Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore(M.P.)

Group No. 07
Presentation–III

on

"TRIPLOGIX"-FLEET MANAGEMENT SYSTEM"

Presented by:
JAI SONI (21100BTCSE09852)
JAYA SINGH(21100BTCSE10476)
HARSH VYAS(21100BTCSE09839)
HIMANCHAL SINGH(21100BTCSE09844)

Guided By:
Dr. Vikas Jain

Department of Computer Science and Engineering
Shri Vaishnav Institute of Information Technology, Indore (M.P.)

Introduction

Efficient fleet management is crucial for modern logistics, yet traditional methods often rely on manual processes, leading to inefficiencies and compliance risks.

This project aims to develop a Truck Fleet Logistics Management System using Next.js and MongoDB to automate key tasks, including document recognition with OCR and LLM.

The system will streamline operations with features like trip duplication, Excel-like data management, and automated reporting, while ensuring regulatory compliance through timely notifications. This comprehensive solution will optimize fleet operations and enhance logistics performance

PROBLEM STATEMENT

The current fleet management system relies on manual processes, causing inefficiencies, errors, and delays.
Key issues include limited automation, inaccurate records, inefficient maintenance tracking, delayed decision-making, data inconsistencies, and poor financial oversight.

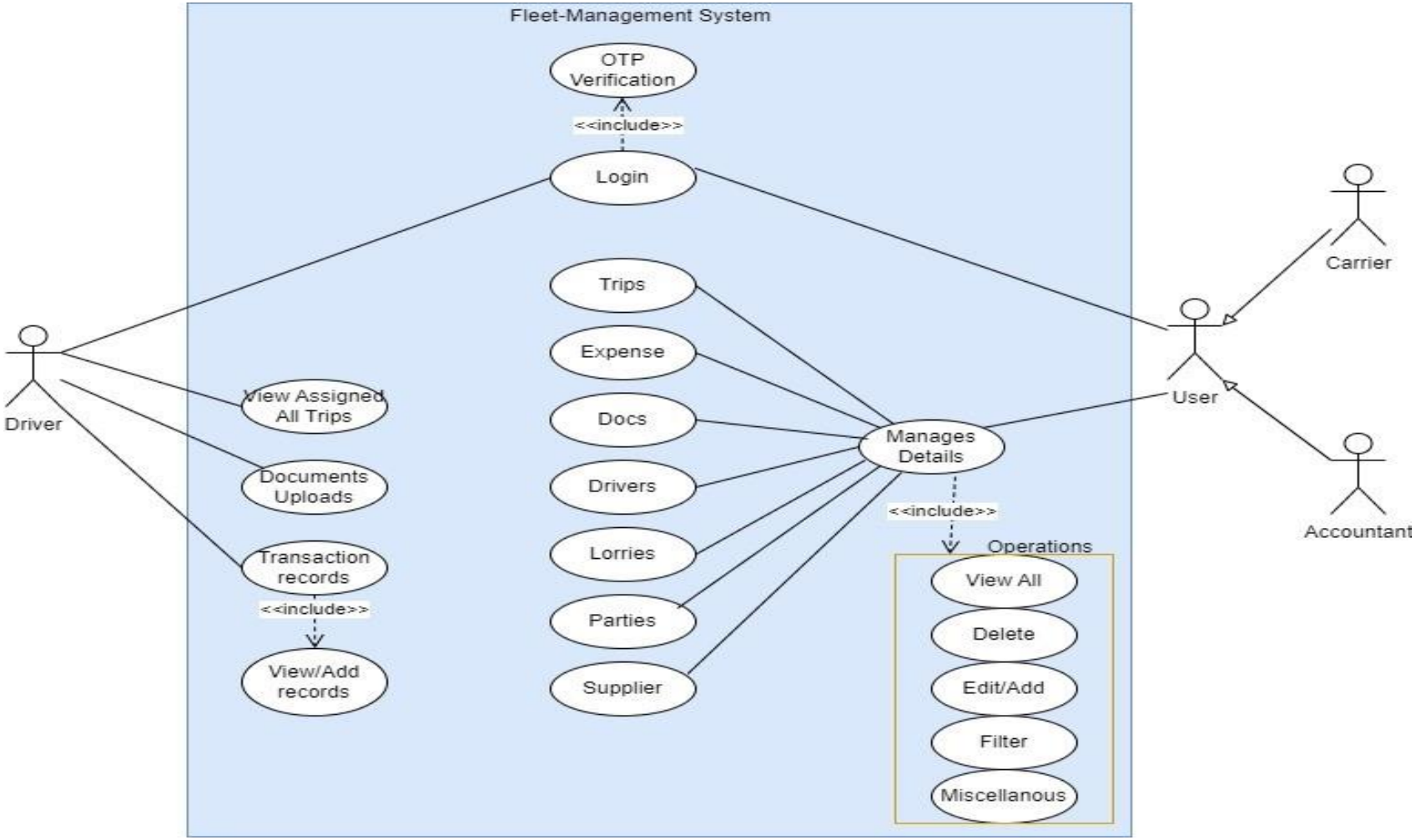
Managers struggle to monitor key metrics like fuel consumption and maintenance costs, missing opportunities for cost savings.

Compliance with regulations is also difficult to track manually, increasing the risk of non-compliance. These challenges emphasize the need for an automated, integrated system to enhance efficiency, reduce costs, and improve operations.

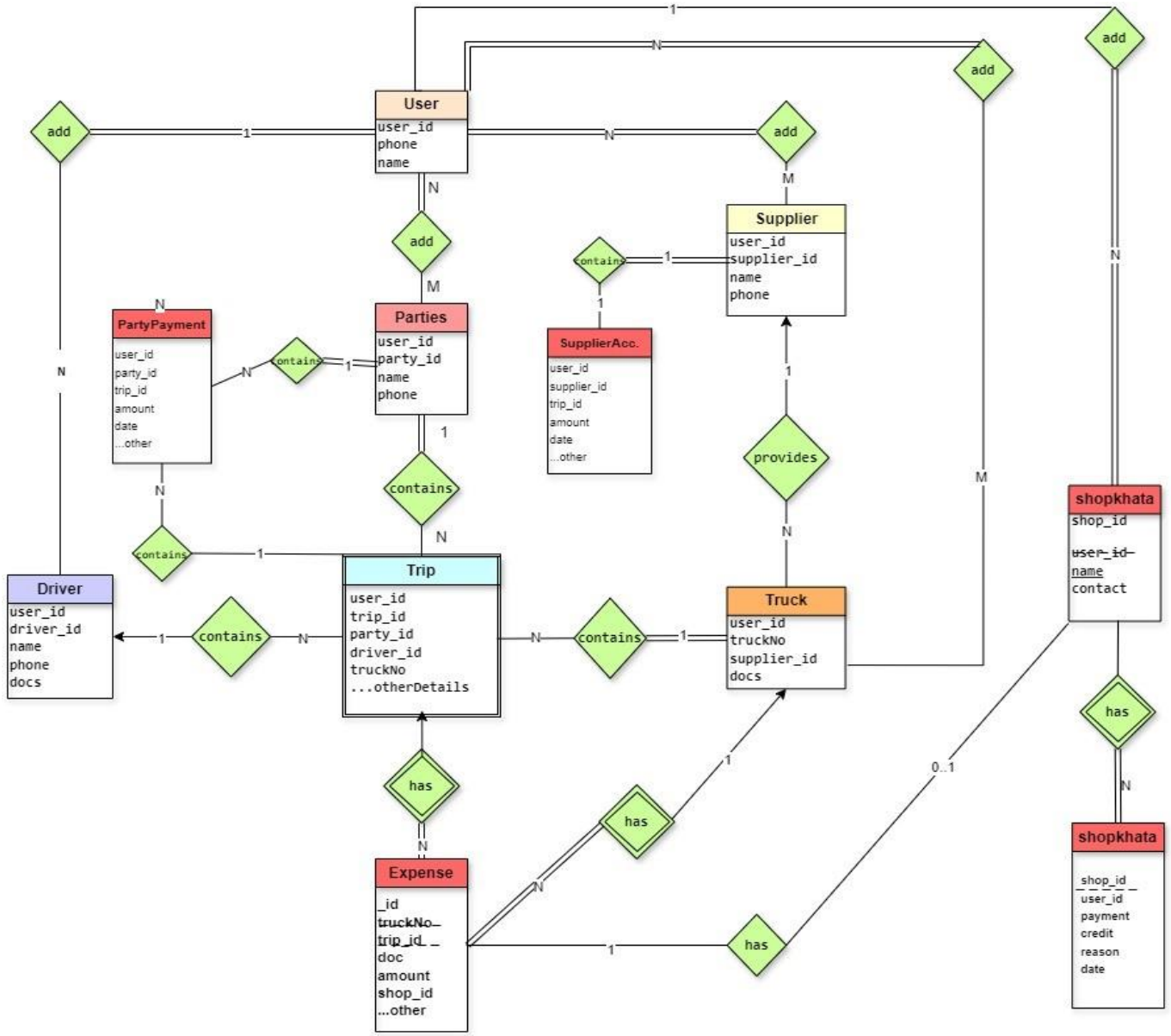
OBJECTIVE

- Automate trip creation, expense tracking, and document handling
- Secure OTP-based login and role-based access permissions
- Use OCR for document management and automated compliance alerts
- Manage trips, driver assignments, and expense reporting
- Generate real-time analytics and monthly reports
- Track vehicle maintenance and spare parts inventory
- Optimize routing and resource allocation
- Provide scalable, user-friendly interface with Excel-like data management
- Enhance financial oversight with automated cost tracking and compliance checks
- Store digital documents securely for easy access during audits

USE CASE DIAGRAM



ER Diagram


User
user_id
phone
name
add
Supplier
user_id
supplier_id
name
phone
contains
SupplierAcc.
user_id
supplier_id
trip_id
amount
date
...other
PartyPayment
user_id
party_id
trip_id
amount
date
...other
Parties
user_id
party_id
name
phone
contains
provides
Driver
user_id
driver_id
name
phone
docs
Trip
user_id
trip_id
party_id
driver_id
truckNo
...otherDetails
Truck
user_id
truckNo
supplier_id
docs
shopkhata
shop_id
user_id
name
contact
has
Expense
_id
truckNo
trip_id
doc
amount
shop_id
...other
shopkhata
shop_id
user_id
payment
credit
reason
date

Code snippet : Middleware.ts

```
 if(roleToken){
    const decodedToken : any= jwt.decode(roleToken as string)
    if(decodedToken?.role.name == 'driver' &&
!request.nextUrl.pathname.includes(`/user/drivers/${decodedToken.role.driver_id}`)){
      return NextResponse.redirect(new URL(`/user/drivers/${decodedToken.role.driver_id}`, request.url))
    }
 }


 if(loggedInUserNotAccessPaths){
    if(token){
      return NextResponse.redirect(new URL('/user/parties', request.url))
    }
 }else{
    if(!token){
      return NextResponse.redirect(new URL('/login', request.url))
    }
 }
```

BACKEND TESTING

•**Backend Testing**: Validated API endpoints using Postman for GET, POST, PUT, DELETE, ensuring correct data responses and status codes.

•**Security Testing**: Tested authentication (OTP, role-based access) and authorization by simulating invalid requests and checking system defenses

•**JWT Cookie Testing**: Verified secure storage of JWT tokens in cookies, tested session management (expired/invalid tokens), and ensured proper flags (HttpOnly, Secure) for protection.

•**Error Handling**: Checked how the system reacts to malformed requests and unauthorized actions.

•**Automated Testing**: Used Postman scripting to automate repetitive tests for reliability and efficiency.

UNIT TESTING

•**Build-Time Testing**: Unit tests were automatically triggered during the build process.

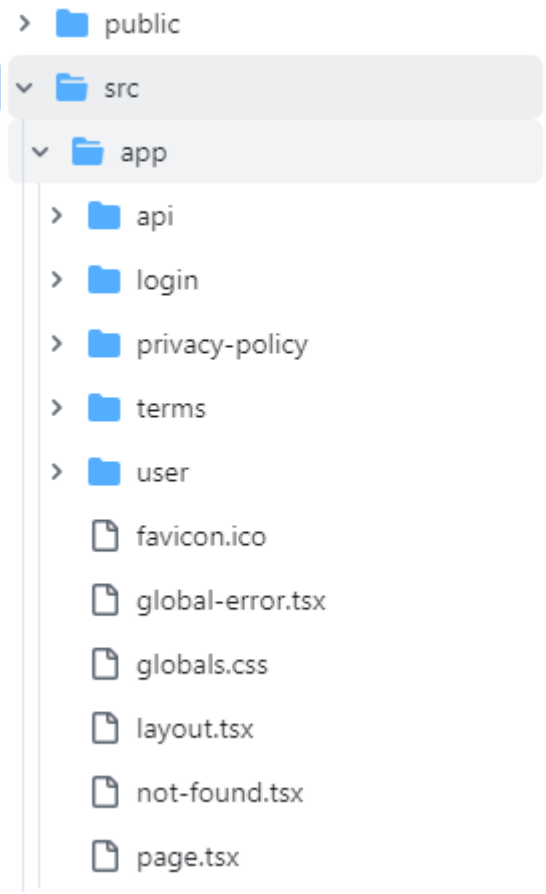
•**TypeScript Advantage**: TypeScript enabled strong type checking, catching errors early and ensuring code reliability.

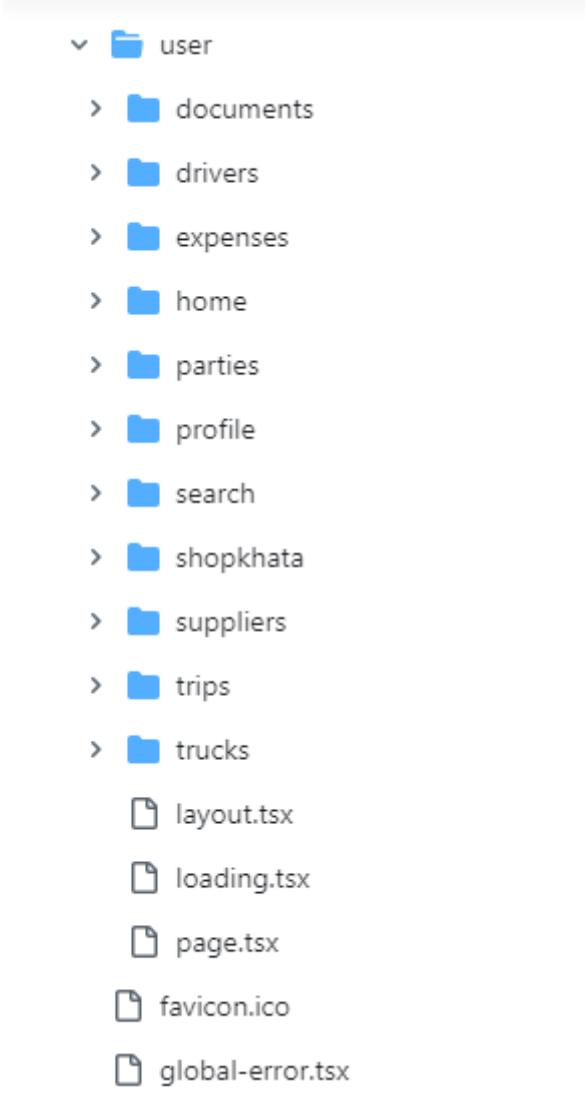
•**Linting**: Integrated linting tools ensured code quality, identifying potential issues such as unused variables, improper syntax, and style violations.

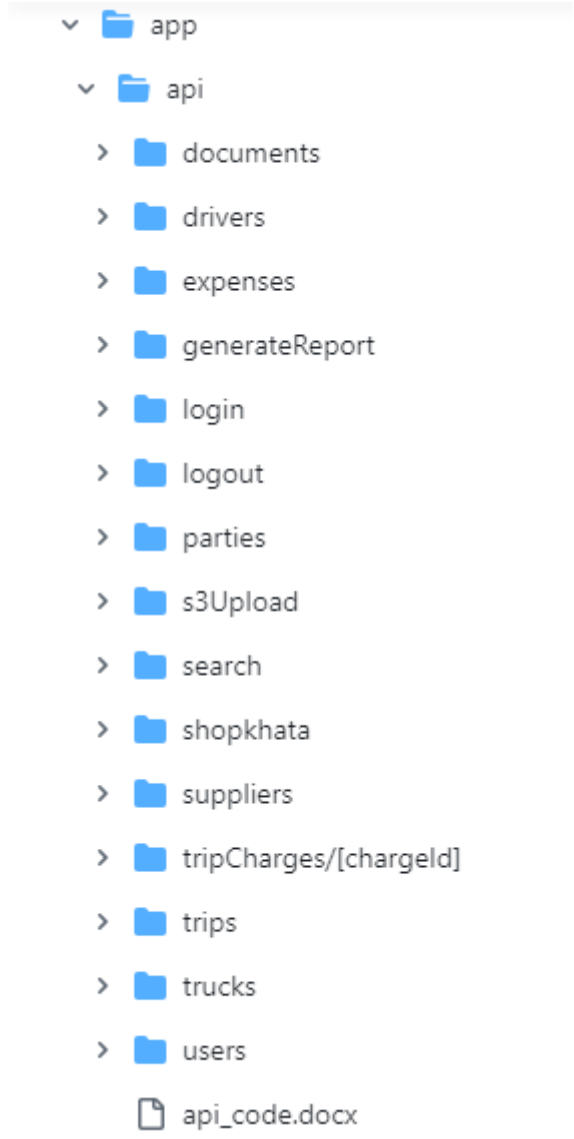
•**Error Detection**: Build would fail if type or syntax errors were detected, preventing faulty code from reaching production.

•**Efficiency**: This approach streamlined development by catching bugs at compile time, saving time during runtime testing.

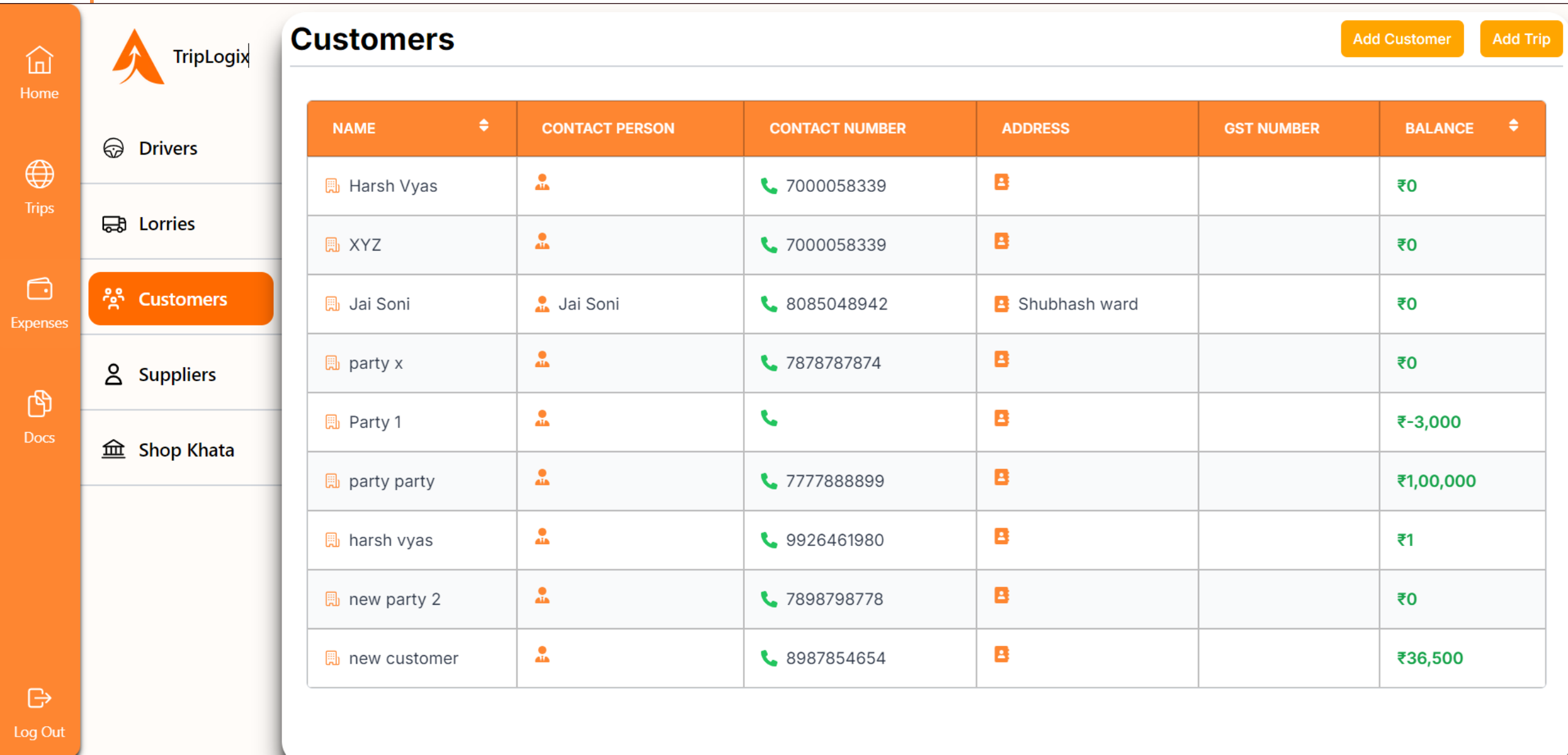
ROUTE STRUCTURE



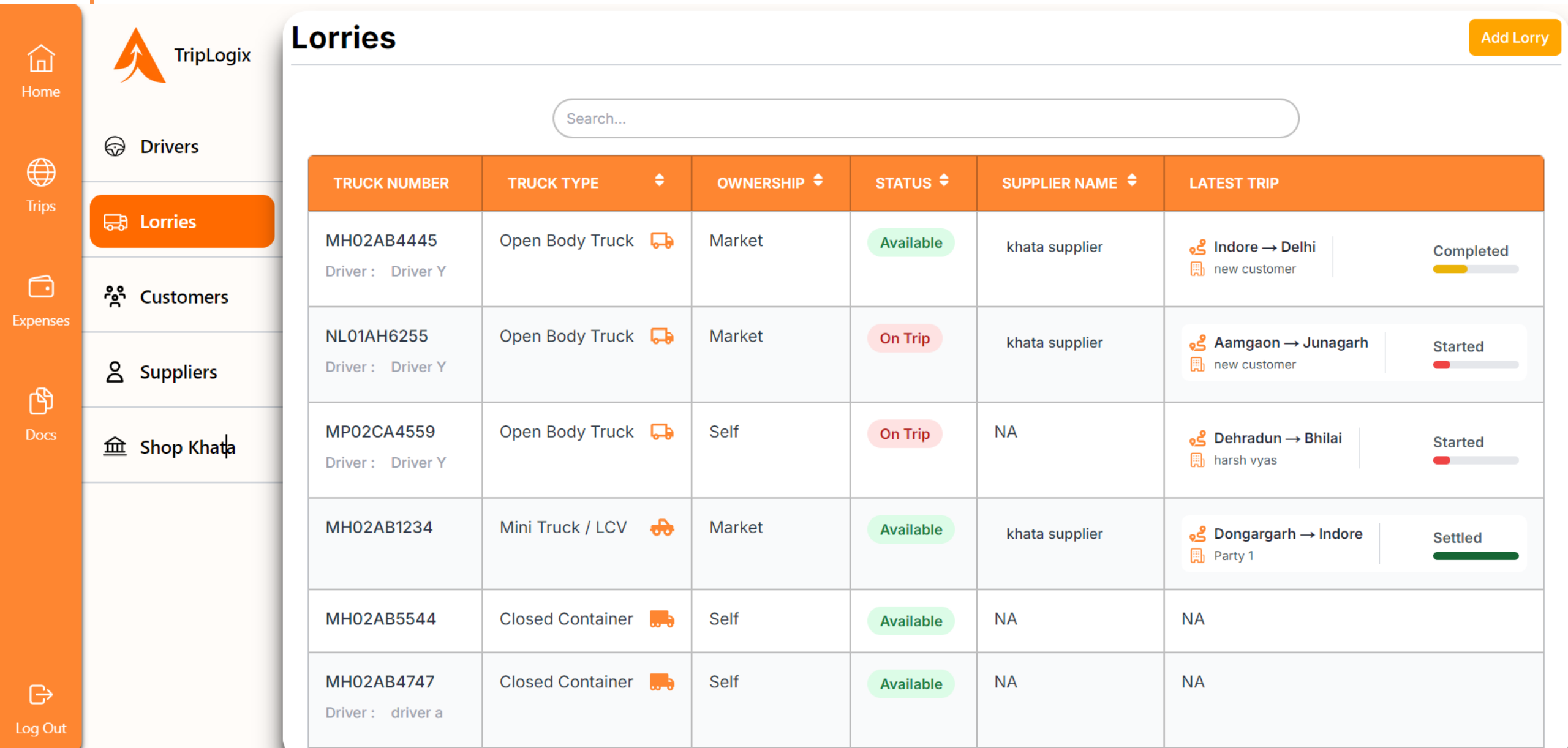




CUSTOMERS PAGE



LORRIES PAGE



DOCUMENTS PAGE

Documents

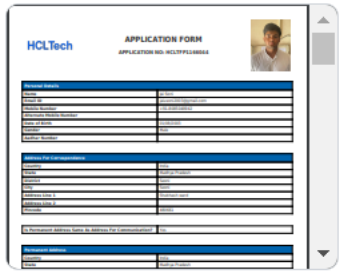
Trip Documents
5 files

Driver Documents
2 files

Lorry Documents
2 files

Company Documents
0 files

Recently Uploaded

Name	Type	Date Uploaded	Validity Date
	Other	19/10/2024	1/1/1970
E-Way Bill System.pdf	E-Way Bill	18/10/2024	8/8/2024