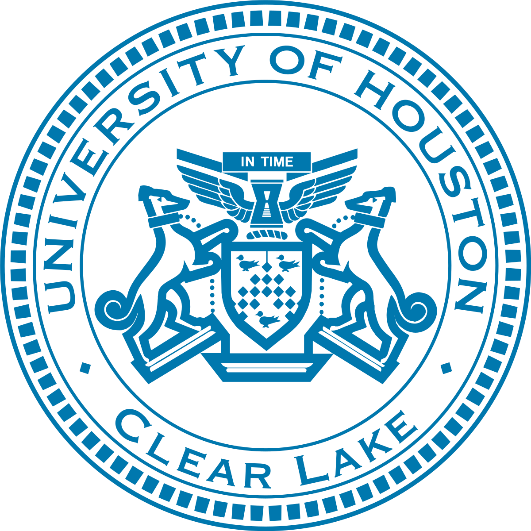
Primary University Logos | Downloads | Marketing and Communications |  University of Houston-Clear Lake

**CSCI 5333: Database Management Systems (DBMS)**

**(Fall 2023)**



**Project**

**Integrated Delivery System**

**Final Report**

***Group Name:*** TechShip Innovators

***Group Members:*** Balaji Bojadla (2286913),

Pranavi Kolagani (2146799)

Contents

[**Introduction** 3](#_Toc152510580)

[Project Description 3](#_Toc152510581)

[Key Features 3](#_Toc152510582)

[Project Objectives 3](#_Toc152510583)

[Stakeholders 4](#_Toc152510584)

[Customers 4](#_Toc152510585)

[Expected Outcomes 5](#_Toc152510586)

[**System Design** 6](#_Toc152510587)

[UML Use Case Diagram 6](#_Toc152510588)

[Sequence Diagram 7](#_Toc152510589)

[Entity Relationship Designations 8](#_Toc152510590)

[Relationship-set designation 9](#_Toc152510591)

[ER Diagram 10](#_Toc152510592)

[Empty Tables 11](#_Toc152510593)

[Relational Schema 12](#_Toc152510594)

[Schema Diagram 13](#_Toc152510595)

[**System Implementation** 14](#_Toc152510596)

[Introduction 14](#_Toc152510597)

[Domains 15](#_Toc152510598)

[Tables 20](#_Toc152510599)

[Insert Sample Data 28](#_Toc152510600)

[Triggers 36](#_Toc152510601)

[Assertions 40](#_Toc152510602)

[**Database Security** 41](#_Toc152510603)

[Create Roles 41](#_Toc152510604)

[Create Users 43](#_Toc152510605)

[Assigning Permissions 45](#_Toc152510606)

[**Connecting PostgreSQL Database to Backend** 47](#_Toc152510607)

[**User Interface** 48](#_Toc152510608)

[**Conclusion** 51](#_Toc152510609)

[**References** 51](#_Toc152510610)

[**Workload distribution** 52](#_Toc152510611)

# **Introduction**

Project Description

The development of the “Integrated Delivery System” project is designed to create an all-inclusive and effective delivery system that provides both individual and business customers with a broad range of features and services. The purpose of this system is to facilitate the process of shipping and receiving, in a simplier, reliable and convenient way for the users.

Key Features

An Integrated Delivery System offers a wide range of services to meet the diverse delivery needs of its customers. Here's what these services can include:

**User Authentication:** The provision of this feature allows users to create secure login accounts and ensure that the system is accessed by authorized individuals only. It is a platform that enables personalized service and safe interactions.

**Shipping Label Generation**: Simply, it lets the user type the key package elements to generate the shipping label with much ease. This information will normally comprise of the sender and the recipient, dimensions, and weight of the package. This eases the loading process and minimizes mistakes.

**Drop-off Points:** A package submission option is available within the project, where users can select a nearby drop-off point, e.g., UPS store. This improves customer convenience by offering alternative delivery points.

**Package Tracking:** Real-time package tracking ensures that the users can see the stage at which the package is and where it is in the process of delivery. This attribute guarantees clarity and peace of mind.

**View Package History:** Their shipping history is available to users, and they can see shipments they had made and track previous package deliveries. It also enables users to keep an organized record of packages and simplifies the package management.

**End-of-Day Operations:** At drop-off points, admins can effectively carry out end-of-day operations of processing and shipping the packages that have accumulated during the day. One of the key features is that packages should be prepared in advance for shipment.

## Project Objectives

The primary objectives of this project are as follows:

* To build a user-friendly and secure platform that enables customers to handle their shipments.
* Improve shipping by offering label generation and real-time tracking of packages.
* Enhancing the efficiency of drop-off points to make it simpler to place packs.
* Transparency and a shipment history for the users to manage their packages.
* Efficient operation of drop-off points to facilitate processing of packages and subsequent delivery.

Stakeholders In an Integrated Delivery System (IDS), stakeholders are the people concerned on how the system works and succeeds in achieving this goal. Let's take a closer look at who these stakeholders are:

**Customers:** These are the people who use the IDS services to send or receive packages, which can either be individuals shipping personal items or companies sending important stuffs.  
  
**Service Providers:** The companies providing courier services, tech solutions, and infrastructure for smooth deliveries offer the basic needs of the IDS.

**Regulatory Authorities:** The government bodies that ensure IDS follows all the safety, environment, and privacy-related rules and regulations.

**Investors and Shareholders:** These include individuals and groups who provide the capital for the business operating IDS, owning shares in these businesses.

**Employees:** These are the people who ensure the smooth running of the IDS. They include the drivers, warehouse workers, and the techie gurus.

**Suppliers:** The companies that supply these tools and resources that make the IDS work.  
  
**Competitors**: The other businesses in the delivery business that compete with the IDS for customers and market share.

Customers  
 Therefore, IDS depends on customers as heartbeats. These are the customers and merchants who rely on the system to deliver or to collect their goods. Here's a closer look at the different types of customers:

**Individual Consumers:** These are the common people who use the IDS to send their packages to their recipients which could be online purchases or presents.

**Businesses:** Movement of raw materials and finished goods is essential to any business, no matter its size, and the IDS plays a crucial role in this. To deliver on time and meet customers’ requirement, their supply chain must flow freely.

**E-commerce Companies:** It is essential for this IDS to ensure the delivery of those products to customers who order through online portals and retailers.

**Manufacturers and Suppliers:** These companies use the IDS to distribute their products to retailers and wholesalers. They also use it to move merchandise within their supply chain.  
  
**Retailers:** A large part of the stores’ replenishment and delivery is dependent on the IDS, particularly where online stores’ orders are involved.

**Healthcare and Pharmaceutical Industry:** IDS will therefore ensure that medical supplies, medications, and equipment are delivered to hospitals, clinics, and pharmacies on time.  
  
**Government Agencies:** IDS is used widely by government organisations for sending mails, public service distribution, and crucial supplies.

## Expected Outcomes

Upon successful implementation of the "Integrated Delivery System," the project anticipates the following outcomes:

* Improved convenience and openness to customer’s ships resulting in customer satisfactions.
* Fast processing of packages in the drop-off points leading to reduced time spent at the waiting stations.
* A user-friendly, organized history of shipments to be accessed easily.
* The real-time package tracking improves the delivery system’s trustworthiness.
* Better data and analytics to optimize the delivery process and look for more opportunities.

# **System Design**

## UML Use Case Diagram

A diagram of a network

Description automatically generated with medium confidence

## Sequence Diagram

A black screen with white text

Description automatically generated

## Entity Relationship Designations

1. **User Entity-Set:**
   * Entity: User
   * Attributes: Phone, Full Name, Email, , Phone Number, Address1, Address2, City, State, Zipcode.
2. **Package Entity-Set:**
   * Entity: Package
   * Attributes: Tracking Number, Weight, Dimensions, Status, Insurance, Description, Cost, Status, Created Timestamp, Estimated Delivery Date, Actual Delivery Date, Insurance, FromUserID (foreign key), ToUserID (foreign key), Store PointID (foreign key).
3. **Store Entity-Set:**
   * Entity: Store Point
   * Attributes: StoreID, Name, Location, Contact Information, Password
4. **Admin Entity-Set:**
   * Entity: Admin
   * Attributes: AdminID, Username, Password, First Name, Last Name, Email, Phone Number.
5. **Dropoff Entity-Set:**
   * Entity: Shipment
   * Attributes: StoreID, Tracking Number, Timestamp, Location ID.
6. **Payment Entity-Set:**
   * Entity: Payment
   * Attributes: PaymentID, Amount, Payment Date, Payment Status, Payment Receipt, UserID (foreign key).
7. **Location Entity-Set:**
   * Entity: Location
   * Attributes: LocationID, Name, Latitude, Longitude.
8. **DeliveryRoute Entity-Set:**
   * Entity: DeliveryRoute
   * Attributes: RouteID, Route Name, Estimated Time of Arrival (ETA), Actual Time of Arrival, Route Description.
9. **Payment Method Entity-Set:**

* Entity: PaymentMethod
* Attributes: MethodID (Primary Key), Method Name, Description.
* Description: Represents various payment methods such as credit card, PayPal, etc. This is a weak entity set that depends on the "Payment" entity set for its existence.

## Relationship-set designation

1. **User-to-Package Relationship:**
   * Relationship Set Name: UserPackages
   * Description: This relationship set represents the association between users and the packages they have sent or are expecting to receive. It captures the fact that a user can be associated with multiple packages.
2. **User-to-Payment Relationship:**
   * Relationship Set Name: UserPayments
   * Description: This relationship set reflects the association between users and their payment transactions. Users can have multiple payment records.
3. **Package-to-Store Relationship:**
   * Relationship Set Name: PackageStore
   * Description: This relationship set links packages to the drop-off points(Store) where they are submitted. Each package is associated with a single drop-off point, and a drop-off point can be linked to multiple packages.
4. **Location-to-DeliveryRoute Relationship:**
   * Relationship Set Name: LocationRoutes
   * Description: This relationship set establishes the association between locations (e.g., drop-off points) and the delivery routes that include these locations. Locations can be part of multiple delivery routes, and each route includes multiple locations.

## ER Diagram

A diagram of a diagram

Description automatically generated with medium confidence

Assumptions:

* Each user's phone number is unique within the system.
* Payment records include payment receipts.
* Assumption that every package and payment record is associated with a user, and each user can have multiple packages and payments. Users are mandatory participants in these relationships.
* Assumption that "Payment Status" can have predefined values such as "approved," "pending," and "declined." This simplifies tracking payment status.
* Assumption that the "Payment Method" entity-set contains predefined payment methods such as credit card, PayPal, etc. This simplifies the system by providing a predefined list of options.
* Assumption that tracking numbers for packages are unique system-wide. This ensures that users can accurately track their packages without ambiguity.
* Assumption that locations (e.g., drop-off) must participate in at least one delivery route, and each delivery route includes at least one location.
* Package tracking includes both estimated and actual delivery dates.
* Aggregated delivery routes are used for optimizing shipment logistics.
* Payment methods only exist in association with a payment record.
* The cardinality of relationships has been established based on the specific business logic and requirements.

## Empty Tables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Admin | | | | | | |
| AdminID | FirstName | LastName | Username | Password | Email | StoreID |
|  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Store | | | | |
| StoreID | Name | Location | Contact | Password |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Dropoff | | | |
| Tracking Number | StoreID | LocationID | Timestamp |
|  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Package | | | | | | | | | | |
| Tracking Number | | From\_UserID | | To\_UserID | Dimensions | | Weight | | Insurance | storeID |
| Description | EDD | ADD | Cost | Type | | Status | | Created Time Stamp | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Location | | | |
| LocationID | Name | Latitude | Longitude |
|  |  |  |  |

|  |  |
| --- | --- |
| LocationRoutes | |
| LocationID | RouteID |
|  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| User | | | | | | | |
| Phone | FullName | Address1 | Address2 | Email | City | State | Zipcode |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Delivery Route | | | | |
| RouteID | RouteName | ETA | ATA | Desc |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Payment Method | | | |
| MethodID | PaymentID | MethodName | Description |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Payment | | | | | |
| PaymentID | Date | Time | Reciept | Status | UserID |
|  |  |  |  |  |  |

## Relational Schema

**Admin**(AdminID, FirstName, LastName, UserName, Password, Email, StoreID)

**Store**(StoreID, Name, Location, Contact, Password)

**Dropoff**(Tracking Number, Storeid, LocationID, Timestamp)

**Package** (Tracking Number, From\_UserID, To\_UserID, Dimensions, Weight, Insurance, storeID

Description , EDD, ADD, Cost, Type, Status, Created Time Stamp)

**Location**(LocationID, Name, Latitude, Longitude)

**LocationRoutes**(LocationID, RouteID)

**User**(Phone, FullName, Address1, Address2,Emial, City, State,Zipcode)

**DeliveryRoute**(RouteID, RouteName, ETA, ATA, Desc)

**PaymentMethod**(MethodID, PaymentID, MethodName, Desc)

**Payment**(PaymentID, Date, Time, Receipt, Status, UserID)

## Schema Diagram

A diagram of a database

Description automatically generated

# **System Implementation**

# Introduction

In the pursuit of creating a robust and efficient database system for the Integrated Delivery System project, **PostgreSQL** emerged as the cornerstone of our implementation. PostgreSQL, renowned for its open-source nature, extensibility, and robust feature set, provided the ideal platform to structure and manage the intricate web of data integral to our project. Leveraging the power of PostgreSQL, we aimed to build a database that seamlessly integrates user registration, package management, and delivery tracking functionalities. This report delves into the details of our database implementation, elucidating the design choices, schema structures, and the utilization of PostgreSQL's features to realize a scalable and secure solution for the complex requirements of an integrated delivery system.

Database Server: **PostgreSQL**

Frontend Technology: **Vue.js**

Backend Technology: **Springboot**

Server: (Embedded) **Tomcat Server**

Database Name: **ids**

# Domains

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN Name AS VARCHAR(255)

CHECK (VALUE ~ '^[A-Za-z]+([A-Za-z ]+)\*$');

Output:

CREATE DOMAIN

Query returned successfully in 51 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN phone\_number AS VARCHAR(10)

CHECK (VALUE ~ '^[0-9]{10}$');

Output:

CREATE DOMAIN

Query returned successfully in 56 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN email AS VARCHAR(255)

CHECK (VALUE ~\* '^[A-Za-z0-9.\_%-]+

@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}$');

Output:

CREATE DOMAIN

Query returned successfully in 43 msec.

A screenshot of a computer code

Description automatically generated

SQL:

CREATE DOMAIN AdminID AS VARCHAR(7)

CHECK (VALUE ~ '^AD[0-9]{5}$');

Output:

CREATE DOMAIN

Query returned successfully in 48 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN PointID AS VARCHAR(7)

CHECK (VALUE ~ '^DP[0-9]{5}$');

Output:

CREATE DOMAIN

Query returned successfully in 55 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN PackageStatus AS VARCHAR(20)

CHECK (VALUE IN ('At Dropoff location',

'In Transit', 'Arriving Late', 'Delivered'));

Output:

CREATE DOMAIN

Query returned successfully in 51 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN LocID AS VARCHAR(10)

CHECK (VALUE ~ '^LOC[0-9]{7}$');

Output:

CREATE DOMAIN

Query returned successfully in 82 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN Tracking AS VARCHAR(18)

CHECK (VALUE ~ '^1Z[0-9A-Z]{16}$');

Output:

CREATE DOMAIN

Query returned successfully in 233 msec.

A screenshot of a computer

Description automatically generated

SQL:

CREATE DOMAIN PaymentStatus AS VARCHAR(15)

CHECK (VALUE IN ('Paid','Pending'));

Output:

CREATE DOMAIN

Query returned successfully in 55 msec.

GUI output after creating domains:

A screenshot of a computer

Description automatically generated

# Tables

Dropoff Point Table: This table represent the Dropoff Points/ Store

A screenshot of a computer

Description automatically generated

SQL:

--Dropoff Store

CREATE TABLE DropoffPoint (

PointID pointid PRIMARY KEY,

Name name,

Address VARCHAR(255),

Contact phone\_number,

Capacity INT,

OperationHours VARCHAR(255)

);

Output:

CREATE TABLE

Query returned successfully in 64 msec.

StoreAdmin Table: This table represent the Dropoff Store owner or admin

A screenshot of a computer

Description automatically generated

SQL:

-- Dropoff Store Admin/ Owners

CREATE TABLE StoreAdmin (

AdminID adminid PRIMARY KEY,

FirstName name,

LastName name,

UserName VARCHAR(50) UNIQUE,

Password VARCHAR(50),

Email email UNIQUE,

PointID pointid,

FOREIGN KEY (PointID) REFERENCES DropoffPoint(PointID)

ON DELETE CASCADE ON UPDATE CASCADE

);

Output:

CREATE TABLE

Query returned successfully in 56 msec.

Location Table: This table represent the location Hubs for all the nearby Dropoff Points.

A screenshot of a computer

Description automatically generated

SQL:

--Hub Location Table

CREATE TABLE Location (

LocationID varchar(10) PRIMARY KEY,

Name name,

Latitude DECIMAL(9, 6),

Longitude DECIMAL(9, 6)

);

Output:

CREATE TABLE

Query returned successfully in 141 msec.

Shipment Table: A "Shipment" represents a collection or group of packages that are transported together as part of a larger batch.

A screenshot of a computer program

Description automatically generated

SQL:

-- Shipment Table

CREATE TABLE Shipment (

ShipmentID tracking PRIMARY KEY,

ShipmentDate DATE,

Status packageStatus,

TVolume DECIMAL(10, 2),

TPackCount INT,

TotalWeight DECIMAL(10, 2),

LocationID varchar(10),

FOREIGN KEY (LocationID) REFERENCES Location(LocationID)

ON UPDATE CASCADE

);

Output:

CREATE TABLE

Query returned successfully in 49 msec.

Customer Table: This table stores the user/ customer details.

A screenshot of a computer

Description automatically generated

SQL:

-- customer Table

CREATE TABLE customer (

UserID email PRIMARY KEY,

FirstName name,

LastName name,

Password VARCHAR(50),

Address VARCHAR(255),

RegistrationDate DATE

);

Output:

CREATE TABLE

Query returned successfully in 150 msec.

Customer\_phone: This table store the multiple phone numbers of customer.

A screenshot of a computer

Description automatically generated

SQL:

-- Customer\_Phone Table

CREATE TABLE customer\_Phone (

UserID email,

Phone phone\_number,

PRIMARY KEY (UserID, Phone),

FOREIGN KEY (UserID) REFERENCES Customer(UserID)

);

Output:

CREATE TABLE

Query returned successfully in 48 msec.

Package Table: This table store all the package information from past and present.

A screenshot of a computer

Description automatically generated

SQL:

-- Package Table

CREATE TABLE Package (

PackageID tracking PRIMARY KEY,

OriginAddr VARCHAR(255),

ActualDeliveryDate DATE,

EstiDeliveryDate DATE,

SpecialInstr VARCHAR(255),

Dimensions VARCHAR(20),

Weight DECIMAL(10, 2),

DestAdd VARCHAR(255),

Status packagestatus,

ShipmentID tracking,

PointID pointid,

UserID email,

FOREIGN KEY (ShipmentID) REFERENCES Shipment(ShipmentID),

FOREIGN KEY (PointID) REFERENCES DropoffPoint(PointID),

FOREIGN KEY (UserID) REFERENCES Customer(UserID)

);

Output:

CREATE TABLE

Query returned successfully in 60 msec.

Payment Table: This table store all the payment details

A screenshot of a computer program

Description automatically generated

SQL:

-- Payment Table

CREATE TABLE Payment (

PaymentID varchar(15) PRIMARY KEY,

Date DATE,

Time TIME,

PaymentMethod VARCHAR(10),

Status paymentStatus,

UserID email,

FOREIGN KEY (UserID) REFERENCES Customer(UserID)

ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT chk\_method CHECK (PaymentMethod IN ('Card', 'Cash'))

);

Output:

CREATE TABLE

Query returned successfully in 60 msec.

# Insert Sample Data

Dropoff Table:

A screenshot of a computer

Description automatically generated

SQL:

-- Inserting Sample Data for DropoffPoint Table

INSERT INTO DropoffPoint (PointID, Name, Address, Contact, Capacity, OperationHours)

VALUES

('DP77058', 'IDS Store Houston', '2347 Bay Area Blvd, Houston', '1234567890', 500, '09:00 AM - 06:00 PM'),

('DP77059', 'IDS Store Clear Lake', '2323 Claer Lake City Blvd, Houston', '9876543210', 650, '10:00 AM - 07:00 PM'),

('DP77546', 'IDS Store Friendswood', '133 N Friendswood Dr, Friendswood', '5429713548', 550, '10:00 AM - 07:00 PM'),

('DP77566', 'IDS Store Lake Jackson', '117 Highway 332 W Ste J, Lake Jackson,', '7891435624', 500, '08:00 AM - 07:00 PM'),

('DP77598', 'IDS Store Webster', '563 W Bay Area Blvd, Webster', '5946276856', 750, '08:00 AM - 07:00 PM');

Output:

INSERT 0 5

Query returned successfully in 132 msec.

Store Admin Table:

A screenshot of a computer

Description automatically generated

SQL:

-- Inserting Sample Data for Store Admin Table

INSERT INTO StoreAdmin (adminid, FirstName, LastName, UserName, Password, Email, PointID)

VALUES

('AD00001', 'Virat', 'Kohli', 'viratekohli', 'adminpass', 'virat@example.com', 'DP77058'),

('AD00002', 'MS', 'Dhoni', 'msdhoni', 'adminpass', 'dhoni@example.com', 'DP77059'),

('AD00003', 'Steve', 'Smith', 'stevesmith', 'adminpass', 'steve@example.com', 'DP77546'),

('AD00004', 'Tony', 'Stark', 'tonystark', 'adminpass', 'tony@example.com', 'DP77566'),

('AD00005', 'Rohit', 'Sharma', 'rohitsharma', 'adminpass', 'rohit@example.com', 'DP77598');

Output:

INSERT 0 5

Query returned successfully in 59 msec.

Location Table:

A screenshot of a computer

Description automatically generated

SQL:

-- Inserting Sample Data for Location Table

INSERT INTO Location (LocationID, Name, Latitude, Longitude)

VALUES ('LOC0000001','LeagueCity', 29.492942, -95.102616),

('LOC0000002','LeagueCity', 29.539185, -95.062989);

Output:

INSERT 0 2

Query returned successfully in 177 msec.

Shipment Table:

A screenshot of a computer

Description automatically generated

SQL:

INSERT INTO Shipment (Shipmentid, ShipmentDate, Status, TVolume, TPackCount, TotalWeight, LocationID)

VALUES

('1Z600B456782495316', '2023-01-15', 'Delivered', 6400, 45, 5050.2, 'LOC0000001'),

('1Z600B456782481656', '2023-01-16', 'Delivered', 6400, 20, 890.8, 'LOC0000002'),

('1Z600B456782459813', '2023-01-17', 'Delivered', 6400, 35, 1800.8, 'LOC0000001'),

('1Z600B456782974624', '2023-01-18', 'Delivered', 6400, 28, 1080.8, 'LOC0000001'),

('1Z600B456782156462', '2023-01-19', 'Delivered', 6400, 36, 1980.8, 'LOC0000002');

Output:

INSERT 0 5

Query returned successfully in 49 msec.

Customer Table:

A computer screen with text on it

Description automatically generated

SQL:

INSERT INTO Customer

(Userid, FirstName, LastName, Password, Address, RegistrationDate)

VALUES

('bob@example.com', 'Bob', 'Johnson', 'userpass', '789 Elm St, CityA', '2023-01-05'),

('eva@example.com', 'Eva', 'Williams', 'userpass', '567 Pine St, CityB', '2023-02-10'),

('john@example.com', 'John', 'Ross', 'userpass', '577 Willems St, CityB', '2023-02-11'),

('vicky@example.com', 'Vicky', 'Roy', 'userpass', '563 Bay Area Blvd, CityB', '2023-02-12'),

('tarak@example.com', 'Tarak', 'Ram', 'userpass', '567 Loke St, CityB', '2023-02-13');

Output:

INSERT 0 5

Query returned successfully in 62 msec.

Customer\_phone table:

A screenshot of a computer

Description automatically generated

SQL:

INSERT INTO Customer\_Phone (UserID, Phone)

VALUES ('bob@example.com', '4576219435'),

('bob@example.com', '7594135468'),

('eva@example.com', '6472135916'),

('john@example.com', '7134859642'),

('vicky@example.com', '5465464865'),

('tarak@example.com', '4156465135');

Output:

INSERT 0 6

Query returned successfully in 51 msec.

Package Table:

A close-up of a computer screen

Description automatically generated

SQL:

-- Inserting Sample Data for Package Table

INSERT INTO Package

(PackageID, OriginAddr, ActualDeliveryDate, EstiDeliveryDate, SpecialInstr, Dimensions, Weight, DestAdd, status, ShipmentID, PointID, UserID)

VALUES

('1ZABC1234567890121', '123 Main St, CityA', null, '2023-01-20', 'Fragile', '12x8x6', 5.2, 'CityB','At Dropoff location', '1Z600B456782156462', 'DP77058', 'tarak@example.com'),

('1ZDEF2345678901232', '456 Oak St, CityB', null, '2023-02-25', 'Handle with care', '10x6x4', 3.5, 'CityC','In Transit', '1Z600B456782156462', 'DP77058', 'tarak@example.com'),

('1ZGHI3456789012343', '789 Pine St, CityC', null, '2023-03-10', 'This side up', '8x6x5', 2.7, 'CityD','At Dropoff location', '1Z600B456782156462', 'DP77058', 'vicky@example.com'),

('1ZJKL4567890123454', '890 Cedar St, CityD', null, '2023-04-02', 'Do not stack', '15x10x8', 7.8,'CityF', 'At Dropoff location', '1Z600B456782974624', 'DP77598', 'john@example.com'),

('1ZMNO5678901234565', '567 Elm St, CityE', null, '2023-04-15', 'Perishable', '6x4x3', 1.5, 'CityF','Arriving Late', '1Z600B456782156462', 'DP77058', 'vicky@example.com'),

('1ZPQR6789012345676', '234 Birch St, CityF', null, '2023-05-05', 'Handle with care', '12x8x6', 4.2, 'CityG','Arriving Late', '1Z600B456782974624', 'DP77598', 'bob@example.com'),

('1ZSTU7890123456787', '890 Maple St, CityG', null, '2023-05-20', 'Fragile', '10x6x4', 3.0, 'CityH','In Transit', '1Z600B456782459813', 'DP77566', 'bob@example.com'),

('1ZVWX8901234567895', '456 Walnut St, CityH', null, '2023-06-02', 'Handle with care', '8x6x5', 2.5, 'CityI','In Transit', '1Z600B456782974624', 'DP77598', 'tarak@example.com'),

('1ZYZA9012345678903', '123 Pine St, CityI','2023-06-15', '2023-06-15', 'This side up', '15x10x8', 6.8, 'CityJ','Delivered', '1Z600B456782481656', 'DP77058', 'john@example.com');

Output:

INSERT 0 9

Query returned successfully in 50 msec.

Payment Table:

A screenshot of a computer screen

Description automatically generated

SQL:

INSERT INTO Payment (Date, Time, paymentid ,paymentmethod, Status, UserID)

VALUES

('2023-01-15', '14:30:00', 'REC123456','Card', 'Paid', 'bob@example.com'),

('2023-02-20', '10:45:00', 'REC789012','Card', 'Paid', 'john@example.com'),

('2023-03-05', '16:20:00', 'REC345678','Card', 'Pending', 'tarak@example.com'),

('2023-03-10', '12:15:00', 'REC901234','Cash', 'Paid', 'vicky@example.com'),

('2023-04-02', '09:30:00', 'REC567890','Card', 'Pending', 'tarak@example.com'),

('2023-04-15', '15:00:00', 'REC175656','Cash', 'Paid', 'john@example.com');

Output:

INSERT 0 6

Query returned successfully in 171 msec.

# Triggers

To enforce a unique shipment and package tracking number of a specific format using a trigger, you can create a trigger that fires before a new record is inserted into the Shipment table. This trigger will check if the tracking number adheres to the specified format and is unique across the system.

SQL:

-- Create a function to generate a unique tracking number

CREATE OR REPLACE FUNCTION generate\_unique\_shipment\_tracking\_number()

RETURNS TRIGGER AS

$$

DECLARE

new\_tracking\_num VARCHAR(18);

BEGIN

-- Generate a new tracking number until a unique one is found

LOOP

new\_tracking\_num := '1Z' || upper(md5(random()::text || clock\_timestamp()::text))::text;

-- Check if the generated tracking number already exists

EXIT WHEN NOT EXISTS (

SELECT 1 FROM Shipment WHERE ShipmentID = new\_tracking\_num

);

END LOOP;

-- Set the generated tracking number for the new shipment

NEW.ShipmentID := new\_tracking\_num;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

-- Create a trigger to invoke the function before inserting into the Shipment table

CREATE TRIGGER unique\_shipment\_tracking\_number\_trigger

BEFORE INSERT ON Shipment

FOR EACH ROW

EXECUTE FUNCTION generate\_unique\_shipment\_tracking\_number();

Output:

CREATE TRIGGER

Query returned successfully in 104 msec.

GUI output:

A screenshot of a computer

Description automatically generated

SQL:

-- Create a function to generate a unique tracking number for the PackageID

CREATE OR REPLACE FUNCTION generate\_unique\_package\_tracking\_number()

RETURNS TRIGGER AS

$$

DECLARE

new\_tracking\_num VARCHAR(18);

BEGIN

-- Generate a new tracking number until a unique one is found

LOOP

new\_tracking\_num := '1Z' || upper(md5(random()::text || clock\_timestamp()::text))::text;

-- Check if the generated tracking number already exists

EXIT WHEN NOT EXISTS (

SELECT 1 FROM Package WHERE PackageID = new\_tracking\_num

);

END LOOP;

-- Set the generated tracking number for the new package

NEW.PackageID := new\_tracking\_num;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

-- Create a trigger to invoke the function before inserting into the Package table

CREATE TRIGGER unique\_package\_tracking\_number\_trigger

BEFORE INSERT ON Package

FOR EACH ROW

EXECUTE FUNCTION generate\_unique\_package\_tracking\_number();

Output:

CREATE TRIGGER

Query returned successfully in 57 msec.

GUI Output:

A screenshot of a box

Description automatically generated

# Assertions

This assertion ensures that the weight of a package must be greater than or equal to 0 and less than or equal to 100, reflecting a valid weight range for packages in the system.

SQL:

-- Create an assertion to check the validity of the Weight attribute

CREATE ASSERTION valid\_package\_weight

CHECK (

NOT EXISTS (

SELECT 1

FROM Package

WHERE Weight IS NOT NULL AND (Weight <= 0 OR Weight > 100)

)

) DEFERRABLE INITIALLY IMMEDIATE;

# **Database Security**

In our Integrated Delivery System project, ensuring the security of our database is a top priority to protect sensitive information and regulate access effectively. We've introduced three distinct roles: 'systemAdmin,' 'storeAdmin,' and 'customer,' forming the backbone of our access control and privilege management strategy. The 'systemAdmin' role, exclusively for system administrators, comes with elevated privileges to oversee and manage the entire database system. For those overseeing store-related operations, the 'storeAdmin' role has been created. Lastly, the 'customer' role caters to our regular users, each enjoying tailored access levels that meet their specific needs. To bring these roles to life, we've established sample users with unique passwords, aligning each with their respective roles. This role-based access control not only enhances our security measures but also simplifies user management. Each role is carefully linked to specific permissions, ensuring precise control over data handling and guaranteeing that users access only the functionalities essential for their roles. This method not only reinforces the integrity of our database but also adheres to established database security best practices, resulting in a robust and well-organized system.

## Create Roles

A screenshot of a computer

Description automatically generated

SQL Queries:

-- Create System Admin Role

CREATE ROLE systemAdmin;

-- Create Store Admin Role

CREATE ROLE storeAdmin;

-- Create User Role

CREATE ROLE customer;

Output:

CREATE ROLE

Query returned successfully in 51 msec.

GUI output after creation of roles:

A screenshot of a computer program

Description automatically generated



## Create Users

A screenshot of a computer

Description automatically generated

SQL Queries:

-- Create Sample Admin User and Assign Admin Role

CREATE USER admin\_user WITH PASSWORD 'admin\_password';

GRANT systemAdmin TO admin\_user;

-- Create Sample Store Admin User and Assign Store Admin Role

CREATE USER storeadmin\_user WITH PASSWORD 'storeadmin\_password';

GRANT storeAdmin TO storeadmin\_user;

-- Create Sample User and Assign User Role

CREATE USER regular\_user WITH PASSWORD 'user\_password';

GRANT customer TO regular\_user;

Output:

GRANT ROLE

Query returned successfully in 132 msec.

GUI output after creating users:

A screenshot of a computer program

Description automatically generated



## Assigning Permissions

System Admin Role:

SQL:

-- Granting Permissions to systemAdmin Role

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO systemAdmin;

GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO systemAdmin;

Output:

GRANT

Query returned successfully in 57 msec.

A screenshot of a computer

Description automatically generated

Store Admin Role:

SQL:

-- Granting Permissions to storeAdmin Role

-- Granting permissions to all tables except DropoffPoint,StoreAdmin and Location

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO storeAdmin;

GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO storeAdmin;

REVOKE ALL PRIVILEGES ON TABLE DropoffPoint, location, storeadmin FROM storeAdmin;

Output:

REVOKE

Query returned successfully in 62 msec.

A screenshot of a computer code

Description automatically generated

Customer Role:

SQL:

-- Granting Permissions to user Role

GRANT INSERT, UPDATE, DELETE ON TABLE Customer TO Customer;

GRANT SELECT, USAGE ON ALL SEQUENCES IN SCHEMA public TO customer;

Output:

REVOKE

Query returned successfully in 62 msec.

A screenshot of a computer

Description automatically generated

**Authorization Graph**: Th below graph represent authorization for **SELECT**, **UPDATE**, **DELETE** on database tables as mentioned.

A diagram of a system

Description automatically generated

# **Connecting PostgreSQL Database to Backend**

1. **Add dependency for PostgreSQL - Spring Data JPA**

Declare the following dependency in your project’s pom.xml file:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <**dependency**>      <**groupId**>org.postgresql</**groupId**>      <**artifactId**>postgresql</**artifactId**>      <**scope**>runtime</**scope**>  </**dependency**>  <**dependency**>      <**groupId**>org.springframework.boot</**groupId**>      <**artifactId**>spring-boot-starter-data-jpa</**artifactId**>  </**dependency**> |

1. **Configure Data Source Properties**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | spring.datasource.url=jdbc:postgresql://localhost:5432/ids  spring.datasource.username=postgres  spring.datasource.password=Password@123  spring.jpa.hibernate.ddl-auto=update  spring.jpa.show-sql=true  spring.jpa.properties.hibernate.format\_sql=true  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL81Dialect |

Next, specify database connection information in the Spring Boot application configuration file (application.properties) as follows:

My PostgreSQL port number:

**A screenshot of a computer

Description automatically generated**

# **User Interface**

Login Page:

A screenshot of a login form

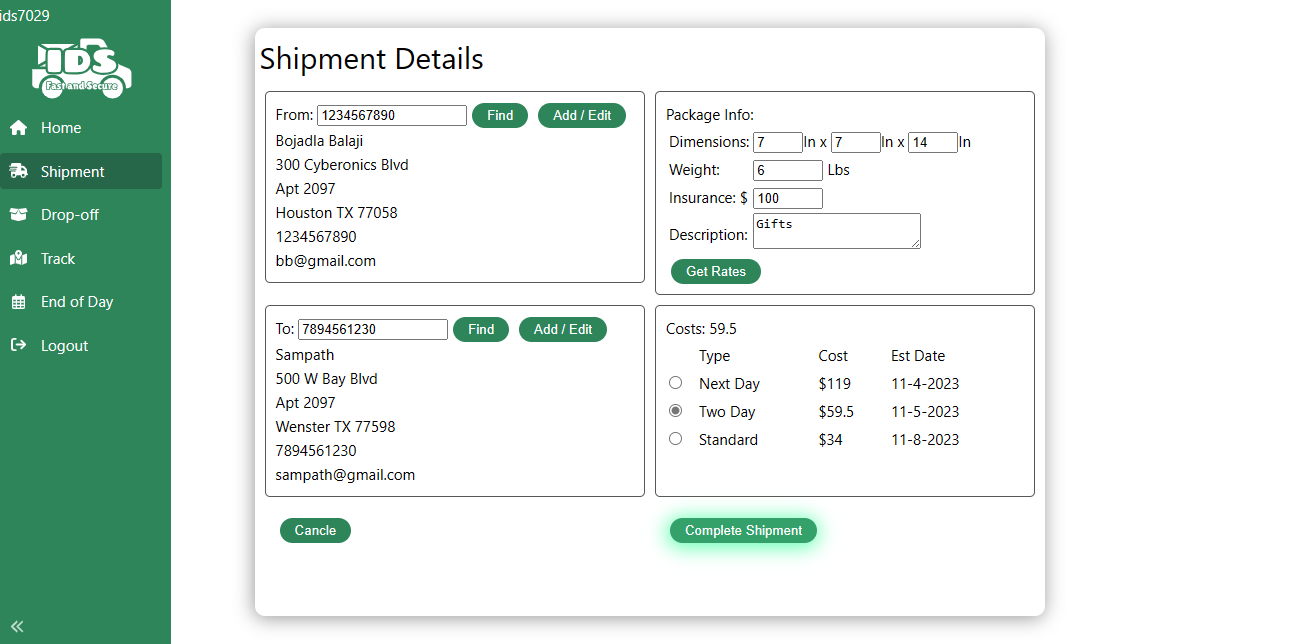
Description automatically generated

Shipment Page:

A screenshot of a computer

Description automatically generated

Shipment Page: (With Details Filled)



Dropoff Page:

A screenshot of a computer

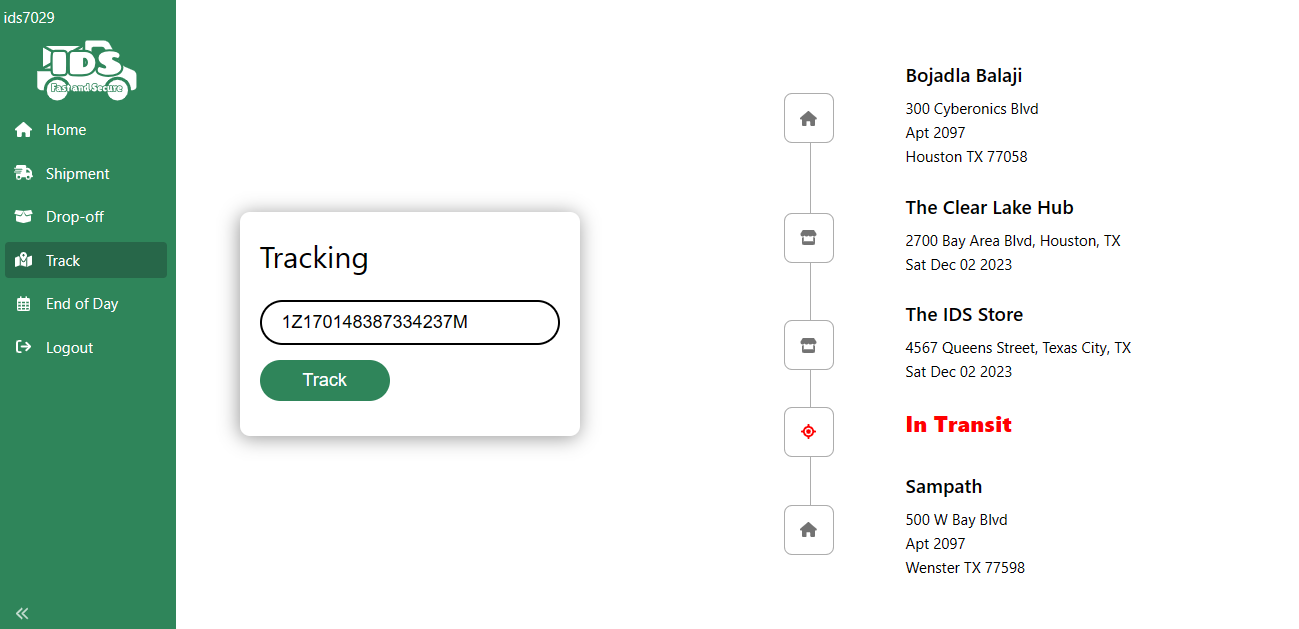
Description automatically generated

Tracking Page:

A screenshot of a tracking number

Description automatically generated

Tracking Page: (After Results)



# **Conclusion**

In summary, the development and execution of the Integrated Delivery System project have marked a significant milestone in the enhancement of the logistics and delivery industry. This project has effectively brought together user registration, package management, payment processing, and package tracking, resulting in a more streamlined delivery process for both end-users and administrators.

The system's strengths lie in its user-friendly interface, secure authentication methods, and real-time tracking capabilities. Together, these features offer a seamless and efficient delivery experience, meeting the dynamic demands of modern delivery services.

The introduction of a weak entity set and aggregation relationships in the ER diagram showcases a sophisticated understanding of data structure complexity and further enriches the system's adaptability and flexibility.

Notably, the project demonstrates a commitment to delivering top-quality solutions and holds promise for further innovations within the delivery industry. It serves as a testament to the effective use of database design and entity-relationship modeling in addressing practical challenges in logistics.

Looking ahead, our aspiration is for the Integrated Delivery System to evolve and respond to the ever-changing landscape of delivery services. We are confident that the principles and methodologies employed in this project will lay a solid foundation for future advancements in the field.

We extend our heartfelt appreciation to all contributors to this project, and we eagerly anticipate the positive impact it will make on the world of integrated delivery services.

# **References**

* The UPS System
* The FedEx Delivery System

# **Workload distribution**

Bojadla Balaji (2286913)

* Getting Started with the project proposal, documented proposal document which includes project description and scope of the project.
* Developed ER diagram, empty tables, and schema diagram to showcase the relationship between the tables.
* In system implementation-built Triggers, Assertions, DB Security & Connections.
* Connecting and Managing Front end and backend database.

Pranavi Kolagani (2146799)

* Getting Started with the project proposal, developed a UML Use Case Diagram to visually represent the interactions between different actors and system functionalities.
* Documented system design document includes key features, project objective, and expected outcome and created Sequence Diagrams with reference to UML Use Case Diagram.
* Built Domains, Tables, and Inserted sample data to the database during system implementation.
* Managing Database and documented Final report.