FleetTax

Vahan Road Tax Tracker for Fleet Owners

Technical Documentation

Developer: Devansh Singh
GitHub: `DevanshSrajput`
Portfolio: `https://devanshsingh.dev`

Version 1.0.0
May 14, 2026

Contents

1 Introduction

1.1 Overview

FleetTax is a fully offline Android application built with Flutter that enables bus and truck fleet owners to efficiently track Vahan road tax payment deadlines. The app eliminates the need for manual tracking by automatically calculating tax expiry dates, sending daily push notifications starting 10 days before expiration, and providing direct integration with the Vahan portal for hassle-free tax payment.

1.2 Key Objectives

- Track tax deadlines for multiple vehicles with zero cloud dependency
- Automatically calculate expiry dates based on tax payment periods (monthly, quarterly, yearly)
- Send timely push notifications to prevent missed payments
- Provide an intuitive, offline-first user experience
- Use modern Flutter architecture with state management and local persistence

1.3 Target Audience

Fleet owners managing multiple commercial vehicles (buses and trucks) who need a reliable, offline solution for tax deadline management in Indian market.

2 Feature Overview

2.1 Core Features

2.1.1 Vehicle Management

- **Add Vehicle**: Register new vehicles with registration number, type (bus/truck), tax period, last payment date, and optional notes
- **Edit Vehicle**: Modify any vehicle details at any time
- **Delete Vehicle**: Remove vehicles with confirmation dialog
- **Vehicle Persistence**: All data stored locally using SQLite

2.1.2 Dashboard

- **Statistics Bar**: Display total vehicles, expired count, due soon count, and valid count
- **Vehicle Cards**: Color-coded status cards showing vehicle details and days remaining
- **Search**: Search vehicles by registration number
- **Filter**: Filter by status (expired/due soon/valid) or vehicle type (bus/truck)
- **Sort**: Sort by expiry soonest, registration number, or vehicle type

2.1.3 Tax Expiry Calculation

- **Auto-Calculation**: System automatically computes validity end date from last paid date + tax period
- **Status Tracking**: Shows days remaining until expiry
- **Three Period Types**:
  - **Monthly**: 1 month validity
  - **Quarterly**: 3 months validity
  - **Yearly**: 12 months validity

2.1.4 Payment Tracking

- **Mark as Paid**: Log new tax payments with payment date, period, and receipt reference
- **Auto-Recalculation**: Expiry date automatically recalculated upon marking as paid
- **Receipt Reference**: Store optional receipt numbers for payment verification

2.1.5 Notifications

- **Daily Alerts**: Push notifications fire every day from T-10 days until tax is marked paid
- **Background Execution**: Uses WorkManager to run even when app is closed
- **Persistent**: Reliably delivers notifications on Android 13+
- **Custom Messages**: Shows vehicle registration and days remaining

2.1.6 Vahan Integration

- **Quick-Launch**: One-tap button to open Vahan portal (vahan.parivahan.gov.in)
- **In-Browser Payment**: Users can complete tax payment directly from Vahan website

2.2 Status Color Coding

- **Expired**: Bold Red (#FF0044) — Tax payment overdue
- **Due Soon**: Bold Orange (#FF6B35) — Tax expiring within 10 days
- **Valid**: Bold Green (#00E676) — Tax payment current and valid

3 Technology Stack

3.1 Frontend Framework

Flutter 3.x: Cross-platform mobile development framework providing:

- Native-like performance
- Material Design UI components
- Hot reload for rapid development
- Compiled to Android native code

3.2 Programming Language

Dart 3.x: Modern, statically-typed language with:

- Strong type system for safety
- Garbage collection
- Async/await for clean asynchronous code

3.3 State Management

Provider: Lightweight, reactive state management providing:

- Model-View-Controller (MVC) pattern
- Scoped state
- Automatic widget rebuilds on state change

3.4 Local Database

SQLite via sqflite: Embedded relational database offering:

- Zero configuration
- On-device persistence
- Full ACID compliance
- Support for complex queries

3.5 Push Notifications

flutter_local_notifications: Cross-platform local notification system

3.6 Background Tasks

WorkManager: Reliable background job scheduling:

- Survives app restart
- Periodic task registration
- Guaranteed execution even when app is closed

3.7 Additional Libraries

- **intl**: Date/time formatting and localization
- **url_launcher**: Open URLs and launch external apps
- **path**: File path manipulation and resolution

4 Architecture

4.1 Project Structure

```
lib/
  main.dart                          # App entry point & initialization
  models/
      vehicle.dart                   # Vehicle data model & business logic
  db/
      database_helper.dart           # SQLite CRUD operations
  services/
      notification_service.dart      # Push notification management
      workmanager_service.dart       # Background task orchestration
  providers/
      vehicle_provider.dart          # State management (Provider pattern)
  screens/
      splash_screen.dart             # App initialization screen
      dashboard_screen.dart          # Main vehicle list view
      add_edit_screen.dart           # Vehicle form (add/edit)
      mark_paid_screen.dart          # Payment logging screen
  widgets/
       vehicle_card.dart             # Individual vehicle tile
       stats_bar.dart                # Statistics summary bar
       filter_chips.dart             # Filter and sort controls
```

4.2 Architectural Patterns

4.2.1 Model-View-ViewModel (MVVM)

The Provider pattern implements MVVM for clean separation of concerns:

- **Model**: Vehicle class encapsulates data and business logic
- **ViewModel**: VehicleProvider manages state and notifies UI of changes
- **View**: Screens and widgets consume state and dispatch actions

4.2.2 Singleton Pattern

DatabaseHelper uses singleton pattern to ensure single database connection throughout app lifecycle.

4.2.3 Service Locator

NotificationService and WorkmanagerService provide static methods for easy access across the app.

5 Database Schema

5.1 Vehicles Table

```
CREATE TABLE vehicles (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    reg TEXT NOT NULL,
    type TEXT NOT NULL,
    tax_period TEXT NOT NULL,
    last_paid TEXT NOT NULL,
    notes TEXT,
    receipt_ref TEXT,
    created_at TEXT NOT NULL
);
```

5.2 Column Descriptions

Column	Type	Description
id	INTEGER	Primary key, auto-incrementing unique identifier
reg	TEXT	Vehicle registration number (e.g., “DL01AB1234”)
type	TEXT	Vehicle type: “bus” or “truck”
tax_period	TEXT	Tax validity period: “monthly”, “quarterly”, or “yearly”
last_paid	TEXT	ISO 8601 date of last tax payment (e.g., “2025-05-01”)
notes	TEXT	Optional notes or comments about the vehicle
receipt_ref	TEXT	Optional receipt reference number for payment verification
created_at	TEXT	ISO 8601 timestamp of record creation

5.3 Queries Overview

5.3.1 Retrieve All Vehicles

```
SELECT * FROM vehicles ORDER BY last_paid DESC;
```

5.3.2 Find Vehicles Expiring Soon

```
SELECT * FROM vehicles
WHERE julianday('now') - julianday(last_paid) >=
    CASE tax_period
        WHEN 'monthly' THEN 30
        WHEN 'quarterly' THEN 90
        WHEN 'yearly' THEN 360
    END - 10;
```

6 Core Components

6.1 Vehicle Model

The Vehicle class encapsulates vehicle data and expiry calculation logic:

```
class Vehicle {
  final int? id;
  final String reg;
  final String type;           // 'bus' or 'truck'
  final String taxPeriod;      // 'monthly', 'quarterly', 'yearly'
  final String lastPaid;       // 'YYYY-MM-DD'
  final String? notes;
  final String? receiptRef;

  DateTime get expiryDate {
    final d = DateTime.parse(lastPaid);
    if (taxPeriod == 'monthly')
      return DateTime(d.year, d.month + 1, d.day);
    if (taxPeriod == 'quarterly')
      return DateTime(d.year, d.month + 3, d.day);
    return DateTime(d.year + 1, d.month, d.day);
  }

  int get daysLeft =>
      expiryDate.difference(DateTime.now()).inDays;

  String get status {
    if (daysLeft < 0) return 'expired';
    if (daysLeft <= 10) return 'soon';
    return 'valid';
  }
}
```

6.2 VehicleProvider

Manages the state of all vehicles and coordinates operations:

- **load()**: Fetch all vehicles from database
- **add()**: Create new vehicle record
- **update()**: Modify existing vehicle
- **delete()**: Remove vehicle from database
- **markPaid()**: Log payment and reschedule notifications
- **getFiltered()**: Return filtered vehicle list based on criteria

6.3 DatabaseHelper

Singleton database manager providing SQLite operations:

- **insert()**: Add new vehicle

- **update()**: Modify vehicle details
- **delete()**: Remove vehicle
- **getAll()**: Retrieve all vehicles
- **database**: Lazy-loaded database connection property

6.4 NotificationService

Handles local push notifications:

- **init()**: Initialize flutter_local_notifications plugin
- **showAlert()**: Display notification for specific vehicle
- **cancelAll()**: Clear all pending notifications

6.5 WorkManager Service

Orchestrates background daily tax expiry checks:

- **callbackDispatcher()**: Entry point for background execution
- **initWorkManager()**: Register periodic daily task
- Executes every 24 hours to check and notify about due taxes

7 Installation Guide

7.1 Prerequisites

- **Flutter SDK**: Version 3.11.5 or later (latest stable recommended)
- **Android Studio**: Latest version with Android SDK 33+ installed
- **Android Device/Emulator**: API 33 (Android 13) or higher recommended
- **Git**: For cloning the repository

7.2 Step 1: Clone Repository

```
git clone https://github.com/DevanshSrajput/FleetTax.git
cd FleetTax
```

7.3 Step 2: Install Flutter Dependencies

```
flutter pub get
```

7.4 Step 3: Verify Setup

```
flutter doctor
```

All items must show checkmark. Fix any reported issues before proceeding.

7.5 Step 4: Run on Emulator

```
# List available devices
flutter devices

# Run on specific device
flutter run -d <device-id>
```

7.6 Step 5: Build Release APK

```
flutter build apk --release
```

Output file: `build/app/outputs/flutter-apk/app-release.apk`

8 Building for Production

8.1 Android Configuration

Update `android/app/src/main/AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

8.2 Build Types

8.2.1 Debug Build

```
flutter run
```

8.2.2 Release Build

```
flutter build apk --release
```

8.2.3 App Bundle (for Play Store)

```
flutter build appbundle --release
```

9 Notification System

9.1 Architecture

The notification system uses a two-layer approach:

1. **Local Notifications**: flutter_local_notifications plugin for showing alerts on device
2. **Background Task**: WorkManager for reliable daily execution

9.2 Flow Diagram

```
App Launch
    ↓
NotificationService.init()
    ↓
initWorkManager()
    ↓
Register PeriodicTask (every 24 hours)
    ↓
Daily Execution:
    → Load all vehicles from DB
    → Check each vehicle's daysLeft
    → If daysLeft <= 10: showAlert()
    → Persist notifications across app restarts
```

9.3 Notification Permissions

9.3.1 Runtime Permission (Android 13+)

In `main.dart`:

```
await Permission.notification.request();
```

9.3.2 Manufacturer-Specific Settings

On Xiaomi, OnePlus, Oppo, Samsung devices:

- Navigate to Settings → Battery/Power
- Find FleetTax in battery optimization list
- Select “Allow background activity”

10 Troubleshooting

10.1 Common Issues

10.1.1 Notifications Not Showing

Cause: Permission not granted at runtime.
Solution:

1. Go to Settings → Apps → FleetTax
2. Enable "Notifications" permission
3. Restart the app

10.1.2 WorkManager Not Firing

Cause: Android power management killing background tasks.
Solution:

1. Check battery optimization settings (see Manufacturer-Specific Settings above)
2. Test manually by adding temporary button:

```
ElevatedButton(
  onPressed: () => Workmanager().registerOneOffTask(
    'test_now', 'daily_tax_check'),
  child: Text('Test: Trigger notifications'),
)
```

10.1.3 SDK Not Found

Cause: Flutter or Android SDK path not configured.
Solution:

```
flutter doctor
# Follow all recommendations
flutter doctor --android-licenses
```

10.1.4 MissingPluginException

Cause: Plugin native code not linked.
Solution:

```
flutter clean
flutter pub get
flutter run
```

10.1.5 App Crashes on Launch

Cause: Likely missing `await` in async initialization.
Solution:

```
flutter logs
# Review full error trace and check main.dart
```

11 API Reference

11.1 Vehicle Class

11.1.1 Constructors

```
Vehicle({
  int? id,
  required String reg,
  required String type,           // 'bus' | 'truck'
  required String taxPeriod,      // 'monthly' | 'quarterly' | 'yearly'
  required String lastPaid,       // ISO 8601 date
  String? notes,
  String? receiptRef,
})
```

11.1.2 Getters

- `DateTime expiryDate`: Calculated tax expiry date
- `int daysLeft`: Days until expiry (negative if expired)
- `String status`: One of “expired”, “soon”, “valid”

11.1.3 Methods

- `Map<String, dynamic> toMap()`: Convert to database-friendly map
- `Vehicle.fromMap()`: Factory constructor from database map

11.2 VehicleProvider Class

11.2.1 Properties

- `List<Vehicle> vehicles`: Current list of all vehicles
- `bool isLoading`: Loading state indicator

11.2.2 Methods

- `Future<void> load()`: Load all vehicles from database
- `Future<void> add(Vehicle v)`: Insert new vehicle
- `Future<void> update(Vehicle v)`: Update existing vehicle
- `Future<void> delete(int id)`: Delete vehicle by ID
- `Future<void> markPaid(int id, String date, String period)`: Log payment
- `List<Vehicle> getFiltered()`: Return filtered list based on current filters

11.3 DatabaseHelper Class

11.3.1 Static Methods

- `Future<Database> get database`: Get or create database connection
- `Future<int> insert(Vehicle v)`: Add vehicle, returns ID
- `Future<int> update(Vehicle v)`: Update vehicle, returns rows affected
- `Future<int> delete(int id)`: Delete vehicle, returns rows affected
- `Future<List<Vehicle>> getAll()`: Get all vehicles from database

12 Contributing

12.1 Getting Started

1. Fork the repository on GitHub
2. Clone your fork: `git clone https://github.com/YOUR-USERNAME/FleetTax.git`
3. Create feature branch: `git checkout -b feature/your-feature-name`
4. Make your changes following the code style
5. Commit with descriptive message: `git commit -m 'feat: add feature description'`
6. Push to branch: `git push origin feature/your-feature-name`
7. Open Pull Request with detailed description

12.2 Code Style Guidelines

- Use consistent formatting (Flutter recommends: 2-space indentation)
- Follow Dart naming conventions (camelCase for variables, PascalCase for classes)
- Add comments for complex logic
- Write self-documenting code when possible
- Run `flutter analyze` before committing

13 Roadmap

13.1 Version 1.0 (Current)

Vehicle CRUD operations

Tax expiry calculation (monthly/quarterly/yearly)

Dashboard with color-coded status

Filter, search, and sort functionality

Mark payment with receipt reference

Daily push notifications via WorkManager

Vahan portal quick-launch

Fully offline operation

Insurance expiry tracking

Fitness certificate (FC) tracking

Google Drive backup and restore

PDF receipt capture per payment

Multi-owner and driver assignment

Play Store release

Cloud synchronization (optional)

14 License

FleetTax is released under the MIT License. See LICENSE file in repository for full text.

```
MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

15 About the Developer

Devansh Singh is a software developer specializing in cross-platform mobile development with Flutter and Dart.

- **GitHub**: `https://github.com/DevanshSrajput`
- **Portfolio**: `https://devanshsingh.dev`
- **Project Repository**: `https://github.com/DevanshSrajput/FleetTax`

This documentation was automatically generated for FleetTax v1.0.0.