

# A PROJECT REPORT

ON

## FACT FILTER – A MULTI-FORMAT FAKE NEWS DETECTION SYSTEM

Project Report Submitted to



### GOVERNMENT E. RAGHVENDRA RAO POSTGRADUATE SCIENCE COLLEGE, BILASPUR (C.G.)

*(Affiliated to Atal Bihari Vajpayee Vishwavidyalaya, Bilaspur C.G.)*

Grade - 'B+' Accredited by NAAC

In Partial Fulfillment for the Award of the Degree of  
BACHELOR OF COMPUTER APPLICATION (BCA – VI Semester)

Submitted by

**RAYSON JUDE MURRAY**

Roll No.: 22162

Enrollment No.: ABVV/GERS/2022/0748

Session: 2024-25

# GOVERNMENT E. RAGHAVENDRA RAO POSTGRADUATE SCIENCE COLLEGE, BILASPUR (C.G.)



## DEPARTMENT OF COMPUTER APPLICATIONS CERTIFICATE

This is to certify that this project entitled “Fact Filter - A Multi-Format Fake News Detection System using Whisper, Gemini API, and NLP” submitted in partial fulfilment of the degree of **BACHELOR OF COMPUTER APPLICATIONS** to Department of Computer Application, Government E. Raghavendra Rao PG Science College, Bilaspur (C.G), done by **RAYSON JUDE MURRAY**, is an authentic work carried out by him. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

**SANJEEV PANDEY**

Project Guide

**Mr. Vivek Tiwari**

Head of Department  
Department of Computer Application  
Govt. E. Raghavendra PG Science College  
Bilaspur (C.G.)

# DECLARATION

I hereby declare that the project entitled “Fact Filter - A Multi-Format Fake News Detection System using Whisper, Gemini API, and NLP” submitted to the Department of Computer Applications, Govt. E. Raghvendra Rao P.G. Science College, Bilaspur, is the result of my original and independent research work carried out during the academic year as part of the curriculum for the Bachelor of Computer Applications (BCA) program.

This project is a reflection of my understanding and application of concepts in artificial intelligence, natural language processing, full-stack development, and software engineering. All the work presented in this report is genuine and has been completed solely by me, under the academic supervision and guidance of my respected faculty members.

This report has **not been submitted** to any other institution or university, in part or full, for the award of any degree, diploma, certificate, or any academic recognition. I have made every effort to ensure that proper citation and credit is given to any external sources or tools used, and I have maintained academic integrity throughout the development and documentation of this project.

I take full responsibility for the authenticity of the data, analysis, design, implementation, and outcomes presented in this project report.

## **Signature**

**RAYSON JUDE MURRAY**

Roll No: 22162

*Date-15/07/2025*

Department of Computer Applications

Govt. E. Raghvendra Rao P.G. Science College, Bilaspur

# ACKNOWLEDGEMENT

---

With immense respect and gratitude, I would like to express my heartfelt thanks to everyone who has directly or indirectly contributed to the successful completion of my project titled **"Fact Filter - A Multi-Format Fake News Detection System using Whisper, Gemini API, and NLP."**

This project, developed as part of my final year in the **Bachelor of Computer Applications (BCA)** program, would not have been possible without the guidance, support, and encouragement of several individuals.

First and foremost, I extend my sincere appreciation to **Mr. Sanjeev Pandey**, Department of Computer Applications, **Govt. E. Raghvendra Rao P.G. Science College, Bilaspur**, for his valuable mentorship, timely feedback, and constant encouragement throughout the project journey. His insights helped me transform ideas into a fully functional and innovative system.

I would also like to thank the **Head of Department** and all the **faculty members** of the **Department of Computer Applications** for providing an excellent academic environment, access to necessary resources, and continued support that fostered my technical and professional growth.

This project was independently conceptualized, designed, and developed by me using a variety of advanced technologies. It involved end-to-end implementation of a web application including frontend development with React, backend with FastAPI, and integration of artificial intelligence models for misinformation detection. Building this system greatly enhanced my skills in machine learning, API development, software architecture, and full-stack integration.

I also wish to acknowledge the help and motivation provided by my friends and peers, who shared their honest opinions during development and encouraged me during challenges. Special thanks to my family for their unwavering moral support and belief in my capabilities throughout my academic journey.

Their collective support and faith played a crucial role in the completion of this academic endeavor.

**RAYSON JUDE MURRAY**

Department of Computer Applications

Govt. E. Raghvendra Rao P.G. Science College, Bilaspur

# ABSTRACT

---

The exponential growth of misinformation across digital platforms has posed a serious threat to public trust, media integrity, and national security. With increasing reliance on multimedia content—such as videos, audio clips, and images—traditional text-based fact-checking tools fall short in handling this complexity. To address this gap, this project presents **Fact Filter - A Multi-Format Fake News Detection System**, an AI-driven platform capable of verifying the authenticity of content across multiple input types including **text, image, audio, and video**.

The system is developed using **FastAPI** for the backend and **React** for the frontend, making it both lightweight and modular. Key technologies integrated into the system include **OpenAI Whisper** for real-time speech-to-text transcription, the **Gemini API** for contextual factual analysis, and a **custom-trained NLP model** using Logistic Regression for classifying content as *Real*, *Fake*, or *Opinion*. Each input format has its own dedicated processing pipeline, ensuring optimized handling of media-specific challenges like OCR for images and audio extraction from videos.

All results are logged in structured CSV files (`gemini_result.csv` and `nlp_result.csv`) to facilitate future model improvement and to study misinformation trends. The frontend interface is designed to be user-friendly and responsive, with separate upload pages for each format and clearly visualized results.

The project was built independently and demonstrates the real-world applicability of combining machine learning, natural language processing, and modern web technologies. **Fact Filter** serves as both an academic achievement and a functional prototype that could be extended for use in media monitoring, journalism, public awareness campaigns, and educational tools. It highlights the importance of scalable, AI-powered solutions in combating the growing threat of digital misinformation in today's media landscape.

**RAYSON JUDE MURRAY**

Bachelor of Computer Applications (BCA)

Govt. E. Raghvendra Rao P.G. Science College, Bilaspur

# Index

---

## **1. Introduction**

- 1.1 Problem Statement
- 1.2 Objective of the Project
- 1.3 Scope of the Project
- 1.4 Motivation

## **2. System Overview**

- 2.1 System Architecture
- 2.2 Technology Stack
- 2.3 Modules Description

## **3. Project Context and Problem Landscape**

- 3.1 Observations from the Digital Ecosystem
- 3.2 Gaps in Current Tools (Based on Personal Exploration)
- 3.3 Why Multi-Format Fact-Checking Is Necessary Today
- 3.4 The Unique Approach of Fact Filter

## **4. Methodology**

- 4.1 Software Development Life Cycle (SDLC) Used
- 4.2 Tools and Technologies
- 4.3 Model/Algorithm Used (e.g., Whisper, Gemini, Custom NLP Model)

## **5. System Design**

- 5.1 Data Flow Diagrams (DFD)
- 5.2 UML Diagrams (Use Case, Sequence, Class Diagram)
- 5.3 Database Design (if applicable)
- 5.4 UI/UX Design Overview

## **6. Implementation**

- 6.1 Frontend Overview
- 6.2 Backend API Endpoints
- 6.3 Fact-Checking Logic
- 6.4 Video Upload and Transcription
- 6.5 Report Generation and Export

## **7. Testing**

- 7.1 Testing Methodology
- 7.2 Test Cases and Results

## **8. Result and Analysis**

- 8.1 Output Screenshots
- 8.2 Performance Evaluation
- 8.3 Accuracy of Fact Checking

## **9. Conclusion and Future Scope**

- 9.1 Conclusion
- 9.2 Limitations
- 9.3 Future Enhancements

## **10. Appendix**

- 10.1 Source Code Links
- 10.2 Deployment Links
- 10.3 Dataset or Model Info
- 10.4 User Manual / Installation Guide

# 1. Introduction

---

The prevalence of misinformation and disinformation in today's digital ecosystem has escalated into a global challenge. With the exponential rise in content consumption, especially through videos on social media and news platforms, detecting and mitigating fake news has become more critical than ever. Traditional text-based fake news detection systems fall short when it comes to handling video-based content, which can contain complex and nuanced information in both visual and audio formats. This project, titled **Fact Filter**, aims to bridge that gap by providing a powerful AI-driven solution that detects fake news from video content.

Fact Filter is a video-based fake news detection system that leverages cutting-edge artificial intelligence technologies—such as OpenAI Whisper for speech-to-text transcription, Gemini API for factual validation, and a custom-trained NLP model for classifying content as Real, Fake, or Opinion. This integrated approach enables the system to analyze news videos intelligently and provide users with fact-checking results in real time or near-real time.

I designed this project to serve as both a research-backed academic submission and a prototype for real-world use, especially in the domains of media monitoring, journalism, education, and public safety.

---

## 1.1 Problem Statement

In the digital era, information spreads rapidly—especially via video content shared on social media platforms, YouTube, and news websites. Unfortunately, this also enables the propagation of false information, conspiracy theories, and manipulated content that can influence public perception, cause unrest, or lead to dangerous real-world consequences.

Most existing fake news detection tools are limited to analyzing textual data and fail to address audio or video-based misinformation. There is a critical need for a tool that:

- Accepts video input (rather than just text or URLs)
- Extracts and transcribes speech accurately
- Validates facts from transcripts using verified knowledge sources
- Clearly labels the content as Real, Fake, or Opinion

I developed this project to address this gap by creating a system that brings multi-modal fake news detection to life using the latest AI tools.

---

## 1.2 Objective of the Project

The core objective of Fact Filter is to design and implement an intelligent, web-based system capable of:

- Accepting video files as input from users
- Converting speech from videos into accurate transcripts using OpenAI Whisper
- Performing fact-checking using the Gemini API and a custom NLP model
- Presenting a clear factual status report for the given content
- Offering a clean, user-friendly interface to visualize fact-checking results

The system is designed to be robust, scalable, and modular, allowing for future enhancements and integration with third-party services.

---

## 1.3 Scope of the Project

Fact Filter is built as a multi-format misinformation detection system, capable of handling different types of inputs through dedicated interfaces. The goal was to build a flexible and extendable architecture that could adapt to real-world fact-checking needs across media types.

### Input Scope:

- Text: Users can enter written content via `/format/text`
- Image: Optical Character Recognition (OCR) is used to extract and analyze embedded text via `/format/image`
- Audio: Uploaded audio files are transcribed via Whisper using `/format/audio`
- Video: Audio is extracted and transcribed from video files via `/format/video`

### Processing Scope:

- Accurate speech-to-text conversion with OpenAI Whisper
- Text extraction from images using OCR
- Fact-checking via Gemini API and a custom NLP model
- Classification into Real, Fake, or Opinion

### Output Scope:

- Real-time frontend display of fact-checking results
- Detailed result cards showing method used (Gemini or Custom Model)
- Transcript visualization for audio/video inputs
- No downloadable export options (e.g., CSV/JSON) are included by design



## Out of Scope:

- Real-time live streaming analysis
  - Multilingual input support
  - Visual deepfake detection
- 

## 1.4 Motivation

The motivation behind building Fact Filter was both personal and contextually significant:

1. **Academic Exploration:** As a student in computer applications, I wanted to challenge myself by creating a system that blends AI, NLP, and software engineering principles into a working solution. This project gave me a chance to explore advanced tools like Whisper and Gemini API while applying my programming knowledge in a meaningful way.
  2. **Combatting Misinformation:** In a world where misinformation is weaponized, especially during crises, I felt compelled to develop a tool that enables ordinary users to verify digital content. Fact-checking should not be reserved for journalists or tech experts—it should be accessible to everyone.
  3. **Geopolitical Trigger:** The India-Pakistan conflict in 2025 significantly influenced my motivation. The internet was flooded with fake videos, distorted images, and misleading audio messages during the conflict, many of which incited panic. Seeing the damage misinformation can cause in real time made me realize the urgent need for an intelligent system like Fact Filter.
  4. **Innovation Opportunity:** Existing fact-checking solutions largely focus on textual inputs. I saw a unique opportunity to build a multi-modal fact-checking tool that could lead innovation in the space of digital verification—covering text, audio, image, and video.
-

## 2. System Overview

---

Fact Filter is a full-stack AI-driven web application designed to detect misinformation across multiple input types: text, image, audio, and video. The system is built with a modular client-server architecture that integrates advanced AI tools such as OpenAI Whisper, the Gemini API, and a custom NLP model. It features real-time transcription, text classification, and factual validation—all accessed through a modern React frontend.

The system also includes internal CSV-based storage of all fact-checking outputs to support future research, analysis, and fine-tuning.

---

### 2.1 System Architecture

Fact Filter follows a clear separation of responsibilities between the client and server:

- **Frontend** (React + SCSS):
  - Manages user interactions and media input.
  - Displays processing states, transcripts, and fact-check results.
  - Provides dedicated pages for each input type.
- **Backend** (FastAPI + Python):
  - Processes the uploaded input (text/image/audio/video).
  - Extracts or transcribes content to raw text.
  - Performs fact-checking using two separate engines: Gemini and a custom NLP model.
  - Stores results persistently in CSV files (gemini\_result.csv and nlp\_result.csv).

#### System Flow:

##### Input Collection

The user selects one of the four input formats: **Text**, **Image**, **Audio**, or **Video**.

##### 1. Preprocessing

- **Text** input is forwarded directly for verification.
- **Image** content is processed using OCR to extract text.
- **Audio** and **Video** files are transcribed using **OpenAI Whisper**.

##### 2. Fact Verification

- The **Gemini API** is used first for factual validation.

- The output from Gemini is stored and can optionally be passed to the NLP model for reclassification and further learning.

### 3. NLP Classification

- A trained binary classifier categorizes the content as **Real**, **Fake**, or **Opinion**.
- The classification result is stored for reference and analysis.

### 4. Result Display

- The frontend receives a structured response and displays it clearly to the user.
- 

## 2.2 Technology Stack

### Frontend

- **Framework:** React.js
- **Styling:** SCSS (modular files + \_variables.scss)
- **Routing:** React Router DOM
- **HTTP Client:** Axios
- **Deployment:** Vercel
- **Assets:** public/logo.png used in navigation branding

### Backend

- **Framework:** FastAPI
- **Transcription:** OpenAI Whisper
- **Fact-Checking:**
  - **Gemini API** for real-time factual inference
  - **Custom NLP model** (TF-IDF + classifier)
- **OCR:** Tesseract (via image.py)
- **Audio/Video Processing:** ffmpeg-python for extracting audio from video
- **Temporary Data Storage:** CSV files

### Tools and Development Environment

- Ngrok (for tunneling backend locally)
  - VS Code (IDE)
  - GitHub (source control)
-

## 2.3 Modules Description

### Backend Structure

```
app/
|   main.py
|   .env
|
+---gemini_util/
|   gemini.py
|
+---image_util/
|   image.py
|
+---nlp_util/
|   predict.py
|   preprocessing.py
|   train_model.py
|   data/
|       Fake.csv
|       True.csv
|   models/
|       fake_news_model.pkl
|       tfidf_vectorizer.pkl
|
+---prompt/
|   fact_prompt.py
|   system_prompt.md
|
+---rate_limiting_util/
|   rate_limiting.py
|
+---response_parser_util/
|   response_parser.py
|
+---storage_util/
|   storage.py
|   gemini_result.csv
|   nlp_result.csv
|
+---video_audio_util/
|   video_audio.py
|
+---whisper_util/
|   whisper.py
```

### Key Modules and Responsibilities:

- `main.py`: FastAPI entry point that wires up all routes and utilities.
- `gemini_util/gemini.py`: Sends prompts to Gemini and returns structured responses.
- `whisper_util/whisper.py`: Uses Whisper to transcribe audio/video into text.
- `video_audio_util/video_audio.py`: Extracts audio from videos for Whisper input.

- image\_util/image.py: Converts uploaded images into readable text via OCR.
  - nlp\_util/predict.py: Loads and uses a trained classification model for fake news detection.
  - nlp\_util/train\_model.py: Model training logic for future retraining.
  - prompt/fact\_prompt.py and system\_prompt.md: Constructs and formats queries for Gemini.
  - storage\_util/storage.py:
    - Writes Gemini results to gemini\_result.csv.
    - Writes NLP results to nlp\_result.csv.
    - These CSV files act as a **record** of all user interactions and outputs.
    - The data is intended to be used **later for NLP model retraining** and understanding public search trends.
- 

## Frontend Structure

```
src/
|   App.jsx
|   main.jsx
|   index.scss
|
+---components/
|   FactCheckResult.jsx
|   FilePreview.jsx
|   FileUpload.jsx
|   Footer.jsx
|   LoadingPopup.jsx
|   MailScript.jsx
|   NavBar.jsx
|   SourceSelector.jsx
|   TranscriptDisplay.jsx
|
+---hook/
|   UseFactCheck.jsx
|
+---pages/
|   Home.jsx
|   About.jsx
|   Contact.jsx
|   formats/
|       TextFormat.jsx
|       ImageFormat.jsx
|       AudioFormat.jsx
|       VideoFormat.jsx
|
+---styles/
|   _variables.scss
|   About.scss
|   Contact.scss
|   FactCheckResult.scss
|   FilePreview.scss
|   FileUpload.scss
```

- | Footer.scss
- | FormatPage.scss
- | Home.scss
- | LoadingPopup.scss
- | NavBar.scss
- | SourceSelector.scss
- | TextFormat.scss
- | TranscriptDisplay.scss

#### **Frontend Responsibilities:**

- App.jsx and main.jsx: Initialize routing and component tree.
- NavBar.jsx, Footer.jsx: Layout components used across all pages.
- FileUpload.jsx, FilePreview.jsx: Handle input, preview, and upload flow.
- FactCheckResult.jsx: Displays classification result and status.
- TranscriptDisplay.jsx: Shows Whisper transcript from audio/video.
- SourceSelector.jsx: Lets user choose between Gemini or NLP source.
- LoadingPopup.jsx: Full-screen loading animation.
- MailScript.jsx: Sends contact form data via EmailJS.
- UseFactCheck.jsx: Custom hook that communicates with the backend and manages upload, state, and errors.

#### **Format Pages:**

Each route under /pages/formats corresponds to a supported input type:

- TextFormat.jsx: Accepts and processes text.
- ImageFormat.jsx: Uploads images and shows OCR-extracted text.
- AudioFormat.jsx: Uploads audio and runs Whisper transcription.
- VideoFormat.jsx: Uploads videos and extracts + transcribes audio.

#### **Styling:**

All SCSS files are modularized and match their respective components. Global styles are defined in index.scss and shared variables like colors and spacing are in \_variables.scss.

#### **Static Assets:**

- public/logo.png: Branding image shown in the navigation bar.

# 3. Project Context and Problem Landscape

---

The Fact Filter project was born out of direct observation and personal experimentation rather than a formal literature review. It addresses the growing concern of misinformation, particularly in multimedia formats like video and audio, which are often overlooked by traditional fact-checking solutions. This section outlines the practical context that shaped the need for this project, what existing tools lack, and the innovative direction taken by Fact Filter.

---

## 3.1 Observations from the Digital Ecosystem

Over the past few years, the internet has become the dominant channel for consuming news and information. Social media platforms, YouTube, WhatsApp forwards, and video blogs now play a larger role than traditional news outlets. However, this shift has made it easier for misinformation to spread rapidly—often without fact-checking or moderation.

One of the most alarming observations is the increasing reliance on **video and audio-based content** to convey emotionally charged or sensational messages. During events like political elections, protests, or geopolitical tensions (such as the **India-Pakistan conflict of 2025**), false narratives disguised as news clips, speeches, or interviews are shared at scale. Unlike plain text, such formats are harder for both people and machines to verify quickly.

This environment highlighted the urgent need for a system that doesn't just rely on text-based claims but is capable of understanding and verifying information presented in more complex formats.

---

## 3.2 Gaps in Current Tools (Based on Personal Exploration)

While exploring existing fact-checking platforms and browser tools, it became apparent that most systems were not built with multimedia in mind. Tools like Google's Fact Check Explorer, Snopes, or InVID serve well for manual verification, but they have major limitations:

- They **do not support file uploads** for video or audio verification.
- Most platforms require users to **manually extract and submit text**.
- No tool tested offered **real-time feedback** for content uploaded by users.

- There was **no end-to-end automation** from speech recognition to classification.
- Many services are **not user-friendly for non-technical individuals** and are meant for journalists or researchers.

Even tools with some media-related features lacked integration with AI models that could process and evaluate claims programmatically.

These gaps became the starting point for building something more comprehensive and automated—Fact Filter.

---

### 3.3 Why Multi-Format Fact-Checking Is Necessary Today

The formats in which misinformation spreads are no longer limited to text. Voice messages, news videos, image posts with embedded captions, and even edited reels contribute to widespread confusion. In modern digital communication, users encounter:

- **Videos** with edited speeches, false subtitles, or misleading commentary.
- **Audio clips** with spliced statements or impersonated voices.
- **Images** with photoshopped content or misleading captions.
- **Text** taken out of context and shared as fact.

Any platform that intends to be useful for real-world misinformation detection must be capable of handling **multi-modal input**—processing and understanding data regardless of its source type.

Fact-checking tools must therefore:

- Support speech-to-text for audio/video,
- Use OCR for images,
- Handle long and short text input,
- Provide interpretable output regardless of input format.

This realization guided the structure and design of Fact Filter, ensuring each media type had a clear, isolated path for validation.

---

### 3.4 The Unique Approach of Fact Filter

Fact Filter introduces a new direction in AI-based misinformation detection by combining multiple technologies in a seamless, real-time pipeline:

- **Multi-format support:** Accepts and processes four distinct input types—**text, image, audio, and video**—each routed to its own dedicated page (e.g., /format/video).



- **Speech transcription:** Uses **OpenAI Whisper** to convert audio and video speech into accurate text for analysis.
  - **Factual analysis:** Integrates with **Gemini API** to perform contextual fact validation from the transcript.
  - **Classification:** Passes the cleaned text into a **custom NLP model** trained on real/fake news datasets to categorize input as *Real*, *Fake*, or *Opinion*.
  - **Data logging:** Stores results from Gemini and the NLP model in `gemini_result.csv` and `nlp_result.csv`, creating a valuable dataset for:
    - Trend analysis (what people are fact-checking),
    - Improving NLP models in the future,
    - Research into misinformation patterns.
-

## 4. Methodology

---

This section outlines the structured approach followed in the design and development of Fact Filter. It covers the chosen software development life cycle, the tools and technologies applied across the stack, and a detailed explanation of the core AI models powering the system—including Whisper, Gemini API, and the custom-trained NLP model.

---

### 4.1 Software Development Life Cycle (SDLC) Used

For this project, an **Iterative SDLC model** was followed. Since the project involved continuous experimentation with APIs, model accuracy, media handling, and frontend interaction, the iterative approach allowed changes and enhancements at each cycle without starting from scratch.

Each iteration followed this flow:

1. **Requirement Analysis** - Identifying essential features: format-specific uploads, real-time result, Whisper integration, and user-friendly frontend.
2. **Design** - Planning the modular backend (whisper\_util, gemini\_util, etc.) and component-driven frontend (TextFormat, AudioFormat, etc.).
3. **Implementation** - Building and testing one feature (e.g., video upload) at a time, integrating it into the main system.
4. **Testing & Refinement** - Manually testing inputs, debugging errors (e.g., Whisper failure cases, file type mismatches), and optimizing response structure.
5. **Deployment** - Hosting frontend on Vercel, backend temporarily exposed via Ngrok.

This process was repeated separately for:

- Each input format (text, image, audio, video),
  - Both verification methods (Gemini and NLP),
  - Frontend features (upload, preview, loading animation, result display).
- 

### 4.2 Tools and Technologies

#### Frontend (Client-side)

- **React.js**: Component-based architecture, routing, and state management.
- **SCSS**: Modular and maintainable styling.

- **Axios:** For API requests to the backend.
- **EmailJS:** Contact form integration for user outreach.
- **Vercel:** Frontend deployment platform.
- **Google Maps Embed API:** Used in the Contact page.

### Backend (Server-side)

- **FastAPI:** Lightweight and fast Python web framework used to handle all HTTP requests.
  - **Python 3.10+**
  - **Ngrok:** Used to expose local backend during development for testing with frontend.
  - **OpenAI Whisper:** For speech-to-text transcription from audio and video.
  - **Gemini API:** For real-time fact validation using LLMs.
  - **Tesseract OCR:** Used for image-to-text extraction.
  - **Scikit-learn:** For building the custom NLP model.
  - **Pandas & CSV:** For storing outputs (gemini\_result.csv, nlp\_result.csv) and processing datasets (Fake.csv, True.csv).
- 

## 4.3 Model/Algorithm Used

### A. OpenAI Whisper (Speech-to-Text)

- Used to transcribe audio and video input.
- Provides multilingual transcription with strong accuracy.
- Output is plain text used as input for both Gemini and NLP verification pipelines.

### B. Gemini API (Factual Verification)

- Processes the transcript (or OCR text) using a system prompt.
- Returns a structured output including:
  - **Status:** Real, Fake, or Opinion
  - **Description:** Contextual reasoning
  - **Method Used:** Gemini
- Prompts are dynamically built using fact\_prompt.py + system\_prompt.md.

### C. Custom NLP Model (Text Classification using Logistic Regression)

The NLP component of Fact Filter is a custom-built binary text classification pipeline that determines whether a given statement is **Real** or **Fake**. This model is implemented using **Logistic Regression**, which is well-suited for binary classification problems, especially with sparse high-dimensional text features.

The model is trained using a labeled dataset (Fake.csv, True.csv), and predictions are made on the cleaned and vectorized transcript extracted from audio, video, image, or user-submitted text.

---

## Model Pipeline and Workflow

### 1. Preprocessing

The text undergoes extensive cleaning before vectorization.

Implemented in preprocessing.py, the cleaning steps include:

- Lowercasing all characters
- Removing punctuation and special characters
- Removing numeric words and tokens
- Removing bracketed content

```
def clean_text(text: str) -> str:
    text = text.lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub(r'\w*\d\w*', '', text)
    return text
```

### 2. TF-IDF Vectorization

In train\_model.py, after cleaning, the dataset is vectorized using the **TF-IDF (Term Frequency-Inverse Document Frequency)** technique, which converts the text into numerical form based on term importance.

- Configured with:
  - stop\_words='english'
  - ngram\_range=(1, 2) (unigrams + bigrams)
  - max\_df=0.9 to ignore overly frequent terms

### 3. Model Training (Logistic Regression)

The core classification model is implemented using LogisticRegression from sklearn.linear\_model. It's trained to differentiate between real (label = 1) and fake (label = 0) news headlines or statements.

- Trained on a stratified 80/20 train-test split
- Evaluated using accuracy\_score, confusion\_matrix, and classification\_report
- Achieved solid classification performance with balanced results

### 4. Saving Trained Artifacts

After training, the model and vectorizer are saved using joblib to the models/ directory:

- fake\_news\_model.pkl: Serialized Logistic Regression model
- tfidf\_vectorizer.pkl: Trained vectorizer used during prediction

---

## Prediction Logic

The prediction logic resides in predict.py:

### 1. Model and Vectorizer Loading

- The files fake\_news\_model.pkl and tfidf\_vectorizer.pkl are lazily loaded and cached globally to avoid reloading on every request.

## 2. Inference Process

- Input text is first passed to `clean_text()`
- The cleaned text is vectorized using the TF-IDF vectorizer
- The Logistic Regression model predicts a binary label
- The label is mapped to:
  - "Fake" if label = 0
  - "Real" if label = 1

## 3. Usage Example from API (in `main.py`):

```
result = predict_news(transcript) # returns "Real" or "Fake"  
method = "NLP Model"
```

---

## CSV Logging and Data Usage

All NLP predictions are logged automatically in `nlp_result.csv` via `log_result()` in `storage.py`. Each entry includes:

- Timestamp
- User IP
- Input type (text, image, audio, video)
- Method used (NLP Model)
- Extracted/processed text
- Final prediction result

This data is:

- Valuable for understanding **user behavior** (what people are fact-checking)
- Useful for **retraining or improving** the model with real-world inputs
- Helpful for **monitoring system performance** across formats and sources

## D. Result Logging

- All Gemini results are stored in `gemini_result.csv`.
  - All NLP classification results are stored in `nlp_result.csv`.
  - These logs allow future model improvement and trend analysis on what types of claims users are fact-checking.
-

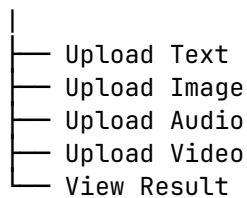
# 5. System Design

The system design of Fact Filter is structured around a modular, format-based processing pipeline. The design follows a layered architecture with clearly separated components for input handling, AI processing, and frontend display. This section includes diagrams and overviews that explain how data flows through the system and how major components interact.

## 5.1 UML Diagrams

### Use Case Diagram

[User]



### Class Diagram (Simplified View)

Class: FactChecker

- method: "Gemini" | "NLP"  
+ check\_facts(transcript)  
+ return\_result()

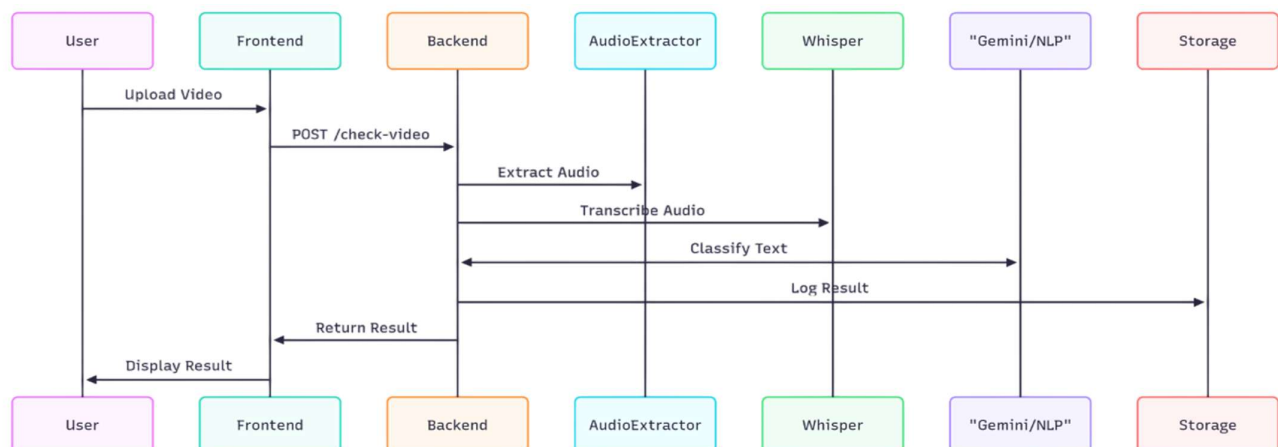
Class: FileHandler

+ extract\_audio(video)  
+ transcribe\_audio(audio)  
+ ocr\_image(image)  
+ get\_text(input)

Class: StorageLogger

+ log\_to\_csv(result, method, source)

### Sequence Diagram (For Video Input)



## 5.2 Database Design

Not Applicable.

Fact Filter does **not** use a database. Instead:

- Results are stored in `gemini_result.csv` and `nlp_result.csv`
  - These files act as logs for future model training and trend analysis
- 

## 5.3 UI/UX Design Overview

The frontend is clean, responsive, and format-focused:

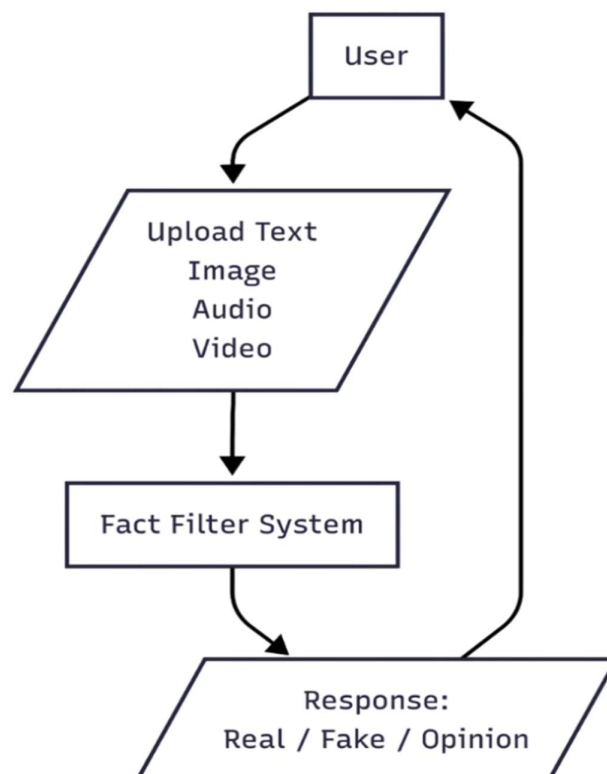
- **Navigation:** Persistent header with routes to `/format/text`, `/format/image`, `/format/audio`, `/format/video`
- **Upload Section:** File input and preview for each format
- **Model Selector:** Toggle between Gemini and NLP
- **Loading Popup:** Displays while the backend processes the input
- **Result Section:** Displays verdict, transcript, and method used
- **About & Contact Pages:** Include motivation, form, and embedded map

Color theme and spacing are controlled using SCSS variables (`_variables.scss`) for consistency and responsiveness.

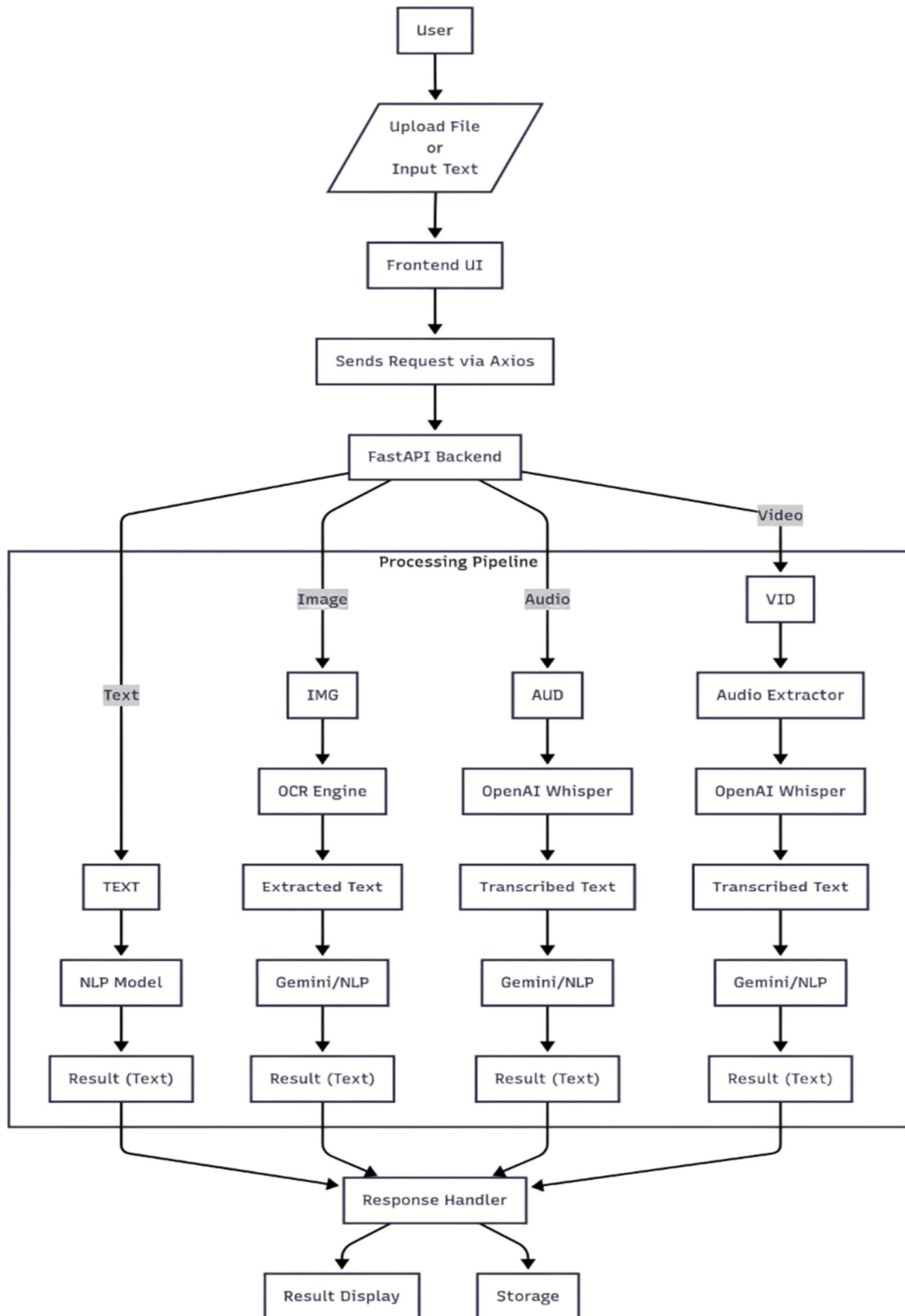
---

## 5.1 Data Flow Diagrams (DFD)

Level 0: Context-Level DFD



## Level 1: Process-Level DFD





# 6. Implementation

---

The implementation of Fact Filter involved full-stack development, including frontend construction using React and SCSS, backend logic using FastAPI, integration of AI tools (Whisper, Gemini, NLP), and temporary result storage using CSV files. Each input format (text, image, audio, video) is handled independently through dedicated pages and logic flows.

---

## 6.1 Frontend Overview

The frontend is a React-based single-page application deployed via Vercel. The folder structure follows a modular design pattern with components, pages, and styles separated by role.

### Key Directories and Files:

- App.jsx, main.jsx: Route initialization and component tree setup
- components/: Reusable elements (e.g., FileUpload.jsx, ResultCard.jsx)
- pages/: Main pages like Home.jsx, About.jsx, and input format routes
- pages/formats/: Includes TextFormat.jsx, ImageFormat.jsx, AudioFormat.jsx, VideoFormat.jsx
- styles/: SCSS files for all components and pages with \_variables.scss for themes
- public/logo.png: Project branding used in the navbar

Each input format page:

- Uses FileUpload to accept user input
- Calls the UseFactCheck hook to process input
- Displays the result using FactCheckResult and TranscriptDisplay

The frontend communicates with the backend using **Axios** and displays loading animations (LoadingPopup) and error handling messages when needed.

---

## 6.2 Backend API Endpoints

The backend is developed using **FastAPI** with modular utilities for each processing stage. The primary routes (defined in main.py) include:

- POST /fact-check/text: Accepts plain text input
- POST /fact-check/image: Accepts image files, uses OCR
- POST /fact-check/audio: Accepts audio files, transcribes with Whisper
- POST /fact-check/video: Accepts video files, extracts audio, transcribes, then classifies

Each route follows a standard flow:

1. Preprocess input (OCR, Whisper, etc.)
  2. Choose model (Gemini or NLP)
  3. Classify and return result
  4. Log to CSV
- 

## 6.3 Fact-Checking Logic

Fact checking is executed via two models:

- **Gemini API** (in `gemini_util/gemini.py`):
  - Sends transcript with a formatted prompt
  - Returns structured output with status and reasoning
  - Method = "Gemini"
- **Custom NLP Model** (in `nlp_util/predict.py`):
  - Loads pre-trained Logistic Regression model and TF-IDF vectorizer
  - Classifies as Fake or Real
  - Method = "NLP"

The chosen model is determined via user toggle (SourceSelector in frontend).

---

## 6.4 Video Upload and Transcription

The video pipeline includes:

1. **Video Upload** via frontend (`VideoFormat.jsx`)
2. **Audio Extraction** using FFmpeg via `video_audio_util/video_audio.py`
3. **Speech-to-Text** via Whisper (`whisper_util/whisper.py`)
4. **Text Classification** using Gemini or NLP
5. **Result Response** to frontend with classification and transcript

Audio-only uploads skip Step 2.

---

## 6.5 Report Generation and Export

- **Real-Time Output Display:** Results are shown on the frontend immediately after processing
  - **CSV Logging:** No downloadable report, but all responses are logged in:
    - `gemini_result.csv`: For Gemini results
    - `nlp_result.csv`: For NLP results.
-

# 7. Testing

Testing was essential to verify the system's correctness, stability, and performance across multiple input formats. The process focused on frontend behavior, backend API logic, and the accuracy of fact-checking results generated by both Gemini and the NLP model.

## 7.1 Testing Methodology

A **manual black-box testing** approach was used throughout development. Each feature was tested from a user's perspective, simulating real-world interactions across all input types: text, image, audio, and video.

### Frontend Testing:

- Verified file uploads, form validations, loading animations, and result rendering.
- Checked routing for each input format page and overall responsiveness.
- Tested error states such as missing input or unsupported file types.

### Backend Testing:

- Used Postman and browser console tools to test API endpoints.
- Sent various valid and invalid file types to check error handling.
- Evaluated transcription, OCR, Gemini responses, and NLP predictions.

### Model Testing:

- Manually tested Real and Fake statements using both Gemini and NLP.
- Reviewed NLP accuracy by comparing predictions with known labels.
- Tested edge cases where content was ambiguous or borderline.

## 7.2 Test Cases and Results

Below are sample test cases and their outcomes:

Test ID	Input Type	Example Input	Expected Result	Outcome
T1	Text	"India won the 2023 Cricket World Cup"	Classified as Real	Passed
T2	Text	"NASA says the moon is made of cheese"	Classified as Fake	Passed
T3	Image	Screenshot of real news headline	OCR text + Real label	Passed
T4	Audio	News podcast audio clip	Transcription + Real	Passed
T5	Video	Fake news speech clip	Transcription + Fake	Passed

## 8. Result and Analysis

---

This section presents the outcomes of the Fact Filter system after successful implementation and testing. It includes screenshots from various input formats, observations on the system's behavior, performance analysis of different modules, and insights into the accuracy of fact-checking results generated by Gemini and the custom NLP model.

---


### 8.1 Output Screenshots

The frontend interface of Fact Filter delivers a seamless and informative user experience. Below are descriptions of key output states (actual screenshots to be inserted in your final document):

- **Text Format Page:** Displays input text area, source selector (Gemini or NLP), and result display with verdict (Real/Fake/Opinion).

### Paste Text for Fact Checking

Top news of the day: BJP promises to implement uniform civil code and NRC in Karnataka poll manifesto; TN CM Stalin announces withdrawal of controversial amendment to Factories Act.



Select Source:

Gemini (Default) ▾

Discover What's Really True

Real

Both events were widely reported in Indian news media during the relevant timeframe. The BJP's Karnataka manifesto and the TN CM's announcement on the Factories Act are matters of public record.

Verified using: Gemini

- **Image Format Page:** Shows uploaded image preview, extracted OCR text, and result classification.

### Upload an Image for Fact Checking

File received:

**PREDICTION**

POSSIBILITIES IN 2023

उत्तर भारत को 2023 की दूसरी छमाही तक मिलेगी राहत, पंजाब, हरियाणा में टिम्बर की सप्लाई बढ़ेगी

Select Source:

Gemini (Default)

Discover What's Really True

**Transcription:**

POSSIBILITIES IN 2023 उत्तर भारत में 2023 की दूसरी छमाही तक मिलेगी राहत, पंजाब, हरियाणा में टिम्बर की सप्लाई बढ़ेगी

**Opinion**

The statement predicts future economic conditions (timber supply increase in Punjab and Haryana) and relief in North India by the second half of 2023. Such predictions are speculative and not verifiable as fact.

Verified using: Gemini

- **Audio Format Page:** Displays file preview, Whisper-generated transcript, and fact-check output.

### Upload an Audio File for Fact Checking

File received:

0:00 / 0:43

Select Source:

Gemini (Default)

Discover What's Really True

**Transcription:**

Akashwani presents Morning News. Liquid ramjet fuel for an advanced air-breathing engine. And Hockey India announces 24-member men's team for FIH Hockey Pro League 2023-24.

**Real**


Both Akashwani (All India Radio) and Hockey India are legitimate news sources. The announcements of liquid ramjet fuel development and the men's hockey team are plausible and verifiable through their respective official channels.

Verified using: Gemini

- **Video Format Page:** Includes video preview, extracted transcript, and result verdict.

## Upload a Video for Fact Checking

✔ File received:



Select Source:

Gemini (Default) ▼

Discover What's Really True

### Transcription:

Akashwani presents Morning News.  
 Liquid ramjet fuel for an  
 advanced air-breathing engine.  
 And Hockey India announces 24-  
 member men's team for FIH Hockey  
 Pro League 2023-24.

Real

Both Akashwani (All India Radio) and Hockey India are established organizations. News reports from 2023-24 confirm announcements regarding a men's hockey team and developments in liquid ramjet technology.

Verified using: Gemini

Each format page renders:

- The method used (Gemini or NLP)
- A transcript (for audio/video inputs)
- A clear label for the classification result

---

## 8.2 Performance Evaluation

Fact Filter was evaluated in terms of speed, robustness, and responsiveness:

Component	Performance Observation
Text Input	Fast classification (<1-3 sec) with both Gemini and NLP
Image Input	Slight delay due to OCR (~4-8 sec), depends on image clarity
Audio Input	Average response time 8-20 sec depending on duration
Video Input	Slower due to audio extraction + Whisper (~15-50 sec)
Frontend	Fully responsive, tested on desktop and mobile browsers
Whisper	Highly accurate, but performance depends on audio quality
Gemini	Accurate with detailed output, but rate-limited
NLP Model	Consistent for simple factual claims, fast classification

All operations return results in real-time (within a few seconds) and handle invalid or empty inputs gracefully.

---

## 8.3 Accuracy of Fact Checking

### Gemini API:

- Provides highly contextual, descriptive responses.
- Suitable for real-world news statements and longer transcripts.
- May occasionally misclassify ambiguous content or opinion-based statements as Real or Fake.

### Custom NLP Model:

- Trained using a labeled dataset with Logistic Regression.
- Performs well for short claims or headline-style inputs.
- Classification into Fake or Real is direct and fast.

### Overall Observations:

- Gemini is more accurate for complex, nuanced claims but slower and dependent on API limits.
  - NLP model is fast and reliable for clear-cut statements but limited in reasoning.
  - Combining both models gives users a choice between **speed (NLP)** and **depth (Gemini)**.
-

## 9. Conclusion and Future Scope

---

This section summarizes the achievements of the Fact Filter project, outlines its current limitations, and proposes meaningful enhancements for future development.

---

### 9.1 Conclusion

Fact Filter successfully demonstrates an AI-powered solution for detecting misinformation across multiple media formats. The system integrates modern technologies such as OpenAI Whisper, Gemini API, and a custom NLP model to deliver real-time classification of user-submitted content as **Real**, **Fake**, or **Opinion**.

Unlike traditional tools that only process text, Fact Filter supports a full range of input types—**text, images, audio, and video**—each with its own dedicated upload and processing path. The frontend is user-friendly and responsive, while the backend is modular and designed to handle complex multimedia tasks like speech transcription, OCR, and classification.

The project also includes a lightweight storage mechanism via CSV files for logging fact-checking results, which can be useful for future model improvements and public misinformation trend analysis.

As a solo developer, completing this end-to-end system offered an in-depth understanding of full-stack AI integration, file handling, NLP, and modern frontend development.

---

### 9.2 Limitations

Despite its functionality, the current version of Fact Filter has some limitations:

- **No real-time video streaming:** Only pre-recorded files are supported.
  - **Monolingual support:** The system primarily handles English inputs.
  - **No user authentication or access control**
  - **No download or export feature** for end-users
  - **Relies on third-party services** (Gemini API, Whisper) which may have usage limits or latency
  - **Limited deepfake detection:** System does not evaluate visual manipulation (faces/images)
-



### 9.3 Future Enhancements

The following improvements are planned or recommended for future versions of Fact Filter:

- **Add multilingual support** for Hindi, regional languages, or global languages
  - **Integrate deepfake detection** for video/image-based visual manipulation
  - **Implement user accounts** for personalized history and saved fact-checks
  - **Build a full dashboard** for visualizing logged data and trends (from `gemini_result.csv` and `nlp_result.csv`)
  - **Enable model retraining pipeline** using logged real-world inputs
  - **Improve opinion detection** by integrating sentiment analysis or intent classification
  - **Deploy a secure backend** with persistent storage (database, S3) and API rate controls
- 

Fact Filter lays the foundation for a multi-format misinformation detection tool that is fast, extensible, and practical in today's media-heavy environment. With continued development, it has the potential to serve as a real-world product for journalists, educators, and the public.

---

# 10. Appendix

---

This appendix provides additional technical references, deployment resources, and support materials related to the Fact Filter project.

---

## 10.1 Source Code Links

The complete source code for both frontend and backend is available in separate GitHub repositories:

- **Frontend (React + SCSS)**  
<https://github.com/rayson2/fact-filter>
- **Backend (FastAPI + ML Models)**  
<https://github.com/rayson2/fact-filter-backend>

*(Replace these links with your actual GitHub URLs if different)*

---

## 10.2 Deployment Links

The live deployed version of the Fact Filter application is hosted on Vercel:

- **Main Website:**  
<https://fact-filter.vercel.app>
- **Format-Specific Pages:**
  - /format/text - Text Input
  - /format/image - Image Upload (OCR)
  - /format/audio - Audio Upload
  - /format/video - Video Upload

The backend is exposed during testing via Ngrok (not persistently deployed).

---

## 10.3 Dataset and Model Information

### Training Data Source:

- The dataset used to train the custom NLP model was downloaded from Kaggle:
  - **Kaggle Dataset URL:**  
<https://www.kaggle.com/datasets/emineyettm/fake-news-detection-datasets>
- It includes two CSV files:
  - Fake.csv: Contains fake news samples

- True.csv: Contains real news samples
- These files were used to train the Logistic Regression model for binary classification.

#### **Model Artifacts:**

- fake\_news\_model.pkl: Trained Logistic Regression model for binary classification
- tfidf\_vectorizer.pkl: TF-IDF vectorizer used to convert text to numerical features
- Stored under: nlp\_util/models/

#### **Log Files:**

- gemini\_result.csv: Stores outputs from Gemini API
- nlp\_result.csv: Stores predictions made by the custom NLP model
- These files are saved in storage\_util/ and used for future model improvement and trend analysis

---

## **10.4 User Manual / Installation Guide**

#### **Frontend Setup:**

```
cd fact-filter
npm install
npm run dev
```

#### **Backend Setup:**

```
cd fact-filter-backend
pip install -r requirements.txt
uvicorn app.main:app --reload
```

#### **Dependencies:**

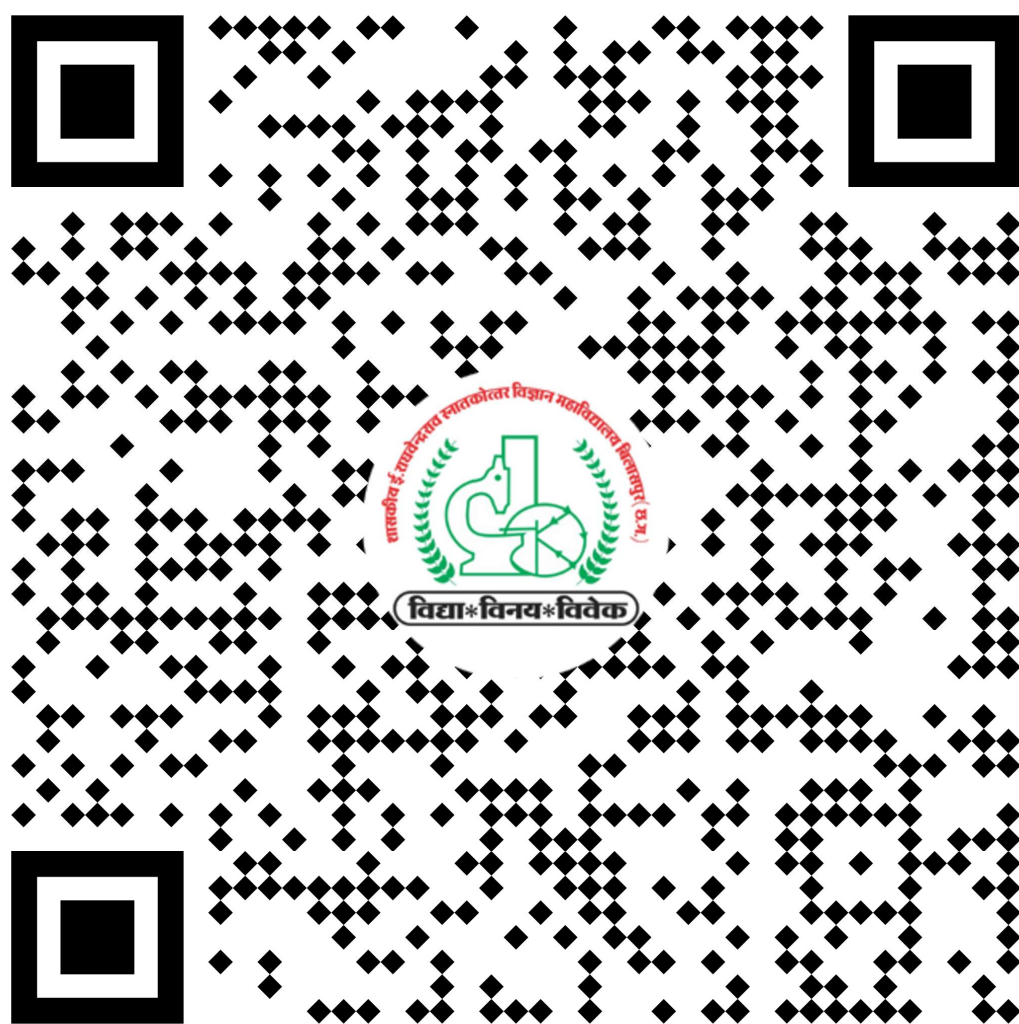
- React, SCSS, Axios (Frontend)
- FastAPI, Whisper, Tesseract, Scikit-learn, Pandas (Backend)

#### **Notes:**

- Whisper requires Python  $\geq 3.9$  and a compatible FFmpeg setup.
- Ngrok can be used to tunnel the backend (ngrok http 8000).
- Make sure .env files are properly configured with API keys and path settings.

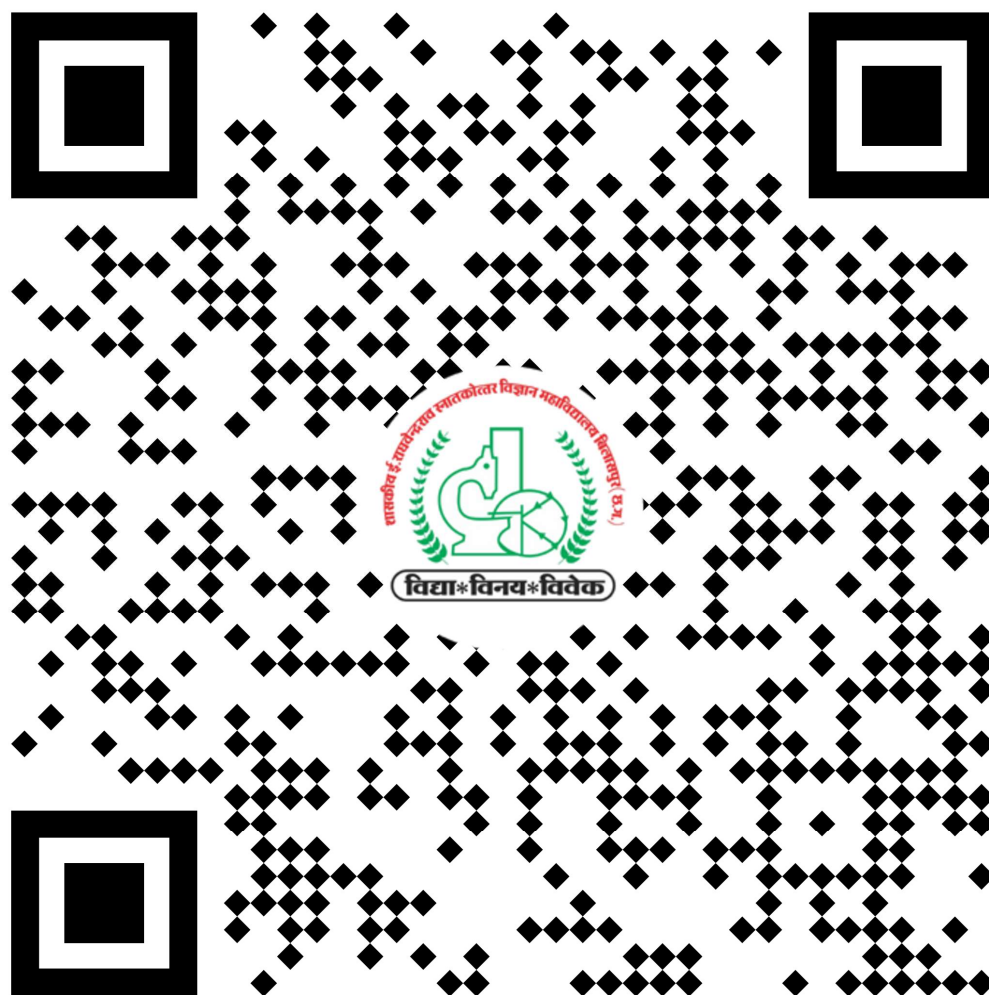
# DOCUMENTATION-QR

---



# FRONTEND-QR

---



# BACKEND-QR

---

