the way from Madan chowk to Bawana is same for all 3 Routes, Bus which came from Old Delhi, Janakpuri, and ISBT. Thus it starts a new trip from Madan chowk.

As we divide a day into 4 mode such as early,afternoon ,evening,late mode.

In early mode, the time gap between two consecutive buses is 10-15 min.

In afternoon mode, the time gap between two consecutive buses is 20-30 min.

In evening mode, the time gap between two consecutive buses is 10-15 min.

In late mode, the time gap between two consecutive buses is 1:30-2:00 hours because at night , the number of passenger is very less so ,we arrange buses at more time interval between the two consecutive bus and also decrease the number of buses.

We restricted that the at the end of evening mode all buses go its starting station. So, that next day the buses will be starting as same schedule.

3.2 Definitions and Notations

The following Definition and Notation are use in Vehicle Scheduling Section in the paper.

i : Number of Fixed Routes.

j: Number of Station.

p: Number of Buses joining i route to j station.

$p_{i,j}$ is binary 0-1 integer which indicates the i route cover j station if 1 then joining is admissible otherwise not

We define 9 fixed routes such as

Starting	Ending
Old Delhi	Janakpuri
Old Delhi	Bawana
Old Delhi	ISBT
ISBT	Bawana
ISBT	Janakpuri
ISBT	Old Delhi
Janakpuri	Old Delhi
Janakpuri	Bawana
Janakpuri	ISBT

Number of Station

Bawana
Madan Chowk
Janakpuri
Old Delhi
ISBT

3.3 Model Formation

Objective Function :

Maximum :

$$\sum_j \sum_i p_{i,j}$$

Subject to :

$$\begin{aligned} \sum_j p_{i,j} &\leq 1 \\ \sum_i p_{i,j} &\leq 1 \end{aligned}$$

Objective Function say that Maximum the number of Buses joining i route to j Station.

First Constraints say that Each trip may be joined with at most one successor trip from fixed route i to j station.

Second Constraints say that, Each trip may be joined with at most one predecessor trip to station j to variable route i.

3.4 Expression of Solution

Maximum :

$$p_{11} + p_{12} + p_{13} + p_{14} + p_{15} + p_{16} + p_{17} + p_{18} + p_{19} + p_{21} + p_{22} + p_{23} + p_{24} + p_{25} + p_{26} + p_{27} + p_{28} + p_{29} + p_{31} + p_{32} + p_{33} + p_{34} + p_{35} + p_{36} + p_{37} + p_{38} + p_{39} + p_{41} + p_{42} + p_{43} + p_{44} + p_{45} + p_{46} + p_{47} + p_{48} + p_{49} + p_{51} + p_{52} + p_{53} + p_{54} + p_{55} + p_{56} + p_{57} + p_{58} + p_{59}$$

Subject to :

$$p_{11} + p_{12} + p_{13} + p_{14} + p_{15} + p_{16} + p_{17} + p_{18} + p_{19} \leq 1$$

$$p_{21} + p_{22} + p_{23} + p_{24} + p_{25} + p_{26} + p_{27} + p_{28} + p_{29} \leq 1$$

$$p_{31} + p_{32} + p_{33} + p_{34} + p_{35} + p_{36} + p_{37} + p_{38} + p_{39} \leq 1$$

$$p_{41} + p_{42} + p_{43} + p_{44} + p_{45} + p_{46} + p_{47} + p_{48} + p_{49} \leq 1$$

$$p_{51} + p_{52} + p_{53} + p_{54} + p_{55} + p_{56} + p_{57} + p_{58} + p_{59} \leq 1$$

$$p_{11} + p_{21} + p_{31} + p_{41} + p_{51} \leq 1$$

$$p_{12} + p_{22} + p_{32} + p_{42} + p_{52} \leq 1$$

$$p_{13} + p_{23} + p_{33} + p_{43} + p_{53} \leq 1$$

$$p_{14} + p_{24} + p_{34} + p_{44} + p_{54} \leq 1$$

$$p_{15} + p_{25} + p_{35} + p_{45} + p_{55} \leq 1$$

$$p_{16} + p_{26} + p_{36} + p_{46} + p_{56} \leq 1$$

$$p_{17} + p_{27} + p_{37} + p_{47} + p_{57} \leq 1$$

$$p_{18} + p_{28} + p_{38} + p_{48} + p_{58} \leq 1$$

$$p_{19} + p_{29} + p_{39} + p_{49} + p_{59} \leq 1$$

With the help of computer Assignment software TORA we find the optimal solution of

this model is 5

in which

$$p_{15} = 1$$

$$p_{24} = 1$$

$$p_{33} = 1$$

$$p_{41} = 1$$

$$p_{51} = 1$$

Chapter 4

Crew Scheduling

4.1 Problem Description

Crew Scheduling Problem in this report is to arrange the duties of the crew in Delhi Transport Corporation's Bus Service. I take a Single type bus line in the report. Each trip has a fixed departure and arrival time at each Bus Stop. All the trips are arranged in ascending order according to departure time.

Each Crew ought to be limited to work in a similar vehicle during the Bus Operating time. Each Crew makes some spread period from the beginning to end the season of his day of work, This restriction is characterized as an upper bound on the all-out time passing from the beginning season of the primary undertaking work out to that crew until the end time of the last task. Each crew has some idle time interval between two trips. The crew does not exceed the maximum time limit.

In the report,I divide a day into 4 mode such as Early , Afternoon ,Evening and Late mode

Early mode defines as working time generally from 7:00 a.m to 13:00 p.m.

Afternoon mode defines as working time generally from 11:00 am to 20:00 p.m.

Evening mode defines as working time generally from 15:00 p.m. to 22:00 p.m.

Late mode defines as working time generally from 22:00 p.m. to 7:00 a.m.(Next day) .

4.1.1 Assumptions:

1. Delay and Cancellation isn't permitted.
2. An Assignment should be performed by a single crew .
3. Only full time crew are included in this model.
4. Each crew has a fixed cost (day by day).
5. A 30- min. Break can't be cover in idle time.
6. No Overtime is allowed.

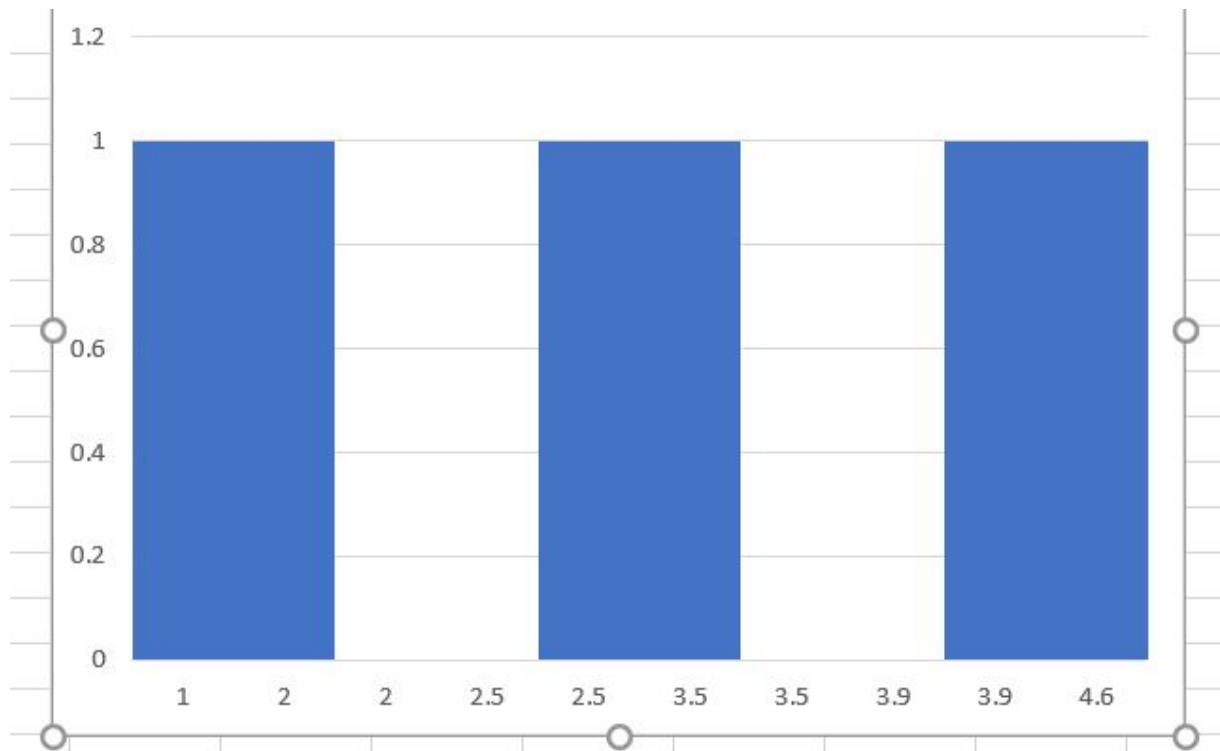
4.2 Model Formation

4.2.1 Model 1

The objective of this modal is minimum the total cost of the crew subject to the all task are covered, eligible task is assigned to crew or not and total working hours of crew is less than or equal to the working limit of the crew

4.2.2 Modal 2

The Objective of this modal is minimize the time length of idle time subject to the number of crew with specific duty is less than or equal to the number of crew available in a day, number of crew with duty mode is greater than or equal to the least number of crew required with day duty, total working hours of the crew is less than or equal to working limit of the crew



Trip Number	Departure Time	Trip Number	Departure Time	Trip Number	Departure Time	Trip Number	Trip Number
1	05:32	45	10:15	89	15:10	133	18:50
2	05:50	46	10:20	90	15:12	134	18:55
3	06:00	47	10:25	91	15:15	135	19:05
4	06:10	48	10:35	92	15:20	136	19:10
5	06:12	49	10:40	93	15:25	137	19:15
6	06:15	50	10:45	94	15:27	138	19:20
7	06:30	51	11:00	95	15:30	139	19:25
8	06:32	52	11:05	96	15:35	140	19:30
9	06:35	53	11:08	97	15:40	141	19:35
10	06:40	54	11:10	98	15:45	142	19:45
11	06:50	55	11:20	99	15:48	143	19:55
12	06:52	56	11:35	100	15:50	144	20:00
13	07:00	57	11:40	101	15:55	145	20:05
14	07:02	58	11:45	102	16:00	146	20:10
15	07:10	59	11:52	103	16:05	147	20:15
16	07:17	60	12:00	104	16:12	148	20:22
17	07:25	61	12:15	105	16:20	149	20:25
18	07:30	62	12:20	106	16:25	150	20:28
19	07:35	63	12:25	107	16:30	151	20:35
20	07:40	64	12:30	108	16:35	152	20:40
21	07:45	65	12:35	109	16:40	153	20:45
22	08:00	66	12:40	110	16:42	154	20:48
23	08:07	67	12:45	111	16:45	155	20:50
24	08:12	68	12:52	112	16:48	156	20:55
25	08:15	69	12:57	113	16:52	157	21:05
26	08:20	70	13:02	114	16:55	158	21:07
27	08:25	71	13:07	115	17:05	159	21:10
28	08:34	72	13:12	116	17:07	160	21:15
29	08:37	73	13:22	117	17:10	161	21:19
30	08:41	74	13:25	118	17:15	162	21:20
31	08:43	75	13:37	119	17:20	163	21:22
32	08:45	76	13:55	120	17:25	164	21:27
33	08:50	77	14:00	121	17:30	165	21:32
34	08:53	78	14:02	122	17:35	166	21:37
35	08:55	79	14:07	123	17:40	167	21:42
36	09:03	80	14:20	124	17:45	168	21:50
37	09:07	81	14:25	125	17:50	169	21:55
38	09:12	82	14:28	126	17:55	170	21:57
39	09:20	83	14:30	127	18:00	171	22:07
40	09:25	84	14:40	128	18:07	172	22:50
41	09:30	85	14:46	129	18:10	173	23:30
42	09:37	86	14:57	130	18:15	174	00:35
43	09:45	87	15:00	131	18:22	175	01:45
44	09:52	88	15:03	132	18:25	176	03:30

4.3 Expression of Solution

In a Delhi Transport Corporation at Bawana Depot, there is 176 trips in a day. The Departure time is mention in above table. The Depot provide 34 Crew in which 9 crews of Early duty, 11 crews of Afternoon duty, 10 crews of Evening duty and 4 crews of late duty.

The Optimal solution of Modal is

Crew Number	Mode	Trip Number	First Trip Departure Time	Last Trip Departure Time	Total Idle time during whole day(minutes)
1	Early	13-24-33-41-61	07:00	12:15	50
2	Early	14-22-32-40-48	07:02	10:35	45
3	Early	15-23-31-42-63	07:10	12:25	48
4	Early	16-25-36-45-50	07:17	10:45	52
5	Early	17-27-35-43-62	07:25	12:20	50
6	Early	18-26-34-47-66	07:30	12:40	39
7	Early	19-29-38-46-65	07:35	12:35	40
8	Early	20-28-37-44-64	07:40	12:30	42
9	Early	21-30-39-49-67	07:45	12:45	43
10	Afternoon	51-71-85-112-124	11:00	17:45	40
11	Afternoon	52-69-83-103-123-134	11:05	18:55	55
12	Afternoon	53-70-80-111-133	11:08	18:53	35
13	Afternoon	54-72-84-106-127	11:10	18:00	30
14	Afternoon	55-74-82-104-125	11:20	17:50	40
15	Afternoon	56-76-97-108-128-135	11:35	19:05	55
16	Afternoon	57-73-81-102-126-136	11:40	19:10	60
17	Afternoon	58-75-86-109-130-137	11:45	19:15	55
18	Afternoon	59-78-98-107-131-140	11:52	19:30	65
19	Afternoon	60-77-101-105-129-138	12:00	19:20	50
20	Afternoon	68-79-100-110-132-139	12:52	19:15	57
21	Evening	87-115-144-152-161	15:00	21:19	40
22	Evening	88-113-143-151-163	15:03	21:22	45
23	Evening	89-116-143-153-162	15:10	21:20	50
24	Evening	90-114-141-154-167	15:12	21:42	38
25	Evening	91-117-146-155-165	15:15	12:32	40
26	Evening	92-119-145-158-168	15:20	21:50	55
27	Evening	93-121-148-156-164	15:25	21:27	49
28	Evening	94-118-147-157-166	15:27	21:37	50
29	Evening	95-120-150-160-169	15:30	21:55	35
30	Evening	96-122-149-159-170	15:35	21:57	40
31	Late	171-176-3-8	22:07	06:32	360
32	Late	172-2-7-11	22:50	06:50	330
33	Late	173-1-6	23:30	06:15	300
34	Late	174-4-9-12	00:35	06:52	280
35	Late	175-5-10	01:45	06:40	90

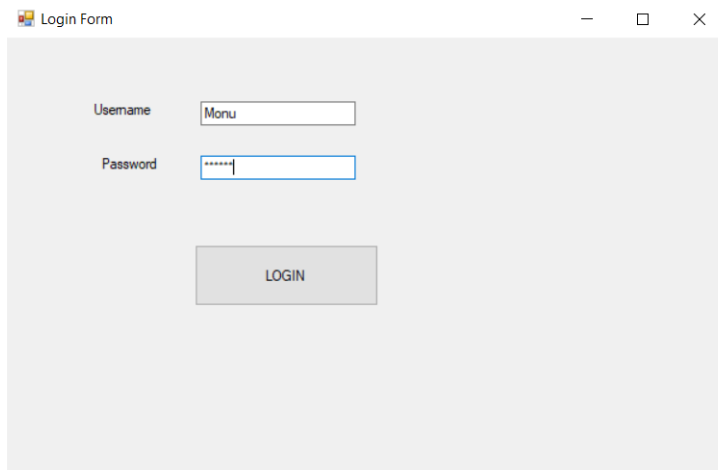
Chapter 5

Computational Results

I develop a database for managing Crews and Buses in depot with the help of Computer Assisted Software which is Microsoft Visual Studio 2019 and Microsoft SQL Serve for front-end and back-end respectively.

5.1 Code

5.1.1 Screen and Code for Login Page



The screenshot shows a Windows application window titled "Login Form". Inside the window, there is a login form with the following elements:

- A "Username" label followed by a text input field containing the text "Monu".
- A "Password" label followed by a password input field containing masked characters "*****".
- A "LOGIN" button located below the password field.

```

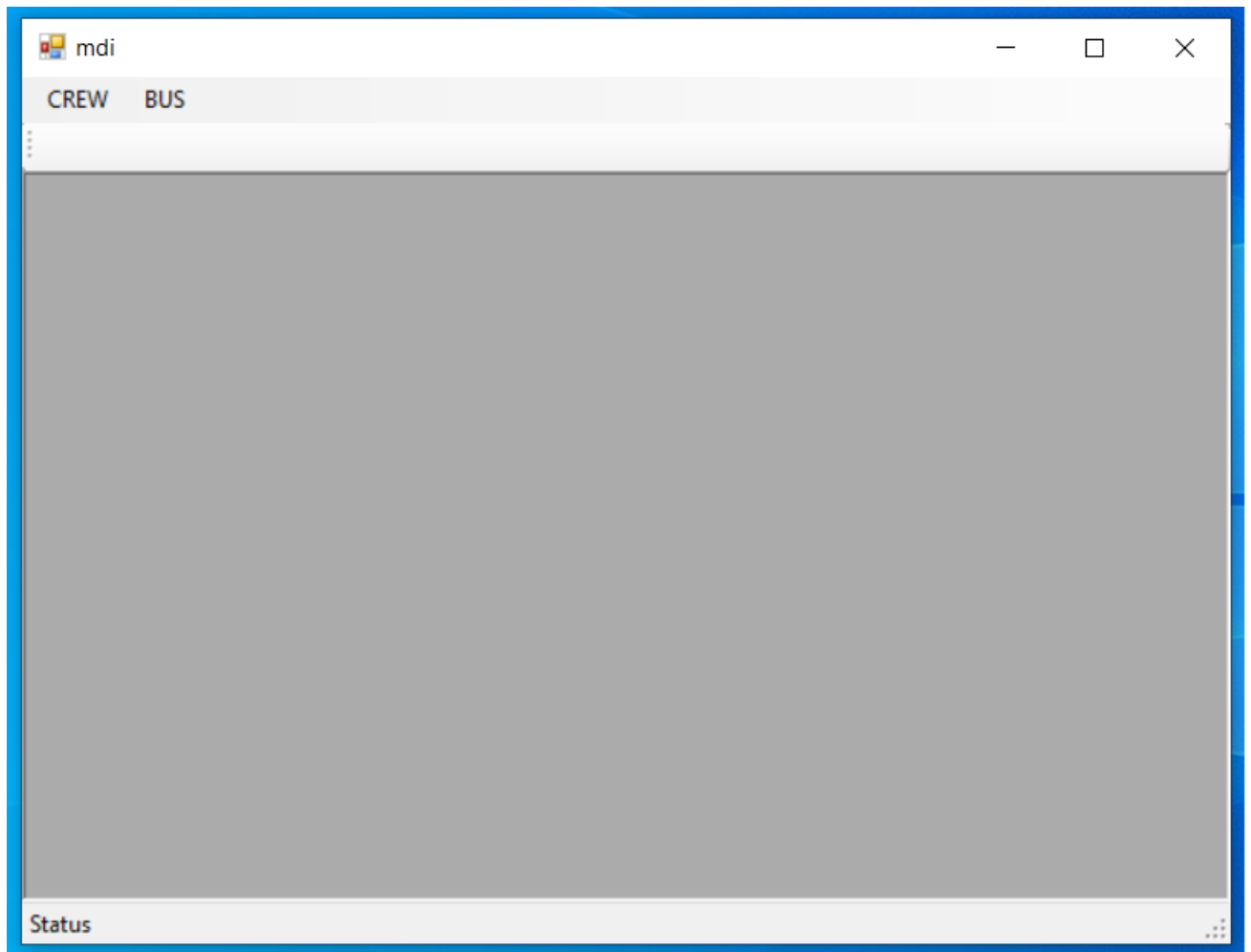
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Transport_management_system
{
    public partial class login : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-3PFNCGBV;Initial
Catalog=Transport;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Mult
iSubnetFailover=False");
        int count = 0;
        public login()
        {
            InitializeComponent();

            private void button1_Click(object sender, EventArgs e)
            {
                SqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "select * from Admin where Username = '" + textBox1.Text
+ "' and Password = '" + textBox2.Text + "' COLLATE Latin1_General_CS_AS ";
                cmd.ExecuteNonQuery();
                DataTable dt = new DataTable();
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                da.Fill(dt);
                count = Convert.ToInt32(dt.Rows.Count.ToString());
                if (count == 0)
                {
                    MessageBox.Show("Username and Password does not match");
                }
                else
                {
                    this.Hide();
                    mdi mu = new mdi();
                    mu.Show();
                }
            }

            private void login_Load(object sender, EventArgs e)
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
                con.Open();
            }
        }
    }
}

```

5.1.2 Screen and Code for MDI Form



```

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;

namespace Transport_management_system
{
    public partial class mdi : Form
    {
        private int childFormNumber = 0;

        public mdi()
        {
            InitializeComponent();
        }

        private void ShowNewForm(object sender, EventArgs e)
        {
            Form childForm = new Form();
            childForm.MdiParent = this;
            childForm.Text = "Window " + childFormNumber++;
            childForm.Show();
        }

        private void OpenFile(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            openFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
            if (openFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = openFileDialog.FileName;
            }
        }

        private void SaveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            saveFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
            if (saveFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = saveFileDialog.FileName;
            }
        }

        private void ToolBarToolStripMenuItem_Click(object sender, EventArgs e)
        {
            toolStrip.Visible = toolBarToolStripMenuItem.Checked;
        }

        private void StatusBarToolStripMenuItem_Click(object sender, EventArgs e)
        {

```

```

        statusStrip.Visible = statusBarToolStripMenuItem.Checked;
    }

    private void CascadeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        LayoutMdi(MdiLayout.Cascade);
    }

    private void TileVerticalToolStripMenuItem_Click(object sender, EventArgs e)
    {
        LayoutMdi(MdiLayout.TileVertical);
    }

    private void TileHorizontalToolStripMenuItem_Click(object sender, EventArgs e)
    {
        LayoutMdi(MdiLayout.TileHorizontal);
    }

    private void ArrangeIconsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        LayoutMdi(MdiLayout.ArrangeIcons);
    }

    private void CloseAllToolStripMenuItem_Click(object sender, EventArgs e)
    {
        foreach (Form childForm in MdiChildren)
        {
            childForm.Close();
        }
    }

    private void addCREWToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Add_Crew ad = new Add_Crew();
        ad.Show();
    }

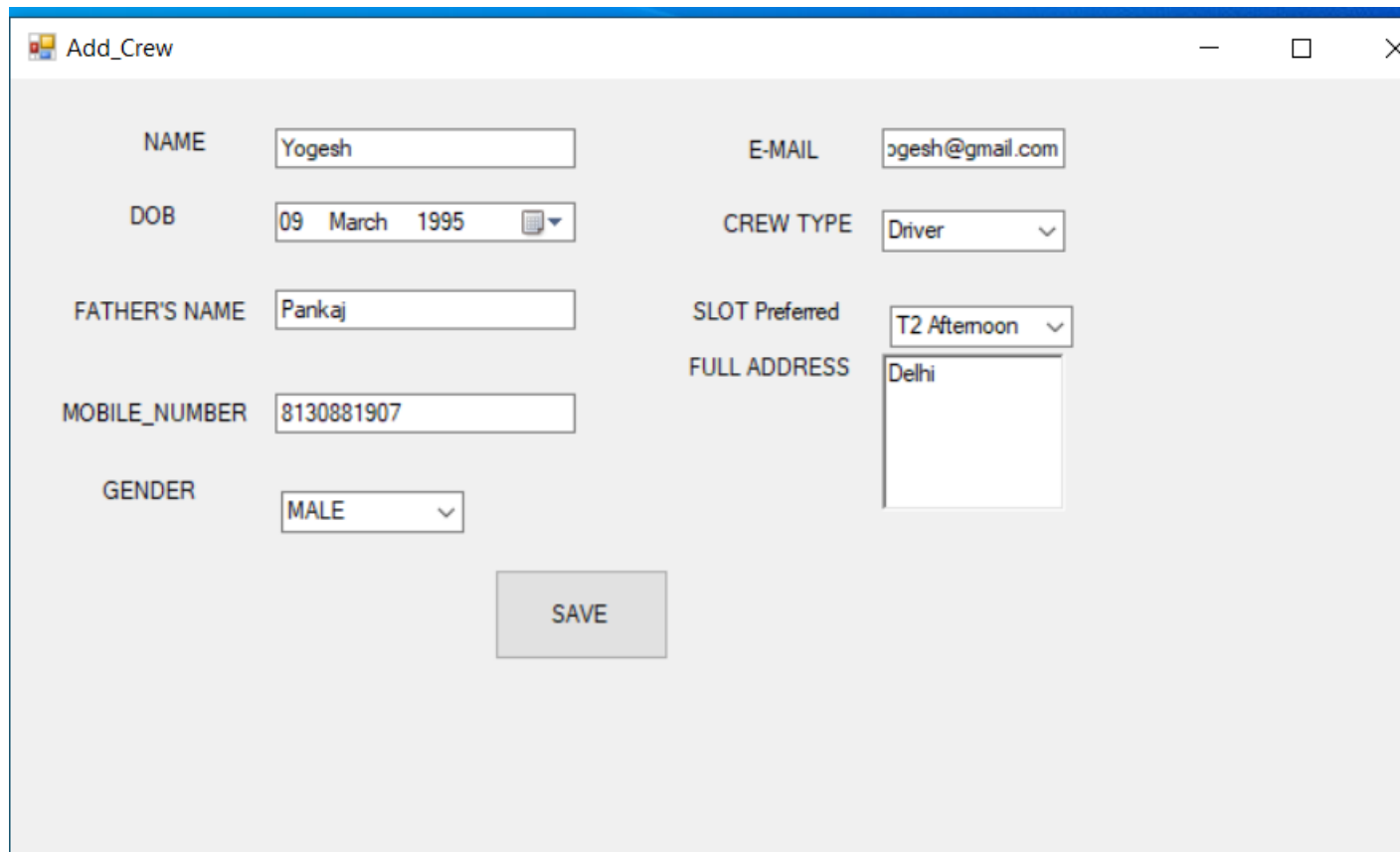
    private void VIEWCREWToolStripMenuItem_Click(object sender, EventArgs e)
    {
        VIEW_CREW vd = new VIEW_CREW();
        vd.Show();
    }

    private void addBusToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Add_Bus ab = new Add_Bus();
        ab.Show();
    }
}

    private void updateDetailsOfBusToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        View_bus mo = new View_bus();
        mo.Show();
    }
}
}

```

5.1.3 Screen and Code for Add Crew



The screenshot shows a window titled "Add_Crew" with a light gray background. The form contains the following fields and controls:

Field Label	Value / Selection
NAME	Yogesh
E-MAIL	yogesh@gmail.com
DOB	09 March 1995
CREW TYPE	Driver
FATHER'S NAME	Pankaj
SLOT Preferred	T2 Afternoon
MOBILE_NUMBER	8130881907
FULL ADDRESS	Delhi
GENDER	MALE

A "SAVE" button is located at the bottom center of the form.


```

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.Data.SqlClient;

namespace Transport_management_system
{
    public partial class Add_Crew : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-3PFNCGBV;Initial
Catalog=Transport;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Mult
iSubnetFailover=False");
        public Add_Crew()
        {
            InitializeComponent();

            private void button1_Click(object sender, EventArgs e)
            {
                SqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "insert into MEMBER(NAME, DOB , FATHER_NAME , MOBILE ,
ADDRESS , CREW_TYPE,EMAIL , GENDER , SLOT) values (@name , @dob ,@father , @mobile
,@address ,@crew ,@email , @gender , @slot)";
                cmd.Parameters.Clear();
                cmd.Parameters.AddWithValue("@name", textBox1.Text);
                cmd.Parameters.AddWithValue("@dob", dateTimePicker1.Text);
                cmd.Parameters.AddWithValue("@father", textBox2.Text);
                cmd.Parameters.AddWithValue("@mobile", textBox3.Text);
                cmd.Parameters.AddWithValue("@address", richTextBox1.Text);
                cmd.Parameters.AddWithValue("@crew", comboBox1.SelectedIndex);
                cmd.Parameters.AddWithValue("@email", textBox4.Text);
                cmd.Parameters.AddWithValue("@gender", comboBox2.SelectedIndex);
                cmd.Parameters.AddWithValue("@slot", comboBox3.SelectedIndex);
                con.Open();
                if (cmd.ExecuteNonQuery() > 0)
                {
                    cmd.CommandText = "SELECT Id FROM MEMBER WHERE NAME = '" +
textBox1.Text + "' AND MOBILE = '" + textBox3.Text + "'";
                    int Id = (int)cmd.ExecuteScalar();
                    MessageBox.Show("Record sucessfully");
                    MessageBox.Show("CREW ID:" + Id.ToString());
                }
                con.Close();
                textBox1.Text = "";
                textBox2.Text = "";
                textBox3.Text = "";
                textBox4.Text = "";
                richTextBox1.Text = "";
                comboBox1.Text = "";
                comboBox3.Text = "";
                comboBox2.Text = "19";
            }
        }
    }
}

```

5.1.4 Screen and Code for View Crew and Update its Information

VIEW_CREW

	Id	NAME	DOB	FATHER_NAME	MOBILE	ADDRESS
▶	1	Rahul	08-03-2021	Amit	9876543219	delhi
*						

NAME

DOB

FATHER'S NAME

MOBILE_NUMBER

UPDATE

E-MAIL

ID

CREW TYPE

GENDER

FULL ADDRESS

Slot Preferred

```

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.Data.SqlClient;

namespace Transport_management_system
{
    public partial class VIEW_CREW : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-3PFNCGBV;Initial
Catalog=Transport;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Mult
iSubnetFailover=False");
        public VIEW_CREW()
        {
            InitializeComponent();
        }

        private void VIEW_CREW_Load(object sender, EventArgs e)
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
            con.Open();

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "select * from MEMBER";
            cmd.ExecuteNonQuery();
            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(dt);
            dataGridView1.DataSource = dt;
        }

        private void textBox1_KeyUp(object sender, KeyEventArgs e)
        {
            try
            {
                dataGridView1.Columns.Clear();
                dataGridView1.Refresh();
                SqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "select * from MEMBER where NAME like ('%" +
textBox1.Text + "%')";
                cmd.ExecuteNonQuery();
                DataTable dt = new DataTable();
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                da.Fill(dt);
                dataGridView1.DataSource = dt;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString());
            }
        }
    }
}

```

```

    }
}

private void textBox4_TextChanged(object sender, EventArgs e)
{
}

private void textBox5_TextChanged(object sender, EventArgs e)
{
}

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
{
    int i;
    i = Convert.ToInt32(dataGridView1.SelectedCells[0].Value.ToString());
    MessageBox.Show(i.ToString());
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = " select * from MEMBER where ID = " + i + "";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();

    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    foreach (DataRow dr in dt.Rows)
    {
        textBox6.Text = dr["Id"].ToString();
        textBox2.Text = dr["NAME"].ToString();
        dateTimePicker1.Text = dr["DOB"].ToString();
        textBox3.Text = dr["FATHER_NAME"].ToString();
        textBox4.Text = dr["MOBILE"].ToString();
        textBox5.Text = dr["EMAIL"].ToString();
        comboBox1.Text = dr["CREW_TYPE"].ToString();
        richTextBox1.Text = dr["ADDRESS"].ToString();
        comboBox2.Text = dr["GENDER"].ToString();
        comboBox3.Text = dr["SLOT"].ToString();
    }
}

private void button1_Click(object sender, EventArgs e)
{
    int i;
    i = Convert.ToInt32(dataGridView1.SelectedCells[0].Value.ToString());
    try
    {
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = " update MEMBER set NAME='" + textBox2.Text +
        "','EMAIL='" + textBox5.Text + "','MOBILE = '" + textBox4.Text + "','FATHER_NAME = '" +
        textBox3.Text + "','ADDRESS = '" + richTextBox1.Text + "','DOB='" + dateTimePicker1.Text
        + "',' CREW_TYPE = '" + comboBox1.Text + "',' GENDER = '" + comboBox2.Text + "',' , SLOT = '" +
        comboBox3.Text + "' where Id = " + i + "";

        cmd.ExecuteNonQuery();

        disp_students();
    }
}

```

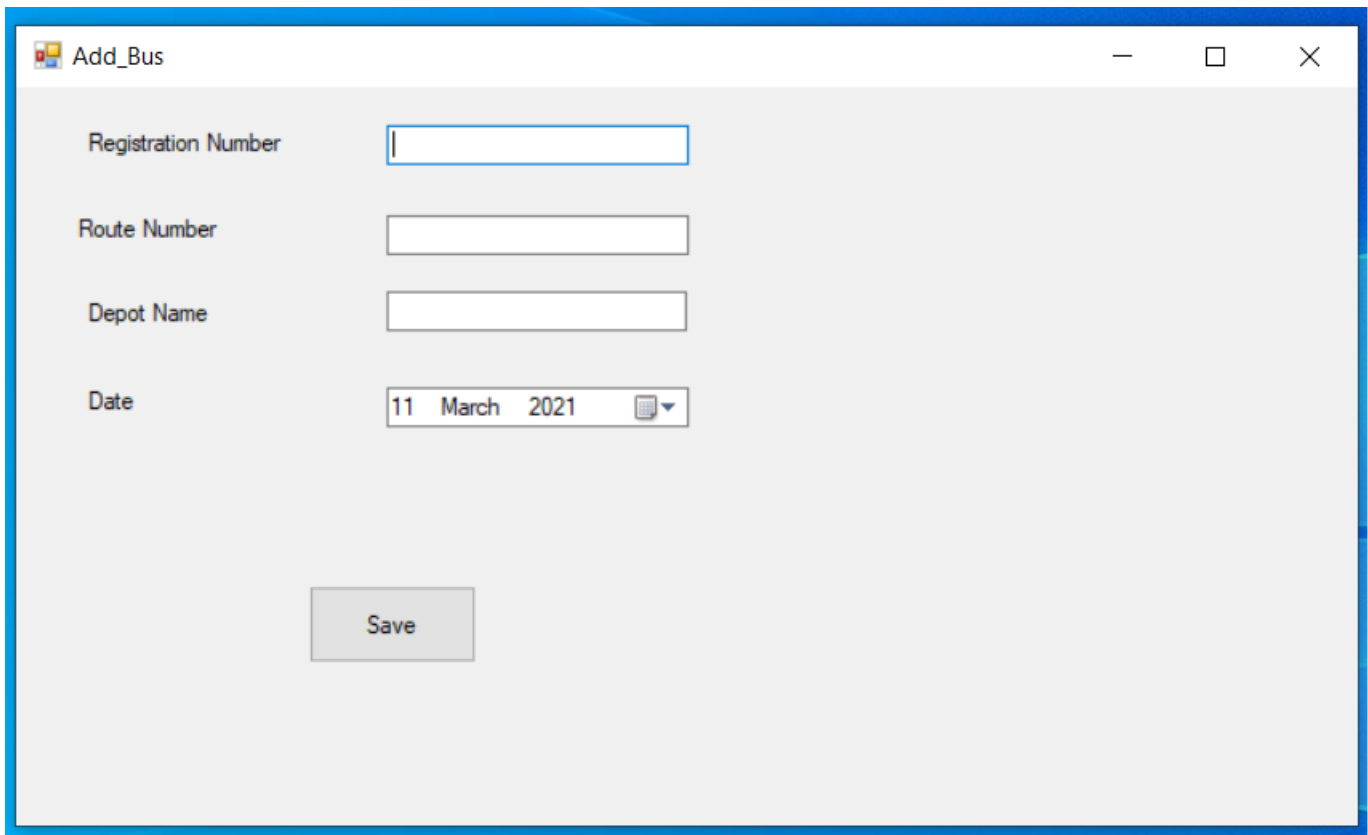
```

        MessageBox.Show("record update successfully");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
public void disp_students()
{
    //con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select * from MEMBER";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    dataGridView1.DataSource = dt;

    //con.Close();
}
}
}

```

5.1.5 Screen and Code for Add Buses



The screenshot displays a software window titled "Add_Bus" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains four input fields arranged vertically on the left side, each with a corresponding label to its left:

- Registration Number**: A text input field with a blue border and a vertical cursor.
- Route Number**: A text input field with a grey border.
- Depot Name**: A text input field with a grey border.
- Date**: A date picker control showing "11 March 2021" and a calendar icon.

Below these fields, centered horizontally, is a grey "Save" button.

```

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.Data.SqlClient;

namespace Transport_management_system
{
    public partial class Add_Bus : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-3PFNCGBV;Initial
Catalog=Transport;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Mult
iSubnetFailover=False");

        public Add_Bus()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "insert into Bus (REGISTRATION_NUMBER , ROUTE_NUMBER ,
DEPOT_NAME , DATE) values ('"+textBox1.Text+"', '"+textBox2.Text+"',
'"+textBox3.Text+"', '"+dateTimePicker1.Text+"')";

            con.Open();
            if (cmd.ExecuteNonQuery() > 0)
            {
                cmd.CommandText = "SELECT Id FROM Bus WHERE REGISTRATION_NUMBER = '" +
textBox1.Text + "'";
                int Id = (int)cmd.ExecuteScalar();
                MessageBox.Show("Record sucessfully");
                MessageBox.Show("UNIQUE BUS ID ACCORDING TO DEPOT:" + Id.ToString());
            }
            con.Close();

            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
        }
    }
}

```

5.1.6 Screen and Code for View , Update and Delete details of Bus

View_bus

	Id	REGISTRATION_	ROUTE_NUMBER	DEPOT_NAME	DATE
▶	2	DL1AC3709	119	BAWANA	09-03-2020
*					

Registration Number

Depot Name

Id

Route Number

Date

Update

Delete


```

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.Data.SqlClient;

namespace Transport_management_system
{
    public partial class View_bus : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=LAPTOP-3PFNCGBV;Initial
Catalog=Transport;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;Mult
iSubnetFailover=False");
        public View_bus()
        {
            InitializeComponent();
        }

        private void View_bus_Load(object sender, EventArgs e)
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
            con.Open();

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "select * from Bus";
            cmd.ExecuteNonQuery();
            DataTable dt = new DataTable();
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(dt);
            dataGridView1.DataSource = dt;
        }

        private void textBox1_KeyUp(object sender, KeyEventArgs e)
        {
            try
            {
                dataGridView1.Columns.Clear();
                dataGridView1.Refresh();
                SqlCommand cmd = con.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = "select * from Bus where REGISTRATION_NUMBER like
('%" + textBox1.Text + "%')";
                cmd.ExecuteNonQuery();
                DataTable dt = new DataTable();
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                da.Fill(dt);
                dataGridView1.DataSource = dt;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString());
            }
        }
    }
}

```

```

    }

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
    {
        int i;
        i = Convert.ToInt32(dataGridView1.SelectedCells[0].Value.ToString());
        MessageBox.Show(i.ToString());
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = " select * from Bus where Id = " + i + " ";
        cmd.ExecuteNonQuery();
        DataTable dt = new DataTable();

        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(dt);
        foreach (DataRow dr in dt.Rows)
        {

            textBox2.Text = dr["REGISTRATION_NUMBER"].ToString();
            dateTimePicker1.Text = dr["DATE"].ToString();
            textBox3.Text = dr["ROUTE_NUMBER"].ToString();
            textBox4.Text = dr["DEPOT_NAME"].ToString();
            textBox5.Text = dr["Id"].ToString();

        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        int i;
        i = Convert.ToInt32(dataGridView1.SelectedCells[0].Value.ToString());
        try
        {

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = " update Bus set REGISTRATION_NUMBER=' " +
textBox2.Text + "',DEPOT_NAME = ' " + textBox4.Text + "',ROUTE_NUMBER = ' " +
textBox3.Text + "',DATE=' " + dateTimePicker1.Text + " ' where Id = " + i + " ";

            cmd.ExecuteNonQuery();

            disp_crews();

            MessageBox.Show("record update successfully");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    public void disp_crews()
    {
        //con.Open();
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "select * from Bus";
        cmd.ExecuteNonQuery();

```

```

        DataTable dt = new DataTable();
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(dt);
        dataGridView1.DataSource = dt;

        //con.Close();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        int i;
        i = Convert.ToInt32(dataGridView1.SelectedCells[0].Value.ToString());
        try
        {
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = " delete from Bus where Id = " + i + "";

            cmd.ExecuteNonQuery();

            disp_crews();

            MessageBox.Show("record update successfully");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

```

Bibliography

- [1] Fischetti, M., Martello, S., Toth, P., 1987. The Fixed Job Schedule Problem with Spread-Time Constraints. *Operations Research* 35, 849–858.
- [2] Fischetti, M., Martello, S., Toth, P., 1989. The Fixed Job Schedule Problem with Working-Time Constraints. *Operations Research* 37, 395–403
- [3] Ugur Eliyi , Deniz T Eliyi , Levent Kandiller ,2016 . A Bus Crew Scheduling Problem with Eligibility Constraints and timit limitations.
- [4] Mingming Chen and Huimin Niu , 2012 . A Model for Bus Crew Scheduling Problem with Multiple Duty Type.
- [5] Avishai (Avi) Ceder,Public Transit Planning and Operation; Theory, modelling and practice