Mechanic Shop Management System

Course: CIS 3400 EMWA [19238]
Group #1
Richard Deleon - Richard.deleon@baruchmail.cuny.edu
Khriz Moreira - Khriz.moreira@baruchmail.cuny.edu
Azhan Talukder - Azhan.talukder@baruchmail.cuny.edu
Jabin Ahmed- Jabin.ahmed@baruchmail.cuny.edu
Muhammad Khan - Muhammad.khan2@baruchmail.cuny.edu

Business Description:
Company Name: AutoTrack
Location: New York, NY
Years in Operation: 6 Years

As business begins to grow our traditional methods of record-keeping no longer work efficiently. With a growing customer base, our mechanic shop needs a digital solution that eliminates manual errors, saves time, and allows for seamless data retrieval. We plan to create a modern, and scalable system that ensures smooth operations and enhances business growth. At AutoTrack, we are committed to revolutionizing the way our mechanic shop manages operations for a more efficient and profitable future.

Our mechanic shop management system is an approach for keeping the mechanic shop running smoothly. It keeps a complete record of every vehicle we've worked on, the repairs we've done, and even customer service history, so we can deliver personalized service and quickly pull up past information when needed. Booking appointments is easier as customers can schedule repairs without the back-and-forth, cutting down on double bookings and keeping our shop organized. Plus, the inventory tool lets us keep an eye on parts and supplies in real-time. When it comes to money, the system handles billing and payments seamlessly, making invoicing faster and keeping our finances in check. A payroll system will ensure both mechanics and employees are getting paid.

Initial list of entities:

1. Customers - Done
2. Vehicles - Khriz
3. Appointments - Richard
4. Repairs/services -
5. Parts & Inventory - Muhammad
6. Invoices & Payments - Azhan
7. Suppliers -
8. Employees - Jabin

Primary Roles:

1) Proposal: Jabin Ahmed
2) Systems Analysis: Richard Deleon
3) Logical Modeling: Muhammad Khan
4) Database Schema Physical Modeling: Azhan Talukder
5) Application Implementation: Khriz Moreira

One Customer may be (0) receiving many invoices (*)
One Invoice must belong (1) to one customer (1)

One Customer may be (0) bringing in many vehicles (*)
One Vehicle must belong (1) to one customer (1)

One Vehicle may be (0) requiring many appointments (*)
One Appointment must be (1) reserved for one vehicle (1)

One Invoice may be (0) including many repairs/services (*)
One Repair may be (0) appearing in many invoices (*)

One Appointment may be (0) leading to many repairs/services.
One Repair must be (1) starting from one appointment (1)

One Repair must be (1) handled by one employee (1)
One Employee may be (0) working on many repairs (*)

One Repair may be (0) requiring many parts (*)
One Part may be (0) used for many repairs.

One Part may be (0) received from many suppliers (*)
One Supplier may be (0) providing many parts (*)

Customer (CustomerID(Key), FirstName, LastName, Email, PhoneNumber, Street, Apartment, City, State, ZipCode)

Invoices (InvoiceID (Key), Service, TotalAmount, PaymentStatus, PaymentDate, CustomerID(FK))

Vehicle (VehicleID(Key), Vin, Make, Model, ModelYear, LicencePlate, CustomerID(FK))

Appointment (AppointmentID(Key), Date_Time, Status, VehicleID(FK))

Service (ServiceID(Key), Service Type, IssueDescription, ServiceCost, AppointmentID(FK), EmployeeID(FK))

Service_Invoices (ServiceID(FK)(Key), InvoiceID(FK)(Key))

Employee (EmployeeID(Key), FirstName, LastName, Title, PhoneNumber, Email, Street, Apartment, City, State, ZipCode, HourlyWage)

Parts (PartID(Key), Name, Description, Price, Quantity)

Service_Inventory (ServiceID(FK)(Key), PartID(FK)(Key)

Supplier (SupplierID(Key), SupplierName, ContactName, PhoneNumber, Email, Street, Apartment, City, State, ZipCode)

Inventory_Supplier (PartID(FK)(Key), SupplierID(FK)(Key))

Customer
FD1: CustomerID → FirstName, LastName, Email, PhoneNumber, Street, Apartment, City, State, ZipCode
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: No, because there are transitive key dependencies.
Solution: Split Customer relations into two new relations named CustomerID and ZipCode

Split
FD1: CustomerID → FirstName, LastName, Email, PhoneNumber, Street, Apartment, ZipCode (FK))
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

FD2: ZipCode → City, State
1NF: Yes, because it was split from a relation.
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Invoices
FD1: InvoiceID → Service, TotalAmount, PaymentStatus, PaymentDate, CustomerID(FK)
1NF: Yes, because it was split from a relation.
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are transitive key dependencies.

Vehicle
FD1: VehicleID → Vin, Make, Model, ModelYear, LicencePlate, CustomerID (FK)
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are transitive key dependencies.

Appointment
FD1: AppointmentID → Date_Time, Status, VehicleID(FK))
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Service
FD1: ServiceID → Service Type, IssueDescription, ServiceCost, AppointmentID (FK), EmployeeID (FK)
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Service_Invoices
FD1: ServiceID, InvoiceID → ServiceID, InvoiceID
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Employee
FD1: EmployeeID → FirstName, LastName, Title, PhoneNumber, Email, Street, Apartment, City, State, ZipCode, HourlyWage
1NF: Yes, because it meets the criteria of a relation.
2NF: Yes, because there are no partial key dependencies.

3NF: No, because there are transitive key dependencies.
Solution: Split Employee relations into two new relations named EmployeeID and ZipCode

Split
FD1: EmployeeID → FirstName, LastName, Title, PhoneNumber, Email, Street, Apartment, ZipCode (FK), HourlyWage
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

We don't need to create a Zipcode since we have an existing one.

Parts
FD1: PartID → Name, Description, Price, Quantity
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Service_Inventory
FD1: ServiceID, PartID → ServiceID, PartID
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Supplier
FD1: SupplierID → SupplierName, ContactName, PhoneNumber, Email, Street, Apartment, City, State, ZipCode
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: No, because there are transitive key dependencies.
Solution: Split Supplier relations into two new relations named SupplierID and ZipCode

Split

FD1: SupplierID → SupplierName, ContactName, PhoneNumber, Email, Street, Apartment, ZipCode (FK)
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

We don't need to create a Zipcode since we have an existing one.

Inventory_Supplier
FD1: PartID, SupplierID → PartID, SupplierID
1NF: Yes, because it meets the definition of a relation
2NF: Yes, because there are no partial key dependencies.
3NF: Yes, because there are no transitive key dependencies.

Final Set of Relations

Customer (CustomerID(Key), FirstName, LastName, Email, PhoneNumber, Street, Apartment, ZipCode (FK))

Zipcode (Zipcode (Key), City, State)

Invoices (InvoiceID (Key), Service, TotalAmount, PaymentStatus, PaymentDate, CustomerID(FK))

Vehicle (VehicleID (Key), Vin, Make, Model, ModelYear, LicencePlate, CustomerID (FK))

Appointment (AppointmentID (Key), Date_Time, Status, VehicleID (FK))

Service (ServiceID(Key), Service Type, IssueDescription, ServiceCost, AppointmentID(FK), EmployeeID(FK))

Service_Invoices (ServiceID (FK)(KEY), InvoiceID (FK)(KEY))

Employee (EmployeeID (Key), FirstName, LastName, Title, PhoneNumber, Email, Street, Apartment, ZipCode (FK), HourlyWage)

Parts (PartID (Key), Name, Description, Price, Quantity)

Service_Inventory (ServiceID(FK)(Key), PartID (FK)(Key)

Supplier (SupplierID (Key), SupplierName, ContactName, PhoneNumber, Email, Street, Apartment, ZipCode(FK))

Inventory_Supplier (PartID(FK)(Key), SupplierID(FK)(Key))

Creating the Database Schema with Structured Query Language and Specified Foreign Keys

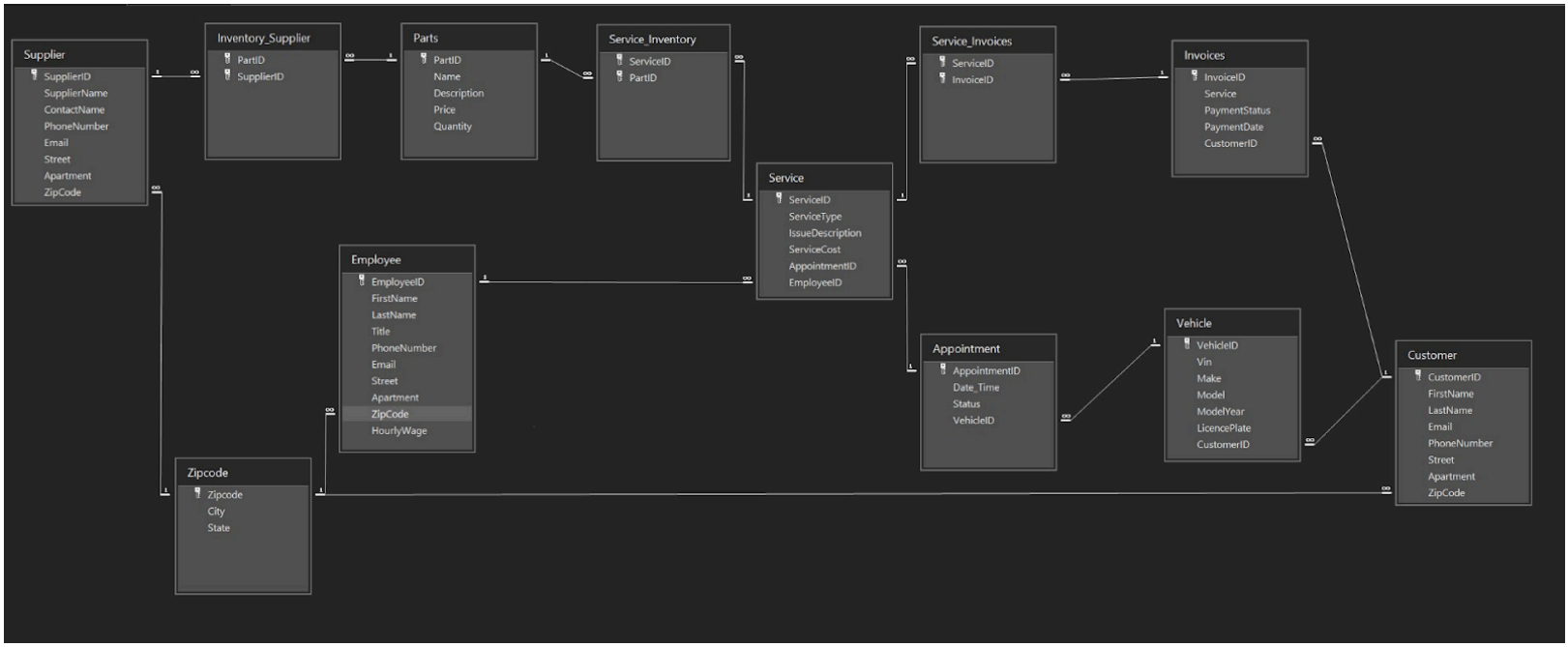
```
CREATE TABLE Zipcode (
Zipcode VARCHAR(10) NOT NULL PRIMARY KEY,
City VARCHAR(50),
State VARCHAR(50)
);
```

```
CREATE TABLE Customer (
CustomerID VARCHAR(10) NOT NULL PRIMARY KEY,
FirstName VARCHAR(50),
LastName VARCHAR(50),
Email VARCHAR(100),
PhoneNumber VARCHAR(20),
Street VARCHAR(100),
Apartment VARCHAR(20),
ZipCode VARCHAR(10),
FOREIGN KEY (ZipCode) REFERENCES Zipcode(Zipcode)
);
CREATE TABLE Employee (
EmployeeID VARCHAR(10) NOT NULL PRIMARY KEY,
FirstName VARCHAR(50),
LastName VARCHAR(50),
Title VARCHAR(50),
PhoneNumber VARCHAR(20),
Email VARCHAR(100),
Street VARCHAR(100),
Apartment VARCHAR(20),
ZipCode VARCHAR(10),
HourlyWage FLOAT,
FOREIGN KEY (ZipCode) REFERENCES Zipcode(Zipcode)
);
CREATE TABLE Invoices (
InvoiceID VARCHAR(10) NOT NULL PRIMARY KEY,
Service VARCHAR(100),
TotalAmount FLOAT,
PaymentStatus VARCHAR(30),
PaymentDate DATE,
CustomerID VARCHAR(10) NOT NULL,
FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
CREATE TABLE Vehicle (
VehicleID VARCHAR(10) NOT NULL PRIMARY KEY,
Vin VARCHAR(17) NOT NULL UNIQUE,
Make VARCHAR(50) NOT NULL,
Model VARCHAR(50) NOT NULL,
ModelYear INTEGER NOT NULL,
LicencePlate VARCHAR(20) NOT NULL UNIQUE,
```

```
CustomerID VARCHAR(10) NOT NULL,
FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
CREATE TABLE Appointment (
AppointmentID VARCHAR(10) NOT NULL PRIMARY KEY,
Date_Time DATETIME,
Status VARCHAR(30),
VehicleID VARCHAR(10) NOT NULL,
FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)
);
CREATE TABLE Service (
ServiceID VARCHAR(10) NOT NULL PRIMARY KEY,
ServiceType VARCHAR(50),
IssueDescription VARCHAR(255),
ServiceCost FLOAT,
AppointmentID VARCHAR(10) NOT NULL,
EmployeeID VARCHAR(10) NOT NULL,
FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID),
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
CREATE TABLE Service_Invoices (
ServiceID VARCHAR(10) NOT NULL,
InvoiceID VARCHAR(10) NOT NULL,
PRIMARY KEY (ServiceID, InvoiceID),
FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID),
FOREIGN KEY (InvoiceID) REFERENCES Invoices(InvoiceID)
);
CREATE TABLE Parts (
PartID VARCHAR(10) NOT NULL PRIMARY KEY,
Name VARCHAR(50),
Description VARCHAR(255),
Price FLOAT,
Quantity INTEGER
);
CREATE TABLE Service_Inventory (
ServiceID VARCHAR(10) NOT NULL,
PartID VARCHAR(10) NOT NULL,
PRIMARY KEY (ServiceID, PartID),
FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID),
FOREIGN KEY (PartID) REFERENCES Parts(PartID)
```

```
);
CREATE TABLE Supplier (
SupplierID VARCHAR(10) NOT NULL PRIMARY KEY,
SupplierName VARCHAR(100),
ContactName VARCHAR(50),
PhoneNumber VARCHAR(20),
Email VARCHAR(100),
Street VARCHAR(100),
Apartment VARCHAR(20),
ZipCode VARCHAR(10),
FOREIGN KEY (ZipCode) REFERENCES Zipcode(Zipcode)
);
CREATE TABLE Inventory_Supplier (
PartID VARCHAR(10) NOT NULL,
SupplierID VARCHAR(10) NOT NULL,
PRIMARY KEY (PartID, SupplierID),
FOREIGN KEY (PartID) REFERENCES Parts(PartID),
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
);
```

Relationship View



Adding Data to the Tables using SQL INSERT Statements

INSERT INTO Zipcode (Zipcode, City, State) VALUES ('10461', 'Bronx', 'NY');
INSERT INTO Zipcode (Zipcode, City, State) VALUES ('11368', 'Queens', 'NY');

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Street, Apartment, ZipCode)
VALUES ('C001', 'Kim', 'Chaewon', 'kim.chaewon@email.com', '718-555-1234', '123 Main St', 'Apt 1A', '10461');
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Street, Apartment, ZipCode)
VALUES ('C002', 'Mark', 'Tuan', 'mark.tuan@email.com', '212-555-6789', '456 Park Ave', NULL, '11368');

INSERT INTO Employee (EmployeeID, FirstName, LastName, Title, PhoneNumber, Email, Street, Apartment, ZipCode, HourlyWage)
VALUES ('E001', 'Jang', 'Wonyoung', 'Mechanic', '718-555-5678', 'jang.wonyoung@email.com', '456 Service Rd', NULL, '10461', 25.5);
INSERT INTO Employee (EmployeeID, FirstName, LastName, Title, PhoneNumber, Email, Street, Apartment, ZipCode, HourlyWage)
VALUES ('E002', 'Kai', 'Cenat', 'Technician', '917-555-7890', 'kai.cenat@email.com', '789 Auto Ln', NULL, '11368', 28.0);

INSERT INTO Invoices (InvoiceID, Service, TotalAmount, PaymentStatus, PaymentDate, CustomerID)
VALUES ('I001', 'Brake Replacement', 300.00, 'Paid', #2024-04-10#, 'C001');
INSERT INTO Invoices (InvoiceID, Service, TotalAmount, PaymentStatus, PaymentDate, CustomerID)
VALUES ('I002', 'Oil Change', 80.00, 'Pending', #2024-04-09#, 'C002');

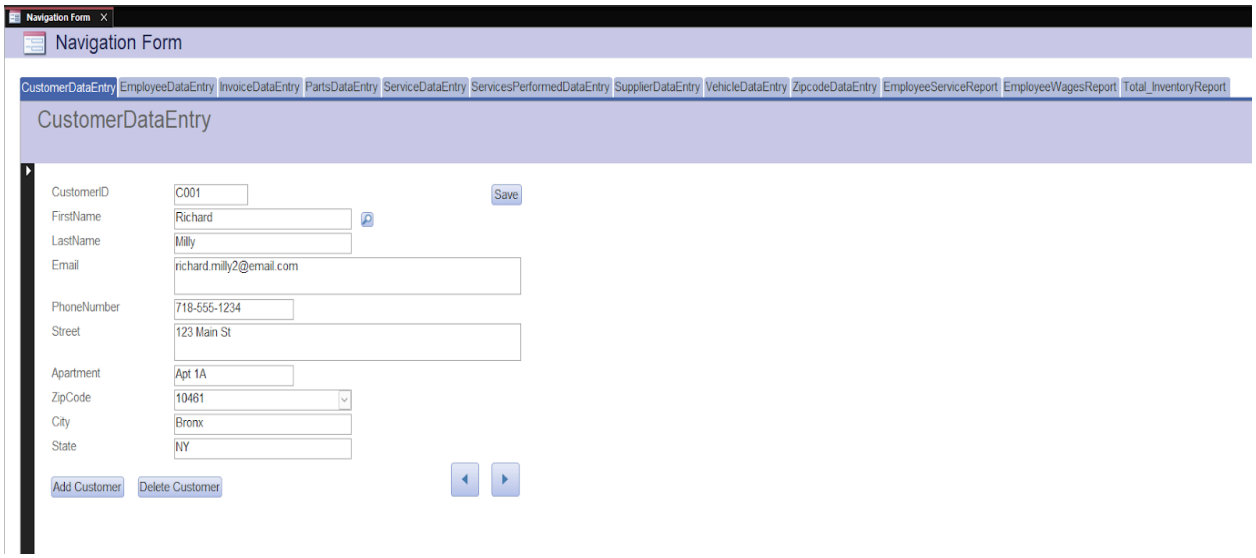
INSERT INTO Vehicle (VehicleID, Vin, Make, Model, ModelYear, LicencePlate, CustomerID)
VALUES ('V001', '1HGCM82633A004352', 'Honda', 'Civic', 2020, 'NY1234', 'C001');
INSERT INTO Vehicle (VehicleID, Vin, Make, Model, ModelYear, LicencePlate, CustomerID)
VALUES ('V002', '1HGFA16596L081111', 'Toyota', 'Camry', 2022, 'NY5678', 'C002');

INSERT INTO Appointment (AppointmentID, Date_Time, Status, VehicleID)

```
VALUES ('A001', #2024-04-15#, 'Completed', 'V001');
INSERT INTO Appointment (AppointmentID, Date_Time, Status, VehicleID)
VALUES ('A002', #2024-04-18#, 'Scheduled', 'V002');

INSERT INTO Service (ServiceID, ServiceType, IssueDescription, ServiceCost,
AppointmentID, EmployeeID)
VALUES ('S001', 'Brake', 'Brake pads worn out', 250.00, 'A001', 'E001');
INSERT INTO Service (ServiceID, ServiceType, IssueDescription, ServiceCost,
AppointmentID, EmployeeID)
VALUES ('S002', 'Oil Change', 'Routine oil service', 60.00, 'A002', 'E002');

INSERT INTO Service_Invoices (ServiceID, InvoiceID) VALUES ('S001', 'I001');
INSERT INTO Service_Invoices (ServiceID, InvoiceID) VALUES ('S002', 'I002');

INSERT INTO Parts (PartID, Name, Description, Price, Quantity)
VALUES ('P001', 'Brake Pad', 'Front brake pads', 50.00, 10);
INSERT INTO Parts (PartID, Name, Description, Price, Quantity)
VALUES ('P002', 'Oil Filter', 'Engine oil filter', 15.00, 20);

INSERT INTO Service_Inventory (ServiceID, PartID) VALUES ('S001', 'P001');
INSERT INTO Service_Inventory (ServiceID, PartID) VALUES ('S002', 'P002');

INSERT INTO Supplier (SupplierID, SupplierName, ContactName, PhoneNumber, Email,
Street, Apartment, ZipCode)
VALUES ('SUP001', 'AutoParts Inc.', 'Azhan', '718-555-8765', 'azhan@autoparts.com', '789
Supplier Blvd', NULL, '10461');
INSERT INTO Supplier (SupplierID, SupplierName, ContactName, PhoneNumber, Email,
Street, Apartment, ZipCode)
VALUES ('SUP002', 'OEM Supplies', 'Richard', '212-555-2222', 'richard@oem.com', '1010
Industry Way', NULL, '11368');

INSERT INTO Inventory_Supplier (PartID, SupplierID) VALUES ('P001', 'SUP001');
INSERT INTO Inventory_Supplier (PartID, SupplierID) VALUES ('P002', 'SUP002');
```

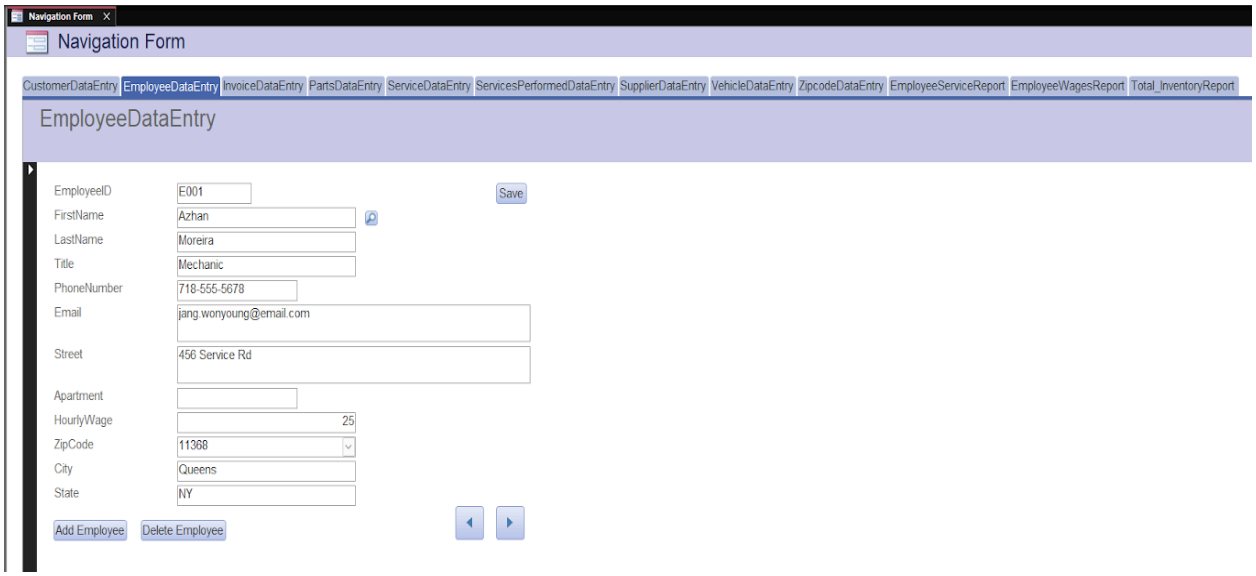
Navigation Forms

Customer Data Entry Navigation Form



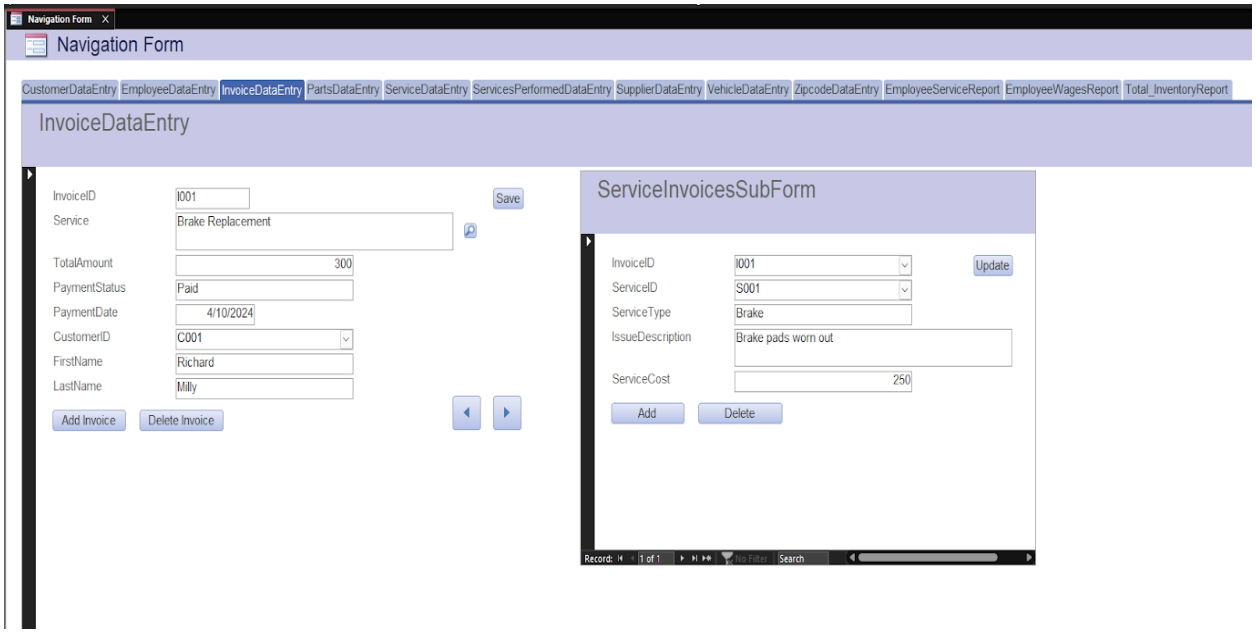
This tab directs the user to the customer form which displays information about each customer, such as their contact and address details. It also allows navigation through different customer records.

Employee Data Entry Navigation Form



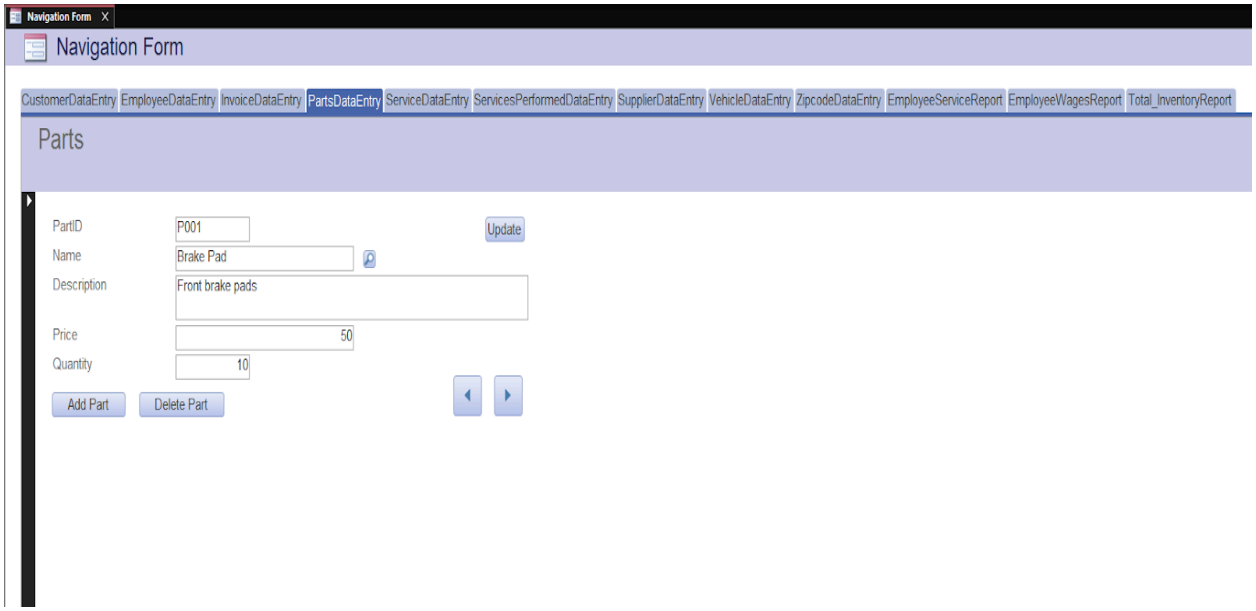
This tab opens the employee form which shows the employee's details, It also lets the user browse through all employee entries.

Invoice Data Entry Navigation Form



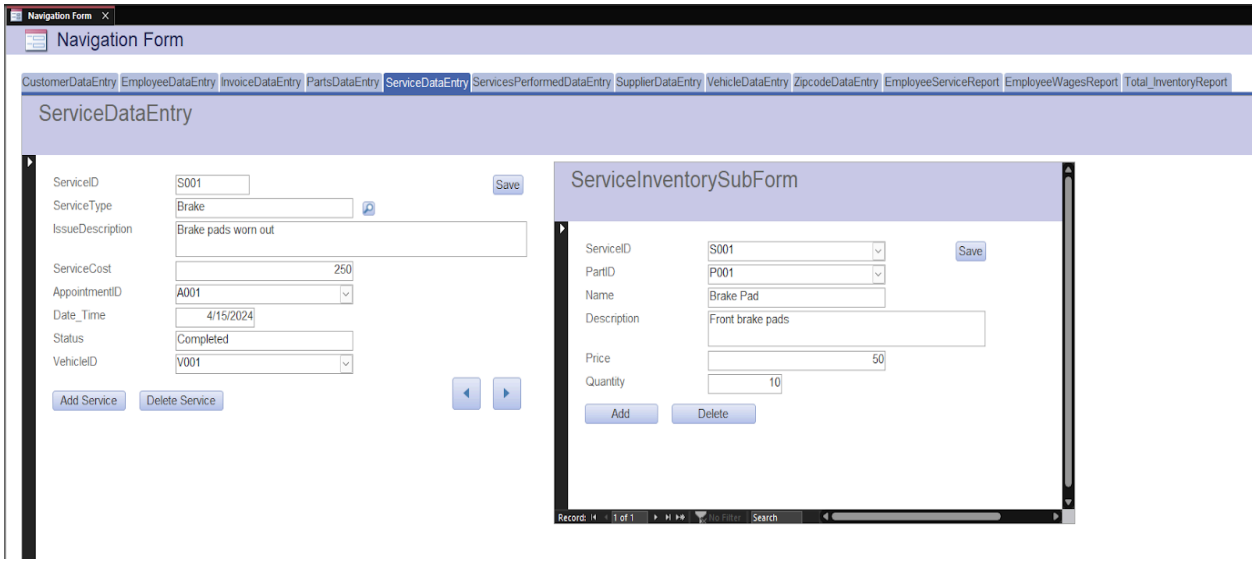
This section logs invoice details such as the invoice ID, service type, and payment information, and also includes a subform to track individual service items linked to each invoice.

Parts Data Entry Navigation Form



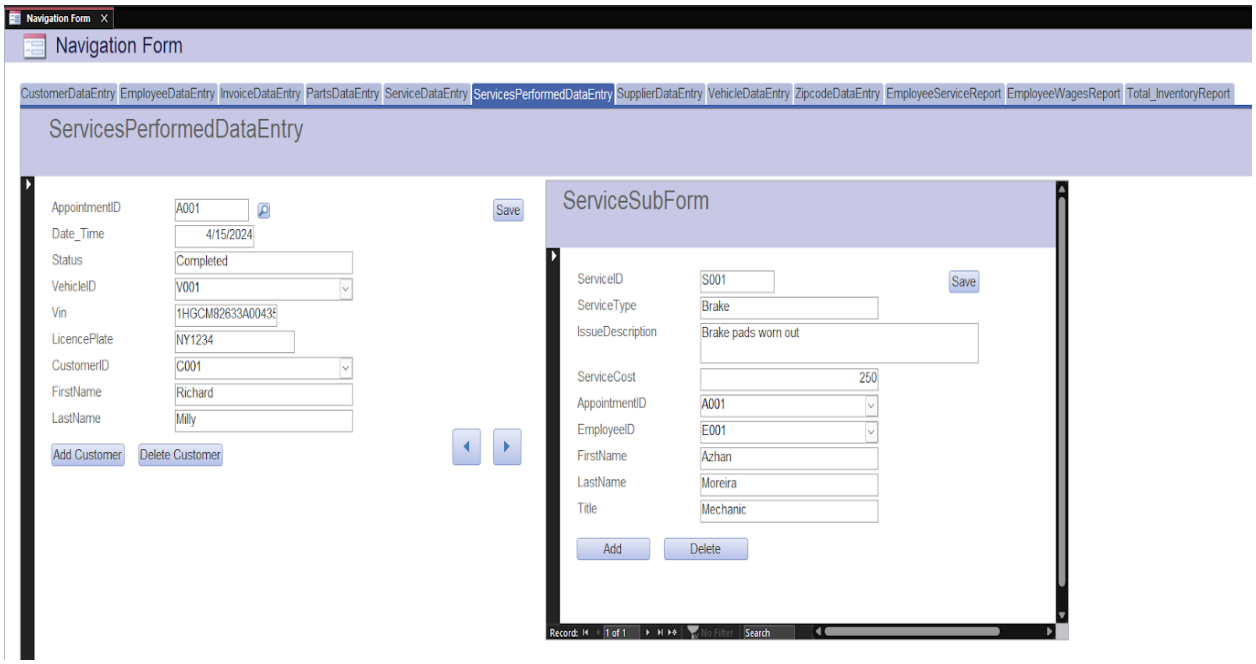
This tab directs the user to the parts form which stores information about each part, such as the item name, price, and quantity. It also allows users to browse through all part records.

Service Data Entry Navigation Form



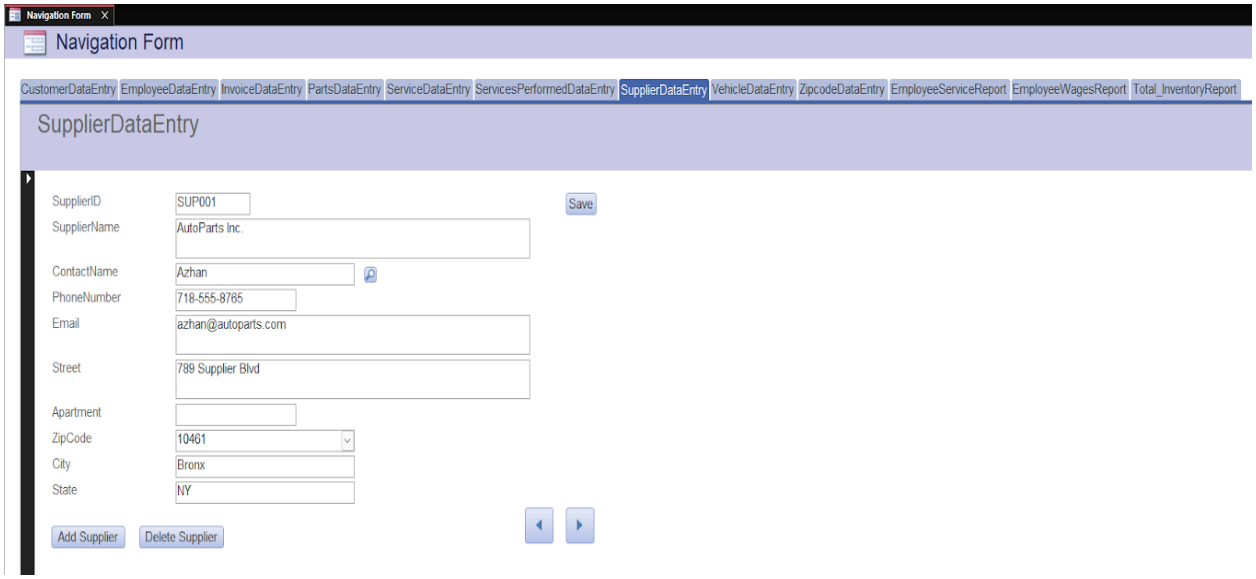
This tab brings the user to the service form where service jobs are recorded along with the labor type, assigned employee, and job details. It includes a subform that tracks the parts used for each service.

Services Performed Data Entry Navigation Form



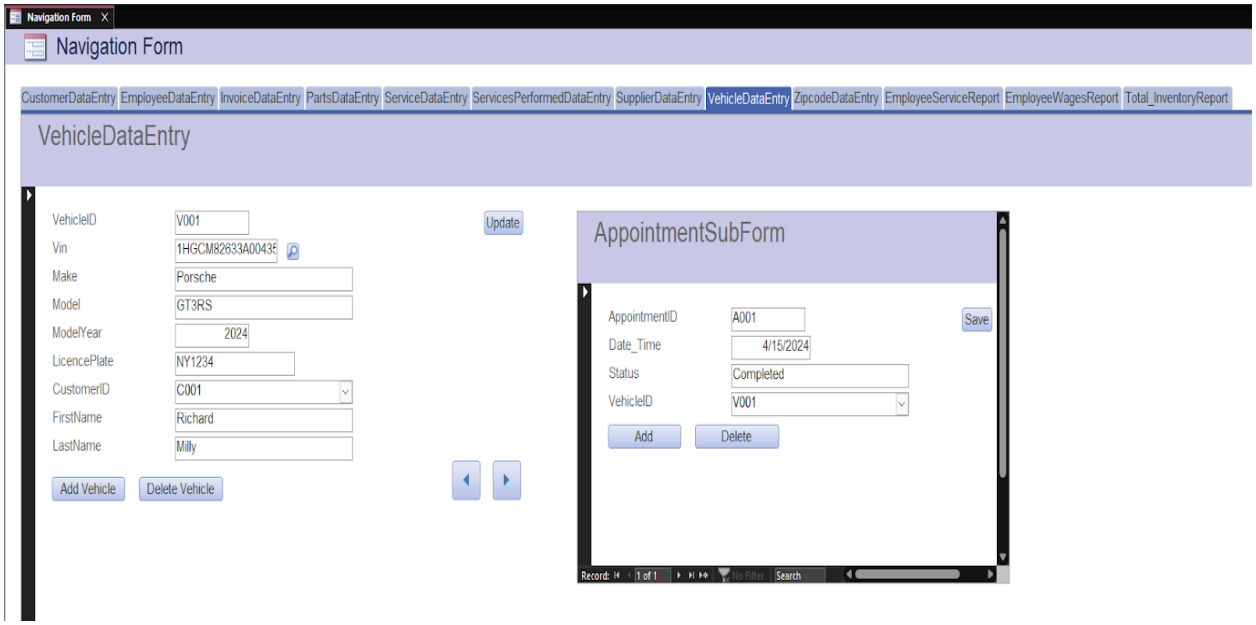
This tab directs the user to the form where completed services are recorded. It logs customer concerns, vehicle details, and service types. The subform on the right shows which services were performed, who completed them, and any parts used in the process.

Supplier Data Entry Navigation Form



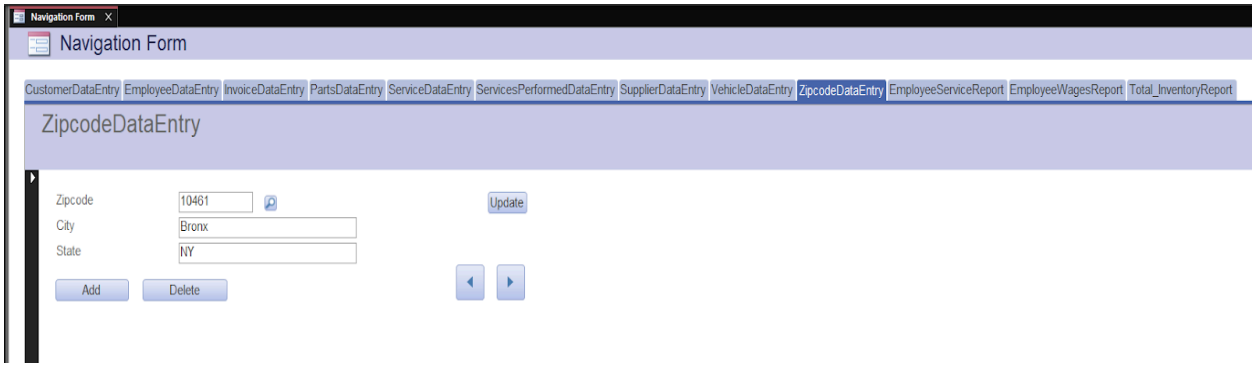
This tab directs the user to the supplier form which stores information about vendors, including their company name, contact details, and the parts they provide. It also lets users scroll through and manage different supplier records.

Vehicle Data Entry Navigation Form



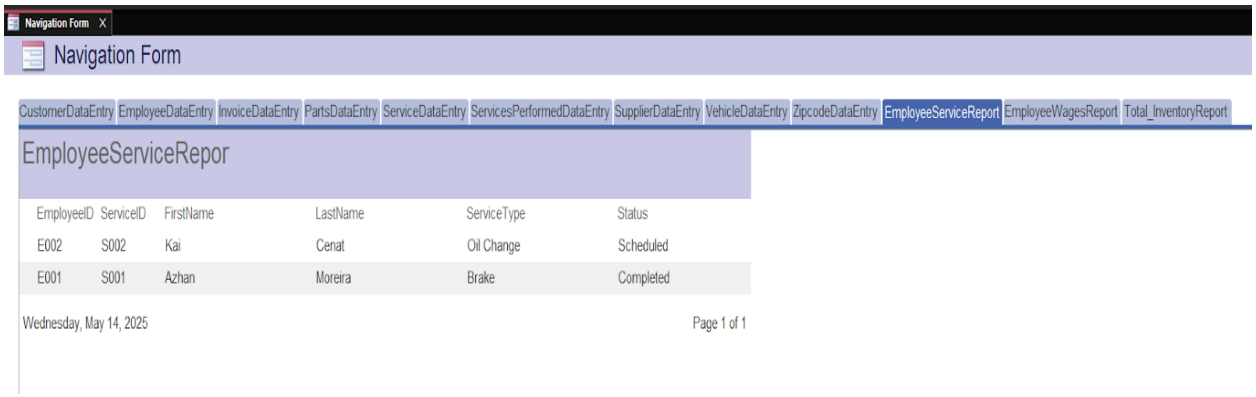
This tab opens the vehicle form which contains vehicle information such as the VIN, make, model, and license plate. The subform shows related appointments connected to each vehicle.

Zip Code Data Entry Navigation Form



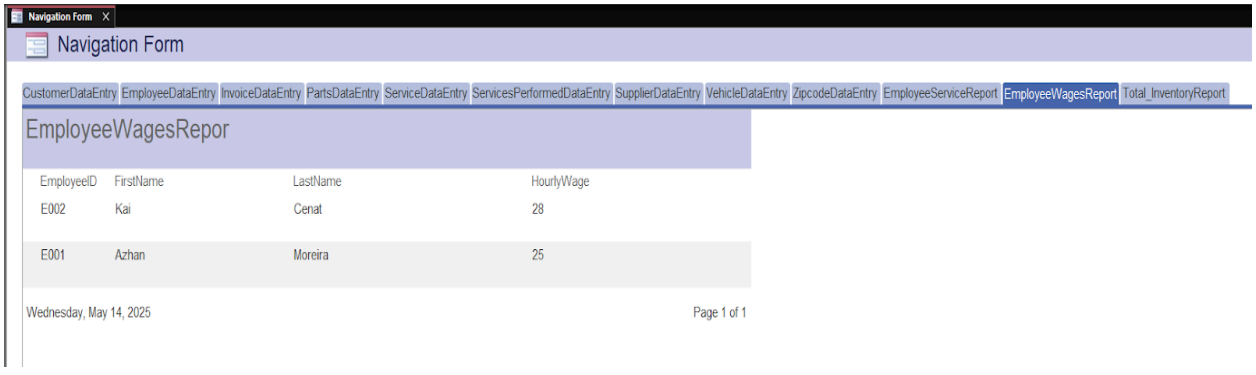
This tab directs the user to the zip code form, which stores information about specific zip codes and the cities they belong to. It allows users to add or browse through different zip code entries.

Employee Service Report Navigation Form



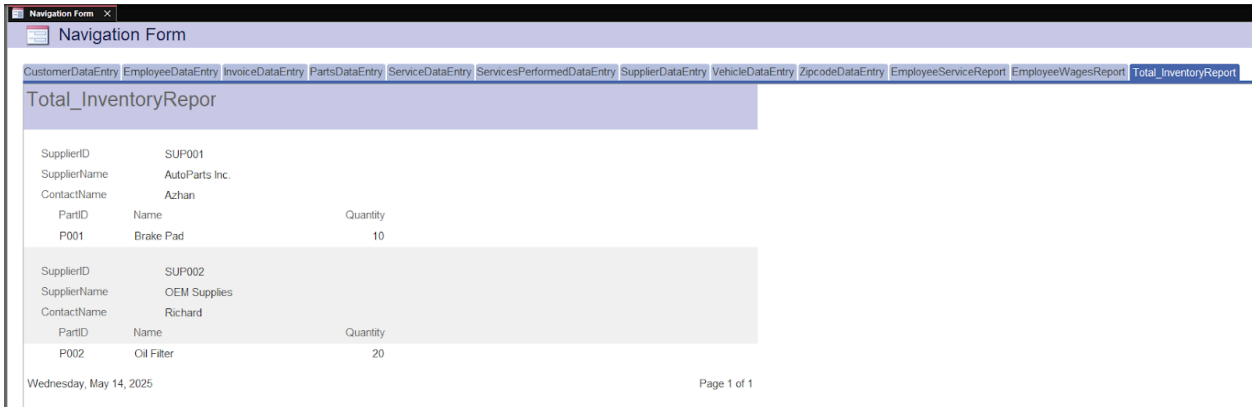
This tab displays a report of services performed by each employee, showing their name, location, service date, and customer involved.

Employee Wages Report Navigation Form



This tab shows a report of employee wages, detailing how many services each employee completed and their total earnings.

Total Inventory Report Navigation Form



This tab presents a detailed report of the current inventor. It shows the supplier ID, name, contact person, and a breakdown of parts they provide.

VBA CODE

Proper Case of First Name and Last Name

```
Private Sub FirstName_AfterUpdate()
    FirstName = StrConv(FirstName, vbProperCase)
End Sub

Private Sub LastName_AfterUpdate()
     LastName = StrConv(LastName, vbProperCase)
End Sub
```

This changes the first letter of both first and last name to be capitalized automatically even if you put richard it will be Richard.

Upper Case for Job Title

```
Private Sub Title_AfterUpdate()
    Title = StrConv(Title, vbProperCase)
End Sub
```

This changes the first letter of the job titled to be capitalized automatically even if you put mechanic it will be Mechanic.

Upper Case for City and State

```
Private Sub City_AfterUpdate()
    City = StrConv(City, vbProperCase)
End Sub
```

```
Private Sub State_AfterUpdate()
    State = UCase(State)
End Sub
```

This automatically capitalized the first letter of the city, and makes the entire state capitalized example ny will be NY, and bronx will be Bronx.

Queries

Supplier

```
SELECT Supplier.SupplierID, Parts.PartID, Supplier.SupplierName, Supplier.ContactName, Parts.Name, Parts.Quantity
FROM Supplier INNER JOIN (Parts INNER JOIN Inventory_Supplier ON Parts.[PartID] = Inventory_Supplier.[PartID]) ON Supplier.[SupplierID] = Inventory_Supplier.[SupplierID];
```

Retrieves the supplier ID, name, and contact details, along with part ID, name, and quantity. shows which supplier provides which parts and in what quantity.

Customer

```
SELECT Customer.CustomerID, Invoices.InvoiceID, Customer.FirstName, Customer.LastName, Invoices.PaymentStatus, Invoices.PaymentDate
FROM Customer INNER JOIN Invoices ON Customer.[CustomerID] = Invoices.[CustomerID];
```

Retrieves the customer's ID, first and last name, and their associated invoice ID, payment status, and date. This helps track customer information and their payment history.

Manufacturer

```
SELECT Service.ServiceID, Service.AppointmentID, Vehicle.VehicleID, Service.ServiceType, Vehicle.Make, Vehicle.Model, Service.ServiceCost
FROM (Vehicle INNER JOIN Appointment ON Vehicle.[VehicleID] = Appointment.[VehicleID]) INNER JOIN Service ON Appointment.[AppointmentID] = Service.[AppointmentID];
```

Retrieves the vehicle ID, make, and model along with the appointment ID, service type, and service cost. This connects vehicles to their service records and helps track service history per manufacturer.

Conclusion

Software

Throughout our mechanics database, we utilized several tools to help us complete the project. The most fundamental tool we used was Microsoft Access to build and manage our database. The tools integrated in the software made it easier to design the tables, initiate relationships coherently while allowing us to create queries. In addition, we used LucidChart to create our

entity relationship model which helped us connect the tables and to normalize before the implementation process. We used Discord and Zoom to communicate with one another for each milestone and we used Google Docs to save our work.

Experiences in the Group Project

The most challenging part of the project was the last milestone which was the application implementation. It took us time to figure out the VBA code and how to build it. When writing code, it's important to be careful because it could disrupt the flow of the application. We had to ensure our forms and reports worked properly and had to test it out several times. It took us some time to figure out the SQL server and we had to ensure there were proper validation rules like preventing blank fields. The easiest part of the project was the normalization part because our relations were already in third normal form for the most part. After defining the entities and relationships accurately, the normalization process is usually the simplest part in creating a database unless there's redundancy and partial dependencies.

We learned that if you plan out how you want to design your database in the beginning then it helps you see what technical features are important to focus on. If we were to do it all over again, we would probably cut down some of the entities to make everything simpler and to allow for the database to run smoothly.