Product Specification: Armada Den Workspace

* **Version:** 1.1
* **Status:** Draft
* **Author:** Natnael
* **Date:** November 6, 2025

1. Introduction & Vision
1.1. The Problem
The modern knowledge worker operates in a state of fragmentation. Information is scattered across email, documents, and the web. Tasks are initiated in one app (e.g., WhatsApp), researched in another (e.g., Google), and executed in a third (e.g., Gmail). This constant context-switching is a significant drain on productivity and focus.
Current AI tools exacerbate this:

* Conversational LLMs (like ChatGPT) are excellent for creative generation but are "un-grounded" and disconnected from real-time information and personal data.

* Factual Engines (like Perplexity) are excellent for grounded, cited search but cannot take action.

* Productivity Apps (like Gmail) are where work happens but lack intelligent, proactive assistance.

1.2. The Solution
Armada Den is an internal AI workspace built to enhance productivity, research, and communication within Hire Armada. It unifies AI chat, real-time knowledge access, and Gmail-based communication into one secure workspace — purpose-built for internal operations and client correspondence.
It combines three core pillars into one seamless experience:

* A Conversational AI: An intelligent assistant to chat, draft, and brainstorm.

* A Grounded Factual Engine: A real-time, cited web search to provide accurate, verifiable answers.

* An Integrated Action Layer: A secure connection to the user's productivity tools (starting with Gmail) to execute real-world tasks.

All of this is unified by a Persistent Memory Layer, allowing Armada Den to learn user preferences, recall context from past conversations, and access relevant data from integrated apps to provide truly personal and effective assistance.
1.3. Vision Statement

To create a single, context-aware AI workspace that seamlessly integrates knowledge retrieval, content creation, and task execution, empowering users to be more focused, creative, and productive.

1.4. Target Audience (Hire Armada Employees)

* Recruiting: Summarize candidate emails, generate outreach drafts.

* Delivery: Individuals who handle diverse roles (sales, marketing, operations) and need an efficient assistant to manage email, research, and automate repetitive tasks.

* BizDev: Research leads, compose intro emails.

* Leadership: Search internal documentation, summarize strategy threads.

2. Scope

2.1. In-Scope (MVP)

* Core conversational AI for natural language interactions (GPT-4o).

* Real-time web search with citations and summaries (Bing/SerpApi).

* fastapi-users authentication with Google OAuth2.

* Gmail integration for reading (summarizing) and sending (drafting, confirming) emails.

* Persistent memory for conversation history (RAG).

* Basic UI/UX elements: chat interface, email composer, and settings for data deletion.

2.2. Out-of-Scope (MVP)

* Integrations with other email providers (e.g., Outlook) or tools (e.g., Slack, Notion, Calendar).

* No other tool integration like slack, github, etc. on v1.

* Advanced enterprise features (e.g., multi-user collaboration, custom models, audit trails).

3. Functional Requirements

3.1. Conversational Core

* Description: Natural language chat powered by LLM for answering questions, generating text, and delegating to tools.

* User Story: "As a user, I want to have a natural, fluid conversation with the AI, so that I can ask questions, brainstorm, and get help with my work."

* Inputs/Behaviors/Outputs:

  * Input: User text query in the command bar.

  * Behavior: LLM processes the query. If no tools are needed, it generates a text response.

  * Output: Streaming text response in the chat UI, with support for Markdown, code blocks, lists, and a "copy to clipboard" button.

3.2. Real-Time Web Search (Grounded Factual Engine)

* Description: Automatically fetch and summarize live web content with citations when a query is factual.

* User Story: "As a researcher, I want to ask about current events or technical topics and get accurate, verifiable answers, so that I can trust the information I'm receiving."

* Inputs/Behaviors/Outputs:

  * Input: A user query identified as "factual" (e.g., "latest news on...", "who won...").

  * Behavior: System triggers a search API (Bing/SerpApi) to get top results. Results are passed to the LLM as context for synthesis.

  * Output: A summarized answer with in-line citations (e.g., [1], [2]). A source viewer allows users to inspect the source URL, title, and snippet for each citation.

3.3. Gmail Integration (Action Layer)

* Description: Securely connect a user's Gmail account via OAuth to allow the AI to read and send emails.

* User Story (Read): "As a busy manager, I want to ask 'summarize my unread emails from this morning,' so I can quickly get up to speed without opening my inbox."

* User Story (Write): "As a marketer, I want to say 'draft an email to client@... about our new feature,' so the AI can generate the first draft for me to review."

* Acceptance Criteria (CRITICAL): The AI cannot send email without explicit user confirmation. The user must be able to edit all fields and manually click a "Send" button in the Email Composer UI to execute the gmail.send action.

3.4. Persistent Memory (RAG)

* Description: Store and recall context from past conversations to provide more personal and relevant assistance.

* User Story: "As a consultant, I want the AI to remember our previous discussion about 'Project X,' so I can ask follow-up questions without repeating myself."

* Inputs/Behaviors/Outputs:

  * Input: All conversations are saved automatically.

  * Behavior: Conversations are chunked, embedded, and stored in a vector database (e.g., Supermemory or Pinecone). New queries retrieve relevant context and inject it into the LLM prompt.

  * Output: The AI's response is context-aware (e.g., "You mentioned last week that Project X...")

4. Non-Functional Requirements (NFRs)

* Performance: Chat response (non-tool) latency: < 2 seconds. Web search + synthesis latency: < 5 seconds. Support 20-30 concurrent users at launch.

* Security: Authentication powered by fastapi-users with Google OAuth2 provider. JWT access tokens + HTTP-only refresh cookies (secure, SameSite=Lax). All transport over HTTPS. Rate limiting & brute-force protection via fastapi-users plugins.

* Scalability: Horizontal scaling via containerization (e.g., Cloud Run, Fargate) for the backend. Serverless frontend hosting (e.g., Vercel).

* Reliability & Availability: 99.9% uptime SLA. Automated database backups (daily). Graceful error handling and retries for external API calls.

* Usability & Accessibility: WCAG 2.1 AA compliant. Intuitive, clean UI with keyboard navigation and screen reader support. Responsive design (desktop-first, mobile-usable).

5. Architecture & System Design
5.1. System Overview
A decoupled web application:

* Frontend (Next.js): A responsive client that handles all UI rendering.

* Backend API (FastAPI): A Python server that manages auth, user data, conversation history, and orchestrates all LLM and tool calls.

* Databases (PostgreSQL + Vector DB): PostgreSQL for relational data (users, sessions) and a Vector DB (Pinecone, Supermemory) for RAG.

* External APIs: OpenAI, Google (Auth/Gmail), Bing/SerpApi, Composio (Tool Agent).

Component	Technology	Rationale
Frontend	Next.js (React)	Ensures fast performance via SSR/SSG, providing a great developer experience and easy deployment via Vercel.
UI Kit	Tailwind CSS + shadcn/ui	Enables rapid, component-based development and guarantees a modern, accessible, and responsive user interface design.
Backend API	FastAPI (Python)	A modern, async-ready framework ideal for LLM and API orchestration, offering built-in support for type hints, Pydantic validation, and automatic OpenAPI documentation.
Data Models	SQLAlchemy + Pydantic	SQLAlchemy provides a robust ORM, while Pydantic ensures strict data validation and type consistency.
Primary Database	PostgreSQL	A reliable, relational database choice, supporting strong data integrity and scalability.
Memory Layer	Supermemory / Pinecone (Vector DB)	Managed vector database solution to enable efficient RAG context retrieval for personalizing AI responses.
Authentication	fastapi-users + fastapi-users[google,oauth] + PostgreSQL adapter	Battle-tested, fully async, built-in email/password + OAuth, automatic user table, reset-password flows, verifiable JWTs.
LLM Orchestration	Composio Agents	Manages tool-calling complexity, streamlines API integrations, and orchestrates structured, multi-step LLM workflows.
Hosting (Frontend)	Vercel	Provides serverless hosting for rapid global deployment, excellent performance, and built-in CI/CD.
Hosting (Backend)	Heroku	Ensure fast deployment, easy scaling of containers (dynos), and unified management across development and initial production environments.
Hosting (Database)	Heroku Postgres	Provides a fully managed, scalable PostgreSQL service directly integrated with the Heroku platform, simplifying setup and deployment stability for the MVP phase.

6. User Experience (UX) Flows
6.1. Flow 1: First-Time User Onboarding

1. User lands on the marketing page and clicks “Sign in with Google.”

2. Redirected to `/auth/google/authorize` (fastapi-users OAuth router).

3. Google consent screen shows only required scopes:
  - `openid` `email` `profile`
  - `https://www.googleapis.com/auth/gmail.readonly`
  - `https://www.googleapis.com/auth/gmail.send`
  - `https://www.googleapis.com/auth/gmail.compose`

4. After consent → callback → fastapi-users creates/updates user → sets refresh cookie + JWT.

5. Frontend reads JWT from `Authorization` header (SSR) or cookie → redirects to `/app`.

6. Modal: “Connect Armada Den to your apps” → “Connect Gmail” (already granted scopes, just UI confirmation).

7. User lands in main chat with welcome message: “Hello! Ask me anything, or try ‘Summarize my unread emails.’”

6.2. Flow 2: Research-to-Email Task

1. User types: "What are the latest advancements in solid-state battery technology?"

2. Armada Den identifies this as factual and shows "Searching the web..."

3. Response: "Here are the latest advancements... [Summary] [1] [2]."

4. User (Follow-up): "Okay, draft an email to my colleague bob@... summarizing these points."

5. The Email Composer (3.3) UI appears, pre-filled with a draft to Bob, summarizing the research.

6. User reviews the draft, makes a small edit, and clicks "Send."

8. Milestones & Deliverables
Week 1: Core Chat, Auth & Foundations
Deliverables:

* Next.js frontend + FastAPI backend connected

* fastapi-users with Google OAuth2 fully wired (login, logout, session persistence)

* User model + PostgreSQL `users` table auto-migrated via Alembic.

* Basic chat UI and message flow (no external tools yet)

* Set up infra for persistence (DB + API models)

Owners: Frontend & Backend Engineers

Week 2: Search, Memory & Gmail Integration
Deliverables:

* Web Search integration (Pillar 2)

* Persistent Memory (Pillar 4)

* Gmail integration + Email Composer (Pillar 3) using user-associated Google tokens stored via fastapi-users OAuth state.

* End-to-end testing and Beta-ready prototype

Owners: Frontend & Backend Engineers
Target: Launch MVP by 10-12-2025.

9. Glossary

* LLM: Large Language Model (e.g., GPT-4o).

* RAG: Retrieval-Augmented Generation – Enhancing an LLM with external data (from our memory DB).

* OAuth 2.0: Authorization framework for secure third-party access (e.g., to Gmail).

* MVP: Minimum Viable Product.

* NFR: Non-Functional Requirement (e.g., performance, security).

* WCAG: Web Content Accessibility Guidelines.

* fastapi-users – Open-source authentication library for FastAPI providing ready-made user management, JWT/cookie auth, OAuth2 clients, password reset, etc.