**Local LLM Chat App - Problem Statement & Solution Architecture**

A real-time chat application that provides conversational AI capabilities through locally hosted Large Language Models (LLMs), eliminating privacy concerns while reducing costs and improving control.

### **Component Architecture**

#### **1. Frontend Layer (Streamlit)**

Streamlit Chat Application

├── User Interface Components

│ ├── Chat Display Area

│ │ ├── Message Rendering

│ │ ├── Markdown Support

│ │ ├── Code Highlighting

│ │ └── Timestamp Display

│ ├── Input Processing

│ │ ├── Text Input Validation

│ │ ├── Command Parsing

│ │ ├── File Upload Support

│ │ └── Keyboard Shortcuts

│ └── Real-time Updates

│ ├── Auto-refresh

│ ├── Typing Indicators

│ └── Status Messages

├── Configuration Panel

│ ├── Model Selection

│ │ ├── Available Models List

│ │ ├── Model Information

│ │ └── Switch Model

│ ├── Chat Settings

│ │ ├── Context Length

│ │ ├── Temperature

│ │ ├── Max Tokens

│ │ └── Response Format

│ └── Session Management

│ ├── Save Conversation

│ ├── Load History

│ ├── Export Chat

│ └── Clear Session

└── State Management

├── Session Persistence

├── Message History

├── User Preferences

└── Error Handling

#### **2. Backend Layer (FastAPI)**

FastAPI Backend Service

├── API Endpoints

│ ├── Chat Endpoints

│ │ ├── POST /chat

│ │ ├── GET /conversation/{id}

│ │ ├── DELETE /conversation/{id}

│ │ └── WebSocket /ws/chat

│ ├── Model Management

│ │ ├── GET /models

│ │ ├── POST /models/load

│ │ ├── DELETE /models/unload

│ │ └── GET /models/{id}/info

│ └── System Endpoints

│ ├── GET /health

│ ├── GET /metrics

│ ├── GET /status

│ └── POST /config

├── Business Logic

│ ├── Chat Engine

│ │ ├── Context Assembly

│ │ ├── Prompt Engineering

│ │ ├── Response Processing

│ │ └── Stream Handling

│ ├── Model Interface

│ │ ├── Ollama Integration

│ │ ├── Model Lifecycle

│ │ ├── Load Balancing

│ │ └── Error Recovery

│ └── Session Management

│ ├── Session Creation

│ ├── Context Tracking

│ ├── Memory Management

│ └── Cleanup

└── Infrastructure

├── Configuration Management

├── Logging System

├── Error Handling

├── Rate Limiting

└── Security Layer

#### **3. AI Engine Layer (Ollama)**

Ollama Local AI Service

├── Model Management

│ ├── Model Registry

│ │ ├── Available Models

│ │ ├── Model Metadata

│ │ ├── Version Control

│ │ └── Dependencies

│ ├── Model Lifecycle

│ │ ├── Download & Install

│ │ ├── Load & Unload

│ │ ├── Update & Patch

│ │ └── Remove & Cleanup

│ └── Resource Management

│ ├── Memory Allocation

│ ├── GPU Utilization

│ ├── CPU Scheduling

│ └── Storage Management

├── Inference Engine

│ ├── Request Processing

│ │ ├── Input Validation

│ │ ├── Tokenization

│ │ ├── Context Preparation

│ │ └── Batch Processing

│ ├── Model Execution

│ │ ├── Forward Pass

│ │ ├── Attention Mechanism

│ │ ├── Generation Logic

│ │ └── Sampling Strategy

│ └── Response Generation

│ ├── Token Decoding

│ ├── Text Generation

│ ├── Formatting

│ └── Post-processing

└── System Integration

├── API Interface

├── Performance Monitoring

├── Error Handling

└── Security Controls

## **Data Flow Architecture**

### **Real-time Chat Flow**

1. User Input Entry

↓

2. Frontend Validation

↓

3. API Request Formation

↓

4. Backend Processing

├── Context Assembly

├── Model Selection

└── Request Preparation

↓

5. Ollama Inference

├── Model Loading

├── Token Processing

└── Response Generation

↓

6. Response Processing

├── Format Validation

├── Content Filtering

└── Streaming Preparation

↓

7. Frontend Display

├── Real-time Rendering

├── Markdown Processing

└── History Update