SUBBURI DHEERAJ VARMA

India - Hyderabad | +91 7981692357 | [dheerajvarma031@gmail.com](mailto:dheerajvarma031@gmail.com) | [LinkedIn](https://www.linkedin.com/in/dheeraj-varma-5061342b1/)

**Assignment Report:** Educational Content Generator

**Introduction:**

The **Educational Content Generator** is a web application designed to generate, refine, and store educational content based on user inputs such as subject, grade level, and topic. The application leverages natural language processing (NLP) to generate detailed educational lessons on various subjects, automatically refine content for readability and inclusivity, and detect biases in the text. This project allows users to interact with the system via a simple user interface, enabling easy generation of content tailored to educational needs.

The application is designed as a modular and scalable platform, allowing future enhancements such as integrating additional AI models or refining the content generation pipeline.

**System Architecture:**

The architecture of the Educational Content Generator is divided into distinct components to address the various requirements of content generation, refinement, bias detection, and system scalability.

**Front-End:**

The front-end is designed using Flask, HTML and CSS to provide a clean and simple user interface. It consists of:

* A form to input the **subject**, **grade level**, and **topic**.
* A display area for the generated content, showing both raw and refined versions.
* A left panel displaying previously generated topics.

**Back-End:**

The back-end is powered by **Flask**, a lightweight Python web framework that handles HTTP requests and manages user interactions. Key components include:

* **Ollama API**: The pre-trained transformer-based language model used to generate educational content from the provided user prompts. Using llama-3.1.8b
* **TextBlob** and **NLTK**: These NLP libraries are used for refining content (e.g., readability and inclusivity) and detecting potential biases.

**Database:**

The system uses **Flask Sessions** to store the history of generated topics. This allows users to access previously generated content during their session without needing a database.

**Features:**

* **Content Generation-**

The system generates educational content based on user input. The user provides a subject, grade level, and topic, and the system generates coherent, contextually relevant content aligned with general curriculum standards. The system uses a transformer-based language model (Ollama API) to process the user prompt and produce raw content.

* **Content Refinement-**

After content generation, the system refines the generated content:

* **Readability**: The Flesch Reading Ease score is calculated to ensure the text is understandable. If the score is low, the content is adjusted for clarity and simplicity.
* **Inclusivity**: The content is checked for exclusive terms and replaced with more inclusive alternatives using predefined mappings (e.g., "he" -> "they").
* **Content History-**

The system stores previously generated topics in a session. The left panel of the web interface shows the list of topics that have been generated during the session, enabling users to refer back to past content.

**Implementation Details:**

* **User Input**: The user provides input (subject, grade level, topic) through a form.
* **Content Generation**: A prompt is created using the user input, and the Ollama API is called to generate raw content.
* **Content Refinement**: The generated content is refined for readability and inclusivity. This is done using TextBlob and NLTK.
* **Bias Detection**: The content is scanned for biased terms, and if any are found, they are flagged.
* **Session Storage**: The topic of the generated content is stored in the session for later reference.
* **Display**: The raw and refined content, along with any detected biases, are displayed to the user.

**Tools and Techniques Used:**

* **Flask**: For building the web application and managing sessions.
* **Ollama API**: For generating educational content.
* **TextBlob**: For refining readability and detecting basic linguistic issues.
* **NLTK**: For tokenization and text analysis, including readability score calculation and bias detection.
* **HTML/CSS**: For creating the user interface and styling.

**Deployment:**

The application is deployed using **Flask**'s built-in server for local testing. For deployment this could be done in a cloud platform such as **AWS** or **GCP.**

**Deployment Instructions for Windows:**

Ensure that you have **Python** installed, and then install the required dependencies for the Flask app:

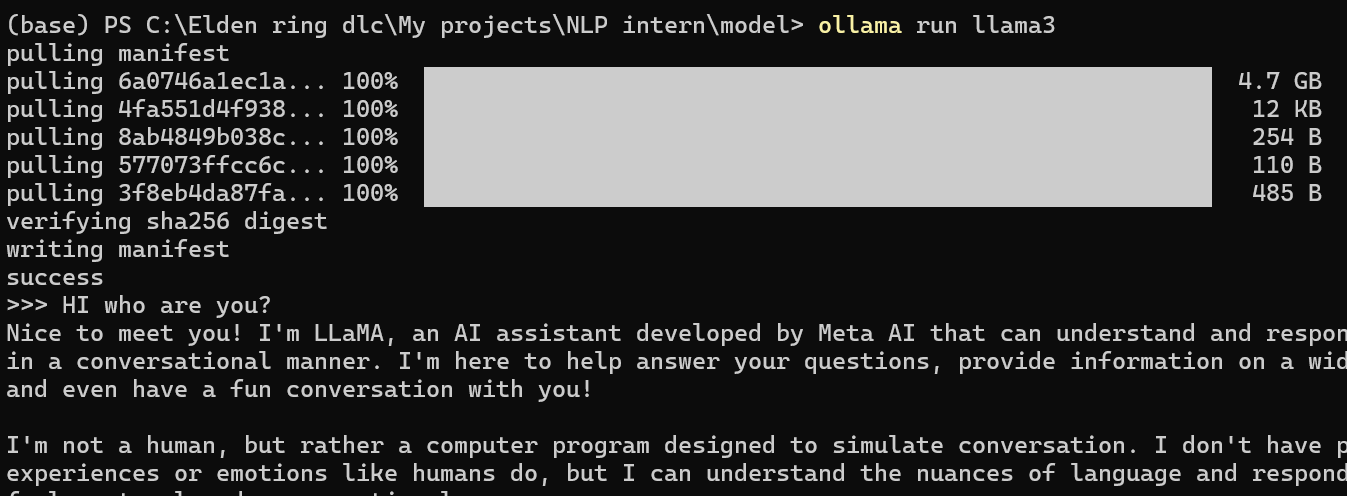
1. Open a terminal (Command Prompt or PowerShell).
2. Install the necessary Python packages:

( pip install Flask requests textblob nltk )

**Run Ollama Locally**

1. Open a terminal window and run the following command to start Ollama with the llama3 model:

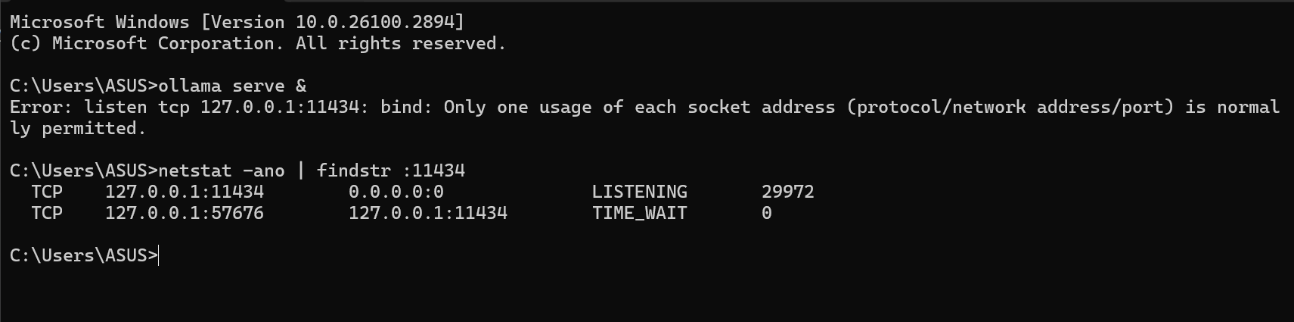
( ollama run llama3 )



This will start the installation process for the llama3 model.

1. Once the installation is complete, open a **new terminal window** and run the following command to start the Ollama server to get the PID:

( ollama serve & )



**( Save the PID** number for later use. )

1. **Update OLLAMA\_API\_URL in the Code**

Now that the server is running, you need to update the OLLAMA\_API\_URL in your Flask code with the correct **PID** (port).

In your app.py file, replace the OLLAMA\_API\_URL with:

( OLLAMA\_API\_URL = http://localhost:PID\_here/api/generate )

**Run the Flask Application**

1. Ensure the **Ollama server** is running.
2. Run the Flask application by executing the following in the terminal:

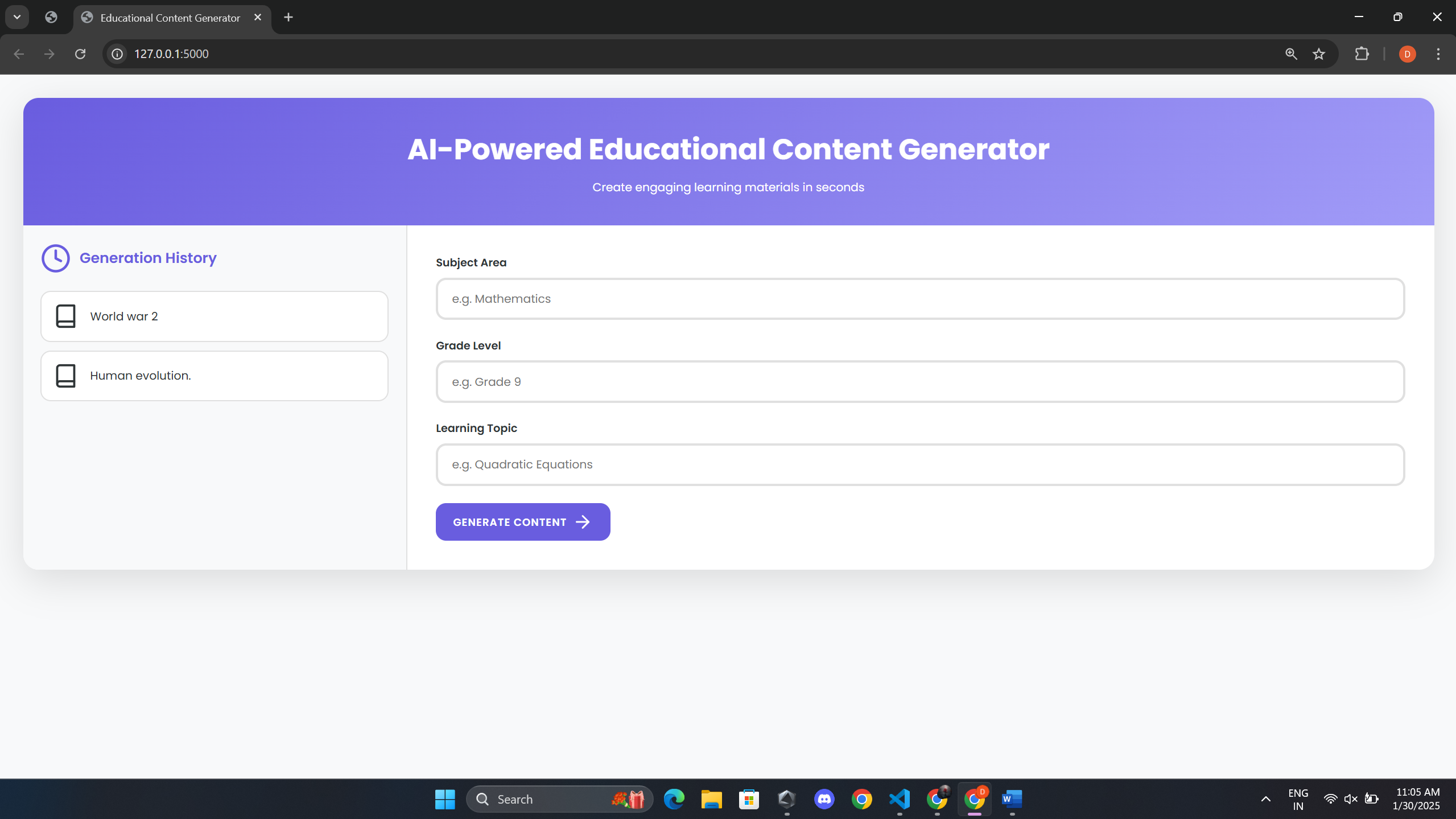
python app.py

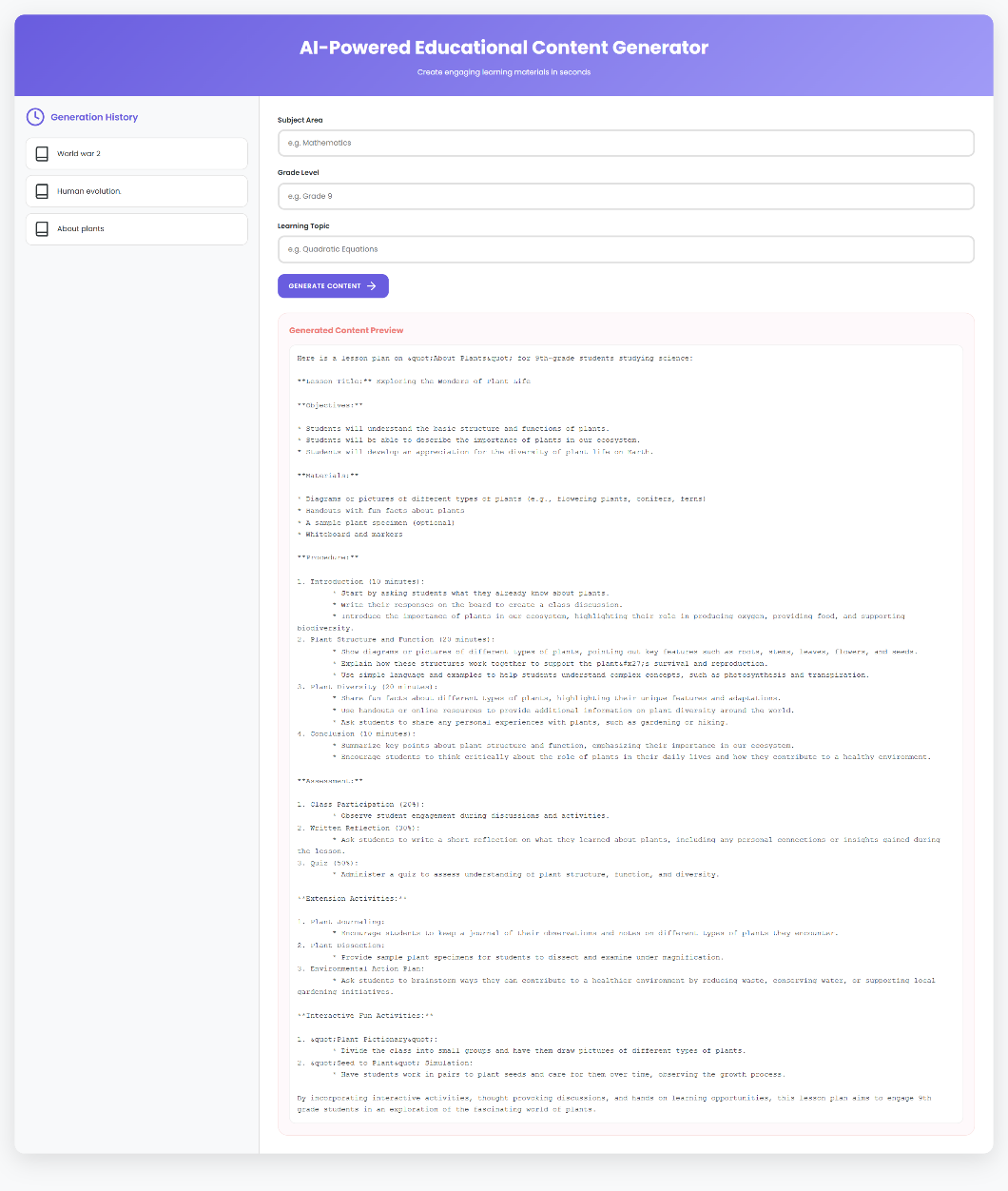
The Flask app will be running locally, and you can access it in a web browser.

Example: http://127.0.0.1:5000/

This setup ensures that your Flask application interacts with the locally hosted **Ollama API** to generate educational content.

Now you are all set to use the application.

**Application Preview:** 



**Conclusion:**

The Educational Content Generator serves as a powerful tool designed to assist instructors in guiding their lectures with AI-driven content generation and refinement. Using Ollama's transformer-based model, the system provides instructors with the following capabilities:

* Generating coherent and contextually relevant content.
* Refining content to meet readability and inclusivity standards.
* Incorporating mechanisms to detect and mitigate biases.
* Structuring the system in a modular and scalable way for future enhancements.

This prototype can be extended to integrate more advanced models like GPT-3.5 or GPT-4, Cloud servers like AWS and additional NLP tasks. It offers a powerful tool for educators and provides a foundation for further development in content generation and refinement.