Perfect — this is the **right moment** to slow down and make sure **nobody gets blocked later** 👍

I'll explain this like a **senior dev onboarding a new team**, not like exam notes.

What you'll get below:

- Clear **mental model** of the system
- **Why** each component exists
- **How** to implement it
- **Common mistakes & how to avoid them**
- So even if someone forgets details, they can recover

Take this as the **master build guide**.

🧠 BIG PICTURE FIRST (READ THIS ONCE)

Your app answers **one question**:

> "Given this article, how strongly do trusted sources support it?"

You are **NOT proving truth**.
You are measuring **support consistency across trusted sources**.

That's why:

- You use similarity, not classification
- You show sources, not just a label
- You add a disclaimer

Everything else is just plumbing.

🏗️ SYSTEM MENTAL MODEL (VERY IMPORTANT)

Think of the app as **6 independent machines** connected in a pipeline:

```
css

Input
 ↓
Text Extraction
 ↓
Text Understanding
 ↓
Source Collection
 ↓
```

```
Comparison Engine
↓
Score + Explanation
```

Each machine should:

- Do **one job**
- Fail **gracefully**
- Never block the entire system

1️⃣ INPUT HANDLING (URL / TEXT / IMAGE)

Goal

Convert **any input** into **clean plain text**.

Why this matters

If text is bad:

- OCR errors
- Scraping noise
- Encoding issues

👉 Everything downstream breaks.

A) URL INPUT

What you do

1. Send HTTP request
2. Extract article body
3. Remove ads, menus, footers

How to implement

- Use `requests` to fetch page
- Use `BeautifulSoup`
- Extract:
  - `<article>`
  - `<p>` tags
- Join paragraphs

Common mistakes

❌ Scraping everything
❌ Including navigation text

❌ Ignoring encoding issues

Rule of thumb

> If you can read it comfortably, NLP can too.

B) TEXT INPUT

What you do

- Minimal validation
- Trim whitespace
- Ensure minimum length

Common mistake

❌ Over-cleaning (removing meaning)

C) IMAGE INPUT (OCR)

What OCR actually does

OCR **does NOT understand language**
It only guesses characters from pixels.

Correct OCR flow

1. Convert to grayscale
2. Remove noise
3. Increase contrast
4. Run OCR
5. Calculate confidence

What to do with bad OCR

- Do NOT fail
- Warn user:

  > “Low confidence OCR, results may be inaccurate”

Mistake to avoid

❌ Pretending OCR is always accurate

2️⃣ TEXT PREPROCESSING (NLP FOUNDATION)

Goal

Make text **machine-comparable**, not pretty.

Steps (in order)

1. Lowercase
2. Remove URLs & symbols
3. Tokenize
4. Remove stopwords
5. Lemmatize

Why lemmatization?

- “running”, “ran”, “runs” → “run”
- Reduces noise

What NOT to do

❌ Remove proper nouns (India, UN, Modi)
❌ Remove numbers blindly

News meaning often depends on names.

3️⃣ VECTOR REPRESENTATION (MOST IMPORTANT CONCEPT)

Question this answers

> “How do we convert text into numbers?”

Because math works on numbers, not words.

A) TF-IDF (Baseline)

What it captures

- Important keywords
- Term importance

Pros

- Fast
- Simple
- Explainable (good for viva)

Cons

- No semantic meaning
- “President” ≠ “Head of State”

B) Sentence-BERT (Semantic)

What it captures

- Meaning, not just words
- Context

Why you test both

Because:

- TF-IDF might work better for factual news
- SBERT might work better for paraphrased articles

Important rule

👉 **Choose ONE final method** after experiments
Don't mix unless you know why.

4️⃣ SOURCE FETCHING (HIGH-RISK MODULE)

Goal

Collect **related articles** from trusted sources.

What you are NOT doing

- You are NOT scraping the entire internet
- You are NOT training a classifier

Correct strategy

1. Use **RSS feeds if available**
2. Fallback to scraping
3. Skip sources that fail

Why skipping is OK

Because your system says:

> "Based on available articles"

Not:

> "Based on all articles in existence"

Caching (VERY IMPORTANT)

Why cache?

- Same article submitted multiple times
- Same source fetched repeatedly

Cache what?

- URL → extracted text
- Text → vector

Mistake to avoid

❌ Re-scraping every time
❌ Re-vectorizing same text

5️⃣ SIMILARITY COMPUTATION

This is NOT classification

You are asking:

> "How similar is this article to trusted articles?"

Cosine Similarity (intuition)

- Measures **angle** between vectors
- Scale doesn't matter
- Meaning does

Similarity ranges

- `0.7+` → Strong support
- `0.4–0.7` → Weak/partial
- `<0.4` → Irrelevant

(Exact thresholds = experimental)

Support vs Contradiction

- High similarity + same stance → support
- High similarity + opposite sentiment → contradiction
- Low similarity → ignore

👉 You **ignore irrelevant articles**
This avoids false penalties.

6️⃣ CREDIBILITY SCORE (CORE LOGIC)

What the score means

> “Percentage of trusted articles that support this claim”

NOT:

- Truth probability
- Fact guarantee

Simple, explainable formula

```ini
Score = (Supporting Articles / Total Considered Articles) × 100
```

Why this works

- Easy to explain
- Transparent
- No black-box ML

Edge case handling

- No articles found → “Insufficient data”
- Mixed support → “Uncertain”

7️⃣ SUMMARY GENERATION

Why summaries?

Users don’t want to read full articles.

Technique

- Extractive summarization
- Pick most relevant sentences

What NOT to do

❌ Generative hallucinations
❌ Long summaries

8️⃣ FASTAPI BACKEND (GLUE OF EVERYTHING)

Why FastAPI?

- Async (good for scraping)
- Auto Swagger docs
- Clean separation

How request flows

1. `/analyze` receives input
2. Input normalized
3. NLP pipeline runs
4. Sources fetched
5. Similarity computed
6. Score calculated
7. Response returned

Important rule

> Backend NEVER assumes frontend is correct

Always validate.

9️⃣ FRONTEND (USER EDUCATION TOOL)

Frontend is not decoration — it's **interpretation**.

What UI must show clearly

- Score meaning
- Supporting articles
- Links
- Disclaimer

Mistake to avoid

❌ Big "FAKE" / "TRUE" labels
This can get you into ethical trouble.

🔟 ERROR HANDLING PHILOSOPHY

Never crash.

Always explain.

Issue	What you do
Source down	Skip
OCR bad	Warn
Low similarity	Mark uncertain
No articles	Insufficient data

1️⃣1️⃣ ETHICS (EXAMINERS CARE)

You must say:

- “This is an assistive tool”
- “Not a final authority”
- “User must cross-check”

This protects you academically & legally.

1️⃣2️⃣ HOW NO ONE GETS STUCK (TEAM RULES)

1. **Modules are independent**
2. If blocked → mock input
3. Log everything
4. Freeze features early
5. Explain failures openly in report

🧠 FINAL MINDSET (IMPORTANT)

This project is **not about being perfect**.

It is about:

- Clear thinking
- Honest engineering
- Responsible AI

You’re building something **real**, not a toy.

If you want next, I can:

- Give **folder structure**
- Write **starter FastAPI code**
- Create **“if this breaks, do this” troubleshooting guide**
- Prepare **viva killer answers**

Just tell me what you want next.