**1️⃣ User Enters the System**

* **Inputs**: Name, grade level, and maybe topic or preferences.
* Your backend checks MongoDB:
  + ✅ If the user exists → greet them
  + ❌ If new → create a new record

**2️⃣ AI Generates a Math Question**

* The app uses your content\_generator.py, powered by GPT:
  + It sends a prompt like:

“Give a Grade 4 student a simple multiplication word problem.”

* + GPT replies with:

“If Maya has 4 baskets with 6 apples each, how many apples does she have in total?”

This question appears in the Streamlit or React UI.

**3️⃣ Student Answers the Question**

* Student types their answer (e.g., “24”)
* You store this in MongoDB along with:
  + Question text
  + Their answer
  + Timestamp
  + Whether it was correct (optional logic)

**4️⃣ Reflection Agent Kicks In**

* After answering, the student is asked:
  + “Why did you choose this answer?”
  + “How confident are you?”
  + Emoji or text input (e.g., 😕 or “Not sure”)
* This response is saved using feedback collection

✅ This builds metacognition: **thinking about their thinking**.

**5️⃣ Sentiment Analysis (Optional AI Module)**

* You analyze emotional feedback:
  + “😕” → confusion
  + “😊” → confident
* Adjust content difficulty or tone accordingly

**6️⃣ Adaptive Logic**

* If the student got several answers wrong:
  + You offer **simpler problems** or **extra hints**
* If the student is performing well:
  + You **increase difficulty** or introduce new topics
* This logic is handled in adaptive\_logic.py

**7️⃣ Track Progress Over Time**

* Every answer, reflection, emotion, and topic is saved
* Over time, the app can:
  + Show the student what they’ve improved in
  + Help teachers or parents see learning gaps
  + Recommend next learning steps

**🎯 The Result?**

A tutor that is:

* Friendly like a real teacher
* Adaptive like a smart app
* Non-judgmental like a diary
* Always available like a 24/7 buddy

Would you like me to draw this as a visual flowchart too? Or generate sample user data to see it working end-to-end in MongoDB Compass?

**---------------------Steps--------------------------**

**🧠 Reflective AI Math Tutor: Ultimate Build Plan**

**📁 PHASE 1: PROJECT SETUP & ENVIRONMENT**

**✅ Step 1: Create Project Folder Structure**

bash

CopyEdit

mkdir reflective-ai-math-tutor

cd reflective-ai-math-tutor

mkdir backend frontend

**✅ Step 2: Create Backend Substructure**

bash

CopyEdit

cd backend

python -m venv venv # Create virtual environment

source venv/bin/activate # Windows: venv\Scripts\activate

touch main.py db.py content\_generator.py sentiment\_analysis.py adaptive\_logic.py requirements.txt .env

**✅ Step 3: Install Backend Dependencies**

bash

CopyEdit

pip install openai pymongo python-dotenv

pip freeze > requirements.txt

📄 Your .env file (in /backend):

env

CopyEdit

OPENAI\_API\_KEY=your\_openai\_key

MONGO\_URI=mongodb+srv://user:password@cluster.mongodb.net/math\_tutor?retryWrites=true&w=majority

**🧠 PHASE 2: CORE BACKEND MODULES**

**✅ Step 4: Build db.py (MongoDB Logic)**

Includes: create user, log question, log feedback, track performance  
✅ **Already Done** – you have this file!

**✅ Step 5: Build content\_generator.py (GPT Questions)**

Function:

python

CopyEdit

generate\_math\_question(grade\_level, topic)

✅ **Already Done**

**✅ Step 6: Build main.py (Testing Script)**

Handles:

* User registration
* GPT question generation
* Manual answer input
* Feedback logging
* Performance review

✅ **Already Done**

**🔄 Step 7: Add adaptive\_logic.py**

Logic to decide difficulty:

python

CopyEdit

def adjust\_difficulty(recent\_logs):

# Check correctness and confusion to decide: increase / decrease / maintain

Use it to tweak GPT prompt.

**🧠 Step 8: Add sentiment\_analysis.py**

Interpret emoji or text:

python

CopyEdit

def interpret\_emoji(emoji): → "confident", "confused"

def interpret\_text(feedback): → "confident", "frustrated"

Use to classify reflection.

**💡 PHASE 3: FRONTEND UI (Streamlit)**

**✅ Step 9: Set Up Frontend Folder**

bash

CopyEdit

cd ../frontend

python -m venv venv

source venv/bin/activate

touch app.py requirements.txt

Install dependencies:

bash

CopyEdit

pip install streamlit

pip freeze > requirements.txt

**🧠 Step 10: Build Streamlit UI in app.py**

UI Screens:

* Name + grade input
* Start learning button
* Show GPT question
* Input answer
* Ask for reflection (text, emoji, confidence)
* Submit button → save to DB
* Show performance dashboard

Import backend logic:

python

CopyEdit

import sys, os

sys.path.append('../backend')

from db import ...

from content\_generator import ...

**📊 PHASE 4: DATA TRACKING & ADAPTIVITY**

**📈 Step 11: Implement Adaptive Difficulty in UI**

* Fetch last 5 answers
* Run through adjust\_difficulty()
* Use that to alter GPT prompt

**😊 Step 12: Save Reflection + Sentiment to DB**

* Add field for emoji and free text
* Use interpret\_emoji() or interpret\_text()
* Save to feedback and performance

**📉 Step 13: Build Performance View**

* Show total questions answered
* Correct %, last 5 emotions
* Trend graph (if using Streamlit charts)

**🐳 PHASE 5: DOCKER DEPLOYMENT (Optional)**

**🛠️ Step 14: Create Backend Dockerfile**

Dockerfile

CopyEdit

FROM python:3.11-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

CMD ["python", "main.py"]

**🛠️ Step 15: Create Frontend Dockerfile**

Dockerfile

CopyEdit

FROM python:3.11-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 8501

CMD ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]

**🛠️ Step 16: Create docker-compose.yml**

yaml

CopyEdit

version: '3.8'

services:

frontend:

build: ./frontend

ports:

- "8501:8501"

backend:

build: ./backend

environment:

- OPENAI\_API\_KEY=${OPENAI\_API\_KEY}

- MONGO\_URI=${MONGO\_URI}

**✅ PHASE 6: TESTING & FINALIZATION**

**🧪 Step 17: Manual Testing**

* Run backend and frontend
* Create multiple users
* Try questions, feedback, emotions
* Check MongoDB Compass for logs

**📄 Step 18: Create Documentation**

* What each file does
* Setup instructions
* .env example
* Screenshots

**🐙 Step 19: Push to GitHub (optional)**

bash

CopyEdit

git init

git add .

git commit -m "Initial AI Math Tutor Commit"

git branch -M main

git remote add origin <your-repo-url>

git push -u origin main

**🔚 Summary Table**

| **Phase** | **Module** | **Status** |
| --- | --- | --- |
| Backend Setup | db.py, .env, content\_generator.py | ✅ |
| Reflection Logic | sentiment\_analysis.py, adaptive\_logic.py | 🔜 |
| Frontend UI | app.py with full form + logic | 🔜 |
| Performance View | Charts, recent feedback | 🔜 |
| Deployment | Dockerfile + docker-compose | 🔜 |

Would you like me to generate the next file (adaptive\_logic.py, sentiment\_analysis.py, or Streamlit app.py) now? You're nearly there! Let’s make it amazing. 🚀