🚆 Distributed Train Ticket Booking System

A fault-tolerant, strongly consistent booking system built on gRPC, Raft and SQLite (async).

Overview

This project implements a **distributed train ticket booking system** where:

✔ Highly Consistent

All booking operations use a **Raft-based replication layer** to guarantee:

- Strong consistency
- Log replication across nodes
- Leader-based command execution
- Fault tolerance (nodes can fail & recover)

✔ Fully Asynchronous

All database operations use **`aiosqlite`**, allowing:

- Non-blocking operations
- Concurrency-safe transactions
- WAL-based high-performance writes

✔ Real Features

The system provides:

- **User Registration & Login**
- **Session Management**
- **Train & Service Management**
- **Seat Booking (transactional)**
- **Payment Confirmation**
- **Full Raft replication across 3 nodes**

✔ Multiple Interfaces

- **CLI Client** (Python interactive menu)
- **Flask Web App** (simple frontend)
- **Distributed gRPC backend** (3-node cluster)

Architecture

```
                          ┌──────────────────────────────┐
                          │      Client (Browser/UI)      │
                          └──────────────┬───────────────┘
                                         │  HTTP/REST
                                         ▼
                         ┌─────────────────────────────────────────┐
                         │          Flask Web Application          │
                         │-----------------------------------------│
                         │ - Accepts client commands               │
                         │ - Sends ALL requests → Raft Leader      │
                         │ - Returns results back to client        │
                         └──────────────┬──────────────────────────┘
                                        │  gRPC
                                        ▼
                          ┌───────────────────────────────────────┐
                          │              Raft Leader              │
                          │---------------------------------------│
                          │ - Handles writes (register, booking)  │
                          │ - Reads from its own DB               │
                          │ - Sends LLM queries to Local LLM      │
                          │ - Replicates logs to followers        │
                          └──────────────┬───────────────┬────────┘
                                         │               │
                                         │               │
                                         │               ▼
                                         │    ┌──────────────────────────┐
                                         │    │     Local LLM Server     │
                                         │    │ - Receives query from    │
                                         │    │   Raft Leader            │
                                         │    │ - Sends generated answer │
                                         │    └──────────────────────────┘
                                         │
                                         │
                 ┌───────────────────────┴──────────────────────────┐
                 │                                                  │
                 ▼                                                  ▼
  ┌──────────────────────────┐                      ┌──────────────────────────┐
  │     Follower Node 1      │                      │     Follower Node 2      │
  │--------------------------│                      │--------------------------│
  │ - Receives log entries   │                      │ - Receives log entries   │
  │ - Applies committed logs │                      │ - Applies committed logs │
  │ - Local SQLite DB        │                      │ - Local SQLite DB        │
  └──────────────────────────┘                      └──────────────────────────┘
```

Each raft node maintains:

- A local SQLite database
- A Raft log file

▯ Setup Instructions

⚙ 1. Install Dependencies

Windows

```
.\setup.ps1
```

Linux / Mac

```
./setup.sh
```

□ 2. Compile Protocol Buffers

Run this **after any change** to `train_booking.proto`.

Windows

```
.\compile_proto.ps1
```

Linux / Mac

```
./compile_proto.sh
```

□ Starting the Server Cluster

Open **three terminals**, one for each Raft node.

Terminal 1

```
python src/server/main.py 1
```

Terminal 2

```
python src/server/main.py 2
```

Terminal 3

```
python src/server/main.py 3
```

Each node will automatically:

- Participate in elections
- Elect a leader
- Replicate logs
- Apply committed commands to its local DB

□□ Running the Client

□ Flask Web Application

Web-based frontend:

```
python web_app/app.py
```

□ Database Tools

To view SQLite DB contents:

□ **Online Viewer:** https://sqliteviewer.app/

Each node maintains its own database file:

```
 Node1TicketBooking.db
Node2TicketBooking.db
Node3TicketBooking.db
```

□ Key Features of the Project

□ Distributed Consensus (Raft)

- Leader election
- Log replication
- Fault recovery
- Heartbeats
- Commit & apply stages

□ Booking Engine

- Transactional seat reservation
- Payment confirmation
- Unique session token enforcement
- Automatic rollback on failure

□ Async SQLite

- WAL mode for performance
- Locks managed using `asyncio.Lock`
- Fully non-blocking server