Courier Management System — Project Report

RISHITH SAHU - PES1UG23CS481
PUNURU SAI MANUJ REDDY - PES1UG23CS454
SECTION :H

Abstract

This project implements a Courier Management System — a web application that allows users to create courier shipments, make payments, track deliveries, and receive notifications. Admins can manage shipments and assign delivery agents, while agents can view assigned shipments and mark them delivered. The system uses Flask (Python) with SQLAlchemy for the backend and MySQL for persistent storage.

Objectives

- Build a functional web application for creating and tracking courier shipments.
- Provide role-based access (User, Admin, Agent).
- Implement payment flow with server-side and client-side validations.
- Provide notifications (email and SMS) with a DB-backed fallback.
- Ensure reliable tracking history and minimal duplication through DB procedures/triggers.

Key Features

- User registration and login.
- Create courier shipments (sender & receiver details, addresses, weight, type).
- Payment page with validation (credit/debit/UPI) and server-side checks.
- Admin dashboard: list couriers, assign agents, view payments.
- Agent dashboard: view assigned couriers, mark delivered.
- Courier tracking history stored in `Courier_tracking`.
- Notifications: email and SMS (configurable in DB); in-app persistence if external services unavailable.
- Database stored procedures, functions, views and triggers to centralize event handling.

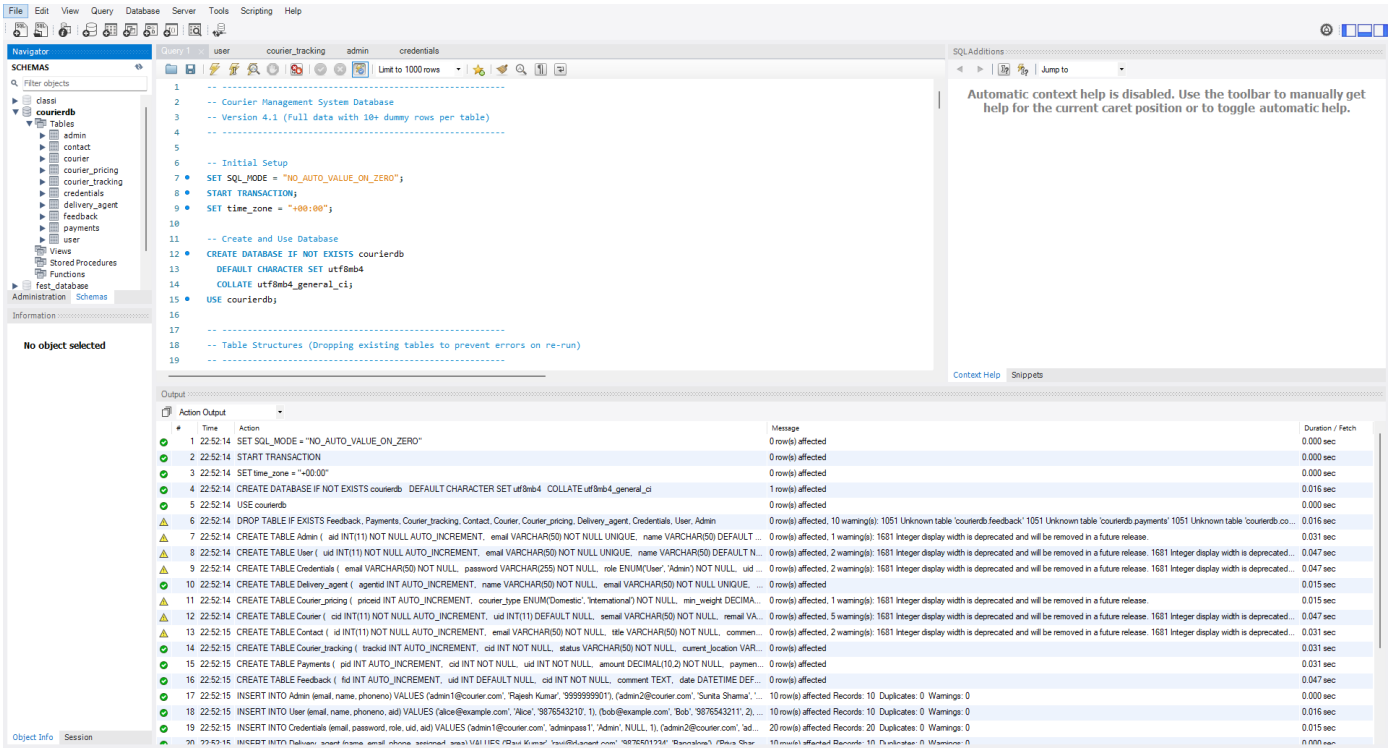
Technologies Used

- Python 3.11
- Flask (web framework)
- Jinja2 (templating)
- SQLAlchemy (ORM)
- MySQL (mysql-connector)
- Bootstrap 5 (UI)
- Optional: Twilio (SMS), SMTP for email

System Architecture (high-level)

- Client (browser): interacts with Flask server via HTML forms and JS validation.
- Server (Flask): handles routes, validation, session-based auth, calls stored procedures for key operations, sends notifications.
- Database (MySQL): stores domain data (Courier, Payments, Courier_tracking, Delivery_agent, Users) and contains stored procedures, functions, views, and triggers to ensure consistent tracking and enable reporting.

Database Design (summary)



Main tables:

- `Courier` — stores shipments (cid, sender/receiver details, addresses, billno, agentid, priceid, date)
- `Courier_tracking` — timeline of status updates for a courier (trackid, cid, status, current_location, updated_at)
- `Payments` — payment information (pid, cid, uid, amount, payment_mode, payment_status, transaction_date)
- `Delivery_agent` — delivery agents (agentid, name, email, phone)
- `Notification_config` — SMTP/Twilio settings (admin-editable)

Stored routines created:

- `sp_mark_payment_completed(p_cid)` — mark payments as Completed and insert `Payment Received` tracking if needed.
- `sp_assign_agent(p_cid, p_agentid)` — set `agentid` for courier and insert an assignment tracking entry if changed.

Stored functions:

- `fn_payment_status(p_cid)` — returns payment status.

• `fn_last_tracking_status(p_cid)` — returns latest tracking status.

Views:

• `vw_courier_summary` — summary per courier (billno, amount, payment_status, last_status, agentid).

• `vw_agent_assignments` — per-agent assigned-count summary.

Triggers:

• `trg_payments_after_insert` — AFTER INSERT on `Payments`, inserts `Payment Received` tracking when appropriate (deduplicates using last status and 120s window).

• `trg_courier_after_update_agent` — AFTER UPDATE on `Courier`, inserts assignment/unassignment tracking entries similar to above.

Notes: Triggers and procedures include deduplication logic to avoid duplicate `Courier_tracking` rows when both app and DB insert similar events.

Implementation Highlights

```
def show_create_object(obj_type, name):
    """Show CREATE statement for PROCEDURE/FUNCTION/VIEW/TRIGGER using SHOW CREATE ..."""
    with app.app_context():
        try:
            q = text(f"SHOW CREATE {obj_type} `{name}`")
            res = db.session.execute(q).fetchall()
            print(f'SHOW CREATE {obj_type} {name}:')
            for r in res:
                print(r)
        except Exception as e:
            print('Error showing create for', obj_type, name, e)
```

```
def call_procedure_mark_payment_completed(courier_id):
    """Call stored procedure sp_mark_payment_completed(:cid) using SQLAlchemy.
    Uses transactional execution via db.engine.begin().
    """
    with app.app_context():
        try:
            with db.engine.begin() as conn:
                conn.execute(text("CALL sp_mark_payment_completed(:cid)"), {"cid": courier_id})
            print('Called sp_mark_payment_completed for cid=', courier_id)
        except Exception as e:
            print('Error calling procedure:', e)
```

```
def demonstrate_trigger_payment(courier_id, user_id, amount=100.0):
    """Insert a Payments row with status Completed and then query Courier_tracking to show trigger effect.

    WARNING: This will insert data into your DB. Use with test/demo data.
    """
    with app.app_context():
        try:
            with db.engine.begin() as conn:
                # Insert a payment row that should fire the payments trigger
                conn.execute(text("INSERT INTO Payments (cid, uid, amount, payment_mode, payment_status, transaction_date) VALUES (:cid, :uid, :amt, 'Credit Card', 'Completed', NOW())"),
                             {"cid": courier_id, "uid": user_id, "amt": amount})
                print('Inserted Payments row (Completed) for cid=', courier_id)
                # read recent tracking rows
                rows = conn.execute(text("SELECT * FROM Courier_tracking WHERE cid = :cid ORDER BY updated_at DESC LIMIT 5"), {"cid": courier_id}).fetchall()
                print('Recent Courier_tracking rows:')
                for r in rows:
                    print(dict(r))
        except Exception as e:
            print('Error inserting payment/reading tracking:', e)
```

• `app.py` contains Flask routes for all pages (index, login/register, dashboard, create_courier, payment, agent/admin dashboards, assign, update status, notify helpers).

• A context processor `inject_now()` provides `now()` to templates for footer timestamps.

• Payment validation:

• Client-side: JS ensures digits-only card entry (16 digits), expiry format, CVV checks, UPI validation.

• Server-side: validates card digits (16), expiry against IST `ist_now()`, CVV length. Calls `sp_mark_payment_completed` to finalize payment.

• Notifications: `notify_parties()` constructs email/SMS messages using DB-backed configuration; `send_email()` and `send_sms()` handle external sends where configured. The DB stores notification settings via `admin_notification_config` route.

• Stored routines centralize key operations. The app was updated to call procedures for payment completion and agent assignment.

How to Run Locally (development)

1 . Ensure MySQL is running and a database `courierdb` exists. Adjust `app.config['SQLALCHEMY_DATABASE_URI']` in `app.py` if needed.

2 . Install Python dependencies (example):

```
pip install -r requirements.txt
```

(If no requirements file, install Flask, SQLAlchemy, mysql-connector-python, werkzeug)

3. Set environment variables for optional SMTP/Twilio if needed:

```
$env:SMTP_SERVER='smtp.example.com'; $env:SMTP_PORT='587'; $env:SMTP_USERNAME='user'; $env:SMTP_PASS
```

4. Run the app:

```
python app.py
```

5. Open http://127.0.0.1:5000 in a browser.

Testing & Verification

• Manual tests performed:

• Create courier → verify `Payments` row created (Pending) and `Courier_tracking` initial Pending entry.

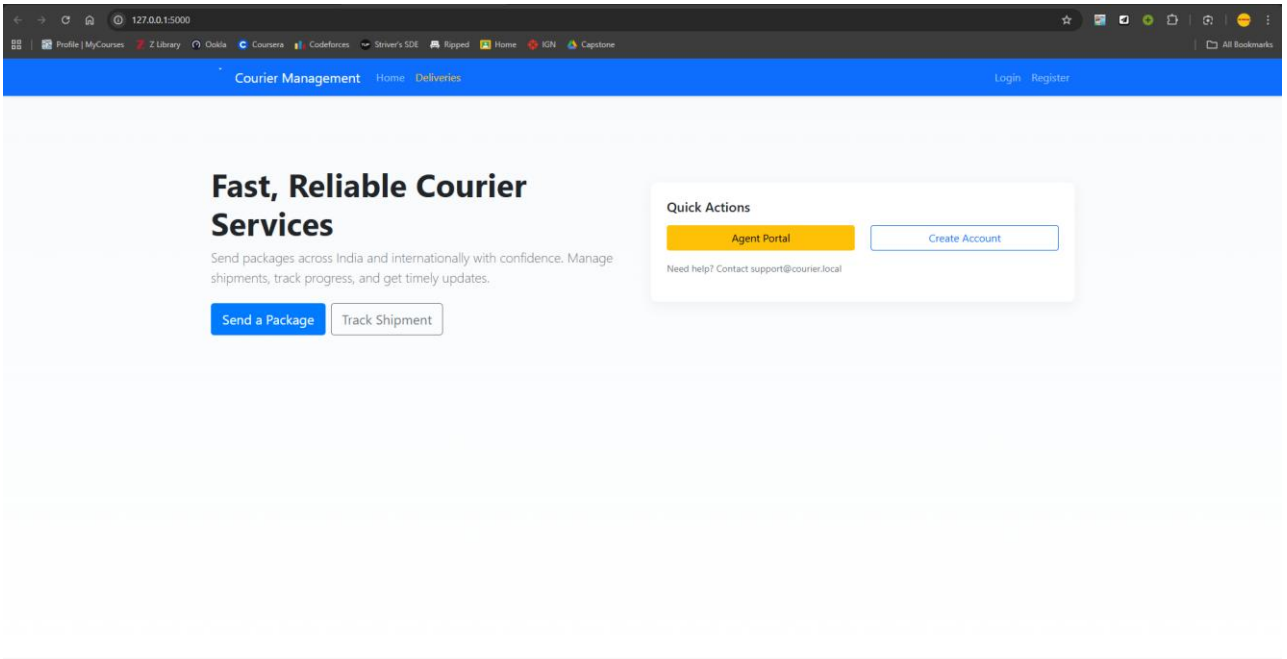
• Payment page validation (client + server) → call `sp_mark_payment_completed` and verify `Courier_tracking` has 'Payment Received'.

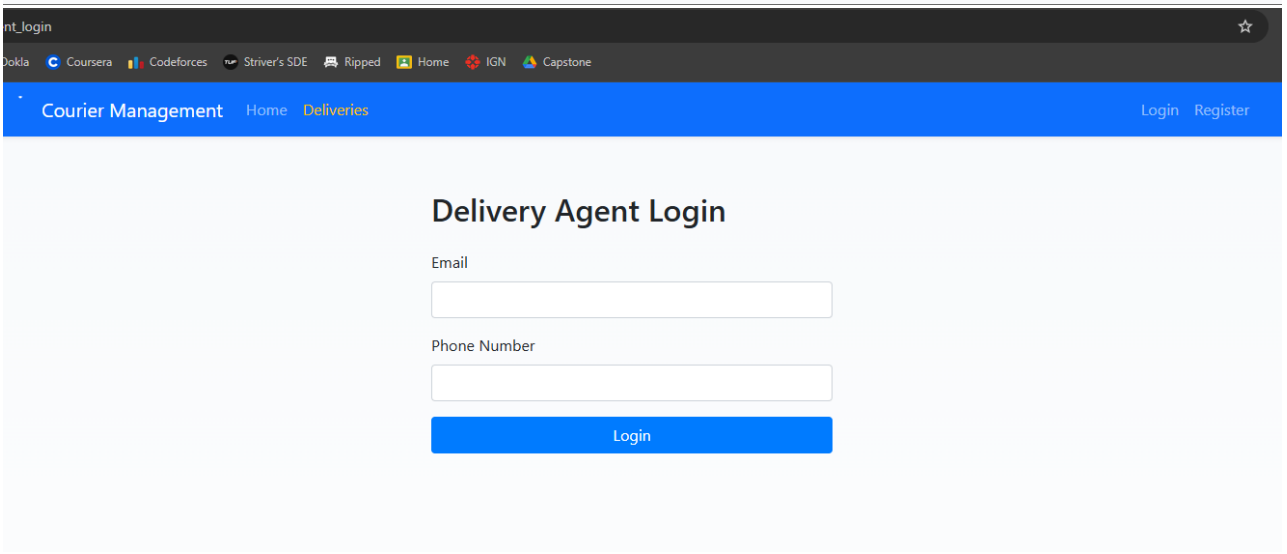
• Assign agent via admin dashboard → `sp_assign_agent` called, courier `agentid` updated and tracking entry added.

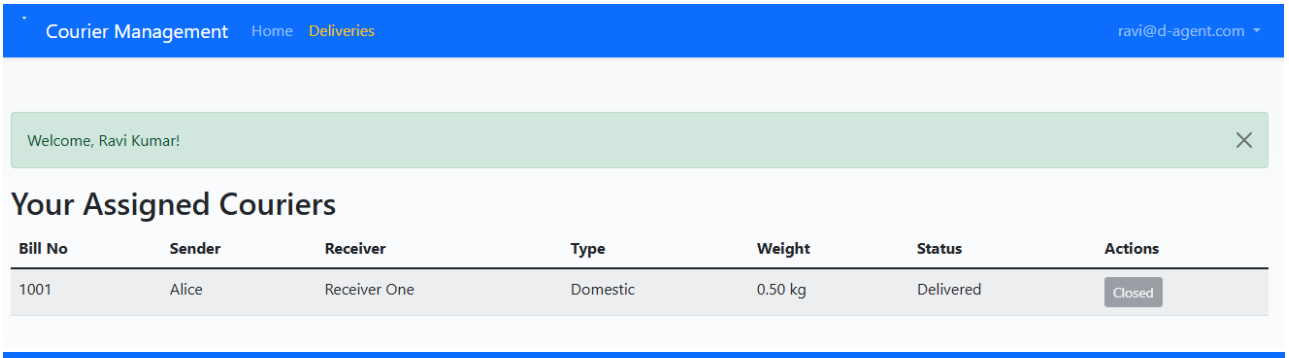
• Notifications: `admin_notify_test` and `notify_test/` endpoints to exercise email/SMS sending; if external not configured, messages are logged and persisted (in-app fallback).

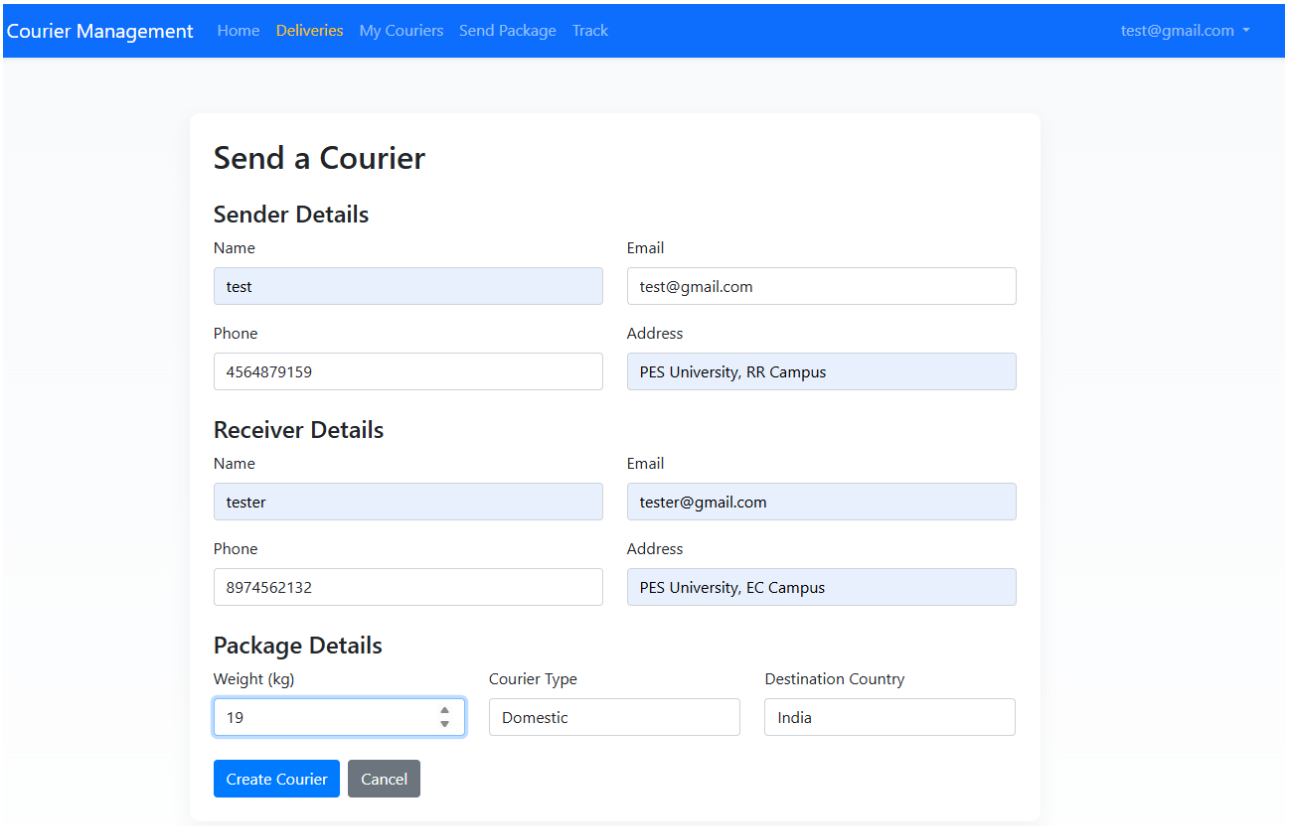
Github Link: https://github.com/RishithSahu/Courier-Management-Service

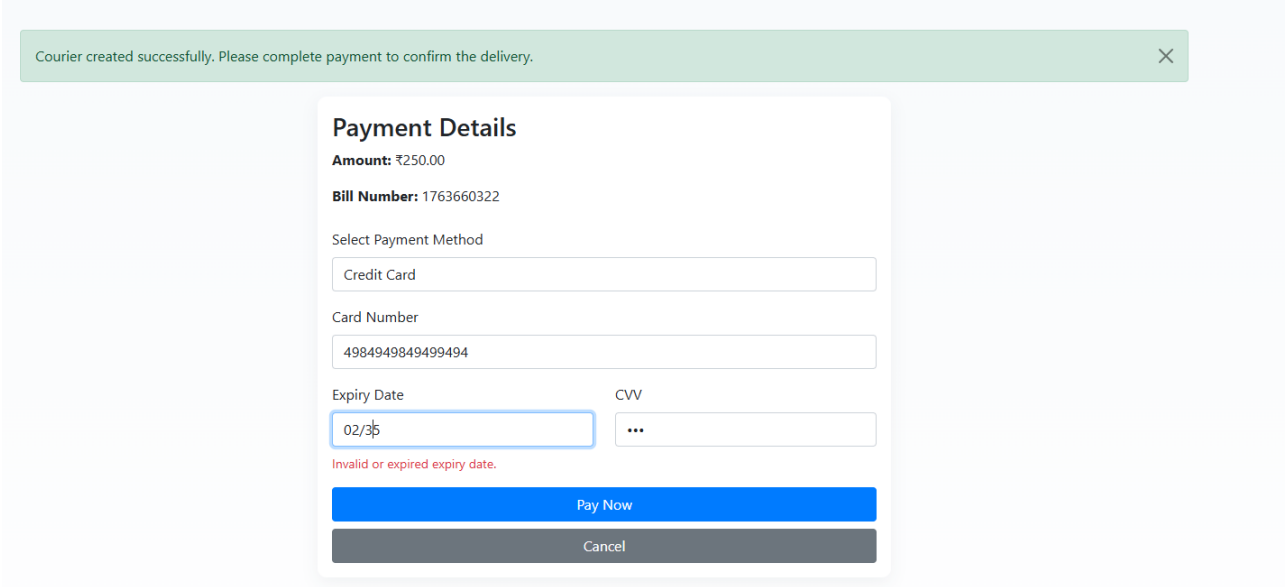
Application Screenshots:

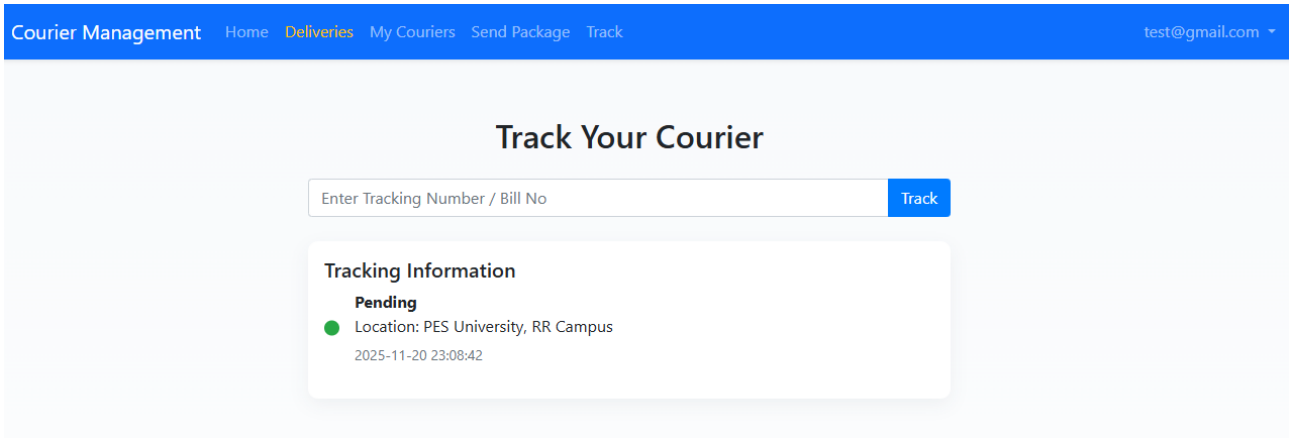


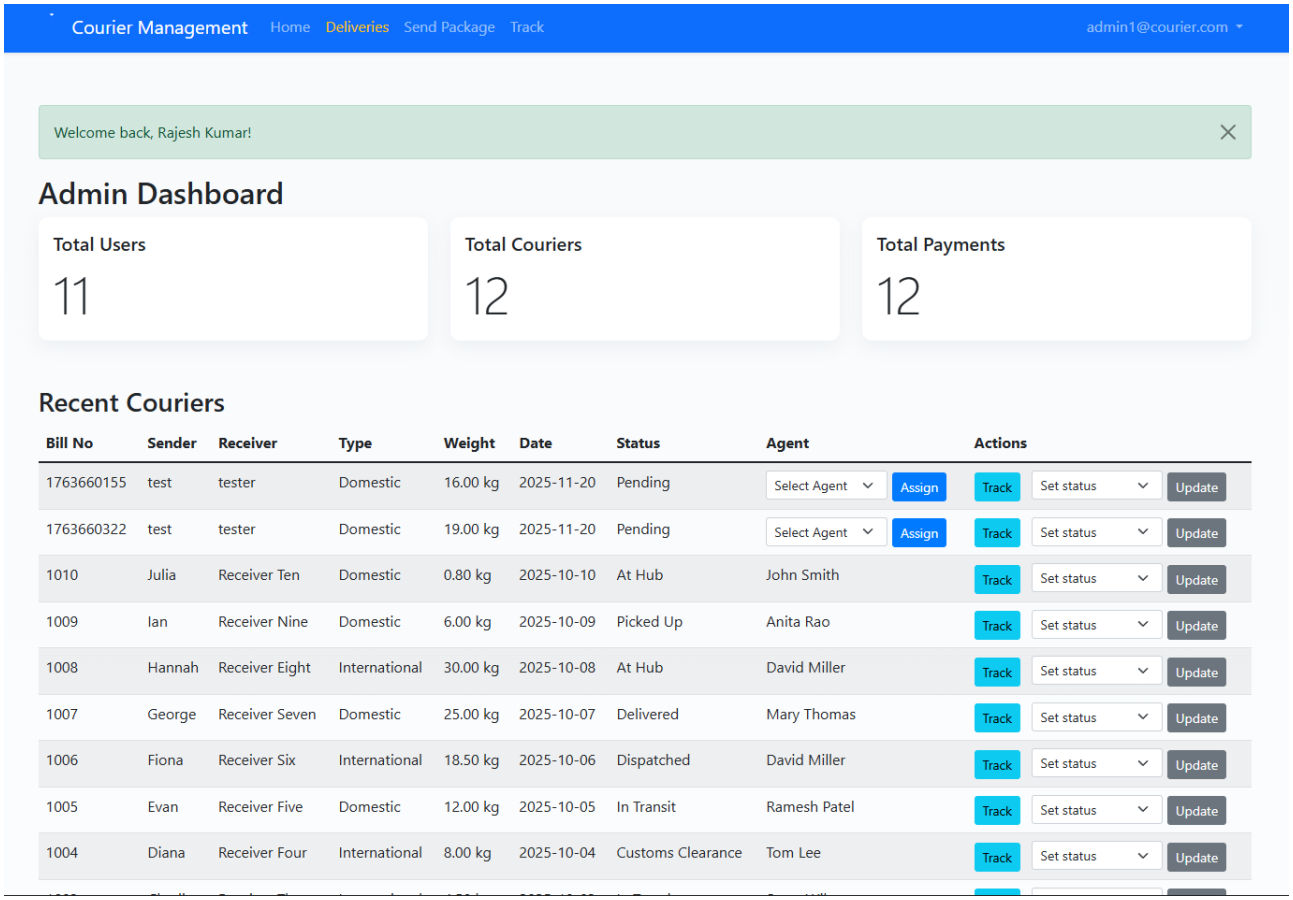



Courier Management
Home
Deliveries
ravi@d-agent.com
Welcome, Ravi Kumar!
Your Assigned Couriers
Bill No
Sender
Receiver
Type
Weight
Status
Actions
1001
Alice
Receiver One
Domestic
0.50 kg
Delivered
Closed


Courier Management
Home
Deliveries
My Couriers
Send Package
Track
test@gmail.com
Send a Courier
Sender Details
Name
test
Email
test@gmail.com
Phone
4564879159
Address
PES University, RR Campus
Receiver Details
Name
tester
Email
tester@gmail.com
Phone
8974562132
Address
PES University, EC Campus
Package Details
Weight (kg)
19
Courier Type
Domestic
Destination Country
India
Create Courier
Cancel


Courier created successfully. Please complete payment to confirm the delivery.
Payment Details
Amount: ₹250.00
Bill Number: 1763660322
Select Payment Method
Credit Card
Card Number
4984949849499494
Expiry Date
02/35
CVV
Invalid or expired expiry date.
Pay Now
Cancel


Courier Management
Home
Deliveries
My Couriers
Send Package
Track
test@gmail.com
Track Your Courier
Enter Tracking Number / Bill No
Track
Tracking Information
Pending
Location: PES University, RR Campus
2025-11-20 23:08:42


Courier Management
Home
Deliveries
Send Package
Track
admin1@courier.com
Welcome back, Rajesh Kumar!
Admin Dashboard
Total Users
11
Total Couriers
12
Total Payments
12
Recent Couriers
Bill No
Sender
Receiver
Type
Weight
Date
Status
Agent
Actions
1763660155 test tester Domestic 16.00 kg 2025-11-20 Pending Select Agent Assign Track Set status Update
1763660322 test tester Domestic 19.00 kg 2025-11-20 Pending Select Agent Assign Track Set status Update
1010 Julia Receiver Ten Domestic 0.80 kg 2025-10-10 At Hub John Smith Track Set status Update
1009 Ian Receiver Nine Domestic 6.00 kg 2025-10-09 Picked Up Anita Rao Track Set status Update
1008 Hannah Receiver Eight International 30.00 kg 2025-10-08 At Hub David Miller Track Set status Update
1007 George Receiver Seven Domestic 25.00 kg 2025-10-07 Delivered Mary Thomas Track Set status Update
1006 Fiona Receiver Six International 18.50 kg 2025-10-06 Dispatched David Miller Track Set status Update
1005 Evan Receiver Five Domestic 12.00 kg 2025-10-05 In Transit Ramesh Patel Track Set status Update
1004 Diana Receiver Four International 8.00 kg 2025-10-04 Customs Clearance Tom Lee Track Set status Update