Database Management Systems Project

Submitted in Partial Fulfillment of requirements for the award of

Degree of Bachelor of Technology in Computer Science and Engineering

Submitted by

Deep Mathukiya (22BCP348)

Darshil Padmani (22BCP321)

Kashyap Vekariya (22BCP346)

Jenil Patel (22BCP332)

Submitted to

Dr. Sonam Nahar

Department of Computer Science and Engineering

School of Technology

Pandit Deendayal Energy University, Gandhinagar

April 2024

Logistics Management System Report

Introduction:

The Logistics Management System (LMS) is a comprehensive solution aligned with India's National Logistics Policy (NLP) 2022, aiming to optimize logistics operations. It focuses on improving route optimization, resource allocation, and inventory management to enhance overall efficiency in the logistics sector.

In today's competitive business landscape, efficient logistics management is crucial for businesses to meet customer demands and remain competitive. LMS addresses the challenges faced by logistics companies in India by leveraging technology and data-driven approaches to streamline processes and improve overall efficiency.

System Architecture:

Overview: The Logistics Management System (LMS) is a pivotal tool designed to enhance logistics operations within the framework of India's National Logistics Policy (NLP) 2022. It serves as a comprehensive solution to optimize route planning, resource allocation, and inventory management, thereby improving overall efficiency in the logistics sector.

Components:

- **Frontend (User Interface):** The frontend of the system is powered by a Streamlit application, providing users with an intuitive interface to interact with the system's functionalities.
- **Backend (Server-side Logic):** The backend of the system comprises server-side components responsible for processing user requests, interacting with the database, and executing business logic. This includes handling data validation, authentication, and authorization.
- **Database Management System (DBMS):** The system utilizes MySQL as the database management system to store and manage data efficiently. MySQL offers robust features for data storage, retrieval, and manipulation, ensuring reliability and scalability.

Data Flow: Data flows seamlessly through the system, starting from user inputs or database queries and culminating in the desired outputs. When a user interacts with the frontend interface, their requests are processed by the backend logic, which may involve querying the database for relevant information or performing calculations. The resulting data is then presented to the user through the frontend interface or used to update the database as necessary.

Integration with NLP 2022: The system aligns closely with the objectives outlined in the NLP 2022, aiming to streamline logistics processes, enhance transparency, and promote sustainability. Specific features, such as route optimization algorithms and inventory management tools, are tailored to meet the requirements and recommendations of the policy, ensuring compliance and alignment with industry standards.

Scalability and Performance: The architecture of the system is designed with scalability and performance in mind, allowing it to accommodate increasing data volumes and user loads efficiently. Measures such as database indexing, query optimization, and caching mechanisms are implemented to ensure optimal performance under varying workloads.

Security: Security is paramount in the logistics management system, and various measures are implemented to protect data and ensure system integrity. Access control mechanisms, encryption protocols, and other security features are employed to safeguard sensitive information and prevent unauthorized access or data breaches.

Dependencies and Technologies:

- **Technologies:** Streamlit for frontend development, Python for backend logic, MySQL for database management.
- **Frameworks:** Streamlit
- **Libraries:** Streamlit, pandas, mysql-connector-python
- **External Services/APIs:** None

Database Tables

Person Table:

- **Columns:** person_id (Primary Key), name
- **Functional Dependencies:**
    - person_id → name (Each person_id uniquely determines the name of the person.)

PersonContact Table:

- **Columns:** person_id (Foreign Key), contact_info (Foreign Key)
- **Functional Dependencies:**
    - (person_id, contact_info) → person_id (Each combination of person_id and contact_info uniquely determines the person_id.

Customer Table:

- **Columns:** customer_id (Primary Key), person_id (Foreign Key), address
- **Functional Dependencies:**
    - customer_id → person_id, address (Each customer_id uniquely determines the associated person_id and address.)

Warehouses Table:

- **Columns:** warehouse_id (Primary Key), name, location, capacity
- **Functional Dependencies:**
    - warehouse_id → name, location, capacity (Each warehouse_id uniquely determines the name, location, and capacity of the warehouse.)

`

Employees Table:

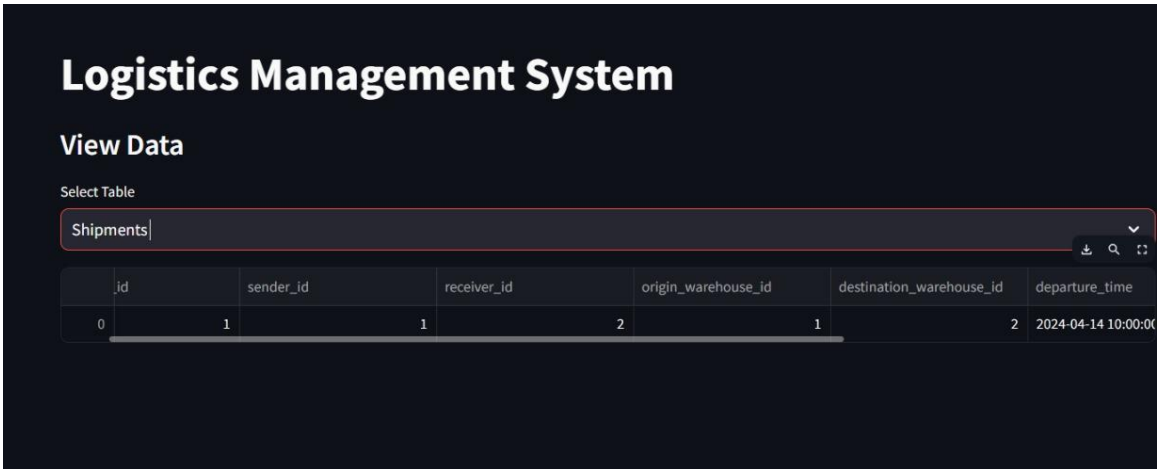
- **Columns:** employee_id (Primary Key), person_id (Foreign Key), position, department
- **Functional Dependencies:**
    - employee_id → person_id, position, department (Each employee_id uniquely determines the associated person_id, position, and department.)

Vehicles Table:

- **Columns:** vehicle_id (Primary Key), type, capacity, status
- **Functional Dependencies:**
    - vehicle_id → type, capacity, status (Each vehicle_id uniquely determines the type, capacity, and status of the vehicle.)

Shipments Table:

- **Columns:** shipment_id (Primary Key), sender_id (Foreign Key), receiver_id (Foreign Key), origin_warehouse_id (Foreign Key), destination_warehouse_id (Foreign Key), departure_time, arrival_time, status
- **Functional Dependencies:**
    - shipment_id → sender_id, receiver_id, origin_warehouse_id, destination_warehouse_id, departure_time, arrival_time, status (Each shipment_id uniquely determines the associated sender_id, receiver_id, origin_warehouse_id, destination_warehouse_id, departure_time, arrival_time, and status.)

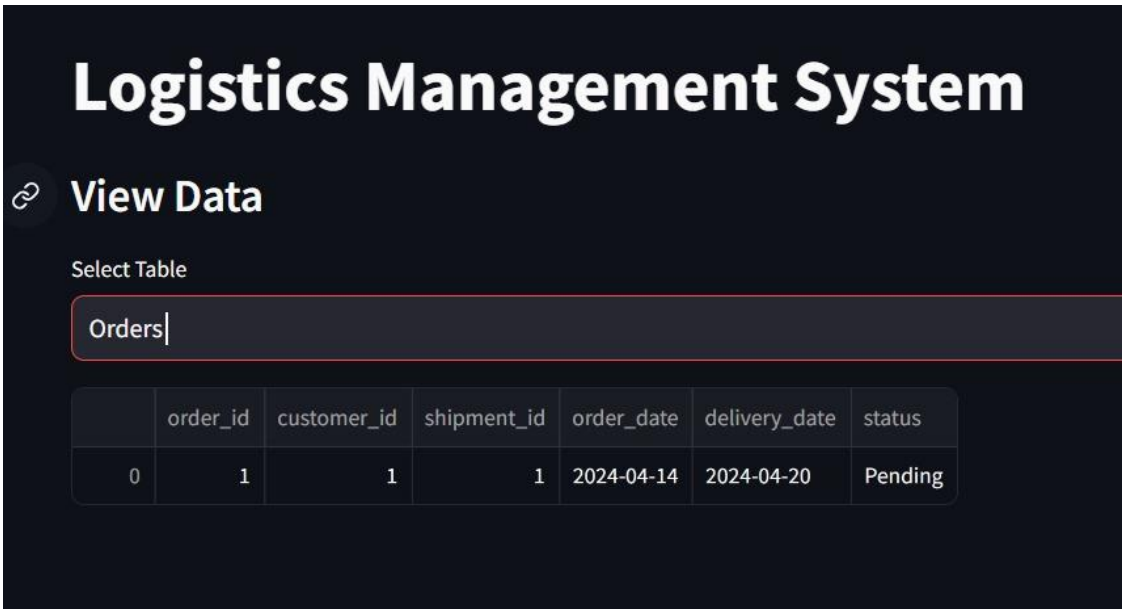


Routes Table:

- **Columns:** route_id (Primary Key), origin_location, destination_location, distance, duration
- **Functional Dependencies:**
    - route_id → origin_location, destination_location, distance, duration (Each route_id uniquely determines the origin_location, destination_location, distance, and duration of the route.)

Orders Table:

- **Columns:** order_id (Primary Key), customer_id (Foreign Key), shipment_id (Foreign Key), order_date, delivery_date, status
- **Functional Dependencies:**
    - order_id → customer_id, shipment_id, order_date, delivery_date, status (Each order_id uniquely determines the associated customer_id, shipment_id, order_date, delivery_date, and status.)

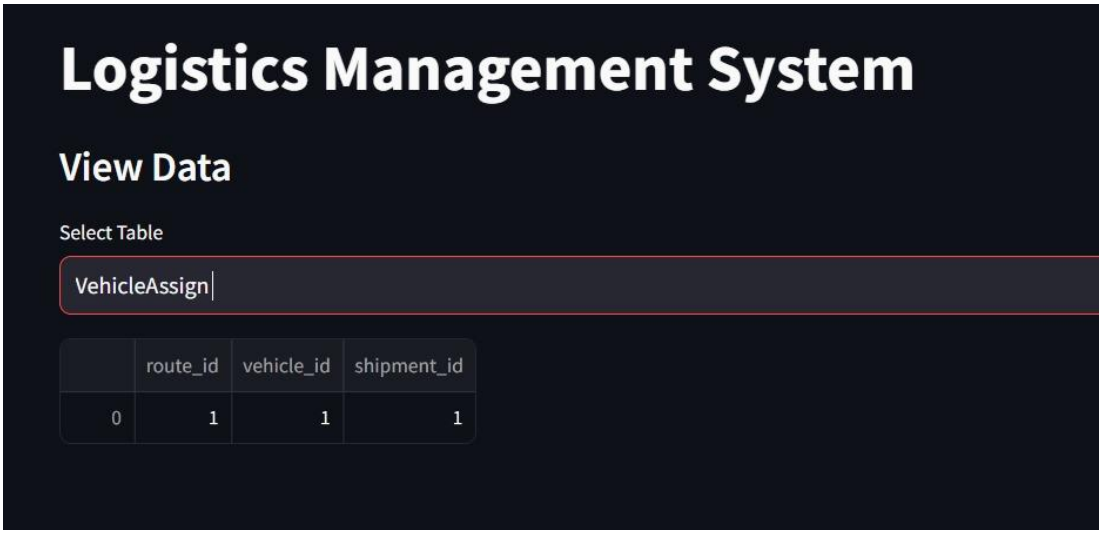


Inventory Table:

- **Columns:** inventory_id (Primary Key), warehouse_id (Foreign Key), quantity
- **Functional Dependencies:**
    - inventory_id → warehouse_id, quantity (Each inventory_id uniquely determines the associated warehouse_id and quantity.)

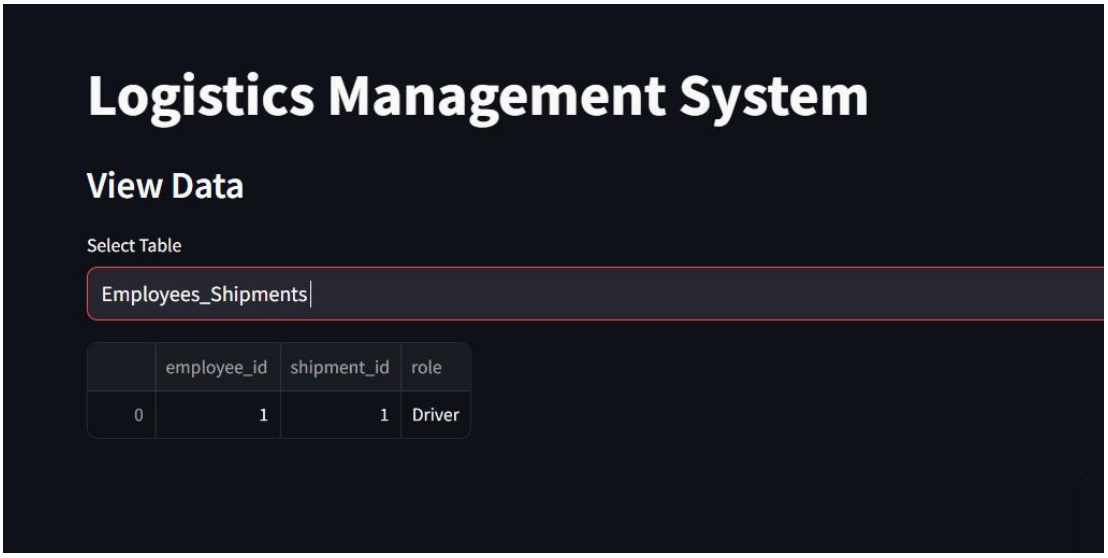
VehicleAssign Table:

- **Columns:** route_id (Foreign Key), vehicle_id (Foreign Key), shipment_id (Foreign Key)
- **Functional Dependencies:**
    - (route_id, vehicle_id, shipment_id) → route_id, vehicle_id, shipment_id (Each combination of route_id, vehicle_id, and shipment_id uniquely determines itself.)

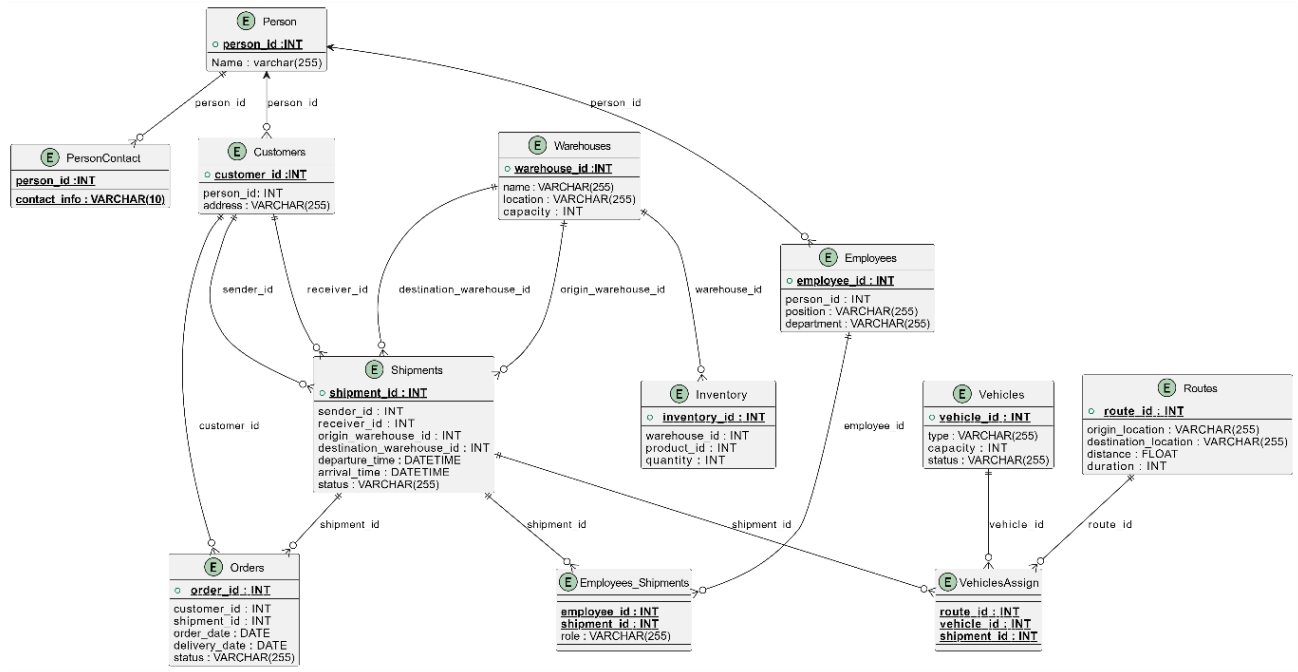


Employees_Shipments Table:

- **Columns:** employee_id (Foreign Key), shipment_id (Foreign Key), role
- **Functional Dependencies:**
    - (employee_id, shipment_id) → employee_id, shipment_id, role (Each combination of employee_id and shipment_id uniquely determines itself.)



Entity-Relationship (ER) Diagram:



Functionality

The Streamlit application provides users with intuitive features to interact seamlessly with the Logistics Management System (LMS) database. Here's how each functionality is presented in the user interface (UI):

Insert Data:

Users can easily insert data into different tables of the LMS database by selecting the desired table from the dropdown menu and filling in the required fields with relevant information. Upon clicking the "Insert" button, the data is submitted to the database.

Delete Data:

The UI allows users to delete data from various tables within the LMS database by selecting the table and providing the primary key or relevant identifier for the data to be deleted. Clicking the "Delete" button initiates the deletion process.

Update Data:

Updating data in the LMS database is straightforward through the Streamlit interface. Users can select the table containing the data to be updated, provide the primary key or identifier, and fill in the fields with the updated information. Clicking the "Update" button applies the changes to the database.

View Data:

Users can conveniently view data from different tables within the LMS database using the Streamlit application. The data is presented in a tabular format within the UI, allowing users to scroll through, filter, and analyze the information as needed.

By presenting these functionalities in the Streamlit UI, users can efficiently manage and analyze data for logistics operations, enhancing decision-making and overall system efficiency.

Normalization:

- The database tables have been normalized to reduce redundancy and improve data integrity.
- Normalization ensures that each table represents a single entity or relationship and that data is organized efficiently.

Example Queries:

1. **Retrieve all shipments sent from a specific warehouse:**

   SELECT * FROM Shipments WHERE origin_warehouse_id = <warehouse_id>;

2. **Update the status of a shipment:**

   UPDATE Shipments SET status = 'Delivered' WHERE shipment_id = <shipment_id>;

3. **Calculate the total quantity of a specific product in inventory:**

   SELECT SUM(quantity) AS total_quantity FROM Inventory WHERE product_id = <product_id>;

4. **Find the average duration of shipments from a specific origin to destination:**

   SELECT AVG(duration) AS average_duration FROM Routes WHERE origin_location = <origin> AND destination_location = <destination>;

5. **Retrieve all shipments with a status of "In Transit" sent from a specific warehouse:**

   SELECT * FROM Shipments WHERE origin_warehouse_id = <warehouse_id> AND status = 'In Transit';

Conclusion:

The Logistics Management System offers a comprehensive solution for optimizing logistics operations. With its user-friendly interface, robust database design, and efficient query capabilities, it provides a solid foundation for enhancing efficiency and effectiveness in India's logistics sector. Further enhancements and integrations can be explored to meet evolving industry needs and align with the objectives of the National Logistics Policy.