Project Title	**GUVI Multilingual GPT Chatbot using Streamlit – Integrated Translation and Domain-Specific Model Deployment**
Skills take away From This Project	**Deep Learning, Natural Language Processing, Hugging face models,LLM, Data preprocessing and cleaning, Transformer model fine-tuning, Frontend design using Streamlit, API integration, Deployment, Real-time chatbot development, Version control using Git**
Domain	**Artificial Intelligence / Natural Language Processing / Web Development**

Problem Statement:

Build a **multilingual chatbot** that uses a pre-trained or fine-tuned GPT model to answer user queries. If the user's input is in a non-English language, the chatbot should translate the input to English using some **translator model**, get a response from the **GPT model**, and then translate the response back to the original language.

The chatbot should be deployed on **Hugging Face Spaces/ AWS** with a clean and interactive UI, enabling real-time conversations to assist learners and enhance their experience on the GUVI platform.

Business Use Cases:

1. **Customer Support Automation**
    - **Scenario:** Organizations receive customer queries in various regional languages, but most automated systems only understand English.
    - **Application:** The chatbot can automatically translate incoming queries to English, process them using the fine-tuned model, and respond in the user's language, enabling 24/7 multilingual support without hiring additional language experts.
2. **E-Learning Platforms**
    - **Scenario:** Educational platforms like GUVI serve a multilingual audience, but automated chat support is often limited to English, creating a barrier for non-English-speaking users.
    - **Application:** The multilingual chatbot can interact with students in their native languages, assisting with course inquiries, enrollment help, and technical support, thereby enhancing accessibility and user satisfaction
3. **Career Guidance and Mentorship Assistant**
    - **Scenario**: Leverage the GPT model to offer career suggestions and roadmaps to learners based on their interests and skill levels.
    - **Application**: The chatbot can suggest learning paths (e.g., Full Stack, Data Science), connect learners to relevant resources, and simulate basic career counseling.
4. **Course Discovery and Recommendation**
    - **Scenario**: Assist users in finding the right course based on their interests, goals, or current skill level.
    - **Application**: The GPT model can ask a few questions to the user and recommend specific GUVI courses, certifications, or masterclasses.

Approach:

1. **Model Selection**

   Select suitable model from Hugging Face, Ensure proper language-to-model routing logic for translation between different language pairs and Generation. Ensure that the Hugging Face platform has a model that handles multiple languages.

2. **Translation Logic**

   Based on selected source and target language, dynamically load or call the appropriate translation model. Ensure that the translation is triggered efficiently on user input.

3. **Streamlit UI/UX Design**

   Create a responsive layout with clear sections for input text, language selectors, and translated output. Optionally add features like copy button, clear text, and theme toggles.

4. **Error Handling & Feedback**

   Handle model loading failures, unsupported translations, or long text inputs gracefully. Provide appropriate user feedback using Streamlit alerts or text boxes.

5. **Model Deployment**

   Deploy the Streamlit app on **Streamlit Cloud**, Hugging Face Spaces, or other platforms. Ensure fast startup and proper handling of API/model latency.

6. **Version Control and Documentation**

   Maintain code in a GitHub repository with proper versioning, commit messages, and a README file outlining setup, usage, and language support.

Steps to Fine-Tune the Model

1. **Data Collection or Extraction**:
   - Gather text data from the Hugging Face Datasets library or Guvi resources for fine-tuning the model.
2. **Data Preparation**:
   - Clean and preprocess the text data, ensuring it is in a format suitable for training (e.g., removing special characters, normalizing text).
3. **Tokenization**:
   - Use model-specific tokenizers (like `AutoTokenizer` from Hugging Face) to split text into tokens that the model understands.
4. **Fine-Tuning**:
   - Use the Hugging Face Transformers library or similar tools to fine-tune the model on the prepared dataset.

Results:

1. **Functional Web Application**: A fully functional web interface built with Streamlit where users can:
   - Input text in any supported language
   - Automatically detect language and translate input to English.
   - Get responses from a fine-tuned **GUVI GPT** model
   - Receive replies translated back into the user's original language/
2. **Scalable Deployment**: A scalable deployment framework using Hugging Face/AWS services.
3. **Documentation**: Comprehensive documentation outlining setup, deployment, and usage instructions.

Project Evaluation metrics:

- You are supposed to write a code in a modular fashion (**in functional blocks**)
- Maintainable: It can be maintained, even as your codebase grows.
- Portable: It works the same in every environment (operating system)
- You have to maintain your code on **GitHub**.(Mandatory)
- You have to keep your **GitHub** repo public so that anyone can check your code.(Mandatory)
- Proper readme file you have to maintain for any project development(Mandatory)
- You should include basic workflow and execution of the entire project in the readme file on **GitHub** (Mandatory)

- Follow the coding standards: https://www.python.org/dev/peps/pep-0008/
- You need to Create a Demo video of your working model and post in **LinkedIn(Mandatory)**

Data Set:

- Gather Relevant text data from the Hugging Face Datasets library to fine-tune the language translator model.
- Gather text data from various sources within GUVI, such as website content, user queries,social media, blog posts, and training materials.

Guidelines

- **Small Scale (Tens of Thousands of Tokens)**:
  - Fine-tuning with around 10,000 to 50,000 tokens can be sufficient for simple tasks or when only minor adjustments to the model's behavior are needed.
  - Suitable for highly specialized tasks with very specific types of content.
- **Medium Scale (Hundreds of Thousands of Tokens)**:
  - Fine-tuning with around 100,000 to 500,000 tokens can provide more substantial adjustments and is often sufficient for many practical applications.
  - Useful for moderately complex content where the model needs to learn specific patterns and vocabulary.
- **Large Scale (Millions of Tokens)**:
  - Fine-tuning with 1 million or more tokens can yield significant improvements, especially for complex or diverse content.
  - Necessary for highly complex tasks or when the model needs to generate very accurate and contextually rich text

Project Deliverables:

1.**Data Files**:

- Data files used in the project - app.py, requirements.txt
- If data cannot be shared, provide information on how to access the data you have used.

2. **Source Code**:

- All scripts and code files used in the project.

- A file or Jupyter Notebook that shows your project in action.
- Ensure the code is well-organized and properly commented for clarity and maintainability.

3. **Documentation - README File:**

- The name of your project and what it's about.
- How to set up everything and run your code.
- A quick overview of how your project is organized.
- Any tools or libraries needed to run your code.

Project Guidelines:

1. Coding Standards

- **Readability**: Write clean and readable code. Use meaningful variable and function names.
- **Comments**: Add comments to explain complex logic or important sections of your code.
- **Consistency**: Stick to a consistent coding style. Use tools like linters (e.g., PEP8 for Python) to maintain consistency.
- **Modularity**: Break your code into functions and modules to make it reusable and easier to understand.
- **Error Handling**: Implement proper error handling to make your code robust and user-friendly.
- **Documentation**: Document your functions and classes using docstrings. Include information about inputs, outputs, and any exceptions.

2. Version Control

- **Use Git**: Use Git for version control to keep track of changes and collaborate with others.
- **Regular Commits**: Commit your code regularly with meaningful commit messages. This helps track progress and rollback changes if needed.
- **Branching**: Use branches for new features or bug fixes. This keeps the main branch stable.
- **Pull Requests**: Use pull requests for code reviews before merging branches. This ensures code quality and catches potential issues early.

- **.gitignore**: Use a .gitignore file to exclude unnecessary files from the repository (e.g., compiled files, temporary files, environment-specific configurations).

Timeline:

14 days from the date of document issuance.

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.
Note: Book the slot at least before 12:00 Pm on the same day

Timing: Monday-Saturday (4:00PM to 5:00PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

For DE/BADM project/class topic doubt slot clarification session:

Booking link : https://forms.gle/NtkQ4UV9cBV7Ac3C8

Session timing:

For DE: 04:00 pm to 5:00 pm every saturday
For BADM 05:00 to 07:00 pm every saturday

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.
Note: This form will Open only on Saturday (after 2 PM) and Sunday on Every Week

Timing:

For BADM and DE
Monday-Saturday (11:30AM to 1:00PM)

For DS and AIML
Monday-Saturday (05:30PM to 07:00PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurRA