**ML-Based Bug Detection and Fixing System**

**Team Members** : Harsh Battalwar [Leader],Diksha Waghmare , Nagesh Kharat.

**Team Guide** : Mr. Shivprasad Titare.

**Project Title:**

Build a machine learning model that can automatically identify bugs (or potential errors) in a given piece of code suggest fixes. This project requires designing, training and evaluating an ML model that can parse source code. classify bug types and generate fix.

**Project Overview**

**Objective:**

To develop a Machine Learning-based system that can automatically identify bugs in source code, classify them into specific categories, and suggest potential fixes, enhancing the efficiency and accuracy of the debugging process.

**Step 1: Define Project Objectives and Requirements**

**Objectives:**

1. **Automated Bug Detection:** Identify errors in source code.
2. **Bug Classification:** Categorize detected bugs into specific types.
3. **Fix Generation:** Propose corrections for identified bugs.
4. **Efficient Debugging:** Minimize manual debugging efforts.
5. **Scalability:** Support multiple programming languages.

**Requirements:**

* **Data:** Datasets containing buggy code snippets and their corrected versions.
* **Tools:** Python, PyTorch, Streamlit, FastAPI, Transformers.
* **Infrastructure:** GPU for model training, IDE (PyCharm).

**Step 2: Setup Development Environment**

1. **Install Required Tools & Libraries:**
   * **Python (3.x):** Core programming language.
   * **Libraries:**

pip install streamlit torch transformers fastapi requests accelerate

* + **IDE:** PyCharm.
  + **Version Control:** GitHub.

1. **Configure Environment:**
   * Set up a virtual environment:

python -m venv bug-fix-env

source bug-fix-env/bin/activate # (Linux/Mac)

ug-fix-env\Scripts\activate # (Windows)

* + Enable GPU acceleration.

1. **Dataset Preparation:**
   * Collect datasets of buggy and fixed code from open-source repositories (e.g., GitHub, Bugzilla).
   * Preprocess data using tokenization and Abstract Syntax Trees (AST).

**Step 3: Utilize a Pretrained Model for Bug Detection and Fixing**

1. **Model Overview:** CodeLlama, a large language model optimized for code understanding and generation.
2. **Installation:** Requires libraries such as **transformers**, **torch**, and **accelerate**.
3. **Loading the Model:** Use Hugging Face’s **AutoModelForCausalLM** and **AutoTokenizer** to load CodeLlama.
4. **Bug Detection & Fixing:** The model processes buggy code snippets and generates corrected versions.
5. **Supported Languages:** Capable of handling multiple programming languages.
6. **Fine-Tuning:** Customize the model using additional labeled datasets for improved performance.
7. **Advantages:** Offers state-of-the-art bug fixing, reduces manual debugging efforts, and enhances coding efficiency.

**Step 4: Develop AI Models**

1. **Model Selection:**
   * Utilize CodeLlama, a pretrained LLM optimized for code understanding and generation.
   * Fine-tune the model for enhanced bug detection and fixing capabilities.
2. **Model Components:**
   * **Bug Detection:** Identify errors in code using the pretrained model.
   * **Bug Classification:** Categorize errors (e.g., syntax, logic, security).
   * **Fix Generation:** Suggest corrections for identified bugs.
3. **Training Strategy:**
   * **Fine-Tuning (Optional):** Train on labeled datasets of buggy and corrected code.
   * **Transfer Learning:** Leverage the existing knowledge of CodeLlama.
4. **Evaluation Metrics:**
   * **Bug Detection:** Accuracy, Precision, Recall, F1-score.
   * **Fix Quality:** BLEU Score, ROUGE Score.
5. **Implementation:**
   * Integrate the model into an interactive debugging system.
   * Ensure scalability for various programming languages.

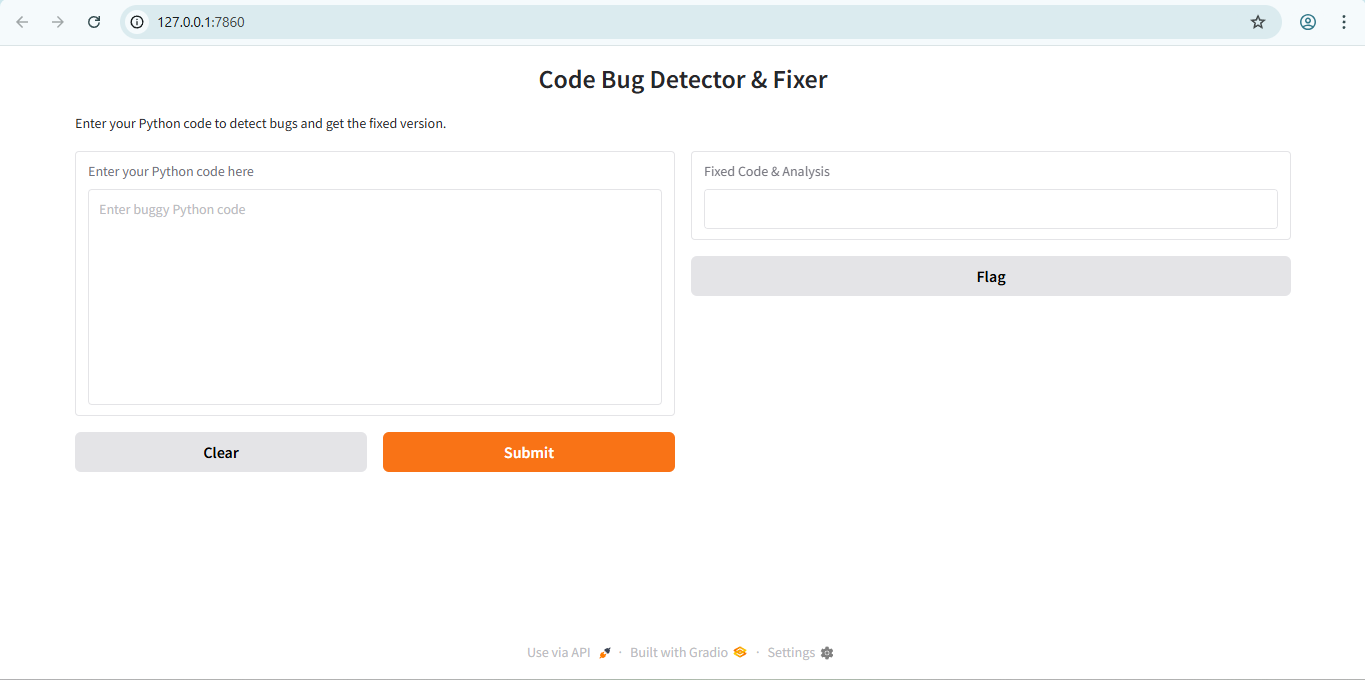
**Summary of Progress**

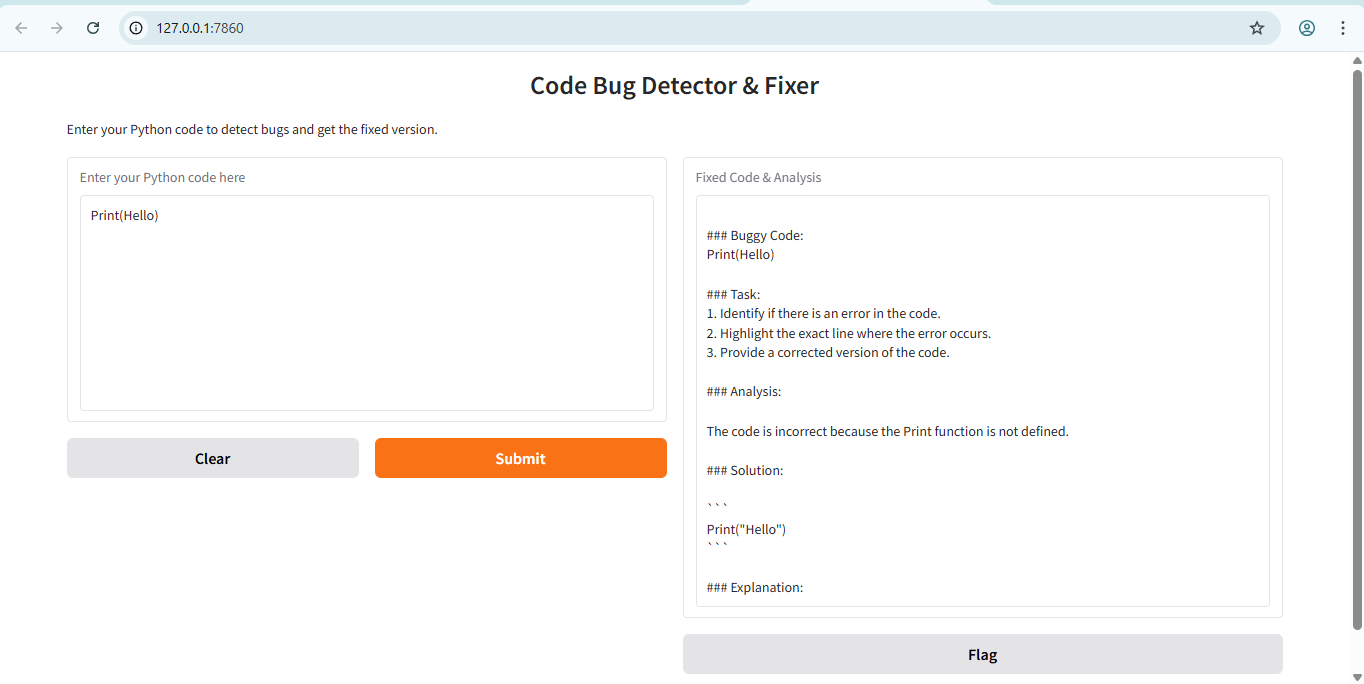
* **Step 1:** Defined Objectives & Requirements.
* **Step 2:** Set Up Development Environment.
* **Step 3:** Utilized a Pretrained Model (CodeLlama).
* **Step 4:** Developed AI Models.

**Next Steps:**

1. **Fine-Tuning the Model:** Train CodeLlama on buggy and corrected code to enhance accuracy and adaptability.
2. **Model Evaluation:** Assess performance using metrics such as Accuracy, F1-score, BLEU, and ROUGE to refine predictions.
3. **Integration & Deployment:** Develop an API, IDE plugin, or web tool for real-time bug detection and fixing.
4. **Performance Optimization:** Improve speed using GPU acceleration and efficient memory management.
5. **User Testing & Feedback:** Collect insights from real-world usage to enhance accuracy and usability.

**Result :**





**Conclusion:**

In this project, we have developed a machine learning-based system capable of identifying bugs or potential errors in source code and providing intelligent fix suggestions. By leveraging a dataset of real-world code examples and applying techniques in natural language processing and deep learning, we built a model that can:

1. **Parse and analyze source code** across various programming languages.
2. **Detect and classify bugs** into predefined categories such as syntax errors, logical bugs, and runtime issues.
3. **Recommend appropriate fixes** based on contextual understanding and learned patterns from the training data.

Our model was trained and evaluated using data derived from the [BigCode The Stack dataset], and we implemented preprocessing, feature extraction (such as tokenization and AST parsing), model training, and fix generation modules. The system demonstrated promising results in terms of accuracy and relevance of its recommendations, highlighting the potential of AI in automating code review and debugging processes.

This project serves as a strong foundation for further improvements, such as:

* Enhancing the fix generation capability using transformer-based models (e.g., CodeLemma).
* Expanding support for multiple languages.
* Integrating the model into developer tools and IDEs for real-time bug detection.

Ultimately, this system can greatly assist developers by improving code quality, reducing debugging time, and streamlining software development workflows.