Newspaper.AI CURSOR PROMPT:

Delete everything currently in render.

There are multiple collaborators on this, so it needs to be a github push pull flow, please review this and the other document.

Cursor, jot these ideas down and put together a strategy to follow and implement, then begin. Review this prompt and do not miss a single detail. Please use Render CLI and Supabase CLI to connect to their backends through cursors. Do not make this on localhost, make this deployable on render instantly. There is a current project existing, start over and delete all of it, using this information. Use test driven development for this whole project. Let's make it look like an award winning news site, take your time with development to ensure it is correct. Do not develop anything on localhost, make everything deployable on render, not meant for mock data or private use make it able to be used by users. Make this additionally desktop and mobile compatible.

Use this API Keys to create an AI news aggregator with a login and register functionality with an extensive onboarding flow and location services to page, that scares news from these API's and provides them to the user by making news with the headlines and 6 key bulletpoints using AI and Use retrieval-augmented generation (RAG) to compile background info: Timeline of events, Historical comparisons, Economic or social implications, Presented only if the user taps for more depth. Then another dropdown on how each article affects the user based on the onboarding questionnaire they filled out. The news will be split into four sections: For You, Trending, Local, International, and National, and all of the news will fall into the relevant categories based on user input and API keys. This should be a worldwide news application, and presenting our original research and summarization through our cost effective AI through summarizing all of this ^ through openrouter.

Real problem: the loss of investigative reporting that takes 30 minutes to explain context has lead to people seeing news and reading news but not understanding the context of how it applies to their life and the economies they are in.
Solution: a news app that combines TikTok level digestion speed but using AI to generate the outcome and affects of every article tailored to the reader so it has meaning
- Frontend: React (w/ Tailwind CSS, shadcn/ui, Framer Motion)
- Backend: Node.js + Supabase + OpenRouter/OpenAI
- Deployment: Render (no localhost)
- Testing: Vitest for unit, Playwright for E2E

Structure code as modular components:
```
- src/
  - components/
  - pages/
  - services/
```

- utils/
- api/
- tests/

Add logging with Sentry, and basic usage analytics via PostHog or Plausible.

1. Sign in page for the user with Google or Email (Supabase intwgration

2. User Profiling for Context

Build a robust profile for each user based on:

- Job / Industry
- Location (Radar)
- Financial situation or interests (investing? entrepreneur? student?)
- Political lean / values (optional, ethically handled)

Current news outlets

- Personal interests (climate, AI, geopolitics, etc.)

1. News aggregator, researcher (API functionality)

○ Mobile device optimization

News APIs USE ALL TO MAXIMUM API REQUESTS PER DAY:

1. **TheNewsAPI**
   - URL: https://www.thenewsapi.com
   - API Key: c3AcERwz2ZsZc0ABKhwc00OnAOqhyCKAySAsdiSc
2. **NewsData.io**
   - URL: https://newsdata.io
   - API Key: pub_8217957942c3b5247b479818ae6984adf5333
3. **ApiTube**
   - URL: https://apitube.io
   - API Key: api_live_kbAHRuznY0gfNohekvP6dDFgtYGzy0Afsnwg1hYA1jRvAOrDEblvhq
4. **Breaking News Trends Detection (GitHub Repo)**
   - URL: @https://github.com/oubaidHL/Breaking-News-Trends-Detection
   - No direct API key provided (this is a code repository, not an API service).
5. **NewsAPI**
   - URL: https://newsapi.org
   - API Key: 87ecc9641d894c0396b5495014879e9d

Please implement: Translation software for multilingual support/.
Supabase Project ID: mrfcrewlkwrqtwzlxpra

https://mrfcrewlkwrqtwzlxpra.supabase.co

Supabase Anon Public:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Im1yZmNyZXdsa3dycXR3

emx4cHJhIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NDQxNDQ3MzIsImV4cCI6MjA1OTcyMDczMn0.6jzgK-K6nntipO0ZSnmXSAvb53xqp7-uQF_S7KHDLJU

```
Service_role
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Im1yZmNyZXdsa3dycXR3emx4cHJhIiwicm9sZSI6InNlcnZpY2Vfcm9sZSIsImlhdCI6MTc0NDE0NDczMiwiZXhwIjoyMDU5NzIwNzMyfQ.pWvGZB2yzW9r2k9pBIX5SWeoV_T-wwbDWLagoO6KyoY
```

Github Repo
The repository used for your Web Service.
echo "# The-PWN" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ShaneAtkins1/The-PWN.git
git push -u origin main

git remote add origin https://github.com/ShaneAtkins1/The-PWN.git
git branch -M main
git push -u origin main

https://github.com/ShaneAtkins1/The-PWN.git
git@github.com:ShaneAtkins1/The-PWN.git

Github Repo
The repository used for your Web Service.
https://github.com/ShaneAtkins1/pwn-project
Render Deploy Hook: https://api.render.com/deploy/srv-d00vp9idbo4c73dn9l10?key=yE-UIxSsPMM
Build Command
Render runs this command to build your app before each deploy.
$ npm install && npm run build
Radar API Keys

Live secret (server)	prj_live_sk_1f190e6c2e1c8ba7e4c19c0782b65754e1000c03
Live publishable (client)	prj_live_pk_f723960ed2a6ed433677b0b36c76897047ff0494
Test secret (server)	prj_test_sk_868bb381de2858ef10987ca95cf4db718fefaab3
Test publishable (client)	prj_test_pk_cebc5866ff9c35883163cb51da946c03aaf1ddb

1. **Google Custom Search API**
    - Search Engine ID: 14062bd40491e400a
    - Google API Key: AIzaSyAuiMJJKV1d9m_f7x08OhnTaCX4DAWEYTc

Openrouter API Key:
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
Google Oath Client ID:
996566628358-adhfr05nun4ngnqkbc84bglpg2msv1j4.apps.googleusercontent.com
Google Cloud Console API Key: AIzaSyAuiMJJKV1d9m_f7x08OhnTaCX4DAWEYTc
Simplified Breakdown of Project.

🔐 **1. Authentication & Onboarding**

- **Tech Stack**: Supabase for login/register with Google + email
- **TDD Coverage**:
- Test for successful user registration/login
- Fail cases: duplicate email, invalid credentials
- Onboarding flow captured in a form wizard (multi-step)
- **Features**:
- Radar integration for location-based personalization
- User questionnaire stored to profile:
- Industry/Job
- Financial interest
- Political lean (ethically handled, opt-in)
- Topics of interest (checklist)
- Current news outlets followed (dropdown w/ custom input)

🧠 **2. AI-Powered News Aggregator**

- **Use APIs** to pull fresh articles into a queue:
- Use pubDate, source, tags, etc. to categorize
- **Summarization Pipeline (OpenAI + RAG)**:
- Headline
- 6 Key Bullet Points (concise)
- Optional "Tap for Context" → provides:
- Timeline of related events
- Historical comparisons
- Economic/social impact
- "Why this matters to you" dropdown → based on user profile tags

---

📲 **3. UX & UI Design (Mobile + Desktop)**

- Split into five sections:
- **For You** – Personalized by location, job, political lean
- **Trending** – Aggregated by engagement
- **Local** – From Radar + APIs + user city
- **National** – Based on user country
- **International** – Top global stories
- **Design system**:
- Dark + light modes
- TikTok-speed UX (scroll-friendly, expandable cards)
- CTA: "Read full article" / "Tap to go deeper" / "How this affects you"

🧪 4. Test-Driven Development (TDD) Architecture

- Full unit and integration test coverage:
- Auth flow
- API pulls
- AI summarization
- RAG-enhanced “context dropdown”
- Profile-based news filtering
- Suggested Testing Tools:
- Vitest or Jest for logic
- Cypress or Playwright for E2E

🧠 5. Architecture & DevOps

- **Frontend**: React + Tailwind (or similar), Cursor-based dev
- **Backend/API Layer**:
- Serverless functions on Render
- Cron jobs for API fetch every 15 min
- **Database**:
- Supabase for users, profiles, and preferences
- Article cache & metadata DB (avoid duplicate rendering)
- **Hosting**: Render (fully deployable public-facing project)
- **RAG Storage**:
- Embedding context into vector DB (e.g., Supabase pgvector or Weaviate)

Additional:

🔔 Notification system	Push alerts or email digests for categories user follows
🧾 Save-to-read-later & history	Personal curation feature
📈 Dashboard	Visualize topics you care about over time
🌐 Multilingual support	For international appeal

Absolutely — to make **Newspaper.AI** truly functional, scalable, and user-ready, here are **key backend, frontend, infrastructure, and AI details** that go *beyond the obvious* to ensure this thing works reliably and intelligently in real-world conditions:

🔧 FUNCTIONALITY & STABILITY ADD-ONS

1. 🧠 AI Summarization Pipeline: Production-Ready

- **Throttle/Queue Summarization**: You *must* queue and throttle calls to OpenAI to avoid hitting rate limits.
- Use job queues (e.g., BullMQ, Graphile Worker, or Render Cron jobs)
- **De-duplication Logic**:
- Hash article titles + source URLs to avoid duplicate summarization/storage.
- **Error handling**:
- If OpenAI fails, show fallback: “Summary not available. Tap for full article.”

- **Summarization Model Design**:
- Prompt: "Summarize this article in 6 key bullet points and 1 headline."
- For RAG context: Embed external data (Wikipedia, prior articles) using pgvector and retrieve based on similarity scores.

---

2. 📡 Real-Time API Handling

- **Rate Limiting Protection**:
- Most news APIs have strict daily limits (even on paid plans). Cache aggressively in Supabase.
- Use Redis or Supabase edge functions to store short-term articles in a TTL cache.
- **Categorization Engine**:
- Use tags from the API + NLP keyword extraction (OpenAI or compromise.js) to auto-categorize into:
- Local / National / International / Trending / For You
- **Time Freshness Sorting**:
- Use pubDate + a "decay" factor to rank "Trending" by recency × engagement (clicks or shares if available).

---

3. 💼 User Data & Personalization

- Store user profile inputs as JSONB in Supabase (flexible schema):
- Store onboarding responses (e.g., topics, politics, job)
- Use that to **rank news articles by relevance score** (match topics + location)
- Optional: Add **"News IQ"** tracker:
- Track how many summaries a user reads, and evolve recommendations over time

---

4. 📲 UI/UX That Doesn't Just Look Good — It Works

- **Adaptive Card Layouts**:
- Use expandable cards with tabs: *Summary*, *Context*, *Impact*
- Optional: Add animation (Framer Motion) for micro-interactions
- **Tap-to-Context UX**:
- When a user taps "Give me context," show:
- Timeline (OpenAI: "Generate timeline of events for [topic]")
- Historical analogies ("This is similar to X event in 2008…")
- Economic/social implications (OpenAI analysis based on article)

---

5. 🌍 Localization & Accessibility

- **Geo support**:
- Use Radar to pull city/state/country for local news filter
- **Multilingual Support** (Optional MVP++ feature):
- Use OpenAI or DeepL to offer summaries in user's native language
- **A11Y Compliant**:
- Ensure keyboard nav, color contrast, alt text for all visuals

---

6. 🔐 Security & Deployment

- **Supabase security rules**:
- Lock down read/write access per user
- **Deployment**:
- Use Render’s Blueprints to auto-deploy PRs
- Use a staging and prod environment (e.g., newspaper-ai-dev and newspaper-ai)
- **Error Logging**:
- Hook in LogRocket or Sentry to trace UI/AI/API issues
- **Offline Mode (Progressive Web App)**:
- Cache summaries for offline reading (even 2–3 articles gives a big UX win)

---

7. 🧪 Testing and Monitoring

- **TDD suggestions**:
- Unit tests for:
- API fetcher
- Article de-duplication
- Summarization logic
- RAG-enhanced context generator
- Integration tests:
- User onboarding flow
- Profile → news relevance
- Summary → tap-to-context
- **Uptime Monitor**:
- Use a free Render health check or Upptime to ping endpoints every 5 mins

---

✅ TL;DR: You Also Need…

Feature	**Reason**
✅ API throttling + caching	Avoid rate limits, improve speed
✅ Deduplication logic	Prevent spam and repetitive content
✅ RAG + vector search for context	Context-aware news summaries
✅ Security rules (Supabase)	Protect user data
✅ Tap-to-context UX	Make summaries more valuable
✅ Production-level deployment config	Don’t rely on local dev
✅ Test coverage (TDD)	Ensure scalability and stability

Full ReadMe:

# Newspaper.AI 🗞️ — Personalized AI-Powered Global News Platform

TL;DR: A deployable, AI-enhanced news platform that delivers globally aggregated news with TikTok-speed summaries, context-aware insights, and personalized impact — powered by OpenAI + RAG and built on Render.

## 🧩 Features

- 🌍 **Worldwide coverage** from 5+ news APIs
- 🔐 **Supabase Auth** — Google & Email login/register
- 🧠 **Onboarding**: Multi-step form capturing location, job, politics, finances, interests
- 💬 **AI Summarization**: 6 bullet points + headline per article

- 📜 **Context On Demand**: Tap for RAG-enhanced historical/economic/social context
- 🎯 ***“Why This Affects You”*** — summaries tailored to your profile
- 🧭 Categorized into: For You, Trending, Local, National, International
- 🌐 **Multilingual Support** via OpenAI / DeepL
- 💻📱 Fully responsive for mobile + desktop
- 🔁 **Cron jobs** pull fresh news every 15 minutes
- 🧪 **Test-Driven Development** with full test coverage

## 🛠️ Tech Stack

Layer	Tooling
Frontend	React, Tailwind CSS, Shadcn UI
Backend	Supabase, Node.js, Serverless (Render)
AI Services	OpenAI (via OpenRouter), pgvector (RAG)
Location	Radar API
News APIs	TheNewsAPI, NewsData.io, ApiTube, NewsAPI, Google Custom Search
Deployment	Render (CI/CD via webhook)
Testing	Vitest, Playwright, TDD-first
Analytics	LogRocket or Sentry (optional)

## 🚀 Deployment Info
- **Render Deploy Hook**:
  `https://api.render.com/deploy/srv-d00vp9idbo4c73dn9l10?key=yE-UIxSsPMM`
- **Supabase Project ID**: `mrfcrewlkwrqtwzlxpra`
- **Build Command**:
  ~~~bash
  npm install && npm run build
MIT License:
---
### 📁 `project-structure.json`
```json
{
  "src": {
    "components": {
      "NewsCard": "Displays headline, 6 bullet points, expandable context",
      "LoginForm": "Handles Supabase login/register",
      "OnboardingWizard": "Multi-step user profiling component",
      "CategoryTabs": "UI for For You, Trending, Local, etc."
    },
    "pages": {
      "Home.tsx": "Landing page with news feed",
      "Auth.tsx": "Login/register page",
      "Onboarding.tsx": "Onboarding flow post-auth"
    },
  ~~~

```
    "services": {
      "newsApi.ts": "Fetch and normalize articles from all APIs",
      "summarize.ts": "Calls OpenRouter to summarize articles",
      "context.ts": "RAG-enhanced context generator",
      "profile.ts": "Handles user profile logic (matching, scoring)"
    },
    "utils": {
      "dedupe.ts": "Prevents duplicate articles",
      "categorize.ts": "Auto-assigns categories to articles",
      "throttle.ts": "Rate-limiting for API calls"
    },
    "api": {
      "routes": {
        "pullNews.ts": "Cron job endpoint to fetch and store articles",
        "summarizeArticle.ts": "Summarization microservice"
      }
    },
    "tests": {
      "unit": {
        "summarize.test.ts": "Tests OpenAI summarization logic",
        "categorize.test.ts": "Tests article tagging logic"
      },
      "e2e": {
        "onboarding.spec.ts": "Tests full onboarding flow",
        "newsflow.spec.ts": "Tests category navigation + summaries"
      }
    }
  },
  "public": {
    "index.html": "Entry point",
    "favicon.ico": "Site icon"
  },
  "env": {
    ".env.example": "All required keys with placeholders"
  }
}
```
Add

Newspaper.AI CURSOR PROMPT:

Cursor, jot these ideas down and put together a strategy to follow and implement, then begin. Review this prompt and do not miss a single detail. Please use Render CLI and Supabase CLI to connect to their backends through cursors. Do not make this on localhost, make this deployable on render instantly.

There is a current project existing, start over and delete all of it, using this information. Use test driven development for this whole project. Let's make it look like an award winning news site, take your time with development to ensure it is correct. Do not develop anything on localhost, make everything deployable on render, not meant for mock data or private use make it able to be used by users. Make this additionally desktop and mobile compatible.

Use these API Keys to create an AI news aggregator with a login and register functionality with an extensive onboarding flow and location services. The app will scrape news from these API's and provide them to users by creating summaries with headlines and 6 key bulletpoints using AI. Use retrieval-augmented generation (RAG) to compile background info: Timeline of events, Historical comparisons, Economic or social implications, presented only if the user taps for more depth. Then include another dropdown on how each article affects the user based on the onboarding questionnaire they filled out.

The news will be split into five sections: For You, Trending, Local, International, and National, with content categorized based on user input and API data. This should be a worldwide news application, presenting our original research and summarization through our cost-effective AI by summarizing all of this through OpenRouter.

CORE PROBLEM: The loss of investigative reporting that provides deep context has led to people consuming news without understanding how it applies to their lives and economies.

SOLUTION: A news app that combines TikTok-level digestion speed with AI-generated analysis of outcomes and effects of every article, tailored to each reader for personal relevance.

TECHNICAL STACK:
- Frontend: React (w/ Tailwind CSS, shadcn/ui, Framer Motion)
- Backend: Node.js + Supabase + OpenRouter/OpenAI
- Deployment: Render (no localhost)
- Testing: Vitest for unit, Playwright for E2E
- Authentication: Supabase Auth with Google & Email options
- Caching: Redis for API response and AI generation caching

- Monitoring: Sentry for error tracking, PostHog or Plausible for analytics

CODE STRUCTURE:
- src/
  - components/
  - pages/
  - services/
  - utils/
  - api/
  - tests/
  - hooks/
  - context/
  - types/

DATABASE DESIGN:
1. Create Supabase tables for:
   - users (profile, preferences, demographics)
   - user_content_interactions (saves, reads, ratings)
   - articles (processed content with AI analysis)
   - categories (taxonomy)
   - user_topics (personalized interests)
2. Implement proper indexing on frequently queried fields
3. Set up row-level security policies for data protection

USER PROFILING:
Build a progressive profiling system that collects:
- Job/Industry
- Location (with geocoding via Radar)
- Financial interests/situation
- Political values (optional, ethically handled)
- Personal interests (climate, AI, geopolitics, etc.)
- News consumption habits
- Communication preferences (notification frequency, format)

PERSONALIZATION ENGINE:
1. Implement a scoring algorithm for content relevance based on:
   - Explicit user preferences
   - Implicit behavior (reading time, clicks, shares)
   - Geographic proximity for local news
   - Topic clustering and user affinity
2. Create feedback loops to improve personalization over time
3. Add explainability features to show why content was recommended

News APIs USE ALL TO MAXIMUM API REQUESTS PER DAY:

1. **TheNewsAPI**
   - URL: https://www.thenewsapi.com
   - API Key: c3AcERwz2ZsZc0ABKhwc00OnAOqhyCKAySAsdiSc
2. **NewsData.io**
   - URL: https://newsdata.io
   - API Key: pub_8217957942c3b5247b479818ae6984adf5333
3. **ApiTube**
   - URL: https://apitube.io
   - API Key: api_live_kbAHRuznY0gfNohekvP6dDFgtYGzy0Afsnwg1hYA1jRvAOrDEblvhq
4. **Breaking News Trends Detection (GitHub Repo)**
   - URL: @https://github.com/oubaidHL/Breaking-News-Trends-Detection
   - No direct API key provided (this is a code repository, not an API service).
5. **NewsAPI**
   - URL: https://newsapi.org
   - API Key: 87ecc9641d894c0396b5495014879e9d

Please implement: Translation software for multilingual support/.

CONTENT PIPELINE:
1. Implement intelligent fetching from news APIs with rate limiting
2. Create processing queue for AI enhancement with priority system
3. Design cache invalidation strategy based on content type and breaking news status
4. Add fallback mechanisms when APIs are unavailable

AI ENHANCEMENT FEATURES:
1. Implement prompt engineering for consistent summarization quality
2. Create specialized prompts for different content types (financial, political, etc.)
3. Add fact-checking integration for controversial topics
4. Implement citation tracking to original sources
5. Design system for human review of AI-generated content when needed

MONETIZATION STRATEGY:
1. Implement freemium model with basic and premium tiers
2. Add subscription management through Stripe
3. Create attribution system for potential revenue sharing with publishers
4. Include newsletter functionality with premium insights

ACCESSIBILITY & COMPLIANCE:
1. Implement WCAG 2.1 AA standards
2. Add keyboard navigation and screen reader support
3. Create data privacy controls for GDPR/CCPA compliance

4. Implement content warning system for sensitive topics
5. Design transparent AI disclosure system

USER ENGAGEMENT:
1. Create notification system with intelligent triggers
2. Implement social sharing with attribution
3. Add save/bookmark functionality with organization options
4. Design daily digest email system
5. Create reading streak and habit-building features

DEPLOYMENT & DEVOPS:
1. Set up CI/CD pipeline with GitHub Actions
2. Implement automated testing requirements (80%+ coverage)
3. Create staging environment for QA
4. Add monitoring for performance and error tracking
5. Implement automated backup system for user data

LAUNCH STRATEGY:
1. Create beta testing program with feedback mechanism
2. Implement feature flagging for gradual rollout
3. Design onboarding A/B tests to optimize conversion
4. Create analytics dashboard for key metrics

This comprehensive approach ensures the platform delivers personalized, contextual news while building a sustainable product with growth potential.

Radar API (NEW)

Live secret (server)	`prj_live_sk_a962f66029924cbd8ebe9914b1d1db2d20ce248b`
Live publishable (client)	`prj_live_pk_1edbf17fac8ded8c1b82c18325bb62ec1aa956b5`
Test secret (server)	`prj_test_sk_c0f3b5843d7185039eb6bf51113602ee726a7e45`

Test publishable (client)	`prj_test_pk_96f89ba996fcca9bc84b26b46203bb8e1161e992`