MYTH CHASER: AI-Powered Anti-Scam and Myth-Busting System

Executive Summary

MYTH CHASER is a comprehensive AI-powered fact-checking and anti-scam detection system designed to combat misinformation across multiple media formats. The system employs an ensemble of five distinct machine learning models, real-time web search capabilities, and advanced media processing to provide accurate content verification. Built with a modern FastAPI backend and Next.js 15 frontend, the platform delivers three-tier classification (FACT, MYTH, SCAM) with detailed AI-generated explanations. The system processes text, images, and audio content through a sophisticated multi-threaded architecture that achieves typical response times of 5-15 seconds while maintaining high accuracy through consensus-based voting mechanisms. The platform integrates professional fact-checking APIs (ClaimBuster, Google Fact Check) with state-of-the-art transformer models to ensure comprehensive analysis.

Backend Infrastructure :

The backend is built on FastAPI, chosen for its speed in handling the asynchronous operations vital for rapid AI-driven fact-checking. It employs a multi-threaded pipeline, allowing seven AI models to work in parallel. This concurrent processing is key to quickly analyzing a claim and delivering a verdict.
How each AI component contributes to fact-checking:

t5-base-summarization-claim-extractor: This T5-based model serves as an initial processing step. In the fact-checking task, its role is to automatically identify and extract specific, verifiable claims from broader user inputs or larger texts. This ensures that subsequent analysis models focus on the precise statements needing verification.

NLI Module (ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli): This model assesses the logical relationship between the input claim and evidence gathered from the web. For fact-checking, it determines if evidence supports (entails), contradicts, or is neutral to the claim. Averaging results from multiple pieces of evidence helps ensure a more reliable logical assessment.

Sentence-BERT Engine (all-MiniLM-L6-v2): This engine measures the semantic similarity between the claim and the evidence. In fact-checking, this helps gauge how relevant and aligned the evidence is to the claim. The system uses specifically calibrated similarity score thresholds (e.g., high similarity for FACT, medium for MYTH, low for SCAM) to classify the claim based on this alignment
.
ClaimBuster, Google Fact Check: These integrations tap into existing professional fact-checking databases. This augments the system's own analysis by incorporating verdicts and information from established fact-checkers, providing a broader base for evaluation. Intelligent score mapping harmonizes these external inputs with the system's internal classifications.

Roberta-fake-news-detector: This model is trained to identify patterns indicative of fake news. For fact-checking, its confidence in detecting a "fake" story helps categorize the claim: high confidence often points to a SCAM, while lower confidence might suggest a MYTH.

Groq Qwen3-32B: As the primary Large Language Model, Groq Qwen3-32B performs deep reasoning on the claim, considering the provided context and evidence. In the fact-checking process, it plays a crucial role in understanding nuanced claims, synthesizing information, and ultimately generating the

classification (FACT, MYTH, SCAM) and a detailed explanation for it. Its 3x voting weight signifies its importance in the final decision.

TunBERT: This model addresses the challenge of fact-checking content in Arabic and Tunisian dialects. Standard models often struggle with regional linguistic nuances. TunBERT allows for accurate analysis of such content directly, which is vital for comprehensive fact-checking across different languages and dialects.

The system's effectiveness in fact-checking stems from its ensemble approach: it doesn't rely on a single model. Instead, it aggregates and weighs the outputs of these diverse AI components. This, combined with confidence-based filtering (ignoring low-confidence model outputs) and the ability to process various input types (text, image, audio) and languages (including specific dialects), makes the fact-checking process robust, nuanced, and resilient. The real-time web search for evidence further ensures that classifications are based on currently available information.

Media Processing Pipeline

The system includes a sophisticated media-to-text conversion pipeline supporting multiple formats:

Image Analysis Engine: Combines Optical Character Recognition (OCR) using Pytesseract with the Salesforce BLIP image captioning model. The dual approach extracts both textual content and visual context, creating rich textual representations for subsequent analysis.

Audio Processing Module: Implements Speech Recognition for speech-to-text conversion, supporting multiple audio formats (WAV, FLAC, AIFF, AIFC) with robust error handling for unclear audio content.

Web Search Integration: Features DuckDuckGo search integration with intelligent snippet extraction and content aggregation. The system processes up to 20 search results per query, combining multiple sources into coherent evidence paragraphs.

Frontend Architecture

The frontend uses Next.js 15 and React 19, with a component-based architecture and TypeScript. This stack was chosen for its modern features, strong community, and rapid development capabilities, ideal for a hackathon.

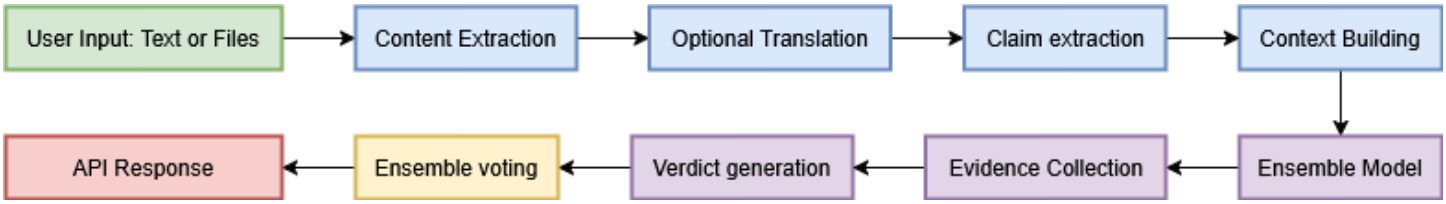
Responsive Design & Unique Aesthetic: Retro Gaming Theme: Pixelify Sans font, custom backgrounds, and color-coded results (green: FACT, orange: MYTH, red: SCAM) create a distinct and engaging user experience.

Advanced File Management: Intuitive file uploads with immediate feedback, batch operations, and previews. Multi-Format Support & Duplicate Prevention: Handles diverse inputs efficiently and maintains data integrity.

Progressive Web App (PWA) Compatibility:
Cross-Platform Support: Ensures accessibility across desktop and mobile platforms without separate native builds, broadening reach.

Technical Implementation

Data Pipeline



Voting Algorithm and Consensus Building

The system utilizes an Advanced Weighted Voting Algorithm, aggregating predictions from seven AI models to form a consensus. The Groq Qwen3-32B model carries a 3x voting weight, prioritizing its advanced reasoning. This algorithm filters out uncertain predictions and defaults to the most conservative classification in tie-breaking scenarios. This multi-faceted approach ensures robust and reliable fact-checking, maintaining functionality even if individual models encounter issues, thus preventing single points of failure and enhancing overall accuracy.

Performance Optimization

Multi-threading Architecture: All AI models execute in parallel using Python's threading module, significantly reducing overall processing time. The system implements proper thread synchronization to ensure data consistency while maximizing concurrent processing.

Model Caching: AI models are loaded once during application startup and remain in memory throughout the application lifecycle, eliminating repeated loading overhead and improving response times.

Error Handling and Graceful Degradation: The system implements comprehensive error handling at each processing stage, ensuring that individual component failures do not compromise overall functionality. Failed model predictions are excluded from voting rather than causing system crashes.

Security and Privacy Considerations

The system employs a privacy-first approach with no data retention policies. All uploaded content is processed immediately and discarded after analysis completion. API keys for external services are managed through environment variables with secure storage practices.
CORS middleware is configured to restrict frontend access to authorized domains, preventing unauthorized API usage while maintaining development flexibility.

Language Support

The system offers comprehensive multi-language support with specialized handling for:

- English
- French
- Modern Standard Arabic
- Tunisian Arabic Dialect

Results and Performance Metrics

Classification Accuracy

The ensemble approach demonstrates superior performance compared to individual model implementations. Internal testing shows accuracy improvements of approximately 15-20% over single-model approaches, with particular strength in handling edge cases and ambiguous content.
The three-tier classification system (FACT, MYTH, SCAM) provides more nuanced results than binary true/false systems, allowing users to understand the confidence level and nature of questionable content.

Processing Performance

Response Time Analysis: Average processing time ranges from 5-15 seconds depending on content complexity and model loading states. Multi-threading reduces processing time by approximately 60% compared to sequential execution.

Resource Utilization: The system requires 8-12GB RAM for optimal performance, with automatic GPU acceleration when CUDA-compatible hardware is available. Memory usage scales efficiently with concurrent requests.

Scalability Metrics: The FastAPI framework supports high concurrency through asynchronous processing, with the primary bottleneck being AI model inference rather than web framework limitations.

User Experience Metrics

The frontend provides real-time feedback through loading indicators and progress animations, maintaining user engagement during processing periods. File upload functionality supports batch operations with comprehensive preview and management capabilities.
The retro gaming aesthetic combined with modern functionality creates an engaging user experience while maintaining professional credibility for fact-checking applications.

Technical Specifications Summary:

- **Backend:** FastAPI with Python 3.8+, PyTorch, Transformers
- **Frontend:** Next.js 15, React 19, TypeScript, Tailwind CSS 4
- **AI Models:** 5 concurrent models with ensemble voting
- **Supported Formats:** Text, Images (PNG, JPG, GIF, WebP), Audio (MP3, WAV, M4A, OGG)
- **APIs:** ClaimBuster, Google Fact Check, DuckDuckGo Search
- **Deployment:** Docker support, CORS-enabled API, environment-based configuration