Algeria Train Tracker

Technical Documentation

Yakoubi Ahmed Moncef
Alem Rayane
Harnane Sami Youcef
Guettaya Abderaouf

July 1, 2025

A full-stack solution for real-time train tracking, smart ticketing, and AI-assisted trip planning.

Contents

1 Problem Statement

Train users in Algeria face persistent issues due to the lack of a real-time, intelligent rail tracking system. Current systems do not provide dynamic updates on train schedules or estimated arrivals, resulting in poor travel planning, missed trains, and loss of prepaid tickets.

Moreover, there is no integrated solution to help passengers determine the optimal transport method from their current GPS location to the nearest train station based on time constraints. This gap leads to inefficient station access and passenger dissatisfaction.

2 Proposed Solution

We propose a full-stack web application that integrates real-time train tracking, interactive mapping, online ticket purchasing, and AI-assisted trip planning. This solution enhances usability, reduces missed departures, and optimizes the passenger experience through automation, geolocation, and adaptive scheduling.

3 Project Description

The proposed system is a full-stack web application designed to enhance train travel in Algeria. It provides real-time train tracking simulations, intelligent ticket purchasing decisions powered by AI, and secure online payment processing.

3.1 System Architecture

- **Frontend**: Built with HTML, CSS, and JavaScript (`script.js`, `train3d.js`, `payment.js`). It offers a responsive, mobile-first user interface for mapping, ticketing, and payments.
- **Backend**: A simple Node.js server (`server.js`) to serve the static frontend files.
- **Backend-as-a-Service (BaaS)**: Supabase is used for secure payment processing, authentication, and database management through Edge Functions.

4 Core Features

1. **Train Tracking Simulation**: The application simulates train movement using animated tracking points along predefined routes based on static timetables.
2. **AI-Powered Time Estimation**: An integrated Gemini API assistant estimates whether a user can reach the station on time and provides color-coded recommendations (Green: On time, Red: Late, Orange: Borderline).
3. **Dynamic Ticket Price Calculation**: The system instantly calculates ticket prices based on the selected line and the number of stations between departure and arrival.
   - **Algiers Line:** 30 DA base + 15 DA per station.
   - **Oran Line:** 400 DA base + 100 DA per station.
4. **Secure Online Payment System**: A real-time payment system using Supabase validates cards, deducts payments, and ensures secure financial transactions through Edge Functions.

5 Database Modeling

The payment system's database is managed by Supabase. The core of this system is the `cards` table, which is designed to securely store user payment information and manage balances for transactions.

5.1 Table Schema: `cards`

This table stores card details and the balance for each registered card. It is initialized and managed through a SQL migration file. The schema is defined as follows:

```
CREATE TABLE IF NOT EXISTS cards (
    id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    card_id VARCHAR(20) UNIQUE NOT NULL,
    ccv VARCHAR(3) NOT NULL,
    expire_month INT NOT NULL,
    expire_year INT NOT NULL,
    phone_number VARCHAR(15),
    balance DECIMAL(10, 2) DEFAULT 10000.00,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

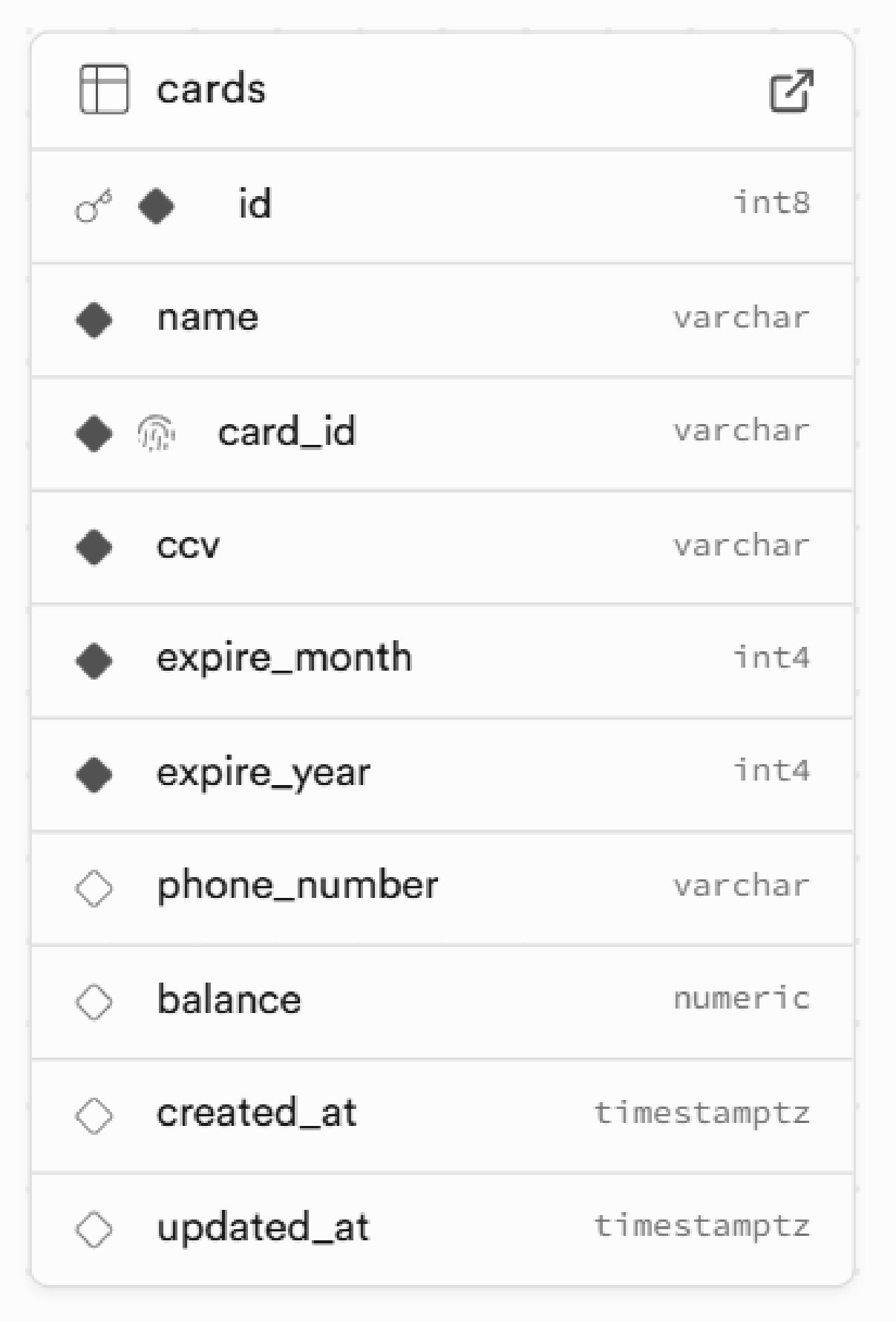

Figure 1: The schema for the `cards` table as viewed in the Supabase Schema Visualizer.

id int8	name varchar	card_id varchar	ccv varchar	expire_mo... i...	expire_year int4	phone_number varchar	bala... num...	created_at timestamptz	updated_at timestamptz
1	ADMIN ACCOUNT	ADMIN001	000	12	2030	+213000000000	3475.00	2025-07-01 01:48:37.428012+0(	2025-07-01 10:17:17.88867+00
2	Ahmed Benali	1234567890123456	123	12	2026	+213555123456	3705.00	2025-07-01 01:48:37.428012+0(	2025-07-01 04:09:19.061989+0
3	Fatima Khelil	2345678901234567	456	6	2027	+213555234567	7500.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
4	Mohamed Saidi	3456789012345678	789	9	2025	+213555345678	3200.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
5	Amina Boudiaf	4567890123456789	321	3	2026	+213666789012	8500.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
6	Youcef Mammeri	5678901234567890	654	11	2027	+213777890123	4200.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
7	Leila Cherif	6789012345678901	987	8	2025	+213888901234	6800.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
8	Karim Belkacem	7890123456789012	147	5	2026	+213999012345	3500.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
9	Nadia Hamidi	8901234567890123	258	10	2027	+213555567890	9200.00	2025-07-01 01:48:37.428012+0(	2025-07-01 01:48:37.428012+00
10	aa	111111111111111	111	11	2026	+21311111111	7820.00	2025-07-01 05:27:35.514727+0(	2025-07-01 10:17:17.846901+00

Figure 2: Sample data in the `cards` table, showing user accounts and the admin account.

6 System Architecture and Design (UML)

The following diagrams illustrate the system's architecture, user interactions, and internal logic.

6.1 Use Case Diagram

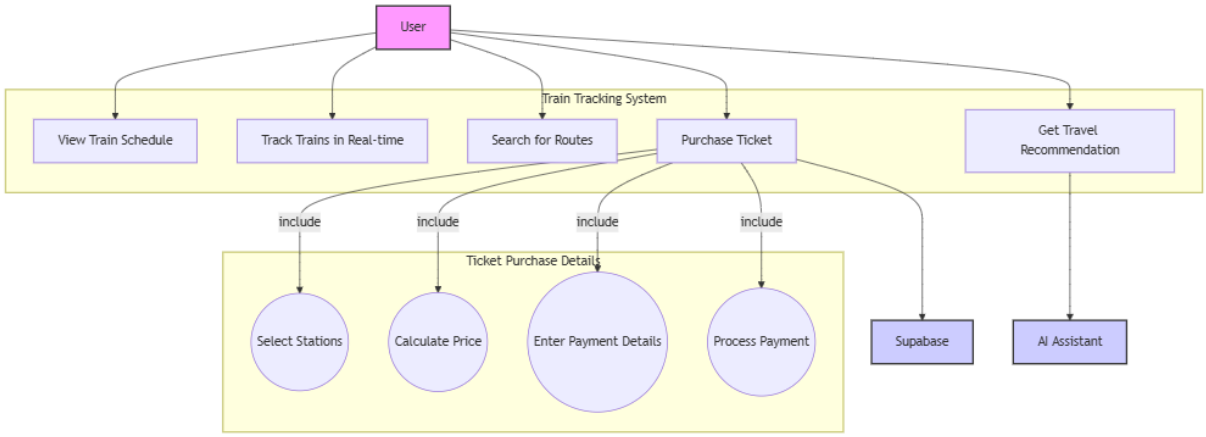

Figure 3: Use Case Diagram showing user interactions with the system.

Explanation: This diagram illustrates the high-level functionalities from the user's perspective. It shows the primary actor (User) and the key actions they can perform, such as purchasing a ticket or getting a travel recommendation. It also identifies the external systems involved, like Supabase for payments and the AI Assistant for recommendations.

6.2 Component Diagram

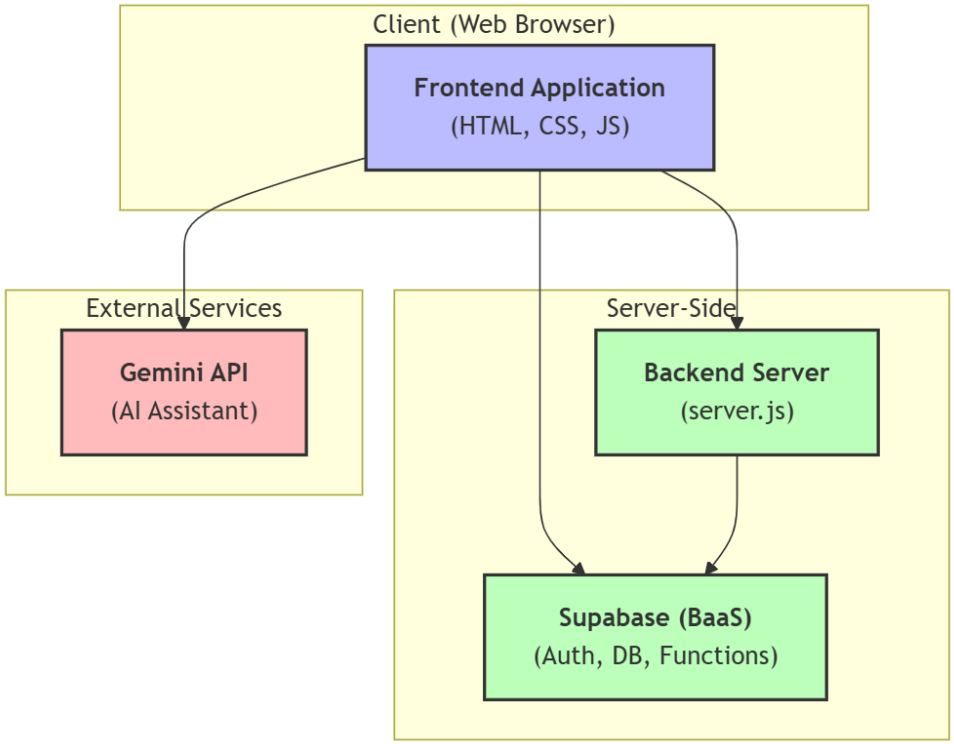

Figure 4: Component Diagram illustrating the high-level software architecture.

Explanation: This diagram shows the main software components and their dependencies. It separates the system into three logical parts: the **Client** (the user's browser), the **Server-Side** (your Node.js server and Supabase), and **External Services** (the Gemini API). This helps visualize how different parts of the application communicate.

6.3 Class Diagram

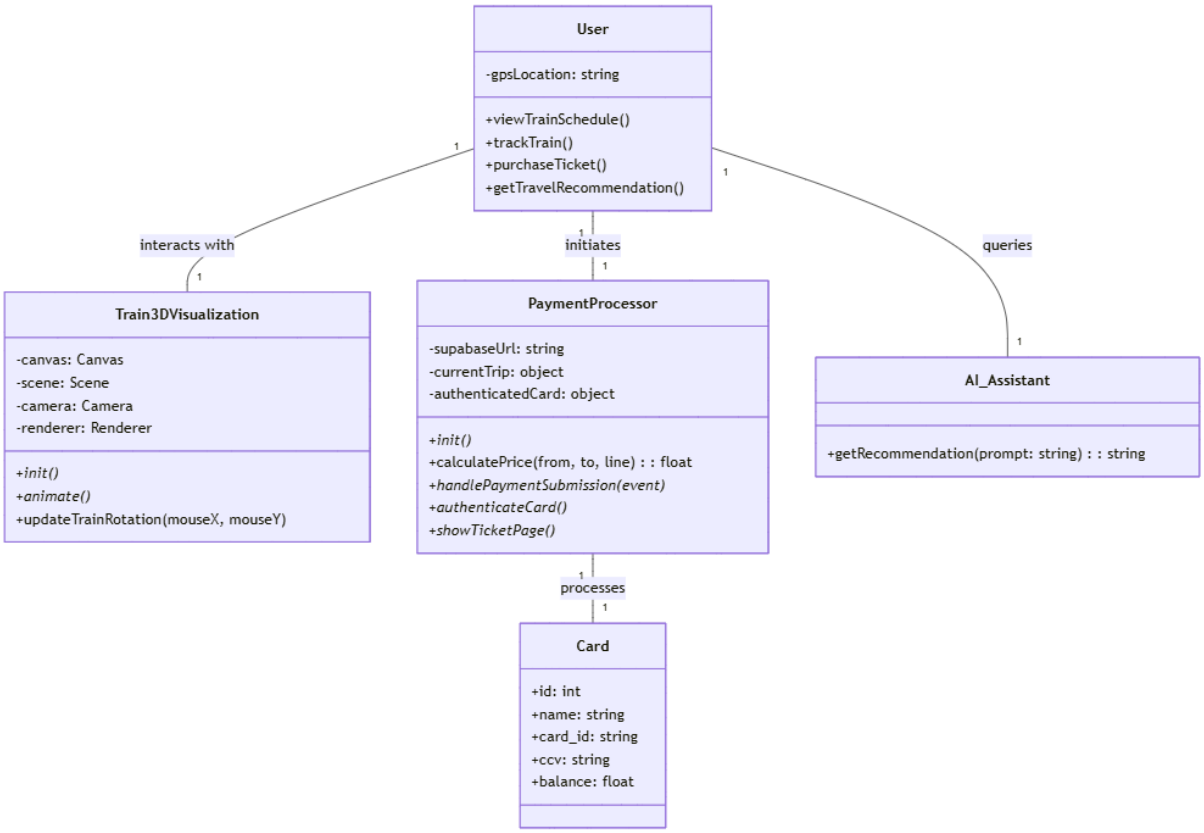

Figure 5: Class Diagram detailing the main software classes and their relationships.

Explanation: The Class Diagram provides a static view of the system's structure. It outlines the main classes, such as `PaymentProcessor` and `Train3DVisualization`, detailing their attributes (data) and methods (functions). The connections show how these classes are related and interact with one another to form the complete application.

6.4 Sequence Diagram

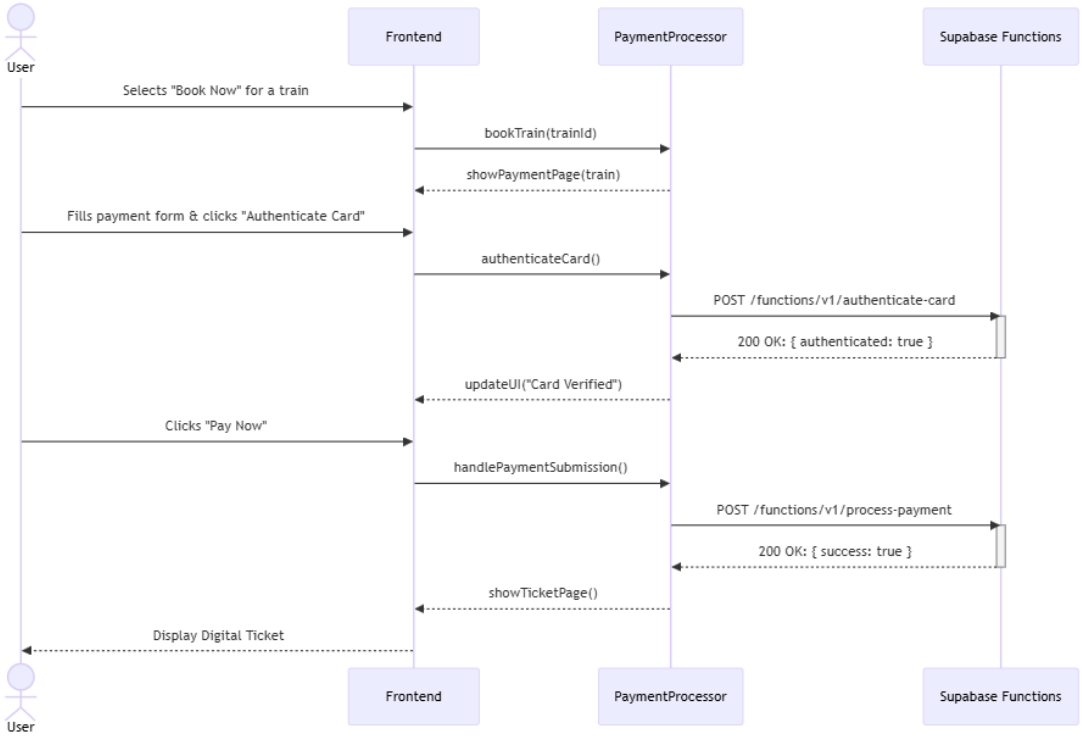

Figure 6: Sequence Diagram illustrating the ticket purchase workflow.

Explanation: This diagram shows the step-by-step flow of interactions for the "Purchase Ticket" scenario. It maps out the sequence of calls between the **User**, the **Frontend**, the **PaymentProcessor**, and the **Supabase Functions**. This is crucial for understanding the logic and timing of the payment process, from authenticating the card to processing the final payment.