Attack Capital Assignment: Unified Inbox for Multi-Channel Customer Outreach

Key Objectives

- **Build a unified communication platform** using Next.js that aggregates messages from SMS (Twilio), WhatsApp (Twilio API),(optional email, and social media (e.g., Twitter/X, Facebook Messenger)) into a single inbox.
- **Enable seamless outreach**: Allow teams to send/reply across channels, schedule messages, manage contacts with history/notes, and collaborate in real-time.
- **Incorporate analytics**: Track engagement metrics like open rates, response times, and conversion funnels in a dashboard.

Tech Stack

- **Frontend/Backend**: Next.js 14+ (App Router, TypeScript).
- **Database**: Postgres via Prisma ORM.
- **Authentication**: Better Auth (with credentials or Google provider; include role-based access for teams, e.g., viewer/editor/admin).
- **Integrations**: Twilio SDK (SMS/WhatsApp), optional Nodemailer or Resend for email, optional Twitter/X API v2, optional Facebook Graph API; optional webhooks for HubSpot/Slack/Zapier.
- **Other**: Well-documented code (JSDoc/TypeDoc), ESLint/Prettier, Git repo with branches.

Quick Setup Steps

1. Create a free Twilio account (twilio.com/try-twilio) and buy a trial number (e.g., US SMS/WhatsApp-enabled).
2. Set up Postgres (local via Docker or cloud like Supabase).

Comprehensive Assignment Specification: Building a Unified Multi-Channel Inbox

This assignment extends telephony into a full CRM-like tool, simulating a "unified box" for sales/support teams to centralize outreach across channels, reducing context-switching. Expect architectural decisions like event-driven updates via WebSockets for collab, secure webhook validation, and scalable queuing for automations. Use AI tools sparingly for boilerplate but derive custom logic, e.g., conflict resolution in shared notes. Submit a public GitHub repo with README.md detailing decisions, setup, and an integration comparison table (e.g., latency/cost per channel) + key decisions.

Project Overview

Develop a full-stack app for team-based customer engagement. Core flow: Authenticate → View unified inbox (threaded by contact) → Send/reply across channels → Schedule automations → Collaborate via notes → Monitor analytics.

- **Channels**: SMS/MMS (Twilio), WhatsApp (Twilio Sandbox), optional Email (IMAP/SMTP or API), Social (Twitter DMs, Facebook Messenger).
- **Trial Mode**: Display Twilio trial number in UI; restrict to verified contacts.
- **Multi-Team**: Real-time presence (e.g., "Editing by @user") and @mentions in notes.
- **Production-Ready**: Secure (OAuth for social APIs, env secrets), observable (logs to DB), extensible (add channels via config). Total effort: 12-18 hours, emphasizing webhook orchestration and data normalization across channels.

Detailed Requirements

1. Authentication and User Management

2. Database (Postgres via Prisma)

- Use Prisma for schema/migrations and queries.
- Normalize messages across channels into a unified Message table; track contacts with history.

3. Core UI/Frontend (Next.js)

- **Unified Inbox**: Kanban-style view (threads by contact), somewhat similar to OpenPhone UI, with channel badges, searchable/filterable (e.g., by status: unread, scheduled).
- **Contact Profile**: Modal with history timeline, notes section (toggle public/private), quick actions (dial/send).

- **Composer**: Rich text editor for cross-channel sends; preview scheduled messages.
- **Team Collab**: Inline @mentions, real-time cursors in notes; conflict-free editing (e.g., via Yjs).
- **Twilio Trial Integration**: Fetch/buy numbers via Twilio API; display in settings UI with "Buy Now" button (sandbox mode).
- **Analytics Dashboard**: Charts for metrics (response time, channel volume); exportable reports.
- **Styling**: Tailwind CSS; responsive
- **Challenge**: Use React Query for optimistic updates; handle media embeds (e.g., WhatsApp images) without blocking renders.

4. Backend Integrations and Features

- **Channel Integrations**:
    - **Twilio SMS/WhatsApp**: Webhooks to /api/webhooks/twilio for inbound; send via client.messages.create() (support MMS attachments). Enable WhatsApp Sandbox for testing.
    - **Email** (optional): Integrate via Resend API for outbound; poll IMAP for inbound or use webhooks (e.g., Gmail API).
    - **Social Media** (optional): Twitter API v2 for DMs (OAuth app setup); Facebook Graph API for Messenger (webhook verification).
    - **Business Tools** (optional): HubSpot API for contact sync (webhook on create/update); Slack/Zapier via outgoing webhooks for notifications (e.g., "@channel new lead").
- **Contact Management**: Unified Contact schema with phone, email, socialHandles; auto-merge duplicates by fuzzy matching.
- **Auto-Text Scheduling**: Implement scheduling simply from Postgres only; cron-like UI for templates (e.g., "Follow-up in 3 days").
- **Internal Notes**: Threaded, role-visible; encrypt private ones (e.g., via Prisma middleware).
- **Calls & Texts**: Embed Twilio Client SDK for in-app VoIP; fallback to browser SMS.
- **Orchestration**: /lib/integrations.ts factory for channels (e.g., const sender = createSender('whatsapp'); await sender.send(payload);).

5. Code Quality and Documentation

- **Maintainability**: Modular, documented, Type-safe with Zod validation.
- **Docs**: Inline JSDoc; ERD diagram (Mermaid in README); setup script (npm run dev launches all services).

Deliverables

- Public GitHub repo.
- In README.md: Include an integration comparison table (latency, cost, reliability per channel) + key decisions.
- Video walkthrough (3-5 min, using Loom or unlisted YouTube): Send cross-channel message, schedule automation, collab on notes, view analytics.

Timeline and Submission

- 4 days from receipt.
- Questions? Email us back.
- Success metric: Functional multi-channel prototype + analysis showing integration depth.

This advances toward a full outreach suite—looking forward to your unified vision!