SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of India)
Re-Accredited by NAAC with 'A++' Grade

A DCDSL PROJECT REPORT

ON

"Cargo Management System"

Submitted By

Student Name	PRN
Panchal Shlok	24070126132
Priyansh Johri	24070126142

UNDER THE GUIDANCE OF

Prof. Pooja Kamat

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

A.Y. 2025-26

TABLE OF CONTENTS

SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of India)
Re-Accredited by NAAC with 'A++' Grade

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

This is to certify that the DCDSL Project work entitled "**Cargo Management System**" is carried out by the **Panchal Shlok and Priyansh Johri,** in **Artificial Intelligence & Machine Learning**, Symbiosis International (Deemed University), Pune during the academic year 2024-2025.

Name and Signature of the Guide

Dr. Shilpa Bade-Gite

Head, Department of AI&ML

1.INTRODUCTION

The **cargo management system** is a **full-stack web application** designed to manage, monitor, and automate cargo operations in logistics and freight companies.
It provides a centralized platform to handle records related to **clients, shipments, vehicles, warehouses, payments, and insurance details** through an intuitive graphical interface connected to a **mysql database**.

- The main goal of this project is to simplify and digitize cargo logistics management by enabling efficient data handling, reducing paperwork, minimizing manual errors, and providing **real-time data accessibility** for better decision-making.
- The system is implemented using **Python (Flask)** for the backend server and API integration, **MySQL** for database management, and a **frontend built with HTML, CSS, and JavaScript** for user interaction.
  This architecture ensures seamless connectivity between the UI and the backend, allowing dynamic operations such as data entry, search, update, and deletion directly from the interface.

2. Problem Statement

The logistics and freight industry involves managing a vast amount of information — including client records, cargo details, shipments, warehouse storage, customs clearance, vehicle allocation, and payments.
Traditionally, this data is handled manually or via disjointed systems, which leads to several operational inefficiencies such as:

- Data redundancy and inconsistency.
- Delays in tracking shipments and updating cargo information.
- Difficulty in data retrieval, editing, and verification.
- Poor integration between departments like billing, shipment, and warehouse units.

To overcome these issues, the Cargo Management System offers a centralized and intelligent database-driven solution that supports real-time CRUD operations and advanced cross-table searching.

It bridges the gap between multiple functional modules, ensuring accuracy, consistency, and improved management efficiency across all cargo processes.

3.System Architecture and Modules

System Architecture

The system follows a **3-tier architecture** that ensures modularity, scalability, and ease of maintenance.

1. **Frontend (Presentation Layer):**
    1. Built using **HTML, CSS, and JavaScript**, this layer provides an intuitive graphical interface for the user.
    2. It allows users to perform various operations such as adding, updating, deleting, and searching records.
    3. The frontend communicates with the backend API using **HTTP requests (GET, POST, PUT, DELETE)** to fetch or modify database information dynamically.
2. **Backend (Application Layer):**
    1. Developed using **Python (Flask)**, this layer handles all **business logic** and acts as a bridge between the frontend and the database.
    2. It processes **API requests**, validates user inputs, and performs CRUD operations through structured endpoints.
    3. Flask's lightweight nature and RESTful API support make it efficient for real-time communication with the MySQL database.
3. **Database (Data Layer):**
    1. The **MySQL** database stores all structured information, including details about clients, shipments, vehicles, warehouses, payments, and insurance claims.
    2. The schema is designed with **primary and foreign keys** to ensure referential integrity and reduce data redundancy.
    3. SQL queries are executed through Flask's MySQL connector to perform operations securely and efficiently.

Modules

The Cargo Management System is divided into multiple functional modules that work together seamlessly:

1. **Client Management Module:** Manages client profiles including contact details, company names, and associated cargo records.
2. **Cargo Management Module:** Stores and tracks details of all cargo items including type, weight, and value.
3. **Shipment Module:** Handles shipment details such as origin, destination, mode of transport, and status updates.
4. **Tracking and Logistics Module:** Maintains real-time tracking logs for every shipment, recording timestamps, locations, and remarks.
5. **Vehicle Management Module:** Manages details about vehicles including type, capacity, driver information, and operational status.
6. **Warehouse and Storage Module:** Tracks warehouse locations, stored cargo, and release times to manage storage efficiently.
7. **Payments and Insurance Module:** Records all payment transactions and insurance claims associated with each cargo or shipment.
8. **Search and Reporting Module:** Allows users to perform smart cross-table searches using client, cargo, or shipment IDs, automatically retrieving all related information across the system.

4. Functional Requirements

User Requirements

- Ability to add, update, and delete records for any entity such as clients, cargo, shipments, etc.
- Perform smart searches using client_id, cargo_id, or shipment_id to retrieve complete relational data.
- Access a user-friendly web interface with simple navigation, alerts, and confirmation messages for every action.
- Real-time feedback through success and error messages to confirm user operations.

System Requirements

- Server: Python runtime (Flask Framework)
- Database: MySQL
- Browser: Google Chrome, Microsoft Edge, or Mozilla Firefox
- Libraries: Flask, Flask-CORS, mysql-connector-python, python-dotenv

Functional Features

1. **CRUD Operations:** Users can perform Create, Read, Update, and Delete actions on any table via RESTful Flask APIs.
2. **Smart Universal Search:** The system supports intelligent search that automatically links data across multiple tables (clients, cargos, shipments, etc.) and presents complete results.
3. **Dynamic Dashboard:** The frontend auto-refreshes after each operation and displays visual feedback (success/error messages) dynamically.
4. **Secure Data Handling:** Input validation and safe SQL query execution prevent SQL injection or data corruption.
5. **Responsive Interface:** The frontend layout adapts for smooth usage on different desktop screen sizes.

6. **Scalable Modular Design:** Each module (clients, cargo, payments, etc.) operates independently but remains connected through relational logic, making future expansion easy.

Entities:

Relational Schema

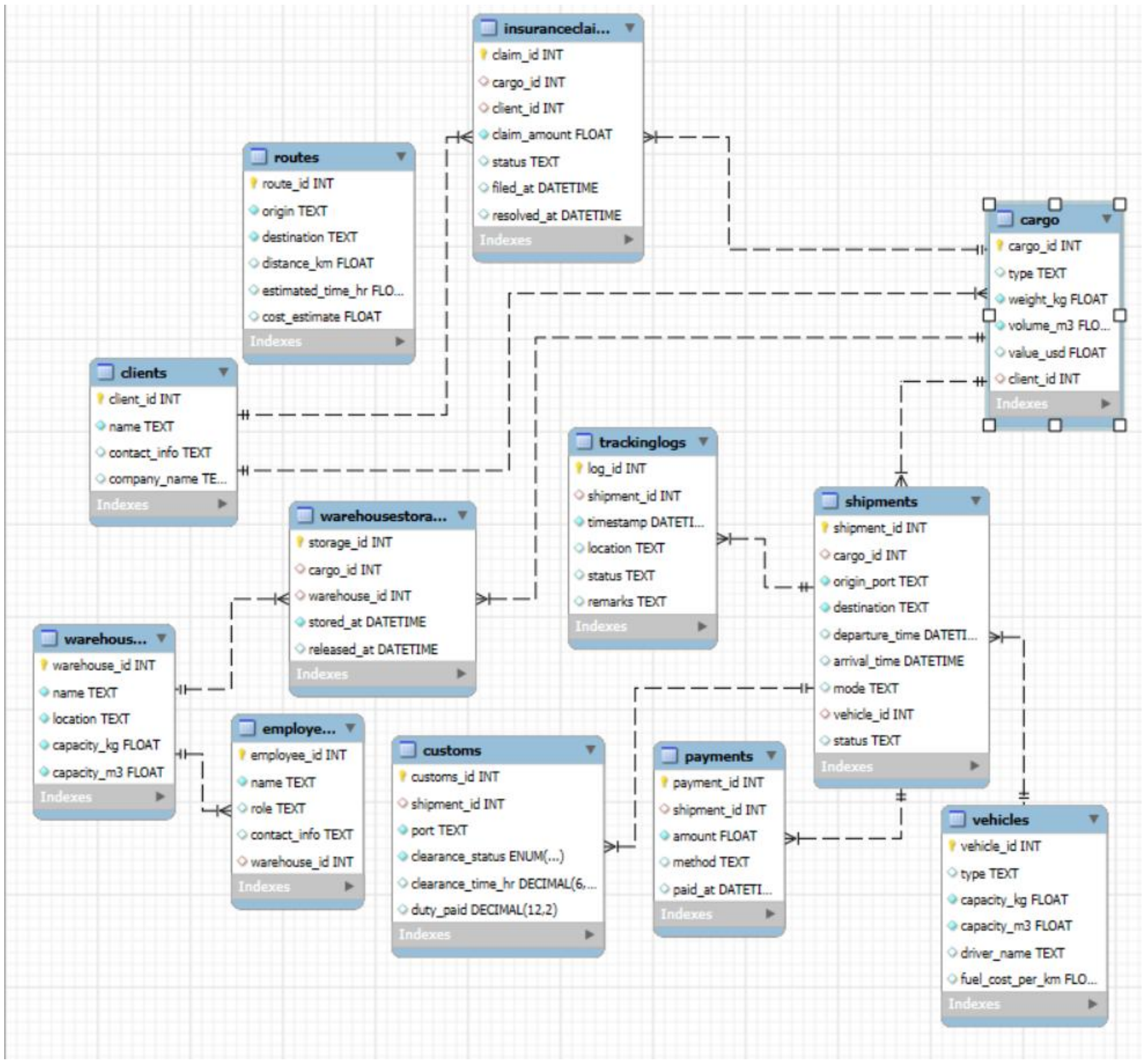
- **Clients** (client_id **PK**, name, contact_info, company_name)
- **Cargo** (cargo_id **PK**, type, weight_kg, volume_m3, value_usd, client_id **FK**)
- **Shipments** (shipment_id **PK**, cargo_id **FK**, origin_port, destination, departure_time, arrival_time, mode, vehicle_id **FK**, status)
- **Customs** (customs_id **PK**, shipment_id **FK**, port, clearance_status, clearance_time_hr, duty_paid)
- **Payments** (payment_id **PK**, shipment_id **FK**, amount, method, paid_at)
- **Trackinglogs** (log_id **PK**, shipment_id **FK**, timestamp, location, status, remarks)
- **Insuranceclaims** (claim_id **PK**, cargo_id **FK**, client_id **FK**, claim_amount, status, filed_at, resolved_at)
- **Vehicles** (vehicle_id **PK**, type, capacity_kg, capacity_m3, driver_name, fuel_cost_per_km)
- **Routes (**route_id **PK**, origin, destination, distance_km, estimated_time_hr, cost_estimate)
- **Warehouse** (warehouse_id **PK**, name, location, capacity_kg, capacity_m3)
- **Employees** (employee_id **PK**, name, role, contact_info, warehouse_id **FK**)
- **warehousestorage**(storage_id **PK**, cargo_id **FK**, warehouse_id **FK**, stored_at, released_at)

Relationships:

- One **Client** → Many **Cargo**
- One **Cargo** → Many **Shipments**
- One **Shipment** → Many **Tracking Logs**
- One **Warehouse** → Many **Employees** and **Storage**

- One **Cargo** → One **Route**, One **InsuranceClaim**, Many **Payments**

Relational Schema



Implementation

A) database setup

1. Install mysql server and node.js.
2. Create a database named cargomanagementsystem.
3. Configure .env file with database credentials.
4. Run npm install to install dependencies.
5. Start server using node server.js.

B) table creation scripts

```
Create table clients (
  client_id int auto_increment primary key,
  name varchar(100),
  contact varchar(15),
  address varchar(255),
  email varchar(100)
);
Create table cargo (
  cargo_id int auto_increment primary key,
  client_id int,
  description text,
  weight float,
  destination varchar(100),
  foreign key (client_id) references clients(client_id)
);
```

(similarly define all other tables like shipments, vehicles, warehouses, etc.)

C) stored procedures, views, and indexes

- Views: cargo_overview view combines client and cargo details.
- Indexes: primary keys on all tables.
- No stored procedures used — crud handled via node.js api.

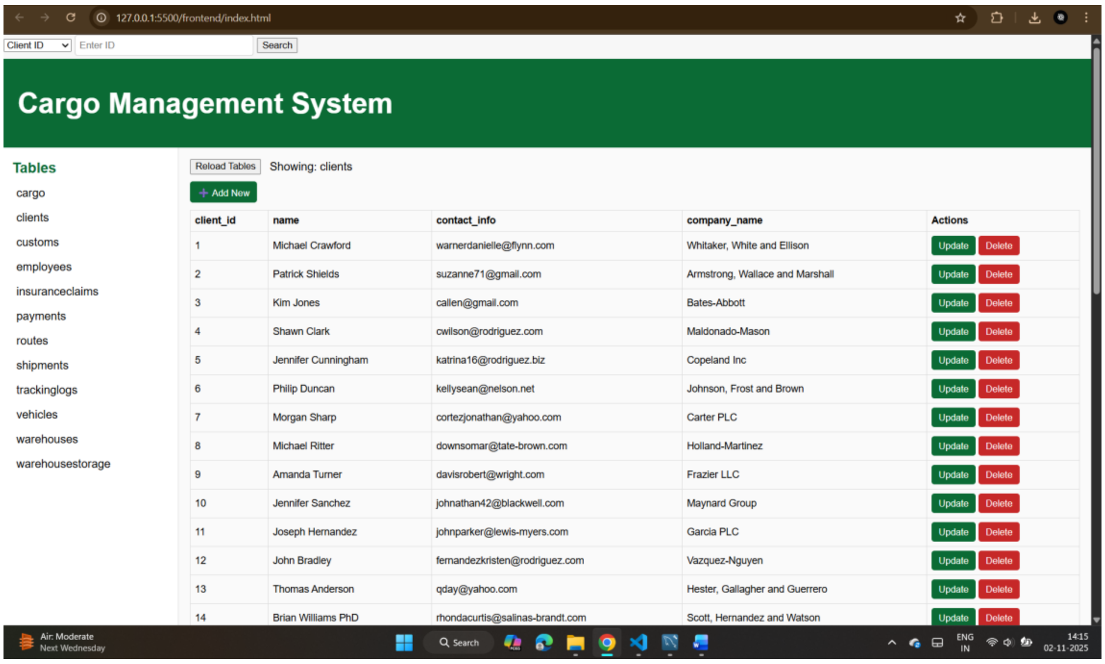
D) sample data

```
Insert into clients values (1, 'ravi kumar', '9876543210', 'delhi', 'ravi@example.com');
Insert into cargo values (1, 1, 'electronics', 500, 'mumbai');
Insert into shipments values (1, 1, 3, '2025-10-01', 'delivered');
```

User Interface

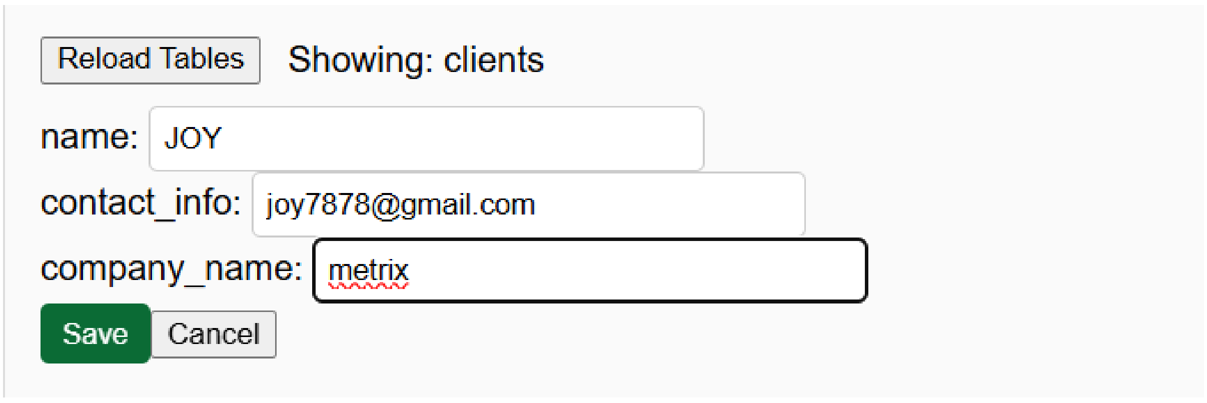
a) Screenshots of the UI

- Dashboard showing all tables.



client_id	name	contact_info	company_name	Actions
1	Michael Crawford	warnerdanielle@flynn.com	Whitaker, White and Ellison	Update Delete
2	Patrick Shields	suzanne71@gmail.com	Armstrong, Wallace and Marshall	Update Delete
3	Kim Jones	callen@gmail.com	Bates-Abbott	Update Delete
4	Shawn Clark	cwilson@rodriguez.com	Maldonado-Mason	Update Delete
5	Jennifer Cunningham	katrina16@rodriguez.biz	Copeland Inc	Update Delete
6	Philip Duncan	kellysean@nelson.net	Johnson, Frost and Brown	Update Delete
7	Morgan Sharp	cortezjonathan@yahoo.com	Carter PLC	Update Delete
8	Michael Ritter	downsomar@tate-brown.com	Holland-Martinez	Update Delete
9	Amanda Turner	davisrobert@wright.com	Frazier LLC	Update Delete
10	Jennifer Sanchez	johnathan42@blackwell.com	Maynard Group	Update Delete
11	Joseph Hernandez	johnparker@lewis-myers.com	Garcia PLC	Update Delete
12	John Bradley	fernandezkristen@rodriguez.com	Vazquez-Nguyen	Update Delete
13	Thomas Anderson	qday@yahoo.com	Hester, Gallagher and Guerrero	Update Delete
14	Brian Williams PhD	rhondacurtis@salinas-brandt.com	Scott, Hernandez and Watson	Update Delete

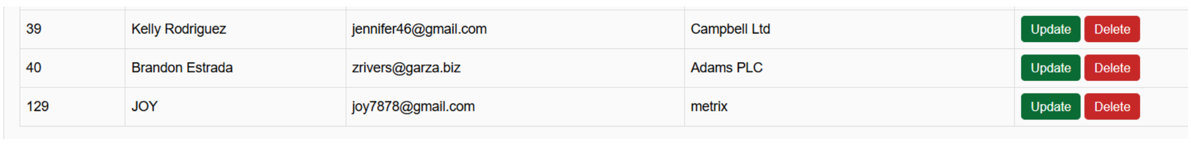
- Add Record form.



name: JOY

contact_info: joy7878@gmail.com

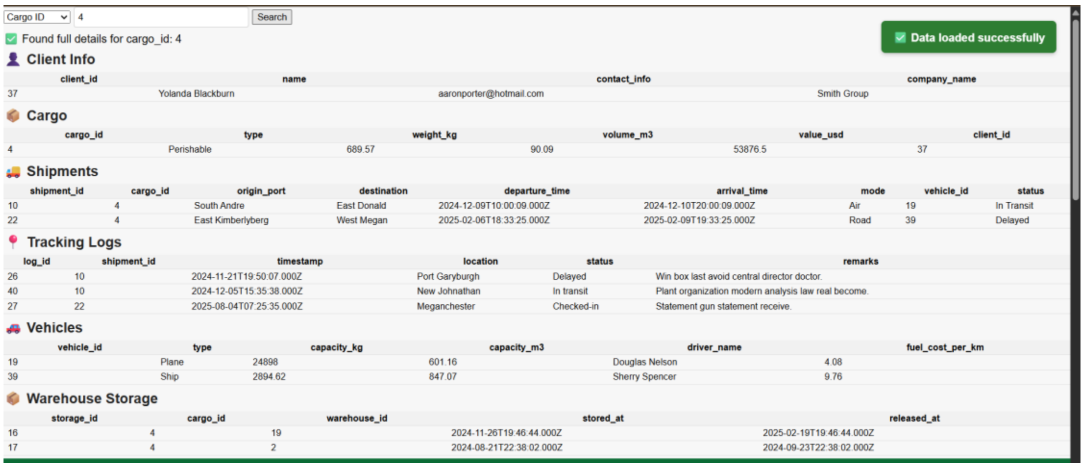
company_name: metrix



39	Kelly Rodriguez	jennifer46@gmail.com	Campbell Ltd	Update Delete
40	Brandon Estrada	zrivers@garza.biz	Adams PLC	Update Delete
129	JOY	joy7878@gmail.com	metrix	Update Delete

Smart Search results with related tables.

Using cargo_id



✅ Found full details for cargo_id: 4

Client Info

client_id	name	contact_info	company_name
37	Yolanda Blackburn	aaronporter@hotmail.com	Smith Group

Cargo

cargo_id	type	weight_kg	volume_m3	value_usd	client_id
4	Perishable	689.57	90.09	53876.5	37

Shipments

shipment_id	cargo_id	origin_port	destination	departure_time	arrival_time	mode	vehicle_id	status
10	4	South Andre	East Donald	2024-12-09T10:00:09.000Z	2024-12-10T20:00:09.000Z	Air	19	In Transit
22	4	East Kimberlyberg	West Megan	2025-02-06T18:33:25.000Z	2025-02-09T19:33:25.000Z	Road	39	Delayed

Tracking Logs

log_id	shipment_id	timestamp	location	status	remarks
26	10	2024-11-21T19:50:07.000Z	Port Garyburgh	Delayed	Win box last avoid central director doctor.
40	10	2024-12-05T15:35:38.000Z	New Johnathan	In transit	Plant organization modern analysis law real become.
27	22	2025-08-04T07:25:35.000Z	Meganchester	Checked-in	Statement gun statement receive.

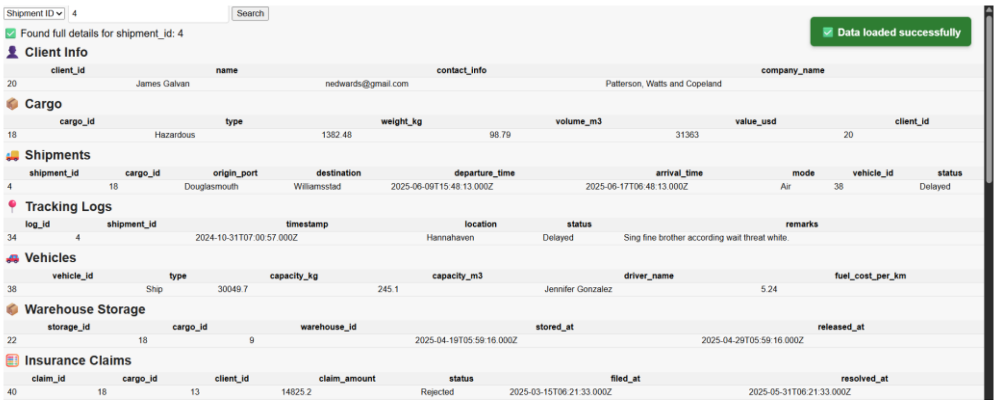
Vehicles

vehicle_id	type	capacity_kg	capacity_m3	driver_name	fuel_cost_per_km
19	Plane	24898	601.16	Douglas Nelson	4.08
39	Ship	2894.62	847.07	Sherry Spencer	9.76

Warehouse Storage

storage_id	cargo_id	warehouse_id	stored_at	released_at
16	4	19	2024-11-26T19:46:44.000Z	2025-02-19T19:46:44.000Z
17	4	2	2024-08-21T22:38:02.000Z	2024-09-23T22:38:02.000Z

Using shipment_id



✅ Found full details for shipment_id: 4

Client Info

client_id	name	contact_info	company_name
20	James Galvan	nedwards@gmail.com	Patterson, Watts and Copeland

Cargo

cargo_id	type	weight_kg	volume_m3	value_usd	client_id
18	Hazardous	1382.48	98.79	31363	20

Shipments

shipment_id	cargo_id	origin_port	destination	departure_time	arrival_time	mode	vehicle_id	status
4	18	Douglasmouth	Williamsstad	2025-06-09T15:48:13.000Z	2025-06-17T06:48:13.000Z	Air	38	Delayed

Tracking Logs

log_id	shipment_id	timestamp	location	status	remarks
34	4	2024-10-31T07:00:57.000Z	Hannahaven	Delayed	Sing fine brother according wait threat white.

Vehicles

vehicle_id	type	capacity_kg	capacity_m3	driver_name	fuel_cost_per_km
38	Ship	30049.7	245.1	Jennifer Gonzalez	5.24

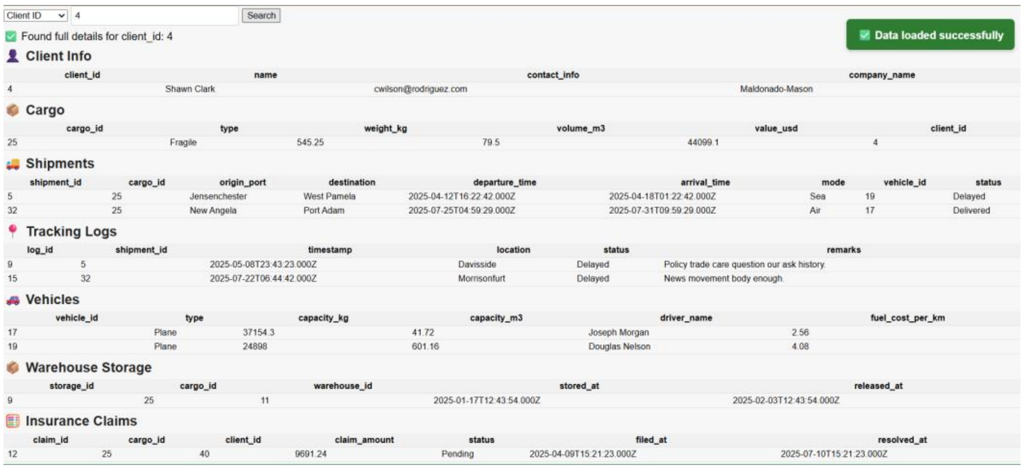
Warehouse Storage

storage_id	cargo_id	warehouse_id	stored_at	released_at
22	18	9	2025-04-19T05:59:16.000Z	2025-04-29T05:59:16.000Z

Insurance Claims

claim_id	cargo_id	client_id	claim_amount	status	filed_at	resolved_at
40	18	13	14825.2	Rejected	2025-03-15T06:21:33.000Z	2025-05-31T06:21:33.000Z

Using Client_id



✅ Found full details for client_id: 4

Client Info

client_id	name	contact_info	company_name
4	Shawn Clark	cwilson@rodriguez.com	Maldonado-Mason

Cargo

cargo_id	type	weight_kg	volume_m3	value_usd	client_id
25	Fragile	545.25	79.5	44099.1	4

Shipments

shipment_id	cargo_id	origin_port	destination	departure_time	arrival_time	mode	vehicle_id	status
5	25	Jensenchester	West Pamela	2025-04-12T16:22:42.000Z	2025-04-18T01:22:42.000Z	Sea	19	Delayed
32	25	New Angela	Port Adam	2025-07-25T04:59:29.000Z	2025-07-31T09:59:29.000Z	Air	17	Delivered

Tracking Logs

log_id	shipment_id	timestamp	location	status	remarks
9	5	2025-05-08T23:43:23.000Z	Davisside	Delayed	Policy trade care question our ask history.
15	32	2025-07-22T06:44:42.000Z	Morrisonfurt	Delayed	News movement body enough.

Vehicles

vehicle_id	type	capacity_kg	capacity_m3	driver_name	fuel_cost_per_km
17	Plane	37154.3	41.72	Joseph Morgan	2.56
19	Plane	24898	601.16	Douglas Nelson	4.08

Warehouse Storage

storage_id	cargo_id	warehouse_id	stored_at	released_at
9	25	11	2025-01-17T12:43:54.000Z	2025-02-03T12:43:54.000Z

Insurance Claims

claim_id	cargo_id	client_id	claim_amount	status	filed_at	resolved_at
12	25	40	9691.24	Pending	2025-04-09T15:21:23.000Z	2025-07-10T15:21:23.000Z

B) usage instructions

1. **Start the backend:** run the flask server using the command:

   Python app.py

(make sure your mysql database is running and credentials are correctly configured in .env.)

2. **Launch the frontend:** open index.html using a local web server (such as code live server or python's built-in http.server).
3. **View tables:** select any table (e.g., *clients*, *cargo*, *shipments*) from the sidebar to view all records dynamically.
4. **Add new records:** click the "add new" button to open a form and insert data into the selected table.
5. **Update or delete records:** use the "update" or "delete" buttons beside each row to modify or remove records directly.
6. **Search functionality:** use the smart search bar to find complete linked details based on client_id, cargo_id, or shipment_id.
7. **Real-time feedback:** all actions (insert, update, delete) instantly refresh the interface and display success/error messages.

Conclusion

The cargo management system efficiently integrates database management, flask-based server logic, and an interactive frontend interface into a unified web solution.This project demonstrates:

- Real-world application of relational database concepts.
- Implementation of restful api communication between frontend and backend.
- Integration of python, mysql, and javascript for seamless data management.
- Practical understanding of ai & data science lab-level project principles — logic, modularity, and scalability.

The system fulfills all objectives by providing a centralized, intelligent, and user-friendly platform to manage logistics data effectively.

References

- Flask documentation: https://flask.palletsprojects.com/
- Mysql official docs: https://dev.mysql.com/doc/
- Python mysql connector: https://pypi.org/project/mysql-connector-python/
- Mozilla developer network (mdn): https://developer.mozilla.org/
- W3schools web development tutorials: https://www.w3schools.com/
- Bootstrap (for ui enhancements): https://getbootstrap.com/