***BLAZE WHEEL***

Database Management System

**Project Report**



Course Code: **CSL-220**

Lab Instructor: **Engr. Ayesha Khan**

Course Instructor: **Engr. Laraib Siddiqui**

**Group Members**

Muhammad Zain **02-131212-077**

Hamza Rajput **02-131212-079**

Muhammad Askari(Group Leader) **02-131212-037**

**Teacher Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Remarks**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submission Date**: 21 June, 2023



**BAHRIA UNIVERSITY**

KARACHI CAMPUS

Department of Software Engineering

**Table of Contents**

[**Introduction:** 3](#_Toc124598174)

[**Technology:** 3](#_Toc124598175)

**Data Manipulation Language**[**:** 3](#_Toc124598182)

**Data Definition Language**[**:** 48](#_Toc124598182)

[**Interfaces:** 52](#_Toc124598182)

[Employee](#_Toc124598183), Technician, and Admin Dashboard[: 52](#_Toc124598184)

[Login Window: 54](#_Toc124598185)

[Vehicle Window: 54](#_Toc124598186)

[Vendor Window: 55](#_Toc124598187)

[Customer Window: 55](#_Toc124598188)

[Employee Window: 56](#_Toc124598189)

[Technician Window: 56](#_Toc124598190)

[Report Window: 57](#_Toc124598190)

[Tech Vehicle Window: 57](#_Toc124598190)

[Sell Window: 58](#_Toc124598190)

[Repair Window: 58](#_Toc124598190)

[Spare Parts Window: 59](#_Toc124598190)

[System Log Window: 59](#_Toc124598190)

[**Conclusion:** 60](#_Toc124598191)

# **Introduction:**

Blaze Wheel, a state-of-the-art vehicle management system designed to streamline and optimize your vehicles operations. Our project focuses on building a strong and effective database infrastructure that acts as the Blaze Wheel system's backbone, guaranteeing flawless data administration.

A strong and scalable architecture specifically designed to handle the complicated requirements of modern vehicle management lies at the basis of Blaze Wheel's database project. We have carefully planned out and developed an organized database schema that effectively arranges and maintains crucial data on automobiles, maintenance logs, maintenance/repair costs, and more.

# **Technology:**

The technology used in making this system is provided in visual studio via windows form application with the use of SQL Database to store all the records which are required to be stored.

# **Data Manipulation Language:**

# Vehicle CRUD:

**Get car:**

create procedure getVehicle

as

begin

select \* from Vehicles

end

**Add car:**

create procedure addVehicle

@Make nvarchar(30),

@Model nvarchar(30),

@Year int,

@Mileage int

as

begin

insert into Vehicles (Make,Model,Year,Mileage,Status,BuyDate) values(@Make,@Model,@Year,@Mileage,'NOT REPAIR',getdate())

end

**Delete car:**

create procedure delVehicle

@VehicleID int

as begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

delete from Vehicles where VehicleID=@VehicleID

end

end

**Add Car altered**

alter procedure addVehicle

@Make nvarchar(30),

@Model nvarchar(30),

@Year int,

@Mileage int,

@Cost int,

@VendorID int

as

begin

insert into Vehicles (Make,Model,Year,Mileage,Status,BuyDate,Cost,VendorID) values(@Make,@Model,@Year,@Mileage,'NOT REPAIR',getdate(),@Cost,@VendorID)

end

**Update Vehicle**

create procedure updateVehicleMake

@VehicleID int,

@Make nvarchar(30)

as

begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

update Vehicles set Make=@Make where VehicleID=@VehicleID

end

end

create procedure updateVehicleModel

@VehicleID int,

@Model nvarchar(30)

as

begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

update Vehicles set Model=@Model where VehicleID=@VehicleID

end

end

create procedure updateVehicleYear

@VehicleID int,

@Year int

as

begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

update Vehicles set Year=@Year where VehicleID=@VehicleID

end

end

create procedure updateVehicleMileage

@VehicleID int,

@Mileage int

as

begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

update Vehicles set Mileage=@Mileage where VehicleID=@VehicleID

end

end

create procedure updateVehicleCost

@VehicleID int,

@Cost int

as

begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

update Vehicles set Cost=@Cost where VehicleID=@VehicleID

end

end

create procedure updateVehicleVendor

@VehicleID int,

@VendorID int

as

begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

update Vehicles set VendorID=@VendorID where VehicleID=@VehicleID

end

end

**Spare Parts CRUD**

create procedure addSparePart

@partName nvarchar(30),

@partDescription nvarchar(100),

@partCost int

as

begin

SET NOCOUNT ON;

insert into SpareParts values (@partName,@partDescription,@partCost)

end

create procedure updateSparePart

@partID int,

@partName nvarchar(30),

@partDescription nvarchar(100),

@partCost int

as

begin

SET NOCOUNT ON;

update SpareParts

set PartName = @partName, PartDescription = @partDescription, PartCost = @partCost where PartID = @partID

end

create procedure deleteSparePart

@partID int

as

begin

SET NOCOUNT ON;

delete from SpareParts where PartID = @partID

end

create procedure readSparePart

as

begin

SET NOCOUNT ON;

select \* from SpareParts

end

**Vendors**

**Add Vendor**

create procedure addVendors

@Name nvarchar(30),

@Address nvarchar(30),

@Phone nvarchar(30)

as

begin

insert into Vendors values(@Name,@Address,@Phone)

end

**Get Vendors**

create procedure getVendors

as

begin

select \* from Vendors

end

**Delete Vendors**

create procedure delVendor

@VendorID int

as begin

if exists (select \* from Vendors where VendorID=@VendorID)

begin

delete from Vendors where VendorID=@VendorID

end

end

**Update vendors**

create procedure updateVendorName

@VendorID int,

@Name nvarchar(30)

as

begin

if exists (select \* from Vendors where VendorID=@VendorID)

begin

update Vendors set Name=@Name where VendorID=@VendorID

end

end

create procedure updateVendorAddress

@VendorID int,

@Address nvarchar(30)

as

begin

if exists (select \* from Vendors where VendorID=@VendorID)

begin

update Vendors set Address=@Address where VendorID=@VendorID

end

end

create procedure updateVendorPhone

@VendorID int,

@Phone nvarchar(30)

as

begin

if exists (select \* from Vendors where VendorID=@VendorID)

begin

update Vendors set Phone=@Phone where VendorID=@VendorID

end

end

**Technicians CRUD**

**Get Technician**

create procedure getTechnician

as

begin

select \* from Technicians

end

**Add Tech**

create procedure addTechnician

@Name nvarchar(30),

@Address nvarchar(50),

@Phone nvarchar(20)

as

begin

insert into Technicians values(@Name,@Address,@Phone)

end

**Delete Tech**

create procedure delTechnician

@TechnicianID int

as

begin

if exists(select \* from Technicians where TechnicianID=@TechnicianID)

begin

delete from Technicians where TechnicianID=@TechnicianID

end

end

**Update Tech**

create procedure updateTechnicianName

@TechnicianID int,

@Name nvarchar(30)

as

begin

if exists (select \* from Technicians where TechnicianID=@TechnicianID)

begin

update Technicians set Name=@Name where TechnicianID=@TechnicianID

end

end

create procedure updateTechnicianAddress

@TechnicianID int,

@Address nvarchar(50)

as

begin

if exists (select \* from Technicians where TechnicianID=@TechnicianID)

begin

update Technicians set Address=@Address where TechnicianID=@TechnicianID

end

end

create procedure updateTechnicianPhone

@TechnicianID int,

@Phone nvarchar(20)

as

begin

if exists (select \* from Technicians where TechnicianID=@TechnicianID)

begin

update Technicians set Phone=@Phone where TechnicianID=@TechnicianID

end

end

**Customer CRUD**

**Add Customer**

create procedure addCustomers

@Name nvarchar(30),

@Address nvarchar(30),

@Phone nvarchar(30)

as

begin

insert into Customers values(@Name,@Address,@Phone)

end

**Get Customers**

create procedure getCustomers

as

begin

select \* from Customers

end

**Delete Customers**

create procedure delCustomer

@CustomerID int

as begin

if exists (select \* from Customers where CustomerID=@CustomerID)

begin

delete from Customers where CustomerID=@CustomerID

end

end

**Update Customers**

create procedure updateCustomerName

@CustomerID int,

@Name nvarchar(30)

as

begin

if exists (select \* from Customers where CustomerID=@CustomerID)

begin

update Customers set Name=@Name where CustomerID=@CustomerID

end

end

create procedure updateCustomerAddress

@CustomerID int,

@Address nvarchar(30)

as

begin

if exists (select \* from Customers where CustomerID=@CustomerID)

begin

update Customers set Address=@Address where CustomerID=@CustomerID

end

end

create procedure updateCustomerPhone

@CustomerID int,

@Phone nvarchar(30)

as

begin

if exists (select \* from Customers where CustomerID=@CustomerID)

begin

update Customers set Phone=@Phone where CustomerID=@CustomerID

end

end

**SpareParts**

create procedure updateSparePartsName

@PartID int,

@PartName nvarchar(30)

as

begin

if exists(select 1 from SpareParts where PartID=@PartID)

begin

update SpareParts set PartName=@PartName where PartID=@PartID

end

end

create procedure updateSparePartsDesc

@PartID int,

@Description nvarchar(100)

as

begin

if exists(select 1 from SpareParts where PartID=@PartID)

begin

update SpareParts set PartDescription=@Description where PartID=@PartID

end

end

create procedure updateSparePartsCost

@PartID int,

@Cost int

as

begin

if exists(select 1 from SpareParts where PartID=@PartID)

begin

update SpareParts set PartCost=@Cost where PartID=@PartID

end

end

**Employee**

**Add Employee**

create procedure addEmployee

@Name nvarchar(30),

@Address nvarchar(30),

@Phone nvarchar(30),

@CommisionPercentage int

as

begin

insert into Employee values(@Name, @Address, @Phone, @CommisionPercentage)

end

**Get Employees**

create procedure getEmployee

as

begin

select\* from Employee

end

**Delete Employees**

create procedure delEmployee

@EmployeeID int

as begin

if exists(select \* from Employee where EmployeeID = @EmployeeID)

begin

delete from Employee where EmployeeID = @EmployeeID

end

end

**Update Employees**

create procedure updateEmployeeName

@EmployeeID int,

@Name nvarchar(30)

as

begin

if exists(select \* from Employee where EmployeeID = @EmployeeID)

begin

update Employee set Name = @Name where EmployeeID = @EmployeeID

end

end

create procedure updateEmployeeCommission

@EmployeeID int,

@CommisionPercentage int

as

begin

if exists(select \* from Employee where EmployeeID = @EmployeeID)

begin

update Employee set CommisionPercentage = @CommisionPercentage where EmployeeID = @EmployeeID

end

end

create procedure updateEmployeeAddress

@EmployeeID int,

@Address nvarchar(30)

as

begin

if exists(select \* from Employee where EmployeeID = @EmployeeID)

begin

update Employee set Address = @Address where EmployeeID = @EmployeeID

end

end

create procedure updateEmployeePhone

@EmployeeID int,

@Phone nvarchar(30)

as

begin

if exists(select \* from Employee where EmployeeID = @EmployeeID)

begin

update Employee set Phone = @Phone where EmployeeID = @EmployeeID

end

end

**Repairs and RepairDetails**

**Get Repair of vehicle**

create procedure getRepair

@TechnicianID int

as

begin

select \* from Repairs where TechnicianID=@TechnicianID

end

**Get RepairDetails of vehicles of a tech**

create procedure getRepairDetails

@TechnicianID int

as

begin

select \* from RepairDetails rd inner join Repairs r on rd.RepairID=r.RepairID where r.TechnicianID=@TechnicianID

end

**Get AvailaleVehicles which are not repaired**

create procedure getAvailaleVehicles

as

begin

select \* from Vehicles where Status='NOT REPAIR'

end

**Technician will select vehicle he want to repair**

create procedure startVehicleRepair

@TechnicianID int,

@VehicleID int

as

begin

insert into Repairs (VehicleID,TechnicianID) values(@VehicleID,@TechnicianID)

update Vehicles set Status='REPAIRING' where VehicleID=@VehicleID

end

**Fixed startVehicleRepair to prevent technician from selecting same car multiple times**

alter procedure startVehicleRepair

@TechnicianID int,

@VehicleID int

as

begin

if not exists(select 1 from Repairs r inner join Technicians t on r.TechnicianID=t.TechnicianID where r.VehicleID=@VehicleID)

begin

insert into Repairs (VehicleID,TechnicianID) values(@VehicleID,@TechnicianID)

update Vehicles set Status='REPAIRING' where VehicleID=@VehicleID

end

end

**Remove Vehicle From Repairs and RepairDetails that Technician is working on**

create procedure VehicleRepairUpdate

@TechnicianID int,

@VehicleID int

as

begin

if exists(select 1 from Repairs r inner join Technicians t on r.TechnicianID=t.TechnicianID where r.VehicleID=@VehicleID)

begin

delete from RepairDetails where RepairID=(select r.RepairID from Repairs r inner join Technicians t on r.TechnicianID=t.TechnicianID where r.VehicleID=@VehicleID)

delete from Repairs where TechnicianID=@TechnicianID

update Vehicles set Status='NOT REPAIR' where VehicleID=@VehicleID

end

end

**Fixed getRepair to only show relevant info to Technician**

alter procedure getRepair

@TechnicianID int

as

begin

select v.VehicleID,v.Make,v.Model,v.Year,v.Status,r.RepairDescription,r.RepairCost,r.TechnicianCharges from Repairs r inner join Vehicles v on r.VehicleID=v.VehicleID where r.TechnicianID=@TechnicianID

end

**Get Vehicles that technician is working on**

create procedure getTechnicianVehicles

@TechnicianID int

as

begin

select v.VehicleID,v.Make,v.Model from Repairs r inner join Vehicles v on r.VehicleID=v.VehicleID where r.TechnicianID=@TechnicianID and v.Status='REPAIRING'

end

**Get Vehicle Repair details**

create procedure getVehicleRepairDetails

@VehicleID int

as

begin

if exists(select 1 from RepairDetails rd inner join Repairs r on rd.RepairID=r.RepairID where r.VehicleID=@VehicleID)

begin

select rd.rdID as [RepairDetail ID],sp.PartName,sp.PartDescription,sp.PartCost,rd.Quantity,sum(sp.PartCost\*rd.Quantity) as Total from

SpareParts sp inner join RepairDetails rd on sp.PartID=rd.PartID

inner join Repairs r on rd.RepairID=r.RepairID

where r.VehicleID=@VehicleID group by rd.rdID,sp.PartName,sp.PartCost,rd.Quantity,sp.PartDescription

end

end

**Update Repair Detail**

create procedure updateRepairDetail

@rdID int,

@Quantity int

as

begin

update RepairDetails set Quantity=@Quantity where rdID=@rdID

end

**Delete From Repair Details**

create procedure deleteRepairDetail

@rdID int

as

begin

delete from RepairDetails where rdID=@rdID

end

**Insert into Repair Detils when part is inserted**

create procedure insertPartRepairDetail

@VehicleID int,

@PartID int,

@Quantity int

as

begin

insert into RepairDetails values((select RepairID from Repairs where VehicleID=@VehicleID),@PartID,@Quantity)

end

**When vehicle is repaired all the charges are calculated**

create procedure MarkVehicleRepaired

@VehicleID int,

@Description nvarchar(100),

@Charges int

as

begin

update Repairs set RepairDescription=@Description,

RepairCost=(select top 1 sum(sp.PartCost\*rd.Quantity) as Total from

SpareParts sp inner join RepairDetails rd on sp.PartID=rd.PartID

inner join Repairs r on rd.RepairID=r.RepairID

where r.VehicleID=2014 group by rollup(sp.PartCost,rd.Quantity) order by Total desc),

CompletionDate=getdate(),

TechnicianCharges=@Charges

where VehicleID=@VehicleID

update Vehicles set Cost=(select RepairCost+TechnicianCharges from Repairs where VehicleID=@VehicleID), Status='REPAIRED' where VehicleID=@VehicleID

end

alter procedure MarkVehicleRepaired

@VehicleID int,

@Description nvarchar(100),

@Charges int

as

begin

update Repairs set RepairDescription=@Description,

RepairCost=(select top 1 sum(sp.PartCost\*rd.Quantity) as Total from

SpareParts sp inner join RepairDetails rd on sp.PartID=rd.PartID

inner join Repairs r on rd.RepairID=r.RepairID

where r.VehicleID=@VehicleID group by rollup(sp.PartCost,rd.Quantity) order by Total desc)+@Charges,

CompletionDate=getdate(),

TechnicianCharges=@Charges

where VehicleID=@VehicleID

update Vehicles set Status='REPAIRED' where VehicleID=@VehicleID

end

alter procedure MarkVehicleRepaired

@VehicleID int,

@Description nvarchar(100),

@Charges int

as

begin

update Repairs set RepairDescription=@Description,

RepairCost=(select top 1 sum(sp.PartCost\*rd.Quantity) as Total from

SpareParts sp inner join RepairDetails rd on sp.PartID=rd.PartID

inner join Repairs r on rd.RepairID=r.RepairID

where r.VehicleID=@VehicleID group by rollup(sp.PartCost,rd.Quantity) order by Total desc)+@Charges,

CompletionDate=getdate(),

TechnicianCharges=@Charges

where VehicleID=@VehicleID

update Vehicles set Status='REPAIRED' where VehicleID=@VehicleID

end

**Fixing anomsly:**

**Vehicle Delete ultimate**

alter procedure delVehicle

@VehicleID int

as begin

if exists (select \* from Vehicles where VehicleID=@VehicleID)

begin

delete from RepairDetails where RepairID=(select RepairID from Repairs where VehicleID=@VehicleID)

delete from Repairs where VehicleID=@VehicleID

delete from Vehicles where VehicleID=@VehicleID

end

end

**Vendor delete ultimate**

alter procedure delVendor

@VendorID int

as begin

if exists (select \* from Vendors where VendorID=@VendorID)

begin

delete from RepairDetails where RepairID=(select RepairID from Repairs where VehicleID=(select VehicleID from Vehicles where VendorID=@VendorID))

delete from Repairs where VehicleID=(select VehicleID from Vehicles where VendorID=@VendorID)

delete from Vehicles where VendorID=@VendorID

delete from Vendors where VendorID=@VendorID

end

end

**Technician delete ultimate**

alter procedure delTechnician

@TechnicianID int

as

begin

if exists(select \* from Technicians where TechnicianID=@TechnicianID)

begin

update Vehicles set Status='NOT REPAIR' where VehicleID=(select VehicleID from Repairs where TechnicianID=@TechnicianID)

delete from RepairDetails where RepairID=(select RepairID from Repairs where TechnicianID=@TechnicianID)

delete from Repairs where TechnicianID=@TechnicianID

delete from Technicians where TechnicianID=@TechnicianID

end

end

end

alter procedure addEmployee

@Name nvarchar(30),

@Address nvarchar(30),

@Phone nvarchar(30),

@CommisionPercentage int,

@Password varchar(100)

as begin

insert into Employee values(@Name, @Address, @Phone, @CommisionPercentage, 'INACTIVE', @Password)

end

create procedure updateEmployeePassword

@EmployeeID int,

@Password varchar(100)

as begin

update Employee set Password = @Password where EmployeeID = @EmployeeID

end

alter procedure addTechnician

@Name nvarchar(30),

@Address nvarchar(50),

@Phone nvarchar(20),

@Password varchar(100)

as

begin

insert into Technicians values(@Name,@Address,@Phone,'INACTIVE',@Password)

end

create procedure updateTechnicianPassword

@TechnicianID int,

@Password varchar(100)

as

begin

update Technicians set Password = @Password where TechnicianID = @TechnicianID

end

**Trigger**

create TRIGGER TrSessionEmp ON Employee

AFTER UPDATE

AS

BEGIN

IF UPDATE(SessionStatus)

BEGIN

DECLARE @NewSessionStatus VARCHAR(10)

DECLARE @OldSessionStatus VARCHAR(10)

SELECT @NewSessionStatus = SessionStatus

FROM inserted

SELECT @OldSessionStatus = SessionStatus

FROM deleted

-- Check if SessionStatus changed to 'ACTIVE' or 'INACTIVE'

IF (@NewSessionStatus = 'ACTIVE' and @OldSessionStatus = 'INACTIVE')

AND (@OldSessionStatus != @NewSessionStatus)

BEGIN

INSERT INTO SessionLog (Usertype, UserID, Username, LoginTime)

SELECT 'Employee',i.EmployeeID,i.Name, GETDATE()

FROM inserted i

END

IF (@NewSessionStatus = 'INACTIVE' and @OldSessionStatus = 'ACTIVE')

AND (@OldSessionStatus != @NewSessionStatus)

BEGIN

update SessionLog

set LogoutTime = GETDATE() where Usertype = 'Employee' and UserID in (select UserID from inserted)

and LogoutTime is null

END

END

END

**Technician login/logout trigger**

create TRIGGER TrSessionTech ON Technicians

AFTER UPDATE

AS

BEGIN

IF UPDATE(SessionStatus)

BEGIN

DECLARE @NewSessionStatus VARCHAR(10)

DECLARE @OldSessionStatus VARCHAR(10)

SELECT @NewSessionStatus = SessionStatus

FROM inserted

SELECT @OldSessionStatus = SessionStatus

FROM deleted

-- Check if SessionStatus changed to 'ACTIVE' or 'INACTIVE'

IF (@NewSessionStatus = 'ACTIVE' and @OldSessionStatus = 'INACTIVE')

AND (@OldSessionStatus != @NewSessionStatus)

BEGIN

INSERT INTO SessionLog (Usertype, UserID, Username, LoginTime)

SELECT 'Technician',i.TechnicianID,i.Name, GETDATE()

FROM inserted i

END

IF (@NewSessionStatus = 'INACTIVE' and @OldSessionStatus = 'ACTIVE')

AND (@OldSessionStatus != @NewSessionStatus)

BEGIN

update SessionLog

set LogoutTime = GETDATE() where Usertype = 'Technician' and UserID in (select UserID from inserted)

and LogoutTime is null

END

END

END

**Admin login/logout**

create TRIGGER TrSessionAdmin ON Admin

AFTER UPDATE

AS

BEGIN

IF UPDATE(SessionStatus)

BEGIN

DECLARE @NewSessionStatus VARCHAR(10)

DECLARE @OldSessionStatus VARCHAR(10)

SELECT @NewSessionStatus = SessionStatus

FROM inserted

SELECT @OldSessionStatus = SessionStatus

FROM deleted

-- Check if SessionStatus changed to 'ACTIVE' or 'INACTIVE'

IF (@NewSessionStatus = 'ACTIVE' and @OldSessionStatus = 'INACTIVE')

AND (@OldSessionStatus != @NewSessionStatus)

BEGIN

INSERT INTO SessionLog (Usertype, UserID, Username, LoginTime)

SELECT 'Administrator',i.AdminID,i.Name, GETDATE()

FROM inserted i

END

IF (@NewSessionStatus = 'INACTIVE' and @OldSessionStatus = 'ACTIVE')

AND (@OldSessionStatus != @NewSessionStatus)

BEGIN

update SessionLog

set LogoutTime = GETDATE() where Usertype = 'Administrator' and UserID in (select UserID from inserted)

and LogoutTime is null

END

END

END

**Customer**

create trigger TrAuditCust on Customers

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Customer',D.CustomerID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Customer',D.CustomerID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Customer',I.CustomerID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

**RD**

create trigger TrAuditRD on RepairDetails

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'RepairDetails',D.rdID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'RepairDetails',D.rdID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'RepairDetails',I.rdID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

create trigger TrAuditRepair on Repairs

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Repairs',D.RepairID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Repairs',D.RepairID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Repairs',I.RepairID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

**Create trigger TrAuditSales on Sale**

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Sale',D.SaleID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Sale',D.SaleID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Sale',I.SaleID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

**Create trigger TrAuditSP on SpareParts**

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'SpareParts',D.PartID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'SpareParts',D.PartID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'SpareParts',I.PartID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

**Create trigger TrAuditVehicle on Vehicles**

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Vehicles',D.VehicleID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Vehicles',D.VehicleID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Vehicles',I.VehicleID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

**Create trigger TrAuditVendors on Vendors**

after insert, update, delete

as

declare @login\_name varchar(128)

SELECT @login\_name = Username

FROM SessionLog

WHERE LogoutTime is null and LoginTime is not null

if exists ( select 0 from Deleted )

begin

if exists ( select 0 from Inserted )

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Vendors',D.VendorID , @login\_name, GETDATE(),'UPDATION'

from deleted D

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Vendors',D.VendorID , @login\_name, GETDATE(),'DELETION' from deleted D

end

end

else

begin

insert into TablesAudit

( TableName,

TableID,

ModifiedBy,

ModifiedDate,

Operation

)

select 'Vendors',I.VendorID , @login\_name, GETDATE(),'ADDITION' from inserted I

end

go

**Sell**

**Load Employees Sell**

create procedure getEmplyeeSells

@EmployeeID int

as

begin

select s.SaleID,v.Make,v.Model,c.Name as Customer,s.SalePrice,s.Commission,s.profit,s.SaleDate from Sale s inner join Vehicles v on s.VehicleID=v.VehicleID inner join Customers c on c.CustomerID=s.CustomerID where s.EmployeeID=@EmployeeID

end

**Load customers in CB**

create procedure readCustomers

as

begin

select CustomerID,Name from Customers

end

**Load repaired vehicle in CB**

create procedure readRepairedCar

as

begin

select VehicleID,Model from Vehicles where Status='REPAIRED'

end

**Get Employee Commission Perc**

create procedure getEmployeeComPerc

@EmployeeID int

as

begin

select CommisionPercentage from Employee where EmployeeID=@EmployeeID

end

**Get Vehicle Cost**

create procedure getVehicleCost

@VehicleID int

as

begin

select Cost from Vehicles where VehicleID=@VehicleID

end

**Get vehicle repair cost**

create procedure getVehicleRepairCost

@VehicleID int

as

begin

select r.RepairCost from Vehicles v inner join Repairs r on v.VehicleID=r.VehicleID where v.VehicleID=@VehicleID

end

**Fixed get repair details for Technician**

alter procedure getRepairDetails

@TechnicianID int

as

begin

select rd.rdID as [Repair Detail ID],v.Model,sp.PartName,sp.PartCost,rd.Quantity, (sp.PartCost\*rd.Quantity) as Total from RepairDetails rd inner join Repairs r on rd.RepairID=r.RepairID inner join SpareParts sp on sp.PartID=rd.PartID inner join Vehicles v on r.VehicleID=v.VehicleID where r.TechnicianID=@TechnicianID

end

**Sell Caralter procedure sellCar**

@EmployeeID int,

@VehicleID int,

@CustomerID int,

@SalePrice int,

@Commission int,

@profit int

as

begin

insert into Sale values(@VehicleID,@CustomerID,@SalePrice,getdate(),@profit,@EmployeeID,@Commission)

update Vehicles set SaleDate=getdate(), Status='SOLD' where VehicleID=@VehicleID

end

**Delete Sale**

create procedure deleteSale

@SaleID int,

@EmployeeID int

as

begin

if exists(select 1 from Sale where SaleID=@SaleID and EmployeeID=@EmployeeID)

begin

declare @VehicleID int;

set @VehicleID=(select VehicleID from Sale where SaleID=@SaleID)

update Vehicles set SaleDate=NULL, Status='REPAIRED' where VehicleID=@VehicleID

delete from Sale where SaleID=@SaleID

end

end

**Data Definition Language:**

create table Customers(

CustomerID int primary key identity(1,1),

Name nvarchar(30) not null,

Address nvarchar(30),

Phone nvarchar(30) not null

)

create table Vendors(

VendorID int primary key identity(1,1),

Name nvarchar(30) not null,

Address nvarchar(30),

Phone nvarchar(30) not null,

)

create table Vehicles(

VehicleID int primary key identity(1,1),

Make nvarchar(30) not null,

Model nvarchar(30) not null,

Year int not null,

Mileage int not null,

Status varchar(10) check (Status IN ('NOT REPAIR','REPAIRED','SOLD')),

VendorID int,

foreign key(VendorID) references Vendors(VendorID)

)

create table Technicians(

TechnicianID int primary key identity(1,1),

Name nvarchar(30) not null

)

create table SpareParts(

PartID int primary key identity(1,1),

PartName nvarchar(30)not null,

PartDescription nvarchar(100),

PartCost int not null

)

create table Repairs(

RepairID int primary key identity(1,1),

VehicleID int,

TechnicianID int,

foreign key(VehicleID) references Vehicles(VehicleID),

foreign key(TechnicianID) references Technicians(TechnicianID),

RepairDescription nvarchar(100),

RepairCost int not null

)

create table RepairParts(

RepairID int primary key identity(1,1),

PartID nvarchar(10),

foreign key(RepairID) references Repairs(RepairID)

)

create table RepairCost(

RCostID int primary key identity(1,1),

RepairID int,

VehicleID int,

foreign key(RepairID) references Repairs(RepairID),

foreign key(VehicleID) references Vehicles(VehicleID),

TotalRepairCost int

)

create table Sale(

SaleID int primary key identity(1,1),

VehicleID int,

CustomerID int,

SalePrice int not null,

SaleDate date,

profit int

foreign key(VehicleID) references Vehicles(VehicleID),

foreign key(CustomerID) references Customers(CustomerID)

)

create table Employee(

EmployeeID int primary key identity(1,1),

Name nvarchar

)

**Changes**

alter table Vendors add

CostPrice int

drop table RepairParts

alter table Technicians add

RepairCharges int

alter table Employee add

Address nvarchar(50),

Phone nvarchar(12),

CommisionPercentage int

alter table Sale add

EmployeeID int

Foreign key (EmployeeID) references Employee(EmployeeID)

drop table RepairCost

create table RepairDetails(

rdID int primary key identity(1,1),

RepairID int,

PartID int,

foreign key (RepairID) references Repairs(RepairID),

foreign key (PartID) references SpareParts(PartID)

)

alter table Vehicles add

BuyDate datetime,

SaleDate datetime

alter table Repairs add

CompletionDate datetime

create table SystemLog(

logID int primary key identity(1,1),

TableName nvarchar(15),

Operation nvarchar(15)

)

alter table SystemLog add

ModifiedBy nvarchar(128),

ModifiedDate datetime

**Cost price of car added in Vehicles and removed from Vendors**

alter table Vehicles add

Cost int

alter table Vendors drop column CostPrice

**RepairCharges of technicians moved to Repairs from Technicians bc each repair has different repair charges**

alter table Technicians drop column RepairCharges

alter table Repairs add

TechnicianCharges int

**Some more details added for technicians**

alter table Technicians add

Address nvarchar(50),

Phone nvarchar(15)

alter table Employee alter column Name nvarchar(30)

**Added Quantity to RepairDetails**

**alter table RepairDetails add Quantity int**

**Status constraint changed in Vehicles**

alter table Vehicles add

Status varchar(10) check (Status IN ('NOT REPAIR','REPAIRING','REPAIRED','SOLD'))

**RepairStatus removed from Repairs**

alter table Repairs drop column RepairStatus

**Repair table allowing null value in RepairCost**

alter table Repairs alter column RepairCost int

alter table Employee

add SessionStatus varchar(10) check (SessionStatus IN ('ACTIVE','INACTIVE')), Password varchar(100)

create table Admin(

Name varchar(30),

Password varchar(100),

SessionStatus varchar(10) check (SessionStatus IN ('ACTIVE','INACTIVE')),

)

alter table Customers

drop column SessionStatus, Password

alter table Technicians

add SessionStatus varchar(10) check (SessionStatus IN ('ACTIVE','INACTIVE')), Password varchar(100)

alter table Admin

add AdminID int identity(1,1)

create table TablesAudit(

AuditID int not null identity(1, 1),

TableName varchar(20),

TableID int,

ModifiedBy varchar(128),

ModifiedDate datetime,

Operation varchar(10)

)

drop table SystemLog

create table SessionLog(

SessionID int identity(1,1),

Usertype varchar(20),

UserID int,

Username varchar(50),

DateTime datetime,

operation varchar(10)

)

alter table SessionLog

drop column operation

alter table SessionLog

drop column DateTime

alter table SessionLog

add LoginTime datetime

alter table SessionLog

add LogoutTime datetime

**Commission column in Sale to calculate Employees total commission easily**

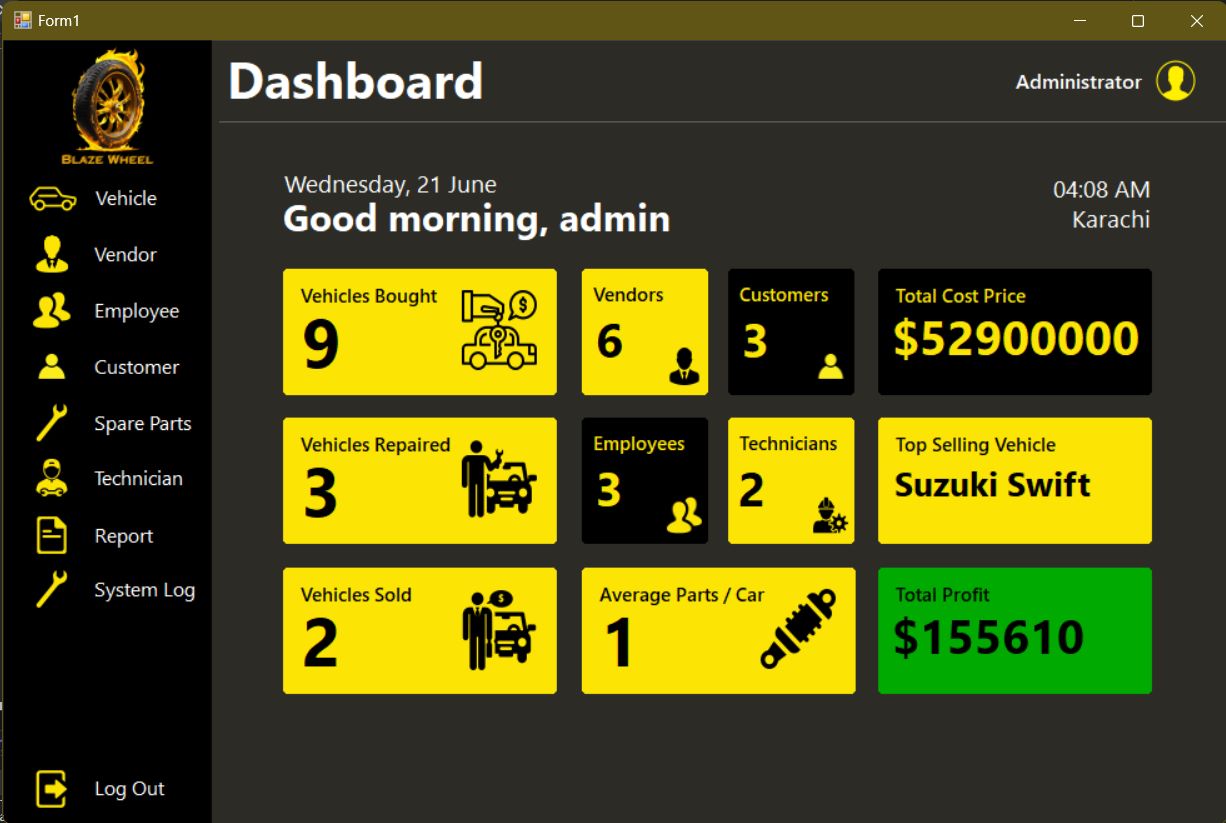
**alter table Sale add**

**Commission int**

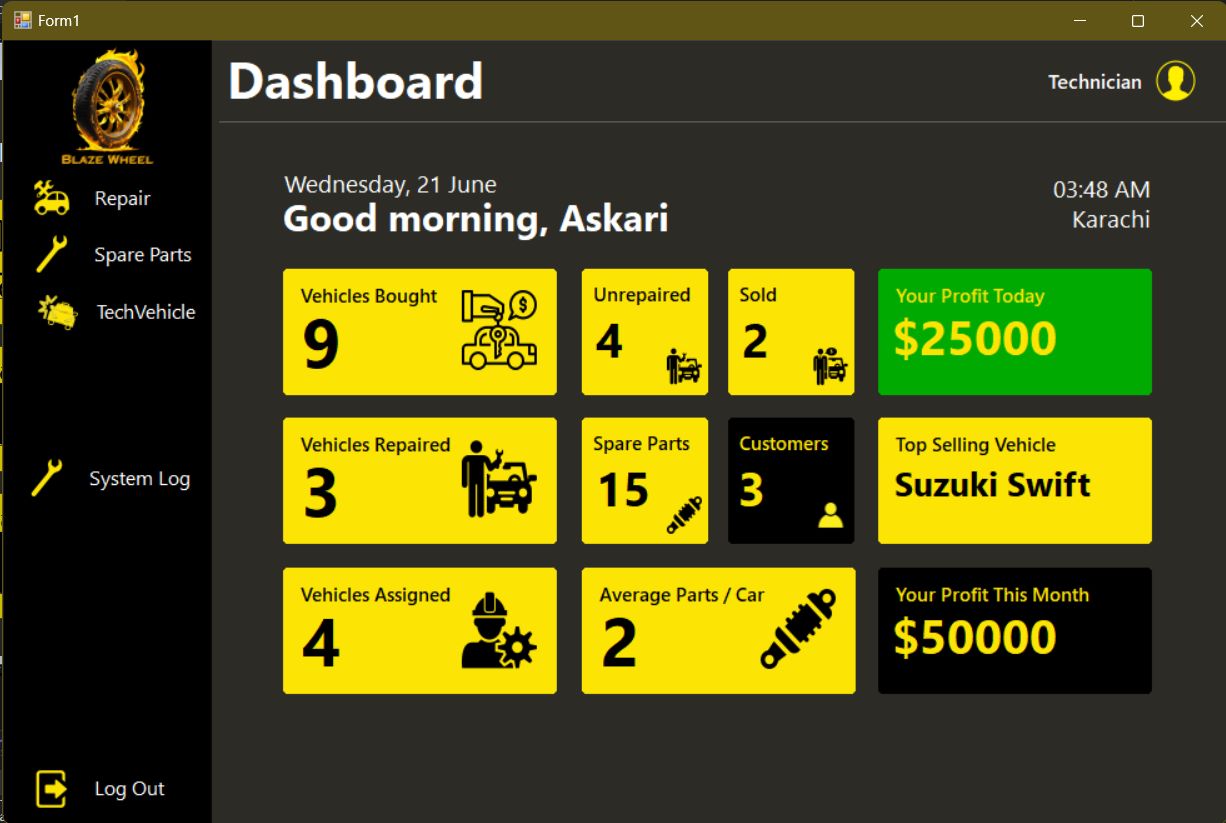
# **Interfaces:**

# Employee, Technician, and Admin Dashboard:

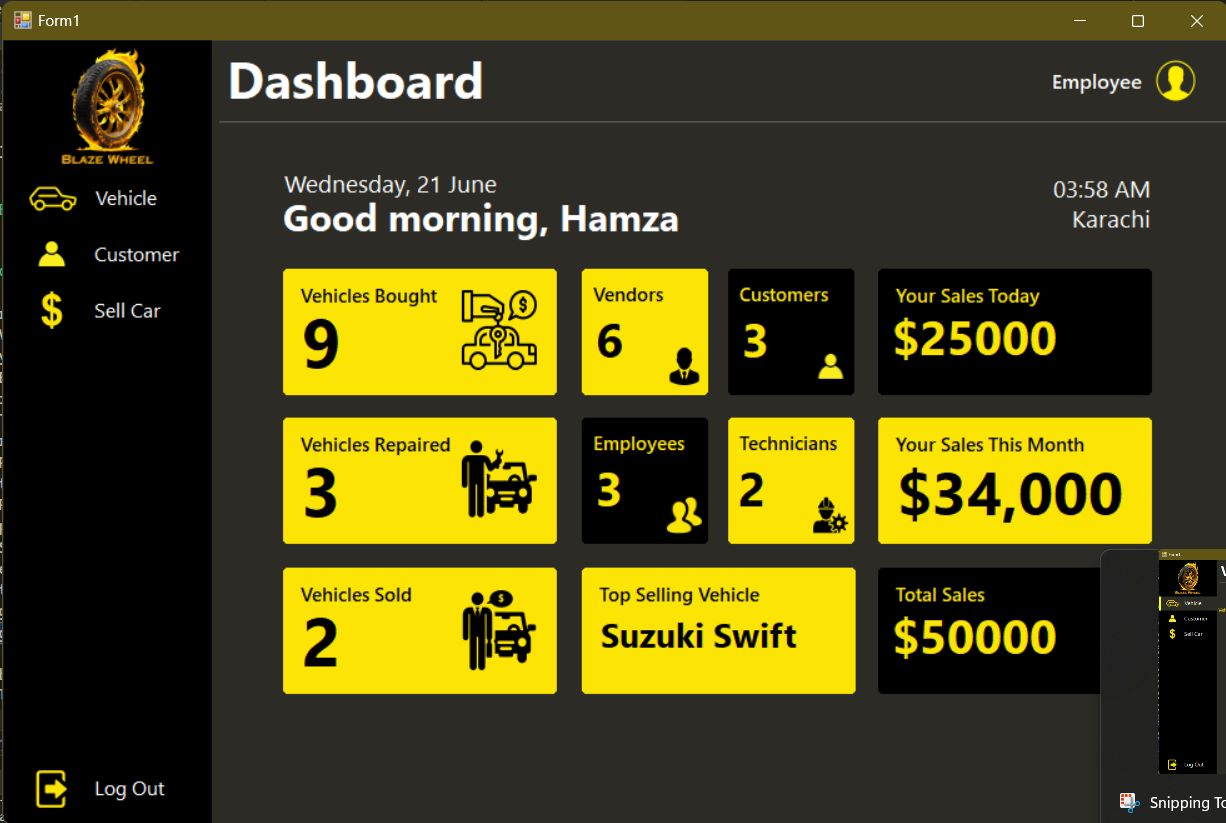
Admin Dashboard:



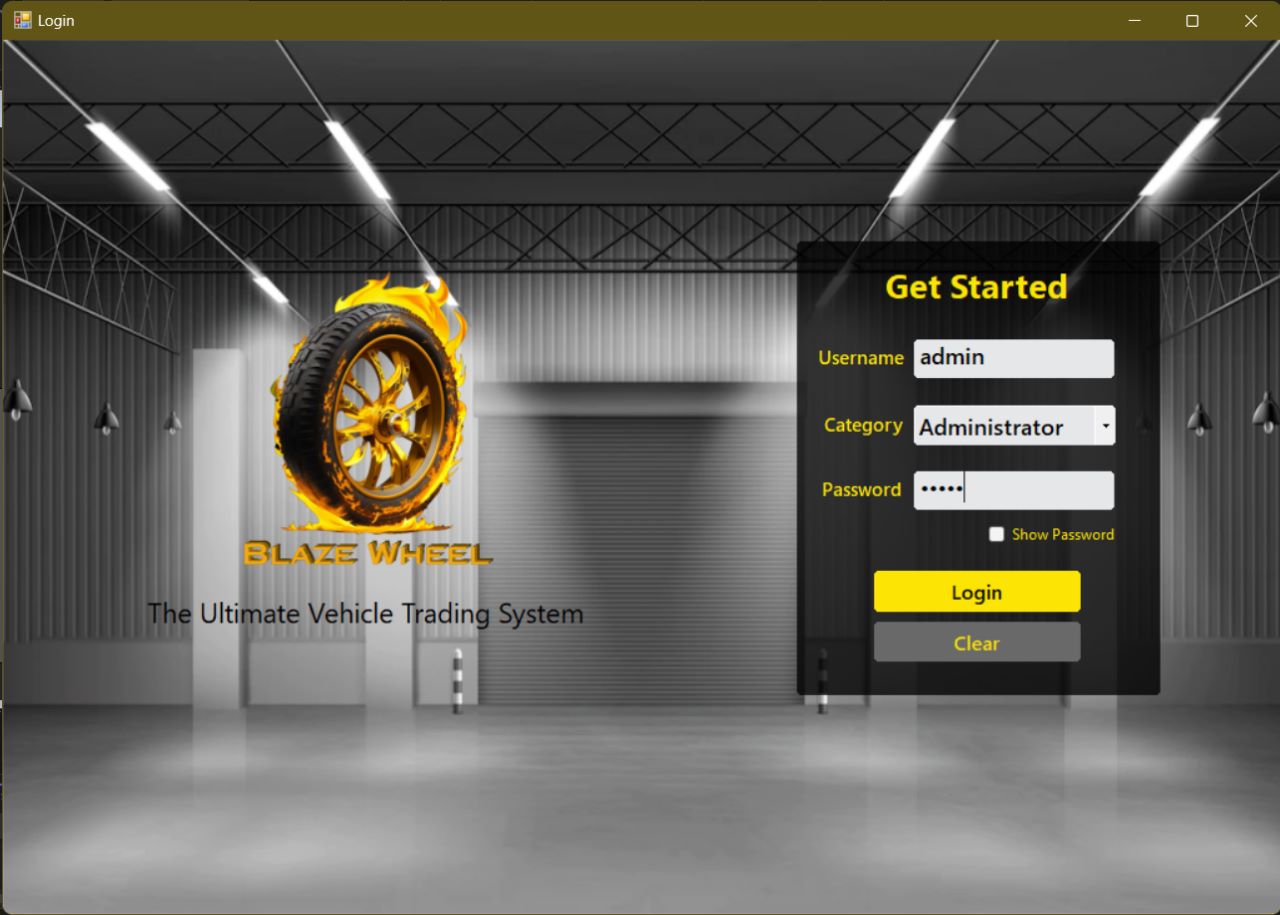
Technician Dashboard:



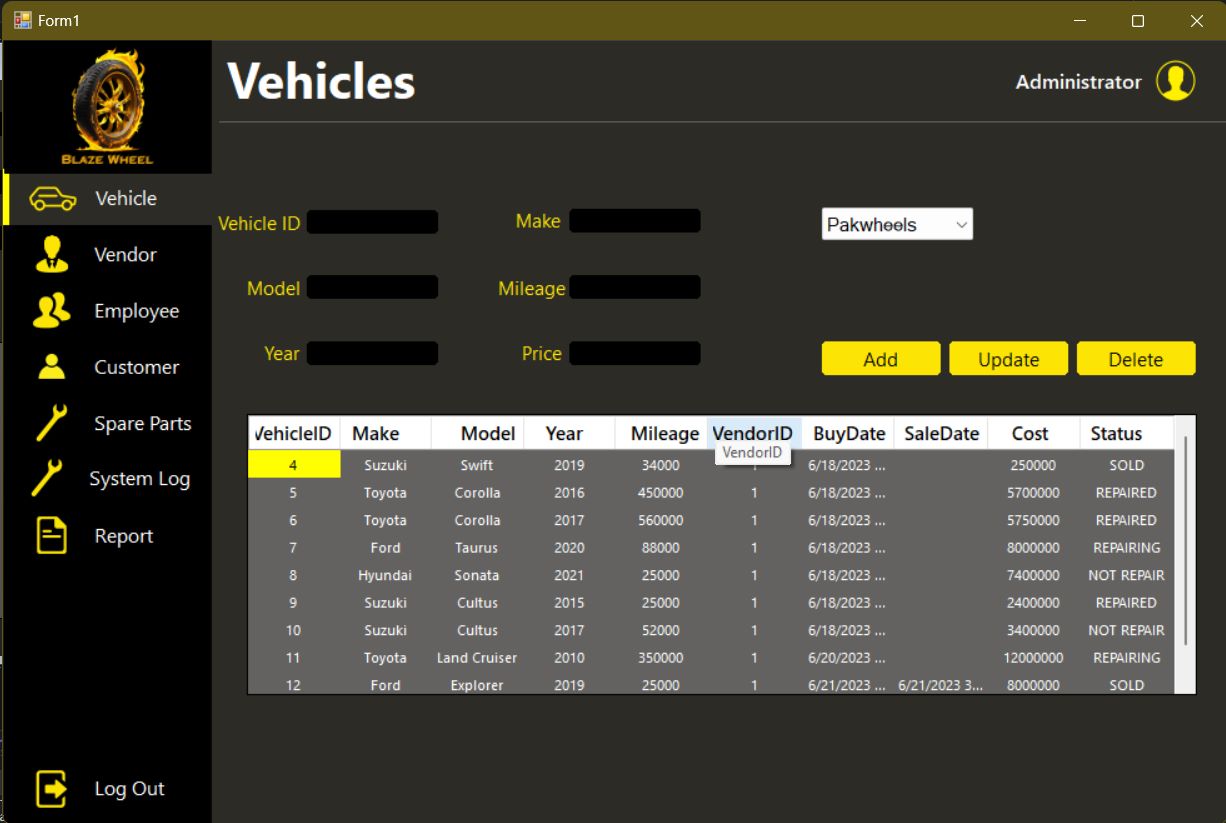
Employee Dashboard:



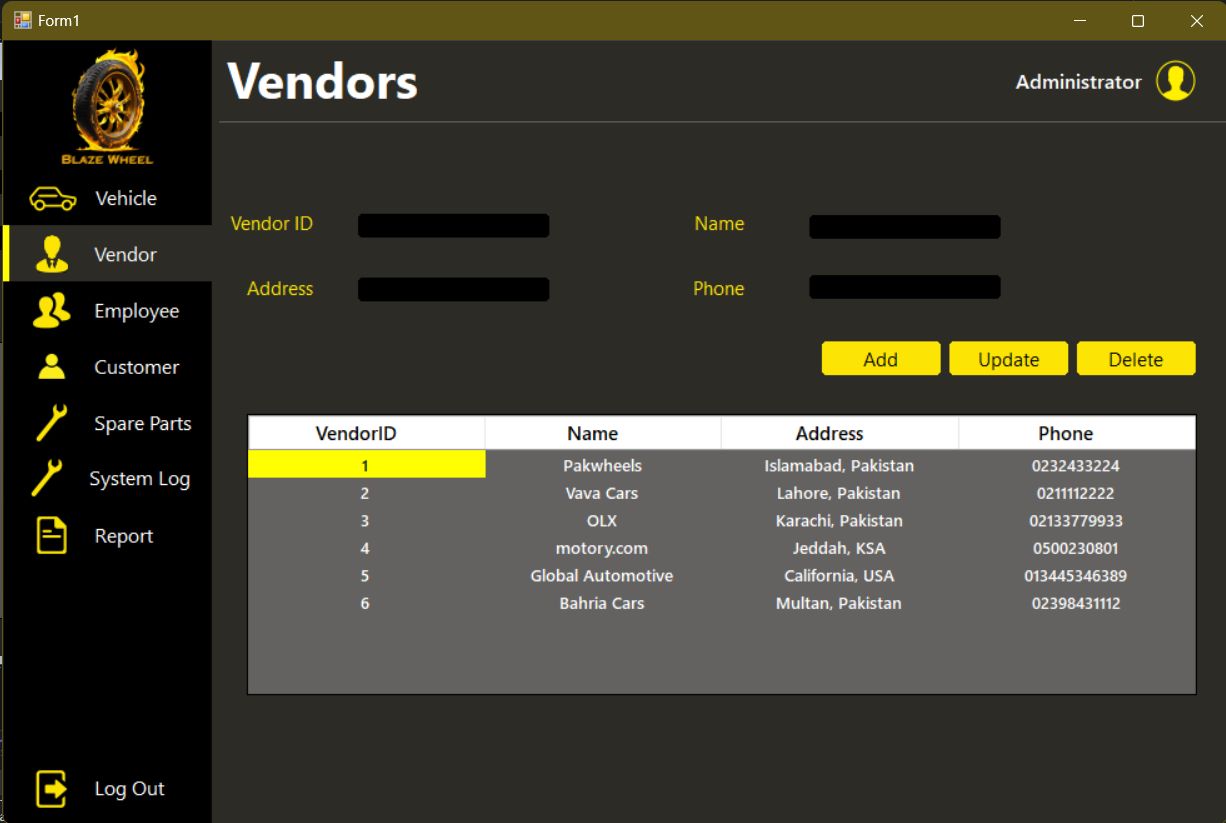
# Login Window:



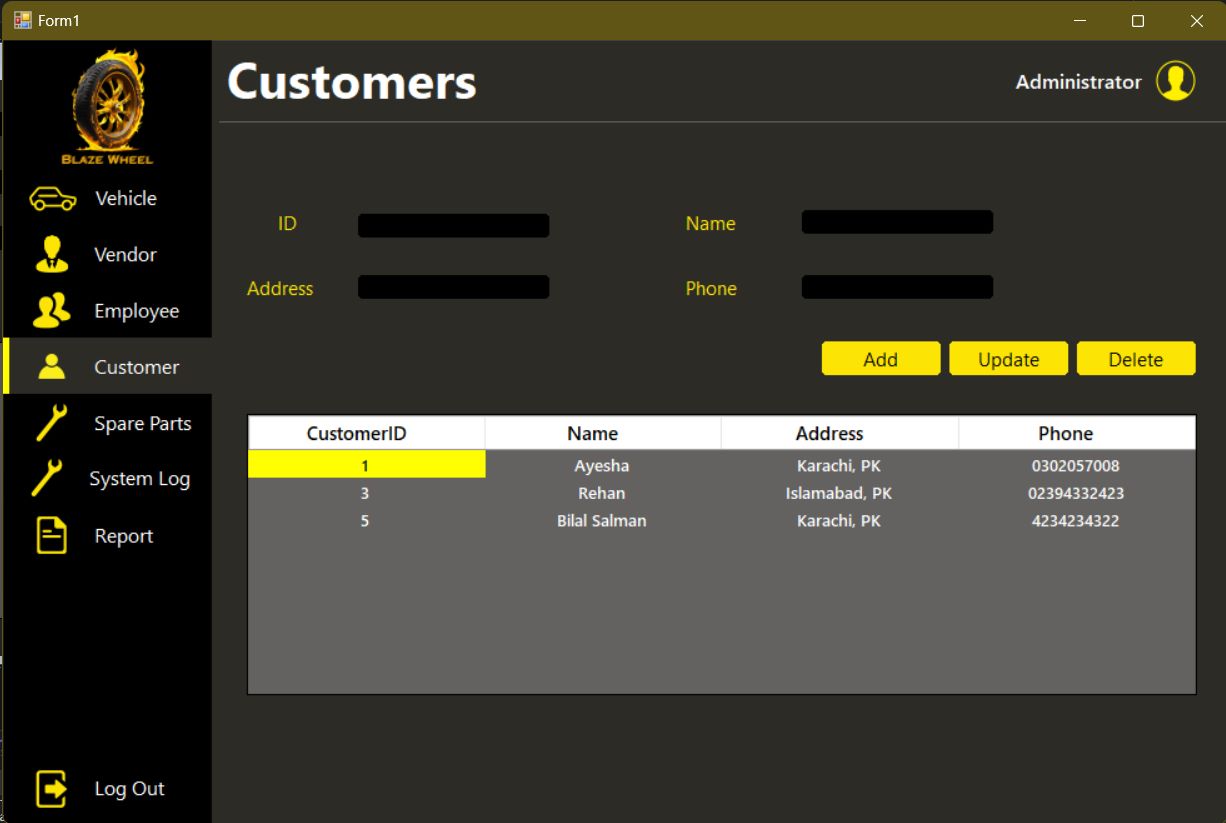
# Vehicle Window:



# Vendor Window:



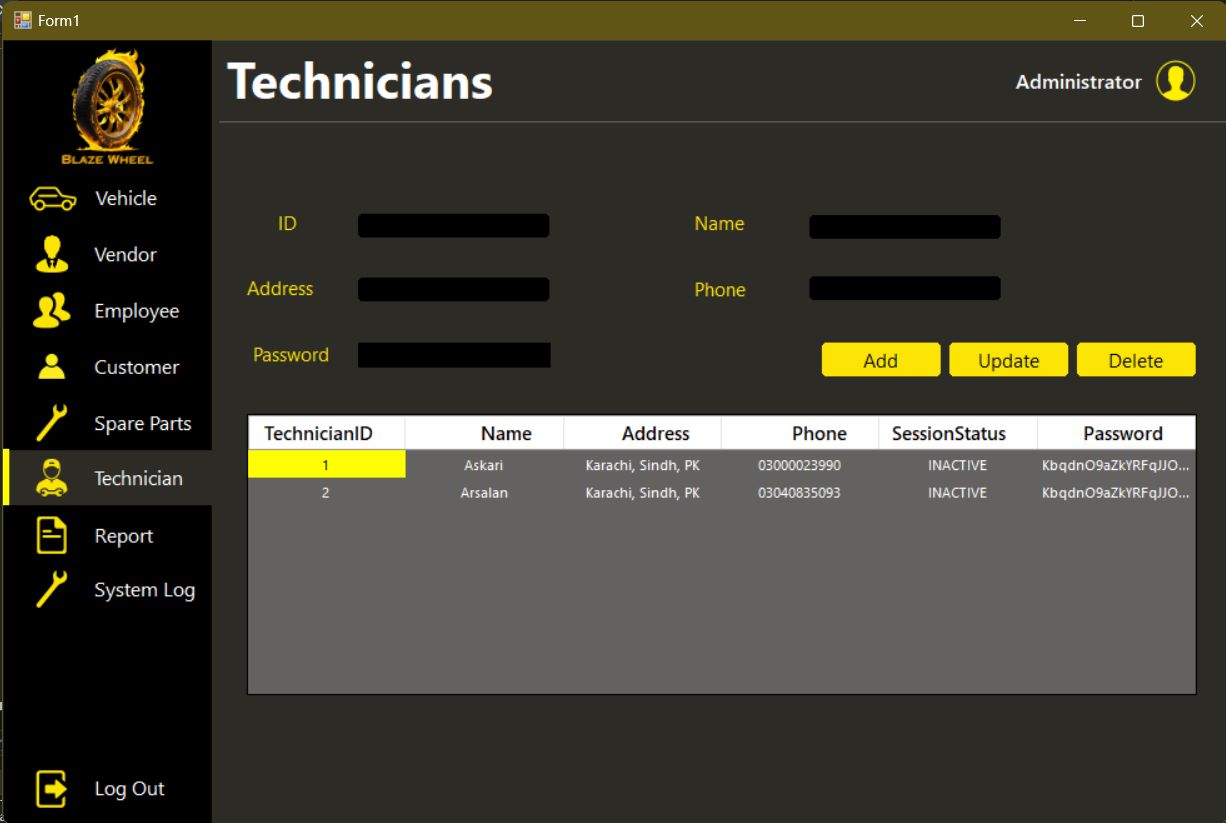
# Customer Window:



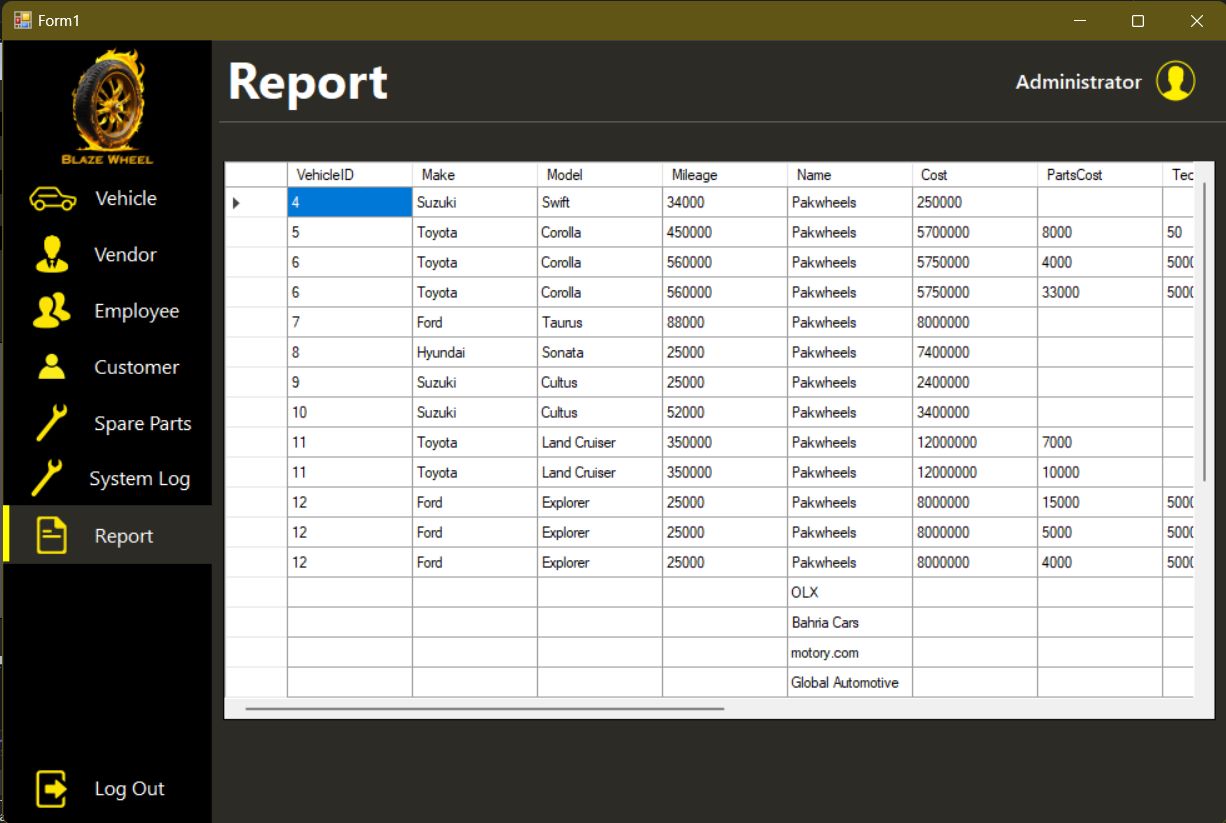
# Employee Window:



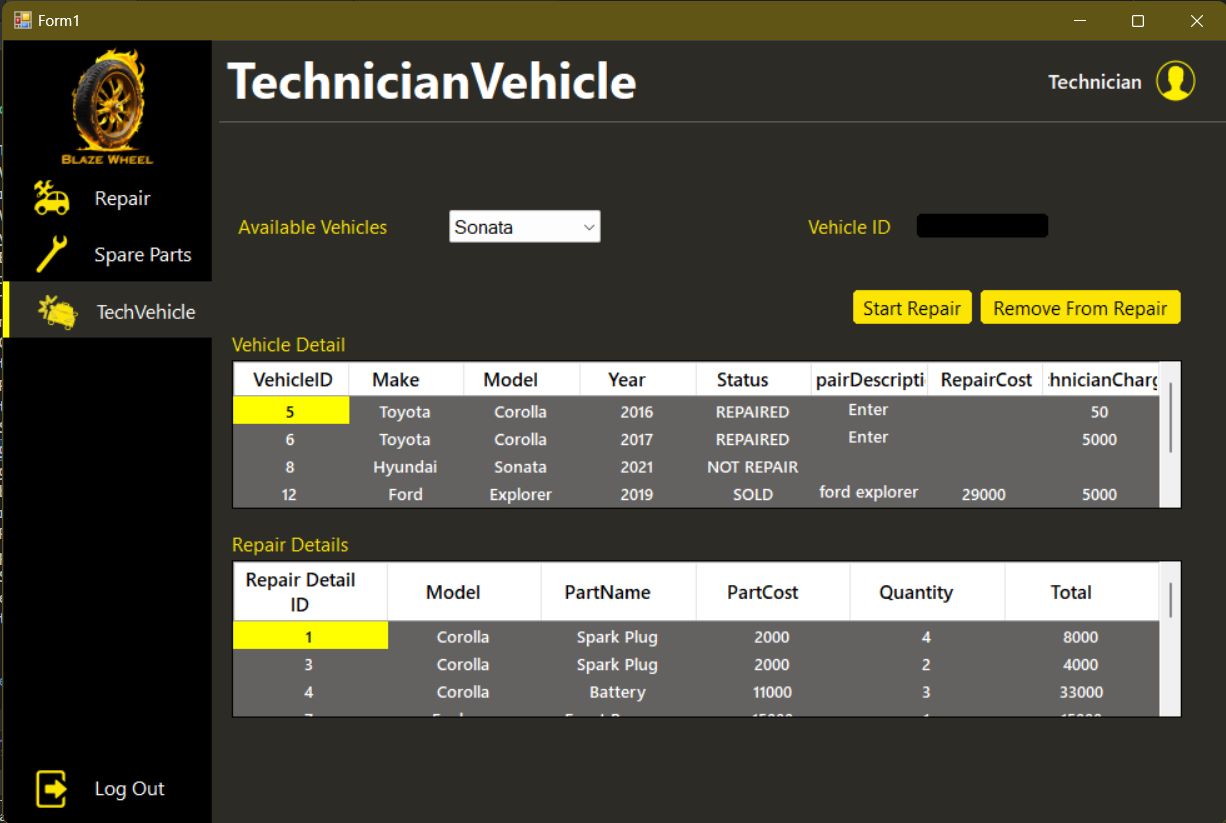
# Technician Window:



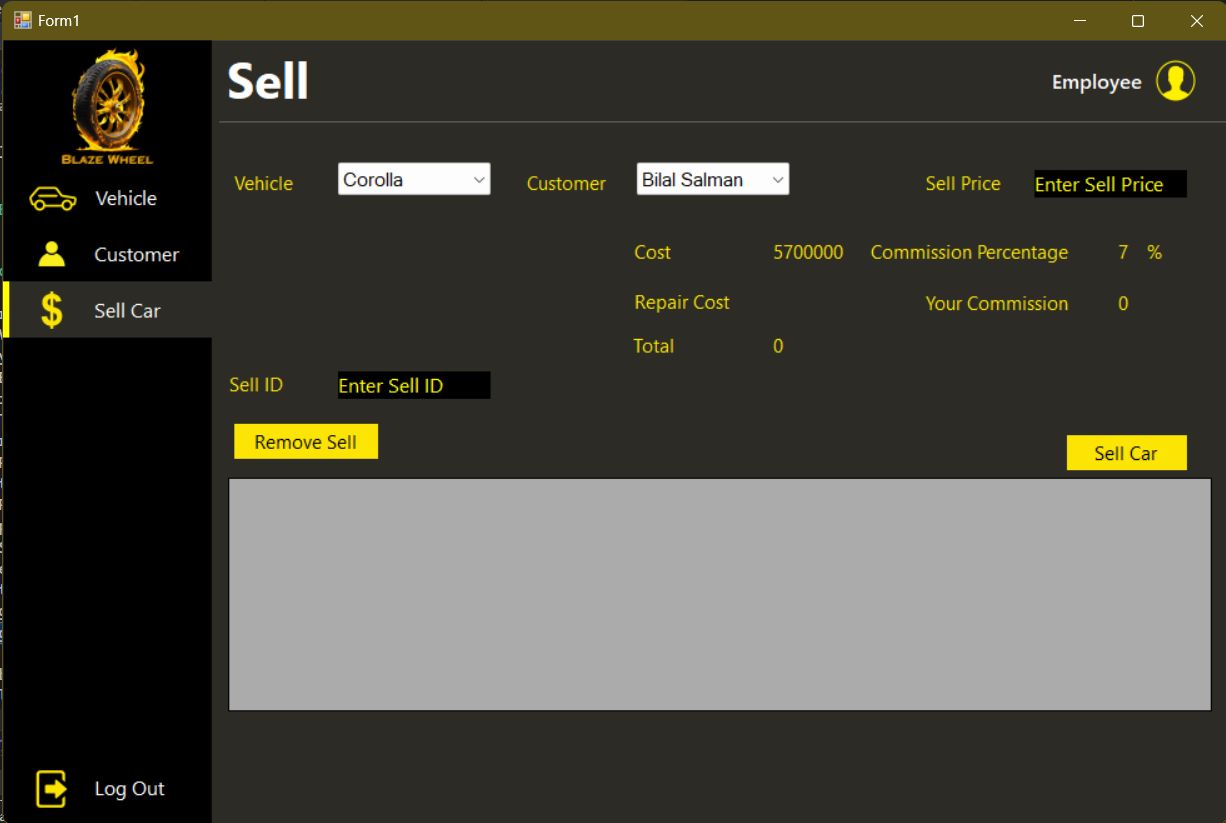
# Report Window:



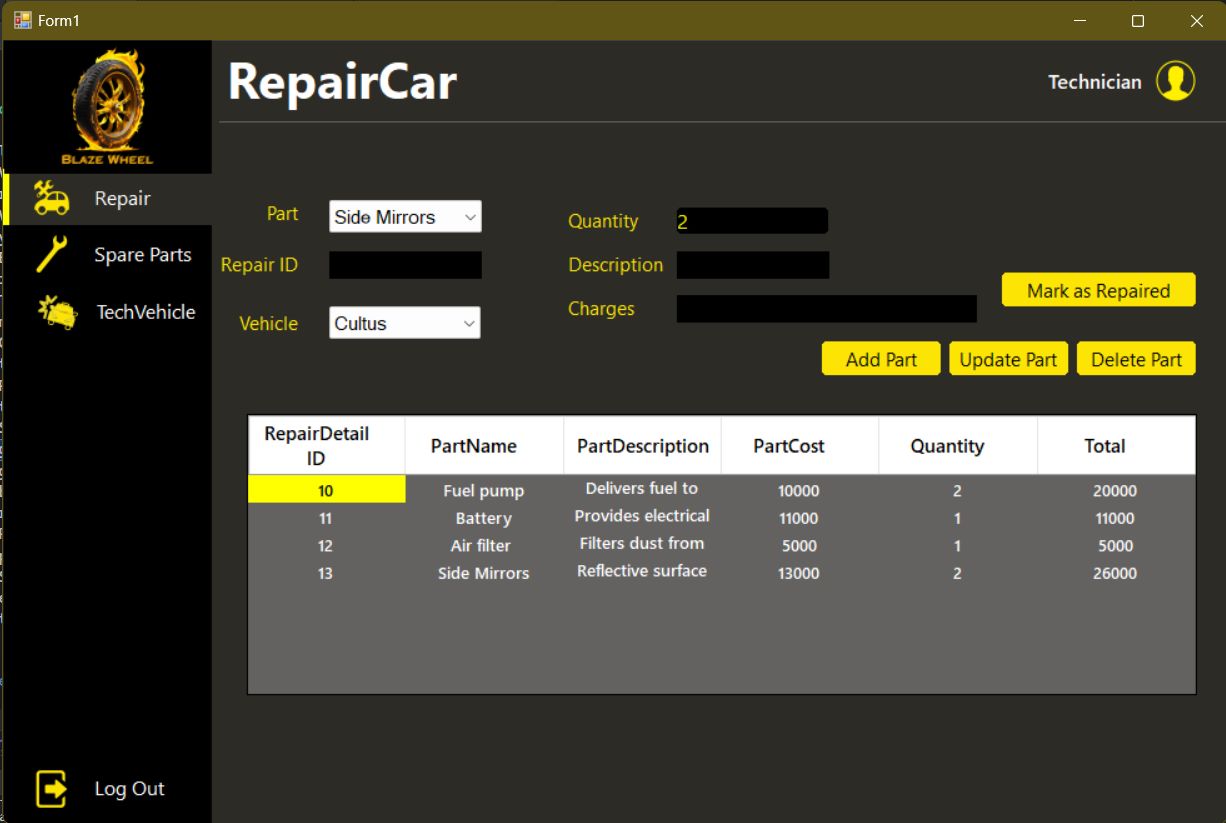
# Tech Vehicle Window:



# Sell Window:



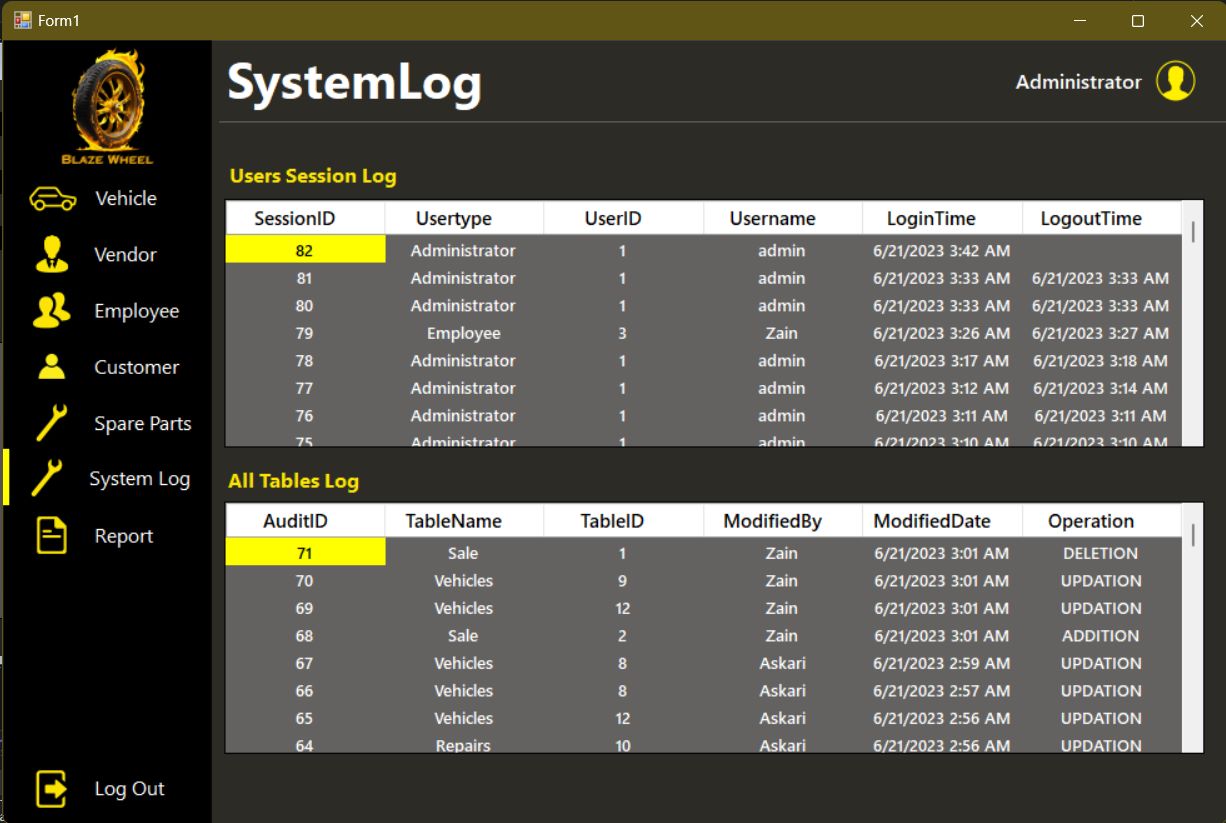
# Repair Window:



# Spare Part Window:

# 

# System Log Window:



# **Conclusion:**

As a result, the database project for Blaze Wheel, our innovative vehicle management system, has successfully met its goals of delivering a strong, effective, and secure data architecture. To ensure seamless data administration and support data-driven decision-making, we have concentrated on creating an organised database schema, utilising cutting-edge technology, and putting best practises into practise throughout this project.

Critical car-related data, such as vehicle details, maintenance history, maintenance/repair prices, and more, can now be efficiently stored and retrieved thanks to the database project. We have achieved ideal performance, assuring quick data access and low latency, by carefully developing the database schema and utilizing effective indexing algorithms.

In conclusion, the Blaze Wheel database project has been effective in giving our vehicle management system a solid, safe, and scalable foundation. We have built an architecture that enables you to streamline operations, acquire actionable insights, and make data-driven decisions to optimise the performance of your fleet and develop your business. We did this by using cutting-edge technology and putting best practices into practise.

As your fleet changes and grows, we are convinced that the Blaze Wheel database project will continue to meet your objectives for vehicle management by providing dependability, efficiency, and security.