# **Learn It Easy — AI-Powered Personalized Educational Tool Documentation**

## **Project Overview**

Learn It Easy is an interactive web application designed to provide age-tailored, simple, and accurate educational explanations on a wide range of topics. It leverages lightweight AI models running locally through the Ollama API, such as Gemini ("gemma:2b") and Mistral, to deliver clear and context-appropriate responses suitable for users from young children to adults.

The app also incorporates basic local solvers for common math problems, enabling fast answers without always querying AI models.

## **Features**

* Age-based content tailoring with language and explanation complexity adapted for different age groups
* Safe prompt filtering to block inappropriate or unsafe user inputs
* Automatic prompt type detection: supports factual queries, code requests, math problems, and conceptual explanations
* Built-in local math problem solvers for percentage calculations, square area calculations, and symbolic math expressions using SymPy
* Integration with Ollama hosting local AI models (such as Gemini or Mistral) for generating detailed explanations or code
* UI built with Streamlit for easy deployment and interactive use
* Option for users to give feedback and request improved explanations
* Sidebar instructions to guide usage and inform about system requirements

## 

## **File Descriptions**

## **1. App.py**

* Main Streamlit application script.
* Handles user input for age and query.
* Conducts input filtering for conversation phrases, unsafe topics, and empty queries.
* Detects the prompt type and delegates either to local math solvers or builds different prompts for AI generation.
* Manages AI model selection and fetching content from the Ollama API via the model logic module.
* Displays responses and manages user feedback to generate improved explanations when requested.

## **2. function.py**

* Contains utility functions for filtering inputs and detecting prompt types.
* Filters out conversational "chatty" inputs not intended for the educational context.
* Blocks unsafe or banned keywords to ensure content appropriateness.
* Recognizes prompt types: code, factual, math, or concept.
* Implements local math problem solvers:
  + Percentage calculations.
  + Area of square based on side length.
  + Advanced math evaluation using SymPy.
* Provides age-based prompt templates that influence the style and complexity of AI-generated responses.

## **3. model\_logic.py**

* Manages AI model initialization and interaction with the local Ollama API endpoint.
* Contains functions to select a model and to send the prompt for content generation.
* Handles API exceptions gracefully.

## **4. UI.py**

* Handles all Streamlit UI components and layouts.
* Provides functions for setting up the main interface including age slider and query text area.
* Sidebar instructions for user guidance.
* Displays AI responses and code snippets with proper formatting.
* Manages user feedback widget for rating responses.

## **Installation & Setup**

1. Clone the repository.
2. Install Python dependencies:

Bash

pip install streamlit sympy request

1. Run Ollama server locally:
   1. Install and start Ollama AI platform on your machine.
   2. Ensure it is running on the default **localhost port 11434.**
2. Run the application:

Bash

streamlit run App.py

1. Use the sidebar to **select the AI mode**l (e.g., gemma:2b, mistral).
2. Enter your age and topic or question and get a personalized learning explanation.

## 

## 

## **Usage**

The interface features an age slider, topic input, and selectable AI model—all designed for user-friendly learning experiences.

## 

## Select your age using the slider.

## Enter your question or topic in the text area.

## The system filters the input, detects its type, and either:

## Answers directly (for math problems it can solve locally),

## Or constructs a tailored prompt and sends it to the selected AI model.

## The AI’s response is displayed clearly with explanations or code if applicable.

## Provide feedback to request a better explanation when needed.

## 

## **Safety & Content Filtering**

To maintain the educational focus and security:

* Inputs containing banned keywords related to violence, adult content, hate speech, etc., are blocked.
* Casual conversational inputs (e.g., greetings) do not trigger responses.
* Empty queries prompt user to input valid topics.
* Responses include fallbacks when the AI cannot solve or answer with certainty.

## **Extending and Customizing**

* Add more local solvers in function.py for additional math topics.
* Expand the banned keywords or conversational phrases lists as appropriate.
* Customize prompt templates in App.py to better fit different domains or age groups.
* Integrate additional AI models supported by Ollama.
* Improve UI styling or add features like saving user sessions or exporting answers.

## **Dependencies**

## Python 3.x

## Streamlit

## SymPy (for symbolic math parsing and evaluation)

## Requests (for HTTP API calls)

## Ollama AI (local AI model server)

## 

## 

## **Challenges Faced**

* Device Constraints: Many users operate on standard laptops or desktops without powerful GPUs, limiting the feasibility of large AI models.
* Model Accuracy vs. Performance: Large models tend to be more accurate but slow and resource-heavy, affecting real-time user experience.
* Response Time: Fast and interactive responses were prioritized, necessitating model and prompt optimization.

## **Rationale for Using Lightweight Models**

* Selected lightweight models like "gemma:2b" and "mistral" because they run efficiently on consumer hardware without significant performance hits.
* Although less powerful than the largest models, combined with carefully crafted prompts and built-in local solvers, they produce sufficiently accurate, age-appropriate answers.
* Ensures accessibility across diverse user devices, including resource-limited environments such as schools or public libraries.

## **Troubleshooting**

* Ensure Ollama server is running and accessible at http://localhost:11434.
* Check your internet connection for first-time setup, if models need downloading.
* If math expressions fail to evaluate, check the query format.
* Feedback "No" triggers a request for improved AI explanation.

## 