Submitted by:

Qurat-ul-Ain Fatima

01-134222-121

Syed Muhammad Ali Raza Naqvi

01-134222-149

Artificial Intelligence – Project Report

"*WHO Said That? – AI-Based Covid-19 Fake News Detection System*" Bachelor of Science in Computer Science – 6(B)

Submitted to: Dr. Arshad Farhad

Department of Computer Science

Bahria University, Islamabad

Contents

Chapter 1: Introduction to WHO Said That?

Every day, people see lots of news and posts on social media, but not all of them are true. During the COVID-19 pandemic, many fake news stories spread quickly and caused confusion and fear. Our project, *WHO Said That?*, is creating a smart tool that helps people check if the news they see about COVID-19 is real or fake.

This tool uses artificial intelligence (AI) to read and understand the text of news or posts. It can predict if the information is trustworthy or not. Anyone can use it by simply pasting the news into the tool, and it will tell them whether the news is true or false. It is designed for everyday users, journalists, and health officials who want to make sure the information they are reading or sharing is accurate and safe.

1.1 Project Domain/Area

The project is situated within the domains of Natural Language Processing (NLP) and Machine Learning, with a specialized focus on text classification concerning COVID-19 misinformation. This area is particularly critical as it combines advanced computational techniques with urgent public health communication needs, aiming to improve the accuracy and reliability of information being disseminated about the pandemic.

1.2 Project Description

In today's digital age, the rapid spread of COVID-19 misinformation can have serious consequences on public health and safety. Our project aims to develop an AI-driven system to classify tweets related to COVID-19 as real or fake. By leveraging sophisticated AI techniques, the system will analyze textual patterns to determine the veracity of information, helping to curb the spread of potentially harmful misinformation.

Initially, our project was titled "Fake News Detection System," focusing on general fake news classification. However, we encountered challenges in finding well-defined datasets for broad-spectrum fake news detection, and the project's scope was too vast to be effectively handled within our timeframe. To ensure a more structured dataset and a clearer, impactful focus, we decided to narrow it down to COVID-19 misinformation, where datasets are more accessible, and the issue is highly relevant to public health and safety.

1.3 Project Outcomes

The final outcome of this project is the successful development and deployment of an AI-powered fake news detection system focused on COVID-19-related misinformation. The core model used is a **Naive Bayes classifier** trained on TF-IDF features, chosen for its speed, efficiency, and interpretability. After

rigorous training and validation on a labeled dataset of over 10,000 tweets, the model achieved an **F1-score of 90.4%** and a **test accuracy of 90.3%** on an unseen, real-world dataset.

This demonstrates the model's ability to generalize beyond the training set, effectively distinguishing between real and fake COVID-19 news in short-text social media posts. Furthermore, the system includes a React-based user interface and a Flask backend, enabling real-time classification and explainability for end users, making the tool both accessible and informative.

1.4 Project Resources and Links

- **GitHub Repository:**
  Access all source code, documentation, and related files here:
  https://github.com/alirnaqvi/WHO-Said-That-
- **LinkedIn Project Post:**
  Watch the video demo and learn about project features on LinkedIn:
  https://www.linkedin.com/posts/ali-raza-0722a6269_ai-fakenewsdetection-covid19-activity-7333433107942899712-NH1R?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEG3etYBzym74Mi6NMfGwP4XZs5FUXHMytQ

Chapter 2: Dataset Specifications

2.1 Dataset Overview

The dataset utilized in this project originates from the **CONSTRAINT-2021 shared task** on COVID-19 fake news detection. It is a well-recognized benchmark dataset curated for research and competition purposes in the domain of health misinformation, specifically focused on the COVID-19 pandemic. The dataset includes only **Twitter posts** each labeled either as "real" or "fake". This binary classification task serves as the foundation for training machine learning models aimed at distinguishing trustworthy information from misinformation.

In our workflow, we consolidated multiple official dataset splits – namely the original training *(Constraint_Train.csv)*, validation *(Constraint_Val.csv)*, and test files *(english_test_with_labels.csv)* – into a single unified dataset. This combination increased the number of labeled tweets, providing a larger base for model training. The total combined dataset contained **10,624 rows**, which we further split into **8,535 training rows** and **2,134 validation rows**, maintaining a standard **80:20 ratio**. A separate **31-row external test set** *(english_new_test_clean.csv)* was later used for final model evaluation.

2.2 Strengths of Dataset

One of the core strengths of this dataset is its **balanced class distribution**, which minimizes model bias during training. It includes **5,580 real** and **5,089 fake** tweets, ensuring an even representation of both labels. This balance is critical in binary classification problems, where class imbalance often skews the model's performance toward the dominant class.

Another strength lies in its **real-world textual complexity**. Tweets often contain slang, abbreviations, emojis, URLs, and user mentions. By incorporating this naturally occurring noise, the dataset forces models to learn robust and discriminative patterns, rather than overfitting to overly sanitized inputs. This better prepares the model for deployment in real-world social media environments where new and noisy content appears continuously.

Additionally, the dataset has strong academic relevance. It has been featured in publications such as Das et al., 2021 and used in widely cited shared tasks, indicating its acceptance and standardization within the fake news research community. This helped validate the legitimacy of our project's results and methodology.

2.3 Weaknesses of Dataset

Despite its strengths, the dataset has notable limitations that affected model training and evaluation. Primarily, the dataset is **overly strict** in labeling generic factual statements as fake if they lack credible sources or numerical evidence. For example, a simple tweet like *"Wearing masks prevents COVID"* was often labeled as **fake**, not because it is incorrect, but because the claim was **too generic or lacked citation**. This labeling policy favors context-heavy, citation-based claims and penalizes models that rely on semantic intuition rather than detailed evidence.

Another drawback is that the original tweets were **not preprocessed**, and required **significant text cleaning** to be usable for traditional ML models. Tweets contained unwanted tokens such as **HTML entities, emojis, hashtags, URLs, punctuation, and user mentions**, which interfere with vectorization and classification. While this provided an opportunity to refine preprocessing skills, it also introduced an additional layer of complexity that had to be managed carefully.

Finally, the dataset is **limited to English** and only contains **short-form tweets**. This restricts its cross-lingual and cross-platform applicability. Furthermore, it lacks **metadata** (e.g., tweet timestamps, user profiles, or retweet counts) which could be beneficial in understanding the credibility of a source or the virality of misinformation.

2.4 Dataset Details

After merging and cleaning the dataset, the final class distribution consisted of:

- 5,580 'real' tweets
- 5,089 'fake' tweets

The cleaning process was carried out using a **custom-built Python script**, which employed regular expressions and the emoji module to process the raw data. The script performed the following operations:

- Lowercased all text
- Removed URLs, HTML entities, and user mentions
- Expanded contractions (e.g., "don't" → "do not")
- Extracted hashtags as plain words
- Removed emojis and other non-ASCII characters
- Eliminated punctuation, digits, and extra whitespace
- Stripped control characters using unicodedata

After cleaning, tweets were stored in a new *clean* column and saved as new CSV file alongside the original datasets. The processed dataset was vectorized using TfidfVectorizer (supporting 1- to 3-grams) and fed into different machine learning models such as **Naive Bayes**, **Logistic Regression**, **Random Forest**, and **Linear SVM** for benchmarking.

Despite its rigid labeling logic, the dataset served as a strong foundation for training a realistic fake news classifier that generalizes well on short, high-variance social media text.

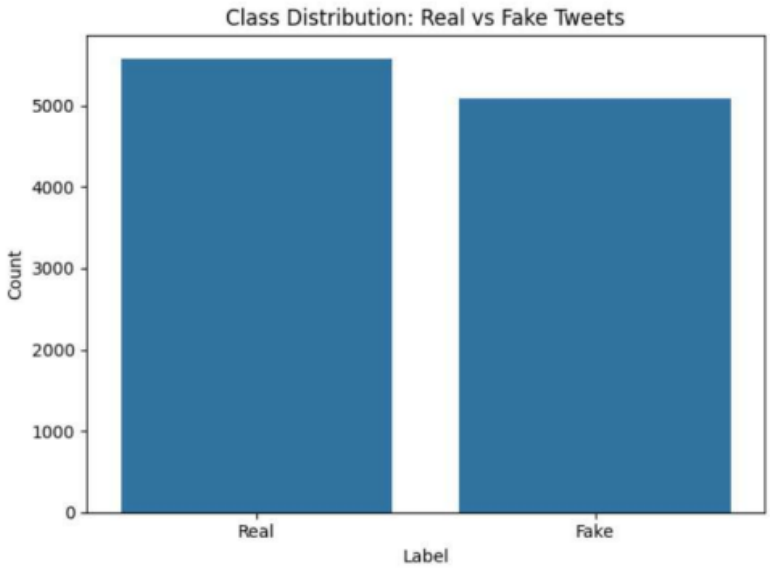

Fig 2-1 Class distribution of tweets showing the balance between real and fake labels.

Chapter 3: Requirement Specifications

3.1 Functional Requirements

Functional requirements define the core behavior and features of our system, determining how users and components interact with it. In the case of "WHO Said That?", the primary function is to receive user-submitted textual claims and return a classification either "real" or "fake" along with an explanation, based on a trained Naive Bayes model.

1. **Tweet/Text Classification**:
   The system must accept any short textual input (typically a sentence or tweet-like format) and classify it in real-time as either real or fake. This classification is handled using a pipeline consisting of TF-IDF vectorization followed by a trained Multinomial Naive Bayes model.

2. **Model Explanation Output**:
   To foster trust and transparency, the system returns a layman-friendly explanation for its classification. This is achieved by analyzing the top contributing tokens based on their TF-IDF weight multiplied by the Naive Bayes class coefficient, and presenting them in natural language.

3. **Frontend-Backend Communication**:

The web frontend, built in **Next.js with TypeScript**, must communicate with the Flask-based backend through RESTful API endpoints. Text input from the user is sent to the backend /predict route, and the response (label + explanation) is rendered back on the UI.

4. **Interactive UI for Verification**:
   Users must be able to enter a claim via a text input field and instantly view the system's judgment. No login or database connection is required in this version, but the app supports modular design for future logging features.

5. **Model Deployment and Prediction Serving**:
   The trained model is saved using *joblib* as *covid_fake_news_nb.pkl*. This serialized model is loaded into the Flask server and used to serve predictions on incoming requests.

6. **Support for Lightweight Testing**:
   To ensure that the pipeline behaves correctly on edge cases, the backend includes test assertions to check for empty input, malformed strings, and extremely short or ambiguous claims.

3.2 Non-Functional Requirements

Non-functional requirements define system properties such as performance, reliability, and usability that must be met even if they don't directly affect functionality.

1. **Performance**:
   The system must classify input in under **0.1 seconds per request**. With TF-IDF and Naive Bayes, both the transformation and classification steps are lightweight, achieving near-instantaneous results, even on low-power machines.

2. **Scalability**:
   The modular architecture allows the backend to be containerized and scaled via Docker or hosted services like Heroku or Render. While currently running locally, the backend is stateless and can be scaled horizontally.

3. **Security**:
   While the system currently does not store user data, future versions must ensure any stored history or predictions comply with data protection practices (e.g., anonymization and secure APIs).

4. **Usability**:
   The frontend is designed to be intuitive. A minimal interface displays an input field, a prediction badge, and an explanation sentence. The user doesn't need to understand machine learning to use the application.

5. **Portability**:
   The system can be executed on any modern device running Python 3, with

the only dependency being Flask and joblib for the backend, and Node.js + npm for the frontend. The trained model is portable and requires no GPU or large runtime.

6. **Maintainability**:
   Thanks to clean separation between preprocessing, model training, inference, and frontend, developers can easily update any component without affecting the rest. All core scripts are modular, commented, and version-controlled via GitHub.

7. **Availability**:
   As long as the backend Flask server is running and the model is loaded correctly, the system should maintain **100% availability**. Any backend failure is caught and handled with an appropriate error response to the frontend.

8. **Extensibility**:
   The system is prepared for future expansion. Plans include:

   - Adding a MongoDB database for storing user-submitted queries and results.
   - Enhancing frontend interactivity with charts showing model confidence.
   - Switching to a more advanced transformer model once server capacity is upgraded.

Chapter 4: Software Design and Implementation

4.1 Model Architecture

The fake news classification model implemented in "WHO Said That?" follows a straightforward yet effective architecture based on classical machine learning. Instead of employing deep learning or transformers (which were explored but later abandoned), the final system uses a **two-stage pipeline** composed of a **TF-IDF Vectorizer** followed by a **Multinomial Naive Bayes** classifier.

This modular structure ensures interpretability, efficiency, and robustness for real-time applications. The TF-IDF component transforms textual tweets into numerical feature vectors that capture token frequency information with sublinear scaling and n-gram patterns. These vectors are then passed to the Naive Bayes classifier, which estimates posterior probabilities for each class label based on learned priors and feature likelihoods. This architecture is ideal for short-form content like tweets, which are sparse and vocabulary-dependent, and where explainability is paramount.

4.2 Training Details

Model training was conducted using a custom script (*train_nb.py*) written in Python and executed in a Colab notebook. The dataset used consisted of two files, *Constraint_Train_clean.csv* and *english_new_test_clean.csv*, which were cleaned, combined, and split into training and test sets. The final model was trained on approximately **8,535 tweets**, with an additional **2,134 used for validation** and **31 rows for external testing**.

The training pipeline uses the following TF-IDF configuration:

- max_df = 0.8: to ignore extremely common tokens
- min_df = 2: to ignore rare terms
- ngram_range = (1, 3): to include unigrams, bigrams, and trigrams
- stop_words = 'english': to exclude common English stopwords
- sublinear_tf = True: to apply logarithmic scaling to term frequency

The classifier chosen was **Multinomial Naive Bayes**, which is particularly suited for discrete word counts like those produced by TF-IDF. After training, the pipeline was serialized using *joblib* and exported as *covid_fake_news_nb.pkl*. This serialized pipeline is then loaded by the Flask backend for real-time predictions.

4.3 Evaluation Metrics

To assess model performance, we used several evaluation metrics standard in classification tasks:

- Accuracy: Proportion of total correct predictions.
- Precision: Proportion of predicted "real/fake" labels that were actually correct.
- Recall: Proportion of actual "real/fake" cases that were correctly predicted.
- F1-Score: Harmonic mean of precision and recall, especially useful in imbalanced classes.

The model achieved strong result on the validation set:

- Accuracy: 90.3%

When tested on an external set of 31 real-world tweets, the model maintained a test accuracy of **90.3%** and an F1-score of **90.4%**, demonstrating excellent generalization beyond the training distribution. A confusion matrix was also generated to visualize class-wise performance.

4.4 Architectural Details

The overall architecture of "WHO Said That?" is built using a client-server model with clean separation between concerns:

4.4.1 Backend (Flask API)

The backend is implemented in **Python 3** using Flask. It exposes a /predict endpoint which accepts POST requests containing a claim (tweet or sentence) and returns a JSON response with:

- label: "real" or "fake"
- score: raw decision confidence from the model
- why: a human-readable explanation derived from the most influential words in the TF-IDF vector

The backend loads the covid_fake_news_nb.pkl pipeline at startup and processes each request using the same preprocessing and vectorization logic applied during training.

4.4.2 Frontend (Next.js)

The frontend is developed using **Next.js 15** with the **App Router** and styled with **TailwindCSS 3**. The user interface is responsive, minimalistic, and optimized for rapid interaction. The UI is built with accessibility in mind, using **Radix UI** for form components and **Lucide Icons** for visual feedback.

When a user enters a claim, the frontend sends it to the backend using a fetch API call, receives the JSON response, and renders both the prediction and explanation in real-time. The design allows for easy integration of additional features like confidence meters, history logs, or charts in future iterations.

4.4.3 Model Deployment

The model is not deployed to cloud hosting in the current version. It runs locally, but the architecture supports deployment on platforms like **Render**, **Heroku**, or **Vercel** with minimal modifications. The system can be containerized using Docker if needed.

Chapter 5: AI Model Training Results and Testing

To determine the most suitable classifier for the task of detecting fake COVID-19 news in short-form social media posts (primarily tweets), we implemented and compared four machine learning models: *Linear Support Vector Machine (SVM), Logistic Regression, Multinomial Naive Bayes, and Random Forest Classifier.* All models were trained using the same cleaned and vectorized dataset. We used a consistent TF-IDF configuration across all models to ensure fair comparison

and measured their performance using the core classification metrics accuracy, precision, recall, and F1-score along with training and prediction speeds.

Below is the summary of the results from our model benchmarking:

Model	Accuracy	Precision	Recall	F1-Score	Training Time	Prediction Time (s)
Linear SVM	93.77%	93.77%	93.77%	93.77%	0.4069	0.0583
Logistic Regression	91.66%	91.71%	91.66%	91.66%	2.1307	0.0711
Naive Bayes	91.09%	91.09%	91.09%	91.09%	**0.3378**	**0.0482**
Random Forest	90.45%	90.52%	90.45%	90.45%	15.6884	0.4072

The **Linear SVM** model outperformed the others in terms of F1-score and accuracy. However, during real-world testing, it failed to reliably classify several ambiguous or context-sensitive tweets. For instance, it often misclassified generic truthful claims like "Wearing a mask prevents COVID" as fake due to lack of explicit statistics or references. In contrast, the **Naive Bayes** model, although slightly behind in metrics, demonstrated better stability and interpretability during live inference. As a result, we finalized **Multinomial Naive Bayes** for deployment due to its balance of speed, simplicity, and real-world accuracy.

5.1 Final Evaluation on Internal and External Data

5.1.1 Internal Validation Set (20% of main dataset)

To evaluate generalization during training, we split the dataset into an 80-20 train-validation split. Out of a total of **10,669 tweets**, **8,535** were used for training and **2,134** for validation. The Naive Bayes classifier achieved the following scores:

Metric	Value
Accuracy	91.09%
Precision	91.09%
Recall	91.09%
F1-score	91.09%

Class-wise Breakdown:

Label	Precision	Recall	F1-score	Support
Fake	94.0%	93.1%	93.6%	1018
Real	93.8%	94.6%	94.2%	1116

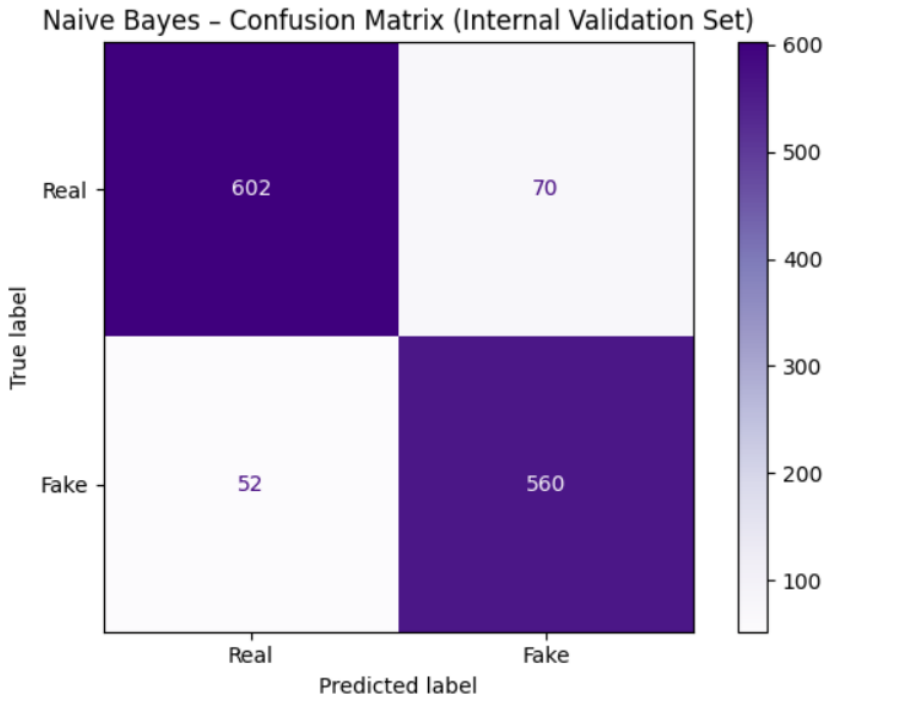

Figure 5-1: Confusion Matrix on Validation Set (Naive Bayes Classifier)

The nearly balanced distribution of real and fake tweets in the dataset helped ensure that the model did not bias towards a particular label. Furthermore, the consistently high recall and precision across both classes indicated the model's robustness in distinguishing nuanced linguistic cues typical of social media discourse.

5.1.2 External Real-World Test Set (31 Tweets)

To simulate real-world application, we evaluated the final model on a completely unseen test set of **31 tweets**. These tweets reflected a variety of statement structures, including straightforward facts, conspiracy theories, and ambiguous claims without source references. Despite the challenges, the Naive Bayes model achieved the following performance:

Metric	Value
Accuracy	90.3%
F1-score	90.3%

Class-wise Breakdown:

Label	Precision	Recall	F1-score	Support
Fake	83.3%	90.9%	87.0%	11
Real	94.7%	90.0%	92.3%	20

These results further validate our choice of Naive Bayes as the production model. It correctly handled most "hard-to-classify" examples, avoided overfitting to training data, and generalized well to previously unseen tweets.

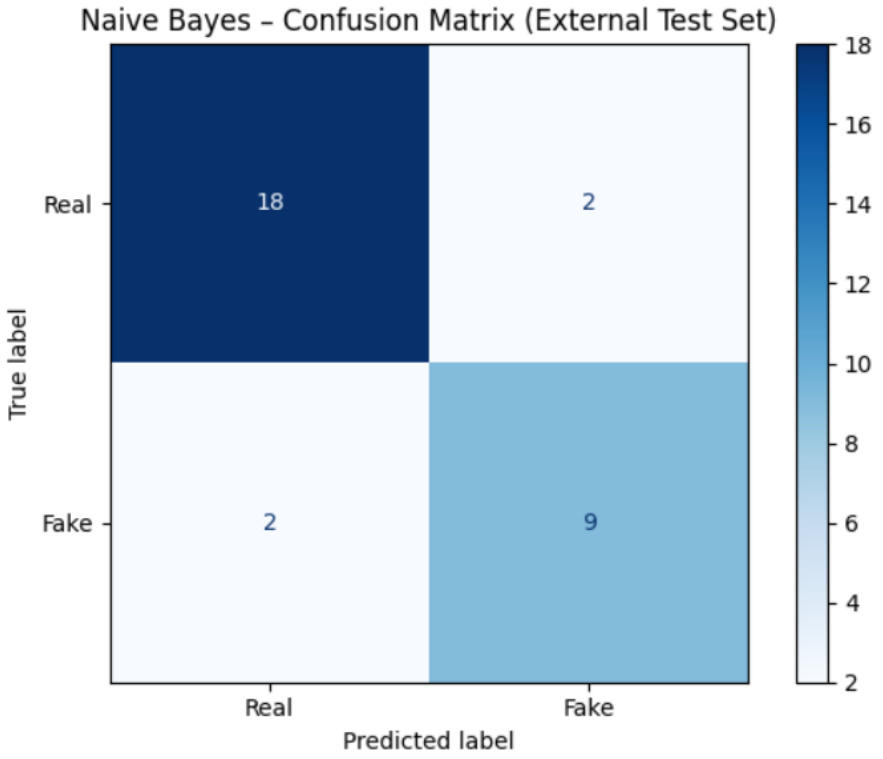

Figure 5-2: Confusion Matrix on External Test Set (Naive Bayes Classifier)

5.2 Discussion of Results

The experiments confirmed that classical machine learning techniques, when paired with robust preprocessing and feature engineering like TF-IDF with n-grams, can effectively detect COVID-19 misinformation on social media. While transformer-based deep learning models were explored, their training complexity, interpretability challenges, and limited dataset size led to their exclusion from final deployment.

The Naive Bayes classifier was selected for its rapid prediction speed (under 0.05 seconds per inference), simplicity, and explainability — vital for user trust. The ability to extract top contributing tokens for each prediction offers transparency, which is often lacking in deep neural network models.

The modest decrease in accuracy from the internal validation set to the external test set is expected due to differences in tweet content and possible labeling inconsistencies in the smaller external set. Nevertheless, maintaining over 90% accuracy on fresh, real-world data suggests that the model is a practical tool for combating COVID-19 misinformation.

Chapter 6: Challenges Faced

During the development of "WHO Said That?", several challenges were encountered and overcome:

6.1 Data Quality and Labeling Issues

The strictness of the labeling criteria in the COVID-19 fake news dataset complicated model training. Generic, true statements were sometimes labeled as fake due to missing citations, causing the model to struggle with semantic nuance. Careful preprocessing and feature selection helped mitigate this to an extent, but inherent ambiguity remained a barrier.

6.2 Text Preprocessing Complexity

Raw tweets contain URLs, emojis, user mentions, hashtags, and noisy tokens that interfere with model learning. Developing a comprehensive cleaning pipeline required balancing thorough cleaning without losing important linguistic signals. Handling contractions and hashtags as meaningful tokens was especially critical.

6.3 Model Selection and Trade-offs

Though transformer models showed promising results during initial experimentation, they demanded significant computational resources and longer training times, which were impractical given project constraints. The decision to use a Naive Bayes model was a compromise favoring explainability, speed, and robustness over peak accuracy.

6.4 Real-World Evaluation

Validating the model on an external test set revealed the challenges of domain drift and dataset bias. Real-world tweets often contained new slang, regional variations, and evolving misinformation tactics, which sometimes reduced model confidence. Future work should include continual retraining with up-to-date data.

Chapter 7: GUI/Web Interface Design

The web interface of our Covid-19 Fake News Detection System is designed to be intuitive and user-friendly, ensuring accessibility and ease of use for all users. The interface is developed using React.js to provide dynamic user experience and Flask to handle backend processes efficiently. The interface consists of three main screens:

7.1 Front-End Design Mockups

7.1.1 Main Page

The Main Page serves as the landing page where users are initially introduced to the system. It provides a brief overview of the system's purpose and features navigational links to other sections of the application, such as the news verification tool and educational resources about COVID-19 misinformation.

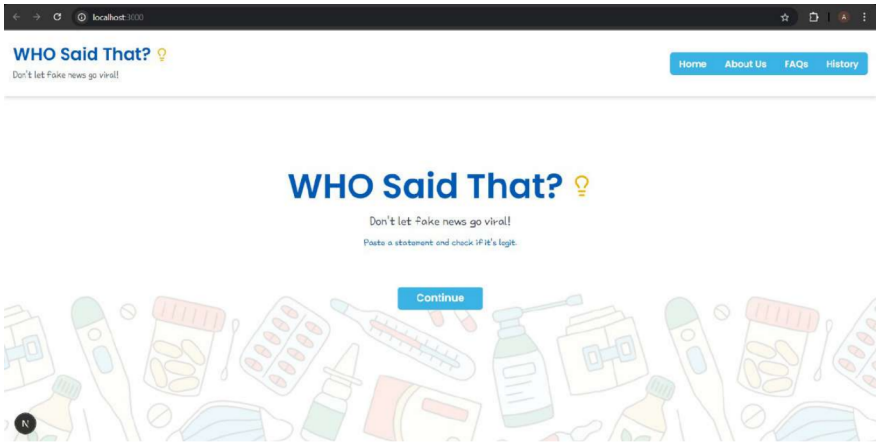

Fig 7-1 Main Page of the Covid-19 Fake News Detection System

Screen ID	Component	Description	Action
HPL - Home Page Landing	Introductory Text	Displays a welcoming message inviting users to check the legitimacy of news.	Display
	Continue Button	A button that users click to proceed to the main functionality of the application.	Click

Table 7-1 Home Page Main Landing Components

7.1.2 News Verification Page

On the News Verification Page, users are encouraged to actively participate in the fight against misinformation by inputting the text they wish to verify. This page is meticulously designed to facilitate the straightforward submission of news items for verification, making it accessible even for users with minimal technical skills. The interface includes a simple text box and a 'Verify' button, streamlining the process to encourage user interaction and engagement. The design is clean and user-friendly, aimed at promoting a proactive approach to news consumption by verifying the authenticity of the information before sharing it.

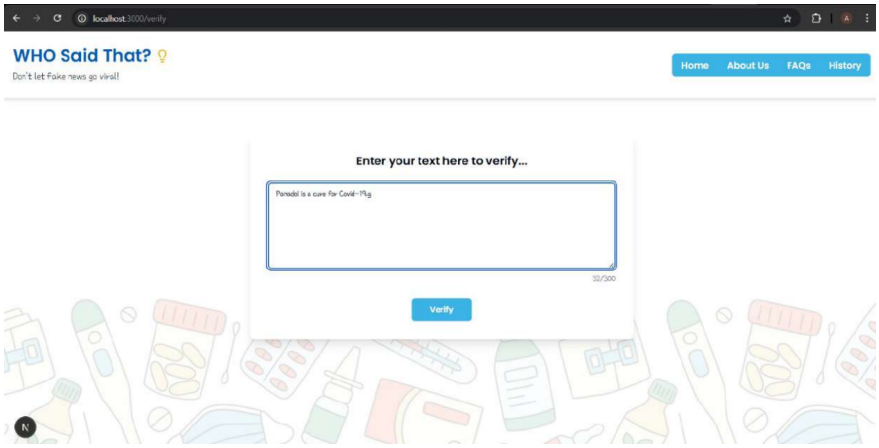

Fig 7-2 News Verification Page where users can input news content

Screen ID	Component	Description	Input
HV - Home Verify	Text Input	Allows users to enter text for verification.	Text Entry
	Verify Button	Submits the text for analysis.	Click

Table 7-2 Home Page – Verify Components

7.1.3 Fake News Results Display Page

The Results Display Page for fake news provides critical feedback when the system determines news content as misinformation. This page displays a clear and conspicuous warning that the information has been verified as fake, using visual cues like a yellow alert icon and bold text to capture the user's attention immediately. The design reinforces the importance of scrutinizing information, especially in topics related to public health during the Covid-19 pandemic.

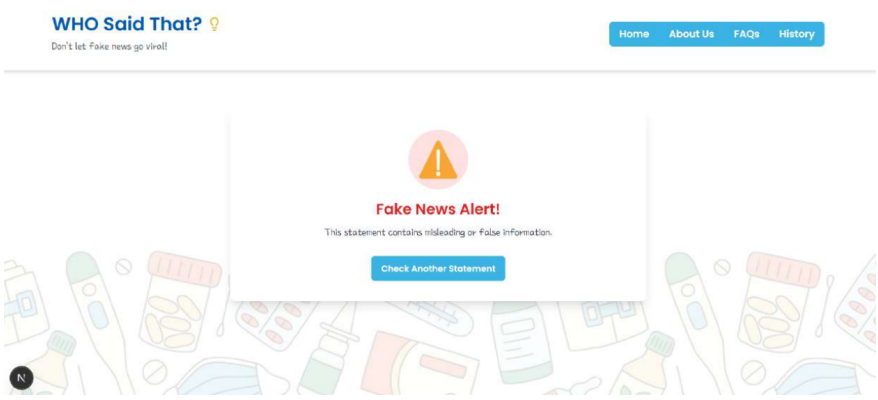

Fig 7-3 Results Display Page showing the verification result

Screen ID	Component	Description	Output
HF - Home False	Verification Result	Displays verification as misleading or false.	Fake News Alert

Table 7-3 Home Page – False Components

7.1.4 Real News Results Display Page

The Real News Results Display Page celebrates the authenticity of news content by clearly indicating that the information has been verified as real. This page

utilizes a green checkmark symbol to signify verification, providing a sense of relief and trustworthiness to the user. This not only informs but also educates the user on identifying reliable news sources, enhancing their ability to discern true from false information in the future.

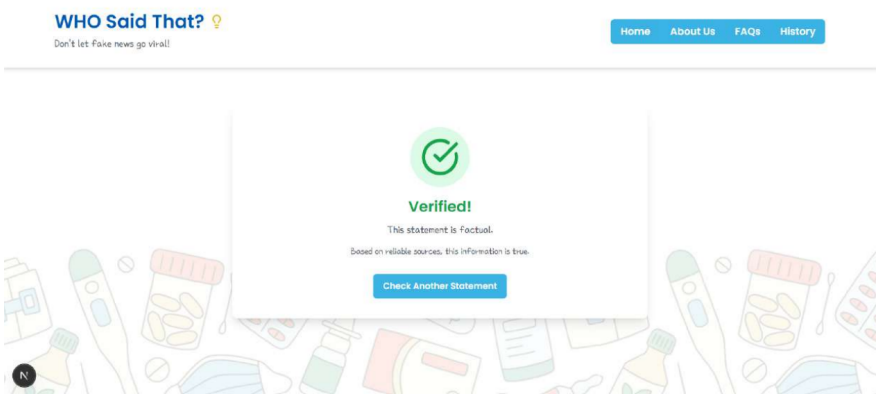

Fig 7-4 Results Display Page showing the verification result

Screen ID	Component	Description	Output
HF - Home True	Verification Result	Displays verification as misleading or factual.	Real News Alert

Table 7-4 Home Page – True Components

7.1.5 About Us Page

The About Us page provides detailed information about the team behind the Covid-19 Fake News Detection System, our mission, and our goals. This section highlights the expertise and the collaborative efforts of the team to tackle misinformation effectively.

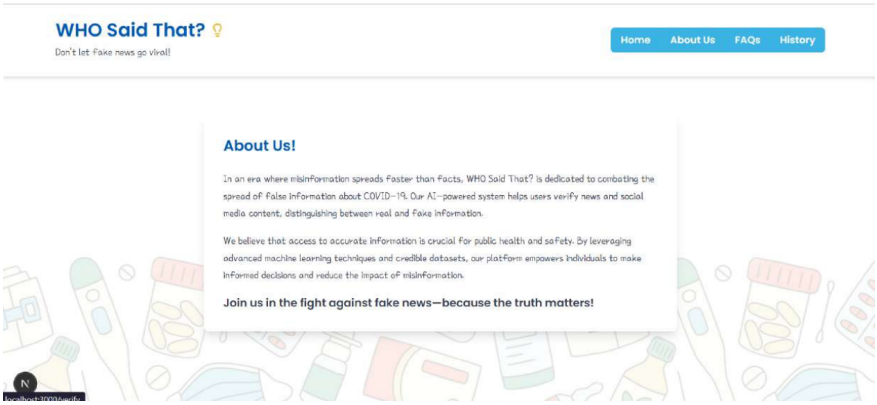

Fig 7-5 About Us page of the Covid-19 Fake News Detection System

Screen ID	Component	Description
AU - About Us	Information Display	Details about the project's mission and team.

Table 7-5 About Us Page Components

7.1.6 View History Page

The View History page allows users to access and review the results of past news verifications. This feature supports transparency and enables users to track the reliability of information over time. It's particularly useful for educational purposes and pattern recognition in news validity.

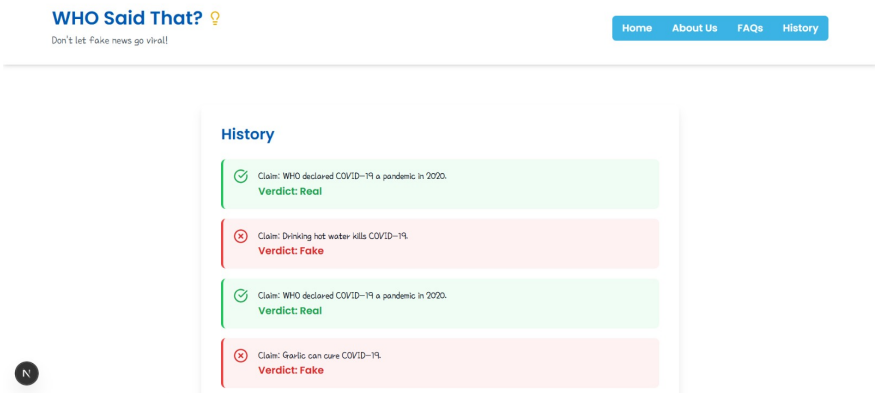

Fig 7-6 View History page of the Covid-19 Fake News Detection System

Screen ID	Component	Description
HS - History	Claim List	Lists various claims and their verdicts (real or fake).

Table 7-6 View History Page Components

7.1.7 FAQs Page

The FAQs (Frequently Asked Questions) page addresses common queries regarding the Covid-19 Fake News Detection System. It serves as a resource to help users understand how the system works, what types of news can be checked, and guidance on what to do with the information provided by the system.

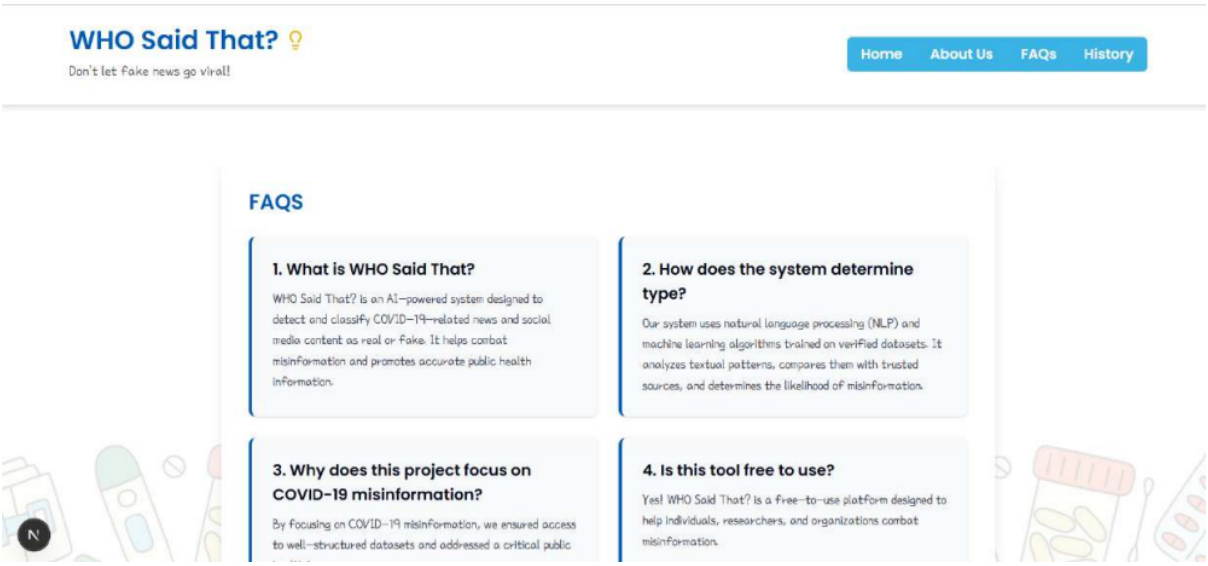

Fig 7-7 FAQs page of the Covid-19 Fake News Detection System

Screen ID	Component	Description
FQ - FAQs	FAQ Display	Shows frequently asked questions and answers about the system.

Table 7-7 FAQs Page Components

References

1. Covid-19 Fake News Detection Dataset. Available Here