URBAN SMART LOGISTICS AND DELIVERY SYSTEM

PROJECT FOR DAMG6210 - DATA MANAGEMENT AND DATABASE DESIGN
UNDER THE GUIDANCE OF PROF. MANUEL D MONTROND

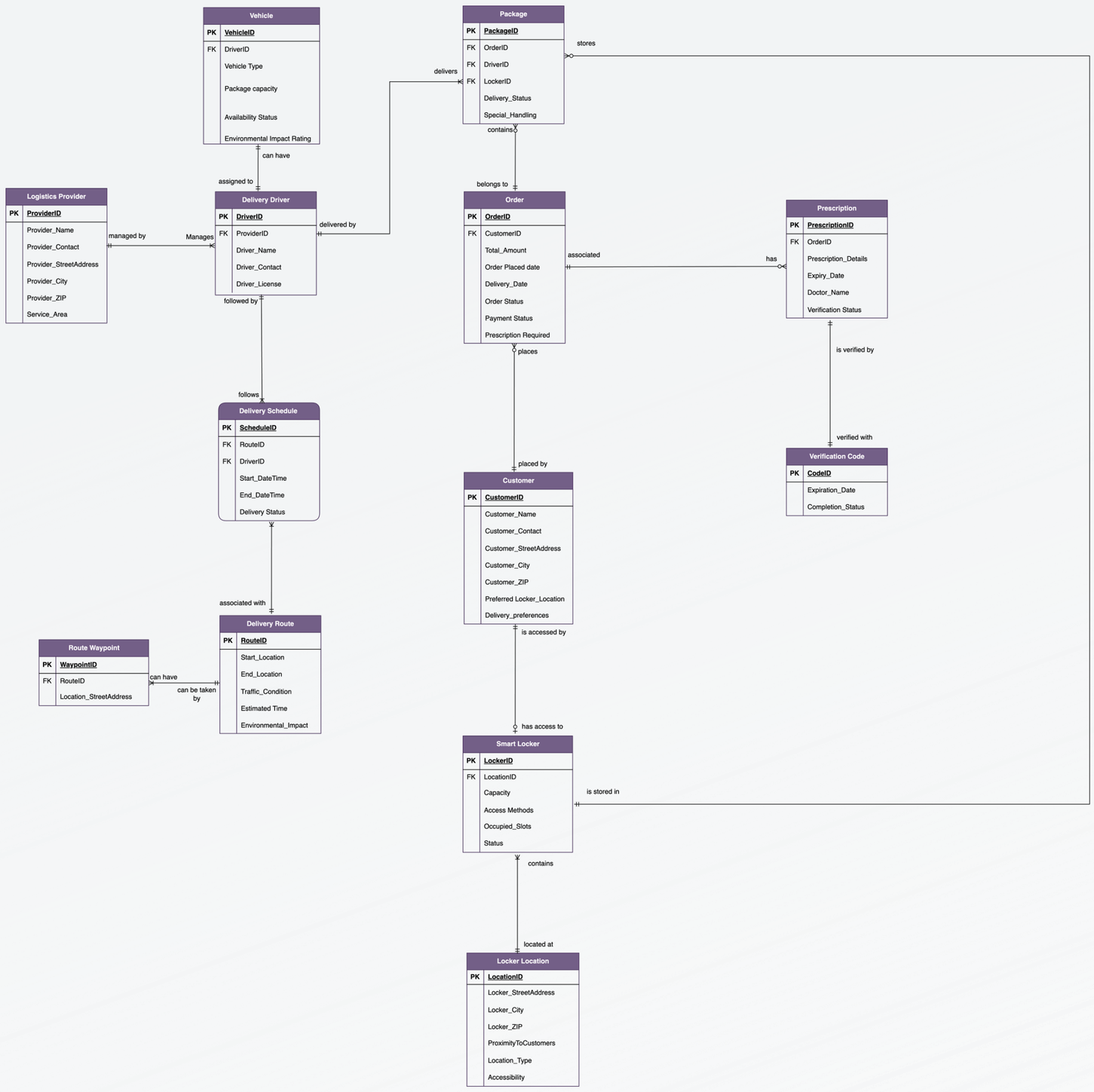
GROUP 3:
PRAGATI NAROTE
RUTUJA DHATRAK
DEEPIKA VADDADI
SAMRUDDHI SAWANT
TAPAS DESAI

INTRODUCTION

The project implements a centralized database-driven platform designed to optimize urban logistics by managing customers, delivery schedules, smart lockers, and prescriptions with secure authentication and route efficiency.

Featuring normalized database structures, advanced encryption for sensitive data, and optimized indexing, the system ensures secure, scalable, and real-time management of urban deliveries, reducing congestion and enhancing user convenience.

ENTITY RELATIONSHIP DIAGRAM



DDL, DML, STORED PROCEDURES

```sql
-- Create Database if not exists
IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'P4_SCHEMA')
BEGIN
    CREATE DATABASE P4_SCHEMA;
END
GO

USE P4_SCHEMA;
GO

-- Drop each table individually in reverse order of dependencies
IF OBJECT_ID('dbo.Waypoint', 'U') IS NOT NULL DROP TABLE Waypoint;
IF OBJECT_ID('dbo.DeliverySchedule', 'U') IS NOT NULL DROP TABLE DeliverySchedule;
IF OBJECT_ID('dbo.Package', 'U') IS NOT NULL DROP TABLE Package;
IF OBJECT_ID('dbo.Prescription', 'U') IS NOT NULL DROP TABLE Prescription;
IF OBJECT_ID('dbo.VerificationCode', 'U') IS NOT NULL DROP TABLE VerificationCode;
IF OBJECT_ID('dbo.[Order]', 'U') IS NOT NULL DROP TABLE [Order];
IF OBJECT_ID('dbo.Customer', 'U') IS NOT NULL DROP TABLE Customer;
IF OBJECT_ID('dbo.DeliveryDriver', 'U') IS NOT NULL DROP TABLE DeliveryDriver;
IF OBJECT_ID('dbo.Vehicle', 'U') IS NOT NULL DROP TABLE Vehicle;
IF OBJECT_ID('dbo.LogisticProvider', 'U') IS NOT NULL DROP TABLE LogisticProvider;
IF OBJECT_ID('dbo.SmartLocker', 'U') IS NOT NULL DROP TABLE SmartLocker;
IF OBJECT_ID('dbo.LockerLocation', 'U') IS NOT NULL DROP TABLE LockerLocation;
IF OBJECT_ID('dbo.DeliveryRoute', 'U') IS NOT NULL DROP TABLE DeliveryRoute;

-- Creating LockerLocation Table
CREATE TABLE LockerLocation (
    LocationID INT PRIMARY KEY IDENTITY(1,1),
    Address VARCHAR(255) NOT NULL,
    City VARCHAR(100) NOT NULL,
    ZipCode VARCHAR(10)
);

-- Creating SmartLocker Table
CREATE TABLE SmartLocker (
    LockerID INT PRIMARY KEY IDENTITY(1,1),
    LocationID INT NOT NULL,
    LockerSize VARCHAR(20) CHECK (LockerSize IN ('Small', 'Medium', 'Large')),
    IsOccupied BIT DEFAULT 0,
    FOREIGN KEY (LocationID) REFERENCES LockerLocation(LocationID)
);

-- Creating LogisticProvider Table
CREATE TABLE LogisticProvider (
    ProviderID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100) NOT NULL,
    ContactNumber VARCHAR(15),
    Address VARCHAR(255),
    ZipCode VARCHAR(10),
    ServiceArea VARCHAR(100)
);

-- Creating Vehicle Table
CREATE TABLE Vehicle (
    VehicleID INT PRIMARY KEY IDENTITY(1,1),
    LicensePlate VARCHAR(15) NOT NULL UNIQUE,
    Model VARCHAR(50),
    Capacity DECIMAL(5, 2) CHECK (Capacity > 0),
    AvailabilityStatus BIT DEFAULT 1
);

-- Creating DeliveryRoute Table
CREATE TABLE DeliveryRoute (
    RouteID INT PRIMARY KEY IDENTITY(1,1),
    StartLocation VARCHAR(255) NOT NULL,
    EndLocation VARCHAR(255) NOT NULL,
    EstimatedTime TIME
);

-- Creating Customer Table (references SmartLocker)
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    PhoneNumber VARCHAR(15),
    StreetAddress VARCHAR(255),
    ZipCode VARCHAR(10),
    City VARCHAR(100),
    PreferredLockerLocation INT,
    DeliveryPreference VARCHAR(50),
    CHECK (LEN(PhoneNumber) = 10),
    FOREIGN KEY (PreferredLockerLocation) REFERENCES SmartLocker(LockerID)
);

-- Creating Order Table (references Customer)
CREATE TABLE [Order] (
    OrderID INT PRIMARY KEY IDENTITY(1,1),
    CustomerID INT NOT NULL,
    OrderDate DATETIME DEFAULT GETDATE(),
    DeliveryDate DATETIME,
    OrderStatus VARCHAR(50) CHECK (OrderStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
    PaymentStatus VARCHAR(50) CHECK (PaymentStatus IN ('Paid', 'Unpaid', 'Refunded')),
    PrescriptionRequired BIT DEFAULT 0,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);

-- Creating Prescription Table (references Order)
CREATE TABLE Prescription (
    PrescriptionID INT PRIMARY KEY IDENTITY(1,1),
    OrderID INT NOT NULL,
    Details TEXT,
    DoctorName VARCHAR(100),
    VerificationStatus VARCHAR(50) CHECK (VerificationStatus IN ('Verified', 'Pending', 'Rejected')),
    FOREIGN KEY (OrderID) REFERENCES [Order](OrderID)
);

-- Creating VerificationCode Table
CREATE TABLE VerificationCode (
    VerificationCodeID INT PRIMARY KEY IDENTITY(1,1),
    ExpirationDate DATETIME NOT NULL,
    EncryptedCode VARBINARY(MAX) -- Added to store encrypted data
);

-- Creating DeliveryDriver Table (references LogisticProvider and Vehicle)
CREATE TABLE DeliveryDriver (
    DriverID INT PRIMARY KEY IDENTITY(1,1),
    ProviderID INT NOT NULL,
    VehicleID INT,
    Name VARCHAR(100) NOT NULL,
    LicenseNumber VARCHAR(20) UNIQUE NOT NULL,
    AvailabilityStatus BIT DEFAULT 1,
    FOREIGN KEY (ProviderID) REFERENCES LogisticProvider(ProviderID),
    FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)
);

-- Creating Package Table (references Order, DeliveryDriver, SmartLocker)
CREATE TABLE Package (
    PackageID INT PRIMARY KEY IDENTITY(1,1),
    OrderID INT NOT NULL,
    DriverID INT,
    LockerID INT,
    DeliveryStatus VARCHAR(50) CHECK (DeliveryStatus IN ('In Transit', 'Delivered', 'Failed')),
    FOREIGN KEY (OrderID) REFERENCES [Order](OrderID),
    FOREIGN KEY (DriverID) REFERENCES DeliveryDriver(DriverID),
    FOREIGN KEY (LockerID) REFERENCES SmartLocker(LockerID)
);

-- Creating DeliverySchedule Table (references DeliveryRoute and DeliveryDriver)
CREATE TABLE DeliverySchedule (
    ScheduleID INT PRIMARY KEY IDENTITY(1,1),
    RouteID INT NOT NULL,
    DriverID INT NOT NULL,
    StartTime DATETIME NOT NULL,
    EndTime DATETIME,
    FOREIGN KEY (RouteID) REFERENCES DeliveryRoute(RouteID),
    FOREIGN KEY (DriverID) REFERENCES DeliveryDriver(DriverID)
);

-- Creating Waypoint Table (references DeliveryRoute)
CREATE TABLE Waypoint (
    WaypointID INT PRIMARY KEY IDENTITY(1,1),
    RouteID INT NOT NULL,
    Location VARCHAR(255) NOT NULL,
    StopTime DATETIME,
    FOREIGN KEY (RouteID) REFERENCES DeliveryRoute(RouteID)
);
```

```sql
USE P4_SCHEMA;
GO

-- Insert data into LockerLocation table
INSERT INTO LockerLocation (Address, City, ZipCode) VALUES
('123 Main St', 'Boston', '02118'),
('456 Elm St', 'Cambridge', '02139'),
('789 Maple St', 'Somerville', '02145'),
('101 Pine St', 'Medford', '02155'),
('202 Oak St', 'Quincy', '02170'),
('303 Cedar St', 'Brookline', '02445'),
('404 Birch St', 'Newton', '02458'),
('505 Spruce St', 'Waltham', '02452'),
('606 Chestnut St', 'Watertown', '02472'),
('707 Walnut St', 'Malden', '02148');

-- Insert data into SmartLocker table
INSERT INTO SmartLocker (LocationID, LockerSize, IsOccupied) VALUES
(1, 'Small', 0),
(2, 'Medium', 1),
(3, 'Large', 0),
(4, 'Small', 1),
(5, 'Medium', 0),
(6, 'Large', 1),
(7, 'Small', 0),
(8, 'Medium', 1),
(9, 'Large', 0),
(10, 'Small', 1);

-- Insert data into LogisticProvider table
INSERT INTO LogisticProvider (Name, ContactNumber, Address, ZipCode, ServiceArea) VALUES
('FastShip', '1234567890', '123 Express Ln, Boston, MA', '02118', 'Greater Boston Area'),
('QuickDeliver', '2345678901', '456 Rapid Rd, Cambridge, MA', '02139', 'Cambridge Area'),
('OnTime', '3456789012', '789 Swift St, Somerville, MA', '02145', 'Somerville Area'),
('FastTrack', '4567890123', '101 Speed Way, Medford, MA', '02155', 'Medford Area'),
('PrimeExpress', '5678901234', '202 Turbo Ave, Quincy, MA', '02170', 'Quincy Area'),
('ShipNow', '6789012345', '303 Rush St, Brookline, MA', '02445', 'Brookline Area'),
('DirectCourier', '7890123456', '404 Pace Blvd, Newton, MA', '02458', 'Newton Area'),
('FlyFast', '8901234567', '505 Quick St, Waltham, MA', '02452', 'Waltham Area'),
('EzyCourier', '9012345678', '606 Accelerate Ave, Watertown, MA', '02472', 'Watertown Area'),
('SureShip', '0123456789', '707 Swift St, Malden, MA', '02148', 'Malden Area');

-- Insert data into DeliveryRoute table
INSERT INTO DeliveryRoute (StartLocation, EndLocation, EstimatedTime) VALUES
('Warehouse A', 'Customer Location X', '08:00:00'),
('Warehouse B', 'Customer Location Y', '09:00:00'),
('Warehouse C', 'Customer Location Z', '07:30:00'),
('Warehouse D', 'Customer Location W', '06:45:00'),
('Warehouse E', 'Customer Location V', '09:15:00'),
('Warehouse F', 'Customer Location U', '10:00:00'),
('Warehouse G', 'Customer Location T', '08:20:00'),
('Warehouse H', 'Customer Location S', '07:50:00'),
('Warehouse I', 'Customer Location R', '06:30:00'),
('Warehouse J', 'Customer Location Q', '09:45:00');

-- Insert data into Customer table
INSERT INTO Customer (Name, Email, PhoneNumber, StreetAddress, ZipCode, City, PreferredLockerLocation, DeliveryPreference) VALUES
('John Doe', 'john.doe@example.com', '1234567890', '123 Main St', '02118', 'Boston', 1, 'Locker Pickup'),
('Jane Smith', 'jane.smith@example.com', '2345678901', '456 Elm St', '02139', 'Cambridge', 2, 'Home Delivery'),
('Sam Brown', 'sam.brown@example.com', '3456789012', '789 Maple St', '02145', 'Somerville', 3, 'Home Delivery'),
('Lisa Black', 'lisa.black@example.com', '4567890123', '101 Pine St', '02155', 'Medford', 4, 'Locker Pickup'),
('Alice Green', 'alice.green@example.com', '5678901234', '202 Oak St', '02170', 'Quincy', 5, 'Locker Pickup'),
('Tom White', 'tom.white@example.com', '6789012345', '303 Cedar St', '02445', 'Brookline', 6, 'Home Delivery'),
('Emma Blue', 'emma.blue@example.com', '7890123456', '404 Birch St', '02458', 'Newton', 7, 'Locker Pickup'),
('Chris Red', 'chris.red@example.com', '8901234567', '505 Spruce St', '02452', 'Waltham', 8, 'Home Delivery'),
('Lucy Gray', 'lucy.gray@example.com', '9012345678', '606 Chestnut St', '02472', 'Watertown', 9, 'Home Delivery'),
('Mark Silver', 'mark.silver@example.com', '0123456789', '707 Walnut St', '02148', 'Malden', 10, 'Locker Pickup');

-- Insert data into Order table
INSERT INTO [Order] (CustomerID, OrderDate, DeliveryDate, OrderStatus, PaymentStatus, PrescriptionRequired) VALUES
(1, '2024-11-01', '2024-11-05', 'Pending', 'Paid', 1),
(2, '2024-11-02', '2024-11-06', 'Shipped', 'Unpaid', 0),
(3, '2024-11-03', '2024-11-07', 'Delivered', 'Paid', 1),
(4, '2024-11-04', '2024-11-08', 'Cancelled', 'Refunded', 0),
(5, '2024-11-05', '2024-11-09', 'Pending', 'Paid', 1),
(6, '2024-11-06', '2024-11-10', 'Shipped', 'Unpaid', 0),
(7, '2024-11-07', '2024-11-11', 'Delivered', 'Paid', 1),
(8, '2024-11-08', '2024-11-12', 'Cancelled', 'Refunded', 0),
(9, '2024-11-09', '2024-11-13', 'Pending', 'Paid', 1),
```

```sql
-- Procedures

-- 1. Assign a driver to a delivery route
IF OBJECT_ID('AssignDriverToRoute', 'P') IS NOT NULL
    DROP PROCEDURE AssignDriverToRoute;
GO
CREATE PROCEDURE AssignDriverToRoute
    @DriverID INT,
    @RouteID INT
AS
BEGIN
    UPDATE DeliverySchedule
    SET DriverID = @DriverID
    WHERE RouteID = @RouteID;
END;
GO

-- 2. Get locker utilization details
IF OBJECT_ID('GetLockerUtilization', 'P') IS NOT NULL
    DROP PROCEDURE GetLockerUtilization;
GO
CREATE PROCEDURE GetLockerUtilization
AS
BEGIN
    SELECT
        LockerSize,
        COUNT(*) AS TotalLockers,
        SUM(CASE WHEN IsOccupied = 1 THEN 1 ELSE 0 END) AS OccupiedLockers
    FROM SmartLocker
    GROUP BY LockerSize;
END;
GO

-- 3. Retrieve prescription details for an order
IF OBJECT_ID('GetPrescriptionDetails', 'P') IS NOT NULL
    DROP PROCEDURE GetPrescriptionDetails;
GO
CREATE PROCEDURE GetPrescriptionDetails
    @OrderID INT
AS
BEGIN
    SELECT *
    FROM Prescription
    WHERE OrderID = @OrderID;
END;
GO

-- 4. Verify a customer's access to a smart locker
IF OBJECT_ID('VerifyCustomerLockerAccess', 'P') IS NOT NULL
    DROP PROCEDURE VerifyCustomerLockerAccess;
GO
CREATE PROCEDURE VerifyCustomerLockerAccess
    @CustomerID INT,
    @LockerID INT
AS
BEGIN
    SELECT CASE
        WHEN EXISTS (SELECT * FROM Customer WHERE CustomerID = @CustomerID AND PreferredLockerLocation = @LockerID)
        THEN 'Access Granted'
        ELSE 'Access Denied'
    END AS AccessStatus;
END;
GO
```

VIEWS, TRIGGERS, ENCRYPTION

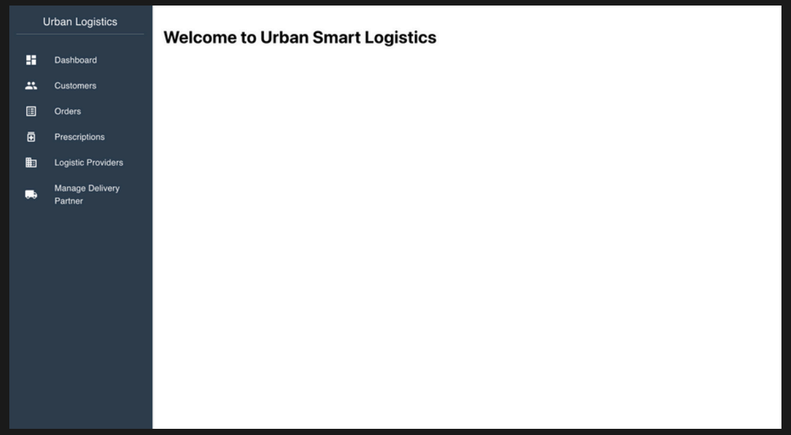
```sql
-- Views

-- 1. View for active delivery schedules
GO
IF OBJECT_ID('ActiveDeliverySchedules', 'V') IS NOT NULL
    DROP VIEW ActiveDeliverySchedules;
GO
CREATE VIEW ActiveDeliverySchedules AS
SELECT
    DS.ScheduleID,
    DR.Name AS DriverName,
    DR.AvailabilityStatus,
    DS.RouteID,
    DS.StartTime,
    DS.EndTime
FROM DeliverySchedule DS
JOIN DeliveryDriver DR ON DS.DriverID = DR.DriverID
WHERE DS.EndTime > GETDATE();
GO

-- 2. View for order and package details
IF OBJECT_ID('OrderPackageDetails', 'V') IS NOT NULL
    DROP VIEW OrderPackageDetails;
GO
CREATE VIEW OrderPackageDetails AS
SELECT
    O.OrderID,
    O.OrderStatus,
    P.PackageID,
    P.DeliveryStatus
FROM [Order] O
JOIN Package P ON O.OrderID = P.OrderID;
GO

-- 3. View for prescription verification status
IF OBJECT_ID('PrescriptionVerificationStatus', 'V') IS NOT NULL
    DROP VIEW PrescriptionVerificationStatus;
GO
CREATE VIEW PrescriptionVerificationStatus AS
SELECT
    P.PrescriptionID,
    P.OrderID,
    P.VerificationStatus,
    C.Name AS CustomerName
FROM Prescription P
JOIN [Order] O ON P.OrderID = O.OrderID
JOIN Customer C ON O.CustomerID = C.CustomerID;

GO
IF OBJECT_ID('RouteInefficiencyAnalysis', 'V') IS NOT NULL
    DROP VIEW RouteInefficiencyAnalysis;
GO
CREATE VIEW RouteInefficiencyAnalysis AS
SELECT
    dr.RouteID,
    -- Convert EstimatedTime (time) to total minutes
    DATEPART(HOUR, dr.EstimatedTime) * 60 + DATEPART(MINUTE, dr.EstimatedTime) AS EstimatedDurati
    ds.StartTime,
    ds.EndTime,
    -- Calculate actual duration in minutes
    DATEDIFF(MINUTE, ds.StartTime, ds.EndTime) AS ActualDuration,
    -- Determine inefficiency status
    CASE
        WHEN DATEDIFF(MINUTE, ds.StartTime, ds.EndTime) > (DATEPART(HOUR, dr.EstimatedTime) * 60
        WHEN DATEDIFF(MINUTE, ds.StartTime, ds.EndTime) < (DATEPART(HOUR, dr.EstimatedTime) * 60
        ELSE 'On-Time'
    END AS InefficiencyStatus
FROM DeliveryRoute dr
JOIN DeliverySchedule ds
    ON dr.RouteID = ds.RouteID;
GO
```

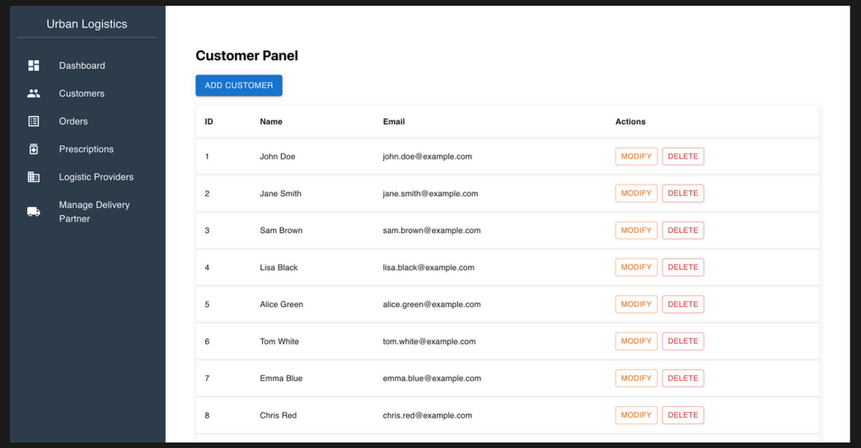
```sql
-- Triggers

-- Update locker status upon package delivery
GO
CREATE TRIGGER UpdateLockerStatus
ON Package
AFTER INSERT, UPDATE
AS
BEGIN
    UPDATE SmartLocker
    SET IsOccupied = CASE
        WHEN (SELECT DeliveryStatus FROM inserted) = 'Delivered' THEN 0
        ELSE 1
    END
    WHERE LockerID IN (SELECT LockerID FROM inserted);
END;
```

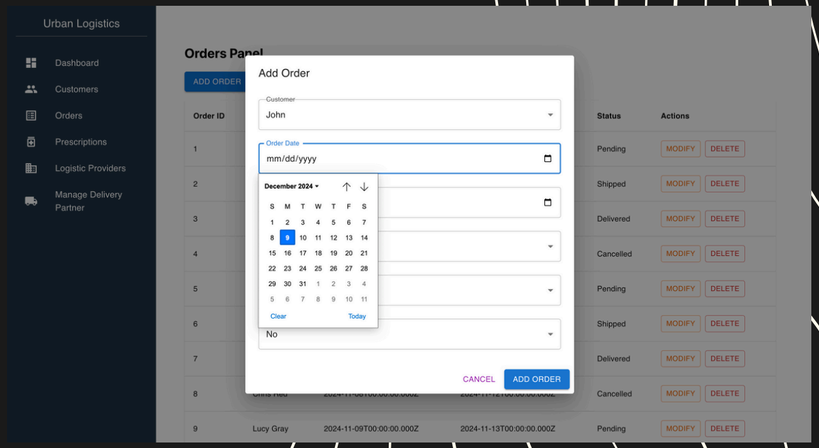
```sql
-- Indexes
USE P4_SCHEMA;
GO

-- 1. Non-clustered index on Customer Email
IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_CustomerEmail' AND object_id = OBJECT_ID('Customer'))
BEGIN
    CREATE NONCLUSTERED INDEX IX_CustomerEmail ON Customer(Email);
END;
GO

-- 2. Non-clustered index on Order Status
IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_OrderStatus' AND object_id = OBJECT_ID('[Order]'))
BEGIN
    CREATE NONCLUSTERED INDEX IX_OrderStatus ON [Order](OrderStatus);
END;
GO

-- 3. Non-clustered index on Prescription Verification Status
IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_PrescriptionStatus' AND object_id = OBJECT_ID('Prescription'))
BEGIN
    CREATE NONCLUSTERED INDEX IX_PrescriptionStatus ON Prescription(VerificationStatus);
END;
GO
```

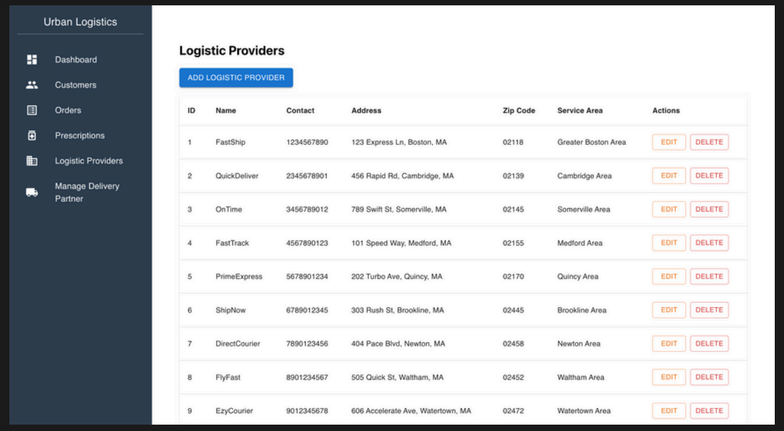
```sql
USE P4_SCHEMA;
GO

-- 1. Create a Master Key (if not already created)
IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = '##MS_DatabaseMasterKey##')
BEGIN
    CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'StrongPassword123!';
END;

-- 2. Open the Master Key for the current session
OPEN MASTER KEY DECRYPTION BY PASSWORD = 'StrongPassword123!';

-- 3. Create a Certificate and Symmetric Key (if not already created)
IF NOT EXISTS (SELECT * FROM sys.certificates WHERE name = 'MyCertificate')
BEGIN
    CREATE CERTIFICATE MyCertificate
    WITH SUBJECT = 'VerificationCode Encryption';
END;

IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = 'MySymmetricKey')
BEGIN
    CREATE SYMMETRIC KEY MySymmetricKey
    WITH ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE MyCertificate;
END;

-- 5. Encrypt existing data into the EncryptedCode column
OPEN SYMMETRIC KEY MySymmetricKey
DECRYPTION BY CERTIFICATE MyCertificate;

UPDATE VerificationCode
SET EncryptedCode = ENCRYPTBYKEY(KEY_GUID('MySymmetricKey'), CAST(ExpirationDate AS NVARCHAR(MAX)));

CLOSE SYMMETRIC KEY MySymmetricKey;

-- 6. Insert new rows with encrypted ExpirationDate values
OPEN SYMMETRIC KEY MySymmetricKey
DECRYPTION BY CERTIFICATE MyCertificate;

CLOSE SYMMETRIC KEY MySymmetricKey;

-- 7. Retrieve and decrypt data for validation
OPEN SYMMETRIC KEY MySymmetricKey
DECRYPTION BY CERTIFICATE MyCertificate;

SELECT
    VerificationCodeID,
    ExpirationDate,
    EncryptedCode,
    CONVERT(NVARCHAR(MAX), DECRYPTBYKEY(EncryptedCode)) AS DecryptedExpirationDate
FROM VerificationCode;

CLOSE SYMMETRIC KEY MySymmetricKey;


SELECT
    VerificationCodeID,
    ExpirationDate,
    EncryptedCode,
```

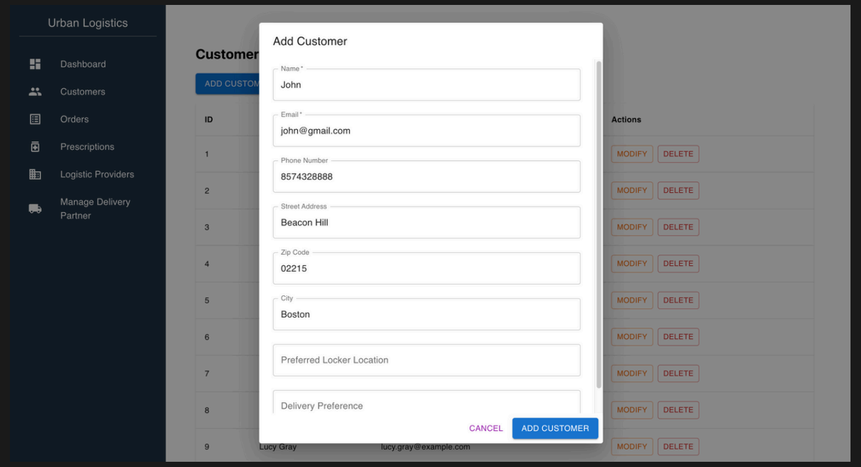
USER INTERFACE AND APPLICATION DEMO

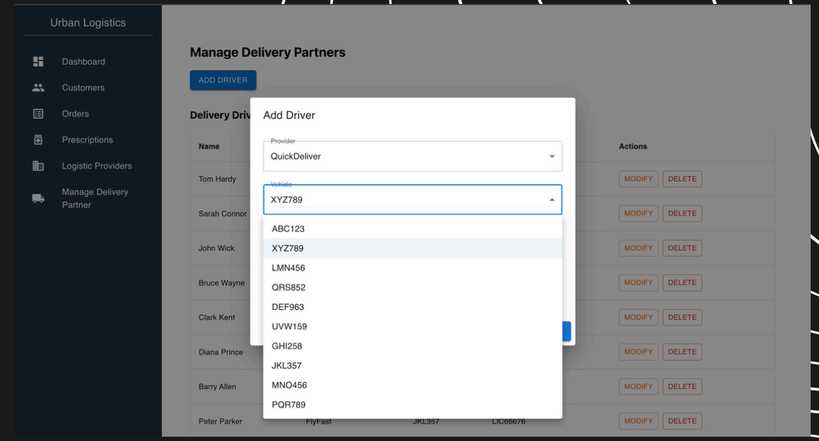








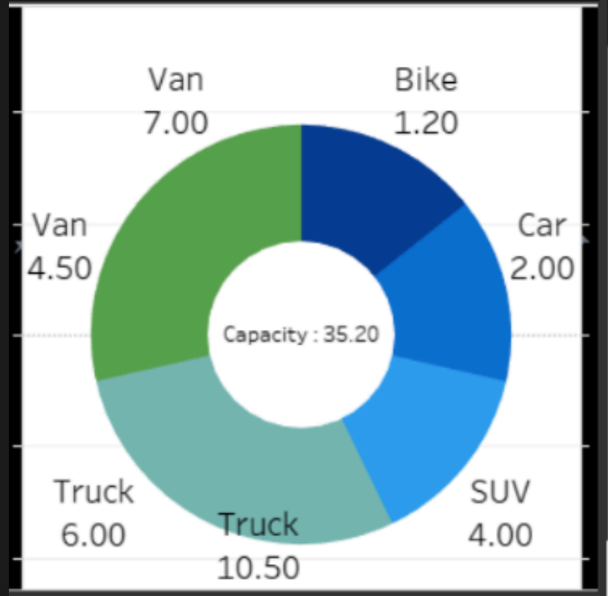




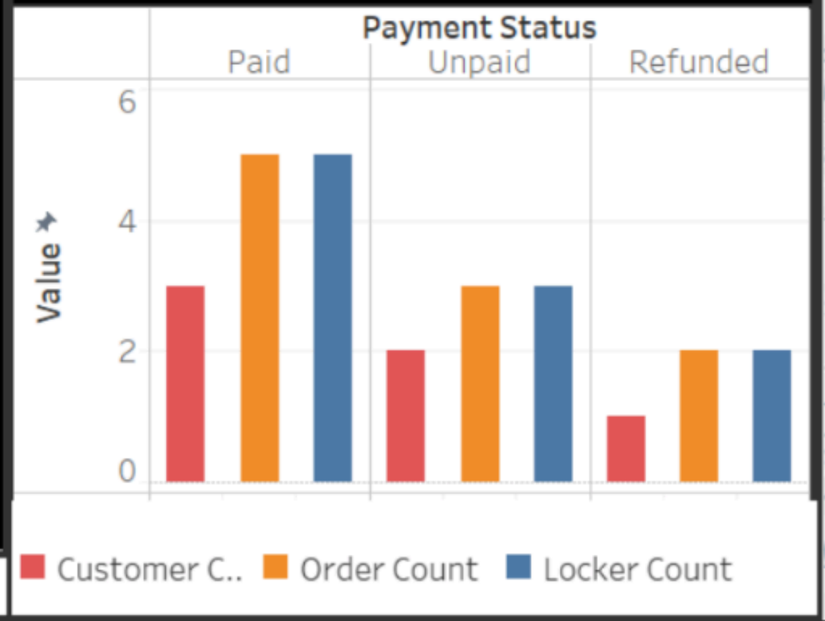
VISUALIZATION

Urban Logistics System

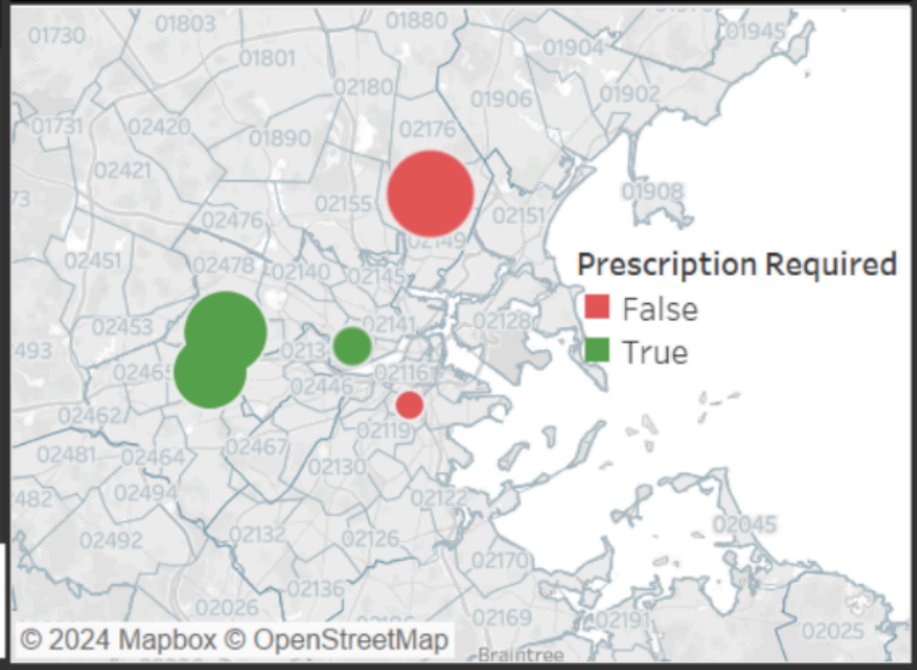
Vehicle Type & Capacity(Kg)



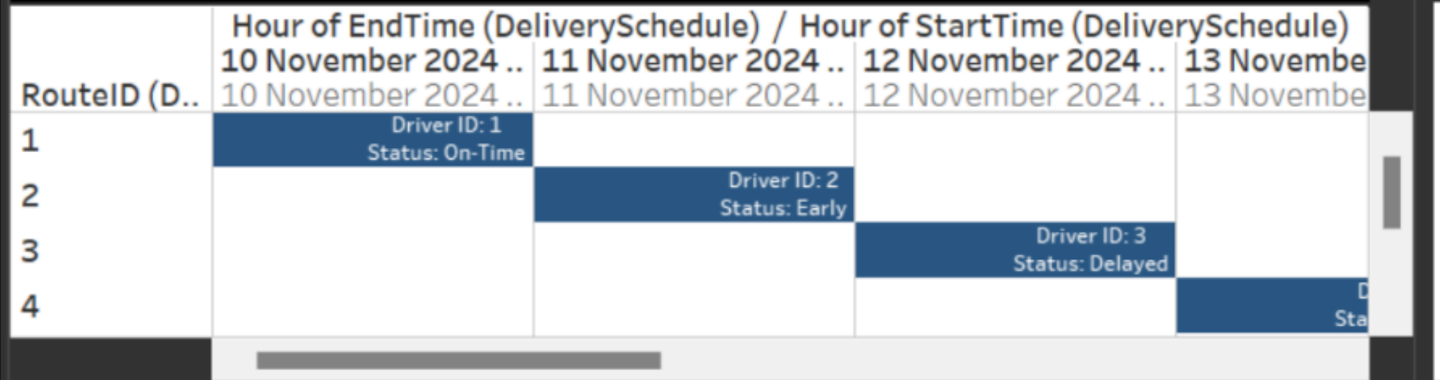
Payment Status vs Number of of Lockers, Orders & Customers



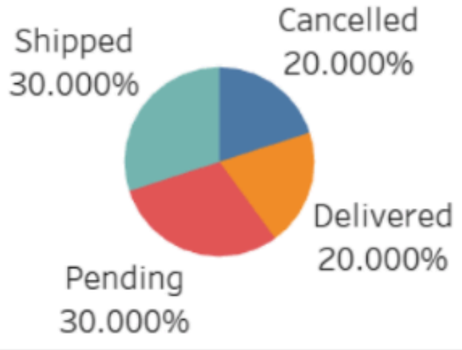
Customer- Location Analysis



Delivery status, Routes & Timings



Status of Orders



CONCLUSION

Normalized database ensures scalability and efficient data management.

Advanced encryption secures sensitive customer and delivery data.

Effective solution for modern logistics challenges.

Enhanced convenience with smart lockers and centralized data handling.

THANK YOU

Group 3

- *Pragati Narote*
- *Rutuja Dhatrak*
- *Deepika Vaddadi*
- *Samruddhi Sawant*
- *Tapas Desai*