Math Mentor AI - System Architecture Document

Document Information

Field	Value
Project	Math Mentor AI
Version	1.0
Date	January 2026
Author	AI Planet Assessment

---

1. Executive Summary

Math Mentor AI is an intelligent math tutoring system designed for JEE-level problems. It combines:

- **Multimodal Input**: Text, Image (OCR), Audio (ASR)
- **RAG-Powered Knowledge**: 17 documents, 571 chunks
- **Multi-Agent Architecture**: 5 specialized AI agents
- **Human-in-the-Loop (HITL)**: Confidence-based intervention
- **Self-Learning Memory**: Learns from user feedback

---

2. High-Level Architecture

2.1 System Overview Diagram

```
┌──────────────────────────────────────────────────────────────────────
│                          USER INTERFACE
│                       (Streamlit Web App)
│
│   ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  ┌─────────────
│   │  Text Input  │  │ Image Upload │  │Audio Record  │  │  Settings Panel
│   └──────────────┘  └──────────────┘  └──────────────┘  └─────────────
│          │                 │                 │
└──────────────────────────────────────────────────────────────────────
           │                 │                 │
           ▼                 ▼                 ▼
┌──────────────────────────────────────────────────────────────────────
│                     INPUT PROCESSING LAYER
│
│   ┌──────────────┐  ┌──────────────┐  ┌──────────────┐
│   │Text Processor│  │OCR Processor │  │ASR Processor │
│   │   (Direct)   │  │  (EasyOCR)   │  │  (Gemini)    │
│   │  + LLM Fix   │  │  + LLM Fix   │  │              │
│   └──────────────┘  └──────────────┘  └──────────────┘
│          │                 │                 │
└──────────────────────────────────────────────────────────────────────
           │                 │                 │
           └─────────────────┼─────────────────┘
                             │
```

```
                              ▼
┌──────────────────────────────────────────────────────────────────────
│                     AGENT ORCHESTRATION LAYER
│
│  ┌──────────┐    ┌──────────┐    ┌──────────┐    ┌──────────┐    ┌──
│  │ PARSER   │───▶│ ROUTER   │───▶│ SOLVER   │───▶│ VERIFIER │───▶│EX
│  │ AGENT    │    │ AGENT    │    │ AGENT    │    │ AGENT    │    │ AG
│  └──────────┘    └──────────┘    └──────────┘    └──────────┘    └──
│                                       │
│                                       ▼
│                             ┌────────────────┐
│                             │  RAG SYSTEM    │
│                             │ (FAISS Index)  │
│                             └────────────────┘
└──────────────────────────────────────────────────────────────────────
                                      │
                                      ▼
┌──────────────────────────────────────────────────────────────────────
│                          EXTERNAL SERVICES
│  ┌────────────────┐   ┌────────────────┐   ┌────────────────┐
│  │ Google Gemini  │   │ FAISS Vector   │   │ SQLite Memory  │
│  │  LLM API       │   │  Database      │   │     Store      │
│  └────────────────┘   └────────────────┘   └────────────────┘
└──────────────────────────────────────────────────────────────────────
```

2.2 Technology Stack

Layer	Technology	Purpose
Frontend	Streamlit	Web UI Framework
LLM	Google Gemini 2.0 Flash	AI Reasoning
Embeddings	all-MiniLM-L6-v2	Vector Embeddings
Vector DB	FAISS	Similarity Search
OCR	EasyOCR + Gemini	Image Text Extraction
ASR	Gemini Audio	Speech Transcription
Database	SQLite	Memory/Feedback Storage
Language	Python 3.10+	Core Development

---

3. Low-Level Design

3.1 Agent Orchestration Pipeline

```
┌──────────────────────────────────────────────────────────────────────
│                       DETAILED AGENT PIPELINE
└──────────────────────────────────────────────────────────────────────

STEP 1: PARSER AGENT
┌──────────────────────────────────────────────────────────────────────
│ Input: Raw problem text
│
```

```
│ Process:
│   1. Clean and normalize text
│   2. Detect mathematical topic (algebra, calculus, etc.)
│   3. Extract variables and constraints
│   4. Check for ambiguities
│
│ Output:
│   {
│     "problem_text": "Solve x² - 5x + 6 = 0",
│     "detected_topic": "algebra",
│     "variables": ["x"],
│     "needs_clarification": false
│   }
│
│ HITL Trigger: If ambiguous → Ask user for clarification
└───────────────────────────────────────────────────────────────────────
                                      │
                                      ▼
STEP 2: ROUTER AGENT
┌───────────────────────────────────────────────────────────────────────
│ Input: Parsed problem structure
│
│ Process:
│   1. Map topic to solver strategy
│   2. Decide: use_rag, use_calculator
│   3. Set RAG retrieval filters
│   4. Estimate difficulty level
│
│ Output:
│   {
│     "solver_type": "algebraic_solver",
│     "use_rag": true,
│     "use_calculator": true,
│     "difficulty": "intermediate",
│     "rag_filters": ["algebra", "quadratic"]
│   }
└───────────────────────────────────────────────────────────────────────
                                      │
                                      ▼
STEP 3: SOLVER AGENT (Core Engine)
┌───────────────────────────────────────────────────────────────────────
│ Input: Problem + Routing config
│
│ Process:
│   1. Query RAG system for relevant knowledge chunks
│   2. Check memory for similar solved problems
│   3. Build context-enriched prompt
│   4. Generate solution with Gemini LLM
│   5. Extract citations and confidence
│
│ RAG Integration:
│   ┌───────────────────────────────────────────────────────────────────
│   │ Query: "quadratic equation x² - 5x + 6"
```

```
│    │
│    │ Retrieved Chunks:
│    │   1. [algebra.md] "Factoring: (x-a)(x-b) = x² - (a+b)x + ab"
│    │   2. [algebra.md] "Zero Product Property: if ab=0, then a=0 or b=
│    │   3. [jee_tips.md] "Always verify roots by substitution"
│    │
│    │ Similarity Scores: [0.92, 0.88, 0.75]
│    └──────────────────────────────────────────────────────────────────
│
│ Output:
│   {
│     "solution": "x = 2 or x = 3",
│     "reasoning_steps": [...],
│     "citations": ["algebra.md", "jee_tips.md"],
│     "confidence": 0.92
│   }
└───────────────────────────────────────────────────────────────────────
                                       │
                                       ▼
STEP 4: VERIFIER AGENT
┌───────────────────────────────────────────────────────────────────────
│ Input: Solution from Solver
│
│ Process:
│   1. Validate logical consistency
│   2. Check domain constraints
│   3. Verify arithmetic (using Python calculator)
│   4. Calculate final confidence score
│
│ Confidence Calculation:
│   final_conf = weighted_avg(
│       rag_coverage: 0.9,     # weight: 0.25
│       citation_quality: 0.9, # weight: 0.20
│       llm_confidence: 0.92,  # weight: 0.30
│       verification: 0.95,    # weight: 0.25
│   ) = 0.93
│
│ Output:
│   {
│     "verdict": "pass",
│     "confidence": 0.93,
│     "issues": []
│   }
│
│ HITL Trigger: If confidence < 70% → Request human review
└───────────────────────────────────────────────────────────────────────
                                       │
                                       ▼
STEP 5: EXPLAINER AGENT
┌───────────────────────────────────────────────────────────────────────
│ Input: Verified solution
│
│ Process:
```

```
│   1. Convert technical solution to student-friendly format
│   2. Add step-by-step breakdown
│   3. Include key concepts
│   4. Add JEE exam tips
│
│ Output:
│   {
│     "final_explanation": "Step 1: Factor x² - 5x + 6...",
│     "key_concepts": ["Quadratic Equations", "Zero Product Property"],
│     "exam_tips": ["Always verify by substitution"]
│   }
└──────────────────────────────────────────────────────────────────────
```

3.2 RAG System Architecture

```
┌──────────────────────────────────────────────────────────────────────
│                        RAG SYSTEM COMPONENTS
└──────────────────────────────────────────────────────────────────────

                    ┌─────────────────────────────────────┐
                    │          KNOWLEDGE BASE             │
                    │     (17 Markdown Documents)         │
                    │                                     │
                    │  ├── algebra.md                     │
                    │  ├── calculus.md                    │
                    │  ├── trigonometry.md                │
                    │  ├── probability.md                 │
                    │  ├── matrices.md                    │
                    │  ├── coordinate_geometry.md         │
                    │  ├── vectors_3d.md                  │
                    │  ├── complex_numbers.md             │
                    │  ├── sequences_series.md            │
                    │  ├── differential_equations.md      │
                    │  ├── binomial_theorem.md            │
                    │  ├── permutations_combinations.md   │
                    │  ├── linear_algebra.md              │
                    │  ├── statistics.md                  │
                    │  ├── common_mistakes.md             │
                    │  ├── problem_templates.md           │
                    │  └── jee_tips.md                    │
                    └─────────────────┬───────────────────┘
                                      │
                                      ▼
┌──────────────────────────────────────────────────────────────────────
│                        INDEX BUILDING PIPELINE
│
│   1. Load Documents     2. Chunk Text        3. Generate Embeddings
│   ─────────────────     ────────────         ───────────────────
│   Read .md files        Split into ~500     all-MiniLM-L6-v2
│   Extract headers       char chunks with     384-dim vectors
│   Parse content         100 char overlap
│
│   4. Build Index        5. Store Metadata
```

```
│
│   FAISS IndexFlatIP      chunks.pkl
│   (Inner Product)        metadata.json
│
│   Result: 571 chunks indexed with semantic search capability
└
```

```
┌
│                          RETRIEVAL PIPELINE
│
│   Query: "How to solve quadratic equations"
│
│   1. Embed Query           2. FAISS Search          3. Filter & Rank
│
│   all-MiniLM-L6-v2         k=5 nearest neighbors    Score threshold: 0.5
│   384-dim vector           Inner product score     Return top matches
│
│   Output: Top-5 relevant knowledge chunks with sources
└
```

3.3 Memory and Self-Learning System

```
┌
│                       MEMORY & SELF-LEARNING SYSTEM
└
```

```
┌
│                          SQLite Database Schema
│
│   TABLE: problems
│
│   ┌
│   │ id            │ problem_text  │ solution   │ confidence  │ feedback
│   │ (PRIMARY)     │ (TEXT)        │ (TEXT)     │ (FLOAT)     │ (TEXT)
│   ├               ┼               ┼            ┼             ┼
│   │ uuid-1        │ "Solve x²"    │ "x=2,3"    │ 0.93        │ "correct"
│   │ uuid-2        │ "∫x²dx"       │ "x³/3+C"   │ 0.87        │ "incorrect"
│   └               ┴               ┴            ┴             ┴
│
│   TABLE: feedback_log
│
│   ┌
│   │ problem_id   │ feedback_type   │ user_correction  │ timestamp
│   │ (FK)         │ (TEXT)          │ (TEXT)           │ (DATETIME)
│   └              ┴                 ┴                  ┴
│
└
```

```
SELF-LEARNING FLOW:
┌
│
│   User Feedback          Memory Check             Solution Enhancement
│
│   ✓ Correct              On new problem:          If similar problem fou
│   ✗ Incorrect            Search memory for        - Use cached solution
│
```

```
│                            similar problems        - Boost confidence
│   If incorrect:            by embedding            - Add source: "memory"
│   - Store correction       similarity
│   - Learn pattern
│
└──────────────────────────────────────────────────────────────────────
```

3.4 Input Processing Flow

```
┌──────────────────────────────────────────────────────────────────────
│                          INPUT PROCESSING DETAIL
└──────────────────────────────────────────────────────────────────────

TEXT INPUT:
┌────────────┐    ┌────────────────┐    ┌───────────────────┐
│ User types │───▶│ Text Validator │───▶│ Clean & Normalize │
│ problem    │    │Check empty     │    │Strip whitespace   │
└────────────┘    └────────────────┘    └───────────────────┘
                  Confidence: 95%

IMAGE INPUT (OCR):
┌────────────┐    ┌────────────────┐    ┌───────────────────┐    ┌──────
│ User uploads│──▶│    EasyOCR     │───▶│ LLM Enhancement   │───▶│  Fi
│ image      │    │ Extract text   │    │Fix ÷ × symbols    │    │  + C
└────────────┘    └────────────────┘    └───────────────────┘    └──────
                  Confidence: 70-90%    Confidence: +15%

AUDIO INPUT (ASR):
┌────────────┐    ┌────────────────┐    ┌───────────────────┐    ┌──────
│ User records│──▶│ Base64 Encode  │───▶│ Gemini Audio API  │───▶│  Tr
│ audio      │    │ WebM format   │    │Transcription     │    │ + No
└────────────┘    └────────────────┘    └───────────────────┘    └──────
                                        Confidence: 85%
```

---

4. Data Flow Diagram

```
┌──────────────────────────────────────────────────────────────────────
│                            COMPLETE DATA FLOW
└──────────────────────────────────────────────────────────────────────

User Input (Text/Image/Audio)
         │
         ▼
   ┌───────────────┐
   │    app.py     │   ───────  Session State Management
   │ (Streamlit)   │            (problem_solved, current_result)
   └───────────────┘
         │
         ▼
   ┌───────────────────────┐    ┌────────────────────┐
   │   Input Processor     │───▶│   Extracted Text   │
```

```
│  (OCR/ASR/Text)            │      │  + Confidence      │
└────────────────────────────┘      └────────┬───────────┘
                                             │
                                             ▼
                                   ┌─────────────────────────┐
                                   │  run_agent_pipeline()   │
                                   └─────────┬───────────────┘
                                             │
         ┌───────────────────────────────────┼──────────────────────────────
         │                                   │
         ▼                                   ▼
┌──────────────┐                    ┌──────────────┐                 ┌──────
│   Parser     │                    │     RAG      │                 │   Me
│   Agent      │                    │  Retriever   │                 │   St
└──────┬───────┘                    └──────┬───────┘                 └──────
       │                                   │
       │    Parsed Problem                 │    Knowledge Chunks
       │                                   │
       └───────────────────────────────────┼──────────────────────────────
                                           │
                                           ▼
                                  ┌─────────────────────────┐
                                  │  Router → Solver →      │
                                  │  Verifier → Explainer   │
                                  └─────────┬───────────────┘
                                            │
                                            │    Result Dictionary:
                                            │    {
                                            │      "success": true,
                                            │      "solver": {...},
                                            │      "verifier": {...},
                                            │      "explainer": {...},
                                            │      "trace": [...]
                                            │    }
                                            ▼
                                  ┌─────────────────────────┐
                                  │    Render Solution      │
                                  │    + Agent Trace        │
                                  │    + Feedback Form      │
                                  └─────────────────────────┘
```

---

5. Security & Error Handling

5.1 API Key Management

- Environment variables (.env)
- Streamlit secrets (for cloud)
- Never committed to git

5.2 Error Handling Strategy

Component	Error Type	Handling
LLM Client	JSON Parse Error	Fallback to text parsing
LLM Client	Rate Limit	Retry with backoff
OCR	Low Confidence	LLM enhancement
ASR	Empty Transcript	User notification
Solver	Type Error	Type coercion
RAG	Index Not Found	Load on demand

5.3 HITL Triggers

Scenario	Threshold	Action
OCR Confidence Low	< 75%	Show edit box
Parser Ambiguous	N/A	Show clarification input
Verifier Uncertain	< 70%	Request human review

---

6. Deployment Architecture

```
┌─────────────────────────────────────────────────────────────────────
│                         STREAMLIT CLOUD DEPLOYMENT
└─────────────────────────────────────────────────────────────────────

┌──────────────────┐      ┌──────────────────┐      ┌─────────────────
│   GitHub Repo    │─────▶│ Streamlit Cloud  │─────▶│  Google Gemini
│  (Source Code)   │      │    (Hosting)     │      │    (LLM API)
└──────────────────┘      └──────────────────┘      └─────────────────
                                   │
                                   │ Secrets:
                                   │ - GOOGLE_API_KEY
                                   │
                                   ▼
                          ┌──────────────────┐
                          │   User Browser   │
                          │    (Web UI)      │
                          └──────────────────┘

Files Required:
├── app.py                 # Entry point
├── requirements.txt       # Dependencies
├── packages.txt           # System packages
├── .streamlit/
│   └── config.toml       # Theme config
└── data/
    └── faiss_index/      # Pre-built index
```

---

7. Performance Metrics

Metric	Value