**Callisto Memory Database Implementation Plan**

Based on the documentation, I'll outline a staged implementation plan for Callisto using Python-compatible technologies.

**Recommended Technologies**

* **Database**: SQLite (via sqlite3)
* **ORM**: SQLAlchemy for clean abstraction
* **Validation**: Pydantic for data models/validation
* **Caching**: In-memory dict cache with SQLite persistence

**Implementation Stages**

**Stage 1: Core Database Setup**

1. Create SQLAlchemy models matching the schema
2. Set up database initialisation and migration system
3. Implement connection pooling and atomic transaction handling
4. Add basic logging and error recovery

**Stage 2: Memory API Development**

1. Create abstract base classes defining the API interface
2. Implement user management (CRUD operations)
3. Build platform integration layer
4. Add essential security features (input sanitisation, validation)

**Stage 3: Knowledge Management**

1. Implement knowledge category management
2. Build user knowledge storage/retrieval methods
3. Add confidence scoring system
4. Develop source tracking mechanisms

**Stage 4: Conversation Integration**

1. Create conversation logging systems
2. Implement message storage with proper indexing
3. Build extraction job queue management
4. Add asynchronous processing capabilities

**Stage 5: Performance Optimisation**

1. Add in-memory caching layer
2. Implement secure deletion procedures
3. Optimise query performance with additional indices
4. Create configuration-driven memory categories

Would you like me to develop a specific stage in more detail, or shall I begin with code samples for Stage 1?