Agentic Platform - Project Requirements Document

Project Name: Agentic Platform
Target Implementation: Claude Code (Opus 4.5)
Document Version: 1.0
Date: January 29, 2026

---

1. Executive Summary

Build a scalable, multi-user platform that enables users to create, configure, and deploy AI agents with custom personalities and purposes. Agents can execute user-defined tasks and communicate with other agents within the platform ecosystem.

---

2. Core Objectives

- Enable non-technical users to create and manage AI agents easily
- Provide robust agent-to-agent communication infrastructure
- Support concurrent execution of multiple agents
- Ensure isolation and security between user workspaces
- Deliver a responsive, intuitive user interface
- Enable extensibility for future capabilities

---

3. Functional Requirements

3.1 User Management

3.1.1 Authentication & Authorization

- User registration with email verification
- Secure login/logout functionality
- Password reset capability
- Role-based access control (Admin, User, Viewer)
- Session management with configurable timeout
- Multi-factor authentication (optional)

3.1.2 User Profiles

- Profile creation and editing

- User preferences storage
- API key management for external integrations
- Usage tracking and quota management

3.2 Agent Creation & Configuration

3.2.1 Agent Builder Interface

- Visual agent creation wizard
- Agent naming and description fields
- Agent avatar/icon selection or upload
- Template library for common agent types
- Clone existing agent functionality
- Import/export agent configurations (JSON format)

3.2.2 Purpose Definition

- Primary purpose/goal text field (max 500 characters)
- Secondary objectives list (up to 5 items)
- Success criteria definition
- Failure handling rules
- Task scope boundaries

3.2.3 Personality Configuration

- Personality trait sliders:
  - Formality level (casual ↔ formal)
  - Verbosity (concise ↔ detailed)
  - Creativity (conservative ↔ innovative)
  - Empathy level
  - Assertiveness
- Communication style selector:
  - Professional
  - Friendly
  - Technical
  - Creative
  - Custom (free text)
- Tone preferences (encouraging, neutral, direct)
- Language and dialect settings

- Custom personality prompt template

3.2.4 Knowledge & Capabilities

- Knowledge domain selection (multiple choice)
- Document upload for agent-specific knowledge
- Tool/integration permissions:
  - Web search
  - Code execution
  - File operations
  - External API access
- Capability restrictions and guardrails
- Custom instruction sets

3.2.5 Agent Metadata

- Creation timestamp
- Last modified timestamp
- Creator/owner information
- Version history
- Usage statistics
- Performance metrics

3.3 Agent Execution Engine

3.3.1 Prompt Execution

- Single prompt execution mode
- Batch prompt processing
- Scheduled execution support
- Prompt templates library
- Variable substitution in prompts
- Prompt history and logging

3.3.2 Conversation Management

- Multi-turn conversation support
- Context window management
- Conversation persistence
- Conversation branching

- Export conversation transcripts
- Conversation search and filtering

3.3.3 Execution Controls

- Start/stop/pause agent execution
- Execution timeout settings
- Rate limiting per agent
- Concurrent execution limits
- Priority queuing
- Resource allocation controls

3.3.4 Output Handling

- Structured output formatting
- File generation and storage
- Stream output for long responses
- Output validation
- Output post-processing hooks

3.4 Agent-to-Agent Communication

3.4.1 Communication Protocols

- Direct messaging between agents
- Broadcast messaging to agent groups
- Request-response patterns
- Pub-sub event system
- Message queuing with persistence
- Message encryption for sensitive data

3.4.2 Communication Configuration

- Communication whitelist/blacklist per agent
- Message format standards (JSON schema)
- Protocol versioning
- Handshake and discovery mechanisms
- Timeout and retry policies

3.4.3 Collaboration Patterns

- Sequential workflows (Agent A → Agent B → Agent C)
- Parallel execution with aggregation
- Debate/consensus mechanisms
- Supervisor-worker patterns
- Delegation and handoff protocols

3.4.4 Communication Monitoring

- Message logging and auditing
- Communication flow visualization
- Performance metrics (latency, throughput)
- Error tracking
- Debug mode for troubleshooting

3.5 Workspace & Organization

3.5.1 Agent Organization

- Project/workspace concept
- Folder hierarchy for agents
- Tagging system
- Search and filter by attributes
- Favorites/bookmarks
- Bulk operations (delete, move, archive)

3.5.2 Sharing & Collaboration

- Share agents with other users (read-only, editable)
- Team workspaces
- Public agent marketplace
- Agent versioning and forking
- Collaborative editing with conflict resolution

3.5.3 Resource Management

- Storage quota per user
- Execution time quotas
- API call limits
- Cost tracking (if using paid APIs)
- Resource usage dashboards

3.6 Monitoring & Analytics

3.6.1 Agent Performance Monitoring

- Response time metrics
- Success/failure rates
- Token usage statistics
- Error logs and alerting
- Performance comparison across agents

3.6.2 System Monitoring

- Active agent count
- System resource utilization
- Queue depths
- API health checks
- Database performance metrics

3.6.3 User Analytics

- Agent creation trends
- Usage patterns
- Popular agent types
- User engagement metrics
- Retention analysis

3.6.4 Logging & Auditing

- Comprehensive audit trails
- Security event logging
- Compliance reporting
- Log retention policies
- Log search and export

3.7 User Interface

3.7.1 Dashboard

- Overview of all user agents
- Recent activity feed
- Quick actions (create, run, view)

- System notifications
- Usage statistics summary

3.7.2 Agent Management Interface

- List view with sorting and filtering
- Grid/card view option
- Bulk selection and actions
- Context menus for quick actions
- Drag-and-drop organization

3.7.3 Agent Detail View

- Configuration panel
- Execution panel
- Communication logs
- Performance metrics
- Edit mode

3.7.4 Execution Interface

- Prompt input area with syntax highlighting
- Real-time output streaming
- Conversation history view
- Agent status indicators
- Stop/pause controls

3.7.5 Communication Visualization

- Agent relationship graph
- Message flow diagram
- Real-time communication feed
- Filter by agent, time, message type

3.8 API & Integration

3.8.1 REST API

- Complete CRUD operations for agents
- Execution endpoints
- Webhook support

- API documentation (OpenAPI/Swagger)
- Rate limiting
- API versioning

3.8.2 WebSocket API

- Real-time agent output streaming
- Live communication events
- Status updates
- Bidirectional communication

3.8.3 External Integrations

- Anthropic API integration (Claude models)
- Cloud storage (S3, GCS, Azure Blob)
- Authentication providers (OAuth, SAML)
- Notification services (email, Slack, Discord)
- Monitoring tools (Datadog, New Relic)

---

4. Non-Functional Requirements

4.1 Performance

- Agent execution latency: < 2 seconds for simple prompts
- UI response time: < 500ms for interactive operations
- Support 1000+ concurrent agent executions
- Database query response: < 100ms for 95th percentile
- Page load time: < 2 seconds
- Websocket message latency: < 100ms

4.2 Scalability

- Horizontal scaling for execution workers
- Database sharding support
- Microservices architecture
- Load balancer compatibility
- Auto-scaling based on demand
- Support 10,000+ active users

- Support 100,000+ agents in the system

4.3 Reliability

- System uptime: 99.9%
- Automatic failover for critical components
- Data backup every 6 hours
- Point-in-time recovery capability
- Graceful degradation during partial outages
- Circuit breaker patterns for external dependencies

4.4 Security

- Data encryption at rest (AES-256)
- Data encryption in transit (TLS 1.3)
- API authentication via JWT tokens
- SQL injection prevention
- XSS and CSRF protection
- Regular security audits
- Principle of least privilege
- Secrets management (vault integration)
- Input validation and sanitization
- Rate limiting to prevent abuse

4.5 Data Privacy & Compliance

- User data isolation
- GDPR compliance:
    - Right to access
    - Right to deletion
    - Right to data portability
    - Consent management
- SOC 2 compliance preparation
- Data residency options
- Privacy policy implementation
- Terms of service

4.6 Usability

- Responsive design (desktop, tablet, mobile)
- Accessibility compliance (WCAG 2.1 Level AA)
- Multi-language support (i18n)
- Dark mode support
- Keyboard shortcuts
- Intuitive navigation
- Contextual help and tooltips
- Onboarding tutorial

4.7 Maintainability

- Modular codebase
- Comprehensive code documentation
- Unit test coverage > 80%
- Integration test suite
- CI/CD pipeline
- Infrastructure as code
- Automated deployment
- Rollback capabilities
- Feature flags for controlled rollouts

4.8 Observability

- Structured logging
- Distributed tracing
- Metrics collection (Prometheus compatible)
- Health check endpoints
- Status page
- Error tracking (Sentry compatible)
- Performance profiling tools

---

5. Technical Architecture

5.1 Technology Stack (Recommended)

Backend

- Language: Python 3.11+ or Node.js 20+
- Framework: FastAPI (Python) or Express.js (Node)
- Database: PostgreSQL 15+ (primary), Redis (cache/queue)
- Message Queue: RabbitMQ or Redis Streams
- Task Queue: Celery (Python) or Bull (Node)
- API Gateway: Kong or Nginx

Frontend

- Framework: React 18+ or Vue 3+
- State Management: Redux/Zustand or Pinia
- UI Components: Material-UI or Ant Design
- Real-time: Socket.io or native WebSocket
- Build Tool: Vite or Webpack

Infrastructure

- Containerization: Docker
- Orchestration: Kubernetes or Docker Swarm
- Cloud Provider: AWS, GCP, or Azure
- CDN: CloudFlare
- Monitoring: Prometheus + Grafana
- Logging: ELK Stack or Loki

5.2 System Components

Core Services

1. **User Service**: Authentication, authorization, profile management
2. **Agent Service**: CRUD operations, configuration management
3. **Execution Service**: Agent runtime, prompt processing, Claude API integration
4. **Communication Service**: Inter-agent messaging, event handling
5. **Storage Service**: File management, document processing
6. **Analytics Service**: Metrics collection, reporting
7. **Notification Service**: Alerts, emails, webhooks
8. **API Gateway**: Request routing, rate limiting, authentication

Support Services

1. **Database**: Primary data store

2. **Cache Layer**: Session storage, frequently accessed data
3. **Message Queue**: Asynchronous task processing
4. **Object Storage**: Files, logs, backups
5. **Search Engine**: Full-text search (Elasticsearch/OpenSearch)

5.3 Data Models

User Model

```
- id (UUID)
- email (string, unique)
- password_hash (string)
- name (string)
- role (enum: admin, user, viewer)
- created_at (timestamp)
- updated_at (timestamp)
- last_login (timestamp)
- preferences (JSON)
- quota_limits (JSON)
```

Agent Model

```
- id (UUID)
- user_id (UUID, foreign key)
- name (string)
- description (text)
- purpose (text)
- personality_config (JSON)
- knowledge_config (JSON)
- capabilities (JSON array)
- communication_config (JSON)
- status (enum: active, inactive, archived)
- created_at (timestamp)
- updated_at (timestamp)
- version (integer)
- metadata (JSON)
```

Execution Model

- id (UUID)
- agent_id (UUID, foreign key)
- user_id (UUID, foreign key)
- prompt (text)
- response (text)
- status (enum: pending, running, completed, failed)
- started_at (timestamp)
- completed_at (timestamp)
- execution_time_ms (integer)
- token_usage (JSON)
- error_message (text, nullable)
- context (JSON)

Message Model (Agent Communication)

- id (UUID)
- from_agent_id (UUID, foreign key)
- to_agent_id (UUID, foreign key)
- message_type (enum: request, response, broadcast, event)
- content (JSON)
- status (enum: sent, delivered, failed)
- created_at (timestamp)
- delivered_at (timestamp)
- metadata (JSON)

5.4 Integration Points

Claude API Integration

- Model selection per agent (Sonnet, Opus, Haiku)
- System prompt construction from personality/purpose
- Streaming response handling
- Error handling and retry logic
- Token usage tracking
- Cost optimization strategies

External Services

- Email service (SendGrid, AWS SES)
- Object storage (S3, GCS, Azure Blob)
- Monitoring (Datadog, New Relic, Prometheus)
- Error tracking (Sentry)
- Authentication (Auth0, Okta)

6. User Stories

6.1 Core User Stories

US-001: As a user, I want to create an agent with a specific purpose so that it can help me accomplish specific tasks.

US-002: As a user, I want to define my agent's personality so that it communicates in a way that matches my preferences.

US-003: As a user, I want to send prompts to my agent and receive responses so that I can interact with it.

US-004: As a user, I want to create multiple agents with different purposes so that I can use specialized agents for different tasks.

US-005: As a user, I want my agents to communicate with each other so that they can collaborate on complex tasks.

US-006: As a user, I want to monitor my agent's performance so that I can understand how well it's working.

US-007: As a user, I want to share my agents with team members so that we can collaborate.

US-008: As a user, I want to organize my agents into projects so that I can manage them effectively.

US-009: As a user, I want to view the conversation history between agents so that I can understand their interactions.

US-010: As a user, I want to export my agent configurations so that I can back them up or transfer them.

6.2 Advanced User Stories

US-011: As a user, I want to create agent workflows where multiple agents work sequentially so that I can automate complex processes.

US-012: As a user, I want to set execution schedules for my agents so that they can run automatically.

US-013: As a user, I want to receive notifications when my agents complete tasks so that I stay informed.

US-014: As a user, I want to upload documents to provide domain-specific knowledge to my agents.

US-015: As a user, I want to clone and modify existing agents so that I can iterate quickly.

US-016: As an admin, I want to monitor system-wide usage so that I can ensure optimal performance.

US-017: As a user, I want to use templates to create agents quickly for common use cases.

US-018: As a user, I want to control which agents can communicate with each other for security purposes.

US-019: As a user, I want to version my agents so that I can roll back to previous configurations.

US-020: As a user, I want to test my agents in a sandbox before deploying them to production.

7. Acceptance Criteria

7.1 Agent Creation

- User can create an agent in less than 2 minutes
- All required fields are validated before submission
- Agent appears in dashboard immediately after creation
- Default configurations are applied if not specified

7.2 Agent Execution

- Agent responds to prompts within 5 seconds for typical requests
- Responses maintain configured personality traits
- Execution history is preserved
- Failed executions include clear error messages

7.3 Agent Communication

- Messages are delivered within 1 second
- Message delivery is guaranteed (retry on failure)
- Communication logs are accessible and filterable
- Users can visualize communication patterns

7.4 User Interface

- All pages load within 2 seconds
- Interface is responsive on devices with 375px+ width
- No horizontal scrolling on supported devices
- All interactive elements have hover/focus states

7.5 Security

- All sensitive data is encrypted
- Authentication failures are logged
- Users can only access their own agents (unless shared)
- API endpoints require valid authentication

7.6 Reliability

- System recovers automatically from transient failures
- No data loss during system updates

- Backups are restorable within 30 minutes
- Failed operations don't leave system in inconsistent state

---

8. Testing Requirements

8.1 Unit Testing

- Test coverage > 80% for all services
- Mock external dependencies (Claude API, etc.)
- Test edge cases and error conditions
- Automated test execution on commit

8.2 Integration Testing

- End-to-end user flows
- Agent-to-agent communication scenarios
- Database transaction handling
- External API integration testing

8.3 Performance Testing

- Load testing with 1000+ concurrent users
- Stress testing to find breaking points
- Latency testing for critical operations
- Database query optimization verification

8.4 Security Testing

- Penetration testing
- Authentication and authorization testing
- Input validation testing
- Dependency vulnerability scanning

8.5 User Acceptance Testing

- Beta testing with 50+ users
- Usability testing sessions
- Accessibility testing with assistive technologies
- Cross-browser compatibility testing

9. Deployment Requirements

9.1 Environments

- Development: Local and cloud-based dev environments
- Staging: Production-like environment for testing
- Production: Multi-region deployment for redundancy

9.2 CI/CD Pipeline

- Automated testing on pull requests
- Automated deployment to staging on merge
- Manual approval for production deployment
- Automated rollback on deployment failure
- Blue-green deployment strategy

9.3 Infrastructure

- Infrastructure as code (Terraform or Pulumi)
- Automated provisioning
- Auto-scaling policies
- Disaster recovery plan
- Backup and restore procedures

9.4 Monitoring & Alerting

- Application performance monitoring
- Infrastructure monitoring
- Log aggregation and analysis
- Alerting for critical failures
- On-call rotation setup

---

10. Documentation Requirements

10.1 User Documentation

- Getting started guide
- Agent creation tutorial

- Agent communication tutorial
- Best practices guide
- FAQ section
- Video tutorials

10.2 API Documentation

- OpenAPI/Swagger specification
- Authentication guide
- Code examples in multiple languages
- Rate limiting documentation
- Webhook documentation

10.3 Developer Documentation

- Architecture overview
- Setup instructions
- Contribution guidelines
- Code style guide
- Database schema documentation
- Deployment guide

10.4 Operations Documentation

- Runbook for common issues
- Backup and recovery procedures
- Scaling guidelines
- Security incident response plan
- Monitoring setup guide

---

11. Project Phases

Phase 1: Foundation (Weeks 1-4)

- Database setup and core models
- User authentication and authorization
- Basic agent CRUD operations
- Simple agent execution with Claude API
- Basic UI for agent management

Phase 2: Core Features (Weeks 5-8)

- Personality and purpose configuration
- Execution engine with context management
- Conversation persistence
- Agent organization (folders, tags)
- Enhanced UI with dashboard

Phase 3: Communication (Weeks 9-12)

- Agent-to-agent messaging infrastructure
- Communication protocols implementation
- Message logging and monitoring
- Communication visualization
- Workflow patterns (sequential, parallel)

Phase 4: Polish & Scale (Weeks 13-16)

- Performance optimization
- Security hardening
- Comprehensive testing
- Documentation
- Beta testing
- Production deployment preparation

Phase 5: Advanced Features (Post-Launch)

- Agent marketplace
- Advanced workflows
- Scheduled executions
- Enhanced analytics
- Mobile applications
- Third-party integrations

---

12. Success Metrics

12.1 User Engagement

- Monthly active users: 1,000+ within 3 months
- Agent creation rate: 5+ agents per active user
- Daily active agents: 50% of created agents
- User retention: 40% after 30 days

12.2 System Performance

- Average response time: < 2 seconds
- System uptime: > 99.5%
- Error rate: < 1%
- Agent execution success rate: > 95%

12.3 Business Metrics

- User growth rate: 20% month-over-month
- Conversion to paid plans (if applicable): 10%
- Customer satisfaction (NPS): > 40
- Support ticket volume: < 5% of active users

---

13. Risk Assessment

13.1 Technical Risks

- **Risk**: Claude API rate limits or downtime
  - **Mitigation**: Implement queue system, retry logic, and fallback options
- **Risk**: Scalability challenges with agent-to-agent communication
  - **Mitigation**: Design for horizontal scaling from start, implement message queuing
- **Risk**: Data consistency in distributed system
  - **Mitigation**: Use transaction boundaries, implement idempotency, eventual consistency where appropriate

13.2 Security Risks

- **Risk**: Unauthorized access to agents or data
  - **Mitigation**: Strong authentication, authorization checks, encryption, audit logging

- **Risk**: Agent prompt injection attacks
    - **Mitigation**: Input sanitization, system prompt protection, output validation

13.3 Business Risks

- **Risk**: Low user adoption
    - **Mitigation**: User research, iterative design, onboarding optimization, marketing
- **Risk**: High API costs
    - **Mitigation**: Usage monitoring, quotas, optimization, tiered pricing model

---

14. Constraints & Assumptions

14.1 Constraints

- Must use Anthropic Claude models via API
- Initial budget: [To be defined]
- Launch timeline: 16 weeks
- Team size: [To be defined]
- Must comply with data protection regulations

14.2 Assumptions

- Users have basic understanding of AI agents
- Anthropic API will maintain current pricing and availability
- Users have modern web browsers (last 2 versions)
- Internet connection required for platform access
- English as primary language (initially)

---

15. Glossary

- **Agent**: An AI-powered entity configured with specific purpose and personality
- **Purpose**: The primary goal or function an agent is designed to accomplish
- **Personality**: The communication style and behavioral traits of an agent
- **Execution**: The process of running an agent with a specific prompt
- **Workspace**: A organizational unit containing multiple agents and resources
- **System Prompt**: Internal instructions that define agent behavior (invisible to end users)
- **User Prompt**: The input provided by users or other agents to initiate execution

- **Message**: A unit of communication between agents
- **Workflow**: A coordinated sequence of agent executions

---

16. Appendices

Appendix A: API Endpoints (High-Level)

Authentication

- POST /api/auth/register
- POST /api/auth/login
- POST /api/auth/logout
- POST /api/auth/refresh

Users

- GET /api/users/me
- PUT /api/users/me
- DELETE /api/users/me

Agents

- GET /api/agents
- POST /api/agents
- GET /api/agents/:id
- PUT /api/agents/:id
- DELETE /api/agents/:id
- POST /api/agents/:id/clone

Executions

- POST /api/agents/:id/execute
- GET /api/agents/:id/executions
- GET /api/executions/:id
- DELETE /api/executions/:id

Communication

- POST /api/agents/:id/send
- GET /api/agents/:id/messages
- GET /api/messages/:id

- WS /api/agents/:id/stream

Workspaces

- GET /api/workspaces
- POST /api/workspaces
- GET /api/workspaces/:id
- PUT /api/workspaces/:id
- DELETE /api/workspaces/:id

Appendix B: Personality Configuration Schema

```json
{
  "personality": {
    "traits": {
      "formality": 0.7,
      "verbosity": 0.5,
      "creativity": 0.6,
      "empathy": 0.8,
      "assertiveness": 0.5
    },
    "style": "professional",
    "tone": "encouraging",
    "language": "en-US",
    "custom_prompt": "Additional personality instructions..."
  }
}
```

Appendix C: Communication Message Schema

```json
```

```
{
  "message": {
    "id": "uuid",
    "from_agent_id": "uuid",
    "to_agent_id": "uuid",
    "type": "request",
    "content": {
      "subject": "Task request",
      "body": "Please analyze this data...",
      "attachments": [],
      "metadata": {}
    },
    "timestamp": "2026-01-29T10:30:00Z",
    "status": "delivered"
  }
}
```

Document History

Version	Date	Author	Changes
1.0	2026-01-29	Claude	Initial requirements document

Approval

This requirements document should be reviewed and approved by:

- [ ] Product Owner
- [ ] Technical Lead
- [ ] Security Team
- [ ] UX/UI Team
- [ ] Stakeholders

End of Requirements Document