SaaS AI Lead Generation Platform - Technical Architecture

Overview

This document outlines the architecture and step-by-step structure for building a SaaS application that:

- Uses AI (via FastAPI) to generate leads based on user input
- Provides a React frontend for users
- Implements a Node.js + Express backend for authentication, usage tracking, and payment
- Uses MongoDB for data persistence
- Offers a freemium model with limited free usage and paid upgrade plans

---

File/Project Structure

```
/saas-app
├── /client             # React Frontend
├── /server             # Express Backend (Auth + Stripe + MongoDB)
├── /ai-agent           # FastAPI Backend (AI Logic)
├── /shared             # Optional: shared contracts/types
├── .env                # Environment variables
└── README.md
```

Pages (Frontend - React)

Page	Path	Purpose
Home	`/`	Landing page
About	`/about`	Information about product/service
Contact	`/contact`	Contact form or support info
Services	`/services`	Details of features and plans
Login	`/login`	User login form
Signup	`/signup`	New user registration
Dashboard	`/dashboard`	Submit prompts, view leads
Pricing	`/pricing`	Plans: Free, Pro, Premium

Backend Routes (Express API)

Auth Routes

- `POST /api/signup` – Register a user
- `POST /api/login` – Login and return JWT
- `GET /api/me` – Return user info (requires token)

Lead Generation Route

- `POST /api/lead`
    - Validate user
    - Check plan + usage
    - Forward prompt to FastAPI
    - Save result in MongoDB
    - Return result to client

Admin Routes

- `GET /api/admin/dashboard` – View admin stats (Requires admin role)

Payment (Stripe) Routes

- `POST /api/checkout` – Create Stripe session for upgrading
- `POST /api/webhook` – Listen to payment events (e.g., plan upgrade)

FastAPI (AI Backend)

Endpoint

- `POST /generate-lead`
    - Input: `{"prompt": "Law firms in NY"}`
    - Output: Structured JSON with lead info

Responsibilities

- Handle incoming prompt
- Run AI inference
- Return result quickly for user display

MongoDB Collections

Users

```
{
  "_id": ObjectId,
  "name": "John",
  "email": "john@example.com",
  "passwordHash": "...",
  "role": "user" | "admin",
  "plan": "free" | "pro" | "premium",
  "searchesUsed": 3,
  "searchesLimit": 5
}
```

Leads

```
{
  "userId": ObjectId,
  "prompt": "Find IT companies",
  "aiResponse": {...},
  "createdAt": "2025-08-01T00:00:00Z"
}
```

Middleware (Express)

`authMiddleware`

- Verifies JWT token
- Attaches user to `req.user`
- Used in all authenticated routes (e.g., `/api/lead`, `/api/admin/...`)

`usageLimitMiddleware`

- Checks user's plan and usage

```
if (user.plan === 'free' && user.searchesUsed >= user.searchesLimit) {
   return res.status(403).json({ message: 'Free plan limit reached. Please upgrade.' });
}
```

- Applied on `/api/lead`

`adminMiddleware`

- Ensures `req.user.role === 'admin'`
- Returns 403 if not admin
- Used for `/api/admin/...` routes

Stripe Plan Integration

Plans

- Free → 5 searches
- Pro → 100 searches/month
- Premium → Unlimited

Steps

1. Create Stripe products
2. Attach Stripe product ID to user
3. On webhook, update user plan + limits

Prompt Flow (User Request Lifecycle)

1. User types prompt on React frontend (Dashboard)
2. React sends POST `/api/lead` with JWT
3. Express server:
    - Validates token (`authMiddleware`)
    - Checks search quota (`usageLimitMiddleware`)
    - Forwards to FastAPI
    - Gets response
    - Saves result to DB
    - Increments usage
4. Response returned to client and shown in UI

Conclusion

This architecture provides a scalable, modular SaaS platform for AI-powered lead generation. It includes:

- Authentication & usage limits
- Admin-only access to dashboard routes
- AI backend integration
- Monetization via freemium + Stripe
- Fast user feedback loop

This is ready to evolve into a production-level SaaS solution.