AI News Poster Agent - Complete Technical Specification

1. Overview

1.1 Product Summary

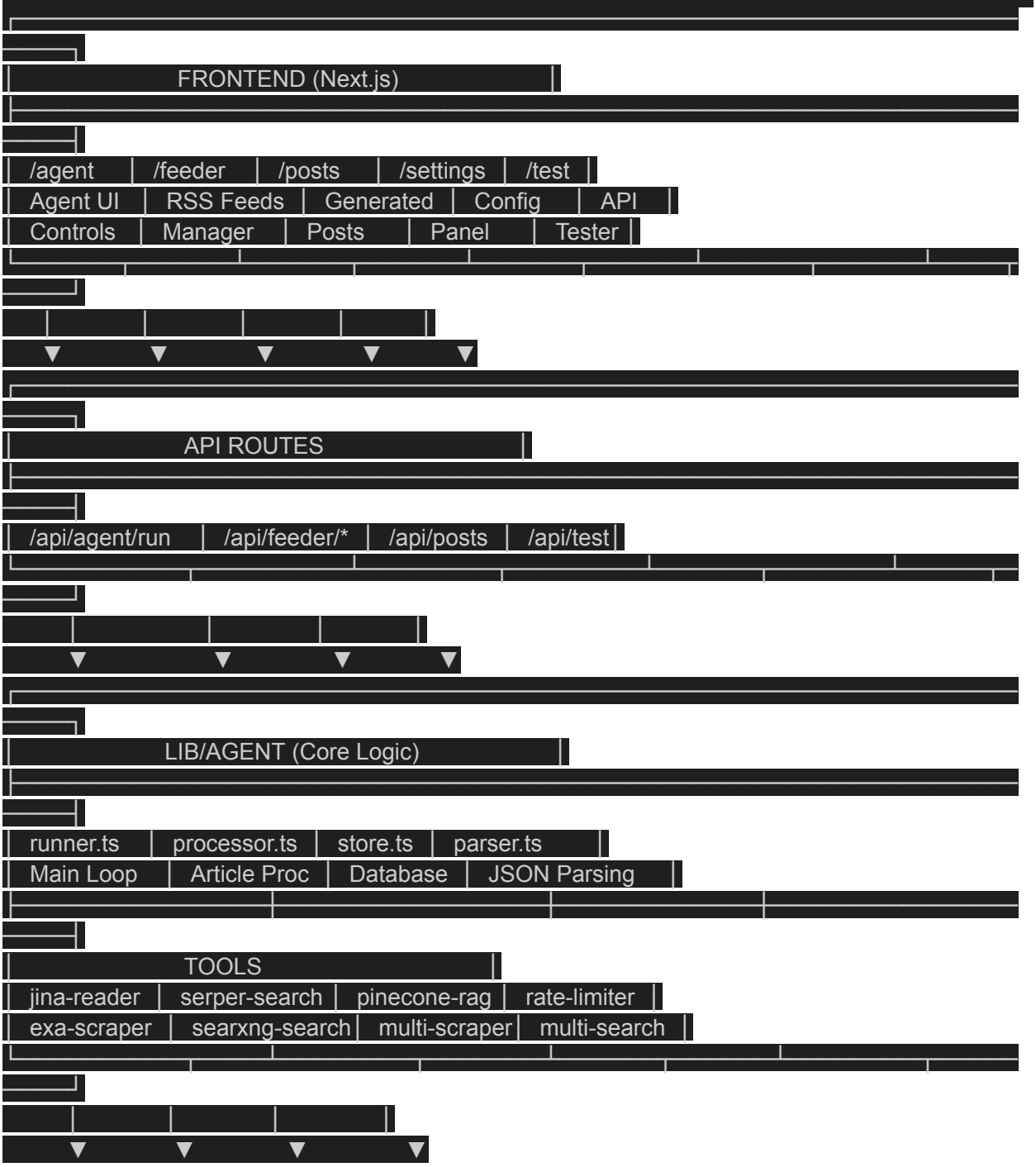
An AI-powered news aggregation and social media post generation system designed for Pakistan news. The agent automatically fetches news from RSS feeds, processes articles using LLM intelligence, and generates platform-specific social media posts (X/Twitter, Instagram, Facebook).

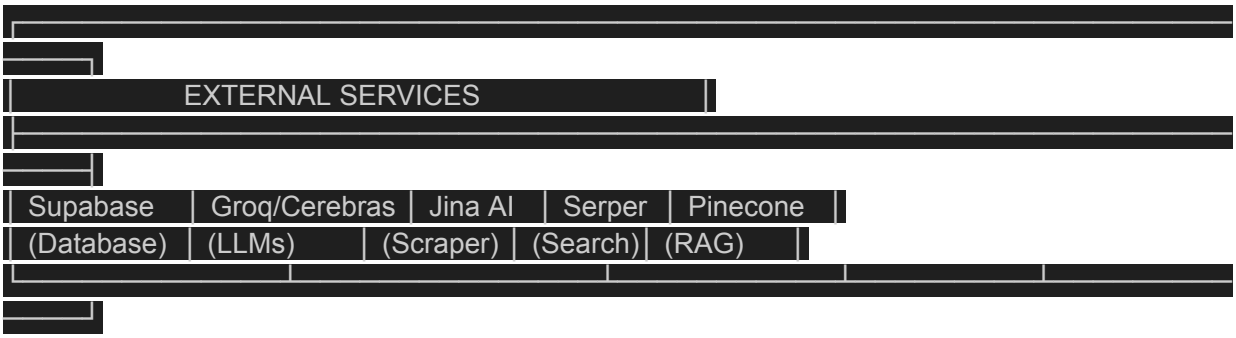
1.2 Tech Stack

Component	Technology
Framework	Next.js 16.1.3 (App Router)
Language	TypeScript 5.x
LLM Framework	LangChain + LangGraph
LLM Providers	Groq, Cerebras
Database	Supabase (PostgreSQL)
Vector DB	Pinecone (RAG)
Scraping	Jina AI, Exa AI
Search	Serper (Google API), SearXNG
Styling	Tailwind CSS 4

Deployment	Vercel

2. System Architecture





---

3. Agent Core Modules

3.1 Module Structure (`lib/agent/`)

File	Purpose	Key Functions
index.ts	Public API exports	Re-exports runAgent, cancelAgent
runner.ts	Main orchestrator	runAgent(), processQueue(), cancelAgent()
processor.ts	Article processing	processArticle(), streamAgentResponse()
store.ts	Database operations	getAgentSettings(), updateQueueItem(), logActivity()
parser.ts	Response parsing	parseAgentResponse(), createFallbackPosts()
prompts.ts	LLM prompts	`SYSTEM_PROMPT`, buildUserMessage()
state.ts	Global state	abort(), isAborted(), createNewAbortController()

agent-factory.ts	LangChain agent	createNewsAgent()
types.ts	TypeScript types	AgentSettings, AgentRun, AgentQueueItem
providers.ts	LLM providers	`getGroqModel()`, `getCerebrasModel()`
groq-llm.ts	Groq integration	Custom LLM wrapper

3.2 Agent Run Lifecycle

YesNoYesNorunAgent CalledLoad Settings from DBApply Provider SettingsCreate Agent Run RecordFetch Unprocessed ArticlesEnqueue ArticlesCreate LangChain AgentFor Each ArticleProcess ArticleStream LLM ResponseTrack Tool CallsParse JSON ResponseValid JSON?Save Generated PostsCreate Fallback PostsMore Articles?Finalize RunUpdate Statistics

---

4. LangChain Agent System

4.1 Agent Creation (

agent-factory.ts)

```
import { createReactAgent } from '@langchain/langgraph/prebuilt';
export function createNewsAgent(model: ChatModel) {
  return createReactAgent({
    llm: model,
    tools: [
      jinaReaderTool,    // Read article content
      serperSearchTool,  // Web search
      newsSearchTool,    // News search
      knowledgeBaseTool, // Pinecone RAG
      findSimilarTool,   // Find similar decisions
      rememberDecisionTool // Store decisions
    ],
  });
}
```

4.2 Tool Definitions (

tools/langchain-tools.ts)

Tool Name	Purpose	Rate Limit
`read_article`	Read full article content via Jina/Exa	20-200 req/min
`search_web`	Google search via Serper/SearXNG	Varies
`search_news`	News-specific search	Varies
`get_guidance`	Query Pinecone knowledge base	No limit
`find_similar`	Find similar past decisions	No limit
`remember_decision`	Store decision in Pinecone	No limit

4.3 Tier System for Processing

The agent uses a **3-tier system** based on information completeness:

Tier	Tools Used	When to Use
Tier 1	None (0 tools)	Title + snippet is sufficient
Tier 2	`read_article`	Need full article content
Tier 3	`read_article` + `search_web`	Need verification/context
Tier 4	Fallback	Parse error, forced generation

---

5. Scraping System

5.1 Multi-Scraper Architecture (

tools/multi-scraper.ts)

5.2 Jina AI Reader (tools/jina-reader.ts)

Mode	Rate Limit	Authentication
Free Tier	20 req/min	None
API Key	200 req/min	`Bearer JINA_API_KEY`

API Endpoint: `https://r.jina.ai/{url}`
Response Format: Clean markdown text with extracted content

5.3 Rate Limiter (tools/rate-limiter.ts)

Provider	Limit	Token Bucket Refill
`jina_free`	20/min	1 token every 3 seconds

`jina_api`	200/min	1 token every 0.3 seconds
`exa`	100/min	1 token every 0.6 seconds

Key Features:

- Token bucket algorithm
- Auto-wait when limit reached
- Logs wait time to console
- Graceful degradation

---

6. Search System

6.1 Multi-Search Architecture (tools/multi-search.ts)

Provider	API	Configuration
Serper	Google Search API	`SERPER_API_KEY` required
SearXNG	Self-hosted meta-search	`SEARXNG_URL` or public instances

6.2 SearXNG Implementation (tools/searxng-search.ts)

Public Fallback Instances:

- `https://searx.be`
- `https://search.sapti.me`
- `https://search.ononoki.org`
- `https://paulgo.io`

Features:

- POST method with form data
- Round-robin instance rotation
- Auto-fallback on 429/403 errors

---

7. Database Schema (Supabase)

7.1 Tables

```
agent_settings
id                UUID PRIMARY KEY
batch_size        INTEGER DEFAULT 10
order_by          TEXT DEFAULT 'pub_date'
order_direction   TEXT DEFAULT 'desc'
model             TEXT DEFAULT 'groq-llama-3.3-70b'
is_active         BOOLEAN DEFAULT false
scraping_provider TEXT DEFAULT 'auto'
search_provider   TEXT DEFAULT 'serper'
auto_run_interval INTEGER DEFAULT 30

agent_runs
id                UUID PRIMARY KEY
started_at        TIMESTAMP
completed_at      TIMESTAMP
articles_processed INTEGER
articles_skipped  INTEGER
posts_generated   INTEGER
errors            JSONB
status            TEXT (running|completed|failed|cancelled)

agent_queue
id                UUID PRIMARY KEY
news_item_id      UUID REFERENCES news_items
run_id            UUID REFERENCES agent_runs
status            TEXT (pending|processing|completed|failed)
decision          TEXT (generate|skip)
reasoning         TEXT
x_post            TEXT
instagram_caption TEXT
facebook_post     TEXT
hashtags          TEXT[]
tool_calls        JSONB

agent_activity
id                UUID PRIMARY KEY
run_id            UUID REFERENCES agent_runs
type              TEXT (info|tool|decision|error|success)
message           TEXT
article_title     TEXT
tool_name         TEXT
```

```
created_at    TIMESTAMP
```

```
news_items
id            UUID PRIMARY KEY
title         TEXT
link          TEXT UNIQUE
content_snippet TEXT
pub_date      TIMESTAMP
source_id     UUID REFERENCES news_sources
is_posted     BOOLEAN DEFAULT false
x_post_content TEXT
instagram_caption TEXT
facebook_post   TEXT
```

---

8. LLM Configuration

8.1 Available Models

Model ID	Provider	Description	Speed
`groq-gpt-oss-120b`	Groq	Powerful reasoning	Medium
`groq-llama-3.3-70b`	Groq	Balanced	Fast
`groq-llama-3.1-8b`	Groq	Fast	Very Fast
`cerebras-gpt-oss-120b`	Cerebras	Ultra-fast	2200 tok/s
`cerebras-llama-3.3-70b`	Cerebras	Ultra-fast	1100 tok/s

8.2 System Prompt Structure

The agent uses a detailed system prompt with:

- **Tier System** explanation
- **Decision Process** steps
- **Output Format** (JSON schema)

- **Post Guidelines** per platform
- **Critical Rules** (always generate, never skip)

---

9. Post Generation

9.1 Output Schema

```
interface GeneratedPosts {
  decision: 'generate' | 'skip';
  reasoning: string;
  x_post?: string;       // Max 280 chars
  instagram_caption?: string;
  facebook_post?: string;
  hashtags?: string[];
}
```

9.2 Platform Requirements

Platform	Character Limit	Tone	Format
X/Twitter	280 chars	Concise, urgent	`📰 headline
Instagram	2200 chars	Engaging, emoji-rich	📰 `headline\n\ndetails\n\n#tags`
Facebook	No limit	Professional, full paragraph	Complete news summary

9.3 Fallback Generation

If LLM parsing fails or tries to skip:

```
function createFallbackPosts(article): GeneratedPosts {
  return {
    xPost: `📰 ${title.slice(0,200)} | ${source} #PakistanNews`,
    instagram: `📰 ${title}\n\n📍 Source: ${source}\n\n#PakistanNews`,
    facebook: `${title}\n\nSource: ${source}`,
    tierUsed: 4  // Fallback tier
  };
}
```

---

10. API Endpoints

10.1 Agent APIs (`/api/agent/`)

Endpoint	Method	Purpose
`/api/agent/run`	POST	Start agent processing
`/api/agent/status`	GET	Get current run status

10.2 Feeder APIs (`/api/feeder/`)

Endpoint	Method	Purpose
`/api/feeder/sources`	GET/POST	Manage RSS sources
`/api/feeder/settings`	GET/PATCH	Feeder settings

10.3 Test API (`/api/test/`)

Endpoint	Method	Tests
`/api/test`	GET	Environment variable status
`/api/test`	POST	Test individual APIs (Serper, Jina, SearXNG)

---

11. Environment Variables

11.1 Required Variables

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=
NEXT_PUBLIC_SUPABASE_ANON_KEY=
# LLM Providers
GROQ_API_KEY=
CEREBRAS_API_KEY=
```

```
# Search
SERPER_API_KEY=
# Vector DB
PINECONE_API_KEY=
PINECONE_INDEX_NAME=
```

11.2 Optional Variables

```
# Scraping (optional, free tier works without)
JINA_API_KEY=
EXA_API_KEY=
# Alternative Search
SEARXNG_URL=
```

---

12. Error Handling

12.1 Error Types

```
interface AgentError {
  article_id: string;
  error: string;
  timestamp: string;
}
```

12.2 Retry Logic

- **Rate Limit (429):** Wait and retry (auto-wait in rate limiter)
- **Timeout:** Mark as failed, continue to next
- **Parse Error:** Use fallback post generation
- **Aborted:** Graceful cancellation via AbortController

---

13. Deployment

13.1 Vercel Configuration (

vercel.json)

```
{
  "functions": {
    "app/api/agent/run/route.ts": { "maxDuration": 60 },
    "app/api/agent/status/route.ts": { "maxDuration": 30 },
    "app/api/test/route.ts": { "maxDuration": 30 }
  },
```

```
  "regions": ["sin1"]
}
```

13.2 Requirements

- **Vercel Pro** recommended for 60s function timeout
- **Supabase** project with all tables created
- **Pinecone** index for RAG functionality

---

14. Version History

Version	Date	Changes
1.0.0	Jan 2026	Initial release with LangChain agent
1.1.0	Jan 2026	Added multi-scraper with Jina/Exa fallback
1.2.0	Jan 2026	Added SearXNG support, rate limiting
1.3.0	Jan 2026	Modular refactoring, Cerebras support