# Metis RAG Project Tasks

This document outlines the current tasks, priorities, and TODOs for the Metis RAG project. It serves as a working document to track progress and guide development efforts.

## Current Priorities

1. ✅ **Fix Memory Functionality Issues**
2. ✅ **Fix Performance and Authentication Issues**
3. **Improve RAG Response Accuracy**
4. **Enhance System Prompts**
5. **Optimize Document Processing**

## Task Breakdown

### Memory Functionality Fixes

* **Fix Frontend Session Management**
  + Add localStorage persistence for conversation IDs
  + Create updateConversationId() function
  + Modify page load event to retrieve stored conversation IDs
  + Fix conversation ID handling in streaming responses
* **Address Authentication Issues**
  + Increase JWT token expiration time
  + Implement token refresh functionality
  + Add frontend token management
  + Update authenticatedFetch to handle token refresh
* **Ensure Conversation Continuity**
  + Modify RAG engine to maintain consistent user IDs
  + Fix conversation history handling
  + Update conversation repository
  + Fix memory buffer integration
* **Fix Foreign Key Violation in Memory Storage**
  + Modify process\_query in memory\_buffer.py to convert string conversation\_ids to UUID objects
  + Add conversation existence verification in add\_to\_memory\_buffer
  + Add proper error handling and logging for memory operations
  + Update type hints to make the expected types clearer
  + Add unit tests to verify the fixes
* **Fix Session Management and ID Lifecycle Issues**
  + Remove manual session closing in memory\_buffer.py that causes IllegalStateChangeError
  + Implement proper error handling for session operations with try/except/finally blocks
  + Ensure sessions are automatically closed when the generator is closed
  + Fix double yield issue in app/db/session.py get\_session function
* **Fix Conversation ID Persistence Timing**
  + Add validation to ensure the conversation ID is properly handled throughout the system
  + Ensure memory operations use the same conversation ID that exists in the database
  + Modify the RAG engine to always respect the conversation ID passed from the API
  + Remove code that generates a new conversation ID in the RAG engine
* **Fix User ID Format Inconsistency**
  + Ensure all user IDs are valid UUIDs throughout the system
  + Generate proper UUIDs for anonymous users instead of string-based session IDs
  + Add proper validation and conversion of user IDs to ensure database compatibility
  + Implement deterministic UUID generation for cases where conversion fails
* **Implement Memory Usage Optimization**
  + Add memory cleanup after query processing to reduce memory usage
  + Implement garbage collection to free up memory
  + Clear cached data and large temporary variables
  + Prevent high memory usage (99.5%) that causes task throttling
* **Add Diagnostics and Testing**
  + Implement memory diagnostics endpoint
  + Add detailed logging for memory operations
  + Create comprehensive memory testing script

### RAG Response Accuracy Improvements

* **Enhance “No Documents Found” Handling**
  + Modify context formatting for empty results
  + Add prominent notice when no documents are found
  + Improve messaging for insufficient context
* **Improve Vector Store Empty Results Handling**
  + Enhance search method to provide clearer empty results
  + Add special result that clearly indicates no documents were found
  + Improve logging for empty search results
* **Fix Citation Handling**
  + Modify context formatting to only include citations when documents exist
  + Ensure empty results clear the sources list
  + Improve citation formatting in responses
* **Add Verification Step Before Response Generation**
  + Add check for sources before generating response
  + Add explicit instruction to not use citations when no sources are found
  + Improve logging for verification step

### System Prompt Enhancements

* **Improve RAG System Prompt**
  + Add stronger instructions for document hallucination prevention
  + Enhance memory retention instructions
  + Add content fabrication prevention guidelines
  + Implement citation misuse prevention
* **Implement User Information Extraction**
  + Add mechanism to extract user information from conversation
  + Create patterns to identify user names
  + Include user information in context for response generation

### Document Processing Optimization

* **Enhance Chunking Strategy**
  + Refine dynamic chunking parameter selection
  + Improve entity preservation in chunks
  + Optimize chunk size and overlap parameters
* **Improve Vector Embedding Performance**
  + Implement batch processing for embeddings
  + Add caching for frequently accessed embeddings
  + Optimize vector database queries

## Testing Plan

* **Memory Functionality Tests**
  + Test conversation persistence across page refreshes
  + Verify memory storage and retrieval
  + Test memory operations in multi-turn conversations
* **RAG Accuracy Tests**
  + Test empty vector store scenarios
  + Test no relevant documents scenarios
  + Test user information retention
  + Test citation handling
* **Performance Tests**
  + Measure document processing times
  + Evaluate response generation latency
  + Test system under various load conditions

## Bug Fixes

* **Fix Document Hallucination Issues**
  + Address system claiming to have non-existent documents
  + Fix improper handling of empty search results
* **Fix Memory Loss Issues**
  + Address system forgetting user information
  + Fix conversation history not being effectively utilized
  + Fix foreign key violation when storing implicit memories
  + Fix session management issues causing IllegalStateChangeError
  + Fix conversation ID persistence timing issues
  + Fix user ID format inconsistency problems
* **Fix Performance and Authentication Issues**
  + Fix incorrect query type detection causing wrong system prompt usage
  + Fix analytics endpoint authentication failures
  + Improve memory management to reduce high memory usage
  + Fix user ID edge cases with “system” and session-based IDs
  + Add performance monitoring to identify bottlenecks
* **Fix Content Fabrication Issues**
  + Prevent generation of text for non-existent documents
  + Improve handling of “no documents found” scenarios
* **Fix Citation Misuse**
  + Prevent use of citation markers when no documents are referenced
  + Ensure citations correspond to actual documents

## Documentation Updates

* **Update API Documentation**
  + Document memory-related endpoints
  + Update authentication endpoints documentation
  + Document new system configuration options
* **Update User Guide**
  + Add memory command usage instructions
  + Update document upload guidelines
  + Add troubleshooting section for common issues
* **Update Developer Documentation**
  + Document memory buffer implementation
  + Update system prompt configuration guide
  + Add testing guidelines for new features
  + Document ID lifecycle and persistence throughout the system
  + Add session management best practices

## Next Steps

1. ✅ Implement the memory functionality fixes to address the most critical issues
2. ✅ Test the memory functionality to ensure it works correctly
3. ✅ Fix the remaining issues identified in log analysis (query classification, analytics auth, memory management, user ID edge cases, performance)
4. Proceed with RAG response accuracy improvements
5. Implement and test the remaining enhancements
6. Update documentation to reflect the changes
7. Create comprehensive unit tests for all fixed components

## Completion Criteria

* ✅ All memory functionality issues are resolved
* ✅ Performance and authentication issues are fixed
* RAG responses accurately reflect available documents
* System prompts effectively guide the LLM behavior
* Document processing is optimized for performance
* Comprehensive tests verify all functionality
* Documentation is updated to reflect all changes

## Metis RAG System Fixes - Implementation Plan

### 1. Root Cause Analysis

#### Issue 1: Conversation ID Persistence Timing

* The API creates a conversation in the database and passes its ID to the RAG engine
* The RAG engine ignores this ID and generates a new one that isn’t persisted in the database
* When memory\_buffer.py tries to use this new ID, it fails to find the conversation in the database

#### Issue 2: User ID Format Inconsistency

* The system expects UUIDs for user\_id in the database
* The RAG engine generates string-based session IDs (e.g., “session\_bb90aa96”) when no authenticated user is present
* These string IDs are incompatible with database columns expecting UUIDs

#### Issue 3: Session Management

* SQLAlchemy async sessions are not being properly managed
* Sessions are being closed while operations are still in progress

### 2. Implementation Details

#### Fix 1: Conversation ID Consistency

* Replace the conversation\_id generation logic in rag\_engine.py
* Always use the provided conversation\_id
* Log warnings when no conversation\_id is provided

#### Fix 2: User ID Handling

* Replace the user\_id generation logic in rag\_engine.py
* Generate proper UUIDs for anonymous users
* Ensure user\_id is a valid UUID
* Generate deterministic UUIDs for invalid user\_id formats

#### Fix 3: Session Management

* Improve get\_session function in session.py
* Implement proper session cleanup
* Add error handling for session operations

#### Fix 4: Memory Optimization

* Implement memory cleanup in rag\_engine.py
* Force garbage collection after query processing
* Clear cached data and large temporary variables

### 3. Testing Plan

1. **Unit Tests**:
   * Test session management with concurrent operations
   * Test UUID conversion and validation
   * Test conversation ID handling
2. **Integration Tests**:
   * Test the full flow from API to RAG engine to memory buffer
   * Verify conversation IDs are consistent throughout the system
   * Verify memory operations work correctly
3. **Load Tests**:
   * Test memory usage under load
   * Verify the system can handle multiple concurrent requests

## Completed Implementation Tasks

* **Fixed session.py to properly manage async sessions** (April 6, 2025)
  + Improved session tracking with session\_yielded flag
  + Added proper rollback handling before closing sessions
  + Added garbage collection to clean up lingering references
  + Improved error handling for session closing
* **Fixed memory\_buffer.py to properly handle sessions** (April 6, 2025)
  + Properly tracked session generators to ensure they can be closed
  + Added proper finally blocks to ensure sessions are closed
  + Used aclose() on the session generator to trigger cleanup
  + Separated session creation and management logic
* **Fixed user ID format inconsistency in the RAG engine** (April 6, 2025)
  + Added more robust type checking for user IDs
  + Used deterministic UUID generation for consistent mapping
  + Updated the user\_id variable after conversion for consistency
  + Improved error handling for invalid user ID formats
* **Fixed conversation ID persistence in the RAG engine** (April 6, 2025)
  + Ensured the RAG engine respects the conversation ID passed from the API
  + Only extracted conversation\_id from conversation\_history if not already provided
  + Properly passed the conversation\_id to the memory buffer
* **Fixed test\_fixes.py to properly manage sessions** (April 6, 2025)
  + Used session generators correctly with proper cleanup
  + Properly checked session state after closing
  + Used async context managers for session management
  + Fixed the test assertions to be more reliable
* **Run the test\_fixes.py script to verify the fixes** (April 6, 2025)
  + Execute the test script to verify session management
  + Verify conversation ID handling
  + Verify memory operations with valid conversation IDs
* **Address any remaining issues found during testing** (April 6, 2025)
  + Fix any issues discovered during testing
  + Update documentation to reflect the changes

## Completed Implementation Tasks (April 6, 2025)

* **Fix Session Management in test\_conversation\_id\_persistence**
  + Modify the test to use separate sessions for RAG engine and database operations
  + Ensure sessions are not closed while they’re still being used
  + Add proper error handling for session operations
  + Implement more robust session lifecycle management
* **Fix RAG Engine to Accept a Database Session**
  + Add db parameter to query method
  + Pass the provided session to memory buffer operations
  + Ensure the session is not closed prematurely
  + Add proper error handling for session operations
* **Fix User ID Format Inconsistency**
  + Ensure valid UUIDs are passed to the RAG engine
  + Fix the root cause of passing invalid string IDs
  + Update tests to use valid UUID strings
  + Add validation for user IDs at the API level
* **Improve Session Management in memory\_buffer.py**
  + Enhance process\_query function to be more robust
  + Add better logging for session operations
  + Ensure sessions are properly tracked and closed
  + Add proper error handling for concurrent operations
* **Improve Session Management in session.py**
  + Enhance get\_session function to handle concurrent operations better
  + Add more robust error handling for session operations
  + Implement better connection pool management
  + Add more detailed logging for session lifecycle events

## Completed Implementation Tasks (April 6, 2025 - Part 2)

* **Fix Incorrect Query Type Detection**
  + Refine code keywords list in \_is\_code\_related\_query method to avoid false positives
  + Add negative patterns (terms that indicate it’s NOT a code query)
  + Implement word boundary checks to prevent partial matches
  + Add more detailed logging of matched keywords/patterns
  + Fix issue with “RAG summary” query being incorrectly classified as code-related
* **Fix Analytics Endpoint Authentication**
  + Modify \_record\_analytics method to include authentication headers
  + Implement a method to get a system JWT token for internal API calls
  + Use environment variables for API endpoint configuration
  + Add retry logic for failed analytics requests
  + Ensure analytics failures don’t impact the main application flow
* **Improve Memory Management**
  + Enhance the \_cleanup\_memory method to be more aggressive
  + Implement more detailed memory usage logging
  + Add adaptive behavior based on system load
  + Clear additional caches and temporary variables
  + Add emergency cleanup for critical memory situations
* **Fix User ID Edge Cases**
  + Ensure all user ID generation paths create valid UUIDs
  + Replace any “system” or session-based IDs with proper UUIDs
  + Add validation for user IDs before database operations
  + Log warnings for invalid user IDs
  + Convert invalid IDs to valid UUIDs when necessary
* **Optimize Performance**
  + Add detailed timing logs for each step of response generation
  + Identify bottlenecks in the process
  + Streamline the prompt creation process
  + Add timing summaries for performance analysis
  + Enhance the caching strategy for frequently used prompts
* **Fix Additional Issues (April 6, 2025 - Part 3)**
  + Add api\_base\_url attribute to SETTINGS object in app/core/config.py
  + Fix \_record\_analytics method to use api\_base\_url correctly
  + Create test script to verify specific fixes
  + Verify fixes with automated tests

## Completed Implementation Tasks (April 6, 2025 - Part 4)

* **Fix Analytics Recording Error**
  + Fix method name mismatch in app/api/analytics.py
  + Change analytics\_repository.create\_analytics\_query() to analytics\_repository.log\_query()
  + Ensure analytics data is properly recorded in the database
  + Verify analytics recording works correctly
* **Improve Query Classification Logic**
  + Enhance the \_is\_code\_related\_query function in app/rag/rag\_engine\_base.py
  + Add specific detection for creative/narrative content (stories, fiction, etc.)
  + Prioritize creative content indicators over code keywords
  + Require multiple context clues for programming language mentions
  + Fix misclassification of queries like “tell me a fictional story”
* **Improve API Error Handling**
  + Update the query\_chat endpoint in app/api/chat.py to handle post-processing errors separately
  + Add try-except blocks around non-critical operations
  + Add a warnings list to track non-critical issues
  + Update the ChatResponse model to include the new warnings field
  + Ensure that even if analytics or other post-processing fails, the user still gets a response

## Next Implementation Tasks (April 7, 2025)

* **Create Unit Tests for Query Classification**
  + Create test cases for code-related queries
  + Create test cases for non-code queries
  + Create test cases for edge cases
  + Verify classification accuracy
  + Add regression tests for fixed issues
* **Implement Memory Usage Monitoring**
  + Create a memory usage dashboard
  + Set up alerts for high memory usage
  + Implement automatic scaling based on memory usage
  + Add memory usage metrics to analytics
  + Create a memory usage report generator
* **Optimize Caching Strategy**
  + Analyze cache hit/miss rates
  + Implement more efficient cache key generation
  + Add cache warming for frequently used prompts
  + Implement cache eviction policies
  + Add cache statistics to analytics
* **Enhance Analytics System**
  + Create a comprehensive analytics dashboard
  + Add more detailed query performance metrics
  + Implement user behavior analytics
  + Add document usage analytics
  + Create periodic analytics reports
* **Improve Error Handling and Logging**
  + Implement centralized error handling
  + Add structured logging for all components
  + Create error reporting dashboard
  + Implement automatic error notification
  + Add error trend analysis
  + Identify bottlenecks in the process
  + Streamline the prompt creation process
  + Reduce unnecessary string operations
  + Enhance the caching strategy for frequently used prompts