
✅ TRAÉ IDE – FULL BUILD PLAN

AI EMAIL WEB APP (Using x-ai/grok-4.1-fast)

This plan is formatted so Trae IDE + the GPT-4.1 coding agent can execute it step-by-step.

🏗️ SECTION 1 — PROJECT INITIALIZATION

1. Create Monorepo Structure

Trae IDE → "Create Project"

/ai-email-app

/frontend (Next.js + Tailwind + Supabase Client)

/backend (FastAPI + MCP + Nylas + Grok)

/workers (Sync Workers + Embeddings)

/infra (Docker + CI/CD + Render Config)

🖥️ SECTION 2 — FRONTEND BUILD PLAN (Next.js + Tailwind, Hosted on Vercel)

2.1 Initialize Frontend Project

npx create-next-app@latest frontend --typescript

cd frontend

npm install @supabase/supabase-js axios swr zustand react-hook-form

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

2.2 Configure Tailwind

Add to tailwind.config.js:

content: ["./app/**/*.{ts,tsx}", "./components/**/*.{ts,tsx}"]

2.3 Key Components Trae IDE Needs to Generate

Components:

- InboxList.tsx
- EmailThread.tsx

- AIReplyBox.tsx
- AttachmentSummary.tsx
- SecurityBanner.tsx
- DocFinderSidebar.tsx

Contexts / Services:

- /lib/supabaseClient.ts
- /lib/api.ts (for backend calls)
- /store/emailStore.ts

Pages:

- /app/inbox/page.tsx
- /app/email/[id]/page.tsx
- /app/settings/page.tsx
- /app/security/page.tsx

2.4 Frontend API Configuration

Create .env.local:

NEXT_PUBLIC_SUPABASE_URL=

NEXT_PUBLIC_SUPABASE_ANON_KEY=

BACKEND_API_URL=https://your-render-service.onrender.com

□ SECTION 3 — AI LAYER BUILD PLAN (x-ai/grok-4.1-fast)

3.1 Install Grok Client

In /backend:

pip install groq

3.2 Create AI Modules

Trae should generate the following files:

backend/ai/generate_reply.py

backend/ai/summarize_thread.py

backend/ai/keywords_to_email.py

backend/ai/phishing_classifier.py

backend/ai/document_relevance.py

3.3 Define Reusable Grok Client

backend/ai/client.py:

```
from groq import Groq


client = Groq(api_key=os.getenv("GROQ_API_KEY"))
```

⚙ SECTION 4 — BACKEND BUILD PLAN (FASTAPI on Render)

4.1 Initialize Backend

```
mkdir backend

cd backend

pip install fastapi uvicorn supabase groq python-dotenv pdfplumber python-docx
pytesseract pandas
```

4.2 Main Entry Point

backend/main.py:

```
from fastapi import FastAPI

from routers import email, ai, attachments, documents, phishing


app = FastAPI()

app.include_router(email.router)

app.include_router(ai.router)

app.include_router(attachments.router)

app.include_router(documents.router)

app.include_router(phishing.router)
```

4.3 Backend API ROUTERS to Generate

EMAIL Router

- /email/sync
- /email/list
- /email/{id}
- /email/search

AI Router

- /ai/generate-reply
- /ai/summarize-thread
- /ai/keywords
- /ai/classify-phishing

DOCUMENT Router

- /documents/find
- /documents/mcp-sync

ATTACHMENTS Router

- /attachments/summary
- /attachments/ocr

PHISHING Router

- /phishing/score
- /phishing/scan-inbox

Trae IDE's GPT-4.1 agent can auto-generate all these routers.

📩 SECTION 5 — EMAIL SYNC (NYLAS)

5.1 Install

pip install nylas

5.2 Worker Process

Create:

workers/email_sync_worker.py

Responsibilities:

- Poll Nylas every 5 minutes

- Store new emails in Supabase
- Generate embeddings
- Trigger phishing scan
- Trigger MCP document search if email mentions a file

🗄️ SECTION 6 — SUPABASE DATABASE PLAN

6.1 Tables to Auto-Create

Trae IDE → “Run SQL in Supabase”:

emails

id UUID PK

user_id UUID

subject TEXT

body_text TEXT

body_html TEXT

sender TEXT

timestamp TIMESTAMP

thread_id TEXT

attachments

id UUID

email_id UUID

storage_path TEXT

mime TEXT

summary TEXT

email_vectors (pgvector)

id UUID

email_id UUID

embedding VECTOR(3072)

document_vectors

id UUID

cloud_id TEXT

embedding VECTOR(3072)

metadata JSON

6.2 Indexes

CREATE INDEX vec_idx ON email_vectors USING hnsw (embedding vector_cosine_ops);

🔍 SECTION 7 — MCP DOCUMENT SEARCH (Google Drive + OneDrive)

7.1 Install

pip install google-api-python-client microsoftgraph-python

7.2 Required Modules

Generate:

backend/mcp/google_drive.py

backend/mcp/onedrive.py

backend/mcp/sharepoint.py

backend/mcp/auto_matcher.py

7.3 Workflow

1. Email references a file (detected using Grok)
2. Worker queues MCP search job
3. Retrieve documents from Google/Microsoft
4. Embed documents into pgvector
5. Best match returned to frontend

📎 SECTION 8 — ATTACHMENT SUMMARIZATION ENGINE

8.1 Attachment Processor Modules

backend/attachments/pdf_processor.py

backend/attachments/docx_processor.py

backend/attachments/image_ocr.py

backend/attachments/csv_processor.py

backend/attachments/summary_engine.py

8.2 Worker for Heavy Jobs

workers/attachment_worker.py

Processes:

- PDF extraction
- OCR
- Summaries
- Infection/suspicious file detection

🔐 SECTION 9 — PHISHING DETECTION

9.1 Classifier Modules

backend/phishing/classifier.py

backend/phishing/url_scanner.py

backend/phishing/domain_rules.py

backend/phishing/content_risk.py

9.2 Output Buckets

- inbox
- promotional
- potential_scam
- scam

📦 SECTION 10 — DOCKERIZATION PLAN

10.1 Backend Dockerfile

FROM python:3.11

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

10.2 Worker Dockerfile

Same but runs:

python workers/email_sync_worker.py

🚀 SECTION 11 — DEPLOYMENT (RENDER + VERCEL)

11.1 Frontend (Vercel)

- Connect GitHub repo
- Auto-deploy
- Add Supabase environment variables

11.2 Backend (Render)

- Deploy Docker image
- Add environment variables:

SUPABASE_URL=

SUPABASE_SERVICE_KEY=

NYLAS_API_KEY=

GROQ_API_KEY=

GOOGLE_OAUTH=

MICROSOFT_OAUTH=

REDIS_URL=

11.3 Workers (Render Background Services)

- Deploy from /workers
- Keep-alive enabled

📊 SECTION 12 — MONITORING (DATADOG)

Install Datadog agent in backend container:

pip install datadog

Send logs & metrics from:

- API routes
- Workers
- AI calls latency

🚀 SECTION 13 — DEVELOPMENT ROADMAP FOR TRAE IDE

Phase 1 — Foundation

✓ Monorepo setup
✓ Frontend skeleton
✓ Backend skeleton
✓ Supabase tables

Phase 2 — AI Features

✓ AI reply
✓ Thread summary
✓ Keyword email generator

Phase 3 — Email & Documents

✓ Nylas sync worker
✓ MCP cloud search

Phase 4 — Security

✓ Phishing classifier
✓ Attachment risk engine

Phase 5 — UI Polishing

✓ Final UX + testing
✓ Onboarding + settings page

☆ Final Notes for Trae IDE + GPT-4.1 Agent

When generating code, instruct Trae:

Follow the Build Plan strictly. Organize files as defined. Use strongly typed code. Prioritize modularity.
