Project Technical Report

a. Problem Context and Project Summary:

In an era where misinformation spreads rapidly across social media and news platforms, distinguishing verified facts from misleading claims has become a pressing global challenge. This project, "**ClaimVerify: Misinformation Detection System with Fact-Check Retrieval***,"* aims to build an Intelligent system that automatically analyzes user-submitted claims, determines their likely veracity, and retrieves the most relevant verified fact-checks to support its judgment. The solution combines a fine-tuned transformer model for claim classification with a dual-stage retrieval pipeline which includes first searching an offline fact-check database and, if no match is found, performing a live web search via Google's Custom Search API. The system outputs a Credibility label (e.g., Likely False, Likely True, or Uncertain), a calibrated confidence score, and a concise explanation highlighting key textual cues influencing the model's decision. Through an interactive web interface, users can submit any claim and receive transparent, evidence-backed feedback, promoting digital literacy and responsible information consumption.

This is my Project GitHub Repository link:

https://github.com/satwik77-dev/UF-EEE6778-Fall25-TermProject

b. Dataset:

This project utilizes a **Hybrid dataset strategy** that combines a curated offline database of verified fact-checks with a live web search fallback for unseen claims. The offline component integrates two major fact-checking datasets: **PolitiFact Fact-Check Dataset** and **Snopes Fact-News Dataset** while the online component uses the **Google Custom Search API** to retrieve additional fact-check articles from trusted sources such as PolitiFact, Snopes, and Reuters when the offline database does not contain a relevant match.

The **PolitiFact Fact-Check Dataset** contains approximately 21,000 expert-verified statements spanning 2008–2022. Each record includes the claim text, verdict label (Six categories: True, Mostly True, Half True, Mostly False, False, Pants on Fire), originator, publication date, and the link to the full fact-check analysis. The **Snopes Fact-News Dataset** provides additional breadth, covering non-political topics such as health, science, and social rumors. It includes claim statements, verdict labels (True, False, Miscaptioned), page summaries, and "What's True/False/Unknown" metadata that can be used for interpretability and context expansion. Together, these two sources form a robust text-based dataset of around **10,000–15,000 unique claims** after cleaning and deduplication.

- **Data Format and Access:**
  Both datasets are in **CSV or JSON** format and will be accessed via Python (Pandas) for preprocessing and embedding. The merged dataset will be indexed in **FAISS** to enable fast semantic similarity search during offline retrieval. For the live search fallback, the system will query the **Google Custom Search API** and extract top article titles, snippets, and URLs for re-ranking.
- **Preprocessing Challenges:**
  Key preprocessing steps include text normalization (lowercasing, punctuation removal), label harmonization between datasets (e.g., mapping *Pants on Fire* and *Miscaptioned* to *False*), and deduplication of overlapping claims. Additionally, care will be taken to remove incomplete or ambiguous records and to handle inconsistent encoding or special characters in scraped data.
- **Ethical and Privacy Considerations:**
  All data sources are publicly available, derived from professional fact-checking organizations, and contain no personal user information. The project does not collect or store individual user data from Google API queries; results are used only for real-time evidence retrieval. Potential biases in fact-check coverage (e.g., regional or topic imbalance) will be acknowledged and mitigated by including diverse, multi-domain claims.

c. Planned Architecture:

The planned system follows a **Hybrid retrieval-and-classification architecture** designed for both accuracy and explainability. The pipeline integrates offline and online data retrieval with transformer-based text understanding to deliver fact-checked, confidence-calibrated outputs to the user in real time. The overall data flow can be summarized as:

1. Data Layer

- **Offline Source:** Preprocessed and Embedded dataset combining PolitiFact and Snopes fact-check records stored in a FAISS index for fast similarity search.
- **Online Source:** Google Custom Search API retrieves top candidate fact-check articles from trusted domains (e.g., PolitiFact, Snopes, Reuters) when the offline index finds no high-similarity match.

2. Retrieval & Ranking Layer

- Claims submitted by users are first normalized (tokenized, lowercased, cleaned) and encoded into sentence embeddings using **Sentence-BERT** or **MiniLM** models from the `sentence-transformers` library.
- The system searches the **FAISS index** to identify semantically similar verified claims. If no offline match exceeds a similarity threshold , the pipeline triggers a **live Google API call**, retrieves top 5–10 web results, embeds and re-ranks them using cosine similarity, and selects the best candidate(s).

3. Classification & Calibration Layer

- A fine-tuned **DistilBERT or RoBERTa Transformer** model classifies the claim into categories such as *Likely True*, *Likely False*, or *Uncertain*.
- Temperature scaling is applied to calibrate the model's confidence scores, ensuring realistic probability estimates rather than overconfident predictions.
- **Explainability tools** such as SHAP or Integrated Gradients highlight key phrases influencing the decision, providing transparency to users.

4. Inference & Aggregation Layer

- The final output aggregates the model's classification, calibrated confidence, explanation tokens, and retrieved fact-check metadata (title, source, URL, similarity score).
- If both offline and online retrievals are weak (below defined thresholds), the system defaults to: *"No verified fact-check found. Model prediction: [Label], Confidence: [Score]."*

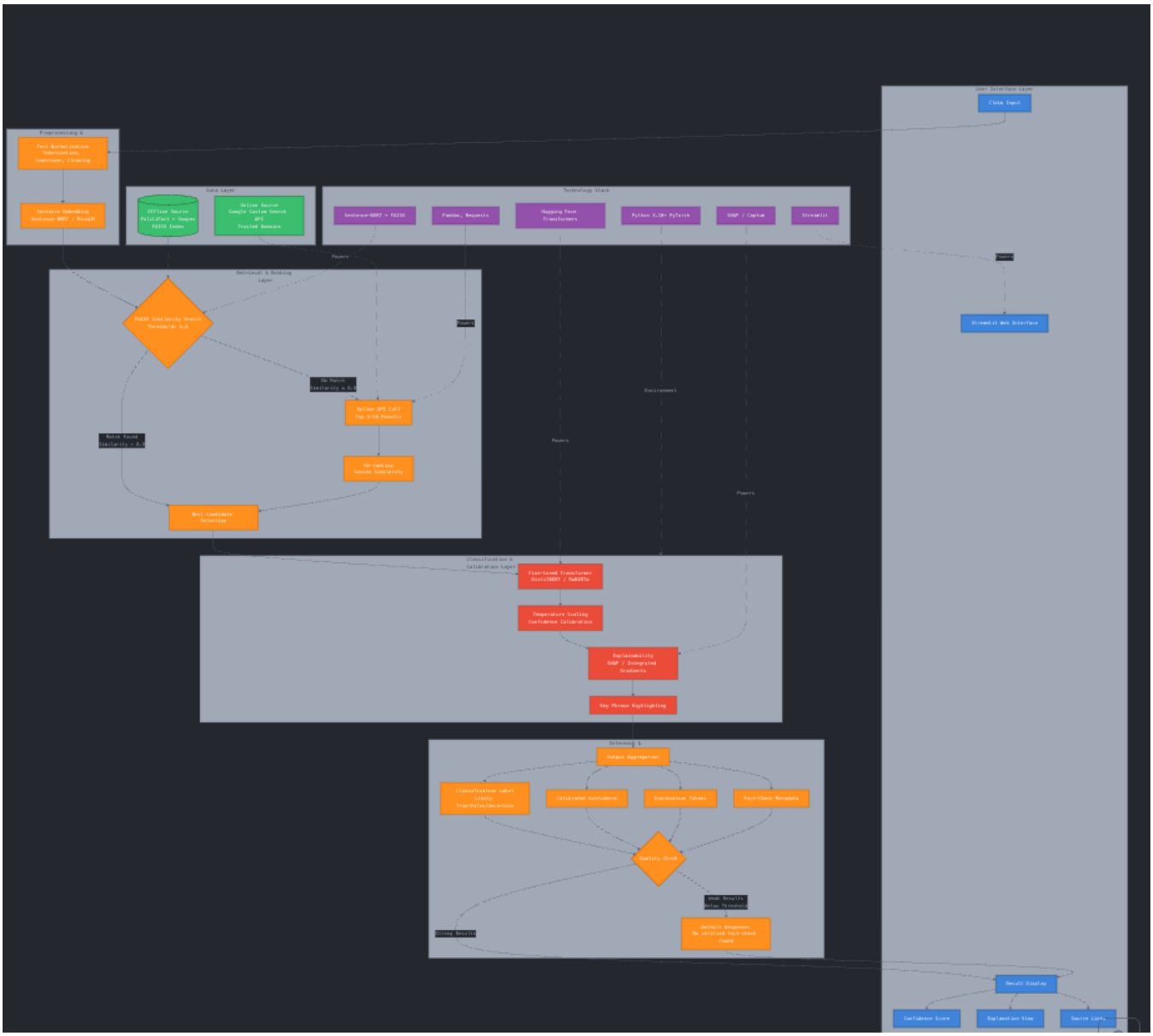

Fig: System Architecture

5. User Interface Layer

- The web interface will be built using **Streamlit**, chosen for its simplicity, interactivity, and compatibility with Python ML pipelines.
- Users can input any textual claim, view the predicted label, confidence score, and highlighted rationale, and access matched fact-check articles via clickable links.
- The interface will clearly indicate whether a result came from the offline database or live Google search, enhancing interpretability and transparency.

Planned Frameworks and Tools:

- **Text Modeling:** Hugging Face Transformers (DistilBERT / RoBERTa)
- **Embeddings & Search:** Sentence-BERT + FAISS
- **Explainability:** SHAP / Captum (Integrated Gradients)
- **Interface:** Streamlit
- **Data Handling & API:** Pandas, Requests, Google Custom Search API
- **Environment:** Python 3.10+, PyTorch

d. User Interface Plan

The user interface will be designed as an **Interactive fact-checking dashboard** built using **Streamlit**, offering a clean and intuitive experience for both casual users and researchers. Its purpose is to make the underlying model's reasoning and evidence fully transparent while allowing real-time exploration of claims.

- **User Input:**

Users will be able to enter Any factual claim or statement in a text input box (e.g., "The unemployment rate reached a 50-year low in 2024").

- **Outputs and Feedback:**

Once the user submits a claim, the system will display:

1. **Predicted Truthfulness Label:**
   (Likely True / Likely False / Uncertain) — this is generated by the transformer classifier.
2. **Confidence Score:**
   A calibrated probability value (0–1) indicating the model's certainty.
3. **Top Supporting or Contradicting Facts:**
   Retrieved from the **offline PolitiFact–Snopes index** or **live Google API search**, including the claim summary, verdict label, and clickable source link.
4. **Highlighted Text Explanation:**
   A visual display of the claim text with color-coded highlights from SHAP or Integrated Gradients, showing which parts of the sentence influenced the model's decision.
5. **Data Source Indicator:**
   A small tag (e.g., *"Verified via PolitiFact"* or *"Fetched from Google Live Search"*) for transparency.

- **Enhancing Usability and Interpretability:**

The interface will emphasize clarity, transparency, and minimal cognitive load for the User.

- Real-time retrieval ensures that users always get either verified references or a clear confidence-based model output.
- Visual explanations make the reasoning process interpretable to non-technical audiences.
- Color-coded cues (green for True, red for False, gray for Uncertain) and confidence bars make results easy to interpret at a glance.
- A feedback button ("Was this accurate?") may be added later to gather user responses for model refinement.

Sample Concept Sketch:

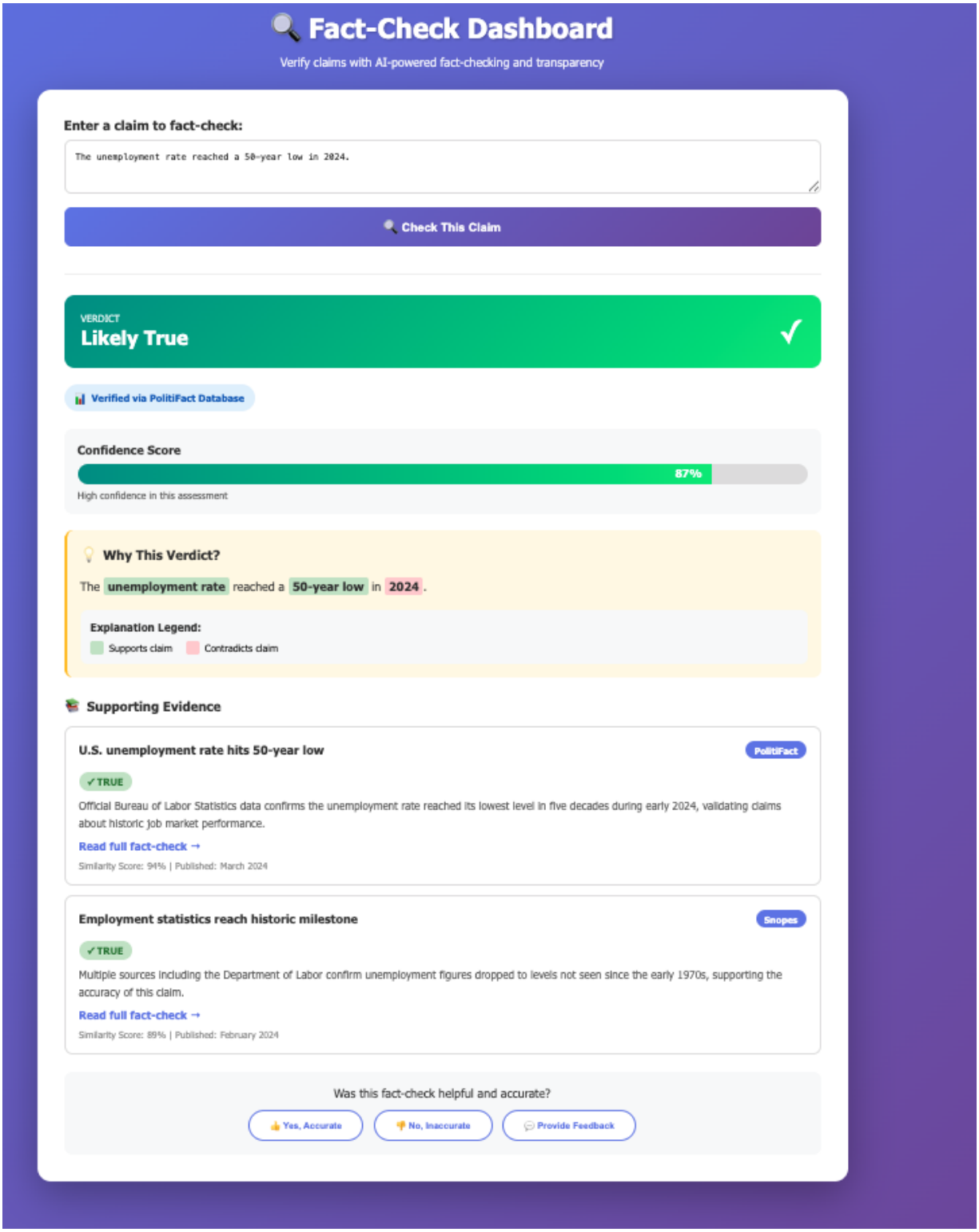

Fig**:** Sample Planned User Interface (UI) Design

e. Innovation and Anticipated Challenges

Innovation:

This project introduces a Hybrid fact-checking framework that bridges offline verified datasets and real-time online evidence retrieval, offering a system that is both grounded and adaptive. Unlike typical classifiers trained on static datasets, this approach dynamically expands its evidence base through Google Custom Search API when an unfamiliar claim is encountered allowing it to remain relevant to emerging events and misinformation trends.

Furthermore, the integration of Semantic similarity retrieval (via Sentence-BERT + FAISS) with transformer-based classification and explainability tools (SHAP / Integrated Gradients) brings interpretability to an otherwise opaque model. The system doesn't just label a claim as "True" or "False" it also provides traceable evidence and rationale behind each prediction, enhancing both trust and usability.

Anticipated Technical Challenges & Mitigation Strategies:

1. **Challenge 1: Inconsistent Data Quality and Label Alignment**

Issue*:* PolitiFact and Snopes use different labeling scales and editorial styles, making unified classification difficult.

Mitigation*:* Establish a Label harmonization map (e.g., grouping Pants on Fire and Miscaptioned into False), apply consistent text preprocessing, and validate mappings through exploratory data analysis before model training.

2. **Challenge 2: Semantic Drift and Real-Time Retrieval Noise**

Issue*:* Live Google search results may include opinion pieces or irrelevant sources that could distort evidence.

Mitigation*:* Restrict API queries to Verified fact-checking domains, implement a re-ranking filter based on cosine similarity threshold (>0.75), and include a fallback rule when no trustworthy evidence is found ("No verified claim available").

3. **Challenge 3: Model Interpretability and Overconfidence**

Issue: Transformer models often produce confident yet misleading predictions, reducing user trust.

Mitigation*:* Apply Temperature scaling for probability calibration and integrate Explainability visualization (highlighted text rationale) directly into the UI so users understand why a decision was made.

f. Implementation Timeline

Week	Focus	Expected Outcome
Oct 20 – Oct 26	**Dataset Integration & Environment Setup**	Collect and merge PolitiFact + Snopes datasets; clean, deduplicate, and harmonize labels. Set up project repository, install dependencies, and verify environment functionality through `setup.ipynb`.
Oct 27 – Nov 2	**Baseline Modeling & Retrieval System**	Implement Sentence-BERT embeddings and FAISS index for semantic search on offline data. Test claim retrieval workflow and confirm similarity-based matching works with sample claims.
Nov 3 – Nov 9	**Transformer Fine-Tuning & Calibration**	Fine-tune DistilBERT or RoBERTa on merged dataset. Evaluate model accuracy and confidence reliability. Apply temperature scaling for calibration and save baseline checkpoints.
Nov 10 – Nov 16	**Explainability Integration & Validation**	Integrate SHAP or Integrated Gradients to visualize key text features. Test explainability outputs with sample claims and validate interpretability on both True/False predictions.
Nov 17 – Nov 23	**Google API Integration (Live Fallback)**	Configure and test Google Custom Search API. Implement logic for switching between offline and live retrieval when no high-similarity result is found. Validate outputs with diverse claims.
Nov 24 – Nov 30	**Streamlit UI Prototype Development**	Build Streamlit-based dashboard with claim input, prediction display, confidence score, and highlighted text explanations. Connect UI to backend model and retrieval logic.
Dec 1 – Dec 7	**System Refinement & Testing**	Conduct full end-to-end testing across multiple claims. Optimize performance, fix inconsistencies, and finalize visual design. Prepare demonstration notebook and usage documentation.
Dec 8 – Dec 11	**Final Presentation & Report Submission**	Complete project demo, finalize technical report and README, polish UI visuals, and ensure repository reproducibility.

g. Responsible AI Reflection

This project emphasizes fairness, transparency, and ethical use of AI. By relying exclusively on publicly available, professionally fact-checked datasets (PolitiFact and Snopes), the system avoids processing any personal or sensitive user data. To promote fairness, the model and retrieval pipeline will account for coverage biases by integrating diverse claim topics (politics, health, social issues) and clearly indicating whether a claim originates from offline or live sources. Transparency is enhanced through explainability visualizations that highlight the textual cues driving predictions, while calibrated confidence scores communicate the model's certainty to users responsibly. Environmental considerations are addressed by limiting large-scale model retraining and performing inference efficiently using lightweight transformer variants (e.g., DistilBERT) to reduce computational overhead. Overall, the design prioritizes trustworthiness and responsible deployment in real-world fact-checking applications.