SOLUTION APPROACH

1️⃣ Problem Understanding(Lets goo)

In today's digital world, customer service platforms are expected to deliver fast, real-time responses. The challenge: building a **scalable, modular system** that supports user login, request submission, and intelligent support integration using .

Goal:

Build a modern customer service platform where:

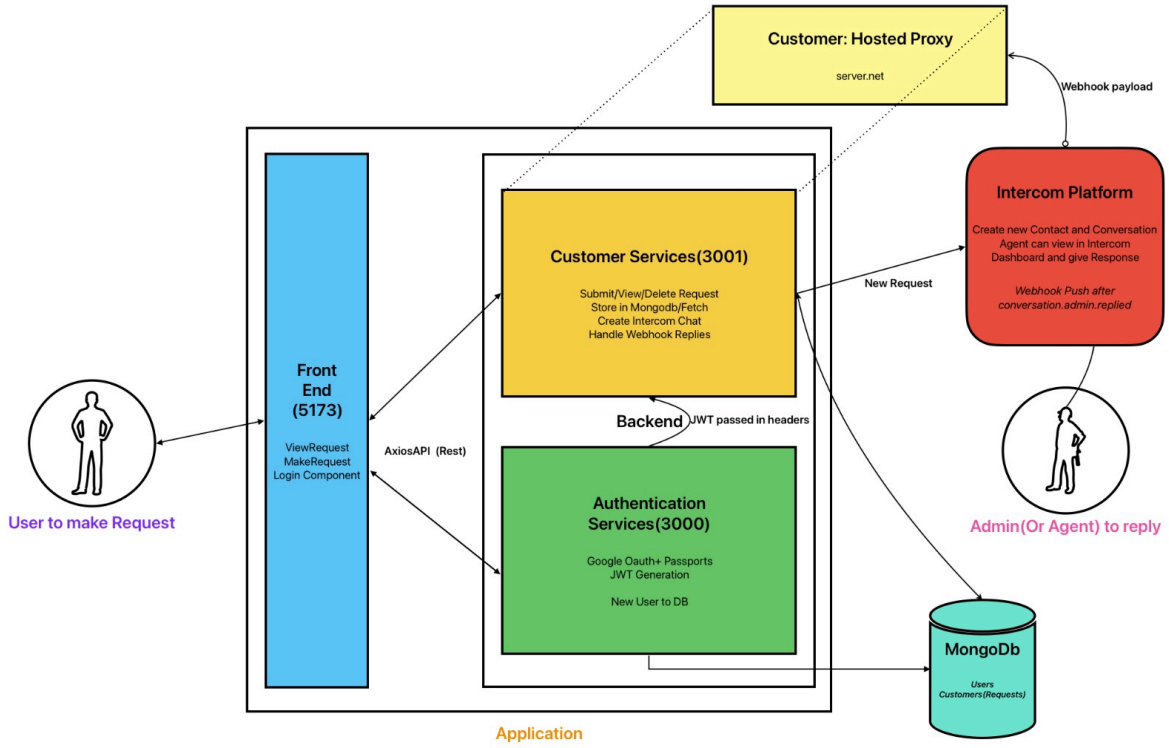
- Users can log in using Google OAuth
- Submit categorized service requests
- Get notified of agent replies
- System integrates with **Intercom** as a support backend
- Architecture should be modular, scalable, and real-time-ready

2️⃣ Our Solution Overview

I built a simple **microservices-based MERN stack** platform with:

- **Secure login via Google OAuth**
- **JWT-based request authentication**
- **Intercom integration** to convert each request into a live support conversation
- **Frontend built using Vite + React**
- **Two-stage response sync**:
  - Stage 1: Cron-based polling
  - Stage 2: Webhook via Serveo for real-time updates(No manual or time based Intevrention)

3️⃣ Solution Architecture



🔁 Workflow Steps

1. User logs in via Google
2. backedn-auth returns a JWT
3. Frontend stores JWT and uses it for all requests
4. User submits request → stored in MongoDB → sent to Intercom
5. Intercom replies are:
    - (Initially) fetched using a polling cron job
    - (Later) pushed via webhook with a Serveo-exposed URL

4️ Tech Stack Why me?

Layer	Technology	Why?
Frontend	React (Vite)	Fast dev cycle, modular UI, SPA-friendly
Backend	Node + Express	Lightweight, event-driven, great for APIs
Database	MongoDB	Flexible schema, ideal for JSON-based requests

Auth	Google OAuth + JWT	Secure, modern login approach
Third-party	Intercom API	Real support platform integration
Hosting	Serveo	Instant tunneling of webhook endpoint, No explicit Installation(used by ssh)

5️⃣ Design Evolution

Phase	Approach	Pros	Cons
Phase 1	Cron-based polling	Easy to implement	Delayed agent response sync- Update manually before timer later being Redundant
Phase 2	Webhook via Serveo	Real-time, instant update from Intercom, Free public HTTPS Endpoint Host Proxy to prevent exposing original URL(Local Host in demo) and forlocal webhook dev	Needs public tunnel (Serveo)

6️⃣ Intercom Integration

Steps Performed:

1. **Create App** at Intercom Developer Hub
2. Get:
   - `INTERCOM_TOKEN` (Personal Access Token)
   - `INTERCOM_WEBHOOK_SECRET` (Client Secret)
3. Add a **Webhook** in Intercom:
   - Set it to your **Serveo-generated URL**, e.g., `https://domainrando.serveo.net/intercom-webhook`
   - Enable `conversation.user.replied` event type
4. On user submission:

- A new conversation is created via API
- `conversationId` is stored in MongoDB

5. When an agent replies:
   - Intercom sends a webhook
   - Backend maps conversation ID → updates `response` field

7️⃣ Why Microservices?

- Separation of concerns: Auth and Requests are cleanly decoupled
- Easier scaling: Add agents/admin panels without rewriting core auth logic
- Independent deployment: Each service runs on its own port

8️⃣ Solution Efficiency

- JWT-based flow ensures **stateless, scalable sessions**
- Serveo allows **zero-config** tunneling for quick webhook integration
- Polling fallback adds **resilience** even if webhook breaks
- MongoDB's schema-less design supports **flexible growth**
- Clear separation of concerns across microservices
- Scalable backend services with MongoDB
- No polling needed after webhook setup
- Secure with OAuth + JWT
- Easily deployable architecture using Serveo or ngrok for testing

---

9️⃣ What Happens Where?

Functionality	Component or File
Login via Google	`backend-auth/routes/authRoutes.js`
JWT handling	`generateToken.js` + `authMiddleware.js`
Request form	`MakeRequest.jsx` + `/add` backend route

View submitted requests	`ViewRequests.jsx` + `/view` backend route
Delete request	`/delete/:id` route
Intercom conversation	`intercom.js` utility + `customerController.js`
Webhook handling	`/intercom-webhook` + HMAC verification
Reply fetch (cron)	`scripts/fetchReplies.js`

1️⃣0️⃣ Future Enhancements

- User Profile
- Admin dashboard for Intercom insights
- Email/SMS notifications to users based on details
- Retry logic for failed webhook delivery
- Redis cache for frequent request lookups
- Analytics dashboard (response times, categories, etc.)
- Deployement Pipeline:Dockerize services for easy CI/CD

1️⃣1️⃣ Summary

This project showcases a clean, modular, and modern full-stack application that balances functionality with performance. By iteratively enhancing the reply-fetch system and adopting microservices with JWT + OAuth, it solves the customer service flow with real-time capabilities, developer friendliness, and integration power via Intercom.

Page written in Notion