Product Context for OllamaNet

Problem Statement

Modern AI applications require sophisticated integration with large language models (LLMs) for conversational capabilities. Developers need a robust, scalable platform that simplifies LLM deployment, management, and interaction while providing secure access, efficient resource utilization, and comprehensive administration. Current solutions often lack proper organization, streaming capabilities, or scalable architecture for production use. Additionally, LLMs often struggle with context limitations and up-to-date information, requiring enhanced capabilities like Retrieval-Augmented Generation (RAG) to provide more accurate and contextually relevant responses.

Solution Overview

Providing a comprehensive microservices platform for integrating Open source's large language model capabilities into applications through a clean, modular architecture. The platform offers conversation management, real-time chat with streaming responses, user authentication, administrative controls, and model exploration capabilities in a cohesive ecosystem. The platform enhances AI model responses through RAG capabilities, allowing conversations to be grounded in specific documents and knowledge bases.

Business Value Proposition

OllamaNet delivers significant value to organizations by:

1. **Reducing Development Time**: Providing a ready-to-use platform for AI integration, eliminating the need to build conversation infrastructure from scratch
2. **Improving User Experience**: Offering real-time streaming responses that reduce perceived latency and enhance engagement
3. **Enhancing Response Quality**: Leveraging RAG capabilities to provide more accurate, contextual, and up-to-date information
4. **Optimizing Resource Utilization**: Implementing sophisticated caching strategies to reduce computational costs
5. **Simplifying Administration**: Offering comprehensive tools for platform management and monitoring
6. **Ensuring Security**: Implementing robust authentication and authorization mechanisms
7. **Supporting Scalability**: Designing for growth with horizontal scaling capabilities
8. **Enabling Knowledge Integration**: Allowing organizations to leverage their internal documents and knowledge bases

User Experience Goals

1. **Seamless Conversations**: Provide natural, responsive chat interactions with AI models
2. **Organization**: Enable intuitive organization of conversations through folders
3. **Discovery**: Facilitate easy exploration and discovery of available AI models
4. **Security**: Ensure secure access through robust authentication and authorization
5. **Administration**: Offer comprehensive tools for platform management
6. **Performance**: Deliver responsive interfaces with optimized caching strategies

7. **Flexibility**: Support various AI models and configurations
8. **Context Richness**: Enhance conversations with document-based context through RAG
9. **Knowledge Integration**: Allow users to upload and leverage their own documents
10. **Feedback Loop**: Enable users to provide feedback on AI responses for continuous improvement

Key Features

Conversation Management

- Create, manage, and organize conversations with AI models
- Real-time streaming chat responses for immediate feedback
- Message history persistence and retrieval
- Folder organization for conversation management
- Note-taking capabilities for conversations
- Feedback collection on AI responses

User Authentication & Management

- Secure registration and login
- JWT-based authentication with refresh tokens
- Role-based access control
- Password management (reset, change)
- Session persistence

AI Model Management & Discovery

- Browse available AI models with detailed information
- Search and filter models by tags and capabilities
- Model metadata management
- Tag-based organization of models
- Installation and removal of models

Administration

- User management with role assignment
- AI model administration
- Tag management for organization
- Operational monitoring and control

API Gateway

- Unified entry point for all services
- Request routing to appropriate microservices
- Authentication and authorization enforcement
- Rate limiting for abuse prevention
- Modular configuration management
- Dynamic configuration reloading

RAG Capabilities

- Document upload and processing
- Text extraction from multiple formats (PDF, Word, Text, Markdown)
- Document chunking and embedding generation
- Vector database integration for semantic search
- Context retrieval for conversation enhancement
- Document-grounded responses with citations
- Document organization and management
- Relevance scoring and ranking

User Journeys

Developer Journey

1. **Discovery**: Developer learns about OllamaNet through documentation or recommendation
2. **Setup**: Developer registers and sets up an account with appropriate permissions
3. **Exploration**: Developer browses available models and their capabilities
4. **Integration**: Developer integrates OllamaNet APIs into their application
5. **Testing**: Developer tests conversation capabilities with various models
6. **Deployment**: Developer deploys their application with OllamaNet integration
7. **Monitoring**: Developer monitors usage and performance metrics
8. **Optimization**: Developer optimizes their integration based on analytics

Business User Journey

1. **Onboarding**: User receives account credentials from administrator
2. **First Interaction**: User creates their first conversation with an AI model
3. **Document Upload**: User uploads relevant documents for context enhancement
4. **Conversation**: User engages in conversation with AI, receiving document-grounded responses
5. **Organization**: User creates folders to organize different conversation topics
6. **Note Taking**: User adds notes to important conversations for future reference
7. **Feedback**: User provides feedback on AI responses to improve quality
8. **Search**: User searches through conversation history for specific information

Administrator Journey

1. **Platform Setup**: Administrator configures the platform for organizational needs
2. **User Management**: Administrator creates and manages user accounts and roles
3. **Model Management**: Administrator adds, updates, and removes AI models
4. **Monitoring**: Administrator monitors system health and performance
5. **Troubleshooting**: Administrator addresses issues and optimizes configuration
6. **Reporting**: Administrator generates usage and performance reports
7. **Scaling**: Administrator plans and implements scaling strategies as usage grows
8. **Updates**: Administrator manages platform updates and enhancements

Integration Points

- **Inference Engine API**: Core integration for AI model operations
- **Redis Cache**: Performance optimization through distributed caching

- **SQL Server**: Persistent storage for all service data
- **Authentication Providers**: JWT token validation and user identity
- **Monitoring Systems**: Performance and health monitoring
- **Pinecone Vector Database**: Semantic search capabilities for RAG
- **Document Storage**: File system or cloud storage for document files
- **RabbitMQ**: Service discovery and configuration updates