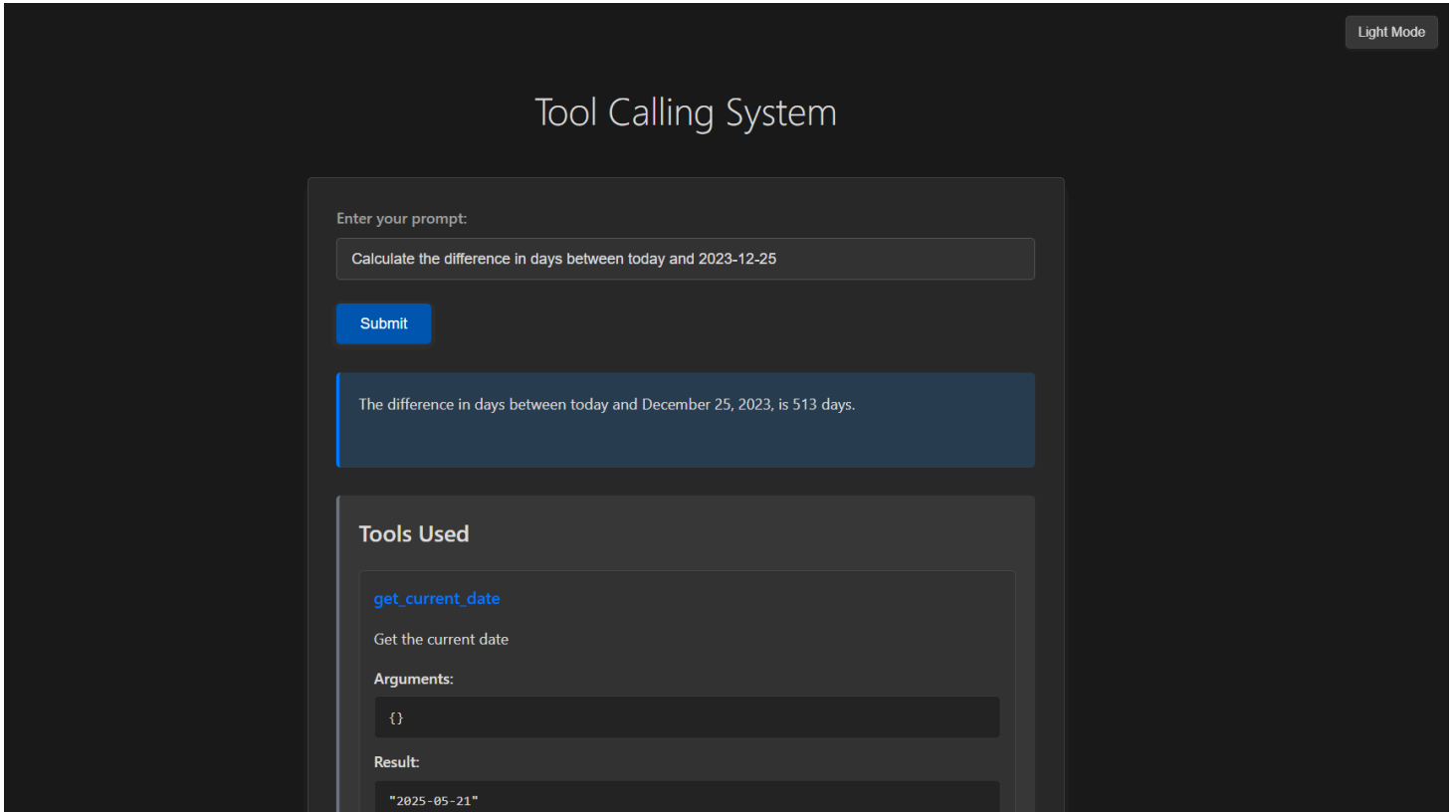


# The IBM Granite Agent: Your Smart Tool-Using LLM Companion!







## Welcome to a Capstone Adventure in AI!

This isn't just another chatbot. This project unveils an **intelligent agent** powered by the formidable **IBM Granite series Large Language Model (LLM)**, all running locally on your machine thanks to Ollama. Witness firsthand how this LLM transcends basic Q&A, acting as a sophisticated agent that understands your needs, reasons about solutions, and dynamically utilizes a suite of tools to get the job done! 🛠️

Whether it's complex date calculations, tricky math problems, or nuanced text analysis, our Granite-powered agent intelligently calls upon specialized tools, extending its capabilities far beyond its inherent knowledge. All interactions and data processing happen on your device, ensuring privacy and control. Explore the future of interactive AI through our sleek web interface (now with Dark Mode! 🌙) or our trusty command-line companion.

# The Agentic Powerhouse: Core Objective & Significance

This capstone project is all about showcasing the **agentic capabilities** of modern LLMs like IBM Granite. We're diving deep into how these models can:

-  **Understand & Decompose**: Brilliantly parse your complex requests in natural language.
-  **Reason & Plan**: Strategically determine the best tools and steps needed to fulfill your query.
-  **Act & Execute**: Dynamically call external functions (our tools!) with precisely formulated arguments.
-  **Synthesize & Respond**: Intelligently process tool outputs to generate coherent, human-like, and accurate answers.

This project demonstrates a pivotal shift in AI – from passive text generators to **proactive, problem-solving agents** that can interact with and leverage external systems. It's a glimpse into a more capable and versatile AI-powered future!

## Project Structure

```
├─ dockerfile                # Docker configuration with Ollama
├─ docker-compose.yml        # Docker Compose for easy setup
├─ .dockerignore             # Docker ignore patterns
├─ toolCalling.py             # Main CLI application
├─ web_app.py                 # Web interface
├─ config.py                  # Configuration settings
├─ requirements.txt           # Dependencies
├─ log.md                     # Log file
├─ README.md                  # Documentation
├─ static/                    # Web assets (CSS, JS)
│   └─ css/
│   └─ js/
├─ templates/                 # HTML templates
│   └─ index.html
└─ tools/                     # Tool implementations
    └─ base.py                 # Base tool class
    └─ date_tools.py           # Date calculation tools
    └─ math_tools.py           # Mathematical tools
    └─ text_tools.py           # Text processing tools
```

# 🌟 Key Features: What Makes This Agent Shine! 🌟

- 🤖 **True Agentic Workflow:** Witness the IBM Granite LLM acting as an intelligent agent, making decisions and orchestrating tool use.
- 🛠️ **Dynamic Function Calling:** See the LLM's prowess in formulating precise requests for external tools and understanding their structured (JSON) responses.
- 🧠 **Superior NLU & Synthesis:** Marvel at how the LLM interprets diverse inputs and crafts natural, human-like explanations from raw tool data.
- 🧩 **Expandable Tool Arsenal:** A modular design means new tools can be easily integrated, constantly boosting the agent's skills!
- 💻 **Dual Interaction Modes:** Choose your adventure! Engage via a polished, modern web UI (complete with a cool dark mode!) or a robust command-line interface.
- 🔍 **Transparent Insights:** The web UI peels back the curtain, showing exactly which tools were used, with what arguments, and their results – a fantastic way to see the agent's "thinking" process!
- 🔒 **Local & Private:** Powered by Ollama, the entire system, including the LLM, runs locally on your machine. This means your data stays private, and you have full control over the environment.
- 📝 **Detailed Logging:** Every step, every tool call, every result meticulously logged for easy debugging and deeper understanding.
- 🐳 **Docker Ready:** Containerized deployment option for easy setup and consistent environments across different systems.

## Tools

### Date Tools

- **DateCalculator:** Calculates dates by adding or subtracting days, months, or years from a given date.
- **DateDifference:** Calculates the difference in days between two dates.
- **GetCurrentDate:** Retrieves the current date in YYYY-MM-DD format.
- **GetCurrentTime:** Retrieves the current time in HH:MM:SS format.
- **GetCurrentDayName:** Retrieves the name of the current day (e.g., Monday).

# Math Tools

- **ExpressionEvaluator:** Evaluates mathematical expressions with support for various math functions.

# Text Tools

- **TextCounter:** Counts the number of words in a given text.
- **TextAnalyzer:** Analyzes text properties including character count, word count, sentence count, and average word length.
- **TextFormatter:** Formats text with operations like uppercase, lowercase, title case, etc.
- **SpecificLetterCounter:** Counts the occurrences of a specific letter or substring within a given text (e.g., "how many 'a's in 'banana'").

# Installation

## Option 1: Local Installation

### 1. Clone the repository:

```
git clone https://github.com/daruoktab/Intelligent-Granite-Agent.git
cd Intelligent-Granite-Agent
```

### 2. Install Ollama:

This project requires Ollama to be installed and running to serve the language model locally on your machine.

- Download Ollama from the official website: <https://ollama.com/download>
- Follow the installation instructions for your operating system (Windows, macOS, or Linux).
- Ensure the Ollama application is running before proceeding.

### 3. Pull the Required Model:

Once Ollama is running, open your terminal or command prompt and pull the IBM Granite model used by this project:

```
ollama pull granite3.3
```

(This model `granite3.3` is set as the `DEFAULT_MODEL` in `config.py`.)

### 4. Create and activate a virtual environment (recommended):

```
python -m venv venv
# On Windows
venv\Scripts\activate
# On macOS/Linux
source venv/bin/activate
```

## 5. Install Python dependencies:

```
pip install -r requirements.txt
```

## Option 2: Docker Installation 🐳 (Recommended for Easy Setup)

For the simplest setup experience, use Docker! This includes everything you need - Python, Ollama, and the Granite model.

## ⚠️ System Requirements & Performance Notes

### Model Information:

- **Granite 3.3 Model Size:** ~4.9GB download
- **RAM Requirements:** Minimum 8GB, Recommended 12GB+
- **Storage:** ~6GB free space for model and container

### Performance Considerations:

- 🚀 **With GPU:** Fast inference (~1-2 seconds per response)
- 🐢 **CPU Only:** Slower but functional (~10-30 seconds per response)
- 💡 **Note:** The application will work on CPU-only systems but responses will be significantly slower

### GPU Support (Optional but Recommended):

- NVIDIA GPUs with CUDA support provide much faster performance
- AMD GPUs are not currently supported by Ollama
- To enable GPU support, see the GPU configuration section below

## Installation Steps:

### 1. Clone the repository:

```
git clone https://github.com/daruoktab/Intelligent-Granite-Agent.git
cd Intelligent-Granite-Agent
```

## 2. One-command setup with Docker Compose (Easiest):

```
docker-compose up -d
```

```
``` This will:
```

- Build the application with all dependencies
- Install and configure Ollama
- Download the Granite 3.3 model (~4.9GB - be patient!)
- Start both services automatically

## 3. For GPU support (NVIDIA only):

- Install [NVIDIA Docker support](#)
- Uncomment the GPU lines in `docker-compose.yml`
- Run: `docker-compose up -d`

## 4. Alternative: Build and run manually:

```
# Build the image (includes Ollama installation)
```

```
docker build -t granite-llm-app .
```

```
# Run CPU-only
```

```
docker run -d -p 12000:12000 -p 11434:11434 --name granite-agent granite-llm-app
```

```
# Run with GPU support (NVIDIA)
```

```
docker run -d -p 12000:12000 -p 11434:11434 --gpus all --name granite-agent granite-llm-app
```

## 5. Access the application:

- Web Interface: `http://localhost:12000`
- Ollama API: `http://localhost:11434` (if needed)


## 6. Check status:

```
# View logs
```

```
docker-compose logs -f
```

```
# Check if services are running
```

```
docker-compose ps
```

 **First Run Notice:** The initial startup takes 5-10 minutes to download the model (~4.9GB). Subsequent starts are much faster (under 30 seconds).

# Usage

## Command Line Interface

### 1. Run the main script:

```
python toolCalling.py
```

### 2. Enter your prompt (examples):

- **Date calculations:**
  - "What date will it be 30 days from today?"
  - "What date was 2 months ago?"
  - "How many days between January 1, 2024 and March 15, 2024?"
  - "What is the current date and time?"
  - "What day of the week is it?"
- **Math calculations:**
  - "What is 2 + 2?"
  - "Calculate 15% of 200"
  - "What is the square root of 144?"
  - "Solve: (5 + 3) \* 2 - 4"
- **Text processing:**
  - "Count the words in this text: 'The quick brown fox jumps over the lazy dog'"
  - "Analyze this paragraph for character and word statistics"
  - "Convert 'hello world' to uppercase"
  - "How many times does 'the' appear in 'the quick brown fox jumps over the lazy dog'?"

### 3. View the results:

The system will determine the appropriate tool to use, display the result, and show which tools were used.

## Web Interface

### 1. Start the web server:

```
python web_app.py
```

### 2. Open your browser:

Navigate to `http://localhost:12000` (or the port specified in your environment)

### 3. Interactive Features:

- Modern, responsive web interface with dark mode support 🌙

- Real-time tool usage visualization
- Chat-like interaction with the AI agent
- Detailed breakdowns showing which tools were called and their results

## Docker Usage

If you're using the Docker version (recommended):

### 1. Quick start:

```
docker-compose up -d
```

### 2. Access the application:

Navigate to `http://localhost:12000`

### 3. Monitor and manage:

```
# View real-time logs
docker-compose logs -f
```

```
# Stop the services
docker-compose down
```

```
# Restart services
docker-compose restart
```

```
# Check service status
docker-compose ps
```

### 4. First-time setup note:

The first run will take 5-10 minutes as it downloads the Granite 3.3 model (~4.9GB). Subsequent starts will be much faster.

### 5. Performance expectations:

- **With GPU:** Responses in 1-3 seconds
- **CPU only:** Responses in 10-30 seconds (still fully functional!)
- Monitor resource usage: `docker stats`

## Requirements

- Python 3.8+
- Ollama (for local LLM hosting)



- Docker (optional, for containerized deployment)

## System Requirements for Optimal Performance

### Minimum Requirements (CPU-only):

- 6GB RAM (12GB recommended)
- 6GB free disk space
- Any modern CPU (will be slow but functional)

### Recommended Setup (GPU-accelerated):

- 12GB+ RAM
- NVIDIA GPU with 6GB+ VRAM
- NVIDIA drivers and Docker GPU support
- 6GB free disk space

### Model Details:

- **Granite 3.3:** ~4.9GB model size
- **Performance:** GPU provides 10-20x faster inference than CPU
- **Compatibility:** Works on both ARM and x86 architectures

## Python Dependencies

- `ollama` - Interface with Ollama for LLM interactions
- `flask` - Web framework for the web interface
- `pydantic` - Data validation and settings management

## Configuration

Edit `config.py` to customize:

- **Default LLM model:** Change the `DEFAULT_MODEL` variable
- **Logging settings:** Modify log file path and level
- **Enabled tools:** Add or remove tools from the `ENABLED_TOOLS` list
- **Port settings:** Adjust the default port for the web interface

Example configuration:

```
# Model configuration
DEFAULT_MODEL = "granite3.3" # or "llama2", "codellama", etc.

# Logging configuration
LOG_FILE = "log.md"
LOG_LEVEL = "DEBUG"

# Tool configuration
ENABLED_TOOLS = [
    "DateCalculator",
    "MathEvaluator",
    "TextAnalyzer"
]
```

## Logging

Logs are written to `log.md` in a markdown-friendly format. This includes:

- Tool function calls with arguments
- Execution results and outputs
- Error messages and debugging information
- Timestamps for all interactions

The logging system helps you understand exactly how the AI agent processes requests and interacts with tools.

## GPU Configuration (Optional but Recommended)

### Why Use GPU?

- **10-20x faster inference** compared to CPU
- Responses in 1-3 seconds instead of 10-30 seconds
- Better user experience for interactive use

### NVIDIA GPU Setup

1. **Install NVIDIA Docker support:**

```
# For Ubuntu/Debian
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo t
sudo apt-get update && sudo apt-get install -y nvidia-docker2
sudo systemctl restart docker
```

```
# For Windows with WSL2
# Install NVIDIA drivers for WSL2 from NVIDIA website
```

## 2. Enable GPU in docker-compose.yml:

Uncomment the GPU configuration lines in the file

## 3. Verify GPU access:

```
docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

# CPU-Only Setup (Default)

- No additional configuration needed
- Works out of the box on any system
- Slower but fully functional
- Perfect for testing and development

# Troubleshooting

## Common Issues

### 1. Ollama Connection Error:

- Ensure Ollama is installed and running
- Verify the model is pulled: `ollama pull granite3.3`
- Check if Ollama is accessible on the default port (11434)

### 2. Port Already in Use:

- Change the port in `web_app.py` or set the `PORT` environment variable
- Kill existing processes using the port

### 3. Docker Issues:

- Ensure Docker has sufficient resources (6GB+ RAM recommended)
- First startup takes time to download the Granite model (~4.9GB)
- For slow responses, consider GPU acceleration if available
- Check both services: `docker-compose ps`

- View detailed logs: `docker-compose logs -f`
- If model download fails, restart: `docker-compose restart`
- Monitor resource usage: `docker stats`

#### 4. Performance Issues:

- **Slow responses on CPU:** This is normal - consider GPU setup for faster inference
- **Out of memory:** Increase Docker memory allocation or reduce system load (model needs ~6GB)
- **Model download stuck:** Check internet connection and disk space (~4.9GB needed)
- **High CPU usage:** Expected during inference on CPU-only systems

#### 5. Model Not Found:

- Pull the required model: `ollama pull granite3.3`
- Update `config.py` with a different model if needed

## Future Enhancements

- **Tool Expansion:** Add more tools for diverse functionalities (e.g., web scraping, file manipulation).
- **Image Support:** Integrate image processing tools for OCR, image analysis, etc.
- **Multi-Turn Conversations:** Enhance the agent's ability to maintain context over multiple interactions.

## Capstone Project Significance

This project serves as a practical demonstration of leveraging advanced LLM capabilities for building intelligent, tool-augmented applications. The use of IBM Granite for sophisticated tool orchestration and natural language interaction highlights the potential for creating more powerful and versatile AI systems.