Wizards MarketAI 360 – Master Product & Technical Specification

1. Executive Summary and Strategic Vision

1.1. Introduction

The marketing paradigm of 2025 has shifted decisively from human-centric execution to agentic orchestration. The complexity of modern digital ecosystems—spanning fragmented social algorithms, generative search engines, and real-time programmatic bidding—has outpaced the capacity of traditional human teams to manage efficiently. **Wizards MarketAI 360** is conceptualized as the definitive answer to this scalability crisis. It is an internal agency platform designed not merely to assist marketers but to function as a "Self-Driving Agency," leveraging a workforce of **267+ Autonomous Agents** to manage multi-brand marketing across seven critical verticals: Social, SEO, Web, Sales, WhatsApp, LinkedIn, and Performance.[1]

This Master Specification outlines the functional, technical, and implementation details for Wizards MarketAI 360. The platform represents a convergence of advanced agentic protocols—specifically the **WAI SDK**, **A2A (Agent-to-Agent)**, **ROMA (Recursive Optimization & Marketing Autonomy)**, and **MCP (Model Context Protocol)**—to create a system capable of reducing operational run-costs by 80-90% while accelerating time-to-market by 50%.[2]

1.2. Strategic Objectives

The primary objective of Wizards MarketAI 360 is to transition the agency model from "billing for hours" to "billing for outcomes" by decoupling throughput from human headcount. The platform aims to achieve the following strategic goals:

- **Operational Autonomy:** Implement the **ROMA Framework** to enable agents to operate at **L1-L4 autonomy levels**, moving from simple task execution to complex, multi-step strategic decision-making without constant human oversight.[1]
- **Universal Interoperability:** Eliminate data silos between advertising platforms (Google, Meta, LinkedIn) and internal tools by utilizing the **Model Context Protocol (MCP)** as a standardized connective tissue.[3]
- **Hybrid Intelligence:** Leverage a "best-of-breed" model approach by integrating **Claude Code Agents** for high-fidelity technical tasks (Web, Dev) and **OpenAI Agents SDK** for high-concurrency conversational tasks (Social, WhatsApp).[5]
- **Future-Proofing:** Adopt a **Quantum Security Framework** to protect proprietary agent strategies and client data against emerging post-quantum cryptographic threats.[1]

2. Technical Architecture & Stack

2.1. System Overview

The platform is built as a modular, event-driven monolith (to start) that can easily scale into microservices. It relies on the **WAI SDK** as the central nervous system.

- **Frontend (The "God Mode" Dashboard):**
  - **Framework:** Next.js 14 (React) + Tailwind CSS.
  - **UI Protocol: AG-UI** (Agent-Generative UI) to render dynamic interfaces based on agent context.[7]
  - **State Management:** Zustand + TanStack Query.
- **Backend (The Orchestrator):**
  - **Runtime:** Python 3.11 (FastAPI) for core logic and AI processing.
  - **Orchestration: WAI SDK Core** (Internal Library).
  - **Task Queue:** Redis + Celery (for asynchronous agent tasks like scraping or report generation).
- **Data Layer:**
  - **Primary DB:** PostgreSQL (Supabase or Neon) for structured data (Users, Brands, Campaigns, Leads).
  - **Vector DB:** Pinecone or Weaviate for **Agent Memory** (Knowledge Base, Brand Voice, Past Strategies).
  - **Cache:** Redis for session state and real-time bidding signals.

2.2. Protocol Implementation

- **MCP (Model Context Protocol):**
  - **Server:** A centralized mcp-router service that proxies requests to external APIs (Google Ads, Meta Ads, LinkedIn).
  - **Authentication:** Unified OAuth2 management for all connected 3rd-party platforms.
- **A2A (Agent-to-Agent):**
  - **Bus:** Redis Pub/Sub channel wizards.a2a where agents broadcast requests (e.g., "Need Creative") and listen for assignments.
  - **Schema:** Strict JSON payloads defining intent, payload, priority, and roma_level.

---

3. Database Schema Architecture

To manage 360-degree marketing at scale, the database must support multi-tenancy (Brands) and hierarchical agent permissions.

3.1. Core Tables (SQL)

SQL

```sql
-- MULTI-TENANCY & USERS
TABLE brands (
  id UUID PRIMARY KEY,
  name VARCHAR,
  industry VARCHAR,
  settings JSONB, -- { "tone": "professional", "colors": ["#fff", "#000"] }
  subscription_tier VARCHAR
);

TABLE users (
  id UUID PRIMARY KEY,
  brand_id UUID REFERENCES brands(id),
  role VARCHAR, -- 'admin', 'viewer', 'approver'
  permissions JSONB
);

-- AGENT FRAMEWORK
TABLE agents (
  id UUID PRIMARY KEY,
  name VARCHAR, -- 'Nexus', 'Prism'
  tier VARCHAR, -- 'Executive', 'Creative', 'Domain'
  capabilities JSONB, -- ["seo_audit", "content_gen"]
  model_config JSONB, -- { "provider": "anthropic", "model": "claude-3.5-sonnet" }
  roma_level INT -- 1 to 4
);

-- WORKFLOW & TASKS
TABLE campaigns (
  id UUID PRIMARY KEY,
  brand_id UUID REFERENCES brands(id),
  vertical VARCHAR, -- 'SEO', 'Social', 'Performance'
  status VARCHAR,
  budget DECIMAL,
  kpi_targets JSONB -- { "roas": 4.0, "cpa": 15.0 }
);

TABLE tasks (
  id UUID PRIMARY KEY,
```

```
  campaign_id UUID REFERENCES campaigns(id),
  assigned_agent_id UUID REFERENCES agents(id),
  parent_task_id UUID, -- For A2A sub-tasks
  status VARCHAR, -- 'pending', 'review', 'completed'
  input_context TEXT,
  output_artifact TEXT, -- Markdown, Code, or JSON
  approval_status VARCHAR -- 'auto', 'waiting_human', 'approved'
);

-- PERFORMANCE DATA
TABLE analytics_snapshots (
  id UUID PRIMARY KEY,
  brand_id UUID,
  source VARCHAR, -- 'google_ads', 'linkedin_organic'
  metrics JSONB, -- { "impressions": 1000, "clicks": 50 }
  timestamp TIMESTAMP
);
```

---

4. Vertical Workflows & KPIs

4.1. Vertical 1: Social Media (The Viral Engine)

- **Agents:** Trend Watcher (Domain), Content Ideation (Creative), Visual Production (Creative/Nanobana), Scheduling (Executive).
- **Workflow:**
  1. **Trend Watcher** scans X/TikTok for trending audio/hashtags relevant to brand.industry.
  2. **Ideation Agent** maps trends to brand pillars and proposes 3 concepts.
  3. **Visual Agent** invokes **Nanobana** to generate high-fidelity memes/images.
  4. **ROMA Check:** If L2, pause for human approval. If L3+, auto-schedule via **Social MCP**.
- **KPIs:** Viral Velocity (shares/hour), Engagement Rate, Sentiment Score.

4.2. Vertical 2: SEO & GEO (Generative Engine Optimization)

- **Agents:** GEO Auditor (Domain), Authority Architect (Creative), Programmatic SEO (Dev).
- **Workflow:**
  1. **GEO Auditor** prompts Perplexity/ChatGPT with "Best [Industry] in [City]" to check citation visibility.
  2. **Authority Architect** rewrites "About Us" and service pages with structured data, TL;DRs, and direct quotes to maximize LLM retrieval.[8]

3. **Programmatic Agent** generates 100+ location-based landing pages using **Claude Code**, deploying directly to WordPress via MCP.[9]

- **KPIs:** Share of Model (SoM - frequency of citation in AI answers), Organic Traffic, Domain Authority.

4.3. Vertical 3: Web (Generative UI & Dev)

- **Agents:** UX Designer (Creative/Gemini 3), Frontend Dev (Dev/Claude), QA Bot (QA).
- **Workflow:**
  1. User prompts: "Need a landing page for our Black Friday sale."
  2. **UX Designer** uses **Gemini 3** to generate a "vibe-coded" interactive mockup.[10]
  3. **Frontend Dev** converts mockup to clean React/Tailwind code.
  4. **QA Bot** runs Playwright tests via **MCP** to ensure mobile responsiveness.[11]
- **KPIs:** Page Load Speed (Core Web Vitals), Conversion Rate, Time-to-Deploy.

4.4. Vertical 4: Sales (Autonomous SDR)

- **Agents:** Prospector (Domain), Personalizer (Creative), Outreach Manager (Executive).
- **Workflow:**
  1. **Prospector** scrapes LinkedIn/Apollo for leads matching ICP.
  2. **Personalizer** analyzes prospect's last 5 posts to generate a custom "icebreaker" intro.
  3. **Outreach Manager** sequences emails/DMs via **HubSpot MCP**. If a reply is received, it classifies intent ("Interested", "Not Now", "Unsubscribe").[12]
- **KPIs:** Meeting Booked Rate, Response Rate, Pipeline Value Generated.

4.5. Vertical 5: WhatsApp (Community & Commerce)

- **Agents:** Community Manager (Creative), Gamification Engine (Dev), Support Concierge (Domain).
- **Workflow:**
  1. **Concierge** handles L1 support queries (24/7) using RAG on brand docs.
  2. **Gamification Engine** triggers weekly quizzes or "Spin the Wheel" offers to drive engagement.[13]
  3. **Community Manager** summarizes group sentiment daily and alerts humans to potential PR crises.
- **KPIs:** Response Time, Community Retention Rate, Commerce Conversion Rate (via chat).

4.6. Vertical 6: LinkedIn (B2B Authority)

- **Agents:** Voice Cloner (Creative), Engagement Rig (Domain), Networker (Executive).
- **Workflow:**
  1. **Voice Cloner** ingests CEO's past blogs to draft thought leadership posts.
  2. **Networker** auto-connects with 20 prospects/day with a blank or highly specific note

(avoiding spammy templates).
3. **Engagement Rig** comments on top industry influencer posts within 5 minutes of publication to hijack visibility.[14]

- **KPIs:** Profile Views, Connection Acceptance Rate, SSI (Social Selling Index) Score.

4.7. Vertical 7: Performance Marketing (Ads)

- **Agents:** Data Analyst (Domain), Bid Adjuster (Executive), Creative Iterator (Creative).
- **Workflow:**
  1. **Data Analyst** pulls cross-channel reports (Google, Meta, LinkedIn) via **MCP**.
  2. **Bid Adjuster (ROMA L4)** detects "Meta CPA > Target" and "Google CPA < Target." It autonomously moves 20% of the daily budget to Google (Arbitrage).[15]
  3. **Creative Iterator** identifies winning ad visual, asks Visual Agent to make 5 variations, and hot-swaps the losers.
- **KPIs:** ROAS (Return on Ad Spend), CPA (Cost Per Acquisition), CAC (Customer Acquisition Cost).

---

5. User Experience: AG-UI & "God Mode"

The interface is not a static SaaS dashboard. It utilizes **AG-UI** to generate interfaces on the fly.

5.1. The "God Mode" Dashboard

- **Chat-First Interface:** The central view is a chat window with the "Chief of Staff" agent.
- **Generative Widgets:**
  - User: "Show me how our Black Friday ads are doing."
  - System: Generates a temporary React component showing a real-time comparison graph of Meta vs. Google ROAS, with a "Pause Underperforming Ads" button embedded directly in the chat.
- **Agent Feed:** A "Matrix-style" scrolling sidebar log showing live agent actions (e.g., *Crawling site... Posting to X...*), providing transparency and trust.

5.2. Admin & Onboarding

- **Brand Onboarding:** A wizard where the admin inputs the brand URL. The **Audit Agent** immediately crawls the site, social profiles, and SEO standing to auto-populate the "Brand Knowledge Base."
- **Permissions:** Granular access control. Admin grants users access to specific Verticals (e.g., "Copywriter" user only sees Social & SEO tools).

---

6. Technical Implementation Guide (Replit Strategy)

Since the backbone is the **WAI SDK** and the development environment is **Replit**, the following prompt acts as a "Master Bootstrapper" to initialize the project structure, dependencies, and core files.

6.1. The "Replit Master Prompt"

Copy and paste this into the Replit Agent or Chat to kickstart development:

> Context: We are building "Wizards MarketAI 360", an internal marketing agency platform.
> Backbone: We already have the wai-sdk orchestration logic in a local directory named wai_sdk. This SDK manages 267 agents, 23 LLMs, and utilizes A2A, MCP, and ROMA protocols.
> Goal: Create the full application structure around this SDK, including the FastAPI backend, Next.js frontend, and PostgreSQL database connection.
> **Instructions:**
>
> 1. **Directory Structure:**
>   - Create a backend/ folder for the Python FastAPI app.
>   - Create a frontend/ folder for the Next.js 14 application.
>   - Ensure wai_sdk/ is treated as a core library (mock it if files are missing for now, but assume it exports an Orchestrator class).
> 2. **Backend Setup (Python/FastAPI):**
>   - Initialize a FastAPI app in backend/main.py.
>   - Create API routes for the 7 verticals:
>   - /api/verticals/social
>   - /api/verticals/seo
>   - /api/verticals/web
>   - /api/verticals/sales
>   - /api/verticals/whatsapp
>   - /api/verticals/linkedin
>   - /api/verticals/performance
>   - Create a database.py using SQLAlchemy to connect to a PostgreSQL database. Define the Brands, Campaigns, and Tasks tables as per the schema:
>   - Brands: id (uuid), name, settings (json).
>   - Campaigns: id, brand_id, vertical, status, kpi_targets (json).
>   - Tasks: id, agent_id, status, roma_level, payload.
>   - Implement an endpoint /api/orchestrate that accepts a prompt, instantiates the wai_sdk.Orchestrator, and delegates the task to the correct agent tier.
> 3. **Frontend Setup (Next.js/React):**

    - Initialize a Next.js app in frontend/.
    - Create a "God Mode" Dashboard layout with a Sidebar (Navigation), Main Chat Area (AG-UI container), and Right Sidebar (Live Agent Activity Feed).
    - Create a useAgent hook that connects to the backend streaming endpoint to render real-time agent thoughts.
4. **Integrations (Mock implementations for now):**
    - Create a services/mcp_router.py in the backend. Add stub functions for google_ads_tool, meta_ads_tool, and wordpress_tool that return dummy JSON data.
5. **Environment:**
    - Create a .env.example file listing required keys: OPENAI_API_KEY, ANTHROPIC_API_KEY, DATABASE_URL, REDIS_URL, WAI_SDK_LICENSE_KEY.

Execution: Please generate the file structure, the main.py entry point, the database models, and the Next.js dashboard page code.

6.2. Implementation Roadmap

1. **Phase 1 (Foundation):** Use the prompt above to scaffold the app. Connect the real PostgreSQL DB. Ensure the wai_sdk can be imported.
2. **Phase 2 (Vertical Logic):** Flesh out the /api/verticals/ endpoints. For "Social," implement the logic to call the *Trend Watcher Agent* from the SDK.
3. **Phase 3 (MCP Connection):** Replace the mock MCP router with actual API calls to Google/Meta/LinkedIn using their respective SDKs or MCP server implementations.
4. **Phase 4 (UI Polish):** Implement the AG-UI rendering in React. Allow the backend to send JSON that the frontend renders as Charts, Tables, or Forms dynamically.

---

7. Conclusion

Wizards MarketAI 360 is not merely a tool update; it is a foundational shift in the operational model of the digital agency. By codifying the roles of 267+ specialized agents and enabling them to collaborate via the A2A protocol and interact with the world via MCP, the platform creates a scalable, self-improving marketing engine. The strategic integration of the ROMA protocol ensures that this powerful autonomy remains safely tethered to human intent, providing the perfect balance between machine speed and human strategy. This architecture positions Wizards Tech Global to lead the industry into the Agentic AI era, delivering outcomes with a speed and efficiency that traditional models cannot match.

Table: Protocol & Technology Summary

Technology	Function	Role in Wizards MarketAI 360
WAI SDK	Orchestration	Manages 267+ agents, model routing (23+ LLMs), and cost optimization.
A2A	Communication	Enables agents to delegate tasks and collaborate (e.g., SEO agent asks Creative agent for assets).
MCP	Integration	Standardizes connection to external tools (Google Ads, Meta Ads, CRMs, CMS).
ROMA	Governance	Defines trust levels (L1-L4) for agent actions (e.g., L4 can spend budget autonomously).
AG-UI	Interface	Allows agents to render custom, generative UI components to the human user.
BMAD	Methodology	Structures the agile development workflow for the Web/Development vertical agents.
Nanobana	Creative AI	Provides consistent, high-fidelity image generation and UI mockups.
Swarm	Architecture	Manages high-volume, lightweight agent clusters for Social and WhatsApp

		tasks.

Works cited

1. README.md
2. Wizards Tech, accessed December 6, 2025, https://www.wizardstechglobal.com/
3. Introducing the Model Context Protocol - Anthropic, accessed December 6, 2025, https://www.anthropic.com/news/model-context-protocol
4. Model Context Protocol (MCP): Boosting AI in Marketing Workflows, accessed December 6, 2025, https://www.cmswire.com/digital-marketing/model-context-protocol-mcp-boosting-ai-in-marketing-workflows/
5. How I Use Claude To Build Launch Plans From Chaos - Product Marketing Hive, accessed December 6, 2025, https://www.productmarketinghive.com/how-i-use-claude-to-build-launch-plans-from-chaos/
6. Swarm: The Agentic Framework from OpenAI - Fluid AI, accessed December 6, 2025, https://www.fluid.ai/blog/swarm-the-agentic-framework-from-openai
7. Step-by-step guide for integrating Microsoft Agent Framework with AG-UI - Medium, accessed December 6, 2025, https://medium.com/data-science-collective/step-by-step-guide-for-integrating-microsoft-agent-framework-with-ag-ui-6f63f4ba9db1
8. Generative engine optimization - Wikipedia, accessed December 6, 2025, https://en.wikipedia.org/wiki/Generative_engine_optimization
9. 2025 Programmatic SEO Playbook: AI, Real-Time Data, and Market Domination - GrackerAI, accessed December 6, 2025, https://gracker.ai/blog/2025-programmatic-seo-playbook-ai-real-time-data-and-market-domination
10. Generative UI: A rich, custom, visual interactive user experience for any prompt, accessed December 6, 2025, https://research.google/blog/generative-ui-a-rich-custom-visual-interactive-user-experience-for-any-prompt/
11. AI Agents and MCP Servers — The Next Frontier of Autonomous DevOps | by Shrinidhi Kulkarni | Oct, 2025, accessed December 6, 2025, https://medium.com/@davidcesc/ai-agents-and-mcp-servers-the-next-frontier-of-autonomous-devops-ffdb1a4f0ebd
12. HubSpot MCP Server, accessed December 6, 2025, https://developers.hubspot.com/mcp
13. Level Up Your User Engagement: Leveraging WhatsApp Chatbots for Gamification, accessed December 6, 2025, https://www.autowhat.app/post/level-up-your-user-engagement-leveraging-whatsapp-chatbots-for-gamification
14. Model Context Protocol (MCP) server for LinkedIn API integration - GitHub, accessed December 6, 2025, https://github.com/Dishant27/linkedin-mcp-server

15. How Real-Time Ad Tech Platforms Use AI to Optimize Bidding - Madgicx, accessed December 6, 2025, https://madgicx.com/blog/real-time-ad-tech-platform-for-bidding