CODE GEN AI & CODE EXPLAINER

AI-Powered Assistant for Code Generation, Analysis, and Explanation

Presented by: Ch. Gayatri

INTRODUCTION

- Code Gen AI & Code Explainer is an intelligent system that helps in automating code writing and understanding.
- Built using AI models (LLMs), it can generate, debug, and explain code in natural language.
- Integrates Streamlit, Ollama, and OCR for file-based code analysis and visualization.
- Aims to make coding easier, faster, and more educational.

OBJECTIVES

- Automate code generation and code correction.
- Provide clear explanations of code logic and flow.
- Identify syntax, logic, and security errors in code.
- Support multiple document formats for versatile input handling.

CORE COMPONENTS

- Streamlit: Builds a clean, interactive chat UI.
- Ollama (LLM Engine): Runs local AI models like codellama, gemma, or llama3.
- OCR (pytesseract): Extracts text from image-based files.
- pdfplumber / python-docx: Extracts readable text from PDFs and Word files.
- Pandas: Reads and displays structured CSV data.
- JSON Storage: Stores conversation history securely.

SYSTEM ARCHITECTURE

- User Input: Chat message or file upload via Streamlit.
- Extraction Layer: OCR + document parsers extract text/code.
- AI Layer (Ollama): LLM processes and analyzes code.
- Response Engine: Generates summary, corrected code, and explanation.
- Storage Layer: Saves chat history for future context.

WORKFLOW

- User uploads a file or types a code-related query.
- App extracts and processes file text.
- AI model analyzes or generates code based on context.
- Response streamed back with explanations and fixes.
- Conversation saved for continued learning or reference.

STREAMLIT FEATURES

- Interactive chat interface with fixed bottom input bar.
- Sidebar options for model selection and chat management.
- File upload button (+) for fast access.
- Real-time streaming AI responses for natural conversation.

OCR (OPTICAL CHARACTER RECOGNITION) MODULE

- Image Files: Handled via pytesseract for OCR-based text extraction.
- PDF Files: Text extracted using pdfplumber.
- DOCX Files: Processed via python-docx.
- CSV / TXT Files: Read directly using Pandas or file I/O.
- Extracted text analyzed automatically by AI for summarization and debugging.

OLLAMA – LOCAL LLM INTEGRATION

- Runs Large Language Models (LLMs) locally (e.g., codellama, gemma, llama3)
- Works offline – no API cost or internet needed
- Supports streaming responses for real-time display in Streamlit
- Easy model switching: Code generation / Explanation / Debugging
- Command-line:

ERROR HANDLING

- Verifies availability of Ollama and essential libraries.
- Catches exceptions from unreadable or corrupted files.
- Prevents multiple LLM calls with session control.
- Ensures smooth reruns and stability during chat interaction.

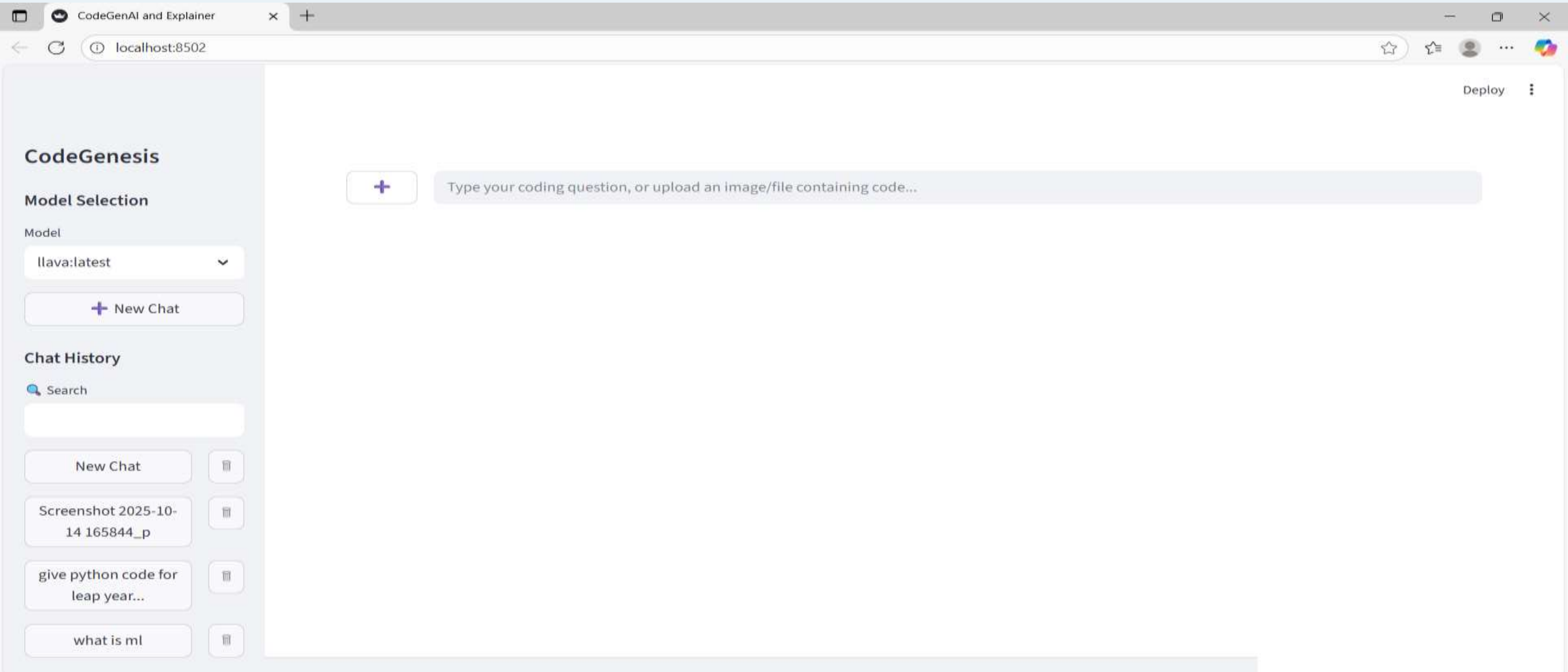
AI RESPONSE GENERATION FLOW

- Receive text / code from user or OCR
- Send prompt with system instructions to Ollama
- Model generates:
- Code summary
- Error detection / fixes
- Security & logic improvements
- Streamlit displays formatted markdown + explanation

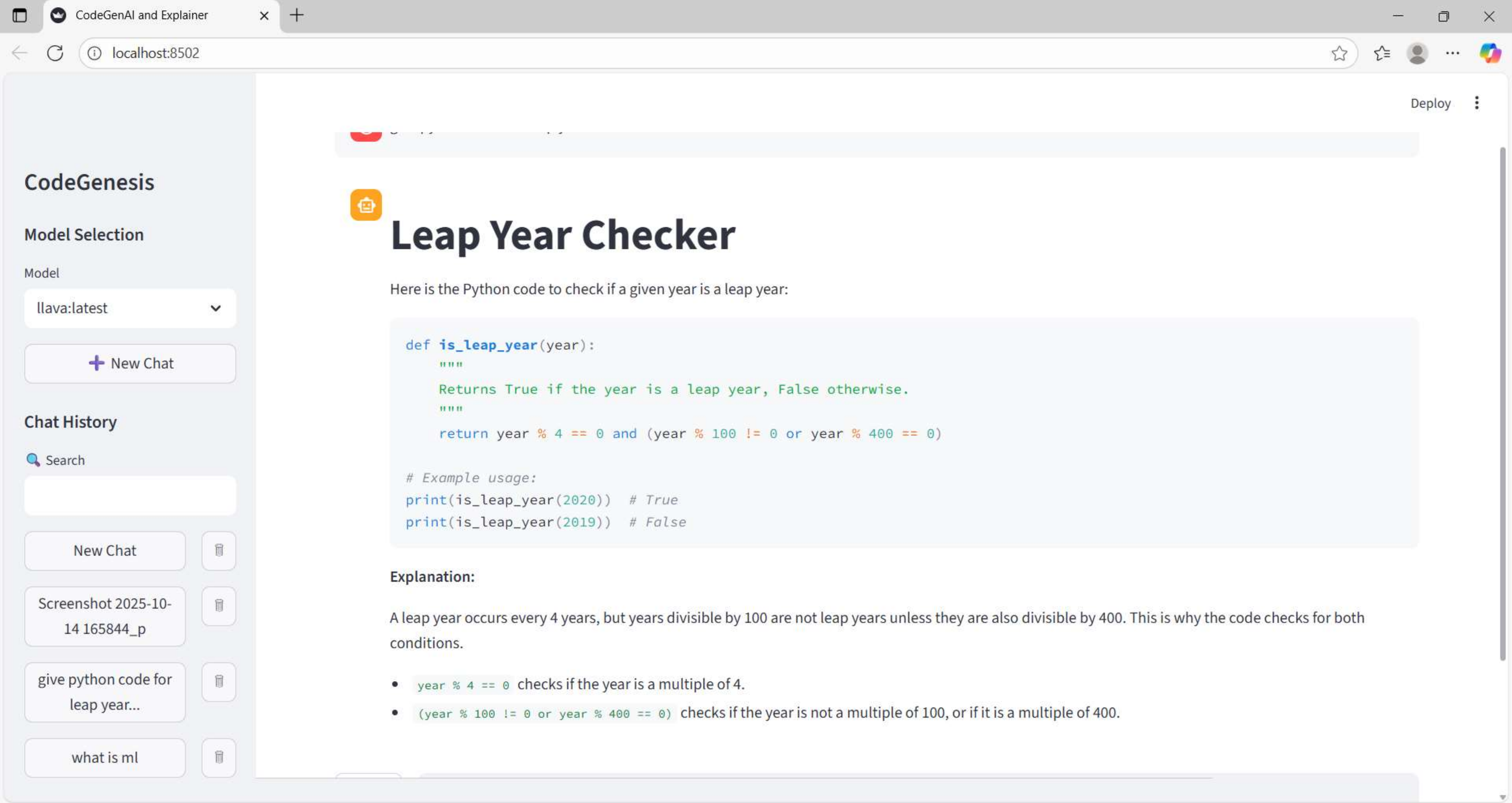
KEY FEATURES

- Multi-file support (Image, PDF, DOCX, CSV, TXT)
- Local AI model with Ollama
- OCR integration for non-text code inputs
- Chat history saved for review
- Real-time streaming AI output

DEMO FLOW



CodeGenAI and Explainer
localhost:8502
Deploy
CodeGenesis
Model Selection
Model
llava:latest
New Chat
Chat History
Search
New Chat
Screenshot 2025-10-14 165844_p
give python code for leap year...
what is ml
Type your coding question, or upload an image/file containing code...



Leap Year Checker

Here is the Python code to check if a given year is a leap year:

```
def is_leap_year(year):
    """
    Returns True if the year is a leap year, False otherwise.
    """
    return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

# Example usage:
print(is_leap_year(2020))  # True
print(is_leap_year(2019))  # False
```

Explanation:

A leap year occurs every 4 years, but years divisible by 100 are not leap years unless they are also divisible by 400. This is why the code checks for both conditions.

- `year % 4 == 0` checks if the year is a multiple of 4.
- `(year % 100 != 0 or year % 400 == 0)` checks if the year is not a multiple of 100, or if it is a multiple of 400.

ADVANTAGES

- No cloud dependency – full offline operation
- Enhanced data privacy
- Multi-language OCR capability
- Scalable architecture for future AI models
- Reduces developer debugging time significantly

FUTURE ENHANCEMENTS

- Add voice input/output
- Multi-model fusion (combine Ollama + OpenAI APIs)
- Integrate code execution sandbox
- Deploy as desktop or web SaaS app

CONCLUSION

- Code Gen AI & Code Explainer bridges human + AI coding skills
- Streamlit + Ollama + OCR = Smart Autonomous Code Assistant
- Enhances developer productivity and learning
- Future-ready AI coding platform

THANK YOU