Cab Booking Platform— Python Stack

Project Summary

A full-stack ride-hailing application with a Python backend (FastAPI) and a React/Next.js frontend (UI using Tailwind). The backend handles authentication verification, ride matching, real-time updates, payment webhooks, and background jobs. The system is designed to be deployable (Docker), scalable (Postgres + Redis + Celery), and production-ready.

Primary Responsibilities

- **Frontend:** Next.js + React + Tailwind (modern UI / SSR benefits)
- **Backend:** FastAPI (Python) — REST + WebSocket endpoints
- **Auth:** Clerk (frontend) or Supabase Auth with Python verification — tokens verified in FastAPI
- **Database:** PostgreSQL (primary), with PostGIS extension for geospatial queries (nearby drivers). Optionally MongoDB for flexible documents.
- **Real-time:** python-socketio (or FastAPI WebSockets) backed by Redis for message brokering
- **Background tasks:** Celery + Redis (or RQ) for sending receipts, heavy jobs, reconciliation
- **Payments:** Stripe (stripe-python) with server-side webhooks
- **Geolocation & Maps:** Google Maps JS on frontend + google-maps-services-python on backend for distance/time/fare calculation
- **Caching/Store:** Redis for ephemeral driver locations, rate-limiting, and presence

Key Features (Python Tech)

- **User Authentication:** Clerk or Supabase Auth on frontend; FastAPI middleware verifies tokens. Role-based access (rider/driver) stored in DB.
- **Ride Booking System:** FastAPI endpoints to create ride requests, estimate fare (distance/time via Google Maps APIs), store requests in Postgres.
- **Driver Dashboard (Optional):** Driver connects via Socket.IO; server pushes ride requests; drivers accept/decline.
- **Google Maps Integration:** Frontend uses Google Maps JS; backend uses Google Maps Python client for distance matrix/directions.
- **Real-Time Updates:** python-socketio or FastAPI WebSockets + Redis pub/sub to push driver location, ETA, status changes.
- **Payment Gateway:** stripe-python to create PaymentIntents; FastAPI webhook endpoint to confirm payments and update ride status.
- **Ride History & Receipts:** Rides stored in Postgres; Celery sends receipt emails and generates PDF receipts (wkhtmltopdf or WeasyPrint).
- **Ratings & Reviews:** Normalized tables; FastAPI endpoints to post/get reviews with aggregation queries.
- **Admin Dashboard (Optional):** Next.js admin UI talking to FastAPI admin endpoints with RBAC.

Architecture

1. **Frontend:** Next.js
2. **Backend:** FastAPI (REST + WebSockets)
3. **Infrastructure:** Postgres + PostGIS, Redis, Celery workers, Stripe, Google Maps APIs

Deployment

- Frontend: Vercel
- Backend: Container host (Render, Railway, AWS ECS, DigitalOcean App)
- Database: Managed Postgres (Supabase/Postgres)
- Redis: Managed service
- Celery workers: Separate containers

Data Models

- **users:** id, name, email, phone, role (rider/driver), profile_pic, rating_avg
- **drivers:** user_id FK, vehicle_info, verified (boolean), current_location (lon/lat point), status (active/offline)
- **rides:** id, rider_id, driver_id, pickup_point (geom), drop_point (geom), status, fare_estimate, fare_actual, distance_meters, duration_secs, created_at
- **payments:** id, ride_id, stripe_payment_intent_id, amount, status
- **reviews:** id, ride_id, rater_id, rated_id, rating, comment

Use PostGIS types for pickup/drop and create geospatial indices for fast nearby-driver queries.

Example API Endpoints (FastAPI)

- **POST /api/auth/session** — verify token from Clerk/Supabase
- **POST /api/rides/request** — create ride request (returns ride id)
- **GET /api/rides/estimate?origin=...&dest=...** — return fare estimate (calls Google Distance Matrix)
- **GET /api/rides/{id}/status** — poll or subscribe via WebSocket
- **POST /api/webhooks/stripe** — Stripe webhook for payment events
- **ws /ws/driver/{driver_id}** — driver socket to receive requests and send location
- **ws /ws/rider/{rider_id}** — rider socket to watch driver

Resources

- **Tutorials:** FastAPI official docs, Next.js docs, Tailwind CSS docs
- **Libraries:** python-socketio, stripe-python, google-maps-services-python, Celery, SQLAlchemy
- **Example Repos:** FastAPI + Next.js starter templates, Stripe webhook examples, Socket.IO integration samples

Backend — key files

backend/requirements.txt

- fastapi
- uvicorn[standard]
- SQLAlchemy
- psycopg2-binary
- alembic
- pydantic
- python-dotenv
- geoalchemy2
- geojson
- python-socketio[asyncio_client]
- python-socketio
- redis
- celery[redis]
- stripe
- googlemaps
- httpx
- python-multipart
- jinja2
- weasyprint

Backend Structure

```
cab-booking-platform/
└── backend/
    ├── Dockerfile
    ├── requirements.txt
    ├── app/
    │   ├── main.py              # FastAPI app entrypoint
    │   ├── core/
    │   │   └── config.py        # Environment config via Pydantic
    │   ├── db/
    │   │   ├── session.py       # SQLAlchemy session setup
    │   │   └── base.py          # Declarative base for models
    │   ├── models/
    │   │   └── models.py        # SQLAlchemy ORM models
    │   ├── schemas/
    │   │   └── schemas.py       # Pydantic request/response schemas
    │   ├── api/
    │   │   ├── deps.py          # Auth/token verification
    │   │   └── routes/
    │   │       ├── auth.py      # (Optional) Auth endpoints
    │   │       ├── rides.py     # Ride request, estimate, match
    │   │       └── payments.py  # Stripe webhook handling
    │   ├── sockets/
    │   │   └── manager.py       # Socket.IO server for live updates
    │   ├── workers/
    │   │   └── tasks.py         # Celery background jobs
    │   └── utils/
    │       └── geo.py           # Geospatial helpers (WKT, distance)
    └── alembic/                 # (Optional) DB migrations
```

Project Structure

```
cab-booking-platform/
├── backend/
│   ├── Dockerfile
│   ├── requirements.txt
│   ├── app/
│   │   ├── main.py
│   │   ├── core/
│   │   │   └── config.py
│   │   ├── db/
│   │   │   ├── session.py
│   │   │   └── base.py
│   │   ├── models/
│   │   │   └── models.py
│   │   ├── schemas/
│   │   │   └── schemas.py
│   │   ├── api/
│   │   │   ├── deps.py
│   │   │   └── routes/
│   │   │   ├── auth.py
│   │   │   └── rides.py
│   │   ├── sockets/
│   │   │   └── manager.py
│   │   ├── workers/
│   │   │   └── tasks.py
│   │   └── utils/
│   │   └── geo.py
│   └── alembic/ (optional)
├── frontend/
│   ├── package.json
│   ├── next.config.js
│   ├── pages/
│   │   ├── _app.js
│   │   ├── index.js
│   │   ├── book.js
│   │   └── ride/[id].js
│   ├── components/
│   │   ├── Map.js
│   │   └── DriverPanel.js
│   └── services/
│   └── api.js
├── docker-compose.yml
└── infra/env.sample
```

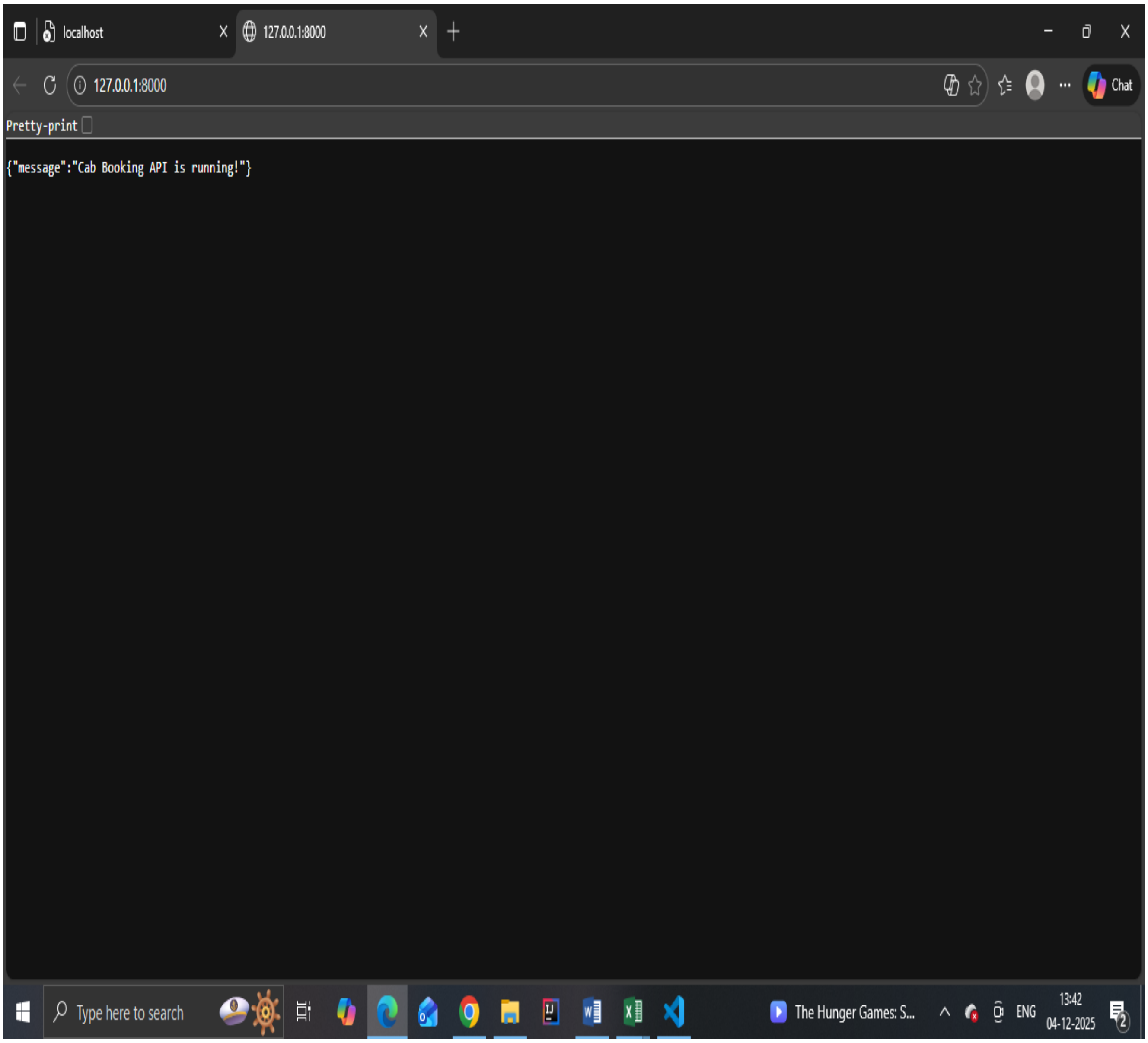
Run the file backend

C:\Users\user\Cab-Booking-Platform\backend>

venv\Scripts\activate

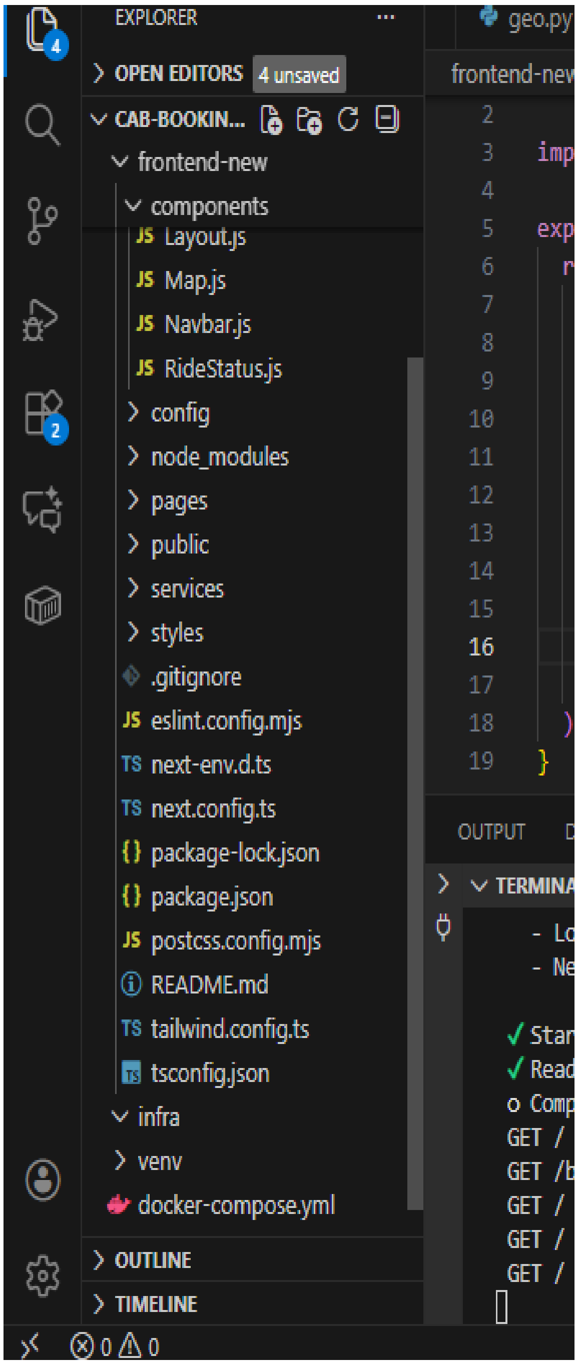
(venv) PS C:\Users\user\Cab-Booking-Platform\backend>

uvicorn app.main:app --reload --port 8000



```
{"message":"Cab Booking API is running!"}
```

Front end Structure

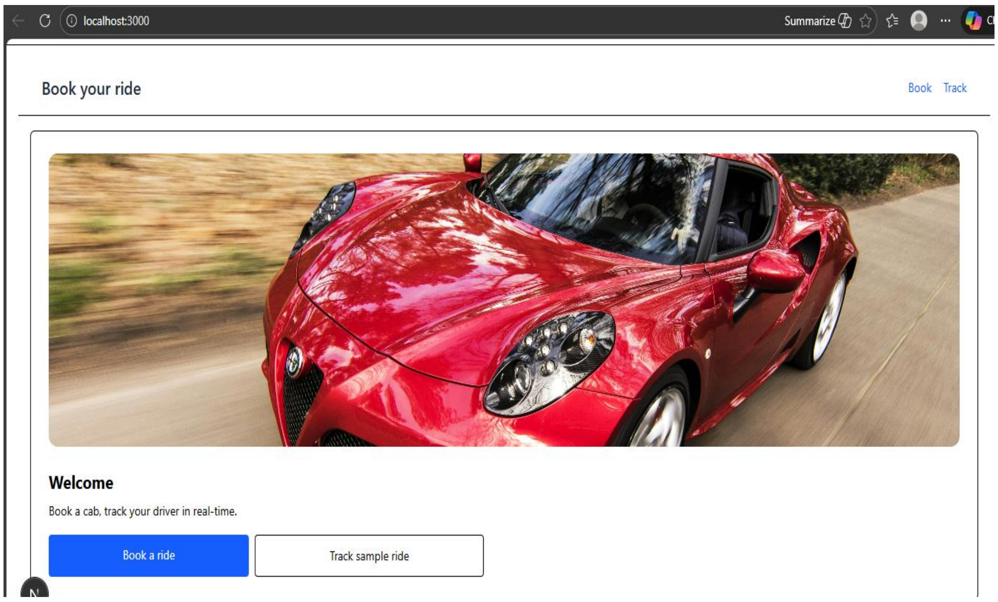


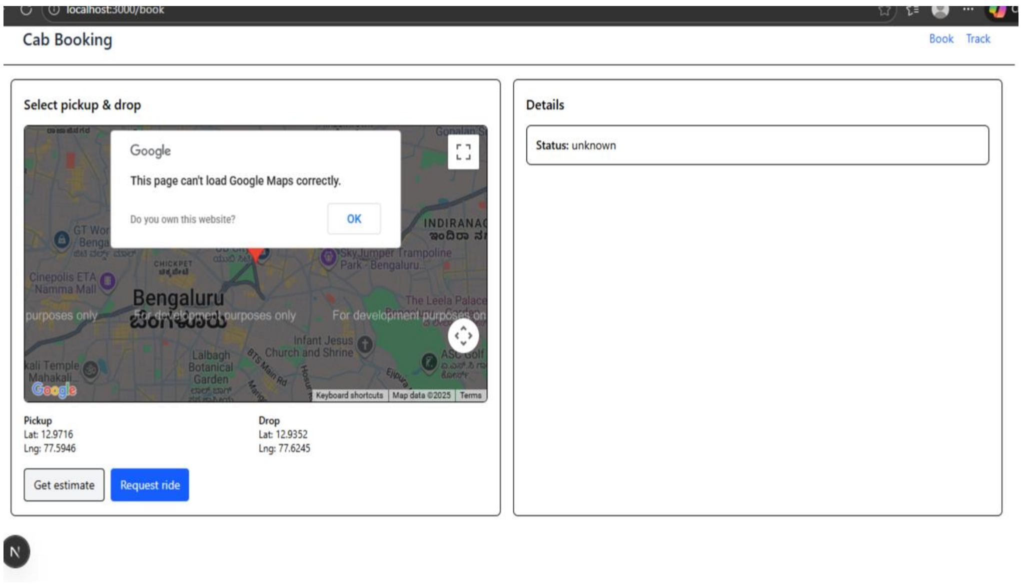
```
EXPLORER                          ...     geo.py
> OPEN EDITORS  4 unsaved                 frontend-new
v CAB-BOOKIN...                             imp
  v frontend-new                            exp
    v components                              r
        Layout.js
        Map.js
        Navbar.js
        RideStatus.js
    > config
    > node_modules
    > pages
    > public
    > services
    > styles                                  )
      .gitignore                            }
      eslint.config.mjs
      next-env.d.ts                       OUTPUT   D
      next.config.ts
      package-lock.json                   > v TERMINA
      package.json                              - Lo
      postcss.config.mjs                        - Ne
      README.md
      tailwind.config.ts                      ✓ Star
      tsconfig.json                           ✓ Read
  v infra                                     o Comp
  > venv                                      GET /
    docker-compose.yml                        GET /b
                                              GET /
> OUTLINE                                     GET /
> TIMELINE                                    GET /
⊗ 0 ⚠ 0
```

Frontend-new running

C:\Users\user\Cab-Booking-Platform\frontend-new

npm run dev





Github Link: https://github.com/ruheenatasneem/CabBooking

Youtube Link: Video link

https://youtu.be/GAe3Z3jE2po

Designed By: Ruheena Tasneem

Special Thanks to every one.