News Feed Application - Package

--- README ---

News Feed Application (Python + Flask)
==========

This is a simple News Feed Application written in Python using Flask.
It aggregates RSS/Atom feeds (via feedparser) and displays them on a web page.

Files:
- app.py : Main Flask application
- templates/index.html : Jinja2 template for display
- requirements.txt : Python dependencies

Setup (Linux/macOS/Windows):
1. Create a virtual environment:
   python3 -m venv venv
2. Activate it:
   - macOS/Linux: source venv/bin/activate
   - Windows (PowerShell): .\venv\Scripts\Activate.ps1
3. Install dependencies:
   pip install -r requirements.txt
4. Run the app:
   export FLASK_APP=app.py
   export FLASK_ENV=development
   flask run
   (On Windows cmd: set FLASK_APP=app.py & set FLASK_ENV=development & flask run)

Usage:
- Open http://127.0.0.1:5000 in your browser.
- By default the app fetches a set of example RSS feeds. You can modify the FEEDS list in app.p
y.

Notes:
- This example uses `feedparser` to parse RSS/Atom feeds.
- For production, consider caching feed results and handling errors more robustly.

--- requirements.txt ---

```
Flask>=2.0
feedparser>=6.0
requests>=2.0
```

--- app.py ---

```
"""
Simple News Feed Aggregator (Flask)
- Aggregates multiple RSS/Atom feeds using feedparser
- Displays latest items sorted by published date
"""

from flask import Flask, render_template, request
import feedparser
import requests
from datetime import datetime, timezone
import time

app = Flask(__name__)

# Example feeds (you can add/remove)
```

```
FEEDS = [
    "http://feeds.bbci.co.uk/news/rss.xml",
    "https://rss.nytimes.com/services/xml/rss/nyt/HomePage.xml",
    "https://www.theguardian.com/world/rss",
    "https://xkcd.com/atom.xml"  # fun example
]

# Simple fetcher - no caching (for demo only)
def fetch_feed(url):
    # Use requests to get raw content (helps with some sites)
    try:
        r = requests.get(url, timeout=10, headers={'User-Agent': 'NewsFeedApp/1.0'})
        raw = r.content
        parsed = feedparser.parse(raw)
        return parsed
    except Exception as e:
        return {"entries": [], "feed": {"title": str(e)}}

def parse_entry(entry):
    # Try several date fields
    published = None
    if 'published_parsed' in entry and entry.published_parsed:
        published = datetime.fromtimestamp(time.mktime(entry.published_parsed), tz=timezone.utc
)
    elif 'updated_parsed' in entry and entry.updated_parsed:
        published = datetime.fromtimestamp(time.mktime(entry.updated_parsed), tz=timezone.utc)
    return {
        "title": entry.get("title", "No title"),
        "link": entry.get("link", "#"),
        "summary": entry.get("summary", ""),
        "published": published
    }

@app.route("/", methods=["GET"])
def index():
    feeds_param = request.args.get("feeds", "")
    # Allow optionally providing comma-separated feeds via ?feeds=url1,url2
    feeds = FEEDS.copy()
    if feeds_param:
        feeds = [u.strip() for u in feeds_param.split(",") if u.strip()]
    all_items = []
    for url in feeds:
        parsed = fetch_feed(url)
        feed_title = parsed.feed.get("title", url)
        for e in parsed.entries:
            item = parse_entry(e)
            item["source"] = feed_title
            all_items.append(item)
    # Filter out items without date, sort by date desc (newest first)
    items_with_date = [it for it in all_items if it["published"] is not None]
    items_no_date = [it for it in all_items if it["published"] is None]
    items_with_date.sort(key=lambda x: x["published"], reverse=True)
    items = items_with_date + items_no_date
    # Limit to 100 items
    items = items[:100]
    return render_template("index.html", items=items, feeds=feeds)

if __name__ == "__main__":
```

```
    app.run(debug=True)
```

--- templates/index.html ---

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>News Feed App</title>
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <style>
    body { font-family: Arial, sans-serif; margin: 40px; background:#f8f9fb; color:#222; }
    .container { max-width:900px; margin:0 auto; }
    header { margin-bottom: 20px; }
    article { background:white; padding:16px; margin-bottom:12px; border-radius:8px; box-shadow
: 0 1px 3px rgba(0,0,0,0.08);}
    h1 { font-size:24px; }
    .meta { color:#666; font-size:13px; margin-bottom:8px; }
    a.title { color:#1a0dab; text-decoration:none; font-weight:600; font-size:18px; }
    .summary { margin-top:8px; color:#333; }
  </style>
</head>
<body>
  <div class="container">
    <header>
      <h1>News Feed Aggregator</h1>
      <p>Feeds: {{ feeds|join(", ") }}</p>
    </header>

    {% for item in items %}
      <article>
        <div class="meta">{{ item.source }} {% if item.published %}- {{ item.published.strftime
('%Y-%m-%d %H:%M UTC') }}{% endif %}</div>
        <a class="title" href="{{ item.link }}" target="_blank" rel="noopener noreferrer">{{ it
em.title }}</a>
        <div class="summary">{{ item.summary|safe }}</div>
      </article>
    {% else %}
      <p>No items found.</p>
    {% endfor %}
  </div>
</body>
</html>
```