

LLM Project: Building a News Research Tool

In this project, I developed an AI-powered web app called the **News Research Tool** to help users instantly access and understand current events. Using **LangChain**, **Groq's LLaMA3**, and **NewsAPI**, the app fetches real-time news articles based on any user query and summarizes them into crisp bullet points.

I started by configuring API keys and integrating the **Groq LLM** with **LangChain**, creating smart prompt templates to ensure neutral, high-quality outputs. Then I connected **NewsAPI** to pull relevant articles and extract key descriptions. This content is passed into the LLM to generate summaries.

The frontend, built with **Streamlit**, includes features like **login authentication**, **example query buttons**, **text input**, and **export to TXT/PDF**. Users can view their **last five searches**, and the app maintains a clean UI with helpful feedback and interactivity. The tool is now live and designed for quick, intelligent news research.



by Debasis Baidya

A blue-toned background graphic featuring a grid of hexagonal icons. The icons include a dollar sign, a cloud, a lightbulb, a Wi-Fi symbol, a gear, a wrench, a truck, and a person on a bicycle. A dark grey rectangular box is overlaid on the right side, containing the text 'RESEARCH NEWS' in white, bold, sans-serif font. A finger is shown pointing at the right edge of this box.


**RESEARCH
NEWS**

Project Objective & Purpose







01

Goal

-  Build an interactive dashboard that summarizes real-time news using LLM along with Top News Header, Articles Used for Summary, Past Queries & Also Downloadable in TXT & PDF File.




02

Core Objectives

-  Use LLM to generate short, crisp news summaries
-  Fetch real-time articles using NewsAPI
-  Build a clean and secure Streamlit-based frontend
-  Allow export of summaries in multiple formats

03


Purpose

-  Help users consume news faster through AI-powered summaries.
-  Make the summary accurate, clean, and non-repetitive.
-  Reduce information overload with concise insights.



Tech Stack Used

 **Python 3.13**


 Streamlit for UI


 reportlab for PDF generation

 Groq's LLaMA3 model – for summarization

 LangChain – LLM chaining logic

 NewsAPI for fetching live news




 python-dotenv for API security

 Streamlit Secrets for keeping Groq & NewsAPI Keys safely

Streamlit Secrets for keeping Groq & NewsAPI Keys safely



1

Languages & Frameworks

- ☐  Python 3.13
- ☐  Streamlit for UI
- ☐  reportlab for PDF generation



2

LLM & Logic

- ☐  Groq's LLaMA3 model – for summarization
- ☐  LangChain - LLM chaining logic


3

Data Source & APIs Integrated

- ☐  NewsAPI for fetching live news
- ☐  groq, langchain-groq – to access Groq's LLaMA3

4

Other Tools

- ☐  python-dotenv for API security
- ☐ Streamlit Secrets for keeping Groq & NewsAPI Keys safely



Backend Setup

01



Environment & Keys

- ❑  Installed required libraries: **langchain, groq, newsapi-python, streamlit, dotenv**
- ❑  Managed API keys using **.env** file

02



LangChain + Groq Setup

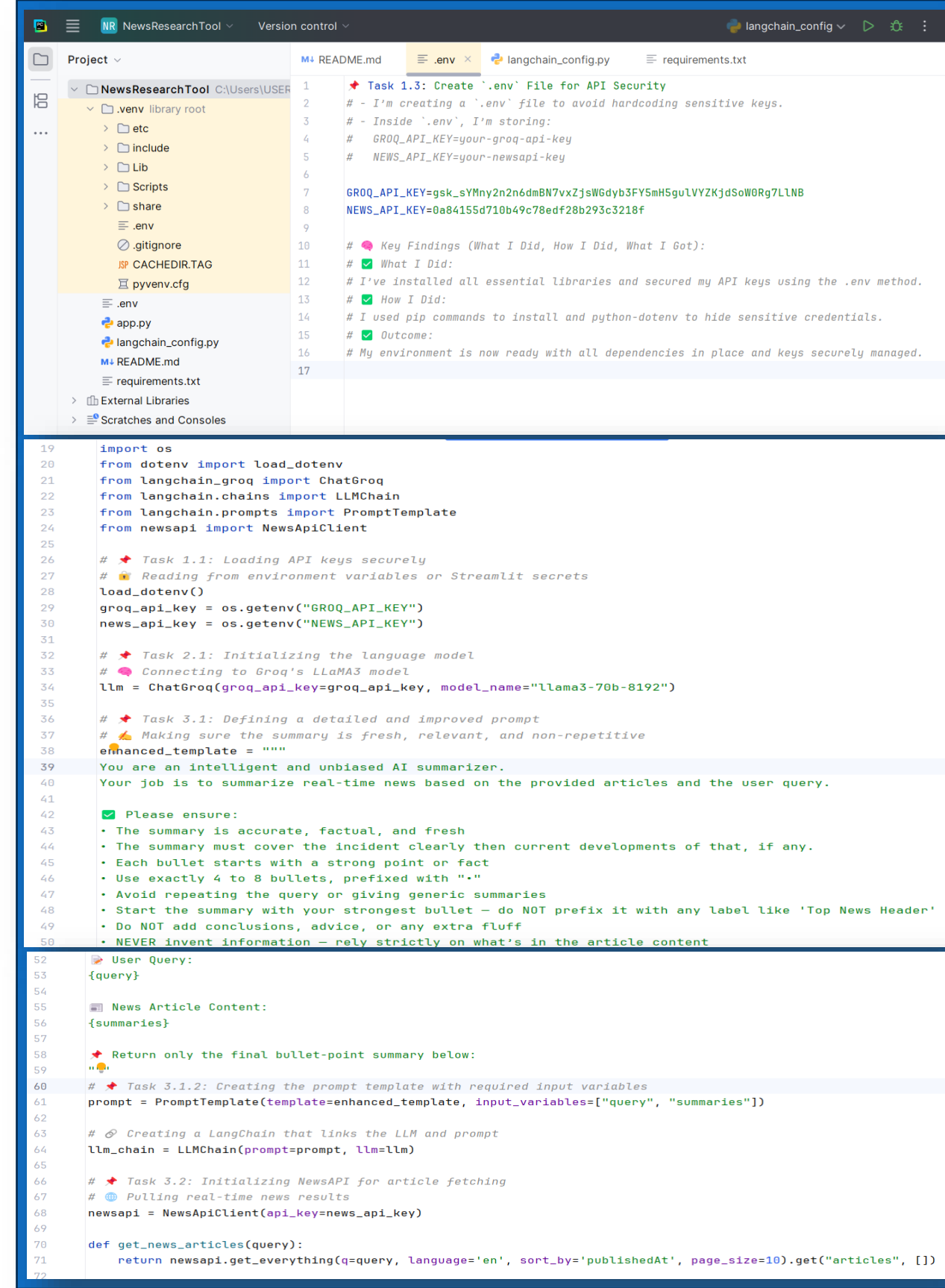
- ❑  Integrated **Groq LLM** via **LangChain**
- ❑  Created **prompt templates** for crisp bullet summaries

03



News Fetching

- ❑  Fetched articles from **NewsAPI** based on query
- ❑  Extracted article descriptions for **summarization**



```
Project v
└─ NewsResearchTool C:\Users\USER
   └─ .env library root
      └─ etc
      └─ include
      └─ Lib
      └─ Scripts
      └─ share
      └─ .env
      └─ .gitignore
      └─ .JSP CACHEDIR.TAG
      └─ .pyvenv.cfg
      └─ .env
      └─ app.py
      └─ langchain_config.py
      └─ README.md
      └─ requirements.txt
      └─ External Libraries
      └─ Scratches and Consoles

1  Task 1.3: Create '.env' File for API Security
2  # - I'm creating a '.env' file to avoid hardcoding sensitive keys.
3  # - Inside '.env', I'm storing:
4  #   GROQ_API_KEY=your-groq-api-key
5  #   NEWS_API_KEY=your-newsapi-key
6
7  GROQ_API_KEY=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
8  NEWS_API_KEY=0a84155d710b49c78edf28b293c3218f
9
10 # 📌 Key Findings (What I Did, How I Did, What I Got):
11 # ✅ What I Did:
12 # I've installed all essential libraries and secured my API keys using the .env method.
13 # ✅ How I Did:
14 # I used pip commands to install and python-dotenv to hide sensitive credentials.
15 # ✅ Outcome:
16 # My environment is now ready with all dependencies in place and keys securely managed.
17





19 import os
20 from dotenv import load_dotenv
21 from langchain_groq import ChatGroq
22 from langchain.chains import LLMChain
23 from langchain.prompts import PromptTemplate
24 from newsapi import NewsApiClient
25
26 # 📌 Task 1.1: Loading API keys securely
27 # 📌 Reading from environment variables or Streamlit secrets
28 load_dotenv()
29 groq_api_key = os.getenv("GROQ_API_KEY")
30 news_api_key = os.getenv("NEWS_API_KEY")
31
32 # 📌 Task 2.1: Initializing the language model
33 # 📌 Connecting to Groq's LLaMA3 model
34 llm = ChatGroq(groq_api_key=groq_api_key, model_name="llama3-70b-8192")
35
36 # 📌 Task 3.1: Defining a detailed and improved prompt
37 # 📌 Making sure the summary is fresh, relevant, and non-repetitive
38 enhanced_template = """
39 You are an intelligent and unbiased AI summarizer.
40 Your job is to summarize real-time news based on the provided articles and the user query.
41
42 ✅ Please ensure:
43 • The summary is accurate, factual, and fresh
44 • The summary must cover the incident clearly then current developments of that, if any.
45 • Each bullet starts with a strong point or fact
46 • Use exactly 4 to 8 bullets, prefixed with "•"
47 • Avoid repeating the query or giving generic summaries
48 • Start the summary with your strongest bullet – do NOT prefix it with any label like 'Top News Header'
49 • Do NOT add conclusions, advice, or any extra fluff
50 • NEVER invent information – rely strictly on what's in the article content
51
52 📌 User Query:
53 {query}
54
55 📌 News Article Content:
56 {summaries}
57
58 📌 Return only the final bullet-point summary below:
59 ""
60
61 # 📌 Task 3.1.2: Creating the prompt template with required input variables
62 prompt = PromptTemplate(template=enhanced_template, input_variables=["query", "summaries"])
63
64 # 📌 Creating a LangChain that links the LLM and prompt
65 llm_chain = LLMChain(prompt=prompt, llm=llm)
66
67 # 📌 Task 3.2: Initializing NewsAPI for article fetching
68 # 📌 Pulling real-time news results
69 newsapi = NewsApiClient(api_key=news_api_key)
70
71 def get_news_articles(query):
72     return newsapi.get_everything(q=query, language='en', sort_by='publishedAt', page_size=10).get("articles", [])
```



LLM Logic | Prompt Engineering & News Scraping with NewsAPI






Prompt Template Highlights

-  Outputs: Bullet-form AI-generated summary
-  Avoids repetition, fluff, and generic intros
-  Starts with the strongest point
-  No hallucination – based only on real content






LangChain Chain Flow

-  Inputs: User query + article text
-  Ensured context-aware, accurate output
-  Better summarization from multiple article sources





Article Fetching Strategy

-  Pulls the latest 10 articles based on the user's query
-  Filters articles by language = 'en' (English)
-  Sorts results by "publishedAt" for most recent coverage



Text Preprocessing

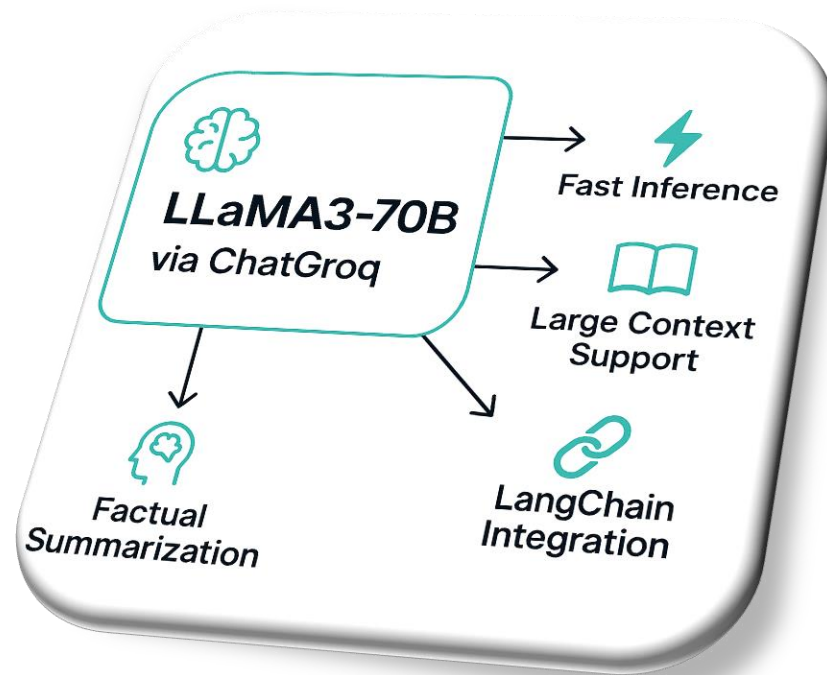
-  Extracts and cleans the description and content fields of each article
-  Combines the cleaned text into a single string as input for the LLM summarizer

```
32 # 🚀 Task 2.1: Initializing the language model
33 # 🧠 Connecting to Groq's LLaMA3 model
34 llm = ChatGroq(groq_api_key=groq_api_key, model_name="llama3-70b-8192")
35
36 # 🚀 Task 3.1: Defining a detailed and improved prompt
37 # 🗣️ Making sure the summary is fresh, relevant, and non-repetitive
38 enhanced_template = """
39 You are an intelligent and unbiased AI summarizer.
40 Your job is to summarize real-time news based on the provided articles and the user query.
41
42 ✅ Please ensure:
43 • The summary is accurate, factual, and fresh
44 • The summary must cover the incident clearly then current developments of that, if any.
45 • Each bullet starts with a strong point or fact
46 • Use exactly 4 to 8 bullets, prefixed with "•"
47 • Avoid repeating the query or giving generic summaries
48 • Start the summary with your strongest bullet – do NOT prefix it with any label like 'Top News Header'
49 • Do NOT add conclusions, advice, or any extra fluff
50 • NEVER invent information – rely strictly on what's in the article content
51
52 🗨️ User Query:
53 {query}
54
55 📰 News Article Content:
56 {summaries}
```

```
70 def get_news_articles(query):
71     return newsapi.get_everything(q=query, language='en', sort_by='publishedAt', page_size=10).get("articles", [])
72
73 # 🚀 Task 3.2.1: Extracting usable text for the model
74 # 📄 Combining content and description from each article
75 def summarize_articles(articles):
76     return ' '.join(
77         article.get('description') or article.get('content') or ''
78         for article in articles
79     )
80
81 # 🚀 Task 3.3: Main entry point for summarization
82 # 🧠 Returning the summary output and top articles for UI rendering
83 def get_summary(query):
84     articles = get_news_articles(query)
85     summaries = summarize_articles(articles)
86
87     if not summaries.strip():
88         return "⚠️ No content found to summarize. Try another topic.", []
89
90     used_articles = [a for a in articles if a.get('description') or a.get('content')]
91     summary_output = llm_chain.run(query=query, summaries=summaries)
92
93     return summary_output, used_articles
```




Why Groq LLM (LLaMA3) | Authentication Logic






Username

Password

Log In

 Shows error for wrong login

-  Username & password gate
-  Hint system for credentials
-  Login state persists during session
- Reset
- Keeps login active

01

Model Used





- ☐  LLaMA3-70B model via ChatGroq

Why I Chose Groq

- ☐  Fast inference & high accuracy
- ☐  Large context support
- ☐  Seamlessly works with LangChain
- ☐  Better factual summarization than other models


03

Security Features

- ☐  Username & password gate
- ☐  Hint system for credentials
- ☐  Login state persists during session
- ☐  Shows error for wrong login

04



Reset Functionality

- ☐  Clears all user input and response
- ☐ Keeps login active after reset

Streamlit Dashboard Core Features




STEP 01

User Authentication & API Keys Storing:

-  Login with username and password: (Demo: Debasis / Baidya123)
-  Groq & News API Keys kept in Streamlit Secret.



STEP 02

Query Input + AI Summary

-  User enters any topic
-  LLM returns a clean bullet summary
-  Shows which articles were used.

STEP 03

Export Options

-  PDF export (Unicode-safe with ReportLab)
-  TXT export (plain text summary)

STEP 04

Past Query History

-  Stores and displays last 5 queries thus allows user to revisit past results

STEP 05

Predefined Example Buttons

-  Includes topics like “Air India Crash”, “India-Pak War”, “Indian Economy”

STEP 06

Session Reset Button

-  Clears all fields without logging out

LLM: News Research Tool

Summarizing real-time news articles smartly using AI  Login Required

Login Required

Username: Debasis | Password: Baidya123

Username

Try: Debasis

Password

Try: Baidya123



Login

App settings

General

Sharing

Secrets

Secrets

Provide environment variables and other secrets to your app using [TOML](#) format. This information is encrypted and served securely to your app at runtime. Learn more about Secrets in our [docs](#). Changes take around a minute to propagate.

```
# 🌟 Task 1.3: Create `.env` File for API Security
# - I'm creating a `TOML` file to avoid hardcoding sensitive keys.
# - Inside `TOML`, I'm storing:
#   GROQ_API_KEY=your-groq-api-key
#   NEWS_API_KEY=your-newsapi-key
```

```
GROQ_API_KEY =
"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"
NEWS_API_KEY = "0a84155d710b49c78edf28b293c3218f"
```

Save changes



Output Demo Snapshot



Header Section



Uses strongest bullet or top article title



AI Summary



Shows 5–8 factual bullet points from articles



Used Articles Section



Displays title, source, date, and clickable links



Download Zone



Export buttons for PDF & TXT

```
if gen_btn:
    if query:
        response, articles = get_summary(query)
        bullet_lines = [f"• {line.strip()}" for line in response.split("•") if line.strip()]
        header_line = articles[0].get("title", "Top News") if articles else (bullet_lines[0][1:].strip() if bullet_lines else "")
        formatted_summary = "\n".join(bullet_lines[1:]) if len(bullet_lines) > 1 else ""

        st.markdown(f"""
            <div style='background-color:#f0f2f6; text-align:center; padding: 0.75rem 1rem; margin-top: 1rem;'>
                <span style='margin:0; font-size: 16px; color: #333;'>📰 <strong>Top News Header:</strong> {header_line}</span>
            </div>
            """, unsafe_allow_html=True)

        st.markdown("<h3 style='text-align:center;'>🧠 AI-Generated News Summary</h3>", unsafe_allow_html=True)
        st.markdown(f"""
            <div style='background-color:#e8f0fe; padding:1rem; border-radius:6px;'>
                <ul style='padding-left: 1.2rem; margin: 0;'>
                    {''.join([f'<li style="margin-bottom: 0.4rem;">{line[1:].strip()}</li>' for line in formatted_summary.splitlines() if line.strip()])}
                </ul>
            </div>
            """, unsafe_allow_html=True)

        articles_text = ""
        if articles:
            st.markdown("<h3 style='text-align:center;'>📄 Articles Used for Summary</h3>", unsafe_allow_html=True)
            top_articles = articles[:3]
            for article in top_articles:
                title = article.get("title", "No title")
                source = article.get("source", {}).get("name", "Unknown Source")
                date = article.get("publishedAt", "").split("T")[0]
                url = article.get("url", "#")
                article_block = f"- {title}\n 📅 {date} | 📍 {source}\n 🔗 [Read More]({url})"
                st.markdown(article_block)
                articles_text += f"{article_block}\n"
            st.markdown(f"""
                <div style='background-color:#e6ffe6; border-radius:6px; padding:0.5rem; margin-top:1rem; text-align:center;'>
                    <span style='color: #2e7d32; font-weight: 600;'>✅ Summary extracted from 3 article(s).</span>
                </div>
                <div style='margin-bottom: 1rem;'></div>
            """, unsafe_allow_html=True)
        else:
            st.warning("⚠️ No articles available.")

    if 'history' not in st.session_state:
        st.session_state.history = []
    st.session_state.history.append((query, formatted_summary))
```


✓ Results, Impact, Learnings & Future Enhancements

01



What I Achieved

- ✈ Built a working LLM-based summarizer
- 🔗 Integrated Groq & NewsAPI with LangChain
- 🛠 Created a clean, exportable UI
- 🔒 Used secure environment-based API access

02



Why It Matters

- 🕒 Saves time for users
- 🧠 Provides accurate, instant news summaries
- 💻 Fully usable tool for research or awareness

03



What I Learned

- 🔧 Prompt engineering for summarization
- 🔑 API key security with .env
- 💻 Streamlit UI/UX design
- 📄 Report generation using ReportLab

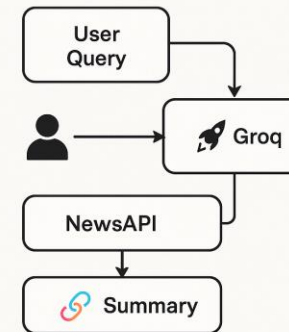
04



Key Outcomes & Milestones

- 🔄 Fully working, smart news research tool
- 🧠 Real-time summarization with AI
- 💻 Deployed Streamlit app with export + security
- 🔗 Made it accessible via public link + demo

What I Achieved



```
summary_tool = load_langchain  
response = summary_tpost(  
    'text' = article_text,  
    'query' = 'Summarize the  
    above article')  
summary = response['text']
```



Why It Matters



Saves time



Provides accurate
instant news



Fully usable
tool for



Prompt engineering
for summarization



API key security
with .env



Streamlit UI/UX
design



Report generation
using ReportLab

Key Outcomes & Milestones



Fully working,
smart news
research tool



Real-time
summarization
with AI



Deployed Streamlit
app with export
+ security



Made it accessible
via public link + demo

LLM: News Research Tool

Summarizing real-time news articles smartly using AI 🤖 Login Required

🌟 Try queries like:

Click any button below to auto-fill the input box

Air India Crash

India-Pak War

Israel-Iran War

IPL 2025 Incident

🔍 Enter your Query

Air India flight crashes in Ahmedabad, India

⚡ Generate Summary

🔄 Reset All

📰 Top News Header: Aviation Ministry Proposes Demolition Rules After Ahmedabad Air India Crash

🧠 AI-Generated News Summary

- On June 12, 2025, an Air India flight crashed after takeoff from Ahmedabad, India, killing 241 people on board, with only one survivor, a man seated in 11A.
- The crash has reignited global concerns about aviation safety, prompting a reflection on past tragedies, including recent air crashes like the AI 171 disaster, which highlight critical safety concerns.
- The focus is on adherence to protocols and the need for stricter checks, with technology, like AI, able to enhance safety through real-time data analysis and predictive maintenance.
- Investigators are currently sifting through wreckage and decoding black boxes to learn the cause of the crash.

📰 Articles Used for Summary

- Aviation Ministry Proposes Demolition Rules After Ahmedabad Air India Crash 📅 2025-06-19 | 🗞️ Thehillstimes.in 🔗 [Read More](#)
- The man in seat 11A survived, but why don't our patients? 📅 2025-06-18 | 🗞️ Kevinmd.com 🔗 [Read More](#)
- From MH370's disappearance to AI171's deadly crash: What really happened in the world's worst air disasters of recent years 📅 2025-06-18 | 🗞️ The Times of India 🔗 [Read More](#)

✅ Summary extracted from 3 article(s).

📄 Download as TXT

📄 Download as PDF

📋 Past Queries

1. Air India flight crashes in Ahmedabad, India

• On June 12, 2025, an Air India flight crashed after takeoff from Ahmedabad, India, killing 241 people on board, with only one survivor, a man seated in 11A. • The crash has reignited global concerns...

2. Israel-Iran War

• The attack on the hospital sparked Israeli officials to demand global outrage and condemnation of the Iranian missile attack. • The escalating conflict has prompted governments worldwide to evacuate...

3. IPL 2025 Incident

• Following Royal Challengers Bengaluru's IPL victory, a celebration parade outside M. Chinnaswamy Stadium resulted in 11 deaths and 50 injuries, with police arresting four individuals, including RCB'...

4. Israel-Iran War

• The attack came after Israel attacked a heavy water reactor in Iran, which the IDF said was a key component in plutonium production" for a nuclear bomb. • Governments around the world are attempting...

5. Air India flight crashes in Ahmedabad, India

• On June 12, 2025, Air India flight crashed minutes after takeoff from Ahmedabad in India, killing 241 people, with only one survivor, a man seated in seat 11A. • The crash has reignited global conce...

🚀 Deployment & Links

🌐 Live App

- 🔗 [Open the Tool](#)

🎬 Demo Video

- 🎥 [Watch Demo](#)

📦 Deployment Highlights

- 🐱‍💻 One-click web access
- 💡 Clean UI with hinting and placeholder text
- 🔒 Secure environment variables

✅ Full Feature List

- 🔒 Login authentication
- 🧠 LLM-powered summarization
- 📡 Real-time news fetching
- ⚙️ Quick query buttons
- 📄 TXT & PDF export
- 🕒 Query history
- 🎨 Styled responsive UI
- 🔄 Reset all input feature



Thank You

For your time & attention