

Nice, this is exactly the kind of question profs ask in reviews 👍

Below is a **clean, exam-ready list of Python frameworks & libraries** you'll realistically use for **this exact app**, grouped by purpose.

I'll also mark **core vs optional** so you don't overclaim.

## PYTHON FRAMEWORKS & LIBRARIES USED

### ♦ 1. Backend & API Layer (CORE)

~~Flask~~ **FastAPI** — ~~star one~~

- Purpose:
  - Build REST APIs
  - Handle article submissions (URL / text / image)
  - Return credibility score & supporting articles
- Why it's used:
  - Lightweight
  - Easy to integrate with NLP & OCR
- Role in project:
  - Central controller of the application

✓ **Mandatory**

### ♦ 2. NLP & Text Processing (CORE)

#### NLTK

- Text preprocessing:
  - Tokenization
  - Stopword removal
  - Lemmatization

#### scikit-learn

- TF-IDF vectorization
- Cosine similarity
- Basic statistical utilities

#### Sentence-Transformers

- Context-aware embeddings (Sentence-BERT)
- Improves semantic similarity across articles

✓ Mandatory

### ♦ 3. Web Scraping & Source Fetching (CORE)

#### Requests

- HTTP requests to news sources
- Fetch HTML or API responses

#### BeautifulSoup (bs4)

- Parse and extract article content from HTML

#### Feedparser

- Read RSS feeds from trusted sources (PTI, ANI, PIB, UN)

✓ Mandatory

### ♦ 4. OCR & Image Processing (CORE)

#### EasyOCR

- Extract text from article screenshots
- Supports noisy images better than pure Tesseract

#### Pytesseract

- OCR alternative / fallback

#### OpenCV (cv2)

- Image preprocessing:
  - Grayscale
  - Thresholding
  - Noise removal

✓ Mandatory

### ♦ 5. Caching & Storage (CORE)

#### SQLite

- Store:
  - Cached articles
  - Vector representations
  - API responses

## Pickle

- Serialize vectorized data for faster reuse

✓ Mandatory

## ♦ 6. Data Handling & Utilities (CORE)

## NumPy

- Vector operations
- Similarity computations

## Pandas

- Result tables
- Experiment analysis

✓ Mandatory

## ♦ 7. Frontend Integration (OPTIONAL but COMMON)

## Jinja2

- Server-side HTML rendering (if using Flask templates)

## Flask-CORS

- Enable frontend ↔ backend communication

⚠ Optional (depends on frontend choice)

## ♦ 8. Deployment & Hosting (OPTIONAL)

## Gunicorn

- Production WSGI server

## Uvicorn

- ASGI server (if using FastAPI)

## Docker (*optional but impressive*)

- Containerization

⚠ Optional (mention only if you use them)

## ♦ 9. Evaluation & Experimentation (SUPPORTING)

## Matplotlib / Seaborn

- Visualize:
  - Similarity distributions
  - Accuracy trends

## Scipy

- Statistical checks

⚠ Supporting tools

## EXAM-SAFE SUMMARY (USE THIS LINE)

*"The application is built using Flask for backend APIs, NLTK and Sentence-BERT for NLP processing, BeautifulSoup and Feedparser for source extraction, EasyOCR for image-based text extraction, and SQLite for caching and storage."*

This sentence alone can save you in a viva 😊

If you want next:

- I can map **which team member uses which library**
- Or help you **decide between Flask vs FastAPI** (prof-friendly choice)
- Or give you a `requirements.txt` ready to use