**URL REDIRECTION TRACING & ANALYSIS**

**Project Overview**

This project addresses the challenge of tracing redirected URLs and analyzing their final destinations to detect potential piracy content. By leveraging AI models, this solution offers a robust approach for identifying and flagging suspicious web pages. The project integrates a Python backend with a React-based frontend to provide an interactive and user-friendly experience.

**Problem Statement**

When URLs redirect to multiple destinations, it becomes challenging to analyze the final content for potential piracy or suspicious activity. This project aims to:

1. Trace the redirection path of a given URL.
2. Analyze the final destination’s content using AI models to detect piracy-related keywords or patterns.
3. Provide a comprehensive and user-friendly platform for this purpose.

**Approach**

The solution is implemented in two major parts: the backend (for URL tracing and analysis) and the frontend (for user interaction).

**Key Steps:**

1. **URL Redirection Tracing**:
   * Use the requests library to follow HTTP redirects and fetch the final URL and its content.
   * Example: Given a shortened URL (e.g., bit.ly/xyz), the script traces it to the actual destination (e.g., https://example.com).
2. **Content Parsing**:
   * Extract the HTML content of the final page using BeautifulSoup.
   * Identify key metadata (title, description, copyright information) and outbound links.
3. **AI-Based Analysis**:
   * Use a pre-trained machine learning model to analyze the extracted content for piracy indicators.
   * Example: A webpage with keywords like “download free movies” or “pirated software” is flagged as suspicious.
4. **Frontend Integration**:
   * Provide an input field for users to enter URLs.
   * Display results, including the traced URL, metadata, and piracy analysis.

**Backend Implementation**

**1. URL Tracing with requests**

* + Handles HTTP requests and automatically follows redirects, simplifying URL tracing.
  + Makes a GET request to the input URL.
  + Follows any redirects until the final destination is reached.
  + Captures the final URL and status code.

**2. HTML Parsing with BeautifulSoup**

* + Parses and extracts meaningful information from HTML content.
  + Finds metadata (e.g., title, description) and outbound links.
  + Example: From a webpage, the parser identifies the title as “Free Movie Downloads” and description as “Download latest movies for free.”

**3. AI Model Integration with scikit-learn and joblib**

* + Detects piracy-related content using machine learning.
  + joblib is used to load serialized models for efficient deployment.
  + Combines textual data (title, description) with additional features (e.g., number of links, presence of keywords like “free”).
  + Uses a classification model (e.g., Logistic Regression) to predict the likelihood of piracy.

**4. Additional Features:**

* **Extracted Metadata:**
  + Title, description, copyright info, and outbound links are analyzed for context.
* **Custom Features for AI Model:**
  + Length of the combined title and description.
  + Count of outbound links.
  + Presence of specific keywords (“free”, “download”, etc.).

**Frontend Implementation**

**1. User Interface with Next.js**

* + Offers server-side rendering and seamless API integration.
  + Provides an input field for users to enter URLs.
  + Submits the URL to the backend and displays results dynamically.

**2. API Integration with Axios**

* + Simplifies HTTP requests and error handling in JavaScript.
  + Sends the URL to the backend’s /trace endpoint.
  + Handles responses and displays trace and analysis results.

**3. Styling with Tailwind CSS**

* + Provides utility-first classes for responsive and customizable design.
  + Styles input fields, buttons, and result cards.
  + Example: The “Trace URL” button changes color on hover, enhancing user experience.

**Tools, Libraries, and Resources**

**Backend:**

1. **Flask:**
   * Framework for building REST APIs.
   * Enables cross-origin requests using flask\_cors.
2. **requests:**
   * Handles HTTP requests and follows redirects.
3. **BeautifulSoup:**
   * Parses HTML content and extracts metadata and links.
4. **scikit-learn:**
   * Provides machine learning tools for training and deploying AI models.
5. **joblib:**
   * Serializes and deserializes machine learning models for efficient deployment.

**Frontend:**

1. **Next.js:**
   * Framework for React-based frontend development with server-side rendering.
2. **Axios:**
   * Simplifies API calls and error handling.
3. **Tailwind CSS:**
   * Utility-first CSS framework for responsive and attractive designs.

**Deployment**

**Local Setup:**

**Backend:**

1. Install dependencies:
2. pip install -r requirements.txt
3. Run the Flask app:
4. python app.py

**Frontend:**

1. Install dependencies:
2. npm install
3. Start the development server:
4. npm run dev

**Environment Variables:**

* Backend URL: NEXT\_PUBLIC\_BACKEND\_URL

**Example Workflow**

1. **Input:** A user enters the URL bit.ly/example.
2. **Backend Processing:**
   * The URL is traced to https://example.com.
   * Metadata and outbound links are extracted.
   * The AI model predicts a 70% likelihood of piracy due to keywords like “free download”.
3. **Frontend Display:**
   * Results show the traced URL, metadata, and a warning about potential piracy.

**Summary**

This project offers a comprehensive solution for tracing URLs and detecting piracy using AI. It combines efficient backend processing with an intuitive frontend, addressing a critical need in media and entertainment industries. By leveraging open-source tools and libraries, the solution is scalable and cost-effective.