

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**

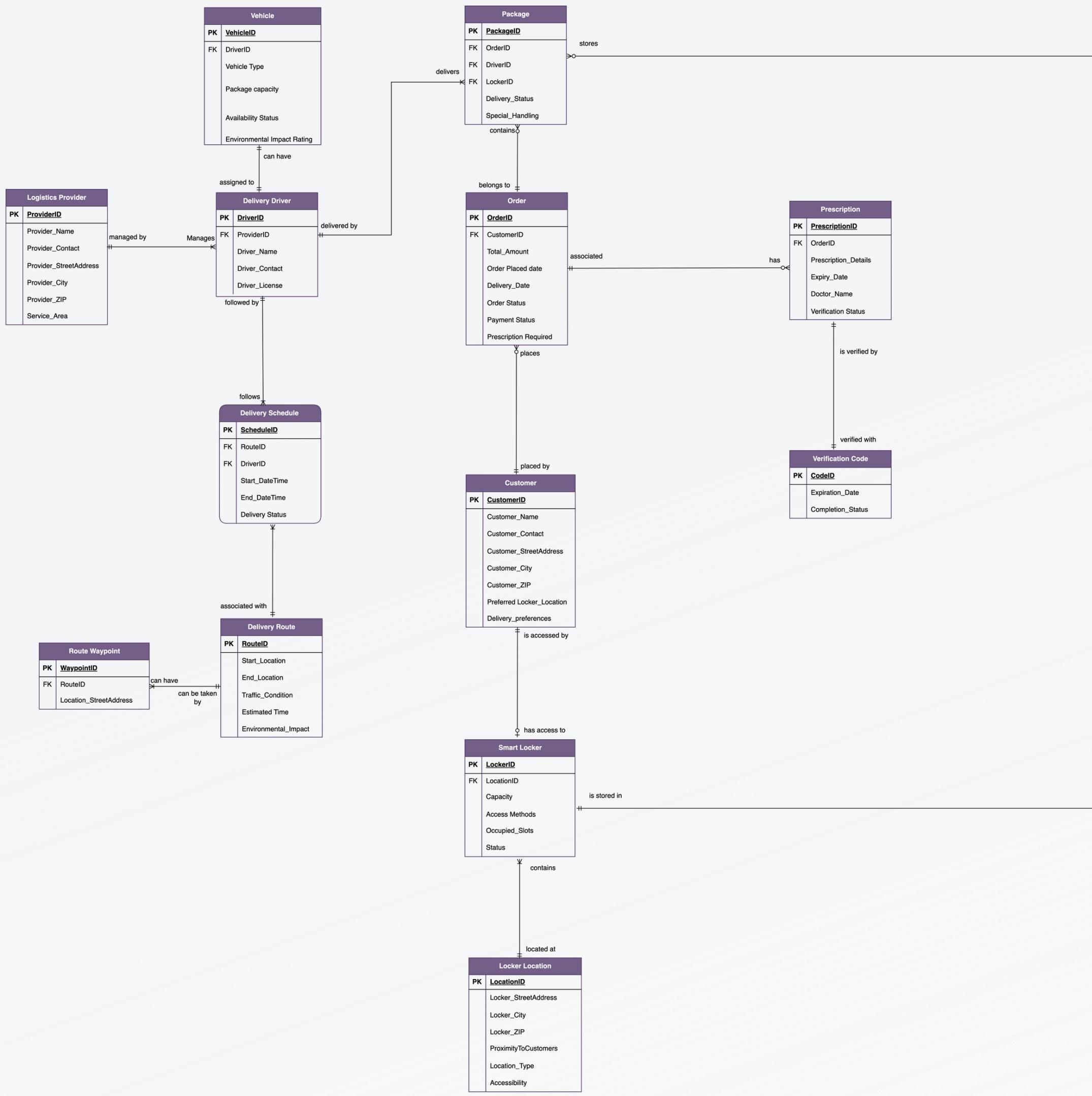
INTRODUCTION

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

```

1 -- Create Database if not exists
2 IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'P4_SCHEMA')
3 BEGIN
4     CREATE DATABASE P4_SCHEMA;
5 END
6 GO
7
8 USE P4_SCHEMA;
9 GO
10
11 -- Drop each table individually in reverse order of dependencies
12 IF OBJECT_ID('dbo.Waypoint', 'U') IS NOT NULL DROP TABLE Waypoint;
13 IF OBJECT_ID('dbo.DeliverySchedule', 'U') IS NOT NULL DROP TABLE DeliverySchedule;
14 IF OBJECT_ID('dbo.Package', 'U') IS NOT NULL DROP TABLE Package;
15 IF OBJECT_ID('dbo.Prescription', 'U') IS NOT NULL DROP TABLE Prescription;
16 IF OBJECT_ID('dbo.VerificationCode', 'U') IS NOT NULL DROP TABLE VerificationCode;
17 IF OBJECT_ID('dbo.Order', 'U') IS NOT NULL DROP TABLE Order;
18 IF OBJECT_ID('dbo.Customer', 'U') IS NOT NULL DROP TABLE Customer;
19 IF OBJECT_ID('dbo.DeliveryDriver', 'U') IS NOT NULL DROP TABLE DeliveryDriver;
20 IF OBJECT_ID('dbo.Vehicle', 'U') IS NOT NULL DROP TABLE Vehicle;
21 IF OBJECT_ID('dbo.LogisticProvider', 'U') IS NOT NULL DROP TABLE LogisticProvider;
22 IF OBJECT_ID('dbo.SmartLocker', 'U') IS NOT NULL DROP TABLE SmartLocker;
23 IF OBJECT_ID('dbo.Locke

```

```

1 USE P4_SCHEMA;
2 GO
3
4 -- Insert data into LockerLocation table
5 INSERT INTO LockerLocation (Address, City, ZipCode) VALUES
6 ('123 Main St', 'Boston', '02118'),
7 ('456 Elm St', 'Cambridge', '02139'),
8 ('789 Maple St', 'Somerville', '02145'),
9 ('101 Pine St', 'Medford', '02155'),
10 ('202 Oak St', 'Quincy', '02178'),
11 ('303 Cedar St', 'Brookline', '02445'),
12 ('404 Birch St', 'Newton', '02458'),
13 ('505 Spruce St', 'Waltham', '02452'),
14 ('606 Chestnut St', 'Watertown', '02472'),
15 ('707 Walnut St', 'Malden', '02148');
16
17 -- Insert data into SmartLocker table
18 INSERT INTO SmartLocker (LocationID, LockerSize, IsOccupied) VALUES
19 (1, 'Small', 0),
20 (2, 'Medium', 1),
21 (3, 'Large', 0),
22 (4, 'Small', 1),
23 (5, 'Medium', 0),
24 (6, 'Large', 1),
25 (7, 'Small', 0),
26 (8, 'Medium', 1),
27 (9, 'Large', 0),
28 (10, 'Small', 1);
29
30 -- Insert data into LogisticProvider table
31 INSERT INTO LogisticProvider (Name, ContactNumber, Address, ZipCode, ServiceArea) VALUES
32 ('FastShip', '1234567890', '123 Express Ln, Boston, MA', '02118', 'Greater Boston Area'),
33 ('QuickDeliver', '2345678901', '456 Rapid Rd, Cambridge, MA', '02139', 'Cambridge Area'),
34 ('OnTime', '3456789012', '789 Swift St, Somerville, MA', '02145', 'Somerville Area'),
35 ('FastTrack', '4567890123', '101 Speed Way, Medford, MA', '02155', 'Medford Area'),
36 ('PrimeExpress', '5678901234', '202 Turbo Ave, Quincy, MA', '02178', 'Quincy Area'),
37 ('ShipNow', '6789012345', '303 Rush St, Brookline, MA', '02445', 'Brookline Area'),
38 ('DirectCourier', '7890123456', '404 Pace Blvd, Newton, MA', '02458', 'Newton Area'),
39 ('FlyFast', '8901234567', '505 Quick St, Waltham, MA', '02452', 'Waltham Area'),
40 ('EzyCourier', '9012345678', '606 Accelerate Ave, Watertown, MA', '02472', 'Watertown Area'),
41 ('SureShip', '0123456789', '707 Swift St, Malden, MA', '02148', 'Malden Area');
42
43 -- Insert data into DeliveryRoute table
44 INSERT INTO DeliveryRoute (StartLocation, EndLocation, EstimatedTime) VALUES
45 ('Warehouse A', 'Customer Location X', '08:00:00'),
46 ('Warehouse B', 'Customer Location Y', '09:00:00'),
47 ('Warehouse C', 'Customer Location Z', '07:30:00'),
48 ('Warehouse D', 'Customer Location W', '06:45:00'),
49 ('Warehouse E', 'Customer Location V', '09:15:00'),
50 ('Warehouse F', 'Customer Location U', '10:00:00'),
51 ('Warehouse G', 'Customer Location T', '08:10:00'),
52 ('Warehouse H', 'Customer Location S', '07:50:00'),
53 ('Warehouse I', 'Customer Location R', '06:30:00'),
54 ('Warehouse J', 'Customer Location Q', '09:45:00');
55
56 -- Insert data into Customer table
57 INSERT INTO Customer (Name, Email, PhoneNumber, StreetAddress, ZipCode, City, PreferredLockerLocation, DeliveryPreference) VALUES
58 ('John Doe', 'john.doe@example.com', '1234567890', '123 Main St', '02118', 'Boston', 1, 'Locker Pickup'),
59 ('Jane Smith', 'jane.smith@example.com', '2345678901', '456 Elm St', '02139', 'Cambridge', 2, 'Home Delivery'),
60 ('Sam Brown', 'sam.brown@example.com', '3456789012', '789 Maple St', '02145', 'Somerville', 3, 'Home Delivery'),
61 ('Lisa Black', 'lisa.black@example.com', '4567890123', '101 Pine St', '02155', 'Medford', 4, 'Locker Pickup'),
62 ('Alice Green', 'alice.green@example.com', '5678901234', '202 Oak St', '02178', 'Quincy', 5, 'Locker Pickup'),
63 ('Tom White', 'tom.white@example.com', '6789012345', '303 Cedar St', '02445', 'Brookline', 6, 'Home Delivery'),
64 ('Ema Blue', 'ema.blue@example.com', '7890123456', '404 Birch St', '02458', 'Newton', 7, 'Locker Pickup'),
65 ('Chris Red', 'chris.red@example.com', '8901234567', '505 Spruce St', '02452', 'Waltham', 8, 'Home Delivery'),
66 ('Lucy Gray', 'lucy.gray@example.com', '9012345678', '606 Chestnut St', '02472', 'Watertown', 9, 'Home Delivery'),
67 ('Mark Silver', 'mark.silver@example.com', '0123456789', '707 Walnut St', '02148', 'Malden', 10, 'Locker Pickup');
68
69 -- Insert data into Order table
70 INSERT INTO Order (CustomerID, OrderDate, DeliveryDate, OrderStatus, PaymentStatus, PrescriptionRequired) VALUES
71 (1, '2024-11-01', '2024-11-05', 'Pending', 'Paid', 1),
72 (2, '2024-11-02', '2024-11-06', 'Shipped', 'Unpaid', 0),
73 (3, '2024-11-03', '2024-11-07', 'Delivered', 'Paid', 1),
74 (4, '2024-11-04', '2024-11-08', 'Cancelled', 'Refunded', 0),
75 (5, '2024-11-05', '2024-11-09', 'Pending', 'Paid', 1),
76 (6, '2024-11-06', '2024-11-10', 'Shipped', 'Unpaid', 0),
77 (7, '2024-11-07', '2024-11-11', 'Delivered', 'Paid', 1),
78 (8, '2024-11-08', '2024-11-12', 'Cancelled', 'Refunded', 0),
79 (9, '2024-11-09', '2024-11-13', 'Pending', 'Paid', 1),
80 (10, '2024-11-10', '2024-11-14', 'Cancelled', 'Refunded', 0);
81
82 -- Creating Customer Table (references SmartLocker)
83 CREATE TABLE Customer (
84     CustomerID INT PRIMARY KEY IDENTITY(1,1),
85     Name VARCHAR(100) NOT NULL,
86     Email VARCHAR(255) UNIQUE NOT NULL,
87     PhoneNumber VARCHAR(15),
88     StreetAddress VARCHAR(255),
89     ZipCode VARCHAR(10),
90     City VARCHAR(100),
91     PreferredLockerLocation INT,
92     DeliveryPreference VARCHAR(50),
93     CHECK (LEN(PhoneNumber) = 10),
94     FOREIGN KEY (PreferredLockerLocation) REFERENCES SmartLocker(LockerID)
95 );
96
97 -- Creating Order Table (references Customer)
98 CREATE TABLE [Order] (
99     OrderID INT PRIMARY KEY IDENTITY(1,1),
100    CustomerID INT NOT NULL,
101    OrderDate DATETIME DEFAULT GETDATE(),
102    DeliveryDate DATETIME,
103    OrderStatus VARCHAR(50) CHECK (OrderStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
104    PaymentStatus VARCHAR(50) CHECK (PaymentStatus IN ('Paid', 'Unpaid', 'Refunded')),
105    PrescriptionRequired BIT DEFAULT 0,
106    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
107 );
108
109 -- Creating Order Table (references Customer)
110 CREATE TABLE [Order] (
111     OrderID INT PRIMARY KEY IDENTITY(1,1),
112     CustomerID INT NOT NULL,
113     OrderDate DATETIME DEFAULT GETDATE(),
114     DeliveryDate DATETIME,
115     OrderStatus VARCHAR(50) CHECK (OrderStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
116     PaymentStatus VARCHAR(50) CHECK (PaymentStatus IN ('Paid', 'Unpaid', 'Refunded')),
117     PrescriptionRequired BIT DEFAULT 0,
118     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
119 );
120
121 -- Creating Prescription Table (references Order)
122 CREATE TABLE Prescription (
123     PrescriptionID INT PRIMARY KEY IDENTITY(1,1),
124     OrderID INT NOT NULL,
125     Details TEXT,
126     DoctorName VARCHAR(100),
127     VerificationStatus VARCHAR(50) CHECK (VerificationStatus IN ('Verified', 'Pending', 'Rejected')),
128     FOREIGN KEY (OrderID) REFERENCES [Order](OrderID)
129 );
130
131 -- Creating VerificationCode Table
132 CREATE TABLE VerificationCode (
133     VerificationCodeID INT PRIMARY KEY IDENTITY(1,1),
134     ExpirationDate DATETIME NOT NULL,
135     EncryptedCode VARBINARY(MAX) -- Added to store encrypted data
136 );
137
138 -- Creating DeliveryDriver Table (references LogisticProvider and Vehicle)
139 CREATE TABLE DeliveryDriver (
140     DriverID INT PRIMARY KEY IDENTITY(1,1),
141     LocationID INT NOT NULL,
142     LockerSize VARCHAR(20) CHECK (LockerSize IN ('Small', 'Medium', 'Large')),
143     IsOccupied BIT DEFAULT 0,
144     FOREIGN KEY (LocationID) REFERENCES LockerLocation(LocationID)
145 );
146
147 -- Creating LogisticProvider Table
148 CREATE TABLE LogisticProvider (
149     ProviderID INT PRIMARY KEY IDENTITY(1,1),
150     Name VARCHAR(100) NOT NULL,
151     ContactNumber VARCHAR(15),
152     Address VARCHAR(255),
153     ZipCode VARCHAR(10),
154     ServiceArea VARCHAR(100)
155 );
156
157 -- Creating Package Table (references Order, DeliveryDriver, SmartLocker)
158 CREATE TABLE Package (
159     PackageID INT PRIMARY KEY IDENTITY(1,1),
160     OrderID INT NOT NULL,
161     DriverID INT,
162     LockerID INT,
163     DeliveryStatus VARCHAR(50) CHECK (DeliveryStatus IN ('In Transit', 'Delivered', 'Failed')),
164     FOREIGN KEY (OrderID) REFERENCES [Order](OrderID),
165     FOREIGN KEY (DriverID) REFERENCES DeliveryDriver(DriverID),
166     FOREIGN KEY (LockerID) REFERENCES SmartLocker(LockerID)
167 );
168
169 -- Creating DeliveryRoute Table
170 CREATE TABLE DeliveryRoute (
171     RouteID INT PRIMARY KEY IDENTITY(1,1),
172     StartLocation VARCHAR(255) NOT NULL,
173     EndLocation VARCHAR(255) NOT NULL,
174     EstimatedTime TIME
175 );
176
177 -- Creating DeliverySchedule Table (references DeliveryRoute and DeliveryDriver)
178 CREATE TABLE DeliverySchedule (
179     ScheduleID INT PRIMARY KEY IDENTITY(1,1),
180     RouteID INT NOT NULL,
181     DriverID INT NOT NULL,
182     StartTime DATETIME NOT NULL,
183     EndTime DATETIME,
184     FOREIGN KEY (RouteID) REFERENCES DeliveryRoute(RouteID),
185     FOREIGN KEY (DriverID) REFERENCES DeliveryDriver(DriverID)
186 );
187
188 -- Creating Customer Table (references SmartLocker)
189 CREATE TABLE Customer (
190     CustomerID INT PRIMARY KEY IDENTITY(1,1),
191     Name VARCHAR(100) NOT NULL,
192     Email VARCHAR(255) UNIQUE NOT NULL,
193     PhoneNumber VARCHAR(15),
194     StreetAddress VARCHAR(255),
195     ZipCode VARCHAR(10),
196     City VARCHAR(100),
197     PreferredLockerLocation INT,
198     DeliveryPreference VARCHAR(50),
199     CHECK (LEN(PhoneNumber) = 10),
200     FOREIGN KEY (PreferredLockerLocation) REFERENCES SmartLocker(LockerID)
201 );
202
203 -- Creating Order Table (references Customer)
204 CREATE TABLE [Order] (
205     OrderID INT PRIMARY KEY IDENTITY(1,1),
206     CustomerID INT NOT NULL,
207     OrderDate DATETIME DEFAULT GETDATE(),
208     DeliveryDate DATETIME,
209     OrderStatus VARCHAR(50) CHECK (OrderStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
210     PaymentStatus VARCHAR(50) CHECK (PaymentStatus IN ('Paid', 'Unpaid', 'Refunded')),
211     PrescriptionRequired BIT DEFAULT 0,
212     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
213 );
214
215 -- Creating Order Table (references Customer)
216 CREATE TABLE [Order] (
217     OrderID INT PRIMARY KEY IDENTITY(1,1),
218     CustomerID INT NOT NULL,
219     OrderDate DATETIME DEFAULT GETDATE(),
220     DeliveryDate DATETIME,
221     OrderStatus VARCHAR(50) CHECK (OrderStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
222     PaymentStatus VARCHAR(50) CHECK (PaymentStatus IN ('Paid', 'Unpaid', 'Refunded')),
223     PrescriptionRequired BIT DEFAULT 0,
224     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
225 );
226
227 -- Procedures
228
229 -- 1. Assign a driver to a delivery route
230 IF OBJECT_ID('AssignDriverToRoute', 'P') IS NOT NULL
231     DROP PROCEDURE AssignDriverToRoute;
232 GO
233
234 CREATE PROCEDURE AssignDriverToRoute
235     @DriverID INT,
236     @RouteID INT
237 AS
238 BEGIN
239     UPDATE DeliverySchedule
240     SET DriverID = @DriverID
241     WHERE RouteID = @RouteID;
242 END;
243 GO
244
245 -- 2. Get locker utilization details
246 IF OBJECT_ID('GetLockerUtilization', 'P') IS NOT NULL
247     DROP PROCEDURE GetLockerUtilization;
248 GO
249
250 CREATE PROCEDURE GetLockerUtilization
251 AS
252 BEGIN
253     SELECT
254         LockerSize,
255         COUNT(*) AS TotalLockers,
256         SUM(CASE WHEN IsOccupied = 1 THEN 1 ELSE 0 END) AS OccupiedLockers
257     FROM SmartLocker
258     GROUP BY LockerSize;
259 END;
260 GO
261
262 -- 3. Retrieve prescription details for an order
263 IF OBJECT_ID('GetPrescriptionDetails', 'P') IS NOT NULL
264     DROP PROCEDURE GetPrescriptionDetails;
265 GO
266
267 CREATE PROCEDURE GetPrescriptionDetails
268     @OrderID INT
269 AS
270 BEGIN
271     SELECT *
272     FROM Prescription
273     WHERE OrderID = @OrderID;
274 END;
275 GO
276
277 -- 4. Verify a customer's access to a smart locker
278 IF OBJECT_ID('VerifyCustomerLockerAccess', 'P') IS NOT NULL
279     DROP PROCEDURE VerifyCustomerLockerAccess;
280 GO
281
282 CREATE PROCEDURE VerifyCustomerLockerAccess
283     @CustomerID INT,
284     @LockerID INT
285 AS
286 BEGIN
287     BEGIN CASE
288         WHEN EXISTS (SELECT * FROM Customer WHERE CustomerID = @CustomerID AND PreferredLockerLocation = @LockerID)
289             THEN 'Access Granted'
290         ELSE 'Access Denied'
291     END AS AccessStatus;
292 END;
293 GO
294
295 -- 5. Check if a specific order has been delivered
296 IF OBJECT_ID('CheckOrderStatus', 'P') IS NOT NULL
297     DROP PROCEDURE CheckOrderStatus;
298 GO
299
300 CREATE PROCEDURE CheckOrderStatus
301     @OrderID INT
302 AS
303 BEGIN
304     SELECT Status
305     FROM Order
306     WHERE OrderID = @OrderID;
307 END;
308 GO
309
310 -- 6. Get all orders for a specific customer
311 IF OBJECT_ID('GetCustomerOrders', 'P') IS NOT NULL
312     DROP PROCEDURE GetCustomerOrders;
313 GO
314
315 CREATE PROCEDURE GetCustomerOrders
316     @CustomerID INT
317 AS
318 BEGIN
319     SELECT *
320     FROM Order
321     WHERE CustomerID = @CustomerID;
322 END;
323 GO
324
325 -- 7. Get all drivers assigned to a specific route
326 IF OBJECT_ID('GetAssignedDrivers', 'P') IS NOT NULL
327     DROP PROCEDURE GetAssignedDrivers;
328 GO
329
330 CREATE PROCEDURE GetAssignedDrivers
331     @RouteID INT
332 AS
333 BEGIN
334     SELECT DriverID
335     FROM DeliverySchedule
336     WHERE RouteID = @RouteID;
337 END;
338 GO
339
340 -- 8. Get all lockers in a specific size
341 IF OBJECT_ID('GetLockersBySize', 'P') IS NOT NULL
342     DROP PROCEDURE GetLockersBySize;
343 GO
344
345 CREATE PROCEDURE GetLockersBySize
346     @LockerSize INT
347 AS
348 BEGIN
349     SELECT *
350     FROM SmartLocker
351     WHERE LockerSize = @LockerSize;
352 END;
353 GO
354
355 -- 9. Get all occupied lockers
356 IF OBJECT_ID('GetOccupiedLockers', 'P') IS NOT NULL
357     DROP PROCEDURE GetOccupiedLockers;
358 GO
359
360 CREATE PROCEDURE GetOccupiedLockers
361 AS
362 BEGIN
363     SELECT *
364     FROM SmartLocker
365     WHERE IsOccupied = 1;
366 END;
367 GO
368
369 -- 10. Get all drivers assigned to a specific route
370 IF OBJECT_ID('GetAssignedDriversForRoute', 'P') IS NOT NULL
371     DROP PROCEDURE GetAssignedDriversForRoute;
372 GO
373
374 CREATE PROCEDURE GetAssignedDriversForRoute
375     @RouteID INT
376 AS
377 BEGIN
378     SELECT DriverID
379     FROM DeliverySchedule
380     WHERE RouteID = @RouteID;
381 END;
382 GO
383
384 -- 11. Get all prescriptions for a specific order
385 IF OBJECT_ID('GetPrescriptionsForOrder', 'P') IS NOT NULL
386     DROP PROCEDURE GetPrescriptionsForOrder;
387 GO
388
389 CREATE PROCEDURE GetPrescriptionsForOrder
390     @OrderID INT
391 AS
392 BEGIN
393     SELECT *
394     FROM Prescription
395     WHERE OrderID = @OrderID;
396 END;
397 GO
398
399 -- 12. Get all lockers in a specific size
400 IF OBJECT_ID('GetLockersBySize', 'P') IS NOT NULL
401     DROP PROCEDURE GetLockersBySize;
402 GO
403
404 CREATE PROCEDURE GetLockersBySize
405     @LockerSize INT
406 AS
407 BEGIN
408     SELECT *
409     FROM SmartLocker
410     WHERE LockerSize = @LockerSize;
411 END;
412 GO
413
414 -- 13. Get all occupied lockers
415 IF OBJECT_ID('GetOccupiedLockers', 'P') IS NOT NULL
416     DROP PROCEDURE GetOccupiedLockers;
417 GO
418
419 CREATE PROCEDURE GetOccupiedLockers
420 AS
421 BEGIN
422     SELECT *
423     FROM SmartLocker
424     WHERE IsOccupied = 1;
425 END;
426 GO
427
428 -- 14. Get all drivers assigned to a specific route
429 IF OBJECT_ID('GetAssignedDriversForRoute', 'P') IS NOT NULL
430     DROP PROCEDURE GetAssignedDriversForRoute;
431 GO
432
433 CREATE PROCEDURE GetAssignedDriversForRoute
434     @RouteID INT
435 AS
436 BEGIN
437     SELECT DriverID
438     FROM DeliverySchedule
439     WHERE RouteID = @RouteID;
440 END;
441 GO
442
443 -- 15. Get all prescriptions for a specific order
444 IF OBJECT_ID('GetPrescriptionsForOrder', 'P') IS NOT NULL
445     DROP PROCEDURE GetPrescriptionsForOrder;
446 GO
447
448 CREATE PROCEDURE GetPrescriptionsForOrder
449     @OrderID INT
450 AS
451 BEGIN
452     SELECT *
453     FROM Prescription
454     WHERE OrderID = @OrderID;
455 END;
456 GO
457
458 -- 16. Get all lockers in a specific size
459 IF OBJECT_ID('GetLockersBySize', 'P') IS NOT NULL
460     DROP PROCEDURE GetLockersBySize;
461 GO
462
463 CREATE PROCEDURE GetLockersBySize
464     @LockerSize INT
465 AS
466 BEGIN
467     SELECT *
468     FROM SmartLocker
469     WHERE LockerSize = @LockerSize;
470 END;
471 GO
472
473 -- 17. Get all occupied lockers
474 IF OBJECT_ID('GetOccupiedLockers', 'P') IS NOT NULL
475     DROP PROCEDURE GetOccupiedLockers;
476 GO
477
478 CREATE PROCEDURE GetOccupiedLockers
479 AS
480 BEGIN
481     SELECT *
482     FROM SmartLocker
483     WHERE IsOccupied = 1;
484 END;
485 GO
486
487 -- 18. Get all drivers assigned to a specific route
488 IF OBJECT_ID('GetAssignedDriversForRoute', 'P') IS NOT NULL
489     DROP PROCEDURE GetAssignedDriversForRoute;
490 GO
491
492 CREATE PROCEDURE GetAssignedDriversForRoute
493     @RouteID INT
494 AS
495 BEGIN
496     SELECT DriverID
497     FROM DeliverySchedule
498     WHERE RouteID = @RouteID;
499 END;
499 GO
500
501 -- 19. Get all prescriptions for a specific order
502 IF OBJECT_ID('GetPrescriptionsForOrder', 'P') IS NOT NULL
503     DROP PROCEDURE GetPrescriptionsForOrder;
504 GO
505
506 CREATE PROCEDURE GetPrescriptionsForOrder
507     @OrderID INT
508 AS
509 BEGIN
510     SELECT *
511     FROM Prescription
512     WHERE OrderID = @OrderID;
513 END;
514 GO
515
516 -- 20. Get all lockers in a specific size
517 IF OBJECT_ID('GetLockersBySize', 'P') IS NOT NULL
518     DROP PROCEDURE GetLockersBySize;
519 GO
520
521 CREATE PROCEDURE GetLockersBySize
522     @LockerSize INT
523 AS
524 BEGIN
525     SELECT *
526     FROM SmartLocker
527     WHERE LockerSize = @LockerSize;
528 END;
529 GO
530
531 -- 21. Get all occupied lockers
532 IF OBJECT_ID('GetOccupiedLockers', 'P') IS NOT NULL
533     DROP PROCEDURE GetOccupiedLockers;
534 GO
535
536 CREATE PROCEDURE GetOccupiedLockers
537 AS
538 BEGIN
539     SELECT *
540     FROM SmartLocker
541     WHERE IsOccupied = 1;
542 END;
543 GO
544
545 -- 22. Get all drivers assigned to a specific route
546 IF OBJECT_ID('GetAssignedDriversForRoute', 'P') IS NOT NULL
547     DROP PROCEDURE GetAssignedDriversForRoute;
548 GO
549
550 CREATE PROCEDURE GetAssignedDriversForRoute
551     @RouteID INT
552 AS
553 BEGIN
554     SELECT DriverID
555     FROM DeliverySchedule
556     WHERE RouteID = @RouteID;
557 END;
558 GO
559
560 -- 23. Get all prescriptions for a specific order
561 IF OBJECT_ID('GetPrescriptionsForOrder', 'P') IS NOT NULL
562     DROP PROCEDURE GetPrescriptionsForOrder;
563 GO
564
565 CREATE PROCEDURE GetPrescriptionsForOrder
566     @OrderID INT
567 AS
568 BEGIN
569     SELECT *
570     FROM Prescription
571     WHERE OrderID = @OrderID;
572 END;
573 GO
574
575 -- 24. Get all lockers in a specific size
576 IF OBJECT_ID('GetLockersBySize', 'P') IS NOT NULL
577     DROP PROCEDURE GetLockersBySize;
578 GO
579
580 CREATE PROCEDURE GetLockersBySize
581     @LockerSize INT
582 AS
583 BEGIN
584     SELECT *
585     FROM SmartLocker
586     WHERE LockerSize = @LockerSize;
587 END;
588 GO
589
590 -- 25. Get all occupied lockers
591 IF OBJECT_ID('GetOccupiedLockers', 'P') IS NOT NULL
592     DROP PROCEDURE GetOccupiedLockers;
593 GO
594
595 CREATE PROCEDURE GetOccupiedLockers
596 AS
597 BEGIN
598     SELECT *
599     FROM SmartLocker
600     WHERE IsOccupied = 1;
601 END;
602 GO
603
604 -- 26. Get all drivers assigned to a specific route
605 IF OBJECT_ID('GetAssignedDriversForRoute', 'P') IS NOT NULL
606     DROP PROCEDURE GetAssignedDriversForRoute;
607 GO
608
609 CREATE PROCEDURE GetAssignedDriversForRoute
610     @RouteID INT
611 AS
612 BEGIN
613     SELECT DriverID
614     FROM DeliverySchedule
615     WHERE RouteID = @RouteID;
616 END;
617 GO
618
619 -- 27. Get all prescriptions for a specific order
620 IF OBJECT_ID('GetPrescriptionsForOrder', 'P') IS NOT NULL
621     DROP PROCEDURE GetPrescriptionsForOrder;
622 GO
623
624 CREATE PROCEDURE GetPrescriptionsForOrder
625     @OrderID INT
626 AS
627 BEGIN
628     SELECT *
629     FROM Prescription
630     WHERE OrderID = @OrderID;
631 END;
632 GO
633
634 -- 28. Get all lockers in a specific size
635 IF OBJECT_ID('GetLockersBySize', 'P') IS NOT NULL
636     DROP PROCEDURE GetLockersBySize;
637 GO
638
639 CREATE PROCEDURE GetLockersBySize
640     @LockerSize INT
641 AS
642
```

VIEWS, TRIGGERS, ENCRYPTION

```
65  -- Views
66
67  -- 1. View for active delivery schedules
68  GO
69  IF OBJECT_ID('ActiveDeliverySchedules', 'V') IS NOT NULL
70  |  DROP VIEW ActiveDeliverySchedules;
71  GO
72  CREATE VIEW ActiveDeliverySchedules AS
73  SELECT
74  |  DS.ScheduleID,
75  |  DR.Name AS DriverName,
76  |  DR.AvailabilityStatus,
77  |  DS.RouteID,
78  |  DS.StartTime,
79  |  DS.EndTime
80  FROM DeliverySchedule DS
81  JOIN DeliveryDriver DR ON DS.DriverID = DR.DriverID
82  WHERE DS.EndTime > GETDATE();
83  GO
84
85  -- 2. View for order and package details
86  IF OBJECT_ID('OrderPackageDetails', 'V') IS NOT NULL
87  |  DROP VIEW OrderPackageDetails;
88  GO
89  CREATE VIEW OrderPackageDetails AS
90  SELECT
91  |  O.OrderID,
92  |  O.OrderStatus,
93  |  P.PackageID,
94  |  P.DeliveryStatus
95  FROM [Order] O
96  JOIN Package P ON O.OrderID = P.OrderID;
97  GO
98
99  -- 3. View for prescription verification status
100 IF OBJECT_ID('PrescriptionVerificationStatus', 'V') IS NOT NULL
101 |  DROP VIEW PrescriptionVerificationStatus;
102 GO
103 CREATE VIEW PrescriptionVerificationStatus AS
104 SELECT
105 |  P.PrescriptionID,
106 |  P.OrderID,
107 |  P.VerificationStatus,
108 |  C.Name AS CustomerName
109 FROM Prescription P
110 JOIN [Order] O ON P.OrderID = O.OrderID
111 JOIN Customer C ON O.CustomerID = C.CustomerID;
112
113 GO
114 IF OBJECT_ID('RouteInefficiencyAnalysis', 'V') IS NOT NULL
115 |  DROP VIEW RouteInefficiencyAnalysis;
116 GO
117 CREATE VIEW RouteInefficiencyAnalysis AS
118 SELECT
119 |  dr.RouteID,
120  -- Convert EstimatedTime (time) to total minutes
121 |  DATEPART(HOUR, dr.EstimatedTime) * 60 + DATEPART(MINUTE, dr.EstimatedTime) AS EstimatedDuration,
122 |  ds.StartTime,
123 |  ds.EndTime,
124  -- Calculate actual duration in minutes
125 |  DATEDIFF(MINUTE, ds.StartTime, ds.EndTime) AS ActualDuration,
126  -- Determine inefficiency status
127 CASE
128 |  WHEN DATEDIFF(MINUTE, ds.StartTime, ds.EndTime) > (DATEPART(HOUR, dr.EstimatedTime) * 60
129 |  WHEN DATEDIFF(MINUTE, ds.StartTime, ds.EndTime) < (DATEPART(HOUR, dr.EstimatedTime) * 60
130 |  ELSE 'On-Time'
131 END AS InefficiencyStatus
132 FROM DeliveryRoute dr
133 JOIN DeliverySchedule ds
134 ON dr.RouteID = ds.RouteID;
135 GO
```

```
137  -- Triggers
138
139  -- Update locker status upon package delivery
140  GO
141  CREATE TRIGGER UpdateLockerStatus
142  ON Package
143  AFTER INSERT, UPDATE
144  AS
145  BEGIN
146    UPDATE SmartLocker
147    SET IsOccupied = CASE
148      WHEN (SELECT DeliveryStatus FROM inserted) = 'Delivered' THEN 0
149      ELSE 1
150    END
151    WHERE LockerID IN (SELECT LockerID FROM inserted);
152  END;
```

```
1  -- Indexes
2  USE P4_SCHEMA;
3  GO
4
5  -- 1. Non-clustered index on Customer Email
6  IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_CustomerEmail' AND object_id = OBJECT_ID('Customer'))
7  BEGIN
8    |  CREATE NONCLUSTERED INDEX IX_CustomerEmail ON Customer(Email);
9  END;
10 GO
11
12 -- 2. Non-clustered index on Order Status
13 IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_OrderStatus' AND object_id = OBJECT_ID('[Order]'))
14 BEGIN
15   |  CREATE NONCLUSTERED INDEX IX_OrderStatus ON [Order](OrderStatus);
16 END;
17 GO
18
19 -- 3. Non-clustered index on Prescription Verification Status
20 IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_PrescriptionStatus' AND object_id = OBJECT_ID('Prescription'))
21 BEGIN
22   |  CREATE NONCLUSTERED INDEX IX_PrescriptionStatus ON Prescription(VerificationStatus);
23 END;
24 GO
```

```
1  USE P4_SCHEMA;
2  GO
3
4  -- 1. Create a Master Key (if not already created)
5  IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = '#MS_DatabaseMasterKey#')
6  BEGIN
7    |  CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'StrongPassword123!';
8  END;
9
10 -- 2. Open the Master Key for the current session
11 OPEN MASTER KEY DECRYPTION BY PASSWORD = 'StrongPassword123!';
12
13 -- 3. Create a Certificate and Symmetric Key (if not already created)
14 IF NOT EXISTS (SELECT * FROM sys.certificates WHERE name = 'MyCertificate')
15 BEGIN
16   |  CREATE CERTIFICATE MyCertificate
17   |  WITH SUBJECT = 'VerificationCode Encryption';
18 END;
19
20 IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = 'MySymmetricKey')
21 BEGIN
22   |  CREATE SYMMETRIC KEY MySymmetricKey
23   |  WITH ALGORITHM = AES_256
24   |  ENCRYPTION BY CERTIFICATE MyCertificate;
25 END;
26
27 -- 5. Encrypt existing data into the EncryptedCode column
28 OPEN SYMMETRIC KEY MySymmetricKey
29 DECRYPTION BY CERTIFICATE MyCertificate;
30
31 UPDATE VerificationCode
32 SET EncryptedCode = ENCRYPTBYKEY(KEY_GUID('MySymmetricKey'), CAST(ExpirationDate AS NVARCHAR(MAX)));
33
34 CLOSE SYMMETRIC KEY MySymmetricKey;
35
36 -- 6. Insert new rows with encrypted ExpirationDate values
37 OPEN SYMMETRIC KEY MySymmetricKey
38 DECRYPTION BY CERTIFICATE MyCertificate;
39
40 CLOSE SYMMETRIC KEY MySymmetricKey;
41
42 -- 7. Retrieve and decrypt data for validation
43 OPEN SYMMETRIC KEY MySymmetricKey
44 DECRYPTION BY CERTIFICATE MyCertificate;
45
46 SELECT
47 |  VerificationCodeID,
48 |  ExpirationDate,
49 |  EncryptedCode,
50 |  CONVERT(NVARCHAR(MAX), DECRYPTBYKEY(EncryptedCode)) AS DecryptedExpirationDate
51 FROM VerificationCode;
52
53
54 CLOSE SYMMETRIC KEY MySymmetricKey;
55
56
57
58 SELECT
59 |  VerificationCodeID,
60 |  ExpirationDate,
61 |  EncryptedCode,
```

USER INTERFACE AND APPLICATION DEMO

Urban Logistics - Dashboard

Welcome to Urban Smart Logistics

Dashboard

- Customers
- Orders
- Prescriptions
- Logistic Providers
- Manage Delivery Partner

Customer Panel

ADD CUSTOMER

ID	Name	Email	Actions
1	John Doe	john.doe@example.com	MODIFY DELETE
2	Jane Smith	jane.smith@example.com	MODIFY DELETE
3	Sam Brown	sam.brown@example.com	MODIFY DELETE
4	Lisa Black	lisa.black@example.com	MODIFY DELETE
5	Alice Green	alice.green@example.com	MODIFY DELETE
6	Tom White	tom.white@example.com	MODIFY DELETE
7	Emma Blue	emma.blue@example.com	MODIFY DELETE
8	Chris Red	chris.red@example.com	MODIFY DELETE

Orders Panel

ADD ORDER

Customer: John

Order Date: mm/dd/yyyy

December 2024

Order ID	Date	Status	Actions
1	2024-11-08	Pending	MODIFY DELETE
2	2024-11-08	Shipped	MODIFY DELETE
3	2024-11-08	Delivered	MODIFY DELETE
4	2024-11-08	Cancelled	MODIFY DELETE
5	2024-11-08	Pending	MODIFY DELETE
6	2024-11-08	Shipped	MODIFY DELETE
7	2024-11-08	Delivered	MODIFY DELETE
8	2024-11-08	Cancelled	MODIFY DELETE
9	2024-11-08	Pending	MODIFY DELETE

Logistic Providers

ADD LOGISTIC PROVIDER

ID	Name	Contact	Address	Zip Code	Service Area	Actions
1	FastShip	123 Express Ln, Boston, MA	02118	Greater Boston Area	<button>EDIT</button> <button>DELETE</button>	
2	QuickDeliver	456 Rapid Rd, Cambridge, MA	02139	Cambridge Area	<button>EDIT</button> <button>DELETE</button>	
3	OnTime	3456789012	789 Swift St, Somerville, MA	02145	Somerville Area	<button>EDIT</button> <button>DELETE</button>
4	FastTrack	4567890123	101 Speed Way, Medford, MA	02155	Medford Area	<button>EDIT</button> <button>DELETE</button>
5	PrimeExpress	5678901234	202 Turbo Ave, Quincy, MA	02170	Quincy Area	<button>EDIT</button> <button>DELETE</button>
6	ShipNow	6789012345	303 Rush St, Brookline, MA	02445	Brookline Area	<button>EDIT</button> <button>DELETE</button>
7	DirectCourier	7890123456	404 Pace Blvd, Newton, MA	02458	Newton Area	<button>EDIT</button> <button>DELETE</button>
8	FlyFast	8901234567	505 Quick St, Waltham, MA	02452	Waltham Area	<button>EDIT</button> <button>DELETE</button>
9	EzyCourier	9012345678	606 Accelerate Ave, Watertown, MA	02472	Watertown Area	<button>EDIT</button> <button>DELETE</button>

Add Customer

Customer

ADD CUSTOMER

Name*: John

Email*: john@gmail.com

Phone Number: 8574328888

Street Address: Beacon Hill

Zip Code: 02215

City: Boston

Preferred Locker Location:

Delivery Preference:

CANCEL ADD CUSTOMER

Manage Delivery Partners

ADD DRIVER

Delivery Driver

Add Driver

Name: QuickDeliver

Vehicle: XYZ789

ABC123

XYZ789

LMN456

QRS852

DEF963

UVW159

GHI258

JKL357

MNO456

PQR789

FlyFast

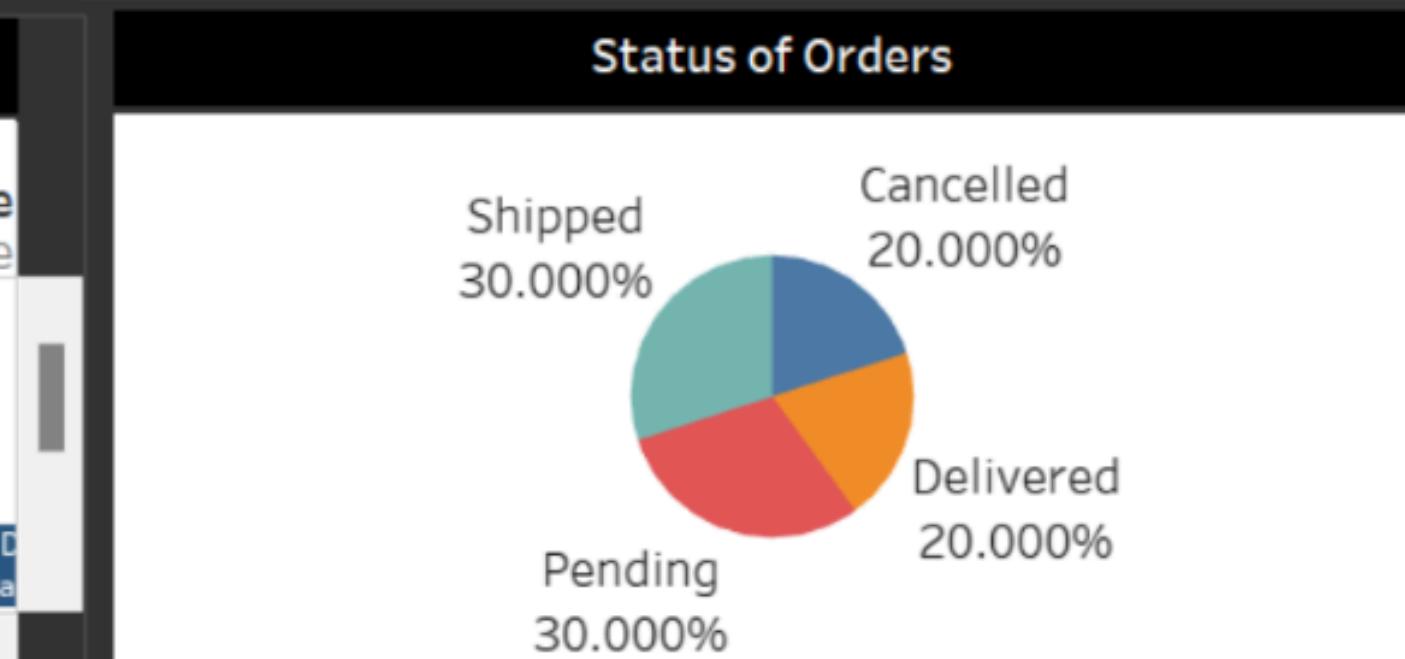
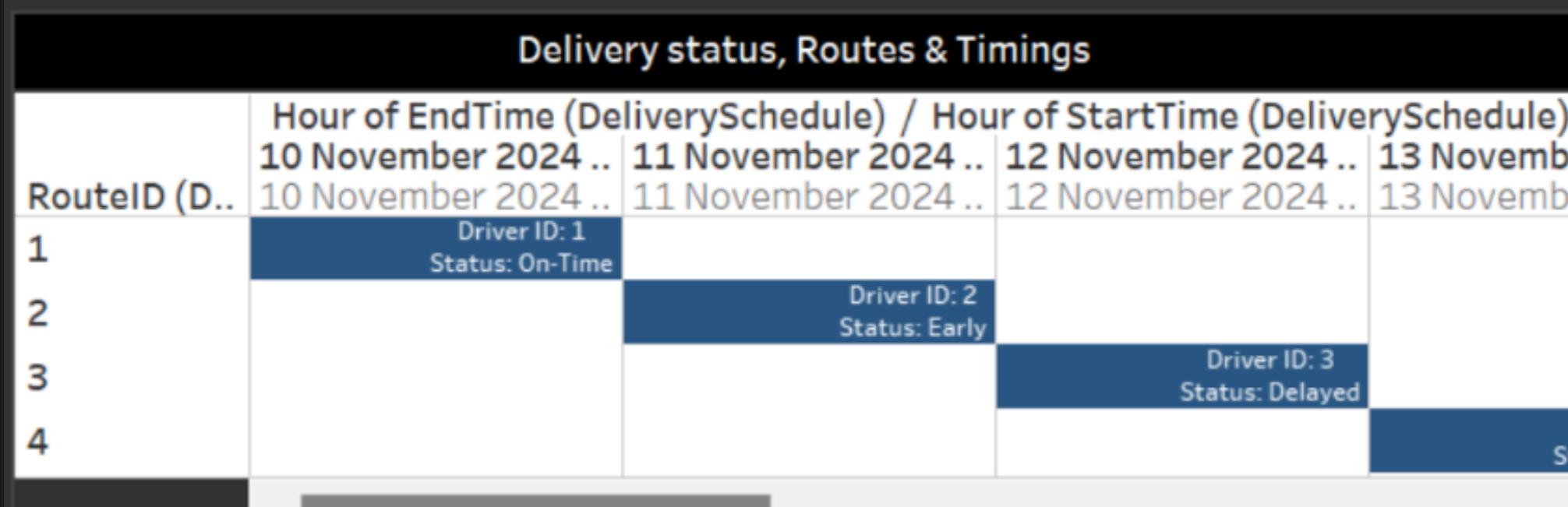
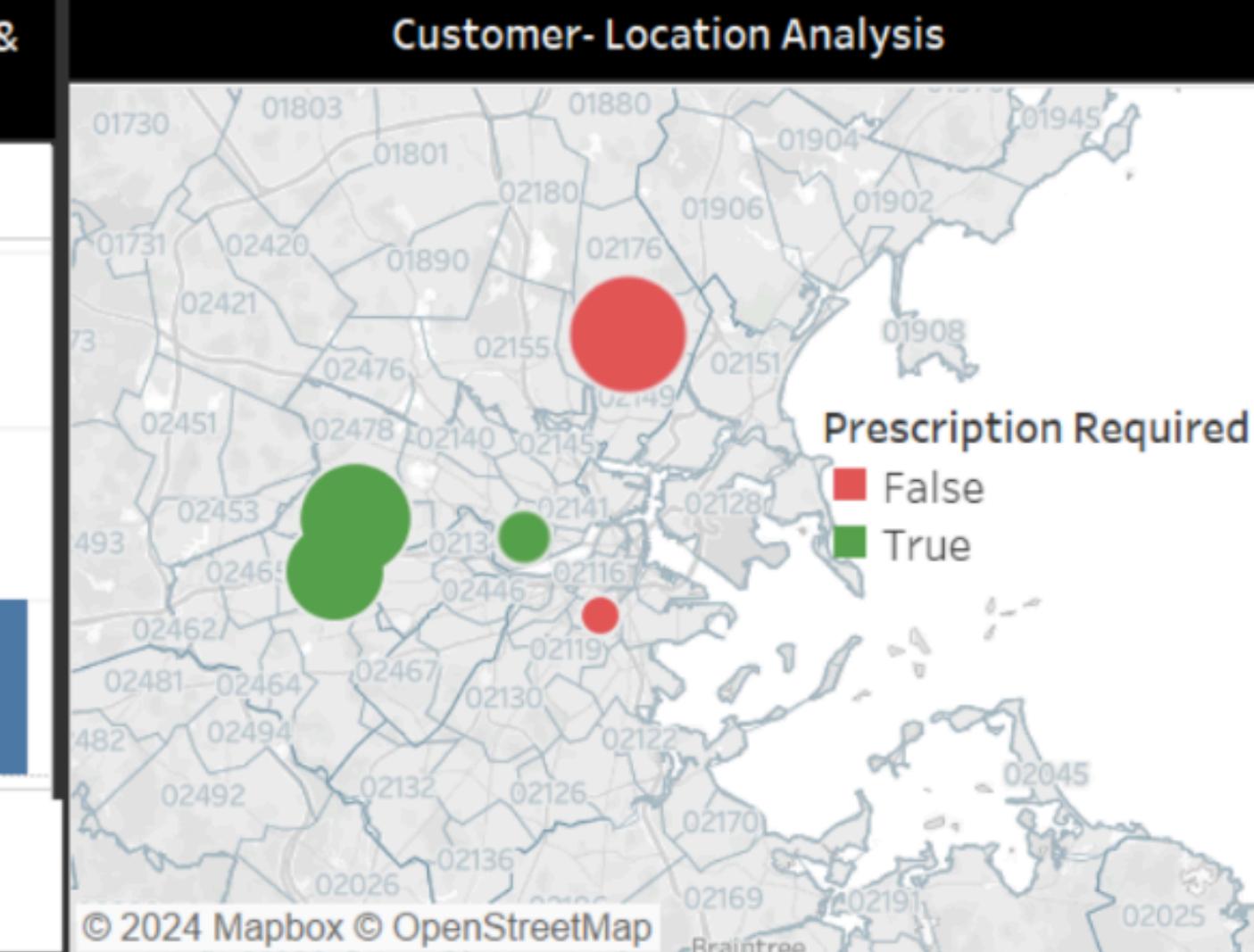
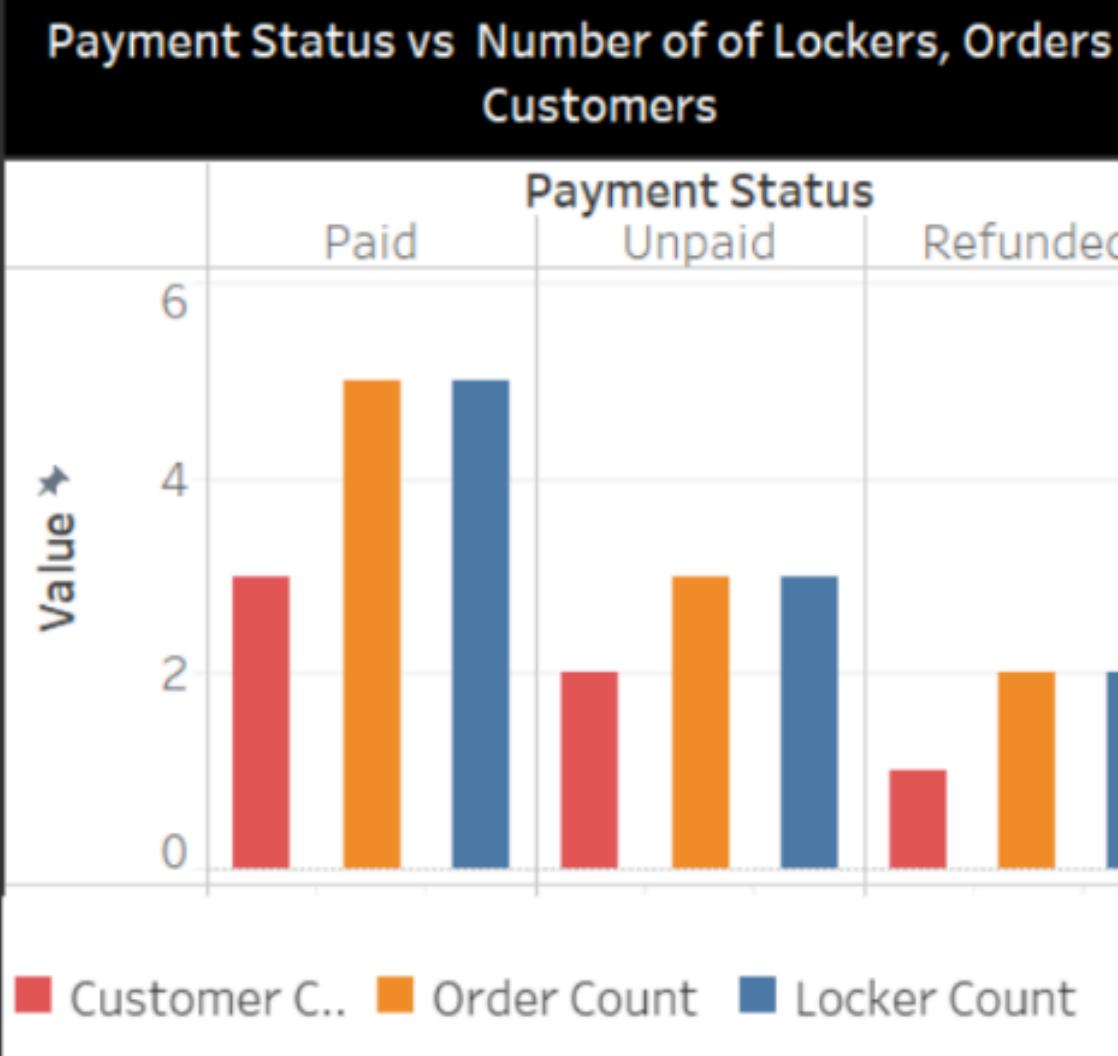
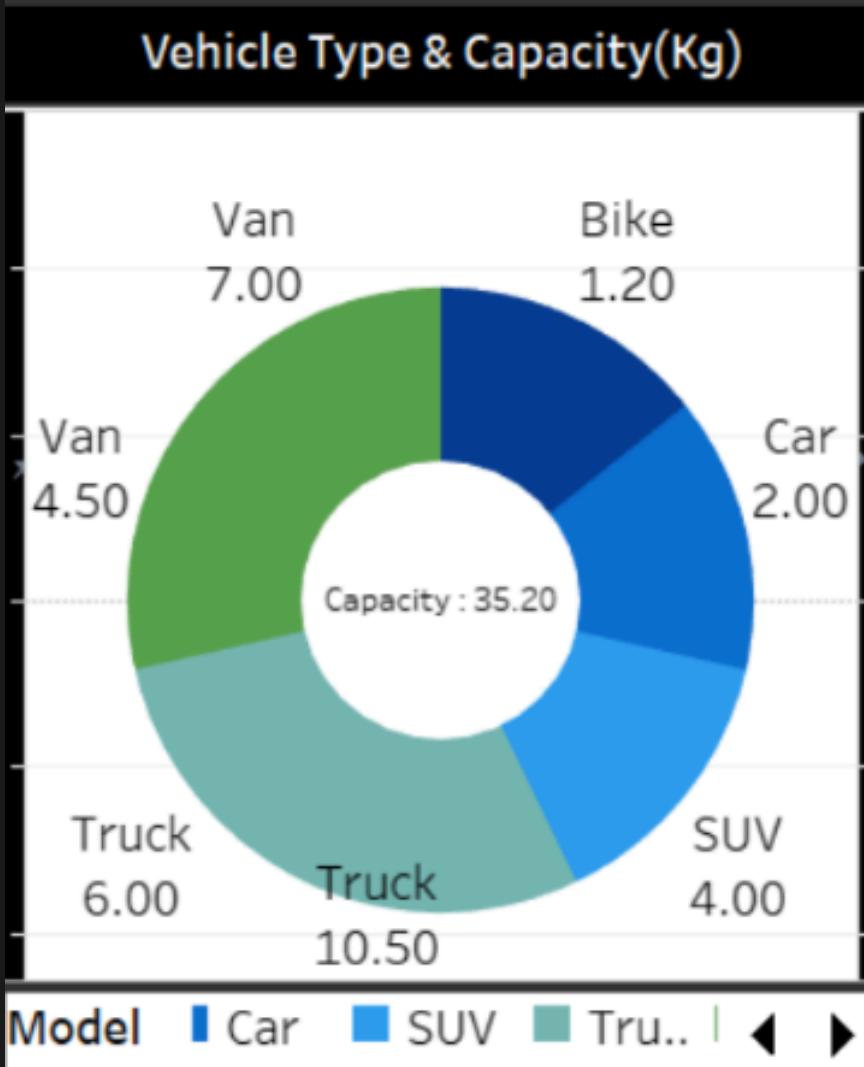
JKL357

LIC66676

Actions

VISUALIZATION

Urban Logistics System



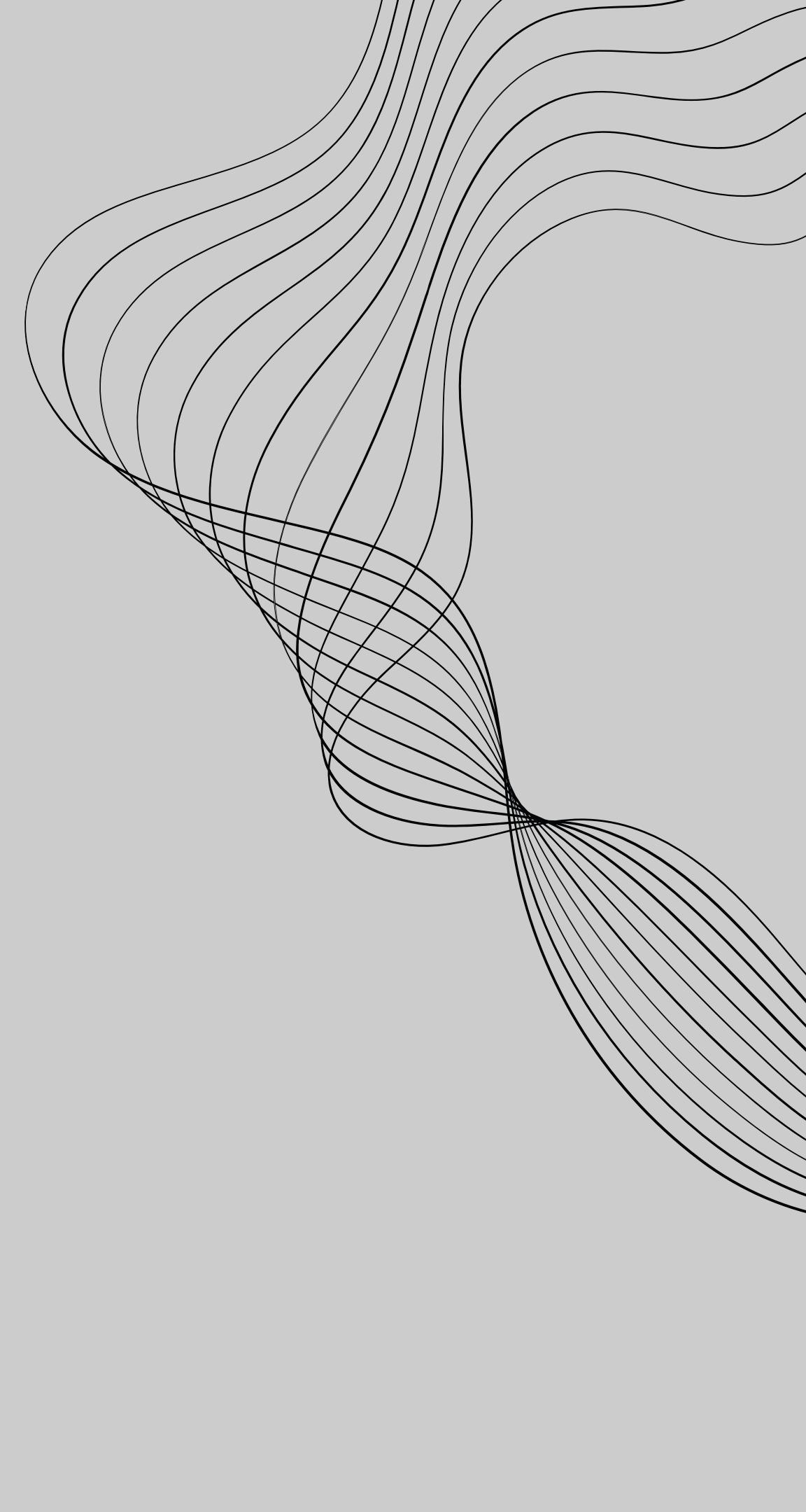
CONCLUSION

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*

