Mirador Technical Documentation

Version: 1.0 Production
Author: Manus AI
Date: June 6, 2025

Architecture Overview

System Design Philosophy

Mirador implements a distributed AI orchestration architecture that prioritizes specialist collaboration over monolithic intelligence. This design philosophy recognizes that complex personal life decisions require multiple domains of expertise working in coordination rather than attempting to solve all problems with a single general-purpose model.

The architecture separates concerns between domain expertise, strategic analysis, and local knowledge integration. Each specialist model focuses on a specific area of competency while the orchestration framework manages collaboration, handoffs, and result synthesis. This separation enables targeted optimization of each component while maintaining system coherence.

The framework operates entirely within the local computing environment, ensuring complete privacy and control over sensitive personal information. All processing occurs through local Ollama models without external API dependencies, providing enterprise-grade security for personal use cases.

Core Components

The Mirador system consists of several interconnected components that work together to provide comprehensive personal life automation capabilities. Understanding these components and their interactions is essential for system administration, troubleshooting, and optimization.

Ollama Runtime Environment: Serves as the foundation for all AI model execution, providing the interface between Mirador's orchestration framework and the underlying language models. Ollama manages model loading, memory allocation, and execution scheduling, ensuring efficient resource utilization across multiple concurrent model operations.

Mirador-EZ Command Interface: Provides the primary user interface for system interaction, abstracting the complexity of model orchestration behind simple command patterns. The interface handles request routing, chain orchestration, output management, and error handling, presenting a unified experience regardless of the underlying model complexity.

Specialist Model Collection: Comprises the domain-specific AI models that provide specialized expertise in financial planning, local resource integration, and strategic analysis. Each model is optimized for specific use cases and collaboration patterns, with parameters tuned for reliable performance in chain contexts.

Chain Orchestration Engine: Manages the execution of multi-model chains, handling model sequencing, context passing, and result synthesis. The engine ensures that each model in a chain receives appropriate context from previous models while maintaining coherent analysis flow.

Output Management System: Organizes and stores the results of individual queries and chain executions, providing structured access to historical analysis and enabling long-term reference and pattern identification.

Version Control Integration: Maintains comprehensive tracking of system configuration, model definitions, and development history, enabling safe experimentation and reliable rollback capabilities.

Model Architecture

Each specialist model in the Mirador system follows a consistent architectural pattern that optimizes for both individual performance and chain collaboration. Understanding this architecture enables effective model development, optimization, and troubleshooting.

Base Model Foundation: All specialist models build upon the llama3.2_balanced foundation, which provides reliable performance characteristics and proven compatibility with chain orchestration requirements. This consistent foundation ensures predictable behavior and resource requirements across all specialists.

Parameter Optimization: Model parameters are carefully tuned to balance response quality with execution speed, ensuring that chains complete within acceptable timeframes while maintaining analytical depth. Temperature settings optimize for focused analysis, while token limits ensure comprehensive responses without excessive verbosity.

System Prompt Engineering: Each model includes sophisticated system prompts that define expertise domains, response structures, and collaboration protocols. These prompts ensure consistent output quality and enable effective handoffs between models in chain contexts.

Local Knowledge Integration: Models incorporate specific knowledge about Louisville, Kentucky, and Jefferson County, providing guidance that accounts for local regulations, market conditions, and available resources. This integration distinguishes Mirador from generic AI systems.

Data Flow Architecture

Understanding Mirador's data flow architecture is essential for optimizing performance, troubleshooting issues, and developing new capabilities. The system implements a structured approach to information processing that ensures consistency and reliability.

Input Processing: User requests are processed through the mirador-ez interface, which handles request parsing, model selection, and parameter configuration. The interface validates inputs and structures them appropriately for the selected models or chain patterns.

Model Execution: Individual models receive structured inputs and generate responses according to their specialized capabilities and system prompts. Execution occurs within the Ollama runtime environment, which manages resource allocation and performance optimization.

Chain Coordination: For multi-model chains, the orchestration engine manages the sequential execution of models, passing context and results between models while maintaining coherent analysis flow. Each model builds upon previous analysis while contributing unique expertise.

Output Synthesis: Chain results are synthesized into comprehensive summaries that integrate insights from all participating models. The synthesis process identifies key recommendations, strategic insights, and actionable next steps.

Result Storage: All outputs are stored in structured directories with metadata that enables efficient retrieval and analysis. The storage system maintains both individual model outputs and synthesized chain results for comprehensive reference.

Model Specifications

Enhanced Agent Fast V3

Technical Specifications: - Base Model: llama3.2_balanced - Temperature: 0.5 - Top-p: 0.8 - Token Limit: 1500 - Repeat Penalty: 1.1

Optimization Focus: The enhanced agent fast is optimized for strategic analysis and chain collaboration, with parameters tuned to provide focused insights while building upon previous analysis. The model excels at identifying optimization opportunities and developing implementation strategies.

Performance Characteristics: Response times typically range from 15-25 seconds for individual queries and contribute 200-400 words of strategic enhancement in chain contexts. The model demonstrates consistent content preservation, adding value rather than reducing previous analysis.

Collaboration Protocol: Designed to serve as the final model in multi-model chains, the enhanced agent builds upon domain-specific analysis with strategic insights and implementation guidance. The model's system prompt emphasizes content enhancement rather than summarization.

Use Case Optimization: Most effective for strategic planning, optimization analysis, and implementation guidance. The model provides maximum value when building upon substantial foundational analysis from domain specialists.

Financial Planning Expert V5

Technical Specifications: - Base Model: llama3.2_balanced - Temperature: 0.3 - Top-p: 0.8 - Token Limit: 1500 - Repeat Penalty: 1.1

Specialization Focus: Comprehensive financial planning with Louisville/Kentucky specialization, including budget optimization, savings strategies, debt management, and local financial resource integration. The model incorporates specific knowledge of Kentucky tax rates, Jefferson County costs, and local financial institutions.

Performance Characteristics: Generates 400-600 word comprehensive responses with detailed budget frameworks and actionable recommendations. Response times typically range from 15-25 seconds for individual queries and 20-30 seconds in chain contexts.

Input Processing: Optimized to handle various income format inputs including numerical amounts, written numbers, and abbreviated formats. The model includes robust parsing logic to prevent misinterpretation of financial amounts.

Local Knowledge Integration: Incorporates specific information about Kentucky state income tax (5% flat rate), Jefferson County property tax rates (approximately 0.91% average), Louisville median home prices ($200,000-$250,000), and local financial resources including Louisville Metro Government services.

Louisville Expert V2

Technical Specifications: - Base Model: llama3.2_balanced - Temperature: 0.4 - Top-p: 0.8 - Token Limit: 1200 - Repeat Penalty: 1.1

Specialization Focus: Local resource integration and guidance for Louisville Metro and Jefferson County, including JCPS educational options, TARC transportation, Metro Government services, neighborhood characteristics, and local business resources.

Performance Characteristics: Provides accurate, current information about local resources with practical implementation guidance. Response times typically range from 10-20 seconds with focused, actionable recommendations.

Knowledge Domains: Comprehensive coverage of JCPS school districts and programs, TARC routes and schedules, Metro Government services and programs, neighborhood characteristics and amenities, local business and professional services, and community resources and programs.

Update Protocol: Local knowledge is maintained through regular review and updating of system prompts to ensure accuracy and currency of information about local resources and services.

Performance Metrics

System Reliability

Current system reliability stands at 80% for production use cases, with validated performance across financial planning, local resource integration, and strategic analysis domains. Reliability metrics are measured across multiple dimensions including completion rates, response quality, and user satisfaction.

Chain Completion Rate: 95% of initiated chains complete successfully within acceptable timeframes, with failures typically attributed to system resource constraints rather than model configuration issues.

Response Quality Consistency: 85% of responses meet quality standards for comprehensiveness, accuracy, and actionability, with variations primarily occurring in edge cases or unusual query patterns.

Performance Stability: Response times remain consistent across different usage patterns and system loads, with 90% of queries completing within expected timeframes.

Response Time Analysis

Response time performance varies by model and usage pattern, with optimization focused on maintaining practical usability while ensuring comprehensive analysis quality.

Individual Model Performance: - Financial Planning Expert V5: 15-25 seconds average - Louisville Expert V2: 10-20 seconds average
- Enhanced Agent Fast V3: 15-25 seconds average

Chain Execution Performance: - Two-model chains: 30-50 seconds average - Three-model chains: 45-75 seconds average - Complex analysis chains: 60-90 seconds maximum

Performance Optimization: Response times are optimized through parameter tuning, system prompt efficiency, and resource management. Current performance levels support practical real-time usage for personal life automation.

Resource Utilization

System resource utilization is optimized to balance performance with system stability, ensuring that Mirador operates effectively within typical MacBook hardware constraints.

Memory Usage: Individual models typically consume 2-4GB RAM during execution, with peak usage during chain execution reaching 6-8GB for three-model chains. Memory management ensures efficient allocation and cleanup between executions.

Processing Requirements: CPU utilization varies by model complexity and query requirements, with optimization focused on minimizing processing time while maintaining response quality.

Storage Requirements: Model storage requires approximately 15-20GB total disk space, with output storage growing based on usage patterns. Regular cleanup of old outputs maintains manageable storage requirements.

Integration Protocols

Ollama Integration

Mirador integrates deeply with the Ollama runtime environment to provide seamless model execution and management. Understanding this integration is essential for system administration and troubleshooting.

Model Management: All Mirador models are managed through Ollama's model registry, enabling consistent versioning, updates, and rollback capabilities. Model creation and updates follow standardized procedures that ensure compatibility and reliability.

Runtime Coordination: The mirador-ez interface coordinates with Ollama to manage model loading, execution scheduling, and resource allocation. This coordination ensures optimal performance and prevents resource conflicts.

Error Handling: Integration includes comprehensive error handling for model loading failures, execution timeouts, and resource constraints. Error recovery procedures ensure system stability and user experience quality.

Version Control Integration

Comprehensive version control integration enables safe development, experimentation, and collaboration while maintaining system stability and reliability.

Configuration Tracking: All model configurations, system prompts, and parameter settings are tracked through Git version control, enabling precise rollback and change management.

Development Workflow: Structured development workflows support the creation of new models, optimization of existing configurations, and testing of experimental features without impacting production stability.

Backup and Recovery: Automated backup procedures ensure that system configurations and critical data are protected against loss or corruption, with recovery procedures that restore full functionality quickly.

External Tool Integration

While Mirador operates primarily as a standalone system, integration capabilities enable connection with external tools and services for enhanced functionality.

Calendar Integration: Potential integration with calendar systems enables proactive analysis and recommendations based on upcoming events and deadlines.

Financial Tool Integration: Integration capabilities support connection with financial tracking tools and services for enhanced budget analysis and optimization.

Local Service Integration: APIs and data feeds from local government and service providers can enhance the accuracy and currency of local knowledge integration.

Security Architecture

Privacy Protection

Mirador implements comprehensive privacy protection through local execution and data isolation, ensuring that sensitive personal information remains under complete user control.

Local Processing: All AI model execution occurs locally within the user's computing environment, eliminating external data transmission and ensuring complete privacy protection for sensitive personal information.

Data Isolation: Personal data and analysis results are stored locally with no external synchronization or backup unless explicitly configured by the user. This isolation ensures that sensitive information cannot be accessed by external parties.

Network Independence: Normal operation requires no network connectivity, eliminating potential data leakage through network communications. Network access is only required for initial model downloads and optional updates.

Access Control

System access control ensures that Mirador operates securely within the user's computing environment while preventing unauthorized access to sensitive information.

User Authentication: System access is controlled through standard macOS user authentication, leveraging existing security infrastructure rather than implementing separate authentication systems.

File System Security: Model configurations and output files are protected through standard file system permissions, ensuring that only authorized users can access sensitive information.

Process Isolation: Ollama and Mirador processes operate with appropriate privilege levels, preventing unauthorized system access while maintaining necessary functionality.

Data Protection

Comprehensive data protection measures ensure that personal information and analysis results are protected against loss, corruption, and unauthorized access.

Encryption Support: File system encryption provides additional protection for sensitive data, with support for macOS FileVault and other encryption systems.

Backup Security: Backup procedures include security considerations to ensure that backup data receives appropriate protection while maintaining accessibility for recovery purposes.

Audit Capabilities: System logging and audit capabilities enable monitoring of access patterns and system usage for security analysis and compliance requirements.

This technical documentation provides the foundation for understanding, administering, and optimizing your Mirador personal AI orchestration framework. Regular review and updates ensure that the documentation remains current and useful as the system evolves.