Hyper-ADS LLD

Background

We're building the HYPER ADS Advertising Advisor to help small businesses capitalize on local events and weather conditions for better advertising decisions. Small business owners often miss revenue opportunities because they don't have time to monitor local happenings or analyze how weather affects customer behavior.

Our solution automatically scrapes EventBrite for local events, collects weather data through a dedicated MCP server, and uses three AI agents (Data Analyst, Weather Analyst, Marketing Strategist) powered by Google Gemini to generate personalized advertising recommendations. The system runs on AWS infrastructure and delivers daily recommendations via email at 7 AM Central Time.

Instead of generic advice, businesses receive specific suggestions like "advertise cold drinks today - it's 85°F with a 2,000-person outdoor festival nearby."

Problems We're Solving

- Information Overload: Small business owners juggle multiple responsibilities and can't constantly monitor event websites and weather forecasts to optimize their marketing.
- Missed Revenue Opportunities: Businesses learn about profitable events too late or don't understand how to capitalize on weather patterns that drive customer behavior.
- Lack of Data-Driven Marketing: Most small businesses rely on intuition rather than data analysis, leading to wasted ad spend and poor ROI.
- Complex Marketing Tools: Existing platforms like Google Ads are overwhelming for small business owners who need simple, actionable guidance.

Detailed Requirements

Functional Requirements

Data Collection

- Daily EventBrite scraping via Lambda function for local events
- Real-time weather data through MCP server integration
- Event deduplication and data cleaning
- User registration with business type and location

AI Processing

- Sequential workflow through three specialized AI agents
- Real-time progress tracking via Server-Sent Events
- Personalized recommendations based on business type and location
- Error handling and retry mechanisms

Delivery System

- Automated daily email delivery at 7 AM CT
- On-demand recommendations through web interface
- HTML-formatted actionable insights
- User data cleanup after 7 days

Non-Functional Requirements

- Performance: Process 500+ events daily, support 1,000+ concurrent users, sub-200ms database queries
- Reliability: 99.5% uptime, automated backups, graceful degradation
- Security: API key authentication, HTTPS encryption, input validation
- Scalability: Auto-scaling App Runner, connection pooling, asynchronous processing

Technical Architecture

- Application: Flask + Gunicorn, CrewAI framework, Google Gemini models, SSE for real-time updates
- Database: PostgreSQL on AWS RDS with serverless scaling
- Infrastructure: AWS App Runner, Lambda, EventBridge, CloudWatch
- Integration: EventBrite scraping, MCP weather server, SMTP email delivery

Key Risks and Mitigation

- API Dependencies: EventBrite structure changes, weather API limits Mitigation: Multiple data sources, robust error handling, usage monitoring

- AI Costs: Gemini API cost escalation with usage growth Mitigation: Usage monitoring, prompt optimization, spending limits

- Data Quality: Web scraping reliability issues Mitigation: Data validation, historical fallbacks, multiple collection strategies

- Email Delivery: Spam filtering, deliverability issues Mitigation: Established providers, proper authentication, clean lists

Critical Dependencies

- External: EventBrite website access, weather APIs, Google Gemini API, AWS services Infrastructure: Docker containerization, database schema, email service configuration Timeline: MCP server deployment before weather integration, EventBridge setup before production
- The most critical dependency is reliable EventBrite scraping - any website changes could break our data pipeline. AI model costs and availability also require continuous monitoring and potential alternative solutions.

Assumptions

Technical Assumptions

- EventBrite will maintain current website structure for scraping accessibility
- Google Gemini API will remain available with predictable pricing
- AWS services will maintain 99.9% uptime for core infrastructure
- Weather APIs will provide accurate 3-day forecasts with minimal latency

Business Assumptions

- Small businesses check email regularly and will act on recommendations
- Local events have measurable impact on nearby business traffic
- Weather patterns correlate predictably with consumer behavior
- Business owners prefer simple recommendations over complex analytics

User Assumptions

- Users will provide accurate business location and type information
- Email addresses will remain active for recommendation delivery
- Businesses operate primarily during standard hours (6 AM - 10 PM)
- Users understand basic advertising concepts and terminology

Solutions

Core Solution Architecture

- Event Intelligence: Automated scraping + AI analysis identifies relevant local opportunities
- Weather Integration: MCP server provides real-time meteorological data for behavior prediction
- AI Recommendation Engine: Three-agent workflow generates personalized, actionable advertising advice
- Automated Delivery: Daily email system ensures timely recommendations without manual intervention

Key Technical Solutions

- Data Reliability: Multi-source fallbacks and validation prevent single points of failure
- Scalability: App Runner auto-scaling handles usage spikes during peak business hours
- User Experience: SSE provides real-time feedback during recommendation generation
- Cost Management: Usage monitoring and prompt optimization control AI processing expenses

High Level Implementation Plan

Phase 1 (Weeks 1-4): Foundation

- Set up AWS infrastructure (RDS, App Runner, Lambda)
- Implement EventBrite scraping pipeline
- Build basic user registration and database schema
- Deploy MCP weather server

Phase 2 (Weeks 5-8): Core Features

- Develop AI agent workflows with CrewAI
- Build recommendation engine and email system
- Implement web interface with SSE
- Set up EventBridge scheduling

Phase 3 (Weeks 9-12): Production Ready

- Add monitoring, logging, and error handling
- Implement security measures and API authentication
- Performance optimization and load testing
- User acceptance testing and feedback integration

Implementation Blockers

Technical Blockers

- CrewAI Library Maturity: Framework is still evolving, causing dependency conflicts and larger Docker images
- EventBrite Anti-Scraping: Website may implement measures blocking automated data collection
- AI Model Consistency: Gemini responses may vary in quality, affecting recommendation reliability
- Email Deliverability: Automated emails risk spam filtering, reducing user engagement

Resource Blockers

- API Cost Scaling: Gemini usage costs could exceed budget with user growth
- Infrastructure Complexity: Managing multiple AWS services requires DevOps expertise
- Data Quality Maintenance: Event data cleaning and validation needs ongoing manual oversight
- Legal Compliance: Data collection practices may require legal review for privacy regulations

Timeline Blockers

- External API Dependencies: Third-party service outages could delay testing and development
- Docker Image Optimization: Large images due to CrewAI dependencies may slow deployment cycles
- User Feedback Loop: Need real users to validate recommendation quality and system effectiveness
- Integration Testing: Coordinating multiple services (scraping, weather, AI, email) increases complexity

Solution Phases

Phase One: Foundation Implementation

Core Search Infrastructure

- Deploy Elasticsearch with automated indexing pipeline for patient records, medical documents, and clinical data
- Implement RESTful search APIs with filtering capabilities (date range, provider, specialty, document type)
- Build responsive web interface with real-time search suggestions and advanced query options
- Establish role-based access control ensuring HIPAA compliance and data security

Key Features Delivered

- Basic keyword and phrase search across structured/unstructured data
- Document preview and metadata display
- Search history and saved queries
- Mobile-optimized interface for healthcare professionals

Technical Stack

- Backend: Node.js/Python with Elasticsearch
- Frontend: React.js with responsive design
- Security: OAuth 2.0, encryption at rest/transit
- Infrastructure: Cloud-based with automated backups

Future Phases

Phase Two: Intelligence Enhancement

- Natural Language Processing for clinical terminology understanding
- Machine learning-powered search result ranking and personalization
- Integration with EHR systems and third-party medical databases
- Advanced analytics dashboard for search patterns and usage insights

Phase Three: AI-Powered Features

- Semantic search using medical ontologies (SNOMED CT, ICD-10)
- Voice search capabilities for hands-free operation
- Predictive search suggestions based on clinical context
- Automated document classification and tagging

Phase Four: Advanced Integration

- Real-time data synchronization across multiple healthcare systems
- API ecosystem for third-party integrations
- Federated search across external medical databases
- Mobile app with offline search capabilities