Loadshare LLaMA Assist - Project Documentation

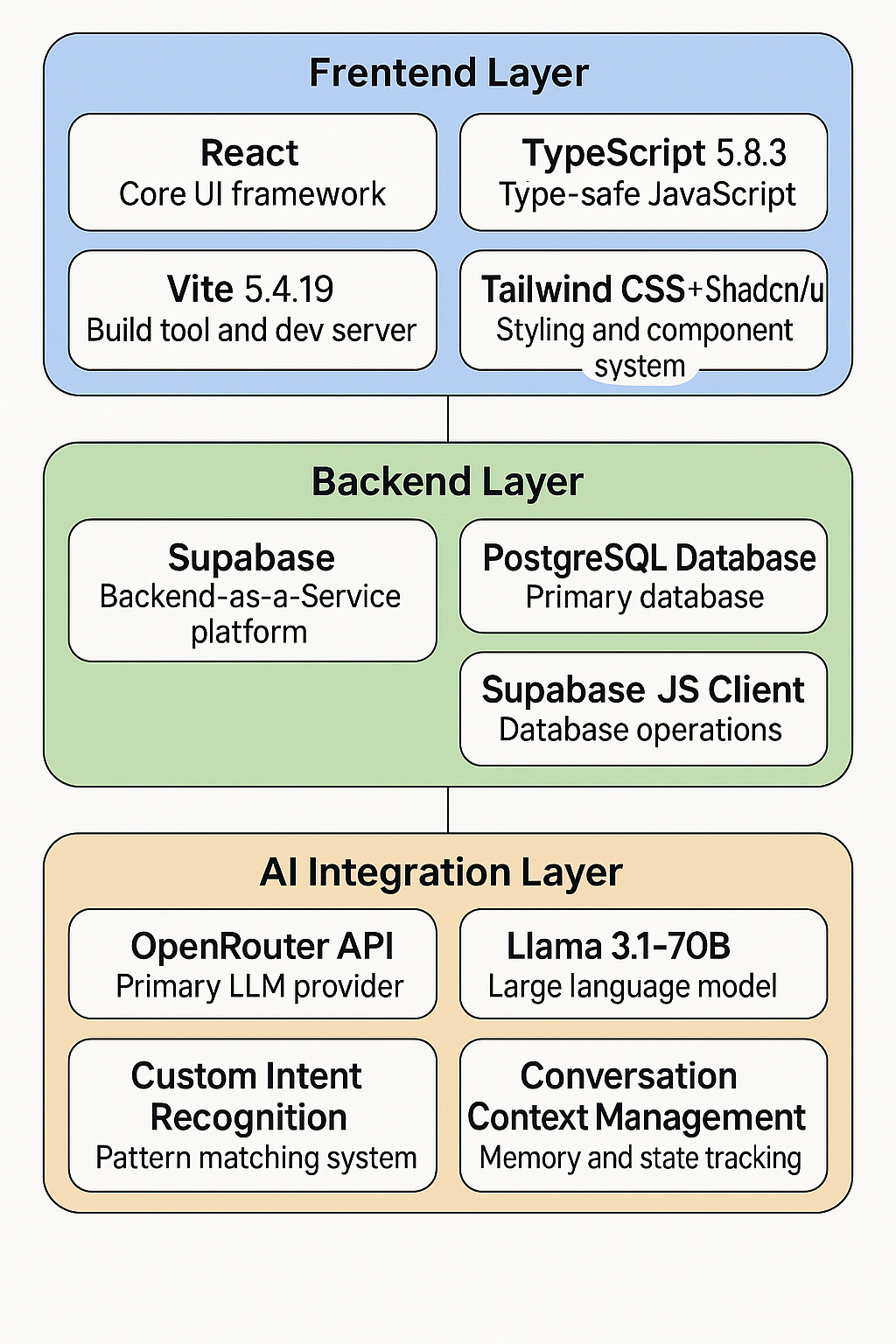
# 1. Project Overview

Loadshare LLaMA Assist is an AI-powered chatbot designed for logistics operations. It integrates a frontend (React, Vite, TailwindCSS) with a backend (Python, FastAPI/Flask), leveraging LLaMA or other LLMs for natural language processing. The system is designed to be deployed on Vercel, ensuring scalability and cloud-first deployment.

# 2. System Architecture

The architecture consists of the following layers:  
- Frontend (React + Vite + TailwindCSS)  
- Backend (Python FastAPI/Flask),Type Script  
- AI Model (LLaMA or HuggingFace integration)  
- Deployment (Vercel / Cloud hosting)

The following diagram illustrates the overall system architecture:



# 3. File & Folder Structure

- index.html → Root HTML entry point  
- vite.config.ts → Vite build configuration  
- tailwind.config.ts → TailwindCSS setup  
- tsconfig.json → TypeScript configuration  
- package.json → Frontend dependencies & scripts  
- requirements.txt → Backend Python dependencies  
- .env → Environment variables (API keys, DB credentials)  
- vercel.json → Vercel deployment configuration

# 4. Tech Stack & Dependencies

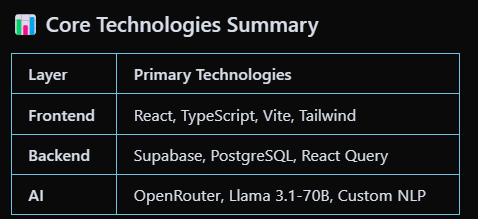
* **Frontend Layer**
* **React 18.3.1 - Core UI framework**
* **TypeScript 5.8.3 - Type-safe JavaScript**
* **Vite 5.4.19 - Build tool and dev server**
* **Tailwind CSS + Shadcn/ui - Styling and component system**

**Backend Layer**

* **Supabase - Backend-as-a-Service platform**
* **PostgreSQL Database - Primary database**
* **Supabase JS Client - Database operations**
* **TanStack React Query - Data fetching and caching**

**AI Integration Layer**

* **OpenRouter API - Primary LLM provider**
* **Llama 3.1-70B Instruct - Large language model**
* **Custom Intent Recognition - Pattern matching system**
* **Conversation Context Management - Memory and state tracking**

****

# 5. Development Workflow

1. Clone repository  
2. Install Node.js & npm dependencies (frontend)  
3. Install Python dependencies (backend)  
4. Configure environment variables in .env  
5. Run frontend with `npm run dev`  
6. Run backend with `uvicorn main:app --reload`

# 6. Deployment Workflow

1. Ensure .env variables are set on Vercel  
2. Run `vercel` in the project directory  
3. Choose project name & confirm build settings  
4. Vercel automatically deploys frontend + backend

# 7. Environment & Security

- Keep `.env` out of version control (already in .gitignore)  
- API keys must be stored in environment variables, not code  
- Use HTTPS for all API requests  
- Sanitize user inputs before sending to backend/LLM

# 8. Scalability & Future Enhancements

- Add caching layer (Redis) for frequent queries  
- Use database (MongoDB/Postgres) for storing logs  
- Support multi-language interactions  
- Deploy backend separately on scalable platforms (AWS/GCP)