Requirements Document for Agentic AI App for Independent Business Consultants

Date Created: July 21, 2025

Time Created: 12:34 PM PDT

Version: 1.3

Overview

This document outlines the requirements for an Agentic AI app designed for small consulting firms (10-15 employees/contractors per tenant, 10-20 tenants initially). The app minimizes administrative expenses and enhances work efficiency with a chat-based UI, targeting a one-person team with minimal coding experience. The development timeline is 3 months with a $15,000 budget, focusing on an MVP using AI agents and frameworks.

1. Functional Requirements

General App Vision

Target User Profile

- Primary Users: Small consulting firms with 10-15 employees/contractors (multi-tenant, 10-20 tenants in year one, scalable).

- Tech Comfort Level: Tech-savvy but prefer a simple, easy-to-understand UI and workflows.

- Primary Interaction: Mostly chat-based (similar to Grok), with natural language text or voice input.

- Regional/Regulatory: Tax requirements (e.g., GDPR) ignored for now.

Core Value Proposition

- Main Pain Point: Minimizes administrative expenses while managing work effectively and efficiently like human experts.

- Differentiation: Proactively negotiates contracts, extracts deliverables/timelines, prompts time entries, creates draft invoices, and sends payment follow-ups, unlike repository tools.

Scalability Goals

- Initial Scale: Support 10-20 customers (100-300 users) in the first year.

- Growth Support: Accommodate more users per tenant (up to 20-25) and additional tenants over time.

Feature-Specific Requirements

1. Client and Contract Management (MVP Feature)

- Data: Track engagement stages (e.g., prospective to project in progress), contact details (name, email, phone, role) extracted from emails/calendars.

- Proactive Management: AI manages contracts with human supervision (e.g., renewal reminders).

- Input: Uploaded PDFs or emails.

- Clauses: Supports various types (e.g., payment terms, NDAs).

2. Deliverable and Compliance Management (Deferred)

- Deliverables: Any type (e.g., reports) with data: description, due date, assigned person, availability, days remaining, invoicing status.

- Compliance: Quality checks.

- AI Role: Suggests deliverables, flags non-compliance, identifies at-risk deliverables.

- Late Deliverables: Proposes recovery options and drafts client communications.

3. Employees/Contractor Management (MVP Feature)

- Number: 10-15 per firm.

- Data: Hire date, contracts, rates (hourly/fixed), availability, skills.

- AI Onboarding: Onboards with supervision, considers US compliance (e.g., 1099 forms).

4. Assignment and Workload Management (Deferred)

- Assignment: AI suggests; human assigns.

- Workload: AI proposes; human finalizes.

- Metrics: Time worked vs. agreed, on-time delivery.

- Conflicts: Human resolves overlapping deliverables (multi-project support).

5. Time Entry per Deliverable (MVP Feature)

- Tracking: Manual entry.

- AI Role: Categorizes (billable/non-billable) and prompts daily logging.

- Granularity: Daily per deliverable, including administrative tasks.

- Integrations: None in v1.

6. Invoicing (Deferred)

- Details: Best practices per contract terms (weekly/monthly, time/material, fixed price).

- AI Role: Generates invoices automatically.

- Payment: Records receipt from bank statements (no processing in v1).

- Recurring/Reminders: Supports recurring invoices, sends late payment reminders.

7. Expense Management (MVP Feature)

- Expenses: All types (travel, subscriptions, supplies).

- AI Role: Categorizes (project-specific/overhead), learns from human modifications.

- Receipts: Photo/PDF uploads with OCR.

- Budgets: None in v1.

8. Real-Time Profit and Loss Statements (Deferred)

- Data: Revenue, expenses (including non-cash like depreciation), profit margins by client.

- Frequency: Daily updates.

- AI Role: Provides insights (e.g., "Client X's margin is low").

- Export: CSV/Excel format (no integrations in v1).

Agentic AI-Specific Requirements

- Autonomy: AI handles tasks (e.g., contract extraction) with human supervision where specified.

- Multi-Agent: 6 agents for MVP (4 specialized + 1 Development Agent + 1 Manager), expandable to 9 later.

- Learning: Adapts to user preferences (e.g., categorization).

- Integrations: None in v1, designed for future API connections.

2. Technical Requirements

Technical Stack

- Frontend: React.js with Tailwind CSS, React-Chat-UI for chat interface (or Bubble for no-code UI).

- Backend: Python with FastAPI (lightweight and scalable).

- AI and Agents: xAI's Grok 3 (free tier via https://x.ai/api), CrewAI for agents, TensorFlow/PyTorch for minimal custom training if needed.

- Database: PostgreSQL (structured data), Redis (caching) - local setup initially.

- Infrastructure: AWS free tier (e.g., S3, Lambda) for minimal costs, Docker for local testing, GitHub Actions for CI/CD.

- Security: OAuth 2.0 with JWT (basic), AES-256 encryption, TLS 1.3 - simplified for prototype.

- Integrations (Future): Planned for calendars, banks.

Architecture

- Design: Simplified multi-tenant (local PostgreSQL), basic microservices for MVP features, event-driven with local task scheduling.

- Data Flow: Chat input (Bubble/UI) -> FastAPI -> CrewAI agents -> PostgreSQL/Redis -> Output to UI.

- Scalability: Local testing with potential AWS scaling (100-300 users initially).

Development Process

- Phases (Adjusted for 3-Month Timeline and One-Person Team):

1. Planning (1 Week): Define scope, set up AI tools, assign Development Agent tasks.

2. Prototyping (4 Weeks): Build MVP with 4 agents using no-code and AI assistance.

3. Development (5 Weeks): Integrate database, refine UI with Development Agent.

4. Testing (1 Week): Basic tests with sample data.

5. Launch (1 Week): Deploy to 5-10 tenants on AWS free tier.

- Tools: Git, pytest, Selenium (basic), Prometheus/Grafana (minimal).

- Constraints:

- Budget: $15,000 (covers tools, minimal outsourcing if needed).

- Timeline: 3 months (12 weeks total).

- Team: One person (you), assisted by AI agents/tools.

Deployment and Maintenance

- Deployment: Manual initial deploy to AWS free tier, GitHub Actions for updates.

- Maintenance: Bi-weekly updates, basic monitoring.

- Backup: Weekly local backups, S3 for critical data.

Risks and Mitigation

- Risk: Tight timeline -> Mitigation: Focus on MVP (4 agents), defer complex features.

- Risk: Budget overrun -> Mitigation: Use free tools, limit AWS usage.

- Risk: Limited coding skills -> Mitigation: Rely on no-code (Bubble), AI assistance (Copilot, Development Agent).

3. Technical Choices and Architecture

Technical Choices

- AI Model: Grok 3 (xAI) selected for its NLP, reasoning, and free tier (via https://x.ai/api), aligning with cost and ease for a non-coder.

- AI Framework: CrewAI chosen for multi-agent support (6 agents), open-source, and ease with guided setup.

- AI-Driven Development Tools:

- GitHub Copilot: AI-assisted coding (free tier, ~$10/month for pro).

- Cursor: AI-powered IDE (free tier, ~$20/month for advanced).

- Development Agent: Custom CrewAI agent to suggest code, tests, UI designs.

- No-Code Platform: Bubble for frontend UI, minimizing coding.

- Rationale: These tools reduce coding needs, fit your budget ($0-$500), and support a 3-month timeline.

Detailed Architecture

- High-Level Design:

- Multi-Tenant: Simplified local setup (PostgreSQL schemas), scalable to AWS.

- Microservices: Basic services (e.g., contract analysis) via FastAPI.

- Event-Driven: Local task scheduling with future RabbitMQ.

- AI Layer: CrewAI agents with Grok 3, hosted locally or AWS Lambda (free tier).

- Data Flow:

1. User chats via Bubble UI, sending requests to FastAPI.

2. CrewAI agents (e.g., Development Agent, Client/Contract Agent) process, pulling data from PostgreSQL, caching in Redis.

3. Outputs (e.g., code suggestions, contract analysis) to UI or files.

4. Storage: Contracts/receipts in local files (S3 later), metadata in PostgreSQL.

- Prototype Focus:

- MVP Agents (4): Client/Contract (extracts data), Employee/Contractor (onboards), Time Entry (prompts/categorizes), Expense (categorizes receipts).

- Development Agent: Assists with coding, UI design, testing.

- Manager Agent: Coordinates MVP agents.

- Tech: Bubble (frontend), FastAPI (backend), CrewAI (agents), Grok 3 (model), local PostgreSQL/Redis.

Agent Design

- Number: 6 agents for MVP.

- MVP Agents (4): Client/Contract, Employee/Contractor, Time Entry, Expense.

- Development Agent: Assists development.

- Manager Agent: Coordinates MVP agents.

- Future Agents (3): Deliverable, Invoicing, Financial (post-MVP).

- Implementation: CrewAI with Grok 3, prebuilt templates, minimal custom tuning.

Current Prototype

- Setup:

- Frontend: Bubble with chat UI.

- Backend: FastAPI (local, http://localhost:8000).

- Agents: CrewAI with Client/Contract Agent initially, Development Agent for assistance.

- Test: Upload a text file (e.g., "Contract with John Doe, due date: 2025-08-21") to extract data.

- AI Development: Copilot/Cursor suggests code; Development Agent reviews.

Budget and Timeline Breakdown

- Budget ($5,000):

- Tools: $500 (Copilot/Cursor subscriptions).

- AWS Free Tier: $0 (local testing, minimal S3/Lambda).

- Contingency: $4,500 (for outsourcing if needed).

- Timeline (12 Weeks):

- Week 1: Planning, env setup.

- Weeks 2-5: Prototype 4 MVP agents.

- Weeks 6-10: Develop integrations, refine UI.

- Weeks 11-12: Testing, launch to 5-10 tenants.

Future Considerations

- Scaling: AWS scaling post-MVP.

- Features: Add remaining agents with additional funding.

Next Steps

- Feedback: Review and suggest changes.

- Prototype Test: Set up and test the MVP agents.

- AI Assistance: Use Development Agent with Copilot/Cursor - request sample scripts if needed.