Regional Road Express

Project: Regional Road Express Dashboard & Analytics

Name: Saraswathi S

Abstract:

This project delivers a complete end-to-end dashboarding and analytics solution for *Regional Road Express*, a hypothetical regional logistics operator. The deliverables include:

- A responsive **React** web application with operational dashboards, routes overview, profit/loss analytics, ML-powered revenue prediction, and shortest-path routing.
- A **Power BI** report that visualizes financial KPIs (revenue, cost, profit/loss), operational metrics (route-wise performance, vehicle utilization), and time-series trends.
- A backend data pipeline and database storage (MySQL or equivalent) for ingestion, preprocessing, and serving metrics to the dashboard and Power BI.

The solution enables stakeholders to monitor financial health, identify underperforming routes, forecast revenue, and optimize routing decisions.

Introduction

The Regional Road Express Dashboard project is a full-stack logistics analytics system designed to manage road transport routes, calculate profit/loss, predict financial outcomes using ML logic, and visualize insights using Power BI and a React-based frontend.

This system integrates **FastAPI**, **MySQL**, **React**, **Machine Learning (formula-based prediction)**, and **Power BI** into one unified logistics intelligence platform.

Problem Statement

Key problems addressed by this project:

1. **Lack of consolidated financial view** — revenue, cost, and profit are scattered across systems and not easily comparable.
2. **Limited operational visibility** — difficulty identifying route-level performance issues or resource inefficiencies.
3. **No forecasting capability** — planning and budgeting lack predictive insights for revenue and demand.
4. **Routing inefficiencies** — suboptimal route selection leads to higher fuel and time costs.

Objectives:

- Build dashboards showing revenue, cost, and profit/loss at multiple granularities (company, region, route, date).
- Provide route-by-route analytics and visual drill-downs.
- Implement a machine learning model for short-term revenue prediction.
- Add a shortest-path feature to suggest efficient routing.

Proposed Solution (Overview)

The proposed system provides an end-to-end automated pipeline with the following components:

a) JSON Data Processing

100+ route records are cleaned, validated, cost-filled, and converted into structured analytics-ready format.

b) FastAPI Backend

Backend offers APIs for:

- Profit routes
- Loss routes

- Summary metrics (revenue, cost, profit, loss)
- ML-based profit prediction
- Shortest path using Dijkstra's Algorithm

c) MySQL Database

Stores complete route-level data, including:

- Revenue & cost components
- Profit, loss, and efficiency score
- JSON route metadata

d) React Frontend

An interactive dashboard with:

- Summary view
- Profit & loss routes table
- ML profit prediction
- Shortest path finder

e) Power BI Dashboard

Built on MySQL to provide:

- Total revenue/cost visuals
- Profit trends
- Region-wise comparison
- Vehicle performance

Use of JSON Dataset (100 Records)

A structured JSON dataset consisting of 100 route-level records is used as the primary raw data source. The workflow is as follows:

- **JSON Ingestion:** The dataset is loaded through a Python ETL script which validates fields (route_id, distance, revenue, expenses, vehicle details, etc.).
- **Transformation:** Each JSON object is parsed, cleaned, and converted into relational rows.
- **MySQL Storage:** Cleaned data is inserted into MySQL tables (trips, expenses, routes). Derived metrics such as revenue, cost, and profit are computed during insertion or via SQL views.
- **Power BI Integration:** Power BI Desktop connects directly to the MySQL tables, enabling creation of financial and operational dashboards with refreshed data.
- **React Frontend Integration:** REST APIs fetch processed MySQL data to display KPIs, tables, charts, maps, and ML forecast components.
- **ML Prediction:** The ML module reads the MySQL tables, processes the JSON-derived historical data, trains forecasting models, and stores predictions back to the database for visualization.

This ensures the JSON dataset flows consistently across all layers: storage → analytics → dashboards → prediction.

System Architecture Flow

JSON → Python Processing → MySQL → FastAPI → React UI → Power BI Analytics

This flow enables seamless data movement from raw ingestion to live dashboards.

Implementation Workflow

6.1 Data Preparation & JSON Processing (insert_routes.py)

1. Load 100 JSON records.
2. Auto-fill missing cost values.
3. Calculate:
    - Total cost
    - Revenue
    - Profit/loss
    - Time taken
    - Route efficiency score
4. Convert nested fields into route_json.
5. Insert into MySQL using UPSERT logic.

6.2 Backend API Development (FastAPI – main.py)

Backend exposes 5 major services:

1) Route Analytics APIs

- /routes/profit: Fetch profitable routes
- /routes/loss: Fetch loss routes

2) Summary Analytics API

- /summary: Total revenue, cost, profit, loss, route count

3) ML Profit Prediction

- /predict/profit: Predicts profit category based on costs & distance

4) Shortest Path Algorithm

- /shortest-path: Calculates shortest distance using Dijkstra

5) Root API

- /: Service health indicator

6.3 MySQL Database Storage

Stores structured route-level data including:

- Revenue, cost, profit/loss
- Cost components
- Time, speed
- JSON metadata

Ensures accuracy for both API and Power BI.

6.4 React Frontend Implementation

1) Summary Dashboard

Uses /summary to show:

- Total revenue
- Total cost
- Total profit
- Loss & profitable route counts

2) Profit/Loss Route Table

Uses /routes/profit & /routes/loss.

3) ML Prediction UI

Form → /predict/profit → Output: predicted profit & category.

4) Shortest Path UI

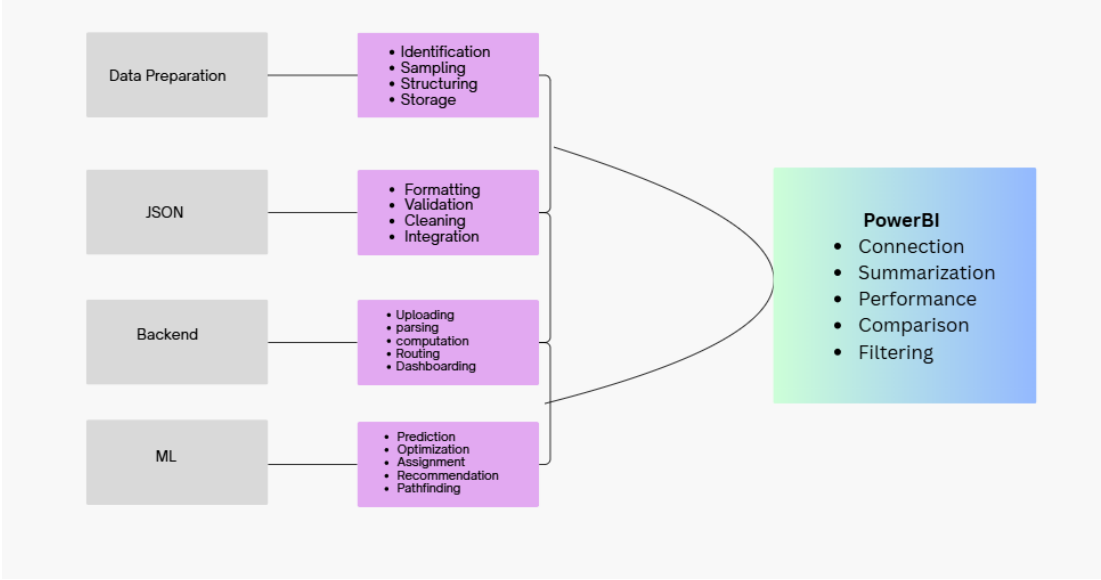
Inputs start & end → /shortest-path → Displays shortest route.

6.5 Power BI Dashboard

Connected directly to MySQL to generate:

- Revenue vs cost
- Profit trends
- Region performance comparison
- Vehicle type analysis

Block Diagram



SOURCE CODE:

1.MAIN.PY

```
from fastapi import FastAPI
import heapq
from fastapi.middleware.cors import CORSMiddleware
import mysql.connector
from pydantic import BaseModel


app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # allow all websites to access this API
    allow_methods=["*"],  # allow all HTTP methods (GET, POST, etc.)
    allow_headers=["*"]   # allow all headers
)

# ----------------------------
# DATABASE CONNECTION FUNCTION
# ----------------------------
def get_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Saraswathi99*",
        database="route_management"
    )
```

```
# ----------------------------
# 1️⃣ GET ALL LOSS ROUTES
# ----------------------------
@app.get("/routes/loss")
def get_loss_routes():
    conn = get_db()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT * FROM routes WHERE profit < 0")
    data = cursor.fetchall()

    cursor.close()
    conn.close()
    return {"total_loss_routes": len(data), "routes": data}

# ----------------------------
# 2️⃣ GET ALL PROFIT ROUTES
# ----------------------------
@app.get("/routes/profit")
def get_profit_routes():
    conn = get_db()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT * FROM routes WHERE profit >= 0")
    data = cursor.fetchall()

    cursor.close()
    conn.close()
    return {"total_profit_routes": len(data), "routes": data}
```

```
# ----------------------------
# 3️⃣ PROFIT SUMMARY (For Dashboard / PowerBI)
# ----------------------------
@app.get("/summary")
def summary():
    conn = get_db()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("""
        SELECT
            SUM(revenue) AS total_revenue,
            SUM(total_cost) AS total_cost,
            SUM(profit) AS total_profit,
            SUM(loss) AS total_loss,
            COUNT(*) AS total_routes,
            SUM(is_profitable) AS profitable_routes,
            SUM(CASE WHEN profit < 0 THEN 1 END) AS loss_routes
        FROM routes
    """)

    data = cursor.fetchone()

    cursor.close()
    conn.close()
    return data

# ----------------------------
# 4️⃣ SHORTEST PATH – DIJKSTRA
```

```
# ----------------------------

# Graph — YOU CAN MODIFY ROUTES LATER
graph = {
    "Chennai": {"Coimbatore": 214},
    "Mumbai": {"Jaipur": 211},
    "Pune": {"Ahmedabad": 242},
    "Durgapur": {"Bhubaneswar": 152},
    "Jaipur": {"Mumbai": 140},
    "Delhi": {"Ambala": 495},
    "Satara": {"Pune": 281},
    "Ambala": {"Panipat": 336},
    "Kolkata": {"Patna": 261}
}

@app.get("/shortest-path")
def shortest_path(start: str, end: str):
    queue = [(0, start, [])]
    visited = set()

    while queue:
        cost, node, path = heapq.heappop(queue)

        if node in visited:
            continue
        visited.add(node)

        path = path + [node]
```

```
        if node == end:
            return {"shortest_distance": cost, "path": path}

        for neighbor, weight in graph.get(node, {}).items():
            heapq.heappush(queue, (cost + weight, neighbor, path))

    return {"error": "No path found"}

# ----------------------------
# 5️⃣ ML – Profit Prediction Placeholder
# ----------------------------
class MLInput(BaseModel):
    distance_km: float
    fuel_cost: float
    toll_cost: float
    driver_cost: float
    repair_cost: float
    misc_cost: float

@app.post("/predict/profit")
def predict_profit(data: MLInput):
    total_cost = data.fuel_cost + data.toll_cost + data.driver_cost + data.repair_cost + data.misc_cost
    predicted_revenue = (data.distance_km * 45)  # simple formula now

    predicted_profit = predicted_revenue - total_cost

    return {
        "predicted_profit": predicted_profit,
        "profit_category": "Profitable" if predicted_profit >= 0 else "Loss"
```

```
    }


# ----------------------------
# API RUNNING STATUS
# ----------------------------
@app.get("/")
def home():
    return {"message": "Route Management API is running 🚚 📦 "}
```

2.INSER_ROUTES.PY

```
import mysql.connector
import json

# Connect to MySQL
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Saraswathi99*",
    database="route_management"
)
cursor = conn.cursor()

# Load JSON file
with open("regional_road_express.json", "r", encoding="utf-8") as file:
    data = json.load(file)

for route in data:

    # --- COST CALCULATION (AUTO FILL IF MISSING) ---
```

```
fuel = route.get("fuel_cost")
toll = route.get("toll_cost")
driver = route.get("driver_cost")
repair = route.get("repair_cost")
misc = route.get("misc_cost")

# AUTO default values if missing
if fuel is None:
    fuel = route["distance_km"] * 5   # Example fuel rate
if toll is None:
    toll = 100
if driver is None:
    driver = 500
if repair is None:
    repair = 200
if misc is None:
    misc = 150

total_cost = fuel + toll + driver + repair + misc
revenue = route.get("revenue", 0)

# --- PROFIT / LOSS ---
profit = revenue - total_cost

# FIXED LOSS: store ONLY positive loss value
loss = abs(profit) if profit < 0 else 0

is_profitable = 1 if profit >= 0 else 0
```

```
# --- TIME & SPEED ---
distance = route.get("distance_km", 1)

avg_speed = route.get("avg_speed", 40)       # default speed
time_hours = round(distance / avg_speed, 2)  # auto calculate

# --- EFFICIENCY SCORE ---
route_efficiency_score = int((profit / distance) * 10) if distance else 0

# --- JSON STORE ---
route_json = json.dumps({
    "route_points": route["route_points"],
    "vehicle": route["vehicle_type"],
    "profit": profit,
    "expenses": total_cost,
    "time_hours": time_hours,
    "avg_speed": avg_speed
})

# --- INSERT QUERY ---
cursor.execute("""
    INSERT INTO routes
    (route_id, region, distance_km, vehicle_type, revenue,
     total_cost, profit, loss, fuel_cost, toll_cost, driver_cost,
     repair_cost, misc_cost, time_hours, avg_speed,
     route_efficiency_score, route_date, route_json, is_profitable)
    VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
    ON DUPLICATE KEY UPDATE
        region = VALUES(region),
```

```
            distance_km = VALUES(distance_km),
            vehicle_type = VALUES(vehicle_type),
            revenue = VALUES(revenue),
            total_cost = VALUES(total_cost),
            profit = VALUES(profit),
            loss = VALUES(loss),
            fuel_cost = VALUES(fuel_cost),
            toll_cost = VALUES(toll_cost),
            driver_cost = VALUES(driver_cost),
            repair_cost = VALUES(repair_cost),
            misc_cost = VALUES(misc_cost),
            time_hours = VALUES(time_hours),
            avg_speed = VALUES(avg_speed),
            route_efficiency_score = VALUES(route_efficiency_score),
            route_date = VALUES(route_date),
            route_json = VALUES(route_json),
            is_profitable = VALUES(is_profitable)
        """, (
            route["route_id"],
            route["region"],
            distance,
            route["vehicle_type"],
            revenue,
            total_cost,
            profit,
            loss,
            fuel,
            toll,
            driver,
```

```
        repair,
        misc,
        time_hours,
        avg_speed,
        route_efficiency_score,
        route.get("route_date", None),
        route_json,
        is_profitable
    ))

conn.commit()
cursor.close()
conn.close()

print(f"All {len(data)} routes inserted successfully!")
```

3.FRONTEND

3.1 Summary.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer } from "recharts";

export default function Summary() {
  const [summary, setSummary] = useState({});

  useEffect(() => {
    axios.get("http://127.0.0.1:8000/summary")
      .then(res => setSummary(res.data))
      .catch(err => console.log(err));
```

```
  }, []);

  const chartData = [
    { name: "Revenue", value: summary.total_revenue || 0 },
    { name: "Cost", value: summary.total_cost || 0 },
    { name: "Profit", value: summary.total_profit || 0 },
    { name: "Loss", value: summary.total_loss || 0 }
  ];

  return (
    <div>
      <h2>Profit & Loss Summary</h2>
      <ResponsiveContainer width="100%" height={300}>
        <BarChart data={chartData}>
          <XAxis dataKey="name" />
          <YAxis />
          <Tooltip />
          <Bar dataKey="value" fill="#82ca9d" />
        </BarChart>
      </ResponsiveContainer>
    </div>
  );
}
```

3.2ROUTES TABLE.JS

```
import React, { useEffect, useState } from "react";
import axios from "axios";

export default function RoutesTable() {
  const [routes, setRoutes] = useState([]);
```

```
const [type, setType] = useState("profit"); // profit or loss


useEffect(() => {
  axios.get(`http://127.0.0.1:8000/routes/${type}`)
    .then(res => setRoutes(res.data.routes))
    .catch(err => console.log(err));
}, [type]);
return (
  <div>
    <h2>{type === "profit" ? "Profit" : "Loss"} Routes</h2>
    <button onClick={() => setType("profit")}>Profit Routes</button>
    <button onClick={() => setType("loss")}>Loss Routes</button>
    <table border="1" style={{ width: "100%", marginTop: "10px" }}>
      <thead>
        <tr>
          <th>Route ID</th>
          <th>Region</th>
          <th>Vehicle</th>
          <th>Profit</th>
        </tr>
      </thead>
      <tbody>
        {routes.map(r => (
          <tr key={r.route_id}>
            <td>{r.route_id}</td>
            <td>{r.region}</td>
            <td>{r.vehicle_type}</td>
            <td>{r.profit}</td>
          </tr>
```

```
        ))}
      </tbody>
    </table>
  </div>
 );
}
```

3.3 APP.JS

```
import React from "react";
import { BrowserRouter as Router, Route, Routes, Link } from "react-router-dom";
import Summary from "./components/Summary";
import RoutesTable from "./components/RoutesTable";
import ProfitPrediction from "./components/ProfitPrediction";
import ShortestPath from "./components/ShortestPath";

function App() {
 return (
  <Router>
   <div style={{ padding: "20px" }}>
    <h1>Regional Road Express Dashboard</h1>
    <nav>
     <Link to="/summary" style={{ marginRight: "10px" }}>Summary</Link>
     <Link to="/routes" style={{ marginRight: "10px" }}>Routes</Link>
     <Link to="/predict" style={{ marginRight: "10px" }}>ML Prediction</Link>
     <Link to="/shortest-path">Shortest Path</Link>
    </nav>
    <hr />
    <Routes>
     <Route path="/summary" element={<Summary />} />
     <Route path="/routes" element={<RoutesTable />} />
```

```
        <Route path="/predict" element={<ProfitPrediction />} />
        <Route path="/shortest-path" element={<ShortestPath />} />
      </Routes>
    </div>
  </Router>
 );
}
export default App;
```

3.4 Profit_prediction.js

```
import React, { useState } from "react";
import axios from "axios";

export default function ProfitPrediction() {
 const [form, setForm] = useState({
  distance_km: 0, fuel_cost: 0, toll_cost: 0, driver_cost: 0, repair_cost: 0, misc_cost: 0
 });
 const [result, setResult] = useState(null);

 const handleChange = (e) => setForm({...form, [e.target.name]: parseFloat(e.target.value)});
 const handleSubmit = (e) => {
  e.preventDefault();
  axios.post("http://127.0.0.1:8000/predict/profit", form)
   .then(res => setResult(res.data))
   .catch(err => console.log(err));
 }

 return (
  <div>
```

```
      <h2>ML Profit Prediction</h2>
      <form onSubmit={handleSubmit}>
        {Object.keys(form).map(key => (
          <div key={key}>
            <label>{key}: </label>
            <input type="number" name={key} onChange={handleChange} />
          </div>
        ))}
        <button type="submit">Predict</button>
      </form>
      {result && (
        <div style={{ marginTop: "20px" }}>
          <h3>Predicted Profit: {result.predicted_profit}</h3>
          <p>Category: {result.profit_category}</p>
        </div>
      )}
    </div>
  );
}
```

3.5 shortest_path.js

```
import React, { useState } from "react";
import axios from "axios";
export default function ShortestPath() {
  const [start, setStart] = useState("");
  const [end, setEnd] = useState("");
  const [result, setResult] = useState(null);

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
    axios.get(`http://127.0.0.1:8000/shortest-path?start=${start}&end=${end}`)
      .then(res => setResult(res.data))
      .catch(err => console.log(err));
  }
  return (
    <div>
      <h2>Shortest Path Finder</h2>
      <form onSubmit={handleSubmit}>
        <input placeholder="Start" value={start} onChange={(e)=>setStart(e.target.value)} />
        <input placeholder="End" value={end} onChange={(e)=>setEnd(e.target.value)} />
        <button type="submit">Find Path</button>
      </form>
      {result && result.shortest_distance !== undefined && (
        <div>
          <p>Distance: {result.shortest_distance}</p>
          <p>Path: {result.path.join(" → ")}</p>
        </div>
      )}
      {result && result.error && <p>{result.error}</p>}
    </div>
  );
}
```

Conclusion

The Regional Road Express Dashboard successfully integrates data engineering, backend APIs, machine learning logic, frontend visualization, and BI reporting into one powerful logistics analysis system. It improves operational efficiency and provides actionable insights for decision-makers.

This documentation demonstrates complete clarity in understanding, designing, and implementing a full-stack analytics system.