Final Project

CL1004 – Object Oriented Programming - Lab

Fall 2023

Submitted by

Muhammad Muzammil Noor	22F-3274
Rafay Adeel	22F-3327

BCS-2C-1

Submitted To

Mr. Muhammad Hannan Farooq

Transport Management System

Table of Contents

Driver Code (Main_Code.cpp)

```
#include "pch.h"
#include <fstream> //File stream for file handling
#include <string>
#include <msclr/marshal_cppstd.h> //library for converting System::String to
std::string
#include "Main_Interface.h" //our main interface, has all the classes
#include <math.h> //using the to just once in the code to calculate square root in
Distance.h
TMS TMS_Main;
int disx = 0;
float r1 = 6, r2 = 6;
using namespace System;
using namespace System::Windows::Forms;
using namespace System::Collections::Generic;
using namespace std;
#pragma region Froms
//header files for all the forms
#include "LogIn.h"
#include "SignUp.h"
#include "Admin_Form.h"
#include "Customer_Form.h"
#include "Driver_Form.h"
#include "Distance.h"
#include "RatingGiver.h"
#include "Driver_Service.h"
#include "Vehicle_Service.h"
#pragma endregion
[STAThread]
int main() {
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew TMS_Forms::Driver_Form());
	return 0;
}
```

Main Interface (Main_Interface.h)

```
#pragma once
#include "Viktor.h"
#include "Viktor.cpp"
bool isNum(char c); //checks if a character is a number (0  to 9)
int StringToInt(std::string str); //takes a string and converts it into an integer
float StringToFloat(std::string str); //takes a string and converts it into a
floating point number
bool isValidCNIC(std::string s); //takes a string and checks if the string follows a
valid CNIC format
bool isValidName(std::string s); //checks if the string is a name (contains only
alphabets)
bool isNull(std::string s); //checks if a string contains anything other than
whitespaces
bool isInt(std::string s); //checks if a string can be turned into an integer
std::string IntToString(int x); //turns an integer into a string
std::string FloatToString(float x); //turns a float into a string
bool ContainsSpaces(std::string s); //checks if a non-null string contains any
spaces
#pragma region Vehicle Model Class
//small struct for vehicle model
struct VclModel {
public:
	std::string Company;
	std::string Model;
	int Year;
	VclModel(std::string Company, std::string Model, int year);
	VclModel();
};
#pragma endregion
#pragma region Account Class
//Account class, contains basic data like account name, cnic balance and password
class Account {
protected:
	std::string FirstName;
	std::string LastName;
	std::string password;
	std::string CNIC;
	int balance;
public:
	std::string getID();
	void setID(std::string Fname, std::string Lname, std::string ID);
	std::string getPass();
	std::string getFName();
	std::string getLName();
	int getbal();
	void addbal(int x);
	void deductbal(int x);
};
#pragma endregion
#pragma region Order Class
//Order class, tracks the cnic of driver, customer and some other data such as
timestamps of placement and completion times
class Order {
protected:
	int PlacedOn;
	int CompletedOn;
	int cost;
	std::string type;
	int OrderID;
	std::string customerID;
	std::string driverID;
	int vehicleID;
	bool Accepted;
	bool Complete;
public:
	Order(); //default constructor, needed to use vectors (Viktors as i like to
call them)
	Order(std::string C_id, std::string D_id, int V_id, std::string type, int id,
int placed, int completed, bool accepted, bool complete, int cost); //constructor
#pragma region Getters
	//just a whole lot of getters
	int getID();
	std::string getCID();
	std::string getDID();
	std::string getType();
	int getVID();
	int getPlaced();
	int getComplete();
	int getCost();
	bool getAccepted();
	bool getCompleted();
#pragma endregion
	void Accept(); //marks order as accepted
	void oComplete(); //marks order as completed
	void Cancel(); //marks order as completed/rejected
	friend std::ostream& operator << (std::ostream& out, const Order O);
//insertion operator for easy file writing
};
#pragma endregion
//Vehice class
#pragma region Vehicle Class
//Vehicle Class
class Vehicle {
public:
	VclModel Model;
	std::string driverID;
	int vehicleID;
	float rating;
	Viktor<float> scores; //each vehicle has this vector (Viktor) of scores, its
used to calculate the rating
	std::string ServiceType;
	std::string Type;
	Vehicle();
	Vehicle(std::string driverID, std::string Type, int ID, VclModel Model,
std::string ServiceType); //constructor
	int getID();
	float ComputeAndReturnRating(); //calculates and returns the rating
	friend std::ostream& operator << (std::ostream& out, const Vehicle V);
//insertion operator for ez writing (again)
};
#pragma endregion
#pragma region Customer Class
```

```
//Customer class, inhereted from Account. has no additional memberx
class Customer :public Account {
private:
public:
	Customer();//default constructor cuz Viktors
	Customer(std::string FName, std::string LName, std::string CNIC, std::string
password, int balance); //constructor
	friend std::ostream& operator << (std::ostream& out, const Customer C);
//insertiooon operatooooor
};
#pragma endregion
#pragma region Driver Class
//Driver Class, has some additional members
class Driver :public Account {
private:
	float rating;
	int experience;
	bool isFree; //tells wether the driver is free or not
public:
	Viktor<Vehicle> Vehicles; //a list of vehicles
	Viktor<float> scores; //and some scores to calculate rating
	Driver(); //default constructor
	Driver(std::string FName, std::string LName, std::string CNIC, std::string
password, int balance, int exp, bool freedom); //constructor
	int getExp();
	int getFreedom();
	void removeVehicle(int x);
	float ComputeAndReturnRating();
	void setFreedom(bool x);
	friend std::ostream& operator << (std::ostream& out, const Driver D);
};
#pragma endregion
#pragma region Main TMS Class
//Transport Management System Class, essentially the core of this project. it links
all the above classes together and operates the system
class TMS {
public:
	Viktor<Customer> C_Accounts; //list of customers
	Viktor<Driver> D_Accounts;//list of drivers
	Viktor<Order> Orders;//list of orders
#pragma region Tools for Saving and Loading Data
	void ClearLoadedData(); //clears all the data loaded in all the Viktors
	void ClearSavedData(); //Clears most of the txt files in Data folder
	void SaveLoadedData(); //Dumps the data from the Viktors into text files
	void LoadSavedData(); //Reads data from text files and then assigns that data
to appropriate objects
#pragma endregion
#pragma region Account Tools
	void DeleteCAccount(int x); //Delete Customer account
	void DeleteDAccount(int x); //Delete Driver Account
	void MakeOrder(Customer C, Driver D, Vehicle V, std::string type, int cost);
//Function for adding a new order
	void CompleteOrder(int x); //self explanatory
	void AcceptOrder(int x); //does what you expect
	void RejectOrder(int x); //same here
	void CancelOrder(int x); //and here
	void SortR(); //Sorts drivers by rating
	void SortE(); //Sorts drivers by experience
```

```cpp
#pragma endregion
#pragma region Tools for Finding and checking Repitition
	bool isUniqueOID(int x); //checks if the order id is unique
	int FindCNIC(std::string s); //Finds a cnic in the array of customers, returns index
	int FindCNIC2(std::string s); //Finds the cnic in the list of drivers, returns index
	int FindID(std::string str, int x); //Finds the vehicle id the list of vehicles of a specific driver
	bool isUniqueCNIC(std::string s); //Checks if the CNIC is unique
	bool idUniqueLisence(int x); //Checks if the vehicle id is unique
#pragma endregion
};
#pragma endregion
template <typename T> //template functon to swap
void swap(T* v1, T* v2);
```

Implementation (Main_Implementation.cpp)

```cpp
#include "pch.h"
#include <fstream>
#include <vector>
#include <string>
#include "Main_Interface.h"
#pragma region Vehicle Model Class
VclModel::VclModel(std::string Company, std::string Model, int year) {
	this->Company = Company;
	this->Model = Model;
	this->Year = year;
}
VclModel::VclModel() {

}
#pragma endregion
#pragma region Account Class
std::string Account::getID() {
	return CNIC;
}
void Account::setID(std::string Fname, std::string Lname, std::string ID) {
	CNIC = ID;
	FirstName = Fname;
	LastName = Lname;
}
std::string Account::getPass() {
	return this->password;
}
std::string Account::getFName() {
	return FirstName;
}
std::string Account::getLName() {
	return LastName;
}
int Account::getbal() {
	return balance;
}
void Account::addbal(int x) {
        this->balance += x;
}
void Account::deductbal(int x) {
        this->balance -= x;
}
#pragma endregion
#pragma region Order Class
Order::Order(std::string C_id, std::string D_id, int V_id, std::string type, int id,
int placed, int completed, bool accepted, bool complete, int cost) {
        this->customerID = C_id;
        this->driverID = D_id;
        this->vehicleID = V_id;
        this->type = type;
        this->OrderID = id;
        this->PlacedOn = placed;
        this->CompletedOn = completed;
        this->Accepted = accepted;
        this->Complete = complete;
        this->cost = cost;
}
Order::Order() {
        this->customerID = "";
        this->driverID = "";
        this->vehicleID = 0;
        this->type = "";
        this->OrderID = 0;
        this->PlacedOn = 0;
        this->CompletedOn = 0;
        this->Accepted = 0;
        this->Complete = 0;
        this->cost = 0;
}
#pragma region Getters
int Order::getID() {
        return this->OrderID;
}
std::string Order::getCID() {
        return this->customerID;
}
std::string Order::getDID() {
        return this->driverID;
}
std::string Order::getType() {
        return this->type;
}
int Order::getVID() {
        return this->vehicleID;
}
int Order::getPlaced() {
        return this->PlacedOn;
}
int Order::getComplete() {
        return this->CompletedOn;
}
int Order::getCost() {
        return this->cost;
}
bool Order::getAccepted() {
	return this->Accepted;
}
bool Order::getCompleted() {
	return this->Complete;
}
#pragma endregion
std::ostream& operator << (std::ostream& out, const Order O) {
	out << O.customerID << " " << O.driverID << " " << O.vehicleID << " " <<
O.type << " " << O.OrderID << " " << O.PlacedOn << " " << O.CompletedOn << " " <<
O.Accepted << " " << O.Complete << " " << O.cost;
	return out;
}
void Order::Accept() {
	this->Accepted = true;
}
void Order::Cancel() {
	this->CompletedOn = time(0);
}
void Order::oComplete() {
	this->Complete = true;
	this->CompletedOn = time(0);
}
#pragma endregion
#pragma region Vehicle Class
Vehicle::Vehicle(std::string driverID, std::string Type, int ID, VclModel Model,
std::string ServiceType) {
	this->driverID = driverID;
	this->Type = Type;
	this->vehicleID = ID;
	this->Model = Model;
	this->ServiceType = ServiceType;
}
Vehicle::Vehicle() {
	this->driverID = "";
	this->Type = "";
	this->vehicleID = 0;
	this->Model.Company = "";
	this->Model.Model = "";
	this->Model.Year = 0;
	this->ServiceType = "";
}
int Vehicle::getID() {
	return this->vehicleID;
}
float Vehicle::ComputeAndReturnRating() {
	this->rating = 0;
	for (int i = 0; i < this->scores.size(); i++) {
		rating += *this->scores[i];
	}
	if (this->scores.size()) this->rating /= this->scores.size();
	int x = this->rating * 100;
	return (x / 100.0);
}
std::ostream& operator << (std::ostream& out, const Vehicle V) {
	out << V.driverID << " " << V.Type << " " << V.vehicleID << " " <<
V.Model.Company << " " << V.Model.Model << " " << V.Model.Year << " " <<
V.ServiceType;
	return out;
```

```
}
#pragma endregion
#pragma region Customer Class
Customer::Customer() {

}
Customer::Customer(std::string FName, std::string LName, std::string CNIC,
std::string password, int balance) {
	this->FirstName = FName;
	this->LastName = LName;
	this->CNIC = CNIC;
	this->password = password;
	this->balance = balance;
}
std::ostream& operator << (std::ostream& out, const Customer C) {
	out << C.FirstName << " " << C.LastName << " " << C.CNIC << " " << C.password
<< " " << C.balance << std::endl;
	return out;
}
#pragma endregion
#pragma region Driver Class
Driver::Driver(std::string FName, std::string LName, std::string CNIC, std::string
password, int balance, int exp, bool freedom) {
	this->FirstName = FName;
	this->LastName = LName;
	this->CNIC = CNIC;
	this->password = password;
	this->balance = balance;
	this->experience = exp;
	this->isFree = freedom;
}
Driver::Driver() {
	this->FirstName = "";
	this->LastName = "";
	this->CNIC = "";
	this->password = "";
	this->balance = 0;
	this->experience = 0;
	this->isFree = 0;
}
int Driver::getExp() {
	return this->experience;
}
int Driver::getFreedom() {
	return this->isFree;
}
float Driver::ComputeAndReturnRating() {
	this->rating = 0;
	for (int i = 0; i < this->scores.size(); i++) {
		rating += *(this->scores[i]);
	}
	if (this->scores.size()) rating /= this->scores.size();
	int x = rating * 100;
	return (x / 100.0);
}
void Driver::setFreedom(bool x) {
	this->isFree = x;
}
```

```
void Driver::removeVehicle(int x) {
	for (int i = x; i < this->Vehicles.size() - 1; i++) {
		*this->Vehicles[i] = *this->Vehicles[i + 1];
	}
	this->Vehicles.pop_back();
}
std::ostream& operator << (std::ostream& out, const Driver D) {
	out << D.FirstName << " " << D.LastName << " " << D.CNIC << " " << D.password
<< " " << D.balance << " " << D.experience << " " << D.isFree << std::endl;
	return out;
}
#pragma endregion
#pragma region Main TMS Class
#pragma region Tools for Saving and Loading Data
void TMS::ClearLoadedData() {
	int i, j, k;
	int x, y, z;
	C_Accounts.clear();
	Orders.clear();
	x = D_Accounts.size();
	for (i = 0; i < x; i++) {
		D_Accounts[i]->scores.clear();
		y = D_Accounts[i]->Vehicles.size();
		for (j = 0; j < y; j++) {
			D_Accounts[i]->Vehicles[j]->scores.clear();
		}
		D_Accounts[i]->Vehicles.clear();
	}
	D_Accounts.clear();

void TMS::ClearSavedData() {
	std::ofstream DataEreaser;
	DataEreaser.open("Data/Customers.txt");
	DataEreaser.close();
	DataEreaser.open("Data/Drivers.txt");
	DataEreaser.close();
	DataEreaser.open("Data/Vehicles.txt");
	DataEreaser.close();
	DataEreaser.open("Data/Ratings.txt");
	DataEreaser.close();
	DataEreaser.open("Data/VRatings.txt");
	DataEreaser.close();
	DataEreaser.open("Data/Orders");
	DataEreaser.close();
}
void TMS::SaveLoadedData() {
	std::ofstream SaveData;
	int i, j, k;
	SaveData.open("Data/Customers.txt");
	for (i = 0; i < C_Accounts.size(); i++) {
		SaveData << *C_Accounts[i];
	}
	SaveData.close();
	SaveData.open("Data/Drivers.txt");
	for (i = 0; i < D_Accounts.size(); i++) {
		SaveData << *D_Accounts[i];
	}
	SaveData.close();
```

```
    SaveData.open("Data/Ratings.txt");
    for (i = 0; i < D_Accounts.size(); i++) {
        for (j = 0; j < D_Accounts[i]->scores.size(); j++) {
            SaveData << D_Accounts[i]->getID() << " " << *(D_Accounts[i]-
>scores[j]) << std::endl;
        }
    }
    SaveData.close();
    SaveData.open("Data/Vehicles.txt");
    for (i = 0; i < D_Accounts.size(); i++) {
        for (j = 0; j < D_Accounts[i]->Vehicles.size(); j++) {
            SaveData << *(D_Accounts[i]->Vehicles[j]) << std::endl;
        }
    }
    SaveData.close();
    SaveData.open("Data/VRatings.txt");
    for (i = 0; i < D_Accounts.size(); i++) {
        for (j = 0; j < D_Accounts[i]->Vehicles.size(); j++) {
            for (k = 0; k < D_Accounts[i]->Vehicles[j]->scores.size(); k++)
{
                SaveData << D_Accounts[i]->getID() << " " <<
D_Accounts[i]->Vehicles[j]->vehicleID << " " << *(D_Accounts[i]->Vehicles[j]-
>scores[k]) << std::endl;
            }
        }
    }
    SaveData.close();
    SaveData.open("Data/Orders.txt");
    for (i = 0; i < Orders.size(); i++) {
        SaveData << *Orders[i] << std::endl;
    }
}
void TMS::LoadSavedData() {
    ClearLoadedData();
    std::ifstream LoadData;
    std::string st1, st2, st3, st4, st5, st6, st7, st8, st9, st10;
    LoadData.open("Data/Customers.txt");
    if (LoadData.is_open()) {
        st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7 = "";
st8 = ""; st9 = ""; st10 = "";
        while (!LoadData.eof()) {
            st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7
= ""; st8 = ""; st9 = ""; st10 = "";
            LoadData >> st1 >> st2 >> st3 >> st4 >> st5;
            Customer temp(st1, st2, st3, st4, StringToInt(st5));
            if (st1 != "") C_Accounts.push_back(temp);
        }
        if (st1 != "" && C_Accounts.size()) C_Accounts.pop_back();
    }
    LoadData.close();
    st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7 = ""; st8 =
""; st9 = ""; st10 = "";
    LoadData.open("Data/Drivers.txt");
    if (LoadData.is_open()) {
        while (!LoadData.eof()) {
            st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7
= ""; st8 = ""; st9 = ""; st10 = "";
            LoadData >> st1 >> st2 >> st3 >> st4 >> st5 >> st6 >> st7;
            Driver temp(st1, st2, st3, st4, StringToInt(st5),
StringToInt(st6), StringToInt(st7));
            if (st1 != "") D_Accounts.push_back(temp);
        }
        if (st1 != "" && D_Accounts.size()) D_Accounts.pop_back();
    }
    LoadData.close();
    if (D_Accounts.size() != 0) {
        st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7 = "";
st8 = ""; st9 = ""; st10 = "";
        LoadData.open("Data/Ratings.txt");
        if (LoadData.is_open()) {
            while (!LoadData.eof()) {
                st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 =
""; st7 = ""; st8 = ""; st9 = ""; st10 = "";
                LoadData >> st1 >> st2;
                if (st1 != "") D_Accounts[FindCNIC2(st1)]-
>scores.push_back(StringToFloat(st2));
            }
            if (st1 != "" && D_Accounts[FindCNIC2(st1)]->scores.size())
D_Accounts[FindCNIC2(st1)]->scores.pop_back();
        }
        LoadData.close();
        st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7 = "";
st8 = ""; st9 = ""; st10 = "";
        LoadData.open("Data/Vehicles.txt");
        if (LoadData.is_open()) {
            while (!LoadData.eof()) {
                st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 =
""; st7 = ""; st8 = ""; st9 = ""; st10 = "";
                LoadData >> st1 >> st2 >> st3 >> st4 >> st5 >> st6 >> st7;
                VclModel temo(st4, st5, StringToInt(st6));
                Vehicle temp(st1, st2, StringToInt(st3), temo, st7);
                if (st1 != "")
                    D_Accounts[FindCNIC2(st1)]-
>Vehicles.push_back(temp);
            }
            if (st1 != "" && D_Accounts[FindCNIC(st1)]->Vehicles.size())
D_Accounts[FindCNIC2(st1)]->Vehicles.pop_back();
        }
        LoadData.close();
        st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7 = "";
st8 = ""; st9 = ""; st10 = "";
        LoadData.open("Data/VRatings.txt");
        if (LoadData.is_open()) {
            while (!LoadData.eof()) {
                st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 =
""; st7 = ""; st8 = ""; st9 = ""; st10 = "";
                LoadData >> st1 >> st2 >> st3;
                if (st1 != "") D_Accounts[FindCNIC2(st1)]-
>Vehicles[FindID(st1, StringToInt(st2))]->scores.push_back(StringToFloat(st3));
            }
            if (st1 != "" && D_Accounts[FindCNIC2(st1)]-
>Vehicles[FindID(st1, StringToInt(st2))]->scores.size()) D_Accounts[FindCNIC2(st1)]-
>Vehicles[FindID(st1, StringToInt(st2))]->scores.pop_back();
        }
        LoadData.close();
    }
    st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7 = ""; st8 =
""; st9 = ""; st10 = "";
    LoadData.open("Data/Orders.txt");
    if (LoadData.is_open()) {
        while (!LoadData.eof()) {
            st1 = ""; st2 = ""; st3 = ""; st4 = ""; st5 = ""; st6 = ""; st7
= ""; st8 = ""; st9 = ""; st10 = "";
            LoadData >> st1 >> st2 >> st3 >> st4 >> st5 >> st6 >> st7 >> st8
>> st9 >> st10;
            Order temp(st1, st2, StringToInt(st3), st4, StringToInt(st5),
StringToInt(st6), StringToInt(st7), StringToInt(st8), StringToInt(st9),
StringToInt(st10));
            if (st1 != "") Orders.push_back(temp);
        }
        if (st1 != "" && Orders.size()) Orders.pop_back();
    }
    LoadData.close();
}

#pragma endregion
#pragma region Account Tools
void TMS::DeleteCAccount(int x) {
    for (int i = 0; i < Orders.size(); i++) {
        if (C_Accounts[x]->getID() == Orders[i]->getCID()) {
            CancelOrder(i);
            i = -1;
        }
    }
    for (int i = x; i < C_Accounts.size() - 1; i++) {
        *C_Accounts[i] = *C_Accounts[i + 1];
    }
    C_Accounts.pop_back();
}
void TMS::DeleteDAccount(int x) {
    for (int i = 0; i < Orders.size(); i++) {
        if (D_Accounts[x]->getID() == Orders[i]->getDID()) {
            RejectOrder(i);
            i = -1;
        }
    }
    for (int i = x; i < D_Accounts.size() - 1; i++) {
        *D_Accounts[i] = *D_Accounts[i + 1];
    }
    D_Accounts.pop_back();
}
void TMS::MakeOrder(Customer C, Driver D, Vehicle V, std::string type, int cost) {
    srand(time(0));
    int O_id = rand() % 1000000 + 1;
    while (!(isUniqueOID(O_id))) {
        int O_id = rand() % 1000000 + 1;
    }
    Order temp(C.getID(), D.getID(), V.getID(), type, O_id, time(0), 0, 0, 0,
cost);
    Orders.push_back(temp);
}
void TMS::CancelOrder(int x) {
    std::ofstream Savvy;
    Savvy.open("Data/All Orders.txt", std::ios::app);
    Orders[x]->Cancel();
    Savvy << Orders[x]->getType() + " Requested by: " <<
C_Accounts[FindCNIC(Orders[x]->getCID())]->getFName() + " " +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getLName() + "%Selected Driver: " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->getFName() + " " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->getLName() + "%Selected Vehicle: " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->Vehicles[FindID(Orders[x]->getDID(),
Orders[x]->getVID())]->Model.Company + " " + D_Accounts[FindCNIC2(Orders[x]-
>getDID())]->Vehicles[FindID(Orders[x]->getDID(), Orders[x]->getVID())]->Model.Model
+ " " + IntToString(D_Accounts[FindCNIC2(Orders[x]->getDID())]-
>Vehicles[FindID(Orders[x]->getDID(), Orders[x]->getVID())]->Model.Year) + "%Order
Cost: " + IntToString(Orders[x]->getCost()) + "%Order Cancelled\n";
    Savvy.close();
    Savvy.open("Data/Past_Services.txt", std::ios::app);
    Savvy.close();
    D_Accounts[FindCNIC2(Orders[x]->getDID())]->setFreedom(0);
    for (int i = x; i < Orders.size() - 1; i++) {
        *Orders[i] = *Orders[i + 1];
    }
    Orders.pop_back();
}
void TMS::AcceptOrder(int x) {
    for (int i = 0; i < Orders.size(); i++) {
        if (x == Orders[i]->getID()) {
            Orders[i]->Accept();
            D_Accounts[FindCNIC2(Orders[i]->getDID())]->setFreedom(1);
        }
    }
}
void TMS::RejectOrder(int x) {
    std::ofstream Savvy;
    Savvy.open("Data/All Orders.txt", std::ios::app);
    Orders[x]->Cancel();
    Savvy << Orders[x]->getType() + " Requested by: " <<
C_Accounts[FindCNIC(Orders[x]->getCID())]->getFName() + " " +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getLName() + "%Selected Driver: " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->getFName() + " " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->getLName() + "%Selected Vehicle: " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->Vehicles[FindID(Orders[x]->getDID(),
Orders[x]->getVID())]->Model.Company + " " + D_Accounts[FindCNIC2(Orders[x]-
>getDID())]->Vehicles[FindID(Orders[x]->getDID(), Orders[x]->getVID())]->Model.Model
+ " " + IntToString(D_Accounts[FindCNIC2(Orders[x]->getDID())]-
>Vehicles[FindID(Orders[x]->getDID(), Orders[x]->getVID())]->Model.Year) + "%Order
Cost: " + IntToString(Orders[x]->getCost()) + "%Order Rejected\n";
    Savvy.close();
    Savvy.open("Data/Past_Services.txt", std::ios::app);
    Savvy << Orders[x]->getType() + " " + Orders[x]->getDID() + " " +
IntToString(Orders[x]->getVID()) + " Requested*by:*" +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getFName() + "*" +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getLName() + "%Order*Status:*Rejected\n";
    Savvy.close();
    D_Accounts[FindCNIC2(Orders[x]->getDID())]->setFreedom(0);
    for (int i = x; i < Orders.size() - 1; i++) {
        *Orders[i] = *Orders[i + 1];
    }
    Orders.pop_back();
}
```

```
void TMS::CompleteOrder(int x) {
	std::ofstream Savvy;
	Savvy.open("Data/All Orders.txt", std::ios::app);
	Orders[x]->oComplete();
	Savvy << Orders[x]->getType() + " Requested by: " <<
C_Accounts[FindCNIC(Orders[x]->getCID())]->getFName() + " " +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getLName() + "%Selected Driver: " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->getFName() + " " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->getLName() + "%Selected Vehicle: " +
D_Accounts[FindCNIC2(Orders[x]->getDID())]->Vehicles[FindID(Orders[x]->getDID(),
Orders[x]->getVID())]->Model.Company + " " + D_Accounts[FindCNIC2(Orders[x]-
>getDID())]->Vehicles[FindID(Orders[x]->getDID(), Orders[x]->getVID())]->Model.Model
+ " " + IntToString(D_Accounts[FindCNIC2(Orders[x]->getDID())]-
>Vehicles[FindID(Orders[x]->getDID(), Orders[x]->getVID())]->Model.Year) + "%Order
Cost: " + IntToString(Orders[x]->getCost()) + "%Order Completed\n";
	Savvy.close();
	Savvy.open("Data/Past_Services.txt", std::ios::app);
	Savvy << Orders[x]->getType() + " " + Orders[x]->getDID() + " " +
IntToString(Orders[x]->getVID()) + " Requested*by:*" +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getFName() + "*" +
C_Accounts[FindCNIC(Orders[x]->getCID())]->getLName() +
"%Order*Status:*Completed\n";
	Savvy.close();
	for (int i = x; i < Orders.size() - 1; i++) {
		*Orders[i] = *Orders[i + 1];
	}
	Orders.pop_back();
}
void TMS::SortR() {
	int lindex = 0;
	for (int i = 0; i < this->D_Accounts.size(); i++) {
		lindex = i;
		for (int j = i; j < this->D_Accounts.size(); j++) {
			if (D_Accounts[j]->ComputeAndReturnRating() >
D_Accounts[lindex]->ComputeAndReturnRating()) {
				lindex = j;
			}
		}
		swap(D_Accounts[i], D_Accounts[lindex]);
	}
}
void TMS::SortE() {
	int lindex = 0;;
	for (int i = 0; i < this->D_Accounts.size(); i++) {
		lindex = i;
		for (int j = i; j < this->D_Accounts.size(); j++) {
			if (D_Accounts[j]->getExp() > D_Accounts[lindex]->getExp()) {
				lindex = j;
			}
		}
		swap(D_Accounts[i], D_Accounts[lindex]);
	}
}
#pragma endregion
#pragma region Tools for Finding
bool TMS::isUniqueOID(int x) {
	for (int i = 0; i < Orders.size(); i++) {
		if (x == Orders[i]->getID()) {
```

```
            return 0;
        }
    }
    return 1;
}
int TMS::FindCNIC(std::string s) {
    for (int i = 0; i < C_Accounts.size(); i++) {
        if (s == C_Accounts[i]->getID()) {
            return i;
        }
    }
    return -1;
}
int TMS::FindCNIC2(std::string s) {
    for (int i = 0; i < D_Accounts.size(); i++) {
        if (s == D_Accounts[i]->getID()) {
            return i;
        }
    }
    return -1;
}
int TMS::FindID(std::string str, int x) {
    for (int i = 0; i < D_Accounts[FindCNIC2(str)]->Vehicles.size(); i++) {
        if (x == D_Accounts[FindCNIC2(str)]->Vehicles[i]->vehicleID) {
            return i;
        }
    }
}
bool TMS::isUniqueCNIC(std::string s) {
    for (int i = 0; i < C_Accounts.size(); i++) {
        if (C_Accounts[i]->getID() == s) {
            return 0;
        }
    }
    for (int i = 0; i < D_Accounts.size(); i++) {
        if (D_Accounts[i]->getID() == s) {
            return 0;
        }
    }
    return 1;
}
bool TMS::idUniqueLisence(int x) {
    for (int i = 0; i < D_Accounts.size(); i++) {
        for (int j = 0; j < D_Accounts[i]->Vehicles.size(); j++) {
            if (x == D_Accounts[i]->Vehicles[j]->getID()) {
                return 0;
            }
        }
    }
    return 1;
}
#pragma endregion
#pragma endregion
#pragma region Functions

bool isNum(char c) {
    if (c < '0' || c > '9') {
        return 0;
    }
    return 1;
}
int StringToInt(std::string str) {
    int x = 0, i;
    for (i = 0; i < str.length(); i++) {
        if (isNum(str[i])) {
            x = (x * 10) + (str[i] - '0');
        }
    }
    return x;
}
float StringToFloat(std::string str) {
    float x = 0, y = 0;
    int i;
    for (i = 0; i < str.length() && str[i] != '.'; i++) {
        if (isNum(str[i])) {
            x = (x * 10) + (str[i] - '0');
        }
    }
    for (i++; i < str.length(); i++) {
        if (isNum(str[i])) {
            y = (y * 10) + (str[i] - '0');
        }
    }
    while (y > 1) {
        y /= 10;
    }
    return x + y;
}
bool isValidCNIC(std::string s) {
    bool validity = 1;
    if (s.length() != 15) {
        return 0;
    }
    for (int i = 0; i < 15; i++) {
        if (i == 5 || i == 13) {
            if (s[i] == '-') {
                validity = 1;
            }
            else
                validity = 0;
        }
        else {
            if (!isNum(s[i])) {
                validity = 0;
            }
        }
    }
    return validity;
}
bool isValidName(std::string s) {
    for (int i = 0; i < s.length(); i++) {
        if ((s[i] >= 'A' && s[i] <= 'Z') || (s[i] >= 'a' && s[i] <= 'z')) {
            //do nothing
        }
        else {
            return 0;
```

```
		}
	}
	return 1;
}
bool isNull(std::string s) {
	if (s == "") {
		return 1;
	}
	for (int i = 0; i < s.length(); i++) {
		if (s[i] != ' ') {
			return 0;
		}
	}
	return 1;
}
bool isInt(std::string s) {
	for (int i = 0; i < s.size(); i++) {
		if (!(isNum(s[i]))) {
			return 0;
		}
	}
	return 1;
}
std::string IntToString(int x) {
	char c;
	std::string str;
	if (x == 0) {
		return "0";
	}
	while (x != 0) {
		c = '0' + x % 10;
		x /= 10;
		str = str + c;
	}
	for (int i = 0; i < str.size() / 2; i++) {
		c = str[i];
		str[i] = str[str.size() - 1 - i];
		str[str.size() - 1 - i] = c;
	}
	return str;
}
std::string FloatToString(float x) {
	int p, q;
	float t = x;
	p = x;
	t = t - p;
	q = t;
	if (t != 0) {
		while (t < 10) {
			t *= 10;
			q = t;
		}
	}
	std::string str = IntToString(p) + "." + IntToString(q);
	return str;
}
bool ContainsSpaces(std::string s) {
	for (int i = 0; i < s.length(); i++) {
```

```
                if (s[i] == ' ') return 1;
        }
        return 0;
}
template <typename T>
void swap(T* v1, T* v2) {
        T k;
        k = *v1;
        *v1 = *v2;
        *v2 = k;
}
#pragma endregion
```

Custom Vector Class (Viktor.h)

```
#pragma once
template <typename Data> //Self Made Vector Class
class Viktor {
	Data* vec; //pointer for DMA
	int length = 0; //the length/size of the Viktor
public:
	Viktor();
	Viktor(int length);
	void push_back(Data x); //Just like it is in vectors
	void pop_back(); //classic vector function, stole for my purposes
	void push_front(Data x); //i got carried away and implemented pop_front and
push_front
	void pop_front(); //but ended up not using them at all
	int size(); //returns the size
	Data* operator [](int n); //overloaded subscript operator to easily access
the indices, returns the address instead of value
	void clear();
};
```

Custom Vector Implementation (Viktor.cpp)

```
#include "pch.h"
#include "Viktor.h"
template <typename Data>
Viktor<Data>::Viktor() {
	this->length = 0;
	this->vec = nullptr;
}
template <typename Data>
Viktor<Data>::Viktor(int length) {
	if (this->vec != nullptr) delete[] this->vec;
	this->length = length;
	this->vec = new Data[length];
}
template <typename Data>
void Viktor<Data>::push_back(Data x) {
	int i;
	Data* ptr = new Data[this->length + 1];
	for (i = 0; i < this->length; i++) {
		ptr[i] = this->vec[i];
	}
	ptr[this->length] = x;
	delete[] this->vec;
	this->vec = ptr;
	this->length += 1;
}
template <typename Data>
void Viktor<Data>::push_front(Data x) {
	int i;
	Data* ptr = new Data[this->length + 1];
	ptr[0] = x;
	for (i = 1; i <= this->length; i++) {
		ptr[i] = this->vec[i - 1];
	}
	delete[] this->vec;
	this->vec = ptr;
	this->length += 1;
}
template <typename Data>
void Viktor<Data>::pop_back() {
	if (this->length > 0) {
		int i;
		Data* ptr = new Data[this->length - 1];
		for (i = 0; i < this->length - 1; i++) {
			ptr[i] = this->vec[i];
		}
		delete[] this->vec;
		this->vec = ptr;
		this->length -= 1;
	}
}
template <typename Data>
void Viktor<Data>::pop_front() {
	if (this->length > 0) {
		int i;
```

```
        Data* ptr = new Data[this->length - 1];
        for (i = 0; i < this->length - 1; i++) {
            ptr[i] = this->vec[i + 1];
        }
        delete[] this->vec;
        this->vec = ptr;
        this->length -= 1;
    }
}
template <typename Data>
int Viktor<Data>::size() {
    return this->length;
}
template <typename Data>
Data* Viktor<Data>::operator [](int n) {
    if (n >= 0 && n < this->length) {
        return &this->vec[n];
    }
}
template <typename Data>
void Viktor<Data>::clear() {
    delete[] this->vec;
    this->vec = NULL;
    this->length = 0;
}
```

Login Page (Login.h)

```
#pragma once
#include "SignUp.h"
#include "Customer_Form.h"
#include "Driver_Form.h"
#include "Admin_Form.h"
extern TMS TMS_Main;
using namespace std;
using namespace System;
using namespace System::Configuration;
namespace TMS_Forms {
    using namespace std;
    using namespace System;
    using namespace System::Windows::Forms;
    using namespace System::Collections::Generic;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
#pragma region Login Form
    public ref class Login_Form : public System::Windows::Forms::Form {
    public:
        Login_Form(void) {
            InitializeComponent();
        }

    protected:
        ~Login_Form() {
            if (components) {
                delete components;
            }
        }
    private: System::Windows::Forms::RichTextBox^ richTextBox1;
    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::TextBox^ textBox1;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::TextBox^ textBox2;
    private: System::Windows::Forms::Button^ button2;
    private: System::ComponentModel::Container^ components;
#pragma region Component Code

          void InitializeComponent(void) {
              this->label1 = (gcnew System::Windows::Forms::Label());
              this->textBox1 = (gcnew System::Windows::Forms::TextBox());
              this->button1 = (gcnew System::Windows::Forms::Button());
              this->label2 = (gcnew System::Windows::Forms::Label());
              this->label3 = (gcnew System::Windows::Forms::Label());
              this->textBox2 = (gcnew System::Windows::Forms::TextBox());
              this->button2 = (gcnew System::Windows::Forms::Button());
              this->SuspendLayout();
              //
              // label1
```

```
                      //
                      this->label1->AutoSize = true;
                      this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label1->Location = System::Drawing::Point(58, 14);
                      this->label1->Name = L"label1";
                      this->label1->Size = System::Drawing::Size(91, 19);
                      this->label1->TabIndex = 0;
                      this->label1->Text = L"Enter CNIC";
                      this->label1->Click += gcnew System::EventHandler(this,
&Login_Form::label1_Click);
                      //
                      // textBox1
                      //
                      this->textBox1->Location = System::Drawing::Point(61, 31);
                      this->textBox1->Name = L"textBox1";
                      this->textBox1->Size = System::Drawing::Size(162, 20);
                      this->textBox1->TabIndex = 1;
                      //
                      // button1
                      //
                      this->button1->BackColor = System::Drawing::Color::Silver;
                      this->button1->Cursor =
System::Windows::Forms::Cursors::Hand;
                      this->button1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button1->Location = System::Drawing::Point(12, 139);
                      this->button1->Name = L"button1";
                      this->button1->Size = System::Drawing::Size(75, 23);
                      this->button1->TabIndex = 2;
                      this->button1->Text = L"Proceed";
                      this->button1->UseVisualStyleBackColor = false;
                      this->button1->Click += gcnew System::EventHandler(this,
&Login_Form::button1_Click);
                      //
                      // label2
                      //
                      this->label2->AutoSize = true;
                      this->label2->Font = (gcnew System::Drawing::Font(L"Book
Antiqua", 9, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label2->Location = System::Drawing::Point(34, 54);
                      this->label2->Name = L"label2";
                      this->label2->Size = System::Drawing::Size(222, 16);
                      this->label2->TabIndex = 3;
                      this->label2->Text = L"(Please use the Format 12345-1234567-
9)";
                      this->label2->Click += gcnew System::EventHandler(this,
&Login_Form::label2_Click);
                      //
                      // label3
                      //
                      this->label3->AutoSize = true;
                      this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label3->Location = System::Drawing::Point(57, 80);
                      this->label3->Name = L"label3";
                      this->label3->Size = System::Drawing::Size(125, 19);
                      this->label3->TabIndex = 4;
                      this->label3->Text = L"Enter Password";
                      this->label3->Click += gcnew System::EventHandler(this,
&Login_Form::label3_Click);
                      //
                      // textBox2
                      //
                      this->textBox2->Location = System::Drawing::Point(61, 99);
                      this->textBox2->Name = L"textBox2";
                      this->textBox2->PasswordChar = '*';
                      this->textBox2->Size = System::Drawing::Size(162, 20);
                      this->textBox2->TabIndex = 5;
                      this->textBox2->TextChanged += gcnew
System::EventHandler(this, &Login_Form::textBox2_TextChanged);
                      //
                      // button2
                      //
                      this->button2->BackColor = System::Drawing::Color::Silver;
                      this->button2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button2->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button2->Location = System::Drawing::Point(169, 139);
                      this->button2->Name = L"button2";
                      this->button2->Size = System::Drawing::Size(103, 23);
                      this->button2->TabIndex = 6;
                      this->button2->Text = L"Create Account";
                      this->button2->UseVisualStyleBackColor = false;
                      this->button2->Click += gcnew System::EventHandler(this,
&Login_Form::button2_Click);
                      //
                      // Login_Form
                      //
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(284, 174);
                      this->Controls->Add(this->button2);
                      this->Controls->Add(this->textBox2);
                      this->Controls->Add(this->label3);
                      this->Controls->Add(this->label2);
                      this->Controls->Add(this->button1);
                      this->Controls->Add(this->textBox1);
                      this->Controls->Add(this->label1);
                      this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::Fixed3D;
                      this->Name = L"Login_Form";
```

```
              this->Text = L"Transport Manager+";
              this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&Login_Form::Login_Form_FormClosing);
              this->Load += gcnew System::EventHandler(this,
&Login_Form::Login_Form_Load);
              this->ResumeLayout(false);
              this->PerformLayout();

          }
#pragma endregion
#pragma region Function Code
    private: System::Void Login_Form_Load(System::Object^ sender,
System::EventArgs^ e) {
        FloatToString(421.123456);
        TMS_Main.ClearLoadedData();
        TMS_Main.LoadSavedData();
    }
    private: System::Void label1_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
        System::String^ s = Convert::ToString(textBox1->Text);
        System::String^ s2 = Convert::ToString(textBox2->Text);
        msclr::interop::marshal_context context;
        std::string cnic = context.marshal_as<std::string>(s);
        std::string pass = context.marshal_as<std::string>(s2);
        if (cnic == "admin lemme in" && pass == "i have the power") {
            Admin_Form^ f2 = gcnew Admin_Form();
            f2->Show();
            this->Hide();
        }
        else {
            if (isValidCNIC(cnic)) {
                if (!(TMS_Main.FindCNIC(cnic) + 1)) {
                    if (!(TMS_Main.FindCNIC2(cnic) + 1)) {
                        MessageBox::Show("Entered CNIC or Password is
Incorrect, No account exists against given credentials Please create an account
first", "Error: Incorrect Credentials", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
                    }
                    else {
                        if (pass ==
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(cnic)]->getPass()) {
                            std::ofstream CurrentAcc;

    CurrentAcc.open("Data/CurrentAcc.txt");
                            CurrentAcc << "Driver " <<
TMS_Main.FindCNIC2(cnic);

                            CurrentAcc.close();
                            Driver_Form^ f2 = gcnew Driver_Form();
                            f2->Show();
                            this->Hide();
                        }
                        else {
                            MessageBox::Show("Entered CNIC or
Password is Incorrect", "Error: Incorrect Credentials", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
                        }
                    }
                }
                else {
                    if (pass ==
TMS_Main.C_Accounts[TMS_Main.FindCNIC(cnic)]->getPass()) {
                        std::ofstream CurrentAcc;
                        CurrentAcc.open("Data/CurrentAcc.txt");
                        CurrentAcc << "Customer " <<
TMS_Main.FindCNIC(cnic);
                        CurrentAcc.close();
                        Customer_Form^ f2 = gcnew Customer_Form();
                        f2->Show();
                        this->Hide();
                    }
                    else {
                        MessageBox::Show("Entered CNIC or Password is
Incorrect", "Error: Incorrect Credentials", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
                    }
                }
            }
            else {
                MessageBox::Show("Please use the proper Format (12345-
1234567-9)", "Error: Invalid CNIC", MessageBoxButtons::OK, MessageBoxIcon::Stop);
            }
        }
    }
    private: System::Void textBox2_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void label3_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void label2_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e) {
        this->Hide();
        TMS_Main.SaveLoadedData();
        TMS_Main.ClearLoadedData();
        Signup_Form^ f2 = gcnew Signup_Form();
        f2->Show();
    }
    private: System::Void Login_Form_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {
        TMS_Main.SaveLoadedData();
        Application::Exit();
    }
#pragma endregion
    };
#pragma endregion
}
```

Sign Up Page (SignUp.h)

```
#pragma once
#include "LogIn.h"
extern TMS TMS_Main;
using namespace std;
using namespace System;
using namespace System::Configuration;
namespace TMS_Forms {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
#pragma region Sign Up Form
    public ref class Signup_Form : public System::Windows::Forms::Form {
    public:
        Signup_Form(void) {
            InitializeComponent();
        }
    protected:
        ~Signup_Form() {
            if (components) {
                delete components;
            }
        }
    private: System::Windows::Forms::TextBox^ textBox1;
    private: System::Windows::Forms::TextBox^ textBox2;
    private: System::Windows::Forms::TextBox^ textBox3;
    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Label^ label5;
    private: System::Windows::Forms::TextBox^ textBox4;
    private: System::Windows::Forms::TextBox^ textBox5;
    private: System::Windows::Forms::Label^ label6;
    private: System::Windows::Forms::Label^ label7;
    private: System::Windows::Forms::Label^ label8;
    private: System::Windows::Forms::Label^ label9;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::RadioButton^ radioButton1;
    private: System::Windows::Forms::RadioButton^ radioButton2;
    private: System::Windows::Forms::TextBox^ textBox6;
    private: System::Windows::Forms::Label^ label10;
    private: System::Windows::Forms::Label^ label11;
    private: System::Windows::Forms::Label^ label12;
    private: System::ComponentModel::Container^ components;

#pragma region Component Code
           void InitializeComponent(void) {
               this->textBox1 = (gcnew System::Windows::Forms::TextBox());
               this->textBox2 = (gcnew System::Windows::Forms::TextBox());
               this->textBox3 = (gcnew System::Windows::Forms::TextBox());
               this->label1 = (gcnew System::Windows::Forms::Label());
```

```
                      this->label2 = (gcnew System::Windows::Forms::Label());
                      this->label3 = (gcnew System::Windows::Forms::Label());
                      this->label4 = (gcnew System::Windows::Forms::Label());
                      this->label5 = (gcnew System::Windows::Forms::Label());
                      this->textBox4 = (gcnew System::Windows::Forms::TextBox());
                      this->textBox5 = (gcnew System::Windows::Forms::TextBox());
                      this->label6 = (gcnew System::Windows::Forms::Label());
                      this->label7 = (gcnew System::Windows::Forms::Label());
                      this->label8 = (gcnew System::Windows::Forms::Label());
                      this->label9 = (gcnew System::Windows::Forms::Label());
                      this->button1 = (gcnew System::Windows::Forms::Button());
                      this->radioButton1 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton2 = (gcnew
System::Windows::Forms::RadioButton());
                      this->textBox6 = (gcnew System::Windows::Forms::TextBox());
                      this->label10 = (gcnew System::Windows::Forms::Label());
                      this->label11 = (gcnew System::Windows::Forms::Label());
                      this->label12 = (gcnew System::Windows::Forms::Label());
                      this->SuspendLayout();
                      //
                      // textBox1
                      //
                      this->textBox1->Location = System::Drawing::Point(39, 35);
                      this->textBox1->Name = L"textBox1";
                      this->textBox1->Size = System::Drawing::Size(217, 20);
                      this->textBox1->TabIndex = 0;
                      this->textBox1->TextChanged += gcnew
System::EventHandler(this, &Signup_Form::textBox1_TextChanged);
                      //
                      // textBox2
                      //
                      this->textBox2->Location = System::Drawing::Point(39, 88);
                      this->textBox2->Name = L"textBox2";
                      this->textBox2->Size = System::Drawing::Size(217, 20);
                      this->textBox2->TabIndex = 1;
                      this->textBox2->TextChanged += gcnew
System::EventHandler(this, &Signup_Form::textBox2_TextChanged);
                      //
                      // textBox3
                      //
                      this->textBox3->Location = System::Drawing::Point(39, 144);
                      this->textBox3->Name = L"textBox3";
                      this->textBox3->Size = System::Drawing::Size(217, 20);
                      this->textBox3->TabIndex = 2;
                      this->textBox3->TextChanged += gcnew
System::EventHandler(this, &Signup_Form::textBox3_TextChanged);
                      //
                      // label1
                      //
                      this->label1->AutoSize = true;
                      this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label1->Location = System::Drawing::Point(35, 19);
                      this->label1->Name = L"label1";
                      this->label1->Size = System::Drawing::Size(90, 19);
                      this->label1->TabIndex = 3;
                      this->label1->Text = L"First Name";
                      //
                      // label2
                      //
                      this->label2->AutoSize = true;
                      this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label2->Location = System::Drawing::Point(35, 72);
                      this->label2->Name = L"label2";
                      this->label2->Size = System::Drawing::Size(89, 19);
                      this->label2->TabIndex = 4;
                      this->label2->Text = L"Last Name";
                      this->label2->Click += gcnew System::EventHandler(this,
&Signup_Form::label2_Click);
                      //
                      // label3
                      //
                      this->label3->AutoSize = true;
                      this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label3->Location = System::Drawing::Point(35, 128);
                      this->label3->Name = L"label3";
                      this->label3->Size = System::Drawing::Size(91, 19);
                      this->label3->TabIndex = 5;
                      this->label3->Text = L"Enter CNIC";
                      //
                      // label4
                      //
                      this->label4->AutoSize = true;
                      this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label4->Location = System::Drawing::Point(35, 195);
                      this->label4->Name = L"label4";
                      this->label4->Size = System::Drawing::Size(125, 19);
                      this->label4->TabIndex = 6;
                      this->label4->Text = L"Enter Password";
                      //
                      // label5
                      //
                      this->label5->AutoSize = true;
                      this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft
YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label5->Location = System::Drawing::Point(35, 250);
                      this->label5->Name = L"label5";
                      this->label5->Size = System::Drawing::Size(146, 19);
                      this->label5->TabIndex = 7;
                      this->label5->Text = L"Confirm Password";
                      //
                      // textBox4
                      //
                      this->textBox4->Location = System::Drawing::Point(39, 211);
                      this->textBox4->Name = L"textBox4";
                      this->textBox4->PasswordChar = '*';
                      this->textBox4->Size = System::Drawing::Size(217, 20);
                      this->textBox4->TabIndex = 8;
                      this->textBox4->TextChanged += gcnew
System::EventHandler(this, &Signup_Form::textBox4_TextChanged);
                      //
                      // textBox5
                      //
                      this->textBox5->Location = System::Drawing::Point(39, 266);
                      this->textBox5->Name = L"textBox5";
                      this->textBox5->PasswordChar = '*';
                      this->textBox5->Size = System::Drawing::Size(217, 20);
                      this->textBox5->TabIndex = 9;
                      this->textBox5->TextChanged += gcnew
System::EventHandler(this, &Signup_Form::textBox5_TextChanged);
                      //
                      // label6
                      //
                      this->label6->AutoSize = true;
                      this->label6->Location = System::Drawing::Point(36, 55);
                      this->label6->Name = L"label6";
                      this->label6->Size = System::Drawing::Size(0, 13);
                      this->label6->TabIndex = 10;
                      //
                      // label7
                      //
                      this->label7->AutoSize = true;
                      this->label7->Location = System::Drawing::Point(36, 108);
                      this->label7->Name = L"label7";
                      this->label7->Size = System::Drawing::Size(0, 13);
                      this->label7->TabIndex = 11;
                      //
                      // label8
                      //
                      this->label8->AutoSize = true;
                      this->label8->Location = System::Drawing::Point(36, 167);
                      this->label8->Name = L"label8";
                      this->label8->Size = System::Drawing::Size(0, 13);
                      this->label8->TabIndex = 12;
                      this->label8->Click += gcnew System::EventHandler(this,
&Signup_Form::label8_Click);
                      //
                      // label9
                      //
                      this->label9->AutoSize = true;
                      this->label9->Location = System::Drawing::Point(36, 286);
                      this->label9->Name = L"label9";
                      this->label9->Size = System::Drawing::Size(0, 13);
                      this->label9->TabIndex = 13;
                      this->label9->Click += gcnew System::EventHandler(this,
&Signup_Form::label9_Click);
                      //
                      // button1
                      //
                      this->button1->BackColor = System::Drawing::Color::Silver;
                      this->button1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button1->Location = System::Drawing::Point(12, 308);
                      this->button1->Name = L"button1";
                      this->button1->Size = System::Drawing::Size(114, 23);
                      this->button1->TabIndex = 14;
                      this->button1->Text = L"Create Account";
                      this->button1->UseVisualStyleBackColor = false;
                      this->button1->Click += gcnew System::EventHandler(this,
&Signup_Form::button1_Click);
                      //
                      // radioButton1
                      //
                      this->radioButton1->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton1->AutoSize = true;
                      this->radioButton1->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte
>(224)), static_cast<System::Int32>(static_cast<System::Byte>(224)),

static_cast<System::Int32>(static_cast<System::Byte>(224)));
                      this->radioButton1->Cursor =
System::Windows::Forms::Cursors::Hand;
                      this->radioButton1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->radioButton1->Location = System::Drawing::Point(343,
20);
                      this->radioButton1->Name = L"radioButton1";
                      this->radioButton1->Size = System::Drawing::Size(120, 49);
                      this->radioButton1->TabIndex = 15;
                      this->radioButton1->Text = L"\nCustomer Account\n ";
                      this->radioButton1->UseVisualStyleBackColor = false;
                      this->radioButton1->CheckedChanged += gcnew
System::EventHandler(this, &Signup_Form::radioButton1_CheckedChanged);
                      //
                      // radioButton2
                      //
                      this->radioButton2->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton2->AutoSize = true;
                      this->radioButton2->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte
>(224)), static_cast<System::Int32>(static_cast<System::Byte>(224)),

static_cast<System::Int32>(static_cast<System::Byte>(224)));
                      this->radioButton2->Cursor =
System::Windows::Forms::Cursors::Hand;
                      this->radioButton2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
```

```
                      this->radioButton2->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->radioButton2->Location = System::Drawing::Point(469,
19);
                      this->radioButton2->Name = L"radioButton2";
                      this->radioButton2->Size = System::Drawing::Size(122, 49);
                      this->radioButton2->TabIndex = 16;
                      this->radioButton2->Text = L"\n   Driver Account  \n ";
                      this->radioButton2->UseVisualStyleBackColor = false;
                      this->radioButton2->CheckedChanged += gcnew
System::EventHandler(this, &Signup_Form::radioButton2_CheckedChanged);
                      // 
                      // textBox6
                      // 
                      this->textBox6->Location = System::Drawing::Point(343, 150);
                      this->textBox6->Name = L"textBox6";
                      this->textBox6->Size = System::Drawing::Size(169, 20);
                      this->textBox6->TabIndex = 18;
                      this->textBox6->TextChanged += gcnew
System::EventHandler(this, &Signup_Form::textBox6_TextChanged);
                      // 
                      // label10
                      // 
                      this->label10->AutoSize = true;
                      this->label10->Font = (gcnew
System::Drawing::Font(L"Microsoft YaHei", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label10->Location = System::Drawing::Point(339, 109);
                      this->label10->Name = L"label10";
                      this->label10->Size = System::Drawing::Size(235, 38);
                      this->label10->TabIndex = 19;
                      this->label10->Text = L"How much Years of\nWork Experience do
you have";
                      this->label10->Click += gcnew System::EventHandler(this,
&Signup_Form::label10_Click);
                      // 
                      // label11
                      // 
                      this->label11->AutoSize = true;
                      this->label11->Location = System::Drawing::Point(340, 173);
                      this->label11->Name = L"label11";
                      this->label11->Size = System::Drawing::Size(0, 13);
                      this->label11->TabIndex = 20;
                      this->label11->Click += gcnew System::EventHandler(this,
&Signup_Form::label11_Click);
                      // 
                      // label12
                      // 
                      this->label12->AutoSize = true;
                      this->label12->Location = System::Drawing::Point(39, 231);
                      this->label12->Name = L"label12";
                      this->label12->Size = System::Drawing::Size(0, 13);
                      this->label12->TabIndex = 21;
                      // 
                      // Signup_Form
                      //
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(603, 343);
                      this->Controls->Add(this->label12);
                      this->Controls->Add(this->label11);
                      this->Controls->Add(this->label10);
                      this->Controls->Add(this->textBox6);
                      this->Controls->Add(this->radioButton2);
                      this->Controls->Add(this->radioButton1);
                      this->Controls->Add(this->button1);
                      this->Controls->Add(this->label9);
                      this->Controls->Add(this->label8);
                      this->Controls->Add(this->label7);
                      this->Controls->Add(this->label6);
                      this->Controls->Add(this->textBox5);
                      this->Controls->Add(this->textBox4);
                      this->Controls->Add(this->label5);
                      this->Controls->Add(this->label4);
                      this->Controls->Add(this->textBox3);
                      this->Controls->Add(this->textBox2);
                      this->Controls->Add(this->textBox1);
                      this->Controls->Add(this->label3);
                      this->Controls->Add(this->label2);
                      this->Controls->Add(this->label1);
                      this->Name = L"Signup_Form";
                      this->Text = L"Transport Manager+";
                      this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&Signup_Form::Signup_Form_FormClosing);
                      this->Load += gcnew System::EventHandler(this,
&Signup_Form::Signup_Form_Load);
                      this->ResumeLayout(false);
                      this->PerformLayout();

                }
#pragma endregion
#pragma region Function Code
      private: System::Void Signup_Form_Load(System::Object^ sender,
System::EventArgs^ e) {
             TMS_Main.LoadSavedData();
             this->radioButton1->Checked = true;
      }
      private: System::Void label2_Click(System::Object^ sender, System::EventArgs^
e) {
      }
      private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
             System::String^ s = Convert::ToString(textBox1->Text);
             msclr::interop::marshal_context context;
             std::string name = context.marshal_as<std::string>(s);
             if (!isValidName(name)) {
                   this->label6->Text = ("Please enter a valid Name, Do not use
Numbers or symbols");
             }
             else {
                   this->label6->Text = ("");
             }
             if (ContainsSpaces(name)) {
                   this->label6->Text = ("Please do not use spaces");
             }
             else {
                   this->label6->Text = ("");
             }
      }
      private: System::Void textBox2_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
             System::String^ s = Convert::ToString(textBox2->Text);
             msclr::interop::marshal_context context;
             std::string name = context.marshal_as<std::string>(s);
             if (!isValidName(name)) {
                   this->label7->Text = ("Please enter a valid Name, Do not use
Numbers or symbols");
             }
             else {
                   this->label7->Text = ("");
             }
             if (ContainsSpaces(name)) {
                   this->label7->Text = ("Please do not use spaces");
             }
             else {
                   this->label7->Text = ("");
             }
      }
      private: System::Void textBox3_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
             System::String^ s = Convert::ToString(textBox3->Text);
             msclr::interop::marshal_context context;
             std::string cnic = context.marshal_as<std::string>(s);
             if (!isValidCNIC(cnic)) {
                   this->label8->Text = ("Please use the Proper Format for CNIC");
             }
             else {
                   this->label8->Text = ("");
             }
      }
      private: System::Void textBox4_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
             System::String^ s1 = Convert::ToString(textBox4->Text);
             msclr::interop::marshal_context context;
             std::string pass = context.marshal_as<std::string>(s1);
             if (ContainsSpaces(pass)) {
                   this->label12->Text = ("Please do not use spaces");
             }
             else {
                   this->label12->Text = ("");
             }
      }
      private: System::Void textBox5_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
             System::String^ s1 = Convert::ToString(textBox4->Text);
             msclr::interop::marshal_context context;
             System::String^ s2 = Convert::ToString(textBox5->Text);
             std::string pass2 = context.marshal_as<std::string>(s1);
             std::string pass1 = context.marshal_as<std::string>(s2);
```

```
            if (s1 != s2) {
                  this->label9->Text = ("Passwords do not match");
            }
            else {
                  this->label9->Text = ("");
            }
      }
      private: System::Void label9_Click(System::Object^ sender, System::EventArgs^
e) {
      }
      private: System::Void label8_Click(System::Object^ sender, System::EventArgs^
e) {
      }
      private: System::Void radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
            if (this->radioButton1->Checked) {
                  this->radioButton2->Checked = false;
            }
            if (this->radioButton1->Checked) {
                  this->radioButton2->Location = System::Drawing::Point(268, 237);
                  this->radioButton1->Location = System::Drawing::Point(270, 182);
                  this->ClientSize = System::Drawing::Size(402, 343);
                  this->label10->Hide();
                  this->textBox6->Hide();
                  this->label11->Hide();
            }
            else {
                  this->radioButton2->Location = System::Drawing::Point(469, 19);
                  this->radioButton1->Location = System::Drawing::Point(343, 20);
                  this->ClientSize = System::Drawing::Size(603, 343);
                  this->label10->Show();
                  this->textBox6->Show();
                  this->label11->Show();
            }
      }
      private: System::Void radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
            if (this->radioButton2->Checked) {
                  this->radioButton1->Checked = false;
            }
            if (this->radioButton2->Checked) {
                  this->radioButton2->Location = System::Drawing::Point(469, 19);
                  this->radioButton1->Location = System::Drawing::Point(343, 20);
                  this->ClientSize = System::Drawing::Size(603, 343);
                  this->label10->Show();
                  this->textBox6->Show();
                  this->label11->Show();
            }
            else {
                  this->radioButton2->Location = System::Drawing::Point(268, 237);
                  this->radioButton1->Location = System::Drawing::Point(270, 182);
                  this->ClientSize = System::Drawing::Size(402, 343);
                  this->label10->Hide();
                  this->textBox6->Hide();
                  this->label11->Hide();
            }
      }
      private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
            msclr::interop::marshal_context context;
            System::String^ s1;
            std::string std1, std2, std3, std4, std5, std6;

            s1 = Convert::ToString(textBox1->Text);
            std1 = context.marshal_as<std::string>(s1);

            s1 = Convert::ToString(textBox2->Text);
            std2 = context.marshal_as<std::string>(s1);

            s1 = Convert::ToString(textBox3->Text);
            std3 = context.marshal_as<std::string>(s1);

            s1 = Convert::ToString(textBox4->Text);
            std4 = context.marshal_as<std::string>(s1);

            s1 = Convert::ToString(textBox5->Text);
            std5 = context.marshal_as<std::string>(s1);

            s1 = Convert::ToString(textBox6->Text);
            std6 = context.marshal_as<std::string>(s1);

            if (ContainsSpaces(std1)) {
                  MessageBox::Show("Please do not use any spaces in the First Name
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                  return;
            }
            if (ContainsSpaces(std2)) {
                  MessageBox::Show("Please do not use any spaces in the Last Name
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                  return;
            }
            if (ContainsSpaces(std3)) {
                  MessageBox::Show("Please do not use any spaces in the CNIC
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                  return;
            }
            if (ContainsSpaces(std4)) {
                  MessageBox::Show("Please do not use any spaces in the Password
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                  return;
            }
            if (isNull(std1)) {
                  MessageBox::Show("Missing Information in First Name Field",
"Error: Missing Credentials", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                  return;
            }
            else {
                  if (!(isValidName(std1))) {
                        MessageBox::Show("Information Entered in the First Name
Field is Invalid", "Error: Invalid Credentials", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
                        return;
                  }
            }
            if (isNull(std2)) {
```

```
            MessageBox::Show("Missing Information in Last Name Field", 
"Error: Missing Credentials", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
        else {
            if (!(isValidName(std2))) {
                MessageBox::Show("Information Entered in the Last Name 
Field is Invalid", "Error: Invalid Credentials", MessageBoxButtons::OK, 
MessageBoxIcon::Hand);
                return;
            }
        }
        if (isNull(std3)) {
            MessageBox::Show("Missing Information in CNIC Field", "Error: 
Missing Credentials", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
        else {
            if (!(isValidCNIC(std3))) {
                MessageBox::Show("Information Entered in the CNIC Field is 
Invalid", "Error: Invalid Credentials", MessageBoxButtons::OK, 
MessageBoxIcon::Hand);
                return;
            }
            else {
                if (!(TMS_Main.isUniqueCNIC(std3))) {
                    MessageBox::Show("Entered CNIC is already in use by 
another account", "Error: Repeated Credentials", MessageBoxButtons::OK, 
MessageBoxIcon::Hand);
                    return;
                }
            }
        }
        if (isNull(std4)) {
            MessageBox::Show("Please choose a password for your account", 
"Error: Missing Credentials", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
        if (isNull(std5)) {
            MessageBox::Show("Please confirm your password", "Error: Missing 
Credentials", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
        for (int i = 0; i < std4.size(); i++) {
            if (std4[i] == ' ') {
                MessageBox::Show("You cannot use spaces in the 
password\nPlease re-enter the password without spaces ( use - or _ instead )", 
"Error: Invalid Credentials", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                return;
            }
        }
        if (std4 != std5) {
            MessageBox::Show("Entered Password and Confirmed Password does 
not match", "Error: Incorrect Credentials", MessageBoxButtons::OK, 
MessageBoxIcon::Hand);
            return;
        }
        if (this->radioButton2->Checked) {
            if (isNull(std6)) {
                MessageBox::Show("Missing Information in Work Experience
Field", "Error: Missing Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
                return;
            }
            else {
                if (!(isInt(std6))) {
                    MessageBox::Show("Information Entered in the Work
Experience Field is Invalid", "Error: Invalid Information", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
                    return;
                }
            }
        }
        if (this->radioButton2->Checked) {
            Driver temp(std1, std2, std3, std4, 0, StringToInt(std6), 0);
            TMS_Main.D_Accounts.push_back(temp);
        }
        else {
            Customer temp(std1, std2, std3, std4, 0);
            TMS_Main.C_Accounts.push_back(temp);
        }
        TMS_Main.SaveLoadedData();
        MessageBox::Show("Account Created Succesfully, Please close the
application to login with your new account", "Error: Account Creation Successful",
MessageBoxButtons::OK, MessageBoxIcon::Information);
    }
    private: System::Void label10_Click(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void textBox6_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ s1 = Convert::ToString(textBox6->Text);
        std::string s2;
        s2 = context.marshal_as<std::string>(s1);
        if (!(isInt(s2))) {
            this->label11->Text = ("Please Enter a Valid Number");
        }
        else {
            this->label11->Text = ("");
        }
    }
    private: System::Void label11_Click(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void Signup_Form_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {
        TMS_Main.SaveLoadedData();
        Application::Exit();
    }
#pragma endregion
    };
#pragma endregion
}
```

Customer Page (Customer_Form.h)

```
#pragma once
#include "Distance.h"
#include "Driver_Service.h"
#include "RatingGiver.h"
#include "Vehicle_Service.h"
int i, dID, vID, oID;
int price = 35;
extern int disx;
extern float r1, r2;
bool HideVcl;
extern TMS TMS_Main;
namespace TMS_Forms {
      using namespace System;
      using namespace System::ComponentModel;
      using namespace System::Collections;
      using namespace System::Windows::Forms;
      using namespace System::Data;
      using namespace System::Drawing;
#pragma region Customer Form
      public ref class Customer_Form : public System::Windows::Forms::Form {
      public:
            Customer_Form(void) {
                  InitializeComponent();
            }
      protected:
            ~Customer_Form() {
                  if (components) {
                        delete components;
                  }
            }
      private: System::Windows::Forms::Label^ label1;
      private: System::Windows::Forms::Label^ label2;
      private: System::Windows::Forms::Label^ label3;
      private: System::Windows::Forms::RadioButton^ radioButton1;
      private: System::Windows::Forms::RadioButton^ radioButton2;
      private: System::Windows::Forms::RadioButton^ radioButton3;
      private: System::Windows::Forms::TextBox^ textBox1;
      private: System::Windows::Forms::Label^ label4;
      private: System::Windows::Forms::Label^ label5;
      private: System::Windows::Forms::Button^ button1;
      private: System::Windows::Forms::Button^ button2;
      private: System::Windows::Forms::Button^ button3;
      private: System::Windows::Forms::RadioButton^ radioButton4;
      private: System::Windows::Forms::RichTextBox^ richTextBox1;
      private: System::Windows::Forms::Button^ button4;
      private: System::Windows::Forms::Button^ button5;
      private: System::Windows::Forms::Button^ button6;
      private: System::Windows::Forms::CheckBox^ checkBox1;
      private: System::Windows::Forms::CheckBox^ checkBox2;
      private: System::Windows::Forms::Button^ button7;
      private: System::Windows::Forms::Button^ button8;
      private: System::Windows::Forms::Button^ button9;
      private: System::Windows::Forms::Button^ button10;
      private: System::Windows::Forms::Button^ button11;
      private: System::ComponentModel::Container^ components;
#pragma region Component Code
                void InitializeComponent(void)
                {
                      System::ComponentModel::ComponentResourceManager^ resources =
(gcnew System::ComponentModel::ComponentResourceManager(Customer_Form::typeid));
                      this->label1 = (gcnew System::Windows::Forms::Label());
                      this->label2 = (gcnew System::Windows::Forms::Label());
                      this->label3 = (gcnew System::Windows::Forms::Label());
                      this->radioButton1 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton2 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton3 = (gcnew
System::Windows::Forms::RadioButton());
                      this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                      this->label4 = (gcnew System::Windows::Forms::Label());
                      this->label5 = (gcnew System::Windows::Forms::Label());
                      this->button1 = (gcnew System::Windows::Forms::Button());
                      this->button2 = (gcnew System::Windows::Forms::Button());
                      this->button3 = (gcnew System::Windows::Forms::Button());
                      this->button4 = (gcnew System::Windows::Forms::Button());
                      this->radioButton4 = (gcnew
System::Windows::Forms::RadioButton());
                      this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
                      this->button5 = (gcnew System::Windows::Forms::Button());
                      this->button6 = (gcnew System::Windows::Forms::Button());
                      this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
                      this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
                      this->button7 = (gcnew System::Windows::Forms::Button());
                      this->button8 = (gcnew System::Windows::Forms::Button());
                      this->button9 = (gcnew System::Windows::Forms::Button());
                      this->button10 = (gcnew System::Windows::Forms::Button());
                      this->button11 = (gcnew System::Windows::Forms::Button());
                      this->SuspendLayout();
                      //
                      // label1
                      //
                      this->label1->AutoSize = true;
                      this->label1->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label1->Location = System::Drawing::Point(12, 19);
                      this->label1->Name = L"label1";
                      this->label1->Size = System::Drawing::Size(66, 26);
                      this->label1->TabIndex = 0;
                      this->label1->Text = L"label1";
                      this->label1->Click += gcnew System::EventHandler(this,
&Customer_Form::label1_Click);
                      //
                      // label2
                      //
                      this->label2->AutoSize = true;
                      this->label2->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label2->Location = System::Drawing::Point(299, 15);
                      this->label2->Name = L"label2";
                      this->label2->Size = System::Drawing::Size(52, 21);
                      this->label2->TabIndex = 1;
                      this->label2->Text = L"label2";
                      this->label2->Click += gcnew System::EventHandler(this,
&Customer_Form::label2_Click);
                      //
                      // label3
                      //
                      this->label3->AutoSize = true;
                      this->label3->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label3->Location = System::Drawing::Point(12, 48);
                      this->label3->Name = L"label3";
                      this->label3->Size = System::Drawing::Size(54, 22);
                      this->label3->TabIndex = 2;
                      this->label3->Text = L"label3";
                      this->label3->Click += gcnew System::EventHandler(this,
&Customer_Form::label3_Click);
                      //
                      // radioButton1
                      //
                      this->radioButton1->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton1->AutoSize = true;
                      this->radioButton1->BackColor =
System::Drawing::Color::WhiteSmoke;
                      this->radioButton1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton1->Location = System::Drawing::Point(16,
131);
                      this->radioButton1->Name = L"radioButton1";
                      this->radioButton1->Size = System::Drawing::Size(88, 49);
                      this->radioButton1->TabIndex = 3;
                      this->radioButton1->TabStop = true;
                      this->radioButton1->Text = L"\nDeposit Money\n ";
                      this->radioButton1->UseVisualStyleBackColor = false;
                      this->radioButton1->CheckedChanged += gcnew
System::EventHandler(this, &Customer_Form::radioButton1_CheckedChanged);
                      //
                      // radioButton2
                      //
                      this->radioButton2->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton2->AutoSize = true;
                      this->radioButton2->BackColor =
System::Drawing::Color::WhiteSmoke;
                      this->radioButton2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton2->Location = System::Drawing::Point(16,
186);
                      this->radioButton2->Name = L"radioButton2";
                      this->radioButton2->Size = System::Drawing::Size(91, 49);
                      this->radioButton2->TabIndex = 4;
                      this->radioButton2->TabStop = true;
                      this->radioButton2->Text = L"\n    Make Order  \n ";
                      this->radioButton2->UseVisualStyleBackColor = false;
                      this->radioButton2->CheckedChanged += gcnew 
System::EventHandler(this, &Customer_Form::radioButton2_CheckedChanged);
                      // 
                      // radioButton3
                      // 
                      this->radioButton3->Appearance = 
System::Windows::Forms::Appearance::Button;
                      this->radioButton3->AutoSize = true;
                      this->radioButton3->BackColor = 
System::Drawing::Color::WhiteSmoke;
                      this->radioButton3->FlatStyle = 
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton3->Location = System::Drawing::Point(16, 
241);
                      this->radioButton3->Name = L"radioButton3";
                      this->radioButton3->Size = System::Drawing::Size(92, 49);
                      this->radioButton3->TabIndex = 5;
                      this->radioButton3->TabStop = true;
                      this->radioButton3->Text = L"\n   View Orders   \n ";
                      this->radioButton3->UseVisualStyleBackColor = false;
                      this->radioButton3->CheckedChanged += gcnew 
System::EventHandler(this, &Customer_Form::radioButton3_CheckedChanged);
                      // 
                      // textBox1
                      // 
                      this->textBox1->Location = System::Drawing::Point(211, 131);
                      this->textBox1->Name = L"textBox1";
                      this->textBox1->Size = System::Drawing::Size(199, 20);
                      this->textBox1->TabIndex = 7;
                      this->textBox1->TextChanged += gcnew 
System::EventHandler(this, &Customer_Form::textBox1_TextChanged);
                      // 
                      // label4
                      // 
                      this->label4->AutoSize = true;
                      this->label4->Font = (gcnew System::Drawing::Font(L"Palatino 
Linotype", 9, System::Drawing::FontStyle::Bold, 
System::Drawing::GraphicsUnit::Point, 
                            static_cast<System::Byte>(0)));
                      this->label4->Location = System::Drawing::Point(208, 114);
                      this->label4->Name = L"label4";
                      this->label4->Size = System::Drawing::Size(155, 17);
                      this->label4->TabIndex = 8;
                      this->label4->Text = L"Enter Ammount to Deposit";
                      this->label4->Click += gcnew System::EventHandler(this, 
&Customer_Form::label4_Click);
                      // 
                      // label5
                      // 
                      this->label5->AutoSize = true;
                      this->label5->Location = System::Drawing::Point(212, 151);
                      this->label5->Name = L"label5";
                      this->label5->Size = System::Drawing::Size(10, 13);
                      this->label5->TabIndex = 9;
                      this->label5->Text = L" ";
                      //
                      // button1
                      //
                      this->button1->BackColor =
System::Drawing::Color::Transparent;
                      this->button1->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Zoom;
                      this->button1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button1->Font = (gcnew System::Drawing::Font(L"Arial
Rounded MT Bold", 26.25F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                      this->button1->Location = System::Drawing::Point(211, 241);
                      this->button1->Name = L"button1";
                      this->button1->Size = System::Drawing::Size(55, 49);
                      this->button1->TabIndex = 10;
                      this->button1->Text = L"<";
                      this->button1->UseVisualStyleBackColor = false;
                      this->button1->Click += gcnew System::EventHandler(this,
&Customer_Form::button1_Click);
                      //
                      // button2
                      //
                      this->button2->BackColor =
System::Drawing::SystemColors::ControlDark;
                      this->button2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button2->Location = System::Drawing::Point(272, 241);
                      this->button2->Name = L"button2";
                      this->button2->Size = System::Drawing::Size(89, 49);
                      this->button2->TabIndex = 11;
                      this->button2->Text = L"Show Vehicles";
                      this->button2->UseVisualStyleBackColor = false;
                      this->button2->Click += gcnew System::EventHandler(this,
&Customer_Form::button2_Click);
                      //
                      // button3
                      //
                      this->button3->BackColor =
System::Drawing::SystemColors::ControlDark;
                      this->button3->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button3->Location = System::Drawing::Point(367, 241);
                      this->button3->Name = L"button3";
                      this->button3->Size = System::Drawing::Size(87, 49);
                      this->button3->TabIndex = 12;
                      this->button3->Text = L"Place Order";
                      this->button3->UseVisualStyleBackColor = false;
                      this->button3->Click += gcnew System::EventHandler(this,
&Customer_Form::button3_Click);
                      //
                      // button4
                      //
                      this->button4->BackColor =
System::Drawing::Color::Transparent;
                      this->button4->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Zoom;
```

```
            this->button4->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
            this->button4->Font = (gcnew System::Drawing::Font(L"Arial
Rounded MT Bold", 26.25F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
            this->button4->Location = System::Drawing::Point(460, 241);
            this->button4->Name = L"button4";
            this->button4->Size = System::Drawing::Size(55, 49);
            this->button4->TabIndex = 13;
            this->button4->Text = L">";
            this->button4->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
            this->button4->UseVisualStyleBackColor = false;
            this->button4->Click += gcnew System::EventHandler(this,
&Customer_Form::button4_Click);
            //
            // radioButton4
            //
            this->radioButton4->Appearance =
System::Windows::Forms::Appearance::Button;
            this->radioButton4->AutoSize = true;
            this->radioButton4->BackColor =
System::Drawing::Color::WhiteSmoke;
            this->radioButton4->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
            this->radioButton4->Location = System::Drawing::Point(15,
296);
            this->radioButton4->Name = L"radioButton4";
            this->radioButton4->Size = System::Drawing::Size(91, 49);
            this->radioButton4->TabIndex = 14;
            this->radioButton4->TabStop = true;
            this->radioButton4->Text = L"\nDelete Account\n ";
            this->radioButton4->UseVisualStyleBackColor = false;
            this->radioButton4->CheckedChanged += gcnew
System::EventHandler(this, &Customer_Form::radioButton4_CheckedChanged);
            //
            // richTextBox1
            //
            this->richTextBox1->BackColor =
System::Drawing::SystemColors::MenuBar;
            this->richTextBox1->Font = (gcnew
System::Drawing::Font(L"Palatino Linotype", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
            this->richTextBox1->Location = System::Drawing::Point(211,
104);
            this->richTextBox1->Name = L"richTextBox1";
            this->richTextBox1->ReadOnly = true;
            this->richTextBox1->Size = System::Drawing::Size(304, 131);
            this->richTextBox1->TabIndex = 16;
            this->richTextBox1->Text = L"";
            this->richTextBox1->TextChanged += gcnew
System::EventHandler(this, &Customer_Form::richTextBox1_TextChanged);
            //
            // button5
            //
```

```
                      this->button5->Font = (gcnew System::Drawing::Font(L"OCR A
Extended", 9, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button5->Location = System::Drawing::Point(272, 291);
                      this->button5->Name = L"button5";
                      this->button5->Size = System::Drawing::Size(42, 21);
                      this->button5->TabIndex = 17;
                      this->button5->Text = L"<";
                      this->button5->UseVisualStyleBackColor = true;
                      this->button5->Click += gcnew System::EventHandler(this,
&Customer_Form::button5_Click);
                      //
                      // button6
                      //
                      this->button6->Font = (gcnew System::Drawing::Font(L"OCR A
Extended", 9, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button6->Location = System::Drawing::Point(319, 291);
                      this->button6->Name = L"button6";
                      this->button6->Size = System::Drawing::Size(42, 21);
                      this->button6->TabIndex = 18;
                      this->button6->Text = L">";
                      this->button6->UseVisualStyleBackColor = true;
                      this->button6->Click += gcnew System::EventHandler(this,
&Customer_Form::button6_Click);
                      //
                      // checkBox1
                      //
                      this->checkBox1->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->checkBox1->AutoSize = true;
                      this->checkBox1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox1->Location = System::Drawing::Point(211, 82);
                      this->checkBox1->Name = L"checkBox1";
                      this->checkBox1->Size = System::Drawing::Size(82, 23);
                      this->checkBox1->TabIndex = 19;
                      this->checkBox1->Text = L"Request Ride";
                      this->checkBox1->UseVisualStyleBackColor = true;
                      this->checkBox1->CheckedChanged += gcnew
System::EventHandler(this, &Customer_Form::checkBox1_CheckedChanged);
                      //
                      // checkBox2
                      //
                      this->checkBox2->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->checkBox2->AutoSize = true;
                      this->checkBox2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox2->Location = System::Drawing::Point(292, 82);
                      this->checkBox2->Name = L"checkBox2";
                      this->checkBox2->Size = System::Drawing::Size(98, 23);
                      this->checkBox2->TabIndex = 20;
                      this->checkBox2->Text = L"Request Delivery";
                      this->checkBox2->UseVisualStyleBackColor = true;
                      this->checkBox2->CheckedChanged += gcnew
System::EventHandler(this, &Customer_Form::checkBox2_CheckedChanged);
                      //
                      // button7
                      //
                      this->button7->BackColor =
System::Drawing::SystemColors::ControlDark;
                      this->button7->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button7->Location = System::Drawing::Point(414, 57);
                      this->button7->Name = L"button7";
                      this->button7->Size = System::Drawing::Size(100, 49);
                      this->button7->TabIndex = 21;
                      this->button7->Text = L"Set Departure and Arrival";
                      this->button7->UseVisualStyleBackColor = false;
                      this->button7->Click += gcnew System::EventHandler(this,
&Customer_Form::button7_Click);
                      //
                      // button8
                      //
                      this->button8->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button8->Location = System::Drawing::Point(521, 104);
                      this->button8->Name = L"button8";
                      this->button8->Size = System::Drawing::Size(91, 37);
                      this->button8->TabIndex = 22;
                      this->button8->Text = L"Sort by Rating";
                      this->button8->UseVisualStyleBackColor = true;
                      this->button8->Click += gcnew System::EventHandler(this,
&Customer_Form::button8_Click);
                      //
                      // button9
                      //
                      this->button9->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button9->Location = System::Drawing::Point(521, 147);
                      this->button9->Name = L"button9";
                      this->button9->Size = System::Drawing::Size(91, 37);
                      this->button9->TabIndex = 23;
                      this->button9->Text = L"Sort by Experience";
                      this->button9->UseVisualStyleBackColor = true;
                      this->button9->Click += gcnew System::EventHandler(this,
&Customer_Form::button9_Click);
                      //
                      // button10
                      //
                      this->button10->Location = System::Drawing::Point(368, 312);
                      this->button10->Name = L"button10";
                      this->button10->Size = System::Drawing::Size(86, 37);
                      this->button10->TabIndex = 24;
                      this->button10->Text = L"View Driver History";
                      this->button10->UseVisualStyleBackColor = true;
                      this->button10->Click += gcnew System::EventHandler(this,
&Customer_Form::button10_Click);
                      //
                      // button11
                      //
                      this->button11->Location = System::Drawing::Point(272, 312);
                      this->button11->Name = L"button11";
                      this->button11->Size = System::Drawing::Size(91, 37);
                      this->button11->TabIndex = 25;
                      this->button11->Text = L"View Vehicle History";
                      this->button11->UseVisualStyleBackColor = true;
                      this->button11->Click += gcnew System::EventHandler(this,
&Customer_Form::button11_Click);
                      //
                      // Customer_Form
                      //
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(635, 361);
                      this->Controls->Add(this->button11);
                      this->Controls->Add(this->button10);
                      this->Controls->Add(this->button9);
                      this->Controls->Add(this->button8);
                      this->Controls->Add(this->button7);
                      this->Controls->Add(this->checkBox2);
                      this->Controls->Add(this->checkBox1);
                      this->Controls->Add(this->button6);
                      this->Controls->Add(this->button5);
                      this->Controls->Add(this->richTextBox1);
                      this->Controls->Add(this->radioButton4);
                      this->Controls->Add(this->button4);
                      this->Controls->Add(this->button3);
                      this->Controls->Add(this->button2);
                      this->Controls->Add(this->button1);
                      this->Controls->Add(this->label5);
                      this->Controls->Add(this->label4);
                      this->Controls->Add(this->textBox1);
                      this->Controls->Add(this->radioButton3);
                      this->Controls->Add(this->radioButton2);
                      this->Controls->Add(this->radioButton1);
                      this->Controls->Add(this->label3);
                      this->Controls->Add(this->label2);
                      this->Controls->Add(this->label1);
                      this->Name = L"Customer_Form";
                      this->Text = L"Transport Manager+";
                      this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&Customer_Form::Customer_Form_FormClosing);
                      this->Load += gcnew System::EventHandler(this,
&Customer_Form::Customer_Form_Load);
                      this->ResumeLayout(false);
                      this->PerformLayout();

                }
#pragma endregion
#pragma region Function Code
      private: System::Void Customer_Form_Load(System::Object^ sender,
System::EventArgs^ e) {
             TMS_Main.LoadSavedData();
             std::ifstream CurrentAcc;
             std::string s1, s2, s3;
             CurrentAcc.open("Data/CurrentAcc.txt");
             while (!CurrentAcc.eof()) {
```

```
            CurrentAcc >> s1 >> s2;
        }
        if (s1 == "Customer") {
            i = StringToInt(s2);
            System::String^ name, ^ balance, ^ cnic;
            s3 = TMS_Main.C_Accounts[i]->getFName() + " " +
TMS_Main.C_Accounts[i]->getLName();
            name = gcnew String(s3.data());
            s3 = TMS_Main.C_Accounts[i]->getID();
            cnic = gcnew String(s3.data());
            s3 = "Balance: PKR " + IntToString(TMS_Main.C_Accounts[i]-
>getbal());
            balance = gcnew String(s3.data());
            this->label1->Text = (name);
            this->label3->Text = (cnic);
            this->label2->Text = (balance);
        }
        else {
            this->Close();
        }
    }
    private: System::Void label2_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void label3_Click(System::Object^ sender, System::EventArgs^
e) {

    }
    private: System::Void label4_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void richTextBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            this->radioButton2->Checked = false;
            this->radioButton3->Checked = false;
            this->richTextBox1->Hide();
            this->textBox1->Show();
            this->label4->Text = ("Enter Ammount to Deposit");
            this->label4->Show();
            this->label5->Show();
            this->button2->Text = ("Deposit");
            this->button2->Location = System::Drawing::Point(321, 171);
            this->button1->Hide();
            this->button2->Show();
            this->button3->Hide();
            this->button4->Hide();
            this->button5->Hide();
            this->button6->Hide();
            this->button7->Hide();
            this->button8->Hide();
            this->button9->Hide();
            this->button10->Hide();
            this->button11->Hide();
            this->checkBox1->Hide();
            this->checkBox2->Hide();
            this->radioButton1->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->radioButton1->BackColor =
System::Drawing::Color::WhiteSmoke;
            this->textBox1->Text = "";
            this->button2->Location = System::Drawing::Point(272, 241);
        }
    }
    private: System::Void radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            this->radioButton1->Checked = false;
            this->radioButton3->Checked = false;
            this->richTextBox1->Show();
            this->button2->Text = ("Show Vehicles");
            this->button3->Text = ("Place Order");
            dID = 0;
            std::string str = TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating());
            System::String^ st = gcnew String(str.data());
            richTextBox1->Text = (st);
            this->textBox1->Hide();
            this->button1->Show();
            this->button2->Show();
            this->button3->Show();
            this->button4->Show();
            this->label4->Hide();
            this->label5->Hide();
            this->radioButton2->BackColor =
System::Drawing::SystemColors::ControlDark;
            this->button2->Text = ("Show Vehicles");
            this->button5->Hide();
            this->button6->Hide();
            this->button8->Show();
            this->button9->Show();
            this->button10->Show();
            this->button11->Hide();
            this->button7->Text = "Set Departure and Arrival";
            this->button7->Show();
            this->checkBox1->Checked = true;
            this->checkBox1->Show();
            this->checkBox2->Show();
            HideVcl = 0;
        }
        else {
            this->radioButton2->BackColor =
System::Drawing::Color::WhiteSmoke;

        }
    }
    private: System::Void radioButton3_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton3->Checked) {
```

```
                    this->radioButton1->Checked = false;
                    this->radioButton2->Checked = false;
                    this->button2->Text = ("Cancel Order");
                    this->button3->Text = ("Confirm Order");
                    this->richTextBox1->Show();
                    this->richTextBox1->Text = ("\nYou have not placed any orders
yet");
                    this->textBox1->Hide();
                    this->button1->Show();
                    this->button2->Show();
                    this->button3->Show();
                    this->button4->Show();
                    this->button5->Hide();
                    this->button6->Hide();
                    this->button7->Text = "Give Rating";
                    this->button7->Show();
                    this->button8->Hide();
                    this->button9->Hide();
                    this->button10->Hide();
                    this->button11->Hide();
                    dID = 0;
                    oID = 0;
                    if (TMS_Main.Orders.size()) {
                           while (TMS_Main.C_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getCID() && oID < TMS_Main.Orders.size() - 1) {
                                  oID++;
                           }
                    }
                    this->label4->Hide();
                    this->label5->Hide();
                    this->checkBox1->Hide();
                    this->checkBox2->Hide();
                    this->radioButton3->BackColor =
System::Drawing::SystemColors::ControlDark;
                    if (TMS_Main.Orders.size()) {
                           int s = TMS_Main.Orders[oID]->getPlaced(), h, m;
                           std::string AmPm = "AM";
                           std::string s2;
                           h = (s / 3600) % 24 + 5;
                           if (TMS_Main.Orders[oID]->getAccepted()) s2 = "Accepted";
                           else s2 = "Not Accepted";
                           if (h > 12) {
                                  h %= 12;
                                  AmPm = "PM";
                           }
                           s %= 3600;
                           m = s / 60;
                           s /= 60;
                           std::string str = "Order for a " + TMS_Main.Orders[oID]-
>getType() + ", Placed On " + IntToString(h) + ":";
                           if (m < 10) str = str + "0";
                           str = str + IntToString(m) + AmPm + "\nDriver: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nStatus: " + s2;
                if (TMS_Main.C_Accounts[i]->getID() ==
TMS_Main.Orders[oID]->getCID()) {
                    System::String^ st2 = gcnew String(str.data());
                    this->richTextBox1->Text = (st2);
                }
            }
        }
        else {
            this->radioButton3->BackColor =
System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void radioButton4_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton4->Checked) {
            this->radioButton1->Checked = false;
            this->radioButton2->Checked = false;
            this->radioButton3->Checked = false;
            this->richTextBox1->Hide();
            this->textBox1->Show();
            this->label4->Show();
            this->label4->Text = ("Please enter your Password to to Delete
the account");
            this->label5->Hide();
            this->button2->Text = ("Delete");
            this->button2->Location = System::Drawing::Point(321, 171);
            this->button1->Hide();
            this->button2->Show();
            this->button3->Hide();
            this->button4->Hide();
            this->button5->Hide();
            this->button6->Hide();
            this->button7->Hide();
            this->button8->Hide();
            this->button9->Hide();
            this->button10->Hide();
            this->button11->Hide();
            this->checkBox1->Hide();
            this->checkBox2->Hide();
            this->radioButton4->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->radioButton4->BackColor =
System::Drawing::Color::WhiteSmoke;
            this->textBox1->Text = "";
            this->button2->Location = System::Drawing::Point(272, 241);
        }
    }
    private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ s1 = Convert::ToString(textBox1->Text);
        std::string s2 = context.marshal_as<std::string>(s1);
        if (!(isInt(s2))) {
            this->label5->Text = ("Please Enter a Number");
        }
        else {
            this->label5->Text = ("");
        }
    }
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            if (TMS_Main.D_Accounts.size()) {
                vID = 0;
                if (dID > 0) {
                    dID--;
                    while (TMS_Main.D_Accounts[dID]->getFreedom()) {
                        dID--;
                    }
                    if (!HideVcl) {
                        std::string str = TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating());
                        System::String^ st = gcnew
String(str.data());
                        richTextBox1->Text = (st);
                    }
                    else {
                        if (TMS_Main.D_Accounts[dID]->Vehicles.size()
> 0) {
                            std::string str =
TMS_Main.D_Accounts[dID]->getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() +
"\nWork Experience: " + IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating:
" + FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n---------
-----------------------------------------------------------------\n" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->ServiceType + " Vehicle (" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Type + ")\n" + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + " Model\nVehicle Rating: " + FloatToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->ComputeAndReturnRating());
                            System::String^ st = gcnew
String(str.data());
                            richTextBox1->Text = (st);
                        }
                        else {
                            std::string str =
TMS_Main.D_Accounts[dID]->getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() +
"\nWork Experience: " + IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating:
" + FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n---------
-----------------------------------------------------------------\n\nThis Driver has
no vehicles yet";
                            System::String^ st = gcnew
String(str.data());
                            richTextBox1->Text = (st);
                        }
                    }
                }
            }
        }
        if (this->radioButton3->Checked) {
            if (TMS_Main.Orders.size()) {
                if (oID > 0) {
                    oID--;
                }
                while (TMS_Main.C_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getCID() && oID > 0) {
                    oID--;
                }
                if (oID <= 0 && TMS_Main.C_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getCID()) {
                    while (TMS_Main.C_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getCID() && oID < TMS_Main.Orders.size() - 1) {
                        oID++;
                    }
                }
                if (TMS_Main.Orders.size()) {
                    int s = TMS_Main.Orders[oID]->getPlaced(), h, m;
                    std::string AmPm = "AM";
                    std::string s2;
                    h = (s / 3600) % 24 + 5;
                    if (TMS_Main.Orders[oID]->getAccepted()) s2 =
"Accepted";
                    else s2 = "Not Accepted";
                    if (h > 12) {
                        h %= 12;
                        AmPm = "PM";
                    }
                    s %= 3600;
                    m = s / 60;
                    s /= 60;
                    std::string str = "Order for a " +
TMS_Main.Orders[oID]->getType() + ", Placed On " + IntToString(h) + ":";
                    if (m < 10) str = str + "0";
                    str = str + IntToString(m) + AmPm + "\nDriver: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nStatus: " + s2;
                    System::String^ st2 = gcnew String(str.data());
                    this->richTextBox1->Text = (st2);
                }
            }
        }
    }
    private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            msclr::interop::marshal_context context;
            System::String^ s1 = Convert::ToString(textBox1->Text);
            std::string s2 = context.marshal_as<std::string>(s1);
            if (!(isInt(s2))) {
                MessageBox::Show("Please enter a valid Number", "Error:
Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            }
            else {
                TMS_Main.C_Accounts[i]->addbal(StringToInt(s2));
                std::string s3 = "Balance: PKR " +
IntToString(TMS_Main.C_Accounts[i]->getbal());
                String^ balance = gcnew String(s3.data());
                this->label2->Text = (balance);
                TMS_Main.SaveLoadedData();
            }
        }
        if (this->radioButton4->Checked) {
            msclr::interop::marshal_context context;
            System::String^ s1 = Convert::ToString(textBox1->Text);
            std::string s2 = context.marshal_as<std::string>(s1);
            if (s2 == TMS_Main.C_Accounts[i]->getPass()) {
                if (MessageBox::Show("Are you sure you want to delete your
account?\nIt cannot be recovered once deleted", "Confirm Account Deletion",
MessageBoxButtons::YesNo, MessageBoxIcon::Warning) ==
System::Windows::Forms::DialogResult::Yes) {
                    TMS_Main.DeleteCAccount(i);
                    MessageBox::Show("Account Deleted, Application will
close now so you may Log-In with or Create another account", "Account Deletion
Succesful", MessageBoxButtons::OK, MessageBoxIcon::Information);
                    this->Close();
                }
            }
            else {
                MessageBox::Show("Incorrect Password Entered", "Error:
Incorrect Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            }
        }
        if (this->radioButton2->Checked) {
            if (!HideVcl) {
                this->button2->Text = ("Hide Vehicles");
                vID = 0;
                this->button5->Show();
                this->button6->Show();
                this->button11->Show();
                if (TMS_Main.D_Accounts[dID]->Vehicles.size() > 0) {
                    std::string str = TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n-------------
-------------------------------------------------------------\n" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->ServiceType + " " +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Type + "\n" + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + " Model\nVehicle Rating: " + FloatToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->ComputeAndReturnRating());
                    System::String^ st = gcnew String(str.data());
                    richTextBox1->Text = (st);
                }
                else {
                    std::string str = TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n-------------
-------------------------------------------------------------\n\nThis Driver has no
vehicles yet";
                    System::String^ st = gcnew String(str.data());
                    richTextBox1->Text = (st);
                }
            }
            else {
                this->button2->Text = ("Show Vehicles");
                this->button5->Hide();
                this->button6->Hide();
                this->button11->Hide();
                std::string str = TMS_Main.D_Accounts[dID]->getFName() + "
" + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating());
                System::String^ st = gcnew String(str.data());
                richTextBox1->Text = (st);
            }
            HideVcl = !HideVcl;
        }
        if (this->radioButton3->Checked) {
            if (TMS_Main.Orders.size()) {
                std::string str = "Are you sure you want to cancel your
order for a  " + TMS_Main.Orders[oID]->getType() + " with " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "'s " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]-
>getDID())]->Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(),
TMS_Main.Orders[oID]->getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nYou will Recieve only PKR " +
IntToString(TMS_Main.Orders[oID]->getCost() * 0.95) + " back";
                System::String^ st2 = gcnew String(str.data());
                if (MessageBox::Show(st2, "Confirm Order Cancellation",
MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
System::Windows::Forms::DialogResult::Yes) {
                    TMS_Main.C_Accounts[i]-
>addbal(TMS_Main.Orders[oID]->getCost() * 0.95);
                    std::string s3 = "Balance: PKR " +
IntToString(TMS_Main.C_Accounts[i]->getbal());
                    System::String^ balance = gcnew String(s3.data());
                    this->label2->Text = (balance);
                    TMS_Main.CancelOrder(oID);
                    oID = 0;
                    this->richTextBox1->Text = ("\nYou have not placed
any orders yet");
                    if (TMS_Main.Orders.size()) {
                        while (TMS_Main.C_Accounts[i]->getID() != 
TMS_Main.Orders[oID]->getCID() && oID < TMS_Main.Orders.size() - 1) {
                            oID++;
                        }
                        int s = TMS_Main.Orders[oID]->getPlaced(), h, 
m;
                        std::string AmPm = "AM";
                        h = (s / 3600) % 24 + 5;
                        if (h > 12) {
                            h %= 12;
                            AmPm = "PM";
                        }
                        s %= 3600;
                        m = s / 60;
                        s /= 60;
                        std::string str = "Order for a " + 
TMS_Main.Orders[oID]->getType() + ", Placed On " + IntToString(h) + ":";
                        if (m < 10) str = str + "0";
                        str = str + IntToString(m) + AmPm + 
"\nDriver: " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]-
>getDID())]->getFName() + " " + 
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getLName() 
+ "\nSelected Vehicle: " + 
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " + 
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " + 
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year);
                        if (TMS_Main.C_Accounts[i]->getID() == 
TMS_Main.Orders[oID]->getCID()) {
                            System::String^ st2 = gcnew 
String(str.data());
                            this->richTextBox1->Text = (st2);
                        }
                    }
                }
            }
        }
    }
    private: System::Void button3_Click(System::Object^ sender, 
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            if (TMS_Main.D_Accounts[dID]->Vehicles.size() > 0) {
                if (!HideVcl) {
                    MessageBox::Show("You have not selected a vehicle, 
please click on \"Show Vehicles\" and select a vehicle", "Error: No Vehicle 
Selected", MessageBoxButtons::OK, MessageBoxIcon::Warning);
                    return;
                }
                else {
                    std::string s1;
                    if (this->checkBox1->Checked) {
                        s1 = "Ride";
```

```
                                      if (TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>ServiceType != "Ride") {
                                             MessageBox::Show("You cannot request a
Ride on a vehicle reserved for Deliveries", "Error: Incorrect Vehicle Selected",
MessageBoxButtons::OK, MessageBoxIcon::Stop);
                                             return;
                                      }
                                }
                                else {
                                      s1 = "Delivery";
                                      if (TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>ServiceType != "Delivery") {
                                             MessageBox::Show("You cannot request a
Delivery on a vehicle reserved for Rides", "Error: Incorrect Vehicle Selected",
MessageBoxButtons::OK, MessageBoxIcon::Stop);
                                             return;
                                      }
                                }
                                int distance = disx;
                                int cost = distance * price;
                                if (cost == 0) {
                                      MessageBox::Show("Arrival and Departure
cannot be at the same location", "Error: Invalid Locations", MessageBoxButtons::OK,
MessageBoxIcon::Stop);
                                      return;
                                }
                                if (s1 == "Ride") cost = cost + (cost * 0.2);
                                std::string str = "Are you sure you want to plave
an order for a  " + s1 + " with " + TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName() + "'s " + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + "\nThis Order will cost you PKR " + IntToString(cost) + "\n\nOnce
placed, the order can be cancelled with only 95% return of order cost";
                                System::String^ st2 = gcnew String(str.data());
                                if (MessageBox::Show(st2, "Confirm Order
Placement", MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
System::Windows::Forms::DialogResult::Yes) {
                                      if (TMS_Main.C_Accounts[i]->getbal() >= cost)
{

      TMS_Main.MakeOrder(*TMS_Main.C_Accounts[i],
*TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.D_Accounts[dID]->getID())],
*TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.D_Accounts[dID]->getID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.D_Accounts[dID]->getID(),
TMS_Main.D_Accounts[dID]->Vehicles[vID]->getID())], s1, cost);
                                             TMS_Main.Orders = TMS_Main.Orders;
                                             dID = 0;
                                             vID = 0;
                                             TMS_Main.C_Accounts[i]-
>deductbal(cost);
                                             std::string s3 = "Balance: PKR " +
IntToString(TMS_Main.C_Accounts[i]->getbal());
                                             System::String^ balance = gcnew
String(s3.data());
                                             this->label2->Text = (balance);
                                             if (TMS_Main.D_Accounts[dID]-
>Vehicles.size() > 0) {
```

```
                                std::string str =
TMS_Main.D_Accounts[dID]->getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() +
"\nWork Experience: " + IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating:
" + FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n---------
-----------------------------------------------------------------\n" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->ServiceType + " " +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Type + "\n" + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + " Model\nVehicle Rating: " + FloatToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->ComputeAndReturnRating());
                                System::String^ st = gcnew
String(str.data());
                                richTextBox1->Text = (st);
                            }
                            else {
                                std::string str =
TMS_Main.D_Accounts[dID]->getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() +
"\nWork Experience: " + IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating:
" + FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n---------
-----------------------------------------------------------------\n\nThis Driver has
no vehicles yet";
                                System::String^ st = gcnew
String(str.data());
                                richTextBox1->Text = (st);
                            }
                        }
                        else {
                            MessageBox::Show("You do not have
sufficent balance to place this order, Please deposit more balance before you
continue", "Error: Insufficent Balance", MessageBoxButtons::OK,
MessageBoxIcon::Stop);
                        }
                    }
                }
            }
            else {
                MessageBox::Show("You cannot place an order with this
driver as they do not own any vehicles", "Error: No Vehicles Found",
MessageBoxButtons::OK, MessageBoxIcon::Warning);
                return;
            }
        }
        if (this->radioButton3->Checked) {
            if (TMS_Main.Orders.size()) {
                if (TMS_Main.Orders[oID]->getAccepted()) {
                    if (r1 != 6 && r2 != 6) {
                        if (MessageBox::Show("By Confirming this
order, you are confirming that the order has been completed\nThe current ratings
selected in the \"Give Ratings\" menu will also be submitted\nAre you sure you want
to confirm?", "Confirm Order Completion", MessageBoxButtons::YesNo,
MessageBoxIcon::Question) == System::Windows::Forms::DialogResult::Yes) {

    TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>addbal(TMS_Main.Orders[oID]->getCost());

    TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>setFreedom(0);
```

```
    TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>scores.push_back(r1);

    TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->scores.push_back(r2);
                            TMS_Main.CompleteOrder(oID);
                            r1 = 6; r2 = 6;
                        }
                    }
                    else {
                        MessageBox::Show("Please Rate the Driver and
the Vehicle", "Error: Rating Submission", MessageBoxButtons::OK,
MessageBoxIcon::Stop);
                        return;
                    }
                }
                else {
                    MessageBox::Show("The driver has not accepted this
order yet", "Error: Un-Accepted Order", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                    return;
                }
            }
        }
    }
    private: System::Void button4_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            vID = 0;
            if (dID < TMS_Main.D_Accounts.size() - 1) {
                dID++;
                while (TMS_Main.D_Accounts[dID]->getFreedom()) {
                    dID++;
                }
                if (!HideVcl) {
                    std::string str = TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating());
                    System::String^ st = gcnew String(str.data());
                    richTextBox1->Text = (st);
                }
                else {
                    if (TMS_Main.D_Accounts[dID]->Vehicles.size() > 0)
{
                        std::string str = TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n-------------
-------------------------------------------------------------\n" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->ServiceType + " Vehicle (" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Type + ")\n" + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + " Model\nVehicle Rating: " + FloatToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->ComputeAndReturnRating());
```

```
                                       System::String^ st = gcnew
String(str.data());
                                       richTextBox1->Text = (st);
                                }
                                else {
                                       std::string str = TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n-------------
-------------------------------------------------------------\n\nThis Driver has no
vehicles yet";
                                       System::String^ st = gcnew
String(str.data());
                                       richTextBox1->Text = (st);
                                }
                          }
                   }
             }
             if (this->radioButton3->Checked) {
                   if (TMS_Main.Orders.size()) {
                          if (oID < TMS_Main.Orders.size() - 1) {
                                oID++;
                          }
                          while (TMS_Main.C_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getCID() && oID < TMS_Main.Orders.size() - 1) {
                                oID++;
                          }
                          if (oID >= TMS_Main.Orders.size() - 1 &&
TMS_Main.C_Accounts[i]->getID() != TMS_Main.Orders[oID]->getCID()) {
                                while (TMS_Main.C_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getCID() && oID > 0) {
                                       oID--;
                                }
                          }
                          if (TMS_Main.Orders.size()) {
                                int s = TMS_Main.Orders[oID]->getPlaced(), h, m;
                                std::string AmPm = "AM";
                                std::string s2;
                                h = (s / 3600) % 24 + 5;
                                if (h > 12) {
                                       h %= 12;
                                       AmPm = "PM";
                                }
                                s %= 3600;
                                m = s / 60;
                                s /= 60;
                                if (TMS_Main.Orders[oID]->getAccepted()) s2 =
"Accepted";
                                else s2 = "Not Accepted";
                                std::string str = "Order for a " +
TMS_Main.Orders[oID]->getType() + ", Placed On " + IntToString(h) + ":";
                                if (m < 10) str = str + "0";
                                str = str + IntToString(m) + AmPm + "\nDriver: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nStatus: " + s2;
                    System::String^ st2 = gcnew String(str.data());
                    this->richTextBox1->Text = (st2);
                }
            }
        }
    }
    private: System::Void button5_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (TMS_Main.D_Accounts[dID]->Vehicles.size() > 0) {
            if (vID > 0) {
                vID--;
            }
            std::string str = TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n-------------
-------------------------------------------------------------\n" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->ServiceType + " Vehicle (" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Type + ")\n" + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + " Model\nVehicle Rating: " + FloatToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->ComputeAndReturnRating());
            System::String^ st = gcnew String(str.data());
            richTextBox1->Text = (st);
        }
    }
    private: System::Void button6_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (TMS_Main.D_Accounts[dID]->Vehicles.size() > 0) {
            if (vID < TMS_Main.D_Accounts[dID]->Vehicles.size() - 1) {
                vID++;
            }
            std::string str = TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName() + "\nWork Experience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\n-------------
-------------------------------------------------------------\n" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->ServiceType + " Vehicle (" +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Type + ")\n" + TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]-
>Model.Year) + " Model\nVehicle Rating: " + FloatToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->ComputeAndReturnRating());
            System::String^ st = gcnew String(str.data());
            richTextBox1->Text = (st);
        }
    }
    private: System::Void button7_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            Distance^ f2 = gcnew Distance();
            f2->Show();
        }
        if (this->radioButton3->Checked) {
            RatingGiver^ f2 = gcnew RatingGiver();
            f2->Show();
        }
    }
    private: System::Void button8_Click(System::Object^ sender,
System::EventArgs^ e) {
        TMS_Main.SortR();
        dID = 0;
        this->radioButton2->Checked = false;
        this->radioButton2->Checked = true;
    }
    private: System::Void button9_Click(System::Object^ sender,
System::EventArgs^ e) {
        TMS_Main.SortE();
        dID = 0;
        this->radioButton2->Checked = false;
        this->radioButton2->Checked = true;
    }
    private: System::Void button10_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (TMS_Main.D_Accounts.size()) {
            std::ofstream Saver;
            Saver.open("Data/ServiceThread.txt");
            Saver << IntToString(dID);
            Driver_Service^ f2 = gcnew Driver_Service();
            f2->Show();
        }
        else {
            MessageBox::Show("There are no drivers signed up yet", "Error:
No Drivers", MessageBoxButtons::OK, MessageBoxIcon::Error);
            return;
        }
    }
    private: System::Void button11_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (TMS_Main.D_Accounts.size()) {
            if (TMS_Main.D_Accounts[dID]->Vehicles.size()) {
                Vehicle_Service^ f2 = gcnew Vehicle_Service();
                f2->Show();
            }
            else {
                MessageBox::Show("Selected Driver has no Vehicles",
"Error: No Vehicles", MessageBoxButtons::OK, MessageBoxIcon::Error);
                return;
            }
        }
        else {
            MessageBox::Show("There are no drivers signed up yet", "Error:
No Drivers", MessageBoxButtons::OK, MessageBoxIcon::Error);
            return;
        }
    }
    private: System::Void label1_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void checkBox1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox1->Checked) {
            this->checkBox2->Checked = false;
            this->checkBox1->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->checkBox1->BackColor = System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void checkBox2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox2->Checked) {
            this->checkBox1->Checked = false;
            this->checkBox2->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->checkBox2->BackColor = System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void Customer_Form_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {
        TMS_Main.SaveLoadedData();
        Application::Exit();
    }
#pragma endregion
    };
#pragma endregion
}
```

Distance Selector (Distance.h)

```
#pragma once
extern int distance;
int ai = 0, aj = 0, di = 0, dj = 0;
namespace TMS_Forms {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
#pragma region Distance Selector
    public ref class Distance : public System::Windows::Forms::Form {
    public:
        Distance(void) {
            InitializeComponent();
        }
    protected:
        ~Distance() {
            if (components) {
                delete components;
            }
        }
    private: System::Windows::Forms::RichTextBox^ richTextBox1;
    private: System::Windows::Forms::TrackBar^ trackBar1;
    private: System::Windows::Forms::TrackBar^ trackBar2;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::Button^ button2;
    private: System::Windows::Forms::CheckBox^ checkBox1;
    private: System::Windows::Forms::CheckBox^ checkBox2;
    private: System::ComponentModel::Container^ components;

#pragma region Component Code
          void InitializeComponent(void) {
              this->richTextBox1 = (gcnew 
System::Windows::Forms::RichTextBox());
              this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());
              this->trackBar2 = (gcnew System::Windows::Forms::TrackBar());
              this->button1 = (gcnew System::Windows::Forms::Button());
              this->button2 = (gcnew System::Windows::Forms::Button());
              this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
              this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))-
>BeginInit();

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))-
>BeginInit();
              this->SuspendLayout();
              // 
              // richTextBox1
              // 
              this->richTextBox1->Location = System::Drawing::Point(57, 
76);
              this->richTextBox1->Name = L"richTextBox1";
```

```
                      this->richTextBox1->ReadOnly = true;
                      this->richTextBox1->Size = System::Drawing::Size(384, 299);
                      this->richTextBox1->TabIndex = 0;
                      this->richTextBox1->Text = L"";
                      // 
                      // trackBar1
                      // 
                      this->trackBar1->BackColor = 
System::Drawing::SystemColors::Control;
                      this->trackBar1->Location = System::Drawing::Point(45, 377);
                      this->trackBar1->Maximum = 135;
                      this->trackBar1->Name = L"trackBar1";
                      this->trackBar1->Size = System::Drawing::Size(406, 45);
                      this->trackBar1->TabIndex = 1;
                      this->trackBar1->TickFrequency = 0;
                      this->trackBar1->TickStyle = 
System::Windows::Forms::TickStyle::TopLeft;
                      this->trackBar1->Scroll += gcnew System::EventHandler(this, 
&Distance::trackBar1_Scroll);
                      // 
                      // trackBar2
                      // 
                      this->trackBar2->AllowDrop = true;
                      this->trackBar2->BackColor = 
System::Drawing::SystemColors::Control;
                      this->trackBar2->Location = System::Drawing::Point(13, 66);
                      this->trackBar2->Maximum = 50;
                      this->trackBar2->Name = L"trackBar2";
                      this->trackBar2->Orientation = 
System::Windows::Forms::Orientation::Vertical;
                      this->trackBar2->Size = System::Drawing::Size(45, 318);
                      this->trackBar2->TabIndex = 2;
                      this->trackBar2->TickFrequency = 0;
                      this->trackBar2->Scroll += gcnew System::EventHandler(this, 
&Distance::trackBar2_Scroll);
                      // 
                      // button1
                      // 
                      this->button1->BackColor = System::Drawing::Color::LawnGreen;
                      this->button1->FlatStyle = 
System::Windows::Forms::FlatStyle::Popup;
                      this->button1->ForeColor = 
System::Drawing::Color::DarkSlateGray;
                      this->button1->Location = System::Drawing::Point(58, 354);
                      this->button1->Name = L"button1";
                      this->button1->Size = System::Drawing::Size(20, 20);
                      this->button1->TabIndex = 3;
                      this->button1->Text = L"D";
                      this->button1->UseVisualStyleBackColor = false;
                      // 
                      // button2
                      // 
                      this->button2->BackColor = 
System::Drawing::Color::DeepSkyBlue;
                      this->button2->FlatStyle = 
System::Windows::Forms::FlatStyle::Popup;
                      this->button2->ForeColor = System::Drawing::Color::DarkGreen;
                      this->button2->Location = System::Drawing::Point(58, 354);
                      this->button2->Name = L"button2";
                      this->button2->Size = System::Drawing::Size(20, 20);
                      this->button2->TabIndex = 4;
                      this->button2->Text = L"A";
                      this->button2->UseVisualStyleBackColor = false;
                      // 
                      // checkBox1
                      // 
                      this->checkBox1->Appearance = 
System::Windows::Forms::Appearance::Button;
                      this->checkBox1->AutoSize = true;
                      this->checkBox1->FlatStyle = 
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox1->Location = System::Drawing::Point(59, 54);
                      this->checkBox1->Name = L"checkBox1";
                      this->checkBox1->Size = System::Drawing::Size(141, 23);
                      this->checkBox1->TabIndex = 5;
                      this->checkBox1->Text = L"Select Departure Location";
                      this->checkBox1->UseVisualStyleBackColor = true;
                      this->checkBox1->CheckedChanged += gcnew 
System::EventHandler(this, &Distance::checkBox1_CheckedChanged);
                      // 
                      // checkBox2
                      // 
                      this->checkBox2->Appearance = 
System::Windows::Forms::Appearance::Button;
                      this->checkBox2->AutoSize = true;
                      this->checkBox2->FlatStyle = 
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox2->Location = System::Drawing::Point(200, 54);
                      this->checkBox2->Name = L"checkBox2";
                      this->checkBox2->Size = System::Drawing::Size(123, 23);
                      this->checkBox2->TabIndex = 6;
                      this->checkBox2->Text = L"Select Arrival Location";
                      this->checkBox2->UseVisualStyleBackColor = true;
                      this->checkBox2->CheckedChanged += gcnew 
System::EventHandler(this, &Distance::checkBox2_CheckedChanged);
                      // 
                      // Distance
                      // 
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode = 
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(454, 420);
                      this->Controls->Add(this->checkBox2);
                      this->Controls->Add(this->checkBox1);
                      this->Controls->Add(this->button2);
                      this->Controls->Add(this->button1);
                      this->Controls->Add(this->trackBar2);
                      this->Controls->Add(this->trackBar1);
                      this->Controls->Add(this->richTextBox1);
                      this->Name = L"Distance";
                      this->Text = L"Distance Selector: Transport Manager+";
                      this->FormClosing += gcnew 
System::Windows::Forms::FormClosingEventHandler(this, 
&Distance::Distance_FormClosing);
                      this->Load += gcnew System::EventHandler(this, 
&Distance::Distance_Load);
```

```
(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))-
>EndInit();

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))-
>EndInit();
              this->ResumeLayout(false);
              this->PerformLayout();

          }
#pragma endregion
#pragma region Function Code
    private: System::Void Distance_Load(System::Object^ sender,
System::EventArgs^ e) {
        this->button1->Location = System::Drawing::Point(58 + ai * 2.67592, 354
- aj * 5.54);
        this->button2->Location = System::Drawing::Point(58 + di * 2.67592, 354
- dj * 5.54);
        this->checkBox1->Checked = true;
    }
    private: System::Void trackBar2_Scroll(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox1->Checked) {
            aj = this->trackBar2->Value;
            this->button1->Location = System::Drawing::Point(58 + ai *
2.67592, 354 - aj * 5.54);
        }
        else {
            dj = this->trackBar2->Value;
            this->button2->Location = System::Drawing::Point(58 + di *
2.67592, 354 - dj * 5.54);
        }
    }
    private: System::Void trackBar1_Scroll(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox1->Checked) {
            ai = this->trackBar1->Value;
            this->button1->Location = System::Drawing::Point(58 + ai *
2.67592, 354 - aj * 5.54);
        }
        else {
            di = this->trackBar1->Value;
            this->button2->Location = System::Drawing::Point(58 + di *
2.67592, 354 - dj * 5.54);
        }
    }
    private: System::Void label1_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void checkBox1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox1->Checked) {
            this->checkBox2->Checked = false;
            this->checkBox1->BackColor =
System::Drawing::SystemColors::ControlDark;
            this->trackBar1->Value = ai;
            this->trackBar2->Value = aj;
        }
```

```
		else {
			this->checkBox1->BackColor = System::Drawing::Color::WhiteSmoke;
		}
	}
	private: System::Void checkBox2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
		if (this->checkBox2->Checked) {
			this->checkBox1->Checked = false;
			this->checkBox2->BackColor =
System::Drawing::SystemColors::ControlDark;
			this->trackBar1->Value = di;
			this->trackBar2->Value = dj;
		}
		else {
			this->checkBox2->BackColor = System::Drawing::Color::WhiteSmoke;
		}
	}
	private: System::Void Distance_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {
		disx = Math::Sqrt(((ai - di) * (ai - di)) + ((aj - dj) * (aj - dj)));
	}
#pragma endregion
	};
#pragma endregion
}
```

Rating Selector (RatingGiver.h)

```
#pragma once
extern float r1, r2;
namespace TMS_Forms {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
#pragma region Rating Form
    public ref class RatingGiver : public System::Windows::Forms::Form {
    public:
        RatingGiver(void) {
            InitializeComponent();
        }
    protected:
        ~RatingGiver() {
            if (components) {
                delete components;
            }
        }
    private: System::Windows::Forms::Label^ label1;
    protected:
    private: System::Windows::Forms::TrackBar^ trackBar1;
    private: System::Windows::Forms::TrackBar^ trackBar2;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::RichTextBox^ richTextBox1;
    private: System::Windows::Forms::RichTextBox^ richTextBox2;
    private: System::ComponentModel::Container^ components;
#pragma region Componenet Code
          void InitializeComponent(void)
          {
              this->label1 = (gcnew System::Windows::Forms::Label());
              this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());
              this->trackBar2 = (gcnew System::Windows::Forms::TrackBar());
              this->label2 = (gcnew System::Windows::Forms::Label());
              this->label3 = (gcnew System::Windows::Forms::Label());
              this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
              this->richTextBox2 = (gcnew
System::Windows::Forms::RichTextBox());

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))-
>BeginInit();

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))-
>BeginInit();
              this->SuspendLayout();
              //
              // label1
              //
              this->label1->AutoSize = true;
```

```
                       this->label1->Font = (gcnew System::Drawing::Font(L"Modern
No. 20", 15.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label1->Location = System::Drawing::Point(12, 9);
                       this->label1->Name = L"label1";
                       this->label1->Size = System::Drawing::Size(379, 24);
                       this->label1->TabIndex = 0;
                       this->label1->Text = L"Please Rate the Driver and the
Vehicle";
                       //
                       // trackBar1
                       //
                       this->trackBar1->Location = System::Drawing::Point(118, 56);
                       this->trackBar1->Name = L"trackBar1";
                       this->trackBar1->Size = System::Drawing::Size(179, 45);
                       this->trackBar1->TabIndex = 1;
                       this->trackBar1->Scroll += gcnew System::EventHandler(this,
&RatingGiver::trackBar1_Scroll);
                       //
                       // trackBar2
                       //
                       this->trackBar2->Location = System::Drawing::Point(118, 117);
                       this->trackBar2->Name = L"trackBar2";
                       this->trackBar2->Size = System::Drawing::Size(179, 45);
                       this->trackBar2->TabIndex = 2;
                       this->trackBar2->Scroll += gcnew System::EventHandler(this,
&RatingGiver::trackBar2_Scroll);
                       //
                       // label2
                       //
                       this->label2->AutoSize = true;
                       this->label2->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label2->Location = System::Drawing::Point(0, 56);
                       this->label2->Name = L"label2";
                       this->label2->Size = System::Drawing::Size(117, 26);
                       this->label2->TabIndex = 3;
                       this->label2->Text = L"Rate Driver";
                       this->label2->Click += gcnew System::EventHandler(this,
&RatingGiver::label2_Click);
                       //
                       // label3
                       //
                       this->label3->AutoSize = true;
                       this->label3->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label3->Location = System::Drawing::Point(0, 117);
                       this->label3->Name = L"label3";
                       this->label3->Size = System::Drawing::Size(124, 26);
                       this->label3->TabIndex = 4;
                       this->label3->Text = L"Rate Vehicle";
                       //
                       // richTextBox1
```

```
                      //
                      this->richTextBox1->Font = (gcnew
System::Drawing::Font(L"Trebuchet MS", 20.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->richTextBox1->Location = System::Drawing::Point(323,
56);
                      this->richTextBox1->Name = L"richTextBox1";
                      this->richTextBox1->ReadOnly = true;
                      this->richTextBox1->Size = System::Drawing::Size(68, 45);
                      this->richTextBox1->TabIndex = 5;
                      this->richTextBox1->Text = L"0.0";
                      //
                      // richTextBox2
                      //
                      this->richTextBox2->Font = (gcnew
System::Drawing::Font(L"Trebuchet MS", 20.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->richTextBox2->Location = System::Drawing::Point(323,
117);
                      this->richTextBox2->Name = L"richTextBox2";
                      this->richTextBox2->ReadOnly = true;
                      this->richTextBox2->Size = System::Drawing::Size(68, 45);
                      this->richTextBox2->TabIndex = 6;
                      this->richTextBox2->Text = L"0.0";
                      //
                      // RatingGiver
                      //
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(409, 193);
                      this->Controls->Add(this->richTextBox2);
                      this->Controls->Add(this->richTextBox1);
                      this->Controls->Add(this->label3);
                      this->Controls->Add(this->label2);
                      this->Controls->Add(this->trackBar2);
                      this->Controls->Add(this->trackBar1);
                      this->Controls->Add(this->label1);
                      this->Name = L"RatingGiver";
                      this->Text = L"Driver Rating: Transport Manager+";
                      this->Load += gcnew System::EventHandler(this,
&RatingGiver::RatingGiver_Load);

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))-
>EndInit();

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))-
>EndInit();
                      this->ResumeLayout(false);
                      this->PerformLayout();

                }
#pragma endregion
#pragma region Function Code
      private: System::Void trackBar1_Scroll(System::Object^ sender,
System::EventArgs^ e) {
        r1 = this->trackBar1->Value / 2.0;
        std::string str = FloatToString(r1);
        System::String^ st2 = gcnew String(str.data());
        this->richTextBox1->Text = (st2);
    }
    private: System::Void trackBar2_Scroll(System::Object^ sender,
System::EventArgs^ e) {
        r2 = this->trackBar2->Value / 2.0;
        std::string str = FloatToString(r2);
        System::String^ st2 = gcnew String(str.data());
        this->richTextBox2->Text = (st2);
    }
    private: System::Void label2_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void RatingGiver_Load(System::Object^ sender,
System::EventArgs^ e) {
    }
#pragma endregion
    };
#pragma endregion
}
```

Driver Service History (Driver_Service.h)

```
#pragma once
extern TMS TMS_Main;
extern int dID;
namespace TMS_Forms {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    /// <summary>
    /// Summary for Driver_Service
    /// </summary>
    public ref class Driver_Service : public System::Windows::Forms::Form {
    public:
        Driver_Service(void) {
            InitializeComponent();
        }
    protected:
        ~Driver_Service() {
            if (components) {
                delete components;
            }
        }
    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::RichTextBox^ richTextBox1;
    private: System::Windows::Forms::Label^ label3;
    private: System::ComponentModel::Container^ components;
#pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
            this->label1->Location = System::Drawing::Point(12, 96);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(66, 26);
            this->label1->TabIndex = 1;
            this->label1->Text = L"label1";
            //
            // label2
            //
```

```
                    this->label2->AutoSize = true;
                    this->label2->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label2->Location = System::Drawing::Point(12, 122);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(54, 22);
                    this->label2->TabIndex = 3;
                    this->label2->Text = L"label2";
                    // 
                    // richTextBox1
                    // 
                    this->richTextBox1->BackColor =
System::Drawing::SystemColors::ControlLight;
                    this->richTextBox1->Font = (gcnew
System::Drawing::Font(L"Rockwell", 18, System::Drawing::FontStyle::Bold));
                    this->richTextBox1->Location = System::Drawing::Point(12, 147);
                    this->richTextBox1->Name = L"richTextBox1";
                    this->richTextBox1->ReadOnly = true;
                    this->richTextBox1->Size = System::Drawing::Size(526, 323);
                    this->richTextBox1->TabIndex = 4;
                    this->richTextBox1->Text = L"";
                    // 
                    // label3
                    // 
                    this->label3->AutoSize = true;
                    this->label3->Font = (gcnew System::Drawing::Font(L"Rockwell
Condensed", 36, System::Drawing::FontStyle::Bold));
                    this->label3->Location = System::Drawing::Point(54, 9);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(442, 56);
                    this->label3->TabIndex = 5;
                    this->label3->Text = L"Driver Service History";
                    // 
                    // Driver_Service
                    // 
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(550, 482);
                    this->Controls->Add(this->label3);
                    this->Controls->Add(this->richTextBox1);
                    this->Controls->Add(this->label2);
                    this->Controls->Add(this->label1);
                    this->Name = L"Driver_Service";
                    this->Text = L"Driver_Service";
                    this->Load += gcnew System::EventHandler(this,
&Driver_Service::Driver_Service_Load);
                    this->ResumeLayout(false);
                    this->PerformLayout();

             }
#pragma endregion
      private: System::Void Driver_Service_Load(System::Object^ sender,
System::EventArgs^ e) {
             int c = 0;
             std::string s1, s2, s3, s4, txt;
```

```
        std::ifstream Loader;
        String^ name, ^ balance, ^ cnic, ^ rating;
        s1 = TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName();
        name = gcnew String(s1.data());
        s1 = TMS_Main.D_Accounts[dID]->getID();
        cnic = gcnew String(s1.data());
        this->label1->Text = name;
        this->label2->Text = cnic;
        std::ifstream Services;
        Services.open("Data/Past_Services.txt");
        this->richTextBox1->Text = "\n\nThis Driver has no past services";
        while (!Services.eof()) {
            s1 = " ";
            s2 = " ";
            s3 = " ";
            s4 = " ";
            Services >> s1 >> s2 >> s3 >> s4;
            if (s2 == TMS_Main.D_Accounts[dID]->getID()) {
                c++;
                for (int i = 0; i < s4.length(); i++) {
                    if (s4[i] == '%') s4[i] = '\n';
                    if (s4[i] == '*') s4[i] = ' ';
                }
                txt = txt + "Order for a " + s1 + "\n" + s4 + "\n---------
--------------------------------------------------------\n";
            }
        }
        for (int i = 0; i < TMS_Main.Orders.size(); i++) {
            if (TMS_Main.Orders[i]->getDID() == TMS_Main.D_Accounts[dID]-
>getID() && !TMS_Main.Orders[i]->getAccepted()) {
                txt = txt + "Order for a " + TMS_Main.Orders[i]->getType()
+ "\nRequested by: " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[i]-
>getCID())]->getFName() + " " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[i]->getCID())]->getLName() +
"\nOrder Status: Not Accepted\n-----------------------------------------------------
------------\n";
                c++;
            }
        }
        if (c > 0) {
            name = gcnew String(txt.data());
            this->richTextBox1->Text = name;
        }
    }
    };
}
```

Vehicle Service History (Vehicle_Service.h)

```
#pragma once
extern TMS TMS_Main;
extern int dID, vID;
namespace TMS_Forms {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    /// <summary>
    /// Summary for Vehicle_Service
    /// </summary>
    public ref class Vehicle_Service : public System::Windows::Forms::Form
    {
    public:
        Vehicle_Service(void)
        {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~Vehicle_Service()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::RichTextBox^ richTextBox1;
    protected:
    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Label^ label5;

    private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container^ components;

#pragma 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>
```

```
             void InitializeComponent(void)
             {
                    this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
                    this->label1 = (gcnew System::Windows::Forms::Label());
                    this->label2 = (gcnew System::Windows::Forms::Label());
                    this->label3 = (gcnew System::Windows::Forms::Label());
                    this->label4 = (gcnew System::Windows::Forms::Label());
                    this->label5 = (gcnew System::Windows::Forms::Label());
                    this->SuspendLayout();
                    // 
                    // richTextBox1
                    // 
                    this->richTextBox1->BackColor =
System::Drawing::SystemColors::ControlLight;
                    this->richTextBox1->Font = (gcnew
System::Drawing::Font(L"Rockwell", 18, System::Drawing::FontStyle::Bold));
                    this->richTextBox1->Location = System::Drawing::Point(12, 161);
                    this->richTextBox1->Name = L"richTextBox1";
                    this->richTextBox1->ReadOnly = true;
                    this->richTextBox1->Size = System::Drawing::Size(526, 323);
                    this->richTextBox1->TabIndex = 4;
                    this->richTextBox1->Text = L"";
                    // 
                    // label1
                    // 
                    this->label1->AutoSize = true;
                    this->label1->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label1->Location = System::Drawing::Point(291, 101);
                    this->label1->Name = L"label1";
                    this->label1->Size = System::Drawing::Size(66, 26);
                    this->label1->TabIndex = 1;
                    this->label1->Text = L"label1";
                    // 
                    // label2
                    // 
                    this->label2->AutoSize = true;
                    this->label2->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label2->Location = System::Drawing::Point(292, 127);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(54, 22);
                    this->label2->TabIndex = 3;
                    this->label2->Text = L"label2";
                    // 
                    // label3
                    // 
                    this->label3->AutoSize = true;
                    this->label3->Font = (gcnew System::Drawing::Font(L"Rockwell
Condensed", 36, System::Drawing::FontStyle::Bold));
                    this->label3->Location = System::Drawing::Point(41, 9);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(467, 56);
```

```
            this->label3->TabIndex = 5;
            this->label3->Text = L"Vehicle Service History";
            this->label3->Click += gcnew System::EventHandler(this,
&Vehicle_Service::label3_Click);
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold));
            this->label4->Location = System::Drawing::Point(7, 101);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(66, 26);
            this->label4->TabIndex = 4;
            this->label4->Text = L"label4";
            //
            // label5
            //
            this->label5->AutoSize = true;
            this->label5->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 12, System::Drawing::FontStyle::Bold));
            this->label5->Location = System::Drawing::Point(8, 127);
            this->label5->Name = L"label5";
            this->label5->Size = System::Drawing::Size(54, 22);
            this->label5->TabIndex = 5;
            this->label5->Text = L"label5";
            //
            // Vehicle_Service
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(550, 497);
            this->Controls->Add(this->label5);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->richTextBox1);
            this->Name = L"Vehicle_Service";
            this->Text = L"Vehicle_Service";
            this->Load += gcnew System::EventHandler(this,
&Vehicle_Service::Vehicle_Service_Load);
            this->ResumeLayout(false);
            this->PerformLayout();

        }
#pragma endregion
    private: System::Void label3_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void Vehicle_Service_Load(System::Object^ sender,
System::EventArgs^ e) {
        int c = 0;
        std::string s1, s2, s3, s4, txt;
        std::ifstream Loader;
        String^ name, ^ vehicle, ^ cnic, ^ lcplt;
        s1 = "Owner: " + TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName();
        name = gcnew String(s1.data());
        s1 = TMS_Main.D_Accounts[dID]->getID();
        cnic = gcnew String(s1.data());
        s1 = TMS_Main.D_Accounts[dID]->Vehicles[vID]->Model.Company + " " +
TMS_Main.D_Accounts[dID]->Vehicles[vID]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[dID]->Vehicles[vID]->Model.Year) + " Model";
        vehicle = gcnew String(s1.data());
        s1 = "Lisnece Plate Number: " + IntToString(TMS_Main.D_Accounts[dID]-
>Vehicles[vID]->getID());
        lcplt = gcnew String(s1.data());
        this->label1->Text = name;
        this->label2->Text = cnic;
        this->label4->Text = vehicle;
        this->label5->Text = lcplt;
        std::ifstream Services;
        Services.open("Data/Past_Services.txt");
        this->richTextBox1->Text = "\n\nThis Vehicle has no past services";
        while (!Services.eof()) {
            s1 = " ";
            s2 = " ";
            s3 = " ";
            s4 = " ";
            Services >> s1 >> s2 >> s3 >> s4;
            if (s2 == TMS_Main.D_Accounts[dID]->getID() &&
TMS_Main.D_Accounts[dID]->Vehicles[vID]->getID() == StringToInt(s3)) {
                c++;
                for (int i = 0; i < s4.length(); i++) {
                    if (s4[i] == '%') s4[i] = '\n';
                    if (s4[i] == '*') s4[i] = ' ';
                }
                txt = txt + "Order for a " + s1 + "\n" + s4 + "\n---------
--------------------------------------------------------\n";
            }
        }
        for (int i = 0; i < TMS_Main.Orders.size(); i++) {
            if (TMS_Main.Orders[i]->getDID() == TMS_Main.D_Accounts[dID]-
>getID() && TMS_Main.D_Accounts[dID]->Vehicles[vID]->getID() == TMS_Main.Orders[i]-
>getVID() && !TMS_Main.Orders[i]->getAccepted()) {
                txt = txt + "Order for a " + TMS_Main.Orders[i]->getType()
+ "\nRequested by: " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[i]-
>getCID())]->getFName() + " " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[i]->getCID())]->getLName() +
"\nOrder Status: Not Accepted\n-----------------------------------------------------
------------\n";
                c++;
            }
        }
        if (c > 0) {
            name = gcnew String(txt.data());
            this->richTextBox1->Text = name;
        }
    }
    };
}
```

Driver Page (Driver_Form.h)

```
#pragma once
extern TMS TMS_Main;
extern int oID = 0, vID = 0;
namespace TMS_Forms {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
#pragma region Driver Form
    public ref class Driver_Form : public System::Windows::Forms::Form {
    public:
        Driver_Form(void) {
            InitializeComponent();
        }

    protected:
        ~Driver_Form() {
            if (components) {
                delete components;
            }
        }
    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::RadioButton^ radioButton1;
    private: System::Windows::Forms::RadioButton^ radioButton2;
    private: System::Windows::Forms::RadioButton^ radioButton3;
    private: System::Windows::Forms::RichTextBox^ richTextBox1;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::Button^ button2;
    private: System::Windows::Forms::Button^ button3;
    private: System::Windows::Forms::Button^ button4;
    private: System::Windows::Forms::Label^ label5;
    private: System::Windows::Forms::TextBox^ textBox1;
    private: System::Windows::Forms::Label^ label6;
    private: System::Windows::Forms::RadioButton^ radioButton4;
    private: System::Windows::Forms::TextBox^ textBox2;
    private: System::Windows::Forms::TextBox^ textBox3;
    private: System::Windows::Forms::TextBox^ textBox4;
    private: System::Windows::Forms::TextBox^ textBox5;
    private: System::Windows::Forms::Label^ label7;
    private: System::Windows::Forms::Label^ label8;
    private: System::Windows::Forms::Label^ label9;
    private: System::Windows::Forms::Label^ label10;
    private: System::Windows::Forms::Button^ button5;
    private: System::Windows::Forms::CheckBox^ checkBox1;
    private: System::Windows::Forms::CheckBox^ checkBox2;
    private: System::Windows::Forms::CheckBox^ checkBox3;
    private: System::Windows::Forms::CheckBox^ checkBox4;
    private: System::Windows::Forms::Label^ label11;
    private: System::Windows::Forms::Label^ label12;
```

```
      private: System::Windows::Forms::Label^ label13;
      private: System::Windows::Forms::Label^ label14;

      private: System::ComponentModel::Container^ components;
#pragma region Component Code
                void InitializeComponent(void)
                {
                      System::ComponentModel::ComponentResourceManager^ resources =
(gcnew System::ComponentModel::ComponentResourceManager(Driver_Form::typeid));
                      this->label1 = (gcnew System::Windows::Forms::Label());
                      this->label2 = (gcnew System::Windows::Forms::Label());
                      this->label3 = (gcnew System::Windows::Forms::Label());
                      this->radioButton1 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton2 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton3 = (gcnew
System::Windows::Forms::RadioButton());
                      this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
                      this->label4 = (gcnew System::Windows::Forms::Label());
                      this->button1 = (gcnew System::Windows::Forms::Button());
                      this->button2 = (gcnew System::Windows::Forms::Button());
                      this->button3 = (gcnew System::Windows::Forms::Button());
                      this->button4 = (gcnew System::Windows::Forms::Button());
                      this->label5 = (gcnew System::Windows::Forms::Label());
                      this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                      this->label6 = (gcnew System::Windows::Forms::Label());
                      this->radioButton4 = (gcnew
System::Windows::Forms::RadioButton());
                      this->textBox2 = (gcnew System::Windows::Forms::TextBox());
                      this->textBox3 = (gcnew System::Windows::Forms::TextBox());
                      this->textBox4 = (gcnew System::Windows::Forms::TextBox());
                      this->textBox5 = (gcnew System::Windows::Forms::TextBox());
                      this->label7 = (gcnew System::Windows::Forms::Label());
                      this->label8 = (gcnew System::Windows::Forms::Label());
                      this->label9 = (gcnew System::Windows::Forms::Label());
                      this->label10 = (gcnew System::Windows::Forms::Label());
                      this->button5 = (gcnew System::Windows::Forms::Button());
                      this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
                      this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
                      this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());
                      this->checkBox4 = (gcnew System::Windows::Forms::CheckBox());
                      this->label11 = (gcnew System::Windows::Forms::Label());
                      this->label12 = (gcnew System::Windows::Forms::Label());
                      this->label13 = (gcnew System::Windows::Forms::Label());
                      this->label14 = (gcnew System::Windows::Forms::Label());
                      this->SuspendLayout();
                      //
                      // label1
                      //
                      this->label1->AutoSize = true;
                      this->label1->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 14.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label1->Location = System::Drawing::Point(12, 19);
                      this->label1->Name = L"label1";
                      this->label1->Size = System::Drawing::Size(66, 26);
                      this->label1->TabIndex = 0;
                      this->label1->Text = L"label1";
                      this->label1->Click += gcnew System::EventHandler(this,
&Driver_Form::label1_Click);
                      //
                      // label2
                      //
                      this->label2->AutoSize = true;
                      this->label2->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label2->Location = System::Drawing::Point(299, 15);
                      this->label2->Name = L"label2";
                      this->label2->Size = System::Drawing::Size(52, 21);
                      this->label2->TabIndex = 1;
                      this->label2->Text = L"label2";
                      this->label2->Click += gcnew System::EventHandler(this,
&Driver_Form::label2_Click);
                      //
                      // label3
                      //
                      this->label3->AutoSize = true;
                      this->label3->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label3->Location = System::Drawing::Point(12, 48);
                      this->label3->Name = L"label3";
                      this->label3->Size = System::Drawing::Size(54, 22);
                      this->label3->TabIndex = 2;
                      this->label3->Text = L"label3";
                      this->label3->Click += gcnew System::EventHandler(this,
&Driver_Form::label3_Click);
                      //
                      // radioButton1
                      //
                      this->radioButton1->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton1->AutoSize = true;
                      this->radioButton1->BackColor =
System::Drawing::Color::WhiteSmoke;
                      this->radioButton1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton1->Location = System::Drawing::Point(16,
131);
                      this->radioButton1->Name = L"radioButton1";
                      this->radioButton1->Size = System::Drawing::Size(100, 49);
                      this->radioButton1->TabIndex = 3;
                      this->radioButton1->TabStop = true;
                      this->radioButton1->Text = L"\nWithdraw Money \n ";
                      this->radioButton1->UseVisualStyleBackColor = false;
                      this->radioButton1->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::radioButton1_CheckedChanged);
                      //
                      // radioButton2
                      //
                      this->radioButton2->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton2->AutoSize = true;
                      this->radioButton2->BackColor =
System::Drawing::Color::WhiteSmoke;
                      this->radioButton2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton2->Location = System::Drawing::Point(16,
186);
                      this->radioButton2->Name = L"radioButton2";
                      this->radioButton2->Size = System::Drawing::Size(101, 49);
                      this->radioButton2->TabIndex = 4;
                      this->radioButton2->TabStop = true;
                      this->radioButton2->Text = L"\n     View Orders    \n ";
                      this->radioButton2->UseVisualStyleBackColor = false;
                      this->radioButton2->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::radioButton2_CheckedChanged);
                      //
                      // radioButton3
                      //
                      this->radioButton3->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton3->AutoSize = true;
                      this->radioButton3->BackColor =
System::Drawing::Color::WhiteSmoke;
                      this->radioButton3->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton3->Location = System::Drawing::Point(16,
241);
                      this->radioButton3->Name = L"radioButton3";
                      this->radioButton3->Size = System::Drawing::Size(102, 49);
                      this->radioButton3->TabIndex = 5;
                      this->radioButton3->TabStop = true;
                      this->radioButton3->Text = L"\n       Vehicles        \n ";
                      this->radioButton3->UseVisualStyleBackColor = false;
                      this->radioButton3->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::radioButton3_CheckedChanged);
                      //
                      // richTextBox1
                      //
                      this->richTextBox1->BackColor =
System::Drawing::SystemColors::MenuBar;
                      this->richTextBox1->Font = (gcnew
System::Drawing::Font(L"Palatino Linotype", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->richTextBox1->Location = System::Drawing::Point(211,
104);
                      this->richTextBox1->Name = L"richTextBox1";
                      this->richTextBox1->Size = System::Drawing::Size(304, 131);
                      this->richTextBox1->TabIndex = 16;
                      this->richTextBox1->Text = L"";
                      //
                      // label4
                      //
                      this->label4->AutoSize = true;
                      this->label4->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label4->Location = System::Drawing::Point(299, 48);
                      this->label4->Name = L"label4";
                      this->label4->Size = System::Drawing::Size(52, 21);
                      this->label4->TabIndex = 7;
                      this->label4->Text = L"label4";
                      //
                      // button1
                      //
                      this->button1->BackColor =
System::Drawing::Color::Transparent;
                      this->button1->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Zoom;
                      this->button1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button1->Font = (gcnew System::Drawing::Font(L"Arial
Rounded MT Bold", 26.25F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button1->Location = System::Drawing::Point(211, 241);
                      this->button1->Name = L"button1";
                      this->button1->Size = System::Drawing::Size(55, 49);
                      this->button1->TabIndex = 10;
                      this->button1->Text = L"<";
                      this->button1->UseVisualStyleBackColor = false;
                      this->button1->Click += gcnew System::EventHandler(this,
&Driver_Form::button1_Click);
                      //
                      // button2
                      //
                      this->button2->BackColor =
System::Drawing::SystemColors::ControlDark;
                      this->button2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button2->Location = System::Drawing::Point(272, 241);
                      this->button2->Name = L"button2";
                      this->button2->Size = System::Drawing::Size(89, 49);
                      this->button2->TabIndex = 11;
                      this->button2->Text = L"Reject Order";
                      this->button2->UseVisualStyleBackColor = false;
                      this->button2->Click += gcnew System::EventHandler(this,
&Driver_Form::button2_Click);
                      //
                      // button3
                      //
                      this->button3->BackColor =
System::Drawing::SystemColors::ControlDark;
                      this->button3->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button3->Location = System::Drawing::Point(367, 241);
                      this->button3->Name = L"button3";
                      this->button3->Size = System::Drawing::Size(87, 49);
                      this->button3->TabIndex = 12;
                      this->button3->Text = L"Accept Order";
                      this->button3->UseVisualStyleBackColor = false;
                      this->button3->Click += gcnew System::EventHandler(this,
&Driver_Form::button3_Click);
                      //
                      // button4
                      //
                      this->button4->BackColor =
System::Drawing::Color::Transparent;
                      this->button4->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Zoom;
                      this->button4->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button4->Font = (gcnew System::Drawing::Font(L"Arial
Rounded MT Bold", 26.25F, System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button4->Location = System::Drawing::Point(460, 241);
                      this->button4->Name = L"button4";
                      this->button4->Size = System::Drawing::Size(55, 49);
                      this->button4->TabIndex = 13;
                      this->button4->Text = L">";
                      this->button4->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                      this->button4->UseVisualStyleBackColor = false;
                      this->button4->Click += gcnew System::EventHandler(this,
&Driver_Form::button4_Click);
                      //
                      // label5
                      //
                      this->label5->AutoSize = true;
                      this->label5->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 9, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label5->Location = System::Drawing::Point(208, 114);
                      this->label5->Name = L"label5";
                      this->label5->Size = System::Drawing::Size(0, 17);
                      this->label5->TabIndex = 8;
                      //
                      // textBox1
                      //
                      this->textBox1->Location = System::Drawing::Point(211, 131);
                      this->textBox1->Name = L"textBox1";
                      this->textBox1->Size = System::Drawing::Size(199, 20);
                      this->textBox1->TabIndex = 7;
                      this->textBox1->TextChanged += gcnew
System::EventHandler(this, &Driver_Form::textBox1_TextChanged);
                      //
                      // label6
                      //
                      this->label6->AutoSize = true;
                      this->label6->Location = System::Drawing::Point(212, 151);
                      this->label6->Name = L"label6";
                      this->label6->Size = System::Drawing::Size(10, 13);
                      this->label6->TabIndex = 9;
                      this->label6->Text = L" ";
                      //
                      // radioButton4
                      //
                      this->radioButton4->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton4->AutoSize = true;
                      this->radioButton4->BackColor =
System::Drawing::Color::WhiteSmoke;
                      this->radioButton4->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton4->Location = System::Drawing::Point(16,
296);
                      this->radioButton4->Name = L"radioButton4";
                      this->radioButton4->Size = System::Drawing::Size(100, 49);
                      this->radioButton4->TabIndex = 17;
                      this->radioButton4->TabStop = true;
                      this->radioButton4->Text = L"\n  Delete Account \n ";
                      this->radioButton4->UseVisualStyleBackColor = false;
                      this->radioButton4->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::radioButton4_CheckedChanged);
                      //
                      // textBox2
                      //
                      this->textBox2->Location = System::Drawing::Point(623, 41);
                      this->textBox2->Name = L"textBox2";
                      this->textBox2->Size = System::Drawing::Size(215, 20);
                      this->textBox2->TabIndex = 18;
                      this->textBox2->TextChanged += gcnew
System::EventHandler(this, &Driver_Form::textBox2_TextChanged);
                      //
                      // textBox3
                      //
                      this->textBox3->Location = System::Drawing::Point(623, 111);
                      this->textBox3->Name = L"textBox3";
                      this->textBox3->Size = System::Drawing::Size(215, 20);
                      this->textBox3->TabIndex = 19;
                      this->textBox3->TextChanged += gcnew
System::EventHandler(this, &Driver_Form::textBox3_TextChanged);
                      //
                      // textBox4
                      //
                      this->textBox4->Location = System::Drawing::Point(623, 173);
                      this->textBox4->Name = L"textBox4";
                      this->textBox4->Size = System::Drawing::Size(215, 20);
                      this->textBox4->TabIndex = 20;
                      this->textBox4->TextChanged += gcnew
System::EventHandler(this, &Driver_Form::textBox4_TextChanged);
                      //
                      // textBox5
                      //
                      this->textBox5->Location = System::Drawing::Point(623, 236);
                      this->textBox5->Name = L"textBox5";
                      this->textBox5->Size = System::Drawing::Size(215, 20);
                      this->textBox5->TabIndex = 21;
                      this->textBox5->TextChanged += gcnew
System::EventHandler(this, &Driver_Form::textBox5_TextChanged);
                      //
                      // label7
                      //
                      this->label7->AutoSize = true;
```

```
                       this->label7->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label7->Location = System::Drawing::Point(619, 20);
                       this->label7->Name = L"label7";
                       this->label7->Size = System::Drawing::Size(182, 21);
                       this->label7->TabIndex = 22;
                       this->label7->Text = L"Vehicle Brand/Company";
                       this->label7->Click += gcnew System::EventHandler(this,
&Driver_Form::label7_Click);
                       //
                       // label8
                       //
                       this->label8->AutoSize = true;
                       this->label8->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label8->Location = System::Drawing::Point(619, 90);
                       this->label8->Name = L"label8";
                       this->label8->Size = System::Drawing::Size(54, 21);
                       this->label8->TabIndex = 23;
                       this->label8->Text = L"Model";
                       this->label8->Click += gcnew System::EventHandler(this,
&Driver_Form::label8_Click);
                       //
                       // label9
                       //
                       this->label9->AutoSize = true;
                       this->label9->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label9->Location = System::Drawing::Point(619, 152);
                       this->label9->Name = L"label9";
                       this->label9->Size = System::Drawing::Size(41, 21);
                       this->label9->TabIndex = 24;
                       this->label9->Text = L"Year";
                       this->label9->Click += gcnew System::EventHandler(this,
&Driver_Form::label9_Click);
                       //
                       // label10
                       //
                       this->label10->AutoSize = true;
                       this->label10->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                       this->label10->Location = System::Drawing::Point(619, 215);
                       this->label10->Name = L"label10";
                       this->label10->Size = System::Drawing::Size(129, 21);
                       this->label10->TabIndex = 25;
                       this->label10->Text = L"Lisence Plate No.";
                       this->label10->Click += gcnew System::EventHandler(this,
&Driver_Form::label10_Click);
                       //
                       // button5
```

```
                      //
                      this->button5->Location = System::Drawing::Point(833, 297);
                      this->button5->Name = L"button5";
                      this->button5->Size = System::Drawing::Size(75, 53);
                      this->button5->TabIndex = 28;
                      this->button5->Text = L"Add Vehicle";
                      this->button5->UseVisualStyleBackColor = true;
                      this->button5->Click += gcnew System::EventHandler(this,
&Driver_Form::button5_Click);
                      //
                      // checkBox1
                      //
                      this->checkBox1->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->checkBox1->AutoSize = true;
                      this->checkBox1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox1->Location = System::Drawing::Point(623, 297);
                      this->checkBox1->Name = L"checkBox1";
                      this->checkBox1->Size = System::Drawing::Size(93, 23);
                      this->checkBox1->TabIndex = 29;
                      this->checkBox1->Text = L"Delivery Vehicle";
                      this->checkBox1->UseVisualStyleBackColor = true;
                      this->checkBox1->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::checkBox1_CheckedChanged);
                      //
                      // checkBox2
                      //
                      this->checkBox2->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->checkBox2->AutoSize = true;
                      this->checkBox2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox2->Location = System::Drawing::Point(623, 322);
                      this->checkBox2->Name = L"checkBox2";
                      this->checkBox2->Size = System::Drawing::Size(95, 23);
                      this->checkBox2->TabIndex = 30;
                      this->checkBox2->Text = L"   Ride Vehicle   ";
                      this->checkBox2->UseVisualStyleBackColor = true;
                      this->checkBox2->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::checkBox2_CheckedChanged);
                      //
                      // checkBox3
                      //
                      this->checkBox3->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->checkBox3->AutoSize = true;
                      this->checkBox3->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox3->Location = System::Drawing::Point(734, 297);
                      this->checkBox3->Name = L"checkBox3";
                      this->checkBox3->Size = System::Drawing::Size(69, 23);
                      this->checkBox3->TabIndex = 31;
                      this->checkBox3->Text = L"Motorcycle";
                      this->checkBox3->UseVisualStyleBackColor = true;
                      this->checkBox3->CheckedChanged += gcnew
System::EventHandler(this, &Driver_Form::checkBox3_CheckedChanged);
                      //
                      // checkBox4
                      //
                      this->checkBox4->Appearance = 
System::Windows::Forms::Appearance::Button;
                      this->checkBox4->AutoSize = true;
                      this->checkBox4->FlatStyle = 
System::Windows::Forms::FlatStyle::Popup;
                      this->checkBox4->Location = System::Drawing::Point(734, 322);
                      this->checkBox4->Name = L"checkBox4";
                      this->checkBox4->Size = System::Drawing::Size(69, 23);
                      this->checkBox4->TabIndex = 32;
                      this->checkBox4->Text = L"      Car      ";
                      this->checkBox4->UseVisualStyleBackColor = true;
                      this->checkBox4->CheckedChanged += gcnew 
System::EventHandler(this, &Driver_Form::checkBox4_CheckedChanged);
                      // 
                      // label11
                      // 
                      this->label11->AutoSize = true;
                      this->label11->Location = System::Drawing::Point(623, 61);
                      this->label11->Name = L"label11";
                      this->label11->Size = System::Drawing::Size(0, 13);
                      this->label11->TabIndex = 33;
                      // 
                      // label12
                      // 
                      this->label12->AutoSize = true;
                      this->label12->Location = System::Drawing::Point(623, 131);
                      this->label12->Name = L"label12";
                      this->label12->Size = System::Drawing::Size(0, 13);
                      this->label12->TabIndex = 34;
                      // 
                      // label13
                      // 
                      this->label13->AutoSize = true;
                      this->label13->Location = System::Drawing::Point(623, 193);
                      this->label13->Name = L"label13";
                      this->label13->Size = System::Drawing::Size(0, 13);
                      this->label13->TabIndex = 35;
                      // 
                      // label14
                      // 
                      this->label14->AutoSize = true;
                      this->label14->Location = System::Drawing::Point(626, 256);
                      this->label14->Name = L"label14";
                      this->label14->Size = System::Drawing::Size(0, 13);
                      this->label14->TabIndex = 36;
                      // 
                      // Driver_Form
                      // 
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode = 
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(920, 361);
                      this->Controls->Add(this->label14);
                      this->Controls->Add(this->label13);
                      this->Controls->Add(this->label12);
                      this->Controls->Add(this->label11);
                      this->Controls->Add(this->checkBox4);
                      this->Controls->Add(this->checkBox3);
                      this->Controls->Add(this->checkBox2);
                      this->Controls->Add(this->checkBox1);
                      this->Controls->Add(this->button5);
                      this->Controls->Add(this->label10);
                      this->Controls->Add(this->label9);
                      this->Controls->Add(this->label8);
                      this->Controls->Add(this->label7);
                      this->Controls->Add(this->textBox5);
                      this->Controls->Add(this->textBox4);
                      this->Controls->Add(this->textBox3);
                      this->Controls->Add(this->textBox2);
                      this->Controls->Add(this->radioButton4);
                      this->Controls->Add(this->label6);
                      this->Controls->Add(this->textBox1);
                      this->Controls->Add(this->label5);
                      this->Controls->Add(this->button4);
                      this->Controls->Add(this->button3);
                      this->Controls->Add(this->button2);
                      this->Controls->Add(this->button1);
                      this->Controls->Add(this->label4);
                      this->Controls->Add(this->richTextBox1);
                      this->Controls->Add(this->radioButton3);
                      this->Controls->Add(this->radioButton2);
                      this->Controls->Add(this->radioButton1);
                      this->Controls->Add(this->label3);
                      this->Controls->Add(this->label2);
                      this->Controls->Add(this->label1);
                      this->Name = L"Driver_Form";
                      this->Text = L"Transport Manager+";
                      this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&Driver_Form::Driver_Form_FormClosing);
                      this->Load += gcnew System::EventHandler(this,
&Driver_Form::Driver_Form_Load);
                      this->ResumeLayout(false);
                      this->PerformLayout();

                }
#pragma endregion
#pragma region Function Code
      private: System::Void Driver_Form_Load(System::Object^ sender,
System::EventArgs^ e) {
             TMS_Main.LoadSavedData();
             std::ifstream CurrentAcc;
             std::string s1, s2, s3;
             CurrentAcc.open("Data/CurrentAcc.txt");
             while (!CurrentAcc.eof()) {
                   CurrentAcc >> s1 >> s2;
             }
             if (s1 == "Driver") {
                   i = StringToInt(s2);
                   String^ name, ^ balance, ^ cnic, ^ rating;
                   s3 = TMS_Main.D_Accounts[i]->getFName() + " " +
TMS_Main.D_Accounts[i]->getLName();
                   name = gcnew String(s3.data());
                   s3 = TMS_Main.D_Accounts[i]->getID();
```

```
                  cnic = gcnew String(s3.data());
                  s3 = "Balance: PKR " + IntToString(TMS_Main.D_Accounts[i]-
>getbal());
                  balance = gcnew String(s3.data());
                  int x = (TMS_Main.D_Accounts[i]->ComputeAndReturnRating()) *
100;
                  s3 = "Rating: " + FloatToString(x / 100.0);
                  rating = gcnew String(s3.data());
                  this->label1->Text = (name);
                  this->label3->Text = (cnic);
                  this->label4->Text = (rating);
                  this->label2->Text = (balance);
                  this->radioButton1->Checked = true;
                  HideVcl = 1;
                  this->label7->Hide();
                  this->label8->Hide();
                  this->label9->Hide();
                  this->label10->Hide();
                  this->textBox2->Hide();
                  this->textBox3->Hide();
                  this->textBox4->Hide();
                  this->textBox5->Hide();
                  this->ClientSize = System::Drawing::Size(546, 361);
            }
            else {
                  this->Close();
            }
      }
      private: System::Void label1_Click(System::Object^ sender, System::EventArgs^
e) {
      }
      private: System::Void label3_Click(System::Object^ sender, System::EventArgs^
e) {
      }
      private: System::Void label2_Click(System::Object^ sender, System::EventArgs^
e) {
      }

      private: System::Void radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
            if (this->radioButton1->Checked) {
                  this->radioButton2->Checked = false;
                  this->radioButton3->Checked = false;
                  this->radioButton4->Checked = false;
                  this->button1->Hide();
                  this->textBox1->Show();
                  this->richTextBox1->Hide();
                  this->label5->Text = ("Enter Ammount to Withdraw");
                  this->label5->Show();
                  this->label6->Show();
                  this->button2->Text = ("Withdraw");
                  this->button2->Location = System::Drawing::Point(321, 171);
                  this->button2->Show();
                  this->button3->Hide();
                  this->button4->Hide();
                  this->radioButton1->BackColor =
System::Drawing::SystemColors::ControlDark;
            }
```

```
        else {
            this->radioButton1->BackColor =
System::Drawing::Color::WhiteSmoke;
            this->textBox1->Text = "";
            this->button2->Location = System::Drawing::Point(272, 241);
        }
    }
    private: System::Void radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            if (!TMS_Main.D_Accounts[i]->getFreedom()) {
                this->radioButton1->Checked = false;
                this->radioButton4->Checked = false;
                this->radioButton3->Checked = false;
                this->button2->Text = "Reject Order";
                this->button3->Text = "Accept Order";
                this->textBox1->Hide();
                this->richTextBox1->Show();
                this->label5->Hide();
                this->label6->Hide();
                this->button1->Show();
                this->button2->Show();
                this->button3->Show();
                this->button4->Show();
                oID = 0;
                this->richTextBox1->Text = "\nYou have no incoming
orders";
                if (TMS_Main.Orders.size()) {
                    while (TMS_Main.D_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getDID() && oID < TMS_Main.Orders.size() - 1) {
                        oID++;
                        while (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID() && (TMS_Main.Orders[oID]->getAccepted())) {
                            oID++;
                        }
                    }
                    if (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID()) {
                        int h, m, s;
                        s = TMS_Main.Orders[oID]->getPlaced();
                        h = (s / 3600) % 24 + 5;
                        s %= 3600;
                        m = s / 60;
                        s /= 60;
                        std::string AmPm = "AM";
                        if (h > 12) {
                            h %= 12;
                            AmPm = "PM";
                        }
                        std::string s1 = "Incoming Order for a " +
TMS_Main.Orders[oID]->getType() + " From " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]->getFName() +
" " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>getLName() + "\nPlaced at " + IntToString(h) + ":";
                        if (m < 10) s1 = s1 + "0";
                        s1 = s1 + IntToString(m) + AmPm +
+"\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nOrder Cost: PKR " + IntToString(TMS_Main.Orders[oID]-
>getCost());
                        System::String^ s2 = gcnew String(s1.data());
                        richTextBox1->Text = s2;
                    }
                    oID = 0;
                }
            }
            else {
                this->radioButton1->Checked = false;
                this->radioButton4->Checked = false;
                this->radioButton3->Checked = false;
                this->textBox1->Hide();
                this->richTextBox1->Text = "\nPlease Complete your Pending
order first";
                this->richTextBox1->Show();
                this->label5->Hide();
                this->label6->Hide();
                this->button1->Hide();
                this->button2->Hide();
                this->button3->Hide();
                this->button4->Hide();
            }
            this->radioButton2->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->radioButton2->BackColor =
System::Drawing::Color::WhiteSmoke;
        }
    }
private: System::Void radioButton3_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton3->Checked) {
            HideVcl = 1;
            this->label7->Hide();
            this->label8->Hide();
            this->label9->Hide();
            this->label10->Hide();
            this->textBox2->Hide();
            this->textBox3->Hide();
            this->textBox4->Hide();
            this->textBox5->Hide();
            this->checkBox1->Hide();
            this->checkBox2->Hide();
            this->ClientSize = System::Drawing::Size(546, 361);
            this->radioButton2->Checked = false;
            this->radioButton1->Checked = false;
            this->radioButton4->Checked = false;
            this->button2->Text = "Remove Vehicle";
            this->button3->Text = "Add\nVehicle";
            this->textBox1->Hide();
            this->richTextBox1->Show();
            this->label5->Hide();
            this->label6->Hide();
            this->button1->Show();
            this->button2->Show();
            this->button3->Show();
            this->button4->Show();
            vID = 0;
            this->richTextBox1->Text = "\nYou do not have any Vehicles yet";
            if (TMS_Main.D_Accounts[i]->Vehicles.size()) {
                std::string s1 = "\n" + TMS_Main.D_Accounts[i]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[i]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[i]->Vehicles[vID]->Model.Year)
+ " Model\nRating: " + FloatToString(TMS_Main.D_Accounts[i]->Vehicles[vID]-
>ComputeAndReturnRating());
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = s2;
            }
            this->radioButton3->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->radioButton3->BackColor =
System::Drawing::Color::WhiteSmoke;
            this->textBox1->Text = "";
            this->ClientSize = System::Drawing::Size(546, 361);
            this->button2->Location = System::Drawing::Point(272, 241);
        }
    }
    private: System::Void radioButton4_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton4->Checked) {
            this->radioButton2->Checked = false;
            this->radioButton1->Checked = false;
            this->radioButton3->Checked = false;
            this->richTextBox1->Hide();
            this->button1->Hide();
            this->textBox1->Show();
            this->label5->Text = ("Please enter your password to Delete the
account");
            this->label5->Show();
            this->label6->Hide();
            this->button2->Text = ("Delete");
            this->button2->Location = System::Drawing::Point(321, 171);
            this->button2->Show();
            this->button3->Hide();
            this->button4->Hide();
            this->radioButton4->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->radioButton4->BackColor =
System::Drawing::Color::WhiteSmoke;
            this->textBox1->Text = "";
            this->button2->Location = System::Drawing::Point(272, 241);
        }
    }
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            if (TMS_Main.Orders.size()) {
                if (oID > 0) {
                    oID--;
                }
                while (TMS_Main.D_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getDID() && oID > 0) {
                    oID--;
                    while (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID() && (TMS_Main.Orders[oID]->getAccepted())) {
                        oID--;
                    }
                }
                if (oID >= 0 && TMS_Main.D_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getDID()) {
                    while (TMS_Main.D_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getDID() && oID < TMS_Main.Orders.size() - 1) {
                        oID++;
                        while (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID() && (TMS_Main.Orders[oID]->getAccepted())) {
                            oID++;
                        }
                    }
                }
                if (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID()) {
                    if (TMS_Main.Orders.size()) {
                        int h, m, s;
                        s = TMS_Main.Orders[oID]->getPlaced();
                        h = (s / 3600) % 24 + 5;
                        s %= 3600;
                        m = s / 60;
                        s /= 60;
                        std::string AmPm = "AM";
                        if (h > 12) {
                            h %= 12;
                            AmPm = "PM";
                        }
                        std::string s1 = "Incoming Order for a " +
TMS_Main.Orders[oID]->getType() + " From " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]->getFName() +
" " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>getLName() + "\nPlaced at " + IntToString(h) + ":";
                        if (m < 10) s1 = s1 + "0";
                        s1 = s1 + IntToString(m) + AmPm +
+"\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nOrder Cost: PKR " + IntToString(TMS_Main.Orders[oID]-
>getCost());
                        System::String^ s2 = gcnew String(s1.data());
                        richTextBox1->Text = s2;
                    }
                }
                else {
                    oID = 0;
                }
            }
        }
        if (this->radioButton3->Checked) {
            if (vID > 0) {
                vID--;
            }
            if (TMS_Main.D_Accounts[i]->Vehicles.size()) {
                std::string s1 = "\n" + TMS_Main.D_Accounts[i]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[i]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[i]->Vehicles[vID]->Model.Year)
+ " Model\nRating: " + FloatToString(TMS_Main.D_Accounts[i]->Vehicles[vID]-
>ComputeAndReturnRating());
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = s2;
            }
        }
    }
    private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            msclr::interop::marshal_context context;
            System::String^ s1 = Convert::ToString(textBox1->Text);
            std::string s2 = context.marshal_as<std::string>(s1);
            if (isInt(s2)) {
                int x = StringToInt(s2);
                if (x <= TMS_Main.D_Accounts[i]->getbal()) {
                    TMS_Main.D_Accounts[i]->deductbal(x);
                    s2 = "Succesfully withdrew PKR " + s2;
                    s1 = gcnew String(s2.data());
                    MessageBox::Show(s1, "Transaction Successful",
MessageBoxButtons::OK, MessageBoxIcon::Information);
                    s2 = "Balance: PKR " +
IntToString(TMS_Main.D_Accounts[i]->getbal());
                    s1 = gcnew String(s2.data());
                    this->label2->Text = (s1);
                    return;
                }
                else {
                    MessageBox::Show("You have insufficent balance to
withdraw this ammount", "Error: Insufficent Balance", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                    return;
                }
            }
            else {
                MessageBox::Show("Please enter a number in the input
field", "Error: Incorrect Information", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                return;
```

```
			}
		}
		if (this->radioButton2->Checked) {
			if (MessageBox::Show("Are you sure you want to Reject this 
Order?", "Confirm Order Rejection", MessageBoxButtons::YesNo, 
MessageBoxIcon::Question) == System::Windows::Forms::DialogResult::Yes) {
	
	TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>addbal(TMS_Main.Orders[oID]->getCost());
				TMS_Main.RejectOrder(oID);
				this->radioButton2->Checked = false;
				this->radioButton2->Checked = true;
			}
		}
		if (this->radioButton3->Checked) {
			if (MessageBox::Show("Are you sure you want to remove this 
vehicle from your account?", "Confirm Vehicle Removal", MessageBoxButtons::YesNo, 
MessageBoxIcon::Question) == System::Windows::Forms::DialogResult::Yes) {
				TMS_Main.D_Accounts[i]->removeVehicle(vID);
				this->radioButton3->Checked = false;
				this->radioButton3->Checked = true;
				MessageBox::Show("Are you sure you want to remove this 
vehicle from your account?", "Vehicle Removal Successful", MessageBoxButtons::OK, 
MessageBoxIcon::Information);
			}
		}
		if (this->radioButton4->Checked) {
			msclr::interop::marshal_context context;
			System::String^ s1 = Convert::ToString(textBox1->Text);
			std::string s2 = context.marshal_as<std::string>(s1);
			if (s2 == TMS_Main.D_Accounts[i]->getPass()) {
				if (MessageBox::Show("Are you sure you want to delete your 
account?\nIt cannot be recovered once deleted", "Confirm Account Deletion", 
MessageBoxButtons::YesNo, MessageBoxIcon::Warning) == 
System::Windows::Forms::DialogResult::Yes) {
					TMS_Main.DeleteDAccount(i);
					MessageBox::Show("Account Deleted, Application will 
close now so you may Log-In with or Create another account", "Account Deletion 
Succesful", MessageBoxButtons::OK, MessageBoxIcon::Information);
					this->Close();
				}
			}
			else {
				MessageBox::Show("Incorrect Password Entered", "Error: 
Incorrect Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
			}
		}
	}
	private: System::Void button3_Click(System::Object^ sender, 
System::EventArgs^ e) {
		if (this->radioButton2->Checked) {
			if (MessageBox::Show("Once you accept an order, you will recieve 
payment in advance but you cannot accept/recieve more orders till the order has been 
confirmed to be completed by the customer", "Confirm Order Acceptance", 
MessageBoxButtons::YesNo, MessageBoxIcon::Question) == 
System::Windows::Forms::DialogResult::Yes) {
				TMS_Main.D_Accounts[i]->addbal(TMS_Main.Orders[oID]-
>getCost());
```

```
                TMS_Main.AcceptOrder(TMS_Main.Orders[oID]->getID());
                std::string s2 = "Balance: PKR " +
IntToString(TMS_Main.D_Accounts[i]->getbal());
                System::String^ s1 = gcnew String(s2.data());
                this->label2->Text = (s1);
                this->radioButton2->Checked = false;
                this->radioButton2->Checked = true;
            }
        }
        if (this->radioButton3->Checked) {
            if (HideVcl) {
                this->label7->Show();
                this->label8->Show();
                this->label9->Show();
                this->label10->Show();
                this->textBox2->Show();
                this->textBox3->Show();
                this->textBox4->Show();
                this->textBox5->Show();
                this->checkBox1->Show();
                this->checkBox2->Show();
                this->ClientSize = System::Drawing::Size(920, 361);
            }
            else {
                this->label7->Hide();
                this->label8->Hide();
                this->label9->Hide();
                this->label10->Hide();
                this->textBox2->Hide();
                this->textBox3->Hide();
                this->textBox4->Hide();
                this->textBox5->Hide();
                this->checkBox1->Hide();
                this->checkBox2->Hide();
                this->ClientSize = System::Drawing::Size(546, 361);
            }
            HideVcl = !HideVcl;
        }
    }
    private: System::Void button4_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            if (TMS_Main.Orders.size()) {
                if (TMS_Main.Orders.size()) {
                    if (oID > 0) {
                        oID++;
                    }
                    while (TMS_Main.D_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getDID() && oID < TMS_Main.Orders.size() - 1) {
                        oID++;
                        while (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID() && (TMS_Main.Orders[oID]->getAccepted())) {
                            oID++;
                        }
                    }
                    if (oID <= TMS_Main.Orders.size() - 1 &&
TMS_Main.D_Accounts[i]->getID() != TMS_Main.Orders[oID]->getDID()) {
```

```
                        while (TMS_Main.D_Accounts[i]->getID() !=
TMS_Main.Orders[oID]->getDID() && oID > 0) {
                            oID--;
                            while (TMS_Main.Orders[oID]->getDID()
== TMS_Main.D_Accounts[i]->getID() && (TMS_Main.Orders[oID]->getAccepted())) {
                                oID--;
                            }
                        }
                    }
                    if (TMS_Main.Orders[oID]->getDID() ==
TMS_Main.D_Accounts[i]->getID()) {
                        if (TMS_Main.Orders.size()) {
                            int h, m, s;
                            s = TMS_Main.Orders[oID]->getPlaced();
                            h = (s / 3600) % 24 + 5;
                            s %= 3600;
                            m = s / 60;
                            s /= 60;
                            std::string AmPm = "AM";
                            if (h > 12) {
                                h %= 12;
                                AmPm = "PM";
                            }
                            std::string s1 = "Incoming Order for a
" + TMS_Main.Orders[oID]->getType() + " From " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]->getFName() +
" " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>getLName() + "\nPlaced at " + IntToString(h) + ":";
                            if (m < 10) s1 = s1 + "0";
                            s1 = s1 + IntToString(m) + AmPm +
+"\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nOrder Cost: PKR " + IntToString(TMS_Main.Orders[oID]-
>getCost());
                            System::String^ s2 = gcnew
String(s1.data());
                            richTextBox1->Text = s2;
                        }
                    }
                }
            }
        }
        if (this->radioButton3->Checked) {
            if (vID < TMS_Main.D_Accounts[i]->Vehicles.size() - 1) {
                vID++;
            }
            if (TMS_Main.D_Accounts[i]->Vehicles.size()) {
                std::string s1 = "\n" + TMS_Main.D_Accounts[i]-
>Vehicles[vID]->Model.Company + " " + TMS_Main.D_Accounts[i]->Vehicles[vID]-
>Model.Model + " " + IntToString(TMS_Main.D_Accounts[i]->Vehicles[vID]->Model.Year)
```

```
+ " Model\nRating: " + FloatToString(TMS_Main.D_Accounts[i]->Vehicles[vID]-
>ComputeAndReturnRating());
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = s2;
            }
        }
    }

    private: System::Void Driver_Form_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {
        TMS_Main.SaveLoadedData();
        Application::Exit();
    }
    private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ s1 = Convert::ToString(textBox1->Text);
        std::string s2 = context.marshal_as<std::string>(s1);
        if (!isInt(s2)) {
            this->label6->Text = ("Please enter a number");
        }
        else {
            this->label6->Text = "";
        }
    }
    private: System::Void label9_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void label7_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void button5_Click(System::Object^ sender,
System::EventArgs^ e) {
        std::string std1, std2, std3, std4, std5, std6;
        msclr::interop::marshal_context context;
        System::String^ s1 = Convert::ToString(textBox2->Text);
        std1 = context.marshal_as<std::string>(s1);
        s1 = Convert::ToString(textBox3->Text);
        std2 = context.marshal_as<std::string>(s1);
        s1 = Convert::ToString(textBox4->Text);
        std3 = context.marshal_as<std::string>(s1);
        s1 = Convert::ToString(textBox5->Text);
        std4 = context.marshal_as<std::string>(s1);
        try {
            if (isNull(std1) || isNull(std2) || isNull(std3) ||
isNull(std4)) {
                throw 1;
            }
            if (ContainsSpaces(std1) || ContainsSpaces(std2) ||
ContainsSpaces(std3) || ContainsSpaces(std4)) {
                throw 0;
            }
        }
        catch (int x) {
            if (x) {
                MessageBox::Show("Please do not leave any input fields
empty, Enter data and try again", "Error: Missing Information",
MessageBoxButtons::OK, MessageBoxIcon::Hand);
```

```
                return;
            }
            else {
                MessageBox::Show("Cannot Enter spaces in input fields, 
please Re-Enter the data without using spaces", "Error: Invalid Information", 
MessageBoxButtons::OK, MessageBoxIcon::Hand);
                return;
            }
        }
        if (!isInt(std3)) {
            MessageBox::Show("Please enter a number in the Vehicle Year 
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
        if (isInt(std4)) {
            if (!TMS_Main.idUniqueLisence(StringToInt(std4))) {
                MessageBox::Show("Entered Lisence plate number is already 
in use by another vehicle", "Error: Repeated Information", MessageBoxButtons::OK, 
MessageBoxIcon::Hand);
                return;
            }
        }
        else {
            MessageBox::Show("Please enter a number in the Licence Plate 
Field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
        if (this->checkBox1->Checked) std5 = "Delivery";
        else std5 = "Ride";
        if (this->checkBox3->Checked) std6 = "Motorcycle";
        else std6 = "Car";
        VclModel m(std1, std2, StringToInt(std3));
        Vehicle temp(TMS_Main.D_Accounts[i]->getID(), std6, StringToInt(std4), 
m, std5);
        TMS_Main.D_Accounts[i]->Vehicles.push_back(temp);
        MessageBox::Show("Succesfully Added the vehicle to your account", 
"Vehicle Addition Successful", MessageBoxButtons::OK, MessageBoxIcon::Information);
        vID = 0;
        std::string se1 = "\n" + TMS_Main.D_Accounts[i]->Vehicles[vID]-
>Model.Company + " " + TMS_Main.D_Accounts[i]->Vehicles[vID]->Model.Model + " " + 
IntToString(TMS_Main.D_Accounts[i]->Vehicles[vID]->Model.Year) + " Model\nRating: " 
+ FloatToString(TMS_Main.D_Accounts[i]->Vehicles[vID]->ComputeAndReturnRating());
        System::String^ s2 = gcnew String(se1.data());
        this->richTextBox1->Text = s2;
        return;
    }
    private: System::Void textBox2_TextChanged(System::Object^ sender, 
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ st1 = gcnew String(this->textBox2->Text);
        std::string s2 = context.marshal_as<std::string>(st1);
        if (ContainsSpaces(s2)) {
            this->label11->Text = "Cannot enter spaces in this input field";
        }
        else {
            this->label11->Text = "";
        }
    }
    private: System::Void textBox3_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ st1 = gcnew String(this->textBox2->Text);
        std::string s2 = context.marshal_as<std::string>(st1);
        if (ContainsSpaces(s2)) {
            this->label12->Text = "Cannot enter spaces in this input field";
        }
        else {
            this->label12->Text = "";
        }
    }
    private: System::Void textBox4_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ st1 = gcnew String(this->textBox4->Text);
        std::string s2 = context.marshal_as<std::string>(st1);
        if (!isInt(s2)) {
            this->label13->Text = "Please enter a number";
        }
        else {
            this->label13->Text = "";
        }
    }
    private: System::Void textBox5_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ st1 = gcnew String(this->textBox5->Text);
        std::string s2 = context.marshal_as<std::string>(st1);
        if (!isInt(s2)) {
            this->label14->Text = "Please enter a number";
        }
        else {
            this->label14->Text = "";
        }
    }
    private: System::Void label8_Click(System::Object^ sender, System::EventArgs^
e) {
    }
    private: System::Void label10_Click(System::Object^ sender,
System::EventArgs^ e) {
    }
    private: System::Void checkBox1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox1->Checked) {
            this->checkBox2->Checked = false;
            this->checkBox1->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->checkBox1->BackColor = System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void checkBox2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->checkBox2->Checked) {
            this->checkBox1->Checked = false;
            this->checkBox2->BackColor = 
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->checkBox2->BackColor = System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void checkBox3_CheckedChanged(System::Object^ sender, 
System::EventArgs^ e) {
        if (this->checkBox3->Checked) {
            this->checkBox4->Checked = false;
            this->checkBox3->BackColor = 
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->checkBox3->BackColor = System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void checkBox4_CheckedChanged(System::Object^ sender, 
System::EventArgs^ e) {
        if (this->checkBox4->Checked) {
            this->checkBox3->Checked = false;
            this->checkBox4->BackColor = 
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->checkBox4->BackColor = System::Drawing::Color::WhiteSmoke;
        }
    }
#pragma endregion
    };
#pragma endregion
}
```

Admin Dashboard (Admin_Form.h)

```
#pragma once
extern TMS TMS_Main;
extern int dID, oID;
int cID;
namespace TMS_Forms {
      using namespace System;
      using namespace System::ComponentModel;
      using namespace System::Collections;
      using namespace System::Windows::Forms;
      using namespace System::Data;
      using namespace System::Drawing;
      public ref class Admin_Form : public System::Windows::Forms::Form {
      public:
             Admin_Form(void) {
                   InitializeComponent();
             }
      protected:
             ~Admin_Form() {
                   if (components) {
                          delete components;
                   }
             }
      private: System::Windows::Forms::RadioButton^ radioButton1;
      protected:
      private: System::Windows::Forms::RadioButton^ radioButton2;
      private: System::Windows::Forms::RadioButton^ radioButton3;
      private: System::Windows::Forms::RadioButton^ radioButton4;
      private: System::Windows::Forms::Label^ label1;
      private: System::Windows::Forms::RichTextBox^ richTextBox1;
      private: System::Windows::Forms::Button^ button1;
      private: System::Windows::Forms::Button^ button2;
      private: System::Windows::Forms::TextBox^ textBox1;
      private: System::Windows::Forms::TextBox^ textBox2;
      private: System::Windows::Forms::Button^ button3;
      private: System::Windows::Forms::Button^ button4;
      private: System::Windows::Forms::Button^ button5;
      private: System::Windows::Forms::Label^ label2;
      private: System::Windows::Forms::Label^ label3;
      private: System::Windows::Forms::Button^ button6;
      private: System::Windows::Forms::Button^ button7;
      private: System::Windows::Forms::Label^ label4;
      private: System::Windows::Forms::Label^ label5;
      private: System::ComponentModel::Container^ components;
#pragma region Component Code
                void InitializeComponent(void) {
                      this->radioButton1 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton2 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton3 = (gcnew
System::Windows::Forms::RadioButton());
                      this->radioButton4 = (gcnew
System::Windows::Forms::RadioButton());
                      this->label1 = (gcnew System::Windows::Forms::Label());
                      this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
                      this->button1 = (gcnew System::Windows::Forms::Button());
                      this->button2 = (gcnew System::Windows::Forms::Button());
                      this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                      this->textBox2 = (gcnew System::Windows::Forms::TextBox());
                      this->button3 = (gcnew System::Windows::Forms::Button());
                      this->button4 = (gcnew System::Windows::Forms::Button());
                      this->button5 = (gcnew System::Windows::Forms::Button());
                      this->label2 = (gcnew System::Windows::Forms::Label());
                      this->label3 = (gcnew System::Windows::Forms::Label());
                      this->button6 = (gcnew System::Windows::Forms::Button());
                      this->button7 = (gcnew System::Windows::Forms::Button());
                      this->label4 = (gcnew System::Windows::Forms::Label());
                      this->label5 = (gcnew System::Windows::Forms::Label());
                      this->SuspendLayout();
                      //
                      // radioButton1
                      //
                      this->radioButton1->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton1->AutoSize = true;
                      this->radioButton1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton1->Location = System::Drawing::Point(11,
155);
                      this->radioButton1->Name = L"radioButton1";
                      this->radioButton1->Size = System::Drawing::Size(135, 75);
                      this->radioButton1->TabIndex = 0;
                      this->radioButton1->Text = L"\n\nView Customer Accounts\n\n
";
                      this->radioButton1->UseVisualStyleBackColor = true;
                      this->radioButton1->CheckedChanged += gcnew
System::EventHandler(this, &Admin_Form::radioButton1_CheckedChanged);
                      //
                      // radioButton2
                      //
                      this->radioButton2->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton2->AutoSize = true;
                      this->radioButton2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->radioButton2->Location = System::Drawing::Point(10,
236);
                      this->radioButton2->Name = L"radioButton2";
                      this->radioButton2->Size = System::Drawing::Size(134, 75);
                      this->radioButton2->TabIndex = 1;
                      this->radioButton2->Text = L"\n\n   View Driver Accounts
\n\n ";
                      this->radioButton2->UseVisualStyleBackColor = true;
                      this->radioButton2->CheckedChanged += gcnew
System::EventHandler(this, &Admin_Form::radioButton2_CheckedChanged);
                      //
                      // radioButton3
                      //
                      this->radioButton3->Appearance =
System::Windows::Forms::Appearance::Button;
                      this->radioButton3->AutoSize = true;
```

```
                       this->radioButton3->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                       this->radioButton3->Location = System::Drawing::Point(10,
317);
                       this->radioButton3->Name = L"radioButton3";
                       this->radioButton3->Size = System::Drawing::Size(135, 75);
                       this->radioButton3->TabIndex = 2;
                       this->radioButton3->Text = L"\n\n    View Current Orders
\n\n ";
                       this->radioButton3->UseVisualStyleBackColor = true;
                       this->radioButton3->CheckedChanged += gcnew
System::EventHandler(this, &Admin_Form::radioButton3_CheckedChanged);
                       //
                       // radioButton4
                       //
                       this->radioButton4->Appearance =
System::Windows::Forms::Appearance::Button;
                       this->radioButton4->AutoSize = true;
                       this->radioButton4->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                       this->radioButton4->Location = System::Drawing::Point(11,
398);
                       this->radioButton4->Name = L"radioButton4";
                       this->radioButton4->Size = System::Drawing::Size(134, 75);
                       this->radioButton4->TabIndex = 3;
                       this->radioButton4->Text = L"\n\n      View Past Orders
\n\n ";
                       this->radioButton4->UseVisualStyleBackColor = true;
                       this->radioButton4->CheckedChanged += gcnew
System::EventHandler(this, &Admin_Form::radioButton4_CheckedChanged);
                       //
                       // label1
                       //
                       this->label1->AutoSize = true;
                       this->label1->Font = (gcnew System::Drawing::Font(L"Rockwell
Condensed", 36, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                              static_cast<System::Byte>(0)));
                       this->label1->Location = System::Drawing::Point(292, 9);
                       this->label1->Name = L"label1";
                       this->label1->Size = System::Drawing::Size(368, 56);
                       this->label1->TabIndex = 4;
                       this->label1->Text = L"Admin Dashboard";
                       //
                       // richTextBox1
                       //
                       this->richTextBox1->BackColor =
System::Drawing::SystemColors::Menu;
                       this->richTextBox1->Font = (gcnew
System::Drawing::Font(L"Rockwell", 18, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                              static_cast<System::Byte>(0)));
                       this->richTextBox1->Location = System::Drawing::Point(221,
125);
                       this->richTextBox1->Name = L"richTextBox1";
                       this->richTextBox1->ReadOnly = true;
                       this->richTextBox1->Size = System::Drawing::Size(439, 213);
                       this->richTextBox1->TabIndex = 5;
```

```
                      this->richTextBox1->Text = L"";
                      // 
                      // button1
                      // 
                      this->button1->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button1->Font = (gcnew
System::Drawing::Font(L"Verdana", 26.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button1->Location = System::Drawing::Point(221, 344);
                      this->button1->Name = L"button1";
                      this->button1->Size = System::Drawing::Size(215, 48);
                      this->button1->TabIndex = 6;
                      this->button1->Text = L"<";
                      this->button1->UseVisualStyleBackColor = true;
                      this->button1->Click += gcnew System::EventHandler(this,
&Admin_Form::button1_Click);
                      // 
                      // button2
                      // 
                      this->button2->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button2->Font = (gcnew
System::Drawing::Font(L"Verdana", 26.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button2->Location = System::Drawing::Point(445, 344);
                      this->button2->Name = L"button2";
                      this->button2->Size = System::Drawing::Size(215, 48);
                      this->button2->TabIndex = 7;
                      this->button2->Text = L">";
                      this->button2->UseVisualStyleBackColor = true;
                      this->button2->Click += gcnew System::EventHandler(this,
&Admin_Form::button2_Click);
                      // 
                      // textBox1
                      // 
                      this->textBox1->Location = System::Drawing::Point(685, 140);
                      this->textBox1->Name = L"textBox1";
                      this->textBox1->Size = System::Drawing::Size(192, 20);
                      this->textBox1->TabIndex = 8;
                      this->textBox1->TextChanged += gcnew
System::EventHandler(this, &Admin_Form::textBox1_TextChanged);
                      // 
                      // textBox2
                      // 
                      this->textBox2->Location = System::Drawing::Point(685, 210);
                      this->textBox2->Name = L"textBox2";
                      this->textBox2->Size = System::Drawing::Size(192, 20);
                      this->textBox2->TabIndex = 9;
                      this->textBox2->TextChanged += gcnew
System::EventHandler(this, &Admin_Form::textBox2_TextChanged);
                      // 
                      // button3
                      // 
                      this->button3->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
```

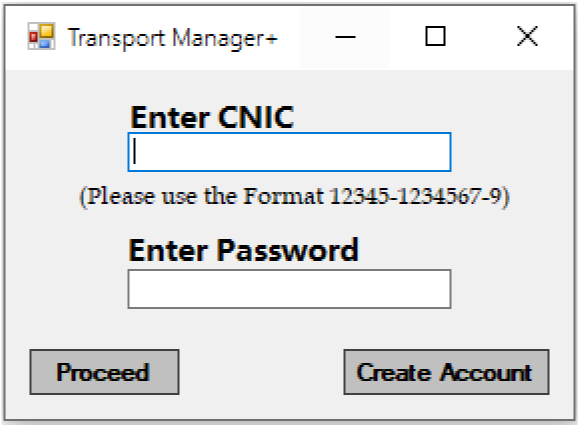
```
                      this->button3->Location = System::Drawing::Point(883, 140);
                      this->button3->Name = L"button3";
                      this->button3->Size = System::Drawing::Size(75, 20);
                      this->button3->TabIndex = 10;
                      this->button3->Text = L"Add";
                      this->button3->UseVisualStyleBackColor = true;
                      this->button3->Click += gcnew System::EventHandler(this,
&Admin_Form::button3_Click);
                      //
                      // button4
                      //
                      this->button4->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
                      this->button4->Location = System::Drawing::Point(883, 210);
                      this->button4->Name = L"button4";
                      this->button4->Size = System::Drawing::Size(75, 20);
                      this->button4->TabIndex = 11;
                      this->button4->Text = L"Remove";
                      this->button4->UseVisualStyleBackColor = true;
                      this->button4->Click += gcnew System::EventHandler(this,
&Admin_Form::button4_Click);
                      //
                      // button5
                      //
                      this->button5->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->button5->Location = System::Drawing::Point(685, 282);
                      this->button5->Name = L"button5";
                      this->button5->Size = System::Drawing::Size(150, 49);
                      this->button5->TabIndex = 12;
                      this->button5->Text = L"Delete Account";
                      this->button5->UseVisualStyleBackColor = true;
                      this->button5->Click += gcnew System::EventHandler(this,
&Admin_Form::button5_Click);
                      //
                      // label2
                      //
                      this->label2->AutoSize = true;
                      this->label2->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label2->Location = System::Drawing::Point(685, 121);
                      this->label2->Name = L"label2";
                      this->label2->Size = System::Drawing::Size(85, 18);
                      this->label2->TabIndex = 13;
                      this->label2->Text = L"Add Balance";
                      //
                      // label3
                      //
                      this->label3->AutoSize = true;
                      this->label3->Font = (gcnew System::Drawing::Font(L"Palatino
Linotype", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                             static_cast<System::Byte>(0)));
                      this->label3->Location = System::Drawing::Point(685, 191);
                      this->label3->Name = L"label3";
                      this->label3->Size = System::Drawing::Size(109, 18);
                      this->label3->TabIndex = 14;
                      this->label3->Text = L"Remove Balance";
                      //
                      // button6
                      //
                      this->button6->Location = System::Drawing::Point(685, 337);
                      this->button6->Name = L"button6";
                      this->button6->Size = System::Drawing::Size(109, 23);
                      this->button6->TabIndex = 15;
                      this->button6->Text = L"Sort by Rating";
                      this->button6->UseVisualStyleBackColor = true;
                      this->button6->Click += gcnew System::EventHandler(this,
&Admin_Form::button6_Click);
                      //
                      // button7
                      //
                      this->button7->Location = System::Drawing::Point(685, 366);
                      this->button7->Name = L"button7";
                      this->button7->Size = System::Drawing::Size(109, 23);
                      this->button7->TabIndex = 16;
                      this->button7->Text = L"Sort by Experience";
                      this->button7->UseVisualStyleBackColor = true;
                      this->button7->Click += gcnew System::EventHandler(this,
&Admin_Form::button7_Click);
                      //
                      // label4
                      //
                      this->label4->AutoSize = true;
                      this->label4->Location = System::Drawing::Point(685, 160);
                      this->label4->Name = L"label4";
                      this->label4->Size = System::Drawing::Size(0, 13);
                      this->label4->TabIndex = 17;
                      //
                      // label5
                      //
                      this->label5->AutoSize = true;
                      this->label5->Location = System::Drawing::Point(685, 230);
                      this->label5->Name = L"label5";
                      this->label5->Size = System::Drawing::Size(0, 13);
                      this->label5->TabIndex = 18;
                      //
                      // Admin_Form
                      //
                      this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                      this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                      this->ClientSize = System::Drawing::Size(1000, 485);
                      this->Controls->Add(this->label5);
                      this->Controls->Add(this->label4);
                      this->Controls->Add(this->button7);
                      this->Controls->Add(this->button6);
                      this->Controls->Add(this->label3);
                      this->Controls->Add(this->label2);
                      this->Controls->Add(this->button5);
                      this->Controls->Add(this->button4);
                      this->Controls->Add(this->button3);
                      this->Controls->Add(this->textBox2);
                      this->Controls->Add(this->textBox1);
                      this->Controls->Add(this->button2);
                      this->Controls->Add(this->button1);
                      this->Controls->Add(this->richTextBox1);
                      this->Controls->Add(this->label1);
                      this->Controls->Add(this->radioButton4);
                      this->Controls->Add(this->radioButton3);
                      this->Controls->Add(this->radioButton2);
                      this->Controls->Add(this->radioButton1);
                      this->Name = L"Admin_Form";
                      this->Text = L"Transport Manager";
                      this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&Admin_Form::Admin_Form_FormClosing);
                      this->Load += gcnew System::EventHandler(this,
&Admin_Form::Admin_Form_Load);
                      this->ResumeLayout(false);
                      this->PerformLayout();

                }
#pragma endregion
      private: System::Void Admin_Form_Load(System::Object^ sender,
System::EventArgs^ e) {
             TMS_Main.LoadSavedData();
             oID = 0;
             cID = 0;
             dID = 0;
             this->radioButton1->Checked = true;
      }
      private: System::Void radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
             if (this->radioButton1->Checked) {
                   this->radioButton2->Checked = false;
                   this->radioButton3->Checked = false;
                   this->radioButton4->Checked = false;
                   this->button1->Show();
                   this->button2->Show();
                   this->button6->Hide();
                   this->button7->Hide();
                   this->richTextBox1->Text = "No customers have signed up yet";
                   this->ClientSize = System::Drawing::Size(1000, 485);
                   cID = 0;
                   if (TMS_Main.C_Accounts.size()) {
                          std::string s1 = "Name: " + TMS_Main.C_Accounts[cID]-
>getFName() + " " + TMS_Main.C_Accounts[cID]->getLName() + "\nCNIC: " +
TMS_Main.C_Accounts[cID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.C_Accounts[cID]->getbal()) + "\n\nPassword: " +
TMS_Main.C_Accounts[cID]->getPass();
                          System::String^ s2 = gcnew String(s1.data());
                          this->richTextBox1->Text = (s2);
                   }
                   this->radioButton1->BackColor =
System::Drawing::SystemColors::ControlDark;
             }
             else {
                   this->radioButton1->BackColor =
System::Drawing::Color::WhiteSmoke;
```

```
        }
    }
    private: System::Void radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton2->Checked) {
            this->radioButton1->Checked = false;
            this->radioButton3->Checked = false;
            this->radioButton4->Checked = false;
            this->button1->Show();
            this->button2->Show();
            this->button6->Show();
            this->button7->Show();
            this->richTextBox1->Text = "No drivers have signed up yet";
            this->ClientSize = System::Drawing::Size(1000, 485);
            this->radioButton2->BackColor =
System::Drawing::SystemColors::ControlDark;
            if (TMS_Main.D_Accounts.size()) {
                std::string s1 = "Name: " + TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nCNIC: " +
TMS_Main.D_Accounts[dID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.D_Accounts[dID]->getbal()) + "\nNo. of Vehicles: " +
IntToString(TMS_Main.D_Accounts[dID]->Vehicles.size()) + "\nExperience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + " Years\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\nPassword: " +
TMS_Main.D_Accounts[dID]->getPass();
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
        else {
            this->radioButton2->BackColor =
System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void radioButton3_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton3->Checked) {
            this->radioButton1->Checked = false;
            this->radioButton2->Checked = false;
            this->radioButton4->Checked = false;
            this->button1->Show();
            this->button2->Show();
            this->richTextBox1->Text = "\nNo Orders placed yet";
            if (TMS_Main.Orders.size()) {
                std::string s1 = "Order for a " + TMS_Main.Orders[oID]-
>getType() + "\nPlaced by: " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]->getFName() +
" " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>getLName() + "\nSelected Driver: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nOrder Cost: " + IntToString(TMS_Main.Orders[oID]-
>getCost());
                if (TMS_Main.Orders[oID]->getAccepted()) s1 = s1 +
"\n\nAccepted";
                else s1 = s1 + "\n\nNot Accepted Yet";
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
            this->ClientSize = System::Drawing::Size(685, 485);
            this->radioButton3->BackColor =
System::Drawing::SystemColors::ControlDark;
        }
        else {
            this->radioButton3->BackColor =
System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void radioButton4_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton4->Checked) {
            this->radioButton1->Checked = false;
            this->radioButton2->Checked = false;
            this->radioButton3->Checked = false;
            this->richTextBox1->Text = "";
            this->button1->Hide();
            this->button2->Hide();
            this->ClientSize = System::Drawing::Size(685, 485);
            this->radioButton4->BackColor =
System::Drawing::SystemColors::ControlDark;
            std::ifstream Ordar;
            std::string str = "", st2;
            Ordar.open("Data/All Orders.txt");
            while (getline(Ordar, st2)) {
                for (int i = 0; i < st2.length(); i++) {
                    if (st2[i] == '%') {
                        st2[i] = '\n';
                    }
                }
                if (st2 != "") {
                    str = str + st2 + "\n------------------------------
----------------------\n";
                }
            }
            System::String^ s3 = gcnew String(str.data());
            this->richTextBox1->Text = (s3);
        }
        else {
            this->radioButton4->BackColor =
System::Drawing::Color::WhiteSmoke;
        }
    }
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            if (TMS_Main.C_Accounts.size()) {
                if (cID > 0) {
                    cID--;
                }
                std::string s1 = "Name: " + TMS_Main.C_Accounts[cID]-
>getFName() + " " + TMS_Main.C_Accounts[cID]->getLName() + "\nCNIC: " +
TMS_Main.C_Accounts[cID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.C_Accounts[cID]->getbal()) + "\n\nPassword: " +
TMS_Main.C_Accounts[cID]->getPass();
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
        if (this->radioButton2->Checked) {
            if (TMS_Main.D_Accounts.size()) {
                if (dID > 0) {
                    dID--;
                }
                std::string s1 = "Name: " + TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nCNIC: " +
TMS_Main.D_Accounts[dID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.D_Accounts[dID]->getbal()) + "\nNo. of Vehicles: " +
IntToString(TMS_Main.D_Accounts[dID]->Vehicles.size()) + "\nExperience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + " Years\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\nPassword: " +
TMS_Main.D_Accounts[dID]->getPass();
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
        if (this->radioButton3->Checked) {
            if (TMS_Main.Orders.size()) {
                if (oID > 0) {
                    oID--;
                }
                std::string s1 = "Order for a " + TMS_Main.Orders[oID]-
>getType() + "\nPlaced by: " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]->getFName() +
" " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>getLName() + "\nSelected Driver: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nOrder Cost: " + IntToString(TMS_Main.Orders[oID]-
>getCost());
                if (TMS_Main.Orders[oID]->getAccepted()) s1 = s1 +
"\n\nAccepted";
                else s1 = s1 + "\n\nNot Accepted Yet";
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
```

```
    }
    private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            if (TMS_Main.C_Accounts.size()) {
                if (cID < TMS_Main.C_Accounts.size() - 1) {
                    cID++;
                }
                std::string s1 = "Name: " + TMS_Main.C_Accounts[cID]-
>getFName() + " " + TMS_Main.C_Accounts[cID]->getLName() + "\nCNIC: " +
TMS_Main.C_Accounts[cID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.C_Accounts[cID]->getbal()) + "\n\nPassword: " +
TMS_Main.C_Accounts[cID]->getPass();
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
        if (this->radioButton2->Checked) {
            if (TMS_Main.D_Accounts.size()) {
                if (dID < TMS_Main.D_Accounts.size() - 1) {
                    dID++;
                }
                std::string s1 = "Name: " + TMS_Main.D_Accounts[dID]-
>getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() + "\nCNIC: " +
TMS_Main.D_Accounts[dID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.D_Accounts[dID]->getbal()) + "\nNo. of Vehicles: " +
IntToString(TMS_Main.D_Accounts[dID]->Vehicles.size()) + "\nExperience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + " Years\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\nPassword: " +
TMS_Main.D_Accounts[dID]->getPass();
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
        if (this->radioButton3->Checked) {
            if (TMS_Main.Orders.size()) {
                if (oID < TMS_Main.Orders.size() - 1) {
                    oID++;
                }
                std::string s1 = "Order for a " + TMS_Main.Orders[oID]-
>getType() + "\nPlaced by: " +
TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]->getFName() +
" " + TMS_Main.C_Accounts[TMS_Main.FindCNIC(TMS_Main.Orders[oID]->getCID())]-
>getLName() + "\nSelected Driver: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]->getFName()
+ " " + TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>getLName() + "\nSelected Vehicle: " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Company + " " +
TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Model + " " +
IntToString(TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.Orders[oID]->getDID())]-
>Vehicles[TMS_Main.FindID(TMS_Main.Orders[oID]->getDID(), TMS_Main.Orders[oID]-
>getVID())]->Model.Year) + "\nOrder Cost: " + IntToString(TMS_Main.Orders[oID]-
>getCost());
```

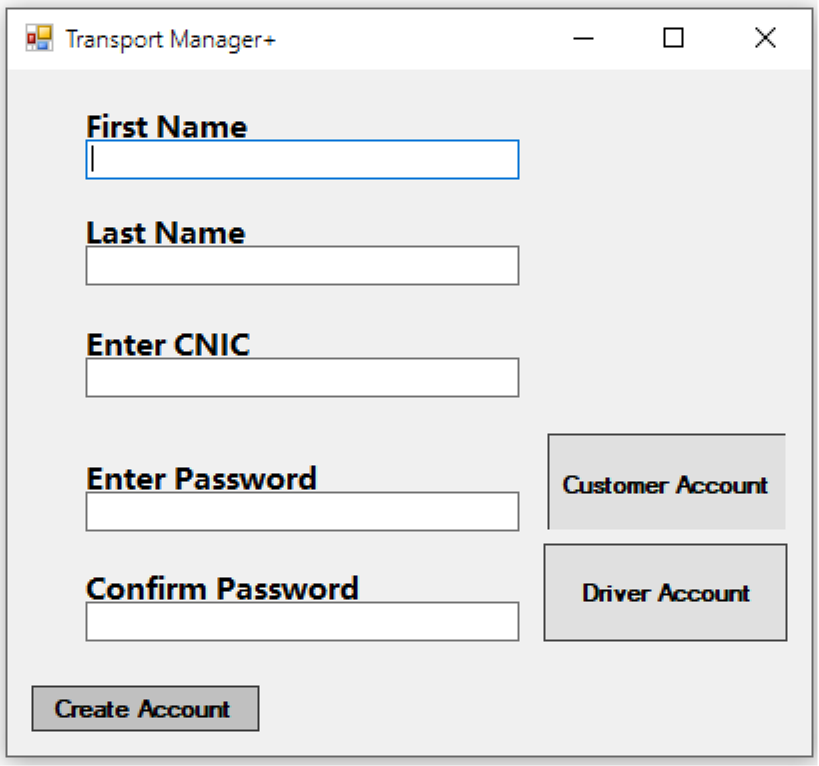
```
                if (TMS_Main.Orders[oID]->getAccepted()) s1 = s1 +
"\n\nAccepted";
                else s1 = s1 + "\n\nNot Accepted Yet";
                System::String^ s2 = gcnew String(s1.data());
                this->richTextBox1->Text = (s2);
            }
        }
    }
    private: System::Void button3_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            System::String^ s = Convert::ToString(textBox1->Text);
            msclr::interop::marshal_context context;
            std::string num = context.marshal_as<std::string>(s);
            if (!isInt(num)) {
                MessageBox::Show("Please enter a number in the input
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Stop);
                return;
            }
            TMS_Main.C_Accounts[cID]->addbal(StringToInt(num));
            num = "Name: " + TMS_Main.C_Accounts[cID]->getFName() + " " +
TMS_Main.C_Accounts[cID]->getLName() + "\nCNIC: " + TMS_Main.C_Accounts[cID]-
>getID() + "\nBalance: PKR " + IntToString(TMS_Main.C_Accounts[cID]->getbal()) +
"\n\nPassword: " + TMS_Main.C_Accounts[cID]->getPass();
            s = gcnew String(num.data());
            this->richTextBox1->Text = (s);
        }
        if (this->radioButton2->Checked) {
            System::String^ s = Convert::ToString(textBox1->Text);
            msclr::interop::marshal_context context;
            std::string num = context.marshal_as<std::string>(s);
            if (!isInt(num)) {
                MessageBox::Show("Please enter a number in the input
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Stop);
                return;
            }
            TMS_Main.D_Accounts[TMS_Main.FindCNIC2(TMS_Main.D_Accounts[dID]-
>getID())]->addbal(StringToInt(num));
            TMS_Main.D_Accounts[dID]->addbal(StringToInt(num));
            num = "Name: " + TMS_Main.D_Accounts[dID]->getFName() + " " +
TMS_Main.D_Accounts[dID]->getLName() + "\nCNIC: " + TMS_Main.D_Accounts[dID]-
>getID() + "\nBalance: PKR " + IntToString(TMS_Main.D_Accounts[dID]->getbal()) +
"\nNo. of Vehicles: " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles.size()) +
"\nExperience: " + IntToString(TMS_Main.D_Accounts[dID]->getExp()) + "
Years\nRating: " + FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating())
+ "\nPassword: " + TMS_Main.D_Accounts[dID]->getPass();
            s = gcnew String(num.data());
            this->richTextBox1->Text = (s);
        }
    }
    private: System::Void button4_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            System::String^ s = Convert::ToString(textBox2->Text);
            msclr::interop::marshal_context context;
            std::string num = context.marshal_as<std::string>(s);
            if (!isInt(num)) {
                MessageBox::Show("Please enter a number in the input 
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Stop);
                return;
            }
            if (StringToInt(num) > TMS_Main.C_Accounts[cID]->getbal()) {
                MessageBox::Show("The customer has insufficent balance for 
this deduction", "Error: Insufficent Balance", MessageBoxButtons::OK, 
MessageBoxIcon::Stop);
                return;
            }
            TMS_Main.C_Accounts[cID]->deductbal(StringToInt(num));
            num = "Name: " + TMS_Main.C_Accounts[cID]->getFName() + " " + 
TMS_Main.C_Accounts[cID]->getLName() + "\nCNIC: " + TMS_Main.C_Accounts[cID]-
>getID() + "\nBalance: PKR " + IntToString(TMS_Main.C_Accounts[cID]->getbal()) + 
"\n\nPassword: " + TMS_Main.C_Accounts[cID]->getPass();
            s = gcnew String(num.data());
            this->richTextBox1->Text = (s);
        }
        if (this->radioButton2->Checked) {
            System::String^ s = Convert::ToString(textBox2->Text);
            msclr::interop::marshal_context context;
            std::string num = context.marshal_as<std::string>(s);
            if (!isInt(num)) {
                MessageBox::Show("Please enter a number in the input 
field", "Error: Invalid Information", MessageBoxButtons::OK, MessageBoxIcon::Stop);
                return;
            }
            if (StringToInt(num) > TMS_Main.D_Accounts[dID]->getbal()) {
                MessageBox::Show("The driver has insufficent balance for 
this deduction", "Error: Insufficent Balance", MessageBoxButtons::OK, 
MessageBoxIcon::Stop);
                return;
            }
            TMS_Main.D_Accounts[dID]->deductbal(StringToInt(num));
            num = "Name: " + TMS_Main.D_Accounts[dID]->getFName() + " " + 
TMS_Main.D_Accounts[dID]->getLName() + "\nCNIC: " + TMS_Main.D_Accounts[dID]-
>getID() + "\nBalance: PKR " + IntToString(TMS_Main.D_Accounts[dID]->getbal()) + 
"\nNo. of Vehicles: " + IntToString(TMS_Main.D_Accounts[dID]->Vehicles.size()) + 
"\nExperience: " + IntToString(TMS_Main.D_Accounts[dID]->getExp()) + " 
Years\nRating: " + FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) 
+ "\nPassword: " + TMS_Main.D_Accounts[dID]->getPass();
            s = gcnew String(num.data());
            this->richTextBox1->Text = (s);
        }
    }
    private: System::Void Admin_Form_FormClosing(System::Object^ sender, 
System::Windows::Forms::FormClosingEventArgs^ e) {
        TMS_Main.SaveLoadedData();
        Application::Exit();
    }
    private: System::Void button5_Click(System::Object^ sender, 
System::EventArgs^ e) {
        if (this->radioButton1->Checked) {
            if (MessageBox::Show("Are you sure you want to delete this 
Customers account?\nOnce deleted all the orders this customer placed will be 
automatically cancelled", "Confirm Account Deletion", MessageBoxButtons::YesNo, 
MessageBoxIcon::Question) == System::Windows::Forms::DialogResult::Yes) {
                TMS_Main.DeleteCAccount(cID);
                cID = 0;
                this->richTextBox1->Text = "No customers have signed up
yet";
                if (TMS_Main.C_Accounts.size()) {
                    std::string s1 = "Name: " +
TMS_Main.C_Accounts[cID]->getFName() + " " + TMS_Main.C_Accounts[cID]->getLName() +
"\nCNIC: " + TMS_Main.C_Accounts[cID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.C_Accounts[cID]->getbal()) + "\n\nPassword: " +
TMS_Main.C_Accounts[cID]->getPass();
                    System::String^ s2 = gcnew String(s1.data());
                    this->richTextBox1->Text = (s2);
                }
            }
        }
        if (this->radioButton2->Checked) {
            if (MessageBox::Show("Are you sure you want to delete this
Drivers account?\nOnce deleted all the orders placed on this driver will be
automatically rejected", "Confirm Account Deletion", MessageBoxButtons::YesNo,
MessageBoxIcon::Question) == System::Windows::Forms::DialogResult::Yes) {

    TMS_Main.DeleteDAccount(TMS_Main.FindCNIC2(TMS_Main.D_Accounts[dID]-
>getID()));
                dID = 0;
                this->richTextBox1->Text = "No drivers have signed up
yet";
                if (TMS_Main.D_Accounts.size()) {
                    std::string s1 = "Name: " +
TMS_Main.D_Accounts[dID]->getFName() + " " + TMS_Main.D_Accounts[dID]->getLName() +
"\nCNIC: " + TMS_Main.D_Accounts[dID]->getID() + "\nBalance: PKR " +
IntToString(TMS_Main.D_Accounts[dID]->getbal()) + "\nNo. of Vehicles: " +
IntToString(TMS_Main.D_Accounts[dID]->Vehicles.size()) + "\nExperience: " +
IntToString(TMS_Main.D_Accounts[dID]->getExp()) + " Years\nRating: " +
FloatToString(TMS_Main.D_Accounts[dID]->ComputeAndReturnRating()) + "\nPassword: " +
TMS_Main.D_Accounts[dID]->getPass();
                    System::String^ s2 = gcnew String(s1.data());
                    this->richTextBox1->Text = (s2);
                }
            }
        }
    }
    private: System::Void button6_Click(System::Object^ sender,
System::EventArgs^ e) {
        TMS_Main.SortR();
        dID = 0;
        this->radioButton2->Checked = false;
        this->radioButton2->Checked = true;
    }
    private: System::Void button7_Click(System::Object^ sender,
System::EventArgs^ e) {
        TMS_Main.SortE();
        dID = 0;
        this->radioButton2->Checked = false;
        this->radioButton2->Checked = true;
    }
    private: System::Void textBox1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ s1 = gcnew String(this->textBox1->Text);
        std::string s2 = context.marshal_as<std::string>(s1);
        if (!isInt(s2)) {
            this->label4->Text = "Please enter a number.";
        }
        else {
            this->label4->Text = "";
        }
    }
    private: System::Void textBox2_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
        msclr::interop::marshal_context context;
        System::String^ s1 = gcnew String(this->textBox2->Text);
        std::string s2 = context.marshal_as<std::string>(s1);
        if (!isInt(s2)) {
            this->label5->Text = "Please enter a number.";
        }
        else {
            this->label5->Text = "";
        }
    }
    };
}
```

Screenshots



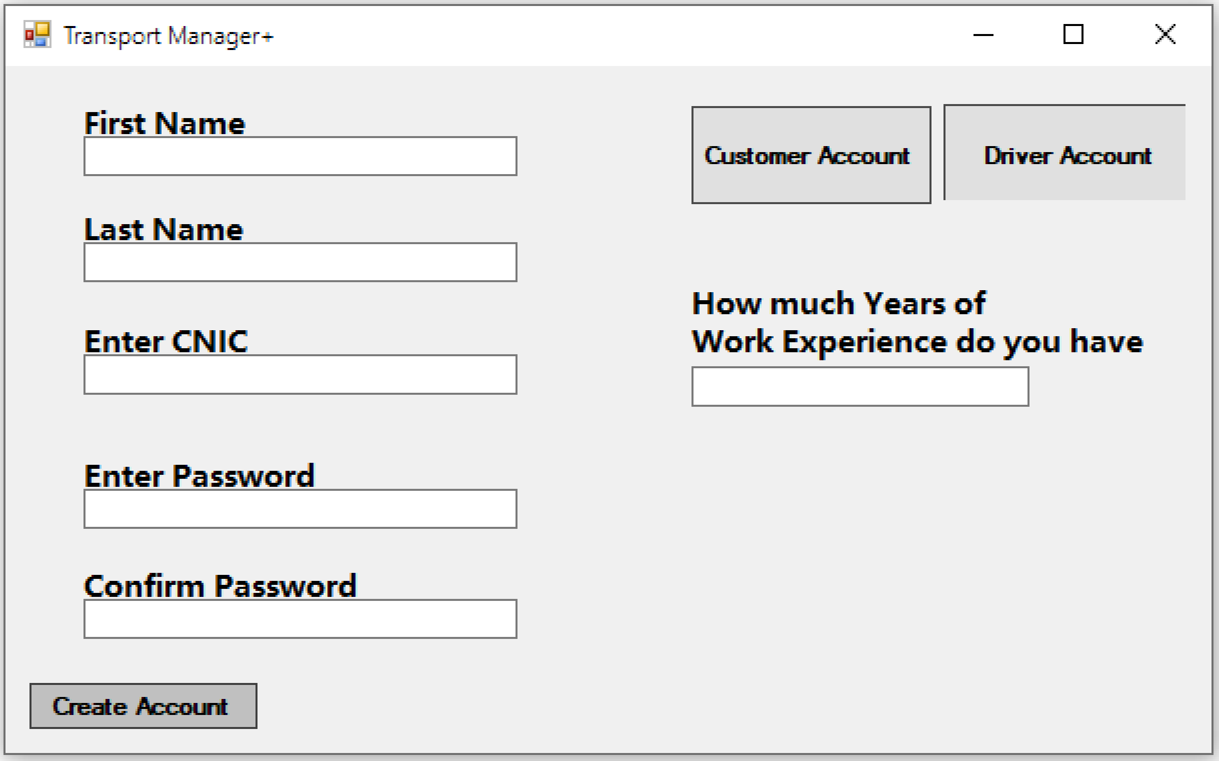
(Please use the Format 12345-1234567-9)

Login Form

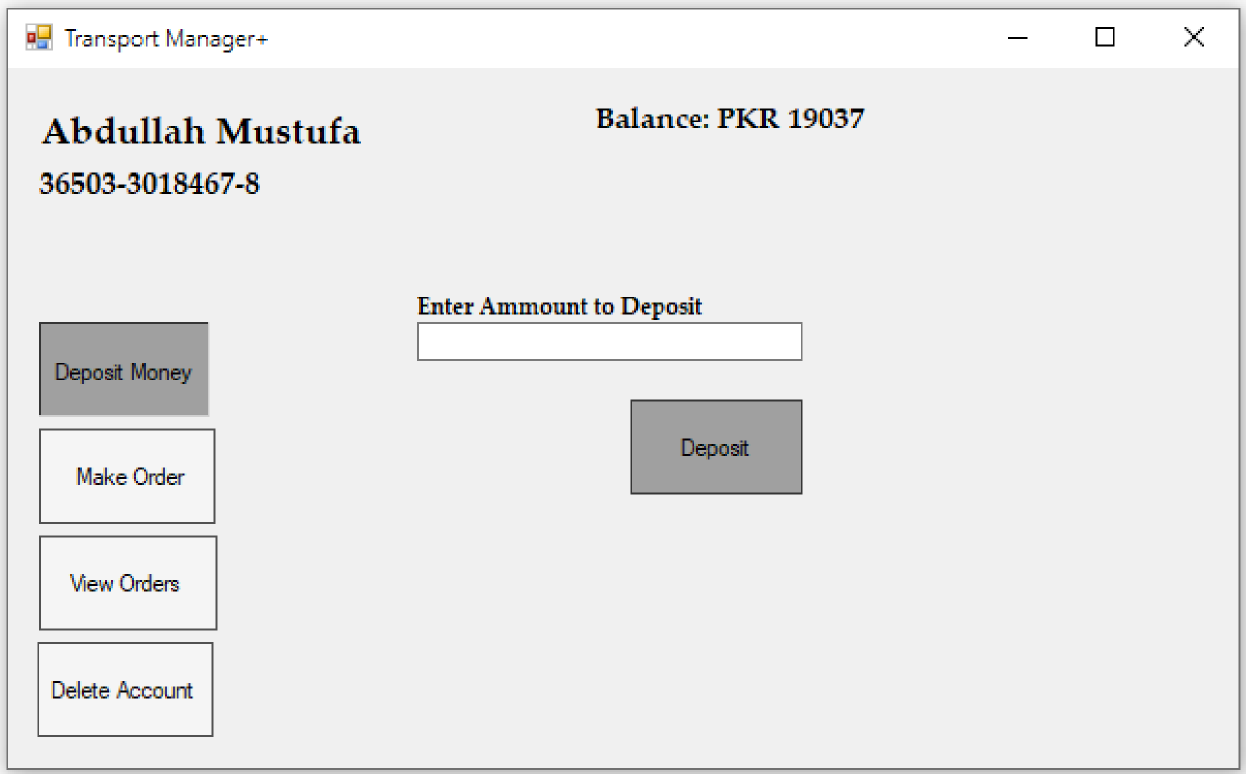


Signup Page (Customer Module)

Customer Page



Signup Page (Driver Module)

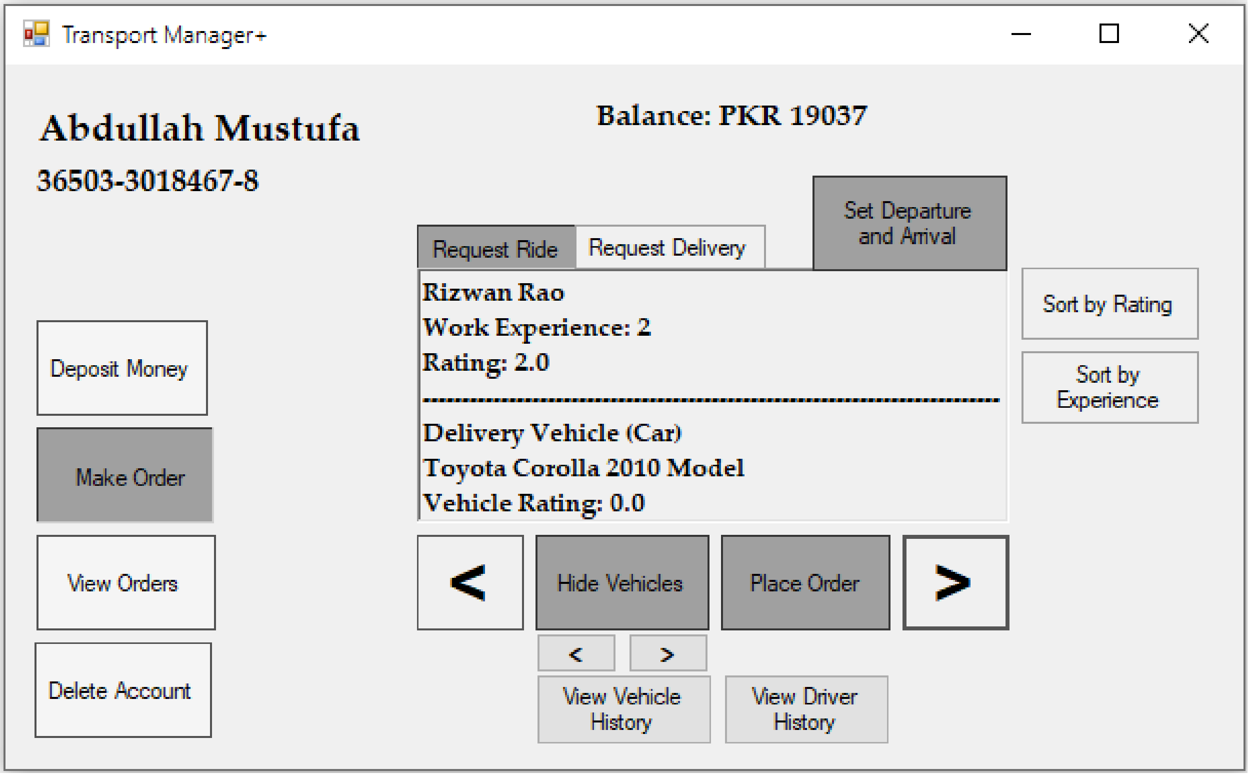


Abdullah Mustufa

36503-3018467-8

Balance: PKR 19037

Enter Ammount to Deposit



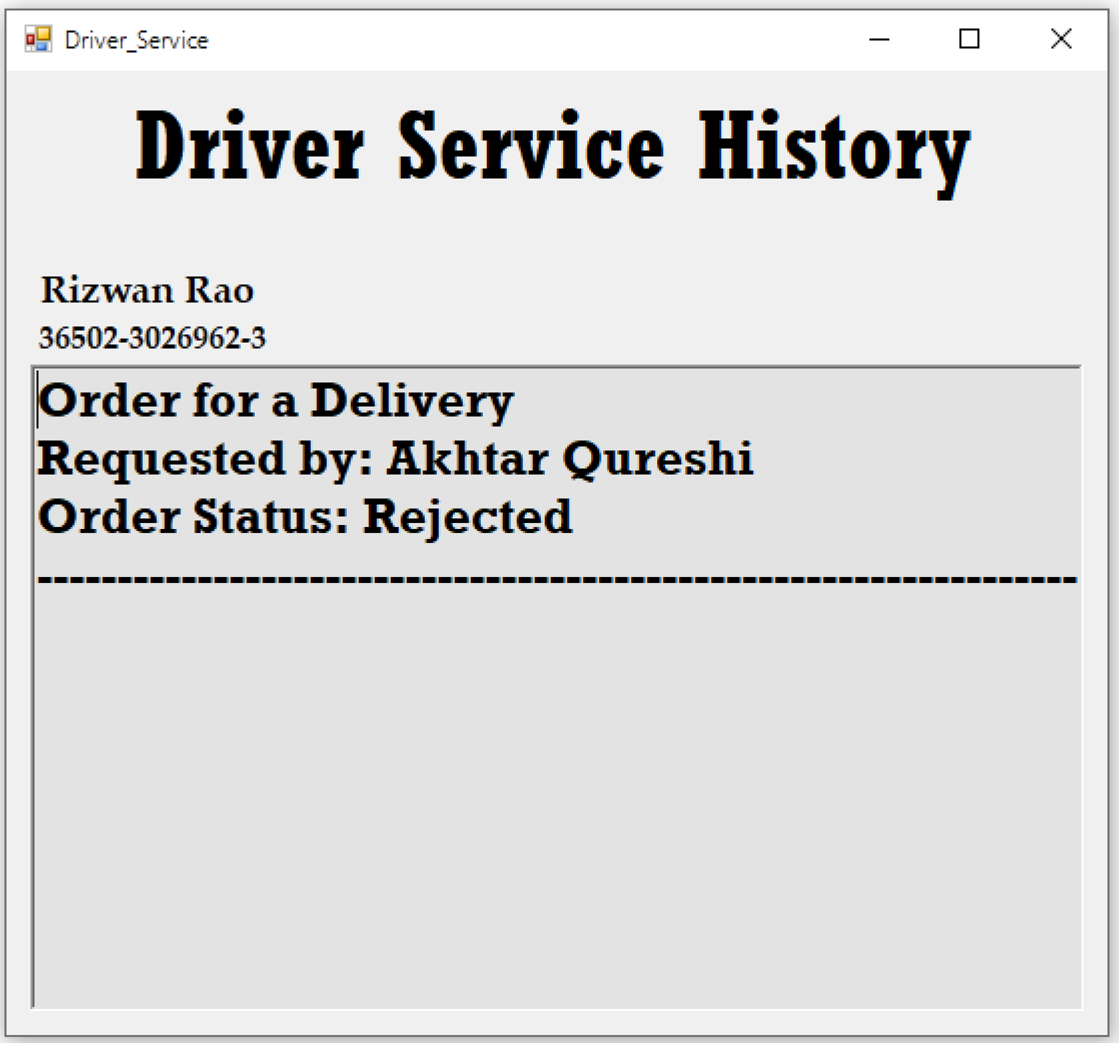
Abdullah Mustufa

36503-3018467-8

Balance: PKR 19037

Rizwan Rao
Work Experience: 2
Rating: 2.0

Delivery Vehicle (Car)
Toyota Corolla 2010 Model
Vehicle Rating: 0.0



Driver Service History

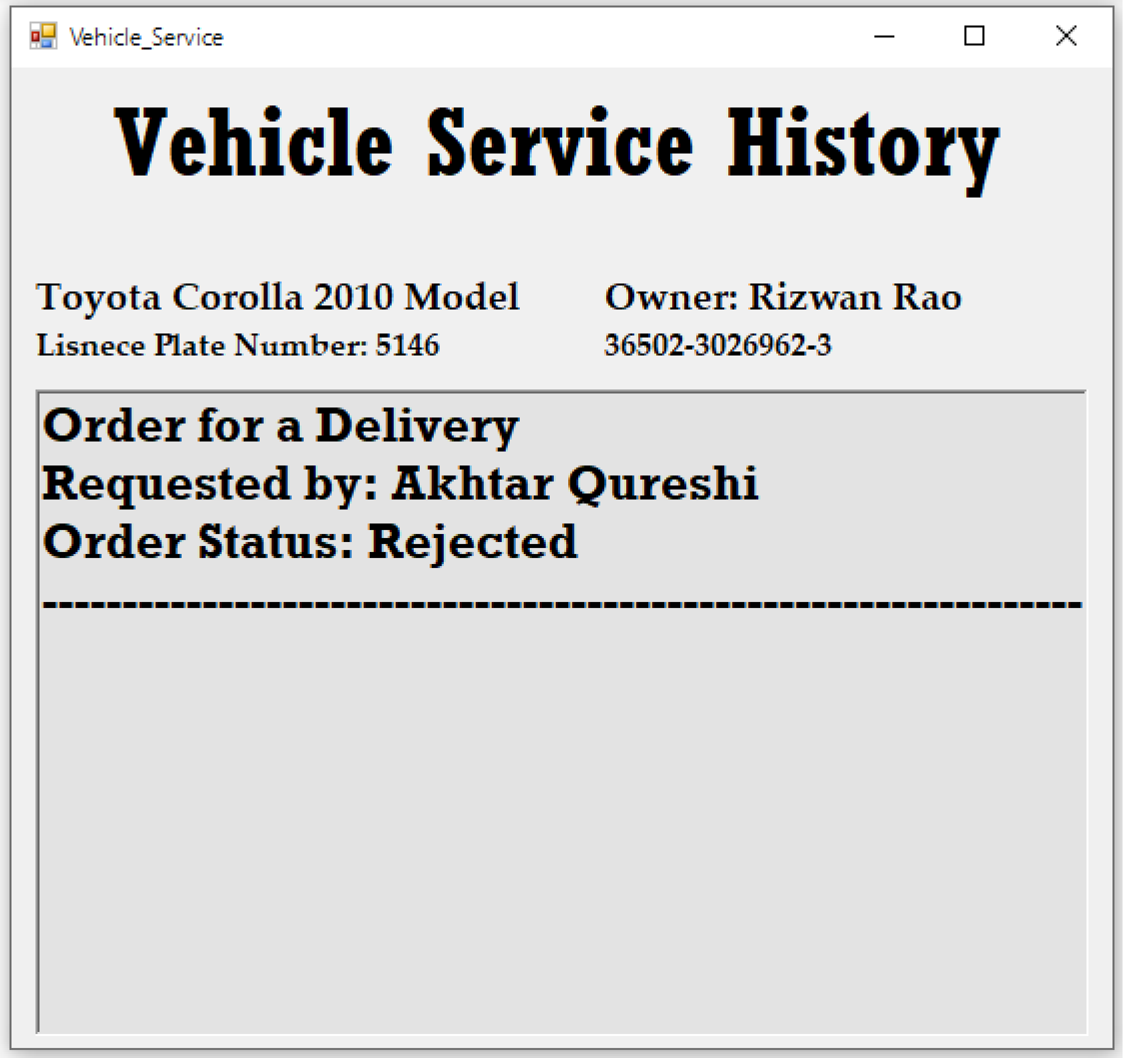
Rizwan Rao

36502-3026962-3

Order for a Delivery

Requested by: Akhtar Qureshi

Order Status: Rejected



Vehicle Service History

Toyota Corolla 2010 Model

Lisnece Plate Number: 5146

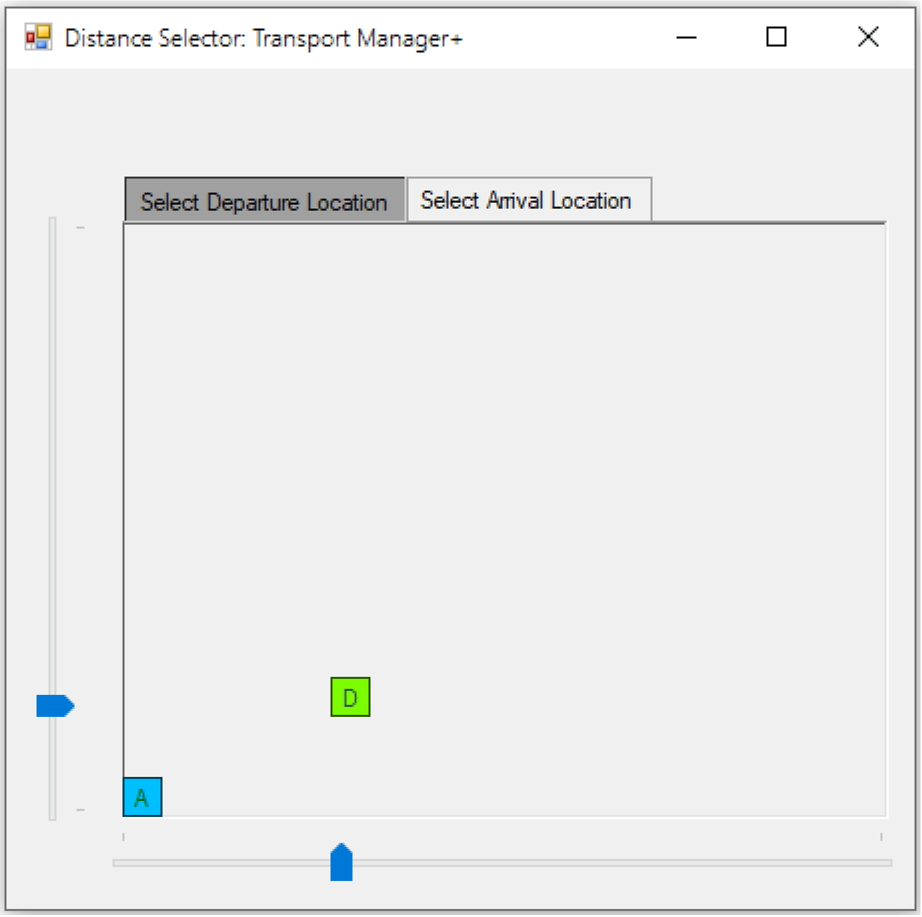
Owner: Rizwan Rao

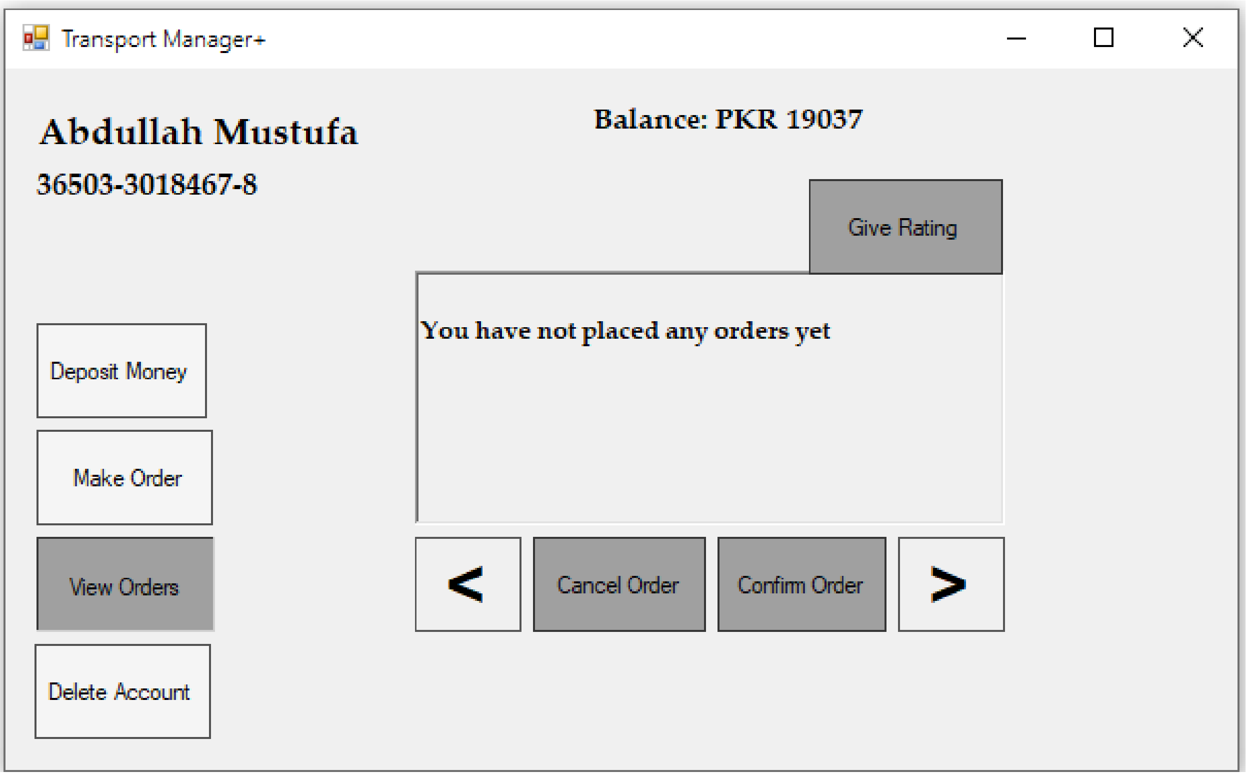
36502-3026962-3

Order for a Delivery

Requested by: Akhtar Qureshi

Order Status: Rejected


Distance Selector: Transport Manager+
Select Departure Location
Select Arrival Location
D
A

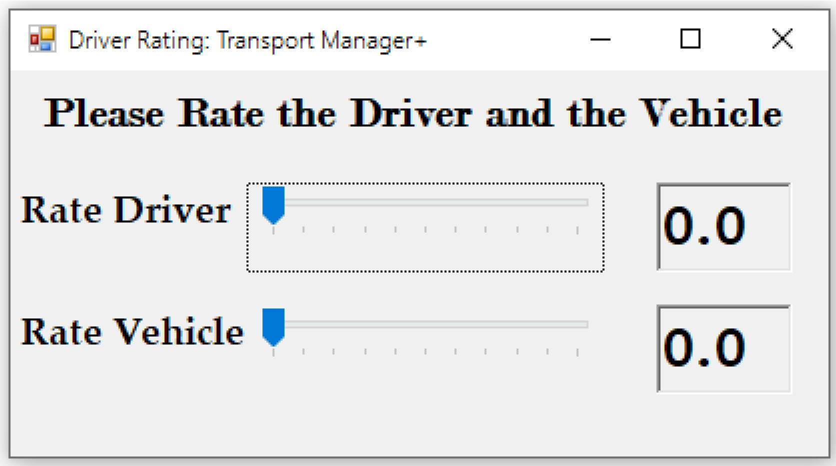


Abdullah Mustufa

36503-3018467-8

Balance: PKR 19037

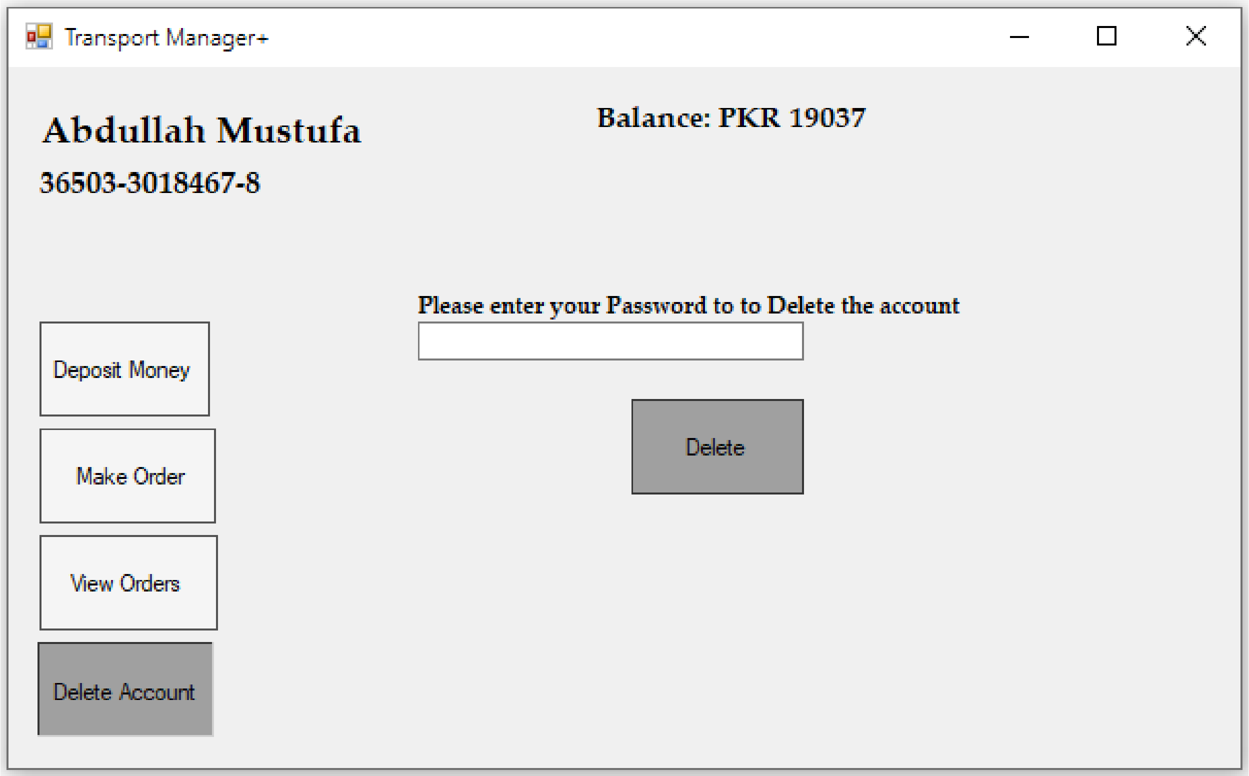
You have not placed any orders yet



Please Rate the Driver and the Vehicle

Rate Driver 0.0

Rate Vehicle 0.0



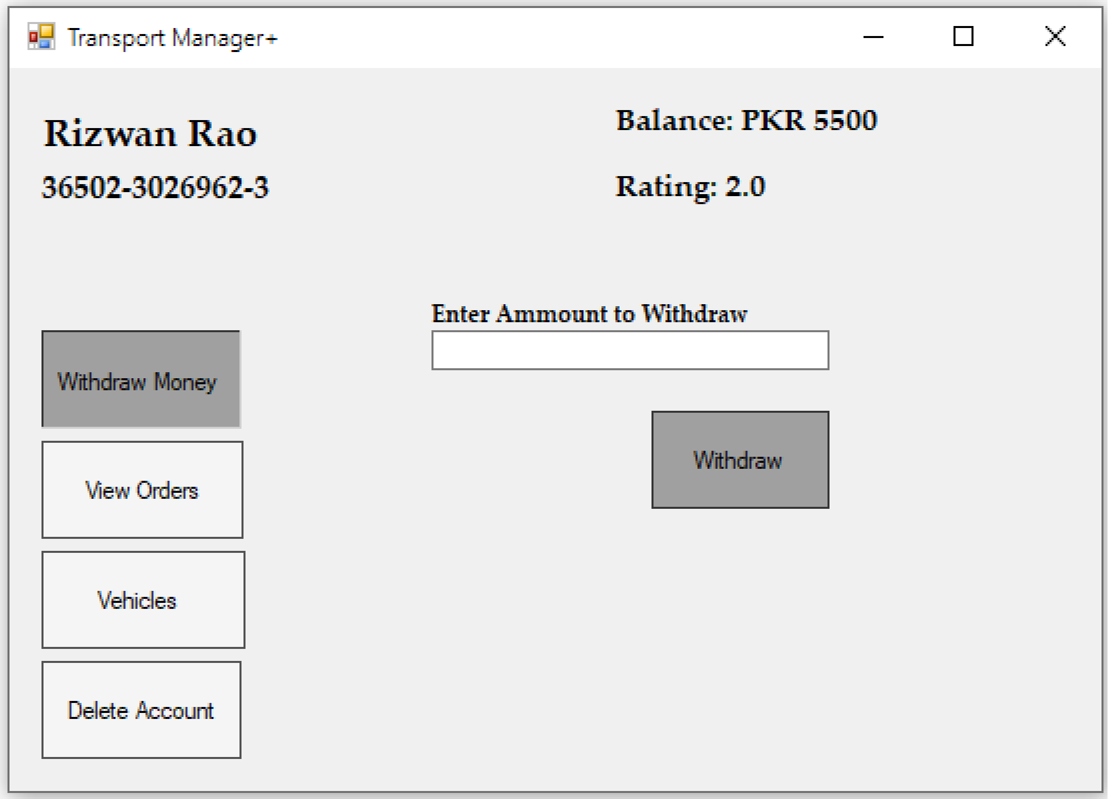
Abdullah Mustufa

36503-3018467-8

Balance: PKR 19037

Please enter your Password to to Delete the account

Driver Page

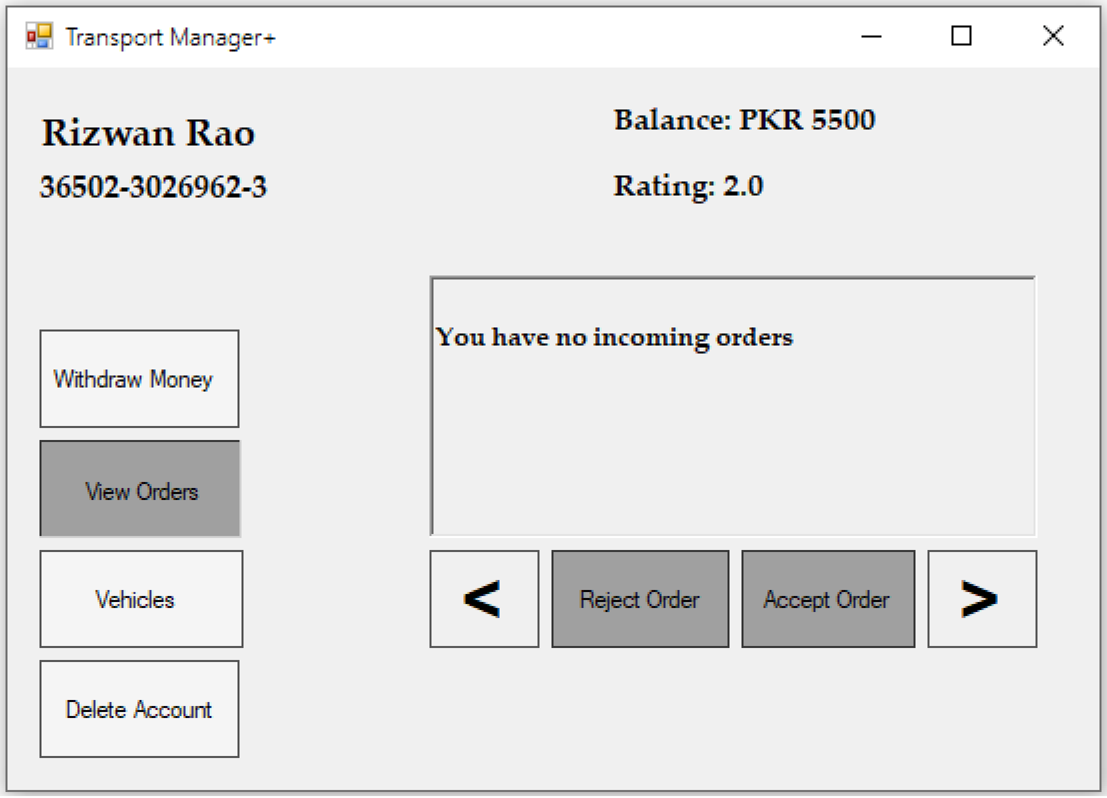


Rizwan Rao

36502-3026962-3

Balance: PKR 5500

Rating: 2.0



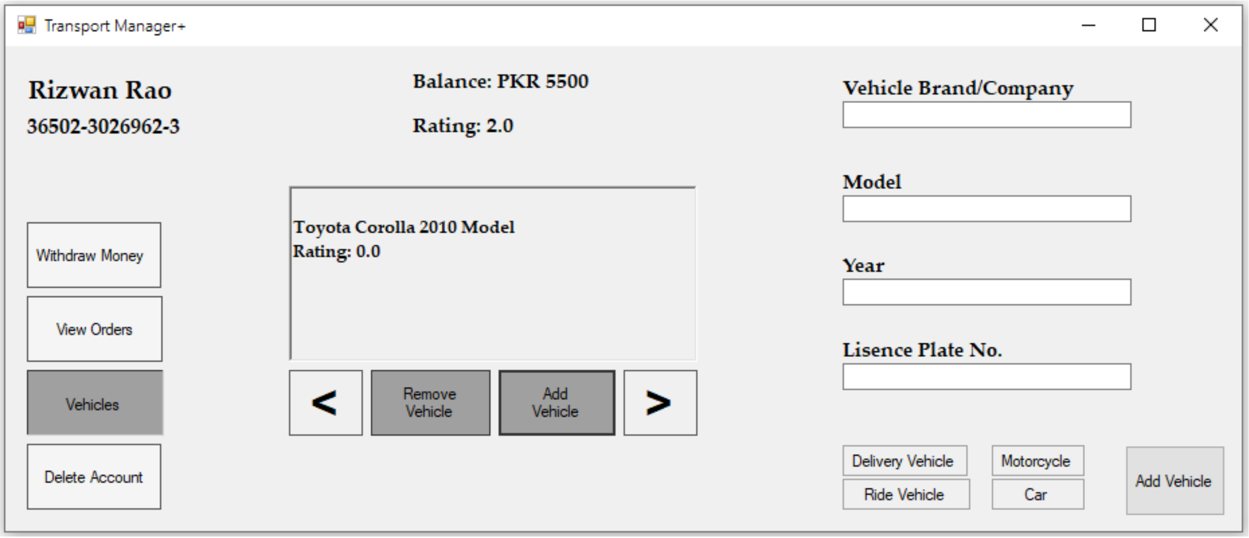
Rizwan Rao

36502-3026962-3

Balance: PKR 5500

Rating: 2.0

You have no incoming orders

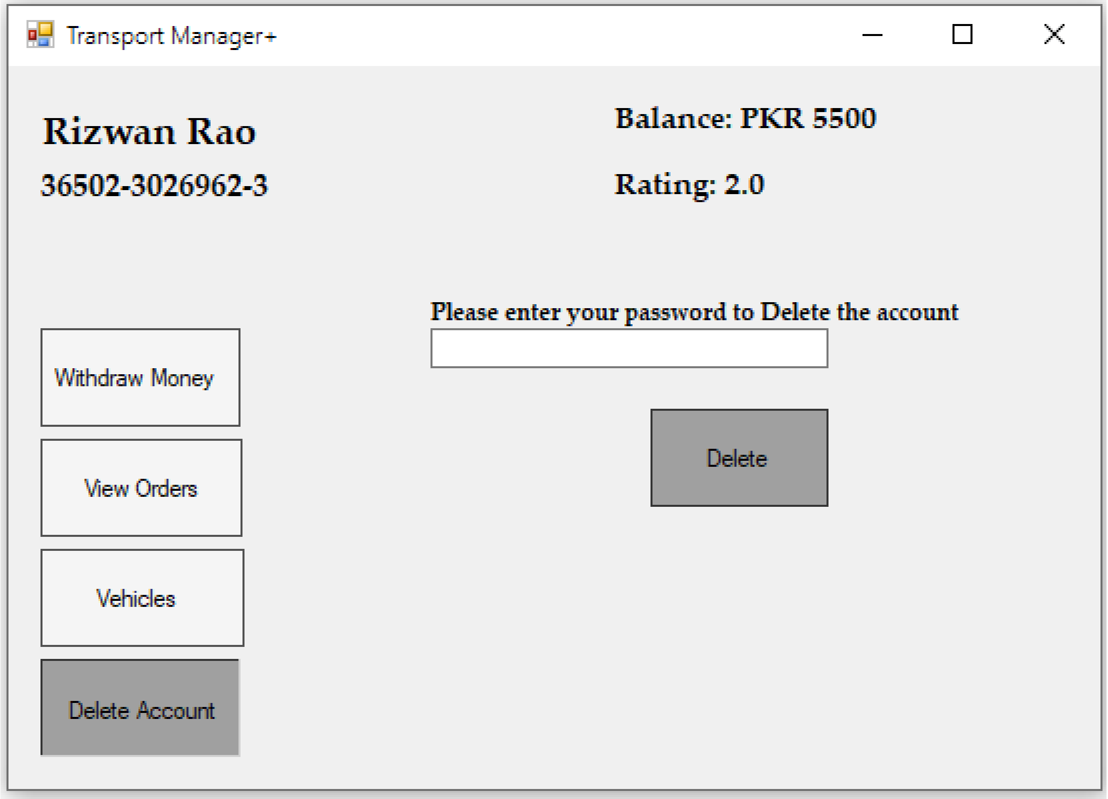


Rizwan Rao
36502-3026962-3

Balance: PKR 5500

Rating: 2.0

Toyota Corolla 2010 Model
Rating: 0.0



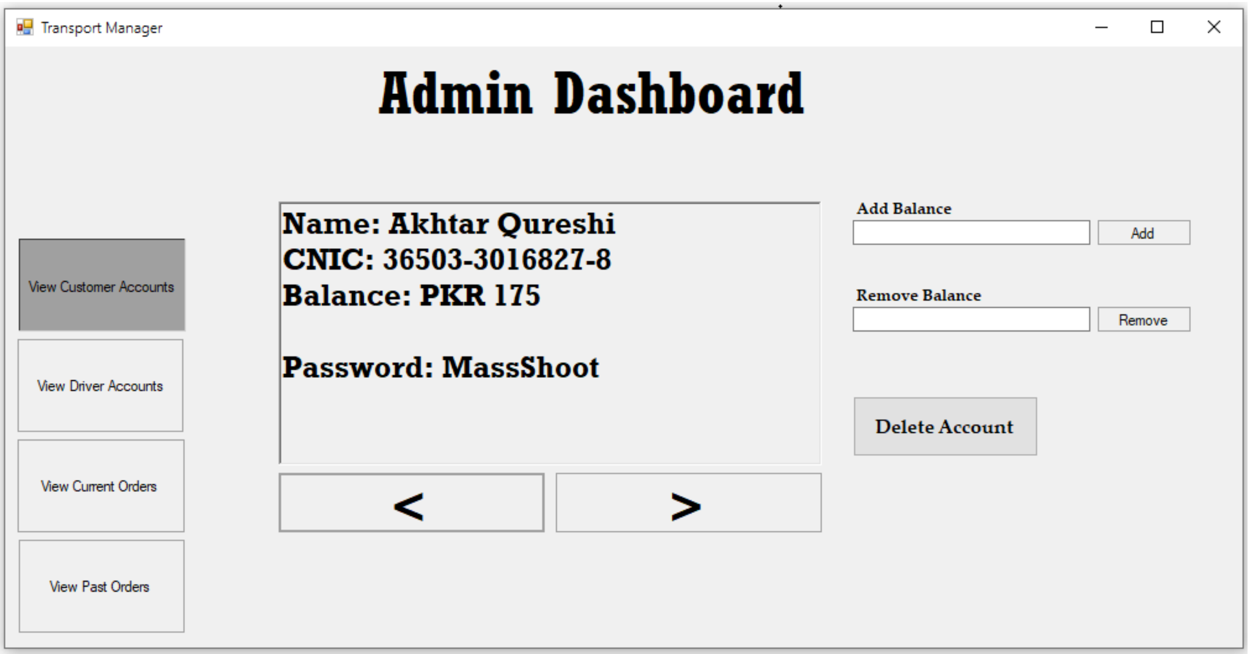
Rizwan Rao
36502-3026962-3

Balance: PKR 5500

Rating: 2.0

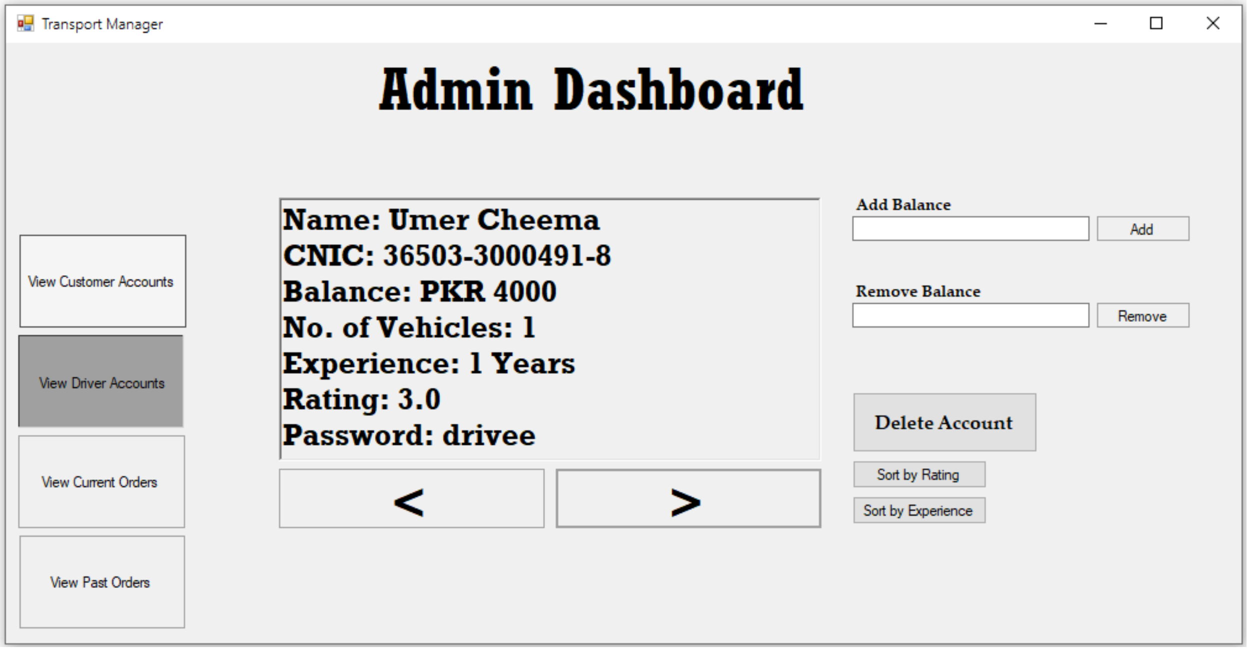
Please enter your password to Delete the account

Admin Dashboard

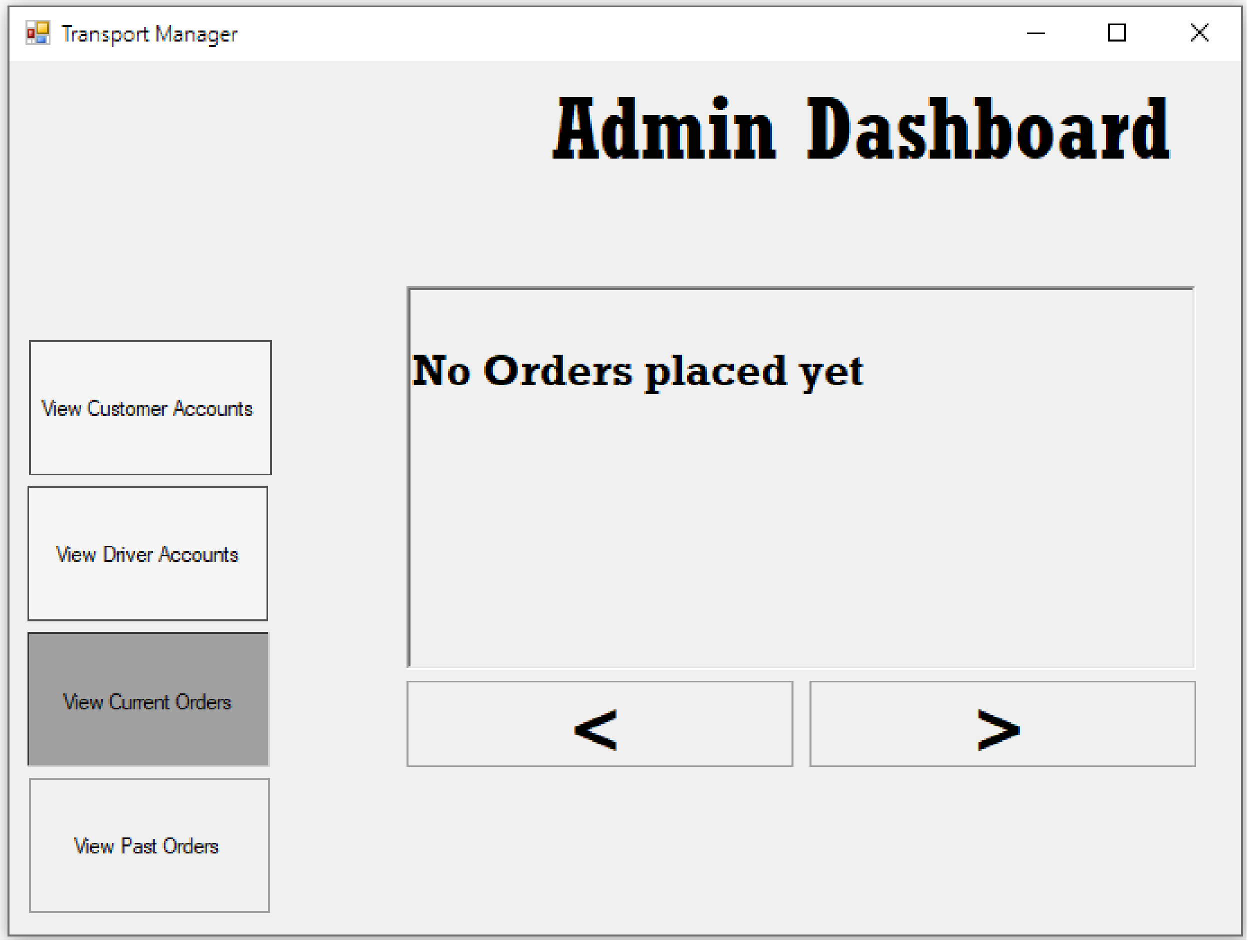


Name: Akhtar Qureshi
CNIC: 36503-3016827-8
Balance: PKR 175

Password: MassShoot

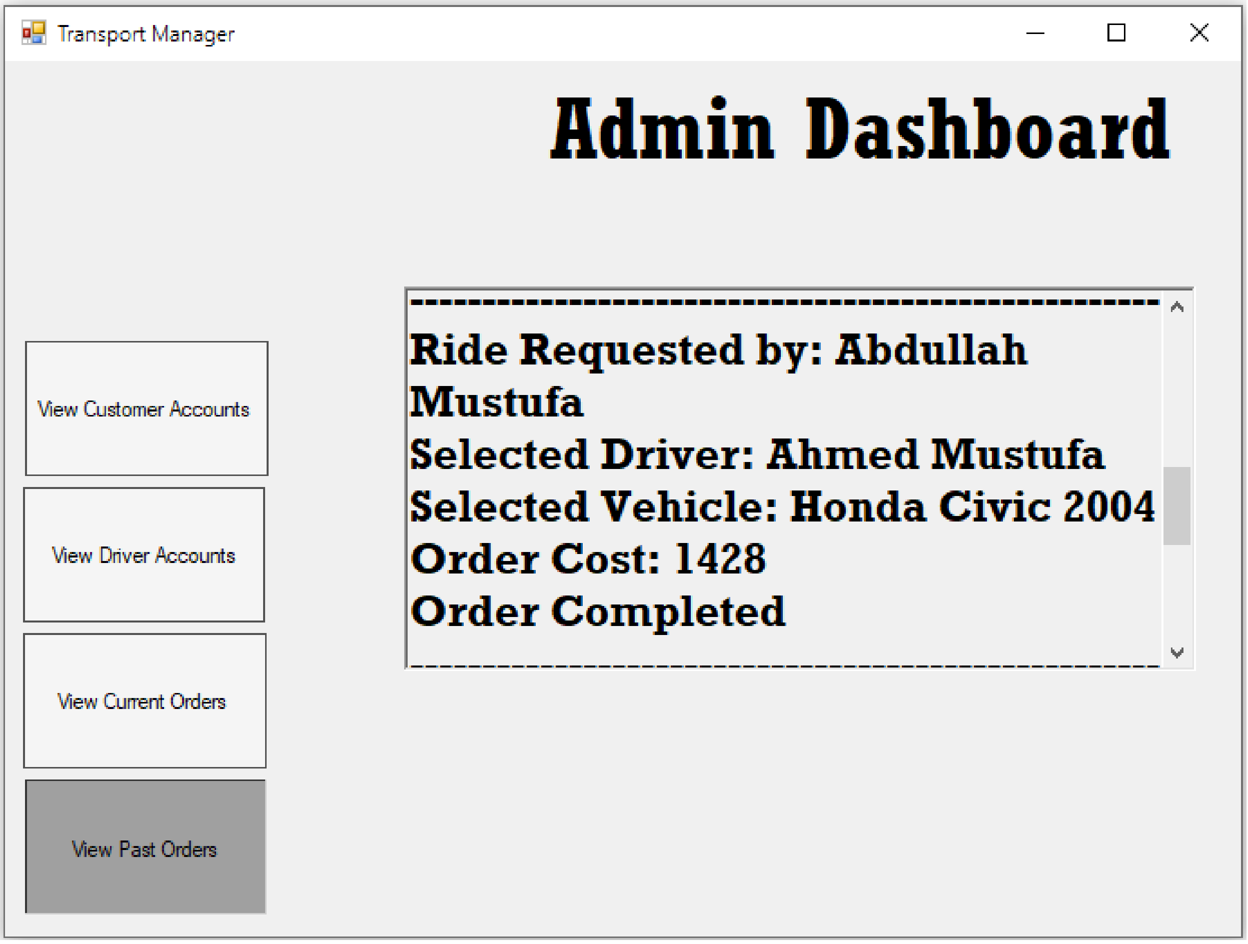


Name: Umer Cheema
CNIC: 36503-3000491-8
Balance: PKR 4000
No. of Vehicles: 1
Experience: 1 Years
Rating: 3.0
Password: drivee



Admin Dashboard

No Orders placed yet



Admin Dashboard

```
------------------------------------------------------
Ride Requested by: Abdullah Mustufa
Selected Driver: Ahmed Mustufa
Selected Vehicle: Honda Civic 2004
Order Cost: 1428
Order Completed
------------------------------------------------------
```