## **Project Overview**

FleetStat is a scalable Vehicle Tracking & Analytics System designed for aspiring data engineers and real-world fleet managers. This project helps build core Python, database, analytics, and deployment skills in a structured, scalable format. The system can evolve from a simple offline tracker to a real-time, GPS-integrated global product.

# **Requirements & Skills Needed**

#### Skills Needed:

- Python (Intermediate)
- SQLite or PostgreSQL (DB fundamentals)
- Pandas, Matplotlib/Seaborn (Data Analysis & Visualization)
- Streamlit or Dash (UI)
- FastAPI or Flask (APIs)
- Git & GitHub (Version Control)
- Deployment: Streamlit Cloud / Render

#### Tools:

- Python 3.10+
- SQLite or PostgreSQL
- Libraries: pandas, matplotlib, folium, geopy, streamlit, fastapi, scikit-learn, fpdf

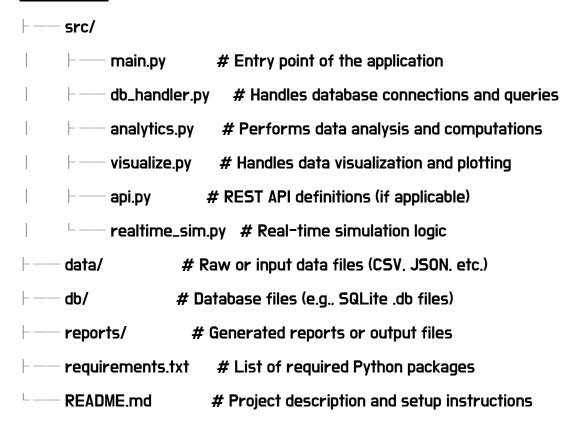
## **Project Structure & Modules**

### Core Modules:

- 1. db\_handler.py Handles database connections, schema, insert/update queries
- 2. main.py Entry point with CLI/Streamlit logic
- 3. analytics.py Performs data aggregation, stats
- 4. visualize.py Chart and map rendering (Matplotlib/Folium)

- 5. api.py REST API (FastAPI)
- 6. realtime\_sim.py Simulated GPS trip feed
- 7. report\_gen.py PDF reporting

## FleetStat/



## Scalability, Mobility & Efficiency Tips

#### Scalability:

- Use PostgreSQL or cloud DB (e.g., Supabase) for multi-user support
- Move from CSV/SQLite to REST API for integration with mobile/web
- Split microservices (API, analytics, dashboard)

#### Mobility:

- Turn dashboard into PWA or integrate with Flutter/React Native mobile frontend
- Real-time updates via WebSocket or long-polling APIs

#### Efficiency:

- Optimize queries with indexing
- Use async APIs (FastAPI)
- Cache analytics for repeated queries

#### User Friendliness:

- Clean UI/UX with filters and summary cards
- Allow data exports (Excel, CSV, PDF)
- Add notifications/reminders (email/API-based)

# **Deployment & Real-World Use Cases**

### Deployment:

- Host dashboard on Streamlit Cloud or Render
- REST API: PythonAnywhere, Railway, Render
- Schedule background tasks with cron or Celery

### Real-World Uses:

- Fleet companies tracking fuel usage
- Logistics firms analyzing delivery efficiency
- Government/public transport planners

#### **Revenue Generation Ideas**

- 1. B2B SaaS Model: Offer dashboard as a subscription
- 2. White-label solution for transport companies
- 3. Analytics reports for logistics optimization
- 4. Fleet health monitoring tools as paid add-on
- 5. Mobile app subscription + analytics API access

# 14-Day Efficient Build Timeline & Checklist

Day 1 (28 Jun 2025): Setup project structure, DB schema, environment

Day 2 (29 Jun 2025): Basic trip & vehicle input (CLI/Streamlit)

Day 3 (30 Jun 2025): Seed data + basic analytics with Pandas

Day 4 (01 Jul 2025): Matplotlib charts + dashboard with Streamlit

Day 5 (02 Jul 2025): Add GPS coordinates and calculate distances

Day 6 (03 Jul 2025): Visualize maps with Folium

Day 7 (04 Jul 2025): Geo analytics (heatmaps, clusters)

Day 8 (05 Jul 2025): Build REST APIs using FastAPI

Day 9 (06 Jul 2025): Simulate live GPS tracking

Day 10 (07 Jul 2025): Streamlit panel for real-time updates

Day 11 (08 Jul 2025): Add fuel prediction using scikit-learn

Day 12 (09 Jul 2025): Auto-generate trip reports (PDF/email)

Day 13 (10 Jul 2025): Deploy dashboard to Streamlit Cloud

Day 14 (11 Jul 2025): Polish, document, push to GitHub, share