

CODE DOCUMENTATION

Project Summary

This project is a modern customer service platform built on a microservices-based MERN stack. It enables users to log in using Google OAuth, submit categorized service requests, and receive responses via Intercom. The system integrates two backend microservices (backend-auth and customrer) and a React-based frontend powered by Vite.

Project Structure Overview

|— backend-auth/

→ Authentication microservice (Google OAuth + J

WT)

|— customer/

→ Service request microservice (CRUD + Intercom in

tegration)

|— frontend/

→ Vite + React frontend UI

|— project-structure.txt

|— README.md

Layer	Technology Used
Frontend	React (Vite), Axios, React Router
Backend	Node.js, Express
Authentication	Google OAuth 2.0, JWT
Database	MongoDB with Mongoose ODM
Third-party	Intercom REST API
Hosting Proxy	Serveo (for webhook tunneling)

Backend Microservices

backend-auth/ (Port: 3000)

Handles user authentication via Google and issues JWTs for session management.

◆ Files & Responsibilities

- **index.js**

Entry point. Initializes Express server, database connection, and routes.

- **config/db.js**

Connects to MongoDB using Mongoose.

- **config/passport.js**

Sets up Google OAuth strategy using passport-google-oauth20.

- **routes/authRoutes.js**

Contains Google login (/auth/google) and callback (/auth/google/callback) endpoints.

- **routes/userRoutes.js**

Optional user profile endpoints (e.g., fetch user details).

- **models/User.js**

Mongoose schema for storing user profile data.

```
{
  email:String,
  password:String,
  googleId:String,
  name:String,
  image:String
}
```

- **models/Profile.js**

(Optional-Future-Non operational) Additional user metadata.

- **middleware/authMiddleware.js**

JWT validation middleware for protecting routes.

- **utils/generateToken.js**

Function to issue JWTs from a user ID.

Customer Service Microservice

customer/ (Port: 3001)

Handles service request creation, viewing, deletion, and Intercom integration.

◆ **Files & Responsibilities**

- **index.js**

Starts the Express server, connects to MongoDB, loads routes.

- **config/db.js**

Mongoose connection for service request database.

- **routes/customerRoutes.js**

Main API routes:

- **POST /add** → Submit a new request
- **GET /view** → View all requests of the user
- **DELETE /delete/:id** → Delete request by ID
- **POST /intercom-webhook** → Webhook endpoint to receive agent replies

- **controllers/customerController.js**

Core business logic for creating, viewing, deleting requests.

- **customer/model/model.js**

Mongoose schema:

```
{
  userId:String,
  name:String,
  phone:Number,
  category:enum[
    'General Queries',
    'Product Features Queries',
    'Product Pricing Queries',
```

```
    'Product Feature Implementation Requests'
  ],
  description:String,
  comments:String,
  response:String(default: 'Your request has been received. Our team wi
  ll respond shortly.')
  conversationId:String,

}
```

- **utils/intercom.js**

API helper that communicates with Intercom to:

- Fetch latest agent replies
- Use access token from `.env`

- **scripts/fetchReplies.js**

Node script (used in cron jobs) that:

- Queries pending requests
- Fetches replies from Intercom
- Updates `response` field in MongoDB

Frontend – `frontend/` (Port: 5173)

A React app powered by Vite for fast builds and modern features.

Key Pages & Components

- **index.html**

Main entry HTML file used by Vite.

- **vite.config.js**

Vite-specific dev server and build configuration.

- **src/main.jsx**

Bootstraps the React app and routes.

- **App.jsx**

Main routing logic between login and dashboard components.



src/pages/

- **Auth/Login.jsx**

Displays the "Login with Google" button which triggers backend-auth route.

- **Auth/authUtils.jsx**

Stores/retrieves JWT from localStorage, redirects after login.

- **Main_dashboard/MakeRequest.jsx**

Form for creating a request (category + description + Details + Additional Comments).

Hits `/add` API with JWT header.

- **Main_dashboard/ViewRequests.jsx**

Displays all submitted requests.

Pulls status (`response`) from backend and renders dynamically.



src/utils/

- **axiosInstance.jsx**

Centralized Axios setup that automatically:

- Adds JWT from localStorage to all outgoing requests
- Simplifies API calls across components



src/assets/

- Contains images/icons used in frontend pages

Workflow Summary

1. Login Flow

User logs in via Google or Traditional Email → JWT returned → stored in frontend(client side)

2. Submit Request

User submits request(becomes customer) → backend stores in MongoDB
→ Intercom conversation created → response="Your request has been

received. Our team will respond shortly."

3. Webhook / Sync

Intercom sends agent reply → webhook endpoint updates request → or fetchRepiles.js updates it via polling(initial), cron job trigger

4. View Requests

Frontend displays updated status, letting users see agent responses directly

Running

Terminal 1

cd backend-auth

npm install

npm run server

Terminal 2

cd customer

npm install

npm run server

Terminal 3

cd customer

npm install

ssh -R 80:localhost:3001 serveo.net

Copy this url as root webhook endpoint in intercom(Hosting Proxy)

(Expose webhook for intercom via serveo- esy no installation required)

Use the generated public URL (e.g., <https://somedomain.serveo.net/intercom-webhook>) as your webhook endpoint in Intercom settings

Terminal 4

cd frontend

npm install

npm run dev

Document written using in Notion

