FactCheckr

Student Names: Muhammad Hassan Naveed Baig & Affan Siddiqui
Supervisor Name: Mr. Abdul Raheem Aleem

Program Name: BS(IT)

Department of Computer Science

BAHRIA UNIVERSITY ISLAMABAD

Certificate

We accept the work contained in the report titled "FactCheckr", written by **Muhammad Hassan Naveed Baig** & **Affan Siddiqui** as a confirmation to the required standard for the partial fulfillment of the Degree of Bachelor of Science in Computer Science. **Approved by...:**

Supervisor: Mr. Raheem Aleem

Internal Examiner:

External Examiner:

Project Coordinator: Dr. Asim Ali

Head of the Department: Dr. Khurram Ehsan

November 3^{rd}, 2025

Abstract

Here is the Abstract for your Final Year Project documentation, synthesized from the information provided in your uploaded file.

Abstract

These are the Abstract of your Final Year Project documentation, which has been synthesised out of the information given in your uploaded file.

Abstract In the age of digitalization, the spread of information over social media and online platforms is incredibly fast, and a very important issue arose: the spread of false information and fake news. The given project is the Fake News Detector, the all-encompassing web-based program that aims to automate the concept of evaluating the credibility of news articles by the means of the sophisticated multi-dimensional analysis. The system uses a hybrid scoring system, which combines four verification elements: machine learning based text reporting, source reputation reporting, clickbait scheme identifying and image integrity reporting based on perceptual hashing.

The core classification engine uses a Supervised Logistic Regression model which is trained on a labeled set of 94, 408 articles using TF-IDF vectorization which identifies linguistic patterns related to deceptive articles. This machine-learning based subdivision is assisted by a database of verified domains and a unique clickbaiting recognition model. The application is designed as a Micro services based system and uses react to entertain the user interface, node.js and express to do the back-end logic, and finally a Python flask service to do machine learning inferences.

The results of the experiment prove that the system has a classification accuracy of 92.5 per cent in the test dataset. By providing features such as real-time URL scraping, live news exploration via NewsAPI, and explainable AI results, the Fake News Detector offers a scalable, accessible, and high-performance solution for journalists, researchers, and the general public to combat misinformation effectively.

Acknowledgment

Praise be to Allah Almighty, the Most Gracious and the Most Merciful, for the benefits and the strength. We would like to express our sincere appreciation and gratitude to **Mr. Abdul Raheem Aleem**, our supervisor, for his knowledgeable advice and availability of comprehensive project information.

We also appreciate **Dr. Asim Ali**, the FYP project coordinator, sharing information on meetings and project milestones.

Muhammad Hassan Naveed Baig
Affan Siddiqui
Bahria University Islamabad, Pakistan

November 2025

"When you write a book, you spend day after day scanning and identifying the trees. When you're done, you have to step back and look at the forest."

Stephen King

Acronyms and Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
CDN	Content Delivery Network
CORS	Cross-Origin Resource Sharing
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DOM	Document Object Model
FR	Functional Requirement
GCP	Google Cloud Platform
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
JWT	JSON Web Token
LIME	Local Interpretable Model-agnostic Explanations
LTS	Long Term Support
ML	Machine Learning
MVC	Model-View-Controller
NFR	Non-Functional Requirement
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
NPM	Node Package Manager
ORM	Object-Relational Mapping
pHash	Perceptual Hashing
RAM	Random Access Memory
REST	Representational State Transfer
SPA	Single Page Application
SQL	Structured Query Language
SSD	Solid State Drive
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User Interface
URL	Uniform Resource Locator
XSS	Cross-Site Scripting

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

Chapter 1

Introduction

1.1 Introduction

In the modern digital age, the international landscape of information distribution has been redesigned radically due to the prevalence of social media and web resources. The technological development has transformed how people access news and received unparalleled access to real-time information. The same convenience has, however, also led to a major challenge affecting society critically: the uncontrolled spread of fake news and misinformation [1, 2]. Fake news is defined as any information that is intentionally produced and reported as authentic news and usually designed in advance to misinform a reader to meet particular goals, including political maneuvering and social instability as well as economic interests through advertisement income [3]. Such a deluge of misleading information is dangerous at the individual level as citizens will find it hard to distinguish between facts and fiction and the societal level as the credibility in legitimate journalism will disappear, and this may pose a danger to the health of the people and the democratic process itself [4].

To address this increasing crisis, this project offers the Fake News Detector, a full web-based application aimed at taking advantage of the power of artificial intelligence and machine learning to tackle the issue of misinformation spreading [5]. Contrary to the conventional fact-checking approaches which are usually labor and time-intensive, the system offers an automated and intelligent platform that could be used to determine the credibility of news articles in real-time. The solution will be designed to cater to a broad range of different types of stakeholders such as journalists, researchers, educators, students, and general users and will provide them with a powerful tool to determine the authenticity of information without any specialized technical knowledge.

The Fake News Detector functioning relies on a complex hybrid scoring system that merges various dimensions of analysis so that the results can be high-fidelity. The system utilizes an oversighted Machine Learning (ML) classification engine and in particular a **Logistic Regression** framework that is trained on a large dataset of 94,408 labeled articles [6]. It is a model that employs linguistic features and patterns of language with the help of **Term Frequency-Inverse Document Frequency (TF-IDF)** vectorization [7] to show validation with an accuracy of 92.5% in determining genuine versus dishonest

content [8]. Based on the textual features of an article, the system is able to produce probability scores to show how likely that material is to be authentic or fraudulent.

The system adds three more layers of verification in order to supplement the machine learning predictions [9]. To assess reputation, first an Article Reputation module compares the article domain with a maintained list of trusted news organizations. Second, there is a Clickbait Detection algorithm that studies the headline to get the sensationalist patterns, including an overuse of punctuation and capitalization, which is typically a sign of low-quality writing [10]. Lastly, there is Image Integrity Analysis component, which identifies manipulated or recycled images using perceptual hashing (pHash) to compute Hamming distances between the image and a database of known fake images [11, 12]. It is a multi-faceted approach that will guarantee that the credibility review has been done in a holistic manner, covering not only the text but also the context and graphics of the news itself [13].

Moreover, it is an app that focuses on the high level of usability and practicality. It has a user friendly web interface which allows different types of input such as direct analysis of URLs and raw text analysis. Also, external services such as NewsAPI are incorporated into the system to enable users to read and verify instantly the most discussed news that is considered trending. The Fake News Detector will effectively transform the current state of misinformation detection, providing the user with easy to understand **AI output, which is visualizing which qualities contributed to a particular prediction**, thus it is effective in both alerting and informing users on the signs of credibility, creating a better and media literate society [14, 15].

1.2 Objective

The main aims of the Fake News Detector system include the creation of a powerful user-friendly application to fight misinformation, which can be attained by achieving the following main objectives:

- **Automated Credibility Assessment:** Offering a user an instant and AI-powered news analysis and text-based content analysis that generates the exact confidence score in the content, which functions as an indicator of whether the content is authentic or fake.

- **Multi-Dimensional Checking:** To apply an advanced hybrid checking system which is able to scan text content through machine learning, check

source domain reputation, identify clickbait patterns, and assess image integrity at the same time.

- **Real-Time News Exploration:** To connect to external news aggregators (NewsAPI) in order to get and automatically process current, trending articles, so users can consume the current events with the immediate credibility background.

- **User Accessibility and Usability:** To develop a use-friendly, receptive web-based interface that permits the tool to be used by a vast extent of audience and accessible to numerous users such as journalists and the general population without having the need of technical skills.

- **Administrative Control and Monitoring:** To provide an all-inclusive dashboard to the system administrators in managing trusted domain databases, setting hybrid scoring weights and system performance analytics monitoring.

- **Explainability and Transparency:** To achieve transparency in the system by giving users explainable AI performance, it is important to give them visual breakdowns, which are detailed on the value of features so that they can comprehend which linguistic or structural factors were used to predict the credibility.

1.3 Problem Statement

Misinformation and fake news are highly dangerous to both people and the entire society and, therefore, the Fake News Detector will be highly required to overcome the following challenges [2]:

- **Inconvenience in Differentiation:** The inability of the citizens to differentiate between genuine and fake news is mostly conditioned by the deficit of the technical experience and emotional appeal that is mainly used in sensationalized news headlines.

- **Lack of Efficiency of the Manual Checking System:** Analogue processes of checking the facts are too time-consuming and effort intensive

and impractical users should be able to properly check every information they hear on a real-time basis.

- **Destruction of Public Fidelity:** Due to the extensive distribution of fake data, there is a massive loss in faith in proper journalism and the media houses under the guise of which it propagates, and as well, it promotes political polarization [4].

- **Poor Availability of Resources:** The majority of the general users do not have access to the advanced tools of checking as the credibility of the information, since the currently existing are too complicated to use, or are not as precise as desired or are not intended to be used by ordinary people.

- **Societal and Economic Risks:** Uncontrolled misinformation has been a source of some societal health risks due to medical fraud and economic consequences due to economic lies and frauds.

1.4 Methodologies

1.4.1 Predictive Planning through Machine Learning

The analysis engine of FactCheckr is a supervised learning model, which is based on a **Logistic Regression** model, and that classifies content [5]. This methodology involves:

- **Data Training:** The model got approximately one dataset of 94,408 typed articles (50,391 real, 44,017 fake) to get a high level of accuracy [8].

- **Preprocessing:** Removing of special characters, lowercasing, and tokenization of raw text in preparation of it before analysis.

- **Regularization:** L2 regularization should be installed in the training phase to avoid overfitting and to have the model performed well in new data.

- **Metric Evaluation:** The performance is evaluated based on accuracy rates of the 92.5% precision, recall, and F1-scores to confirm the performance accuracy of the model.

1.4.2 Natural Language Processing (NLP)

To convert textual data in the form of numerical feature vectors, FactCheckr uses Term Frequency-Inverse Document Frequency (TF-IDF) [7]:

- **Feature Extraction:** The extraction of unigrams and bigrams in order to make sense out of words that are not singular.
- **Noise Removal:** To remove linguistic noise in the analysis, the stop words will be removed.
- **Vocabulary Optimization:** To trade-off between performance and computation, a maximum of 10,000 most informative features are used.
- **Normalization:** A process which transforms the raw text into the uniform feature representations which are required by the machine learning model input.

1.4.3 Source Verification

The platform bases itself on the domain reputation approach to evaluate the authority of the source of information [3]:

Trusted Database: A database can be consulted about the trusted integrity of news outlets by querying more than 50 trusted news sources to cross-reference the original source of an article.

- **Reputation Scoring:** Score of trust between 0 and 100; this is provided by journalistic standards.
- **Categorization:** The grouping domains into different categories including highly trusted, trusted, unknown or untrusted based on their score.
- **Hybrid Integration:** Considering the score of the reputation of the source as an important element (20%) as a part of the final hybrid credibility of FactCheckr.

1.4.4 Clickbait Detection

FactCheckr applies pattern matching and key-word searching to detect sensationalized articles aimed at fooling a reader [16]:

- **Pattern Recognition:** Detecting the spelling syntax features (e.g., excessive capitalization, e.g. ALL CAPS, punctuation, e.g. ___ ate) [17].

- **Keyword Analysis:** Identifying clickbait expressions such as "You Won't Believe" or "SHOCKING", through an established library.
- **Emotional Scam:** Flagging keywords that are aimed at creating a sense of urgency induced by the emotion of any kind (like BREAKING or EXPOSED).
- **Scoring Aggregation:** Generation of a 0-1 clickbait score grounded on the quality of these signs in the headline and text.

1.4.5 Image Integrity Analysis

The platform uses Perceptual Hashing (pHash) algorithms to identify one of possible courses of actions: visual material manipulation or reuse [11]:

- **Hash Generation:** Generation of difference hash (dHash) of each image of an article retrieved.
- **Comparison of similarity:** The Hamming distance is calculated between the new image hash and database containing the images of known fake news images [12].
- **Detecting Manipulation:** This identifies instances of out-of-context usage of the content by flagging them as more than 95% similar to known manipulated images.
- **Persistence:** Storing hashes in an image hash database of custom `imageHashService` to be cross-referenced rewardingly.

1.5 Tools and Technologies

FactCheckr is based on the modern technology stack to offer performance, scalability and maintainability, often following the Microservices architectural approach [18, 19].

1.5.1 Front-end Technologies

The user interface is developed based on React 18 to provide responsive and interactive experience to the user [20]:

- **React 18:** A powerful page with the use of component based architecture and hooks.

- **Vite:** Uses the same tool as the build tool to guarantee reduced development times, as well as, better bundling than using traditional tools.
- **Chart.js:** It has been integrated into charting as an active data visualization and analytics dashboard tool.
- **Framer Motion:** It is to be used to make declarative transitions between layout and animations that are smooth.
- **Lucide React:** Offering a modern and coherent set of icons to the interface of the application.
- **Vanilla CSS:** The assumption is that good styling is possible without the huge overhead of frameworks such as creating a full Dark theme.

1.5.2 Back-end Technologies

The thematic application back-end (referred to as the back-end system) is constructed based on Express.js operating on a Node.js runtime platform [21]:

- **Express.js:** It is used as a framework of the web application to work with routing and RESTful API requests.
- **Prisma ORM:** SQLite-specific type-safe database query and schema migration tools.
- **Puppeteer and Cheerio:** Puppeteer and Cheerio are used to do automated web scraping, Puppeteer is capable of using JavaScript intense sites whereas Cheerio can give a quick static parse.
- **JWT (jsonwebtoken):** The secure stateless server side administration authentication [22].
- **Bcrypt:** Bcrypt is a way to be assured of security since passwords are hashed with salt generation as it becomes resistant to brute force attacks [23].

1.5.3 Artificial Intelligence

Credibility predictions Python-based ML services are included in FactCheckr:

- **Python 3.11:** Python runtime environment of the machine learning microservice.

- **Flask:** Lightweight framework that is used as an API endpoint in running the ML model.
- **scikit-learn:** The main library that was used in the implementation of the Logistic Regression model and TF-IDF vectorizer.
- **NLTK:** The Natural Language Toolkit was used in the task of preprocessing the text such as tokenization and removal of stop words.
- **Joblib:** A high-performance model serialization and loading.

1.5.4 Third-Party Services

Additional APIs and services will be connected to the platform in order to complement its functionality:

- **NewsAPI:** Gathers real-time content on more than 80,000 sources to provide the live news exploration functionality.
- **Docker:** Containerizing is used to frontend, backend and ML services to be deployed in environments that are similar.
- **Nginx:** Nginx is the front end webserver and reverse proxy of high performance serving of static files.

Chapter 2

Literature Review

Fake news has become a major problem in the information ecosystem in a digital world [1]. The high rate of content spreading via social media and online platforms have led to the necessity of a transition between the manual verification system to automated and AI-based ones. In this literature review, the comparisons are made between the existing methodologies and systems of misinformation detection and the usability of the suggested system FactCheckr.

A study proved that aided learning portal, especially ensemble algorithms, can be used to detect fake news with an accuracy of more than 90%, applied to the detection of linguistic characteristics and patterns of propagation [24]. Additionally, [5] investigated the methods of Natural Language Processing (NLP) and concluded that TF-IDF vectorization and logistic regression are a best match of accuracy and computational efficiency when used in real-time.

Although deep learning-based models such as BERT have demonstrated better performance, [25] has indicated that they have high computing demand that would not be suitable in resource-bound web applications as adopted by FactCheckr. Further, it is emphasized that multi-modal analysis is required since image manipulation very frequently goes hand-in-hand with the misinformation of text, which proves the correctness of the image integrity aspect included in this project [9].

2.1 Analysis

Evaluation of existing literature and systems shows that there is a clear anomaly between the high accuracy academic models and the available user tools. Some of the existing offerings can be divided into two groups: hand fact-checking organizations or sophisticated commercial enterprise solutions.

A professional journalist of the fact-checking process offers the greatest level of accuracy, but has severe problems with scalability with time lag up to hours to days [2]. On the other hand, almost all automated commercial functions such as the Logically AI provide speed, yet are usually paywalled and not easily accessible to ordinary citizens.

Recent studies highlighted the role of explainable AI, stating that the users tend to trust the systems that can explain their rationale [14]. Most currently available automated tools can be described as black box, which means

the score comes out-of-context. The solution is FactCheckr that features visualizations of importance. Moreover, even though such systems as ClaimBuster can be very effective at determining claims that are "check-worthy," they cannot examine the credibility of the information or profile image integrity. FactCheckr at least partially fills these gaps with a multi-dimensional analysis (Text, Source, Clickbait, and Image) in a free and easy to use web-based application.

2.2 Existing Systems

2.2.1 FactCheck.org

FactCheck.org is the non-profit organization which uses the help of professional journalists and fact-checks manually the statements and speeches of politicians, viral fake news. Its editorial process is strict and this is why it has been thought to be a Gold Standard of accuracy. It is limited to a large extent, however, in comparison with FactCheckr. It is totally human-based, and thus cannot handle news volumes generated on a daily basis. It is neither able to analyse arbitrary text or URLs typed in by a user in real-time, nor does it take anything like an hour to have a rumour spread and a verification article published.

Figure 2.1: FactCheck.org Website Interface

2.2.2 Full Fact (UK)

The UK fact-checking organization called Full Fact is an independent one and has started to use automation tools, along with human fact-checkers. They use AI to identify claims in sentences and compare them to an existing database of already fact-verified statements. Although Full Fact is a hybrid model, it is geographically restricted (UK-centric) and its tools are usually aimed at matching the claims, but not the analysis of the credibility of the unknown but newly published articles. It is not based publicly on the interpretation of arbitrary URLs or the integrity of images through perceptual hashing of the agenda like FactCheckr.

Figure 2.2: Full Fact Automated Tools

2.2.3 ClaimBuster

ClaimBuster is an academic initiative or rather a project by a university that aims at determining what they call as check-worthy factual claims in political transcripts through Natural Language Processing. It puts a point whether a sentence is more likely to have a factual statement or not. ClaimBuster will however only identify claims; it does not identify whether they are true or false. It is not as broad a multi-dimensional analysis as FactCheckr (Source, Image, Clickbait) and does not compare full news stories to be which their credibility goes all the way to the wire.

Figure 2.3: ClaimBuster Interface

2.3 FactCheckr Compared to Existing Systems

The FactCheckr system compares with proposed FactCheckr, FactCheck.org and ClaimBuster as seen in Table 2.1.

Table 2.1: Comparison of FactCheckr with Existing Systems

Feature	FactCheckr (Proposed)	FactCheck.org	ClaimBuster
Analysis Speed	Instant (200-500ms)	55 Hours to Days	Instant
Automation Level	Full-fledged Automated	55 Manual	Automated
Text Analysis	ML Credibility Prediction	Human Analysis	△ Claim Detection Only
URL Scraping	Automated (Cheerio/Puppeteer)	Manual Reading	55 Not Provided
Image Verification	Perceptual Hashing (pHash)	55 No mechanism	55 No mechanism
Source Checking	Trusted Domain Database	Extensive Manual Check	55 No Source Check
Availability	Free Web Application	Free Website	△ Academic / API
Explainability	Visuals on Feature Importance	Written Reports	△ Score Only
Real-time News	Live NewsAPI Support	55 Not Provided	55 Not Provided

Legend: = Feature Available, 55 = Not Available, △ = Limited/Partial Support

FactCheckr stands out in the sense that it is a multi-dimensional approach to the comparison as seen above. Although systems such as FactCheck.org have high authority, they are not fast and their scalability is comparable to that of an automated solution. In contrast, the automated tools such as ClaimBuster are regularly narrowed so to certain activities (claim spotting) as opposed to checking the entire article. FactCheckr stands out in its integration of the text analysis, source verification, clickbaiting, and image integrity into one platform and accessible to the general audience.

Chapter 3

Requirement Specification

3.1 Existing Systems

The existing fake news detectors broadly fall into three categories: manual fact-checking services, browser extensions, and simple academic prototypes[1]. Some, such as FactCheck.org, use manual services; thus, there is a significant lag between the spread of a claim and its verification. The use of browser extensions can be susceptible to not analyzing the content of new articles, relying instead on static domain lists. Although accurate, academic tools are often unavailable to the general public or lack accessible interfaces. These existing systems face several critical limitations:

- **Slow Fact-Checking:** Manual fact-checking can take hours or even days, by which time misinformation has often already gone viral.
- **Limited Scope:** Many tools disregard the broader context, such as image integrity and clickbait patterns, focusing only on specific claims.
- **Inadequate Accessibility:** Automated solutions are typically technical research projects that are not designed for non-technical users.
- **Lack of Explainability:** Existing tools often provide a score without offering reasons why an article was flagged as fake[15].

3.2 Proposed System

FactCheckr proposes a Web-based, multifaceted platform to automate the credibility appraisal of content of news. The system introduces an Au tomated Analysis Engine with easy to use interface for instant text and URL verifications. One of the innovations is the Hybrid Scoring System that incorporates Machine Learning (Logistic Regression)[6], Source Verification, Clickbait Detection[10], and Image Integrity Analysis (perceptual hashing)[11] into one comprehensive score. The system helps in improving user experience through Live News Exploration, supported by NewsAPI, and marketing Explainable AI visualizations[15].

Figure 3.1: Proposed System Architecture: Showing the interaction between the Frontend, Analysis Engine, and External APIs.

3.3 Requirements Specifications

3.3.1 Functional Requirements

User Functional Requirements The table below outlines the requirements for the General User module.

Table 3.1: User Functional Requirements

Code	Name	Description
FR1	Text Analysis	Users must be able to input raw text (up to 50,000 characters). The system must preprocess and analyze it using the ML model.
FR2	URL Analysis	Users must be able to submit an article URL. The system must scrape content, verify the source, and analyze text and images.
FR3	Live News Feed	Users must be able to view trending news articles fetched from NewsAPI, with automatic credibility scores displayed.
FR4	History Tracking	Users must be able to view a chronological history of their past analyses, filterable by result (Real/Fake).
FR5	Explanation View	Users must be able to view detailed explanations for predictions, including feature importance charts and component score breakdowns.
FR6	Dashboard Analytics	Users must be able to view statistical summaries of their analysis history, including accuracy trends and source reliability charts.

Administrator Functional Requirements The table below outlines the requirements for the Administrator module.

Table 3.2: Administrator Functional Requirements

Code	Name	Description
FR7	Admin Login	Admins must be able to securely log in using a username and password, authenticated via JWT[22].

Table 3.2: Administrator Functional Requirements

Code	Name	Description
FR8	Source Management	Admins must be able to add, edit, or delete domains in the Trusted Sources database and assign reputation scores.
FR9	Scoring Configuration	Admins must be able to adjust the weights for the Hybrid Scoring algorithm (ML, Source, Clickbait, Image) via a slider interface.
FR10	System Monitoring	Admins must be able to view system health statistics and total analysis counts.

3.3.2 Non-Functional Requirements

The following table defines the quality attributes and performance constraints.

Table 3.3: Non-Functional Requirements

Code	Name	Description
NFR1	Performance	Text analysis must complete within 500ms. URL scraping (Cheerio) within 2s, and ML inference within 150ms.
NFR2	Reliability	System uptime should be 99%. The system must handle scraping failures gracefully without crashing.
NFR3	Security	Passwords must be hashed (Bcrypt)[23]. API endpoints must be protected against SQL injection and XSS.
NFR4	Usability	The interface must be responsive (mobile/desktop) and intuitive for non-technical users.
NFR5	Scalability	The architecture must support 100 concurrent users and 1000 analyses per hour.
NFR6	Portability	The system must be containerized using Docker to ensure consistent deployment across environments.

3.4 Software Requirements

Development Environment

- **IDE:** VS Code, WebStorm, or similar.
- **Runtime:** Node.js 20.x, Python 3.11+.

- **Containerization:** Docker 24.x & Docker Compose.
- **Version Control:** Git.

Frontend Stack

- **Framework:** React 18.2.0[20].
- **Build Tool:** Vite 4.4.5.
- **Styling:** Vanilla CSS, Lucide React Icons.
- **Visualization:** Chart.js + react-chartjs-2.

Backend & ML Stack

- **Server Framework:** Express.js 4.18.2.
- **ML Service:** Flask 3.0.0, scikit-learn 1.3.0, NLTK.
- **Database:** SQLite with Prisma ORM 5.22.0[26].
- **Authentication:** JWT, Bcrypt.

3.5 Hardware Requirements

- **CPU:** 2 Cores minimum (4 Cores recommended for concurrent scraping).
- **RAM:** 4GB minimum (ML model + Puppeteer require significant memory).
- **Storage:** 10GB SSD for database and container images.
- **Network:** Stable broadband connection.

3.6 System Use Case Diagrams

A use case diagram is a visual representation of the functionalities of a system and how users interact with it.

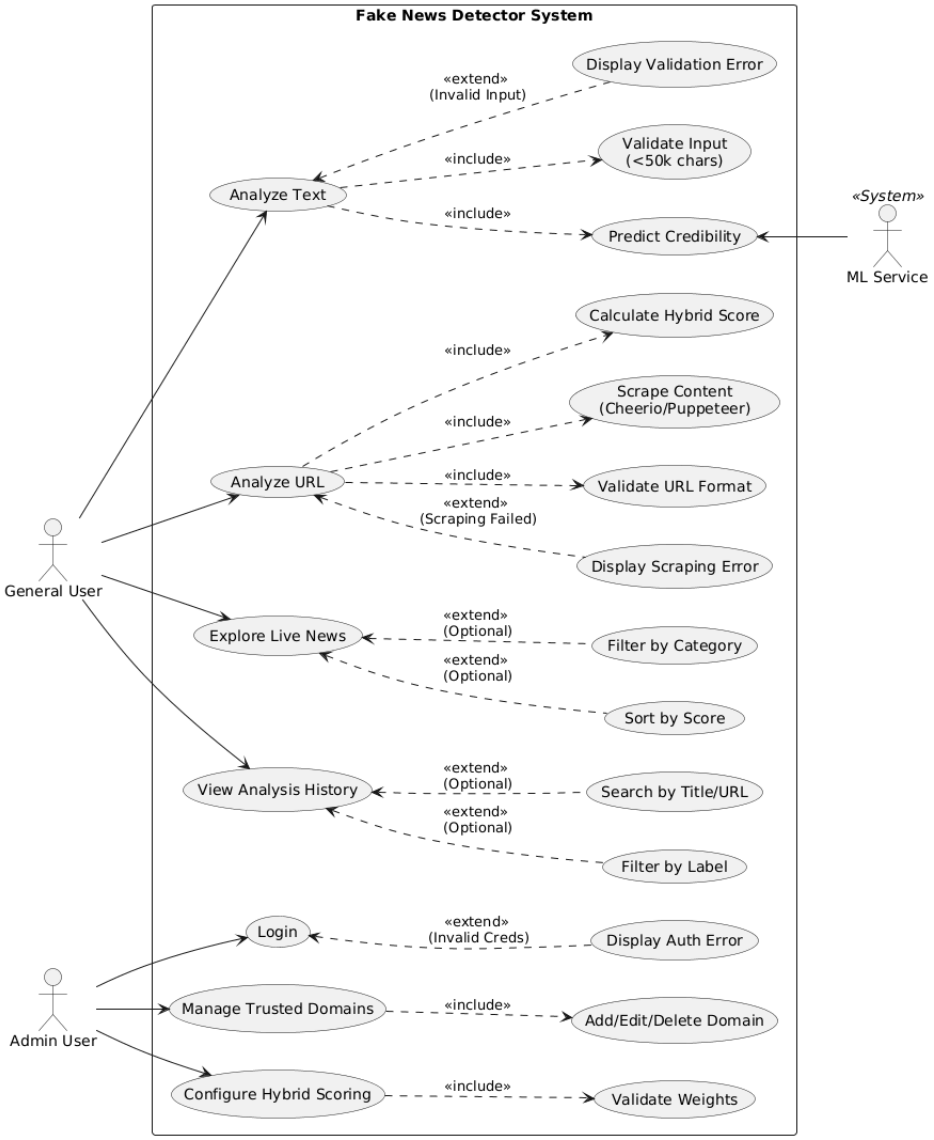

Figure 3.2: ClaimBuster Interface

3.7 Use Cases

Figure 3.3: Flowchart of the Hybrid Scoring Logic (Use Case 002)

Table 3.4: Use Case-001: Analyze Text

Use Case ID	Use Case-001
Name	Analyze Text
Actors	General User, ML Service
Description	Allows a user to verify the credibility of raw text content.
Precondition	User is on the "Analyze Text" page.
Postcondition	Credibility score and explanation are displayed and saved to history.
Normal Flow	1. User enters text into the input area. 2. User clicks "Analyze". 3. System validates text length. 4. System sends text to ML Service. 5. ML Service preprocesses and predicts credibility. 6. System displays result (Real/Fake) and confidence score.
Exception	E1: Input empty: System shows "Input cannot be empty". E2: Service offline: System shows "Analysis service unavailable".

Figure 3.4: Process Flow for Image Integrity Checking using Perceptual Hashing

Table 3.5: Use Case-002: Analyze URL

Use Case ID	Use Case-002
Name	Analyze URL
Actors	General User, Scraper, ML Service
Description	Verifies a news article via its URL using hybrid analysis.
Precondition	User has a valid news article URL.
Postcondition	Full hybrid analysis (Text, Source, Image, Clickbait) is displayed.
Normal Flow	1. User inputs URL and clicks "Analyze". 2. System scrapes article content (Title, Text, Images). 3. System performs ML text analysis. 4. System checks Source Reputation database. 5. System calculates Clickbait and Image Integrity scores. 6. System computes and displays weighted Hybrid Score.
Exception	E1: Invalid URL: System prompts for correct format. E2: Scraping blocked: System asks user to paste text manually.

Table 3.6: Use Case-003: Explore Live News

Use Case ID	Use Case-003
Name	Explore Live News
Actors	General User, NewsAPI
Description	Displays trending news articles with pre-calculated credibility scores.
Precondition	NewsAPI service is active.
Postcondition	User views a grid of verified news articles.
Normal Flow	1. User navigates to "Explore" page. 2. System fetches latest headlines from NewsAPI. 3. System automatically analyzes content of fetched articles. 4. System displays articles with "Real" or "Fake" badges. 5. User filters by category (e.g., Tech, Health).
Exception	E1: API Rate Limit: System shows cached news or error message.

Table 3.7: Use Case-004: View Analysis History

Use Case ID	Use Case-004
Name	View Analysis History
Actors	General User, Database
Description	Allows user to review past analyses.
Precondition	User has performed at least one analysis.
Postcondition	Chronological list of past results is shown.
Normal Flow	1. User navigates to "History" page. 2. System queries local database for saved analyses. 3. System displays list sorted by date. 4. User filters by Label (Real/Fake). 5. User clicks an item to view full details.
Exception	E1: No history: System shows "No analyses found".

Table 3.8: Use Case-005: Admin Login

Use Case ID	Use Case-005
Name	Admin Login
Actors	Administrator, System
Description	Secure authentication for administrative access.
Precondition	User is on Admin Login page.
Postcondition	Admin is authenticated and redirected to Dashboard.
Normal Flow	1. Admin enters username and password. 2. Admin clicks "Login". 3. System verifies credentials against database. 4. System generates and returns JWT. 5. System redirects to Admin Panel.
Exception	E1: Wrong password: System shows "Invalid credentials".

Table 3.9: Use Case-006: Manage Trusted Domains

Use Case ID	Use Case-006
Name	Manage Trusted Domains
Actors	Administrator, Database
Description	Add or update news sources in the trusted database.
Precondition	Admin is logged in.
Postcondition	Domain database is updated.
Normal Flow	1. Admin navigates to "Manage Sources". 2. Admin selects "Add Domain". 3. Admin writes into domain URL as well as Trust Score (0-100). 4. Admin saves changes. 5. System updates the database.
Exception	E1: Duplicate domain: System shows error "Domain already exists".

Table 3.10: Use Case-007: Configure Hybrid Scoring

Use Case ID	Use Case-007
Name	Configure Hybrid Scoring
Actors	Administrator, System
Description	Adjusts the weights of the scoring components (ML, Source, etc.).
Precondition	Admin is logged in.
Postcondition	New scoring logic is applied to future analyses.
Normal Flow	1. Admin is then taken to Scoring Config. 2. ML, Source, Clickbait, Image Weights Have sliders adjusted by the Admin. 3. Admin clicks "Save". 4. System We validate total = 100%. 5. System stores new configuration.
Exception	E1: Invalid weights: System prompts to ensure total is 100%.

Table 3.11: Use Case-008: View Analytics

Use Case ID	Use Case-008
Name	View Analytics
Actors	General User, System
Description	Displays visual statistics of analysis history.
Precondition	User has analysis history.
Postcondition	Charts and graphs are displayed.
Normal Flow	1. User navigates to "Dashboard". 2. Analysis data is fetched by the system. 3. System draw Pie Chart (Real vs Fake). 4. System draws Line Chart (Activity over time). 5. User filters by date range.
Exception	E1: Data load error: System shows empty charts.

Table 3.12: Use Case-009: Check Image Integrity

Use Case ID	Use Case-009
Name	Check Image Integrity
Actors	System, Image Hash Service
Description	Automatically checks images for manipulation during URL analysis.
Precondition	URL analysis is in progress.
Postcondition	Image integrity score is returned.
Normal Flow	1. System extracts images from URL. 2. System generates pHash for each of the images. 3. System compares hash with fake image database that is known to it. 4. We have that system calculated the hamming distance. 5. System labels images which are very similar to fakes.
Exception	E1: Image load fail: System skips image analysis.

Table 3.13: Use Case-010: View Explainability Report

Use Case ID	Use Case-010
Name	View Explainability Report
Actors	General User, System
Description	Shows why a specific prediction was made using feature importance.
Precondition	Text or URL analysis is complete.
Postcondition	Visual explanation of the score is displayed.
Normal Flow	1. User expands "Why this result?" section. 2. System shows topping words contributing to score on "Real". 3. System shows best words that contribute to "Fake" score. 4. System shows breakdown of the Hybrid Score components (Source, Text, Image). 5. Prediction logic is understood by the user.
Exception	E1: No features: System shows "Insufficient text for explanation".

Table 3.14: Use Case-011: Filter Live News

Use Case ID	Use Case-011
Name	Filter Live News
Actors	General User, NewsAPI
Description	Filters trending articles by category (Tech, Health, Politics).
Precondition	User is on the Live News Explore page.
Postcondition	News grid updates to show only selected category.
Normal Flow	1. User clicks on category tab (e.g. "Health"). 2. System asks for any particular category data from NewsAPI. 3. System has rapid credibility check on new articles. 4. System outputs articles with removed badges.
Exception	E1: API Error: System shows "Category unavailable".

Table 3.15: Use Case-012: View Source Credibility

Use Case ID	Use Case-012
Name	View Source Credibility
Actors	General User, Database
Description	Detailed info about the publisher's reputation.
Precondition	URL Analysis is complete.
Postcondition	Domain trust score and category are displayed.
Normal Flow	1. User Hovers over Source Score Badge 2. System obtains Domain information from Trusted Database. 3. System displays Trust Score (0-100) and Category (e.g. "Satire"). 4. Justification is viewed by the user.
Exception	E1: Unknown Domain: System displays "Source not in database".

Table 3.16: Use Case-013: Clear Analysis History

Use Case ID	Use Case-013
Name	Clear Analysis History
Actors	General User, Local Storage
Description	Removes all saved analysis records from the user's device.
Precondition	User has history items.
Postcondition	History list is empty.
Normal Flow	1. User adjusts to History page. 2. User clicks "Clear All". 3. System for asking for confirmation. 4. User confirms. 5. System deletes records from local database system.
Exception	None.

Table 3.17: Use Case-014: View Methodology

Use Case ID	Use Case-014
Name	View Methodology
Actors	General User, System
Description	Explains how the hybrid scoring system works.
Precondition	None.
Postcondition	User understands the scoring weights.
Normal Flow	1. User clicks "How it Works" link. 2. System displays breakdown of ML, Source, Clickbait, and Image components. 3. System shows current active weights.
Exception	None.

Table 3.18: Use Case-015: Admin Login

Use Case ID	Use Case-015
Name	Admin Login
Actors	Administrator, System
Description	Authenticates administrative access to the backend.
Precondition	User is on Admin route.
Postcondition	Admin Dashboard is displayed.
Normal Flow	1. Known by username and password, username/password combination values are entered by Admin. 2. Password hashing using Bcrypt is carried out by this system. 3. System verifies the credentials against database. 4. System issues JWT token. 5. Admin gets redirected to Dashboard
Exception	E1: Invalid Login: System shows error message.

Table 3.19: Use Case-016: Admin View System Stats

Use Case ID	Use Case-016
Name	Admin View System Stats
Actors	Administrator, Database
Description	Displays overview of system usage and accuracy.
Precondition	Admin is logged in.
Postcondition	Statistics cards are displayed.
Normal Flow	1. Admin navigates to "Overview". 2. System queries total analyses count. 3. System calculates Real/Fake ratio. 4. System displays counts and recent activity log.
Exception	E1: DB Connection Error: Stats show as zero.

Table 3.20: Use Case-017: Add Trusted Domain

Use Case ID	Use Case-017
Name	Add Trusted Domain
Actors	Administrator, Database
Description	Adds a new news source to the verification database.
Precondition	Admin is on Manage Sources page.
Postcondition	New domain is available for verification.
Normal Flow	1. Admin clicks "Add Domain". 2. Admin enters URL (e.g., "bbc.com"). 3. Admin assigns Trust Score (e.g., 100). 4. Admin clicks Save. 5. System insert record to SQLite database.
Exception	E1: Duplicate: System alerts that domain exists.

Table 3.21: Use Case-018: Edit Domain Score

Use Case ID	Use Case-018
Name	Edit Domain Score
Actors	Administrator, Database
Description	Updates the reputation score of an existing source.
Precondition	Admin selects a domain from list.
Postcondition	Trust score is updated.
Normal Flow	1. Admin locates domain in list. 2. Admin changes score value. 3. Admin clicks "Update". 4. System modifies the record.
Exception	None.

Table 3.22: Use Case-019: Delete Trusted Domain

Use Case ID	Use Case-019
Name	Delete Trusted Domain
Actors	Administrator, Database
Description	Removes a source from the trusted list.
Precondition	Admin is on Manage Sources page.
Postcondition	Domain is removed.
Normal Flow	1. Admin clicks "Delete" icon on a domain row. 2. System requests confirmation. 3. Admin confirms. 4. System deletes the record.
Exception	None.

Table 3.23: Use Case-020: Configure Scoring Weights

Use Case ID	Use Case-020
Name	Configure Scoring Weights
Actors	Administrator, System
Description	Adjusts the influence of ML, Source, Clickbait, and Image scores.
Precondition	Admin is on Configuration page.
Postcondition	New scoring algorithm is active.
Normal Flow	1. Admin tweaks sliders (e.g. increase ml weight). 2. Admin clicks "Save Config". 3. System validates total is 100%. 4. System Saves new weights to database.
Exception	E1: Invalid Total: System forces weights to sum to 100.

Table 3.24: Use Case-021: Cancel Analysis

Use Case ID	Use Case-021
Name	Cancel Analysis
Actors	General User, System
Description	Stops a long-running scraping or analysis process.
Precondition	Analysis is in progress (Loading state).
Postcondition	Process aborts and returns to input.
Normal Flow	1. User clicks "Cancel" button during loading. 2. System aborts HTTP request to backend. 3. UI resets to input state.
Exception	None.

Table 3.25: Use Case-022: Export Analysis Result

Use Case ID	Use Case-022
Name	Export Analysis Result
Actors	General User, System
Description	Saves the verification report as a PDF or image.
Precondition	Analysis results are displayed.
Postcondition	File is downloaded to user device.
Normal Flow	1. User clicks "Share/Export". 2. User selects "Download PDF". 3. System generates PDF with score and charts. 4. Browser initiates download.
Exception	E1: Generation Error: System shows "Export failed".

Table 3.26: Use Case-023: Check Image Integrity

Use Case ID	Use Case-023
Name	Check Image Integrity
Actors	System, Image Hash Service
Description	Verifies if images in an article are reused or manipulated.
Precondition	URL scraping extracted images.
Postcondition	Image integrity score is computed.
Normal Flow	1. System generates perceptual hash (dHash) for image 2. System asked hash database for simple matches. 3. We have that system calculated the hamming distance. 4. System flags possible fakes (¿95% match).
Exception	E1: Image Load Fail: Image is skipped.

Table 3.27: Use Case-024: Admin Logout

Use Case ID	Use Case-024
Name	Admin Logout
Actors	Administrator, System
Description	Securely ends the administrative session.
Precondition	Admin is logged in.
Postcondition	User redirected to Login page.
Normal Flow	1. Admin clicks "Logout" in sidebar. 2. System invalidates or removes local JWT. 3. System redirects to Login screen.
Exception	None.

Table 3.28: Use Case-025: Search Analysis History

Use Case ID	Use Case-025
Name	Search Analysis History
Actors	General User, Database
Description	Finds a past analysis by title or keyword.
Precondition	User is on History page.
Postcondition	Filtered list is displayed.
Normal Flow	1. User enters keyword in search bar. 2. System filters local history records. 3. System displays matching analysis cards.
Exception	E1: No results: System shows "No matches found".

Table 3.29: Use Case-026: View Scraping Error

Use Case ID	Use Case-026
Name	View Scraping Error
Actors	General User, System
Description	Handling failures when a URL cannot be scraped.
Precondition	Scraping process fails (e.g., paywall).
Postcondition	User is prompted to use Text Analysis.
Normal Flow	1. Scraper detects block/failure. 2. System says "Could not read URL content" 3. System suggests "Try investing the text manually". 4. The user gets redirected at Text Input.
Exception	None.

Table 3.30: Use Case-027: View Cached Results

Use Case ID	Use Case-027
Name	View Cached Results
Actors	General User, Database
Description	Serving previously analyzed URLs instantly.
Precondition	URL was analyzed recently.
Postcondition	Results displayed without new scraping.
Normal Flow	1. User submits a URL. 2. System checks the database for recent entry (Same URL). 3. System searches for saved analysis. 4. System immediate return of result is after skipping scraping.
Exception	None.

Table 3.31: Use Case-028: Admin View Domain Details

Use Case ID	Use Case-028
Name	Admin View Domain Details
Actors	Administrator, Database
Description	Inspects details of a trusted source entry.
Precondition	Admin is on Manage Sources page.
Postcondition	Details modal is displayed.
Normal Flow	1. Admin clicks on a domain row. 2. System displays URL, Trust Score, and Category. 3. System shows date added.
Exception	None.

Table 3.32: Use Case-029: Sort Live News

Use Case ID	Use Case-029
Name	Sort Live News
Actors	General User, System
Description	Orders news articles by credibility or date.
Precondition	Live News page is populated.
Postcondition	Articles are reordered.
Normal Flow	1. User clicks Sort dropdown. 2. User selects "Most Credible" or "Latest". 3. System reorders the article grid accordingly.
Exception	None.

Table 3.33: Use Case-030: Admin Reset Weights

Use Case ID	Use Case-030
Name	Admin Reset Weights
Actors	Administrator, System
Description	Restores hybrid scoring weights to default values.
Precondition	Admin is on Configuration page.
Postcondition	Weights reset to default.
Normal Flow	1. Admin clicks "Reset to Defaults". 2. System sets ML=70, Source=20, Clickbait=20, Image=15. 3. Admin saves configuration.
Exception	None.

Table 3.34: Use Case-031: Filter History by Label

Use Case ID	Use Case-031
Name	Filter History by Label
Actors	General User, System
Description	Shows only "Real" or "Fake" results in history.
Precondition	User is on History page.
Postcondition	Filtered list is shown.
Normal Flow	1. User clicks Filter dropdown. 2. User selects "Fake News Only". 3. System hides "Real" results from view.
Exception	None.

Table 3.35: Use Case-032: Search Trusted Domains

Use Case ID	Use Case-032
Name	Search Trusted Domains
Actors	Administrator, System
Description	Finds a specific domain in the admin list.
Precondition	Admin is on Manage Sources page.
Postcondition	Specific domain is highlighted.
Normal Flow	1. Admin enters domain name in search box. 2. System filters the table rows. 3. Admin views the trust score for that domain.
Exception	None.

Table 3.36: Use Case-033: View Clickbait Details

Use Case ID	Use Case-033
Name	View Clickbait Details
Actors	General User, System
Description	Shows which words triggered the clickbait detection.
Precondition	Analysis complete with Clickbait detected.
Postcondition	Trigger words are highlighted.
Normal Flow	1. User hovers over Clickbait Score. 2. System displays list of triggers (e.g., "SHOCKING", "!!!"). 3. User sees contribution to score.
Exception	None.

Table 3.37: Use Case-034: Admin Monitor API Usage

Use Case ID	Use Case-034
Name	Admin Monitor API Usage
Actors	Administrator, System
Description	Checks NewsAPI quota usage.
Precondition	Admin is on Dashboard.
Postcondition	Usage stats displayed.
Normal Flow	1. Admin views "System Health" card. 2. System shows requests made vs daily limit (e.g., 50/100). 3. Admin assesses need for upgrade.
Exception	None.

Table 3.38: Use Case-035: View Accuracy Metrics

Use Case ID	Use Case-035
Name	View Accuracy Metrics
Actors	General User, System
Description	Displays the model's validation accuracy.
Precondition	User is on Dashboard/About page.
Postcondition	Model accuracy (92.5%) is shown.
Normal Flow	1. User scrolls to "System Performance". 2. System displays chart of Model Accuracy. 3. User sees Precision and Recall values.
Exception	None.

Table 3.39: Use Case-036: Bookmark Article

Use Case ID	Use Case-036
Name	Bookmark Article
Actors	General User, Local Storage
Description	Saves a live news article for later reading.
Precondition	User is on Live News page.
Postcondition	Article saved to bookmarks.
Normal Flow	1. User clicks Bookmark icon on news card. 2. System saves article metadata to local storage. 3. Icon changes to solid/filled state.
Exception	None.

Table 3.40: Use Case-037: Remove Bookmark

Use Case ID	Use Case-037
Name	Remove Bookmark
Actors	General User, Local Storage
Description	Un-saves a news article.
Precondition	Article is bookmarked.
Postcondition	Article removed from bookmarks.
Normal Flow	1. User clicks Bookmark icon again. 2. System removes record from storage. 3. Icon reverts to outline state.
Exception	None.

Table 3.41: Use Case-038: Delete History Item

Use Case ID	Use Case-038
Name	Delete History Item
Actors	General User, Local Storage
Description	Removes a single analysis from history.
Precondition	User is on History page.
Postcondition	Item is removed.
Normal Flow	1. User clicks "Delete" (trash icon) on specific item. 2. System removes item from list. 3. System updates local storage.
Exception	None.

Table 3.42: Use Case-039: System Auto-Scrape

Use Case ID	Use Case-039
Name	System Auto-Scrape
Actors	System, NewsAPI
Description	Background process to check trending news credibility.
Precondition	User opens Explore page.
Postcondition	Articles displayed with pre-computed scores.
Normal Flow	1. App fetches new articles. 2. System iterates through articles. 3. System runs lightweight analysis (Title/Source only). 4. System assigns preliminary score/badge.
Exception	None.

Chapter 4

System Design

4.1 System Overview

FactCheckr is a three-level web-based application, which is to be contemporary and be developed in a microservices architecture[18] to assure scalability, maintenance, and the ability to scale resources-intensive aspects autonomously. There are four distinct layers that make up this system and which work in harmony to provide real-time credibility testing.

Presentation Layer interacts with the User Interface (UI) using React 18[20] and Vite with a component-based architecture, which relies on the principles of Material Design that ensure the responsive experience of desktop and mobile users. Below this solution is the Business Logic Layer which is based on express.js and node.js[21] and it is the part that structures the flow of the application, authentication and data processing. Data Layer In SQLite Data Layer using Prisma ORM[26] manages the idea of persistence, which defines the ability to do database operations in a type-safe manner. Finally, the ML Service Layer is a specialist microservice that serves to run a Logistic Regression model[6] and receive the explanations of the importance of the features, which does not depend on the main application thread.

4.2 FactCheckr System Architecture

FactCheckr platform adheres to decoupled client-server. It makes use of Single Page Application (SPA) as the frontend which communicates with the backend via REST APIs. The server serves as the center of interest to pass through requests between the client and the database as well as the third-party machine learning service.

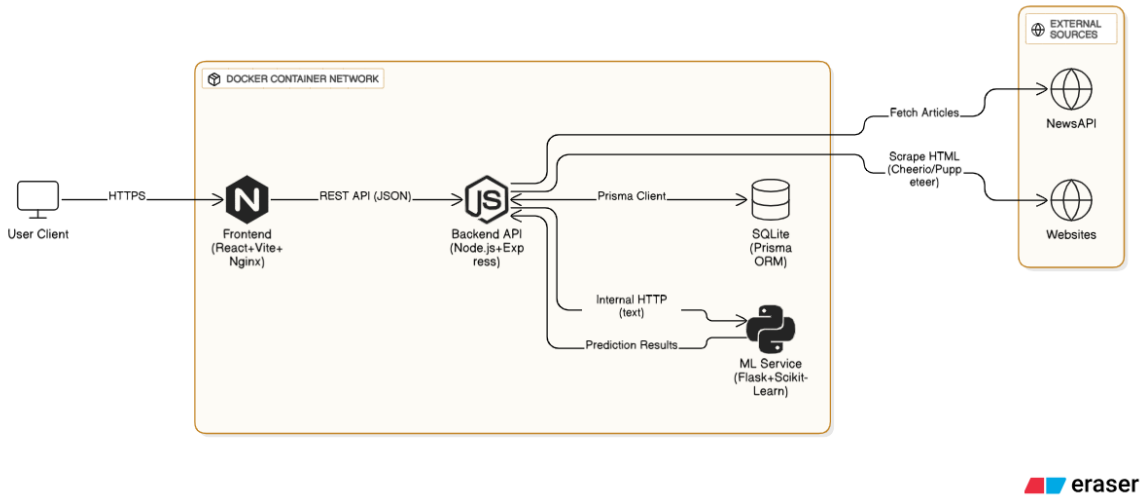

Figure 4.1: FactCheckr System Architecture Diagram

The architecture focuses on use of strict separation of concern:

- **Frontend Container:** Nginx controlled, with the support of static React files.
- **Backend Container:** Node.js/Express environment for handling of business logic, web scraping with Puppeteer/Cheerio and trusted sources verification.
- **ML Container:** Python/Flask service that serves the scikit-learn model and TF-IDF vectorizer and it can be reached by a dedicated Internet Protocol (IP) endpoint.
- **Database:** SQLite file storage provides data persistence at the least configuration overhead, and it addresses the project portability needs.

4.3 Design Constraints

The FactCheckr platform is conceptualized by certain technical and operational constraints to determine the breadth and scope of the platform.

- **Technical Limitations:** Although having the English language is still compatible with the system at the moment, the training data used in the

creation of the ML model is English. Machine learning model (including the vectorizer) should be accommodated in existing memory (say 10MB), and inference should take not more than 150ms to make the user interface responsive.

- **Scraping Constraints:** The process of automated scraping is very dependent on the structure of the target webpage. Other news providers employ anti-bot methods, potentially blocking Puppeteer; a fallback system relying on Cheerio is employed but not able to render JavaScript heavy websites.

- **Resource Constraints:** Minimal RAM of 2GB is needed to load the necessary libraries (Pandas, NLTK) and models on the ML service. The NewsAPI underlines the exploration of live news and has the rate limit of 100 requests per day on the free tier.

- **Security Constraints:** Only stateless JWT authentication[22] is used on administrative activity with an expiration time of 24 hours. The input text is checked in order to avoid XSS attacks and the operations with writing database are performed in a sequence to control the ACID.

4.4 Design Methodology

The Agile Development Methodology is applied in the project, and 2-week sprints are used to allow one to constantly test and improve the hybrid scoring algorithm.

Phase 1: Architecture and Data (Sprint 1-2)
A layout of the microservices and a definition of a Prisma database schema, as well as pre-processing the data to use it in training the ML model.

Phase 2: ML Service Implementation (Sprint 3)
Making the Logistic Regression model, creating Flask API and extracting the importance of features.

Phase 3: Backend Core (Sprint 4-5)
The adoption of the Express.js routes, Cheerio and Puppeteer scrapers, as well as the implementation of the Source Verification logic.

Phase 4: Frontend (Sprint 6-7)

Development of the React interface, addition of Chart.js to enable visualization of the data and link the interface to the backend APIs.

Phase 5: Integration and Deployment (Sprint 8)

All services dockerization, configuration of the Nginx, and the last system testing.

4.5 High Level Architecture

The high-level architecture explains how data follows in a sequence when a verification request is being taken.

4.5.1 Activity Diagram: Analysis of a URL

The analysis of the URL process is shown in the following diagram. It initially checks the URL, tries to scrape the content, then checks on the source domain against the Trusted Database, runs the ML model on the text, calculates the scores of clickbait and establishes the image integrity with the help of pHash[11].

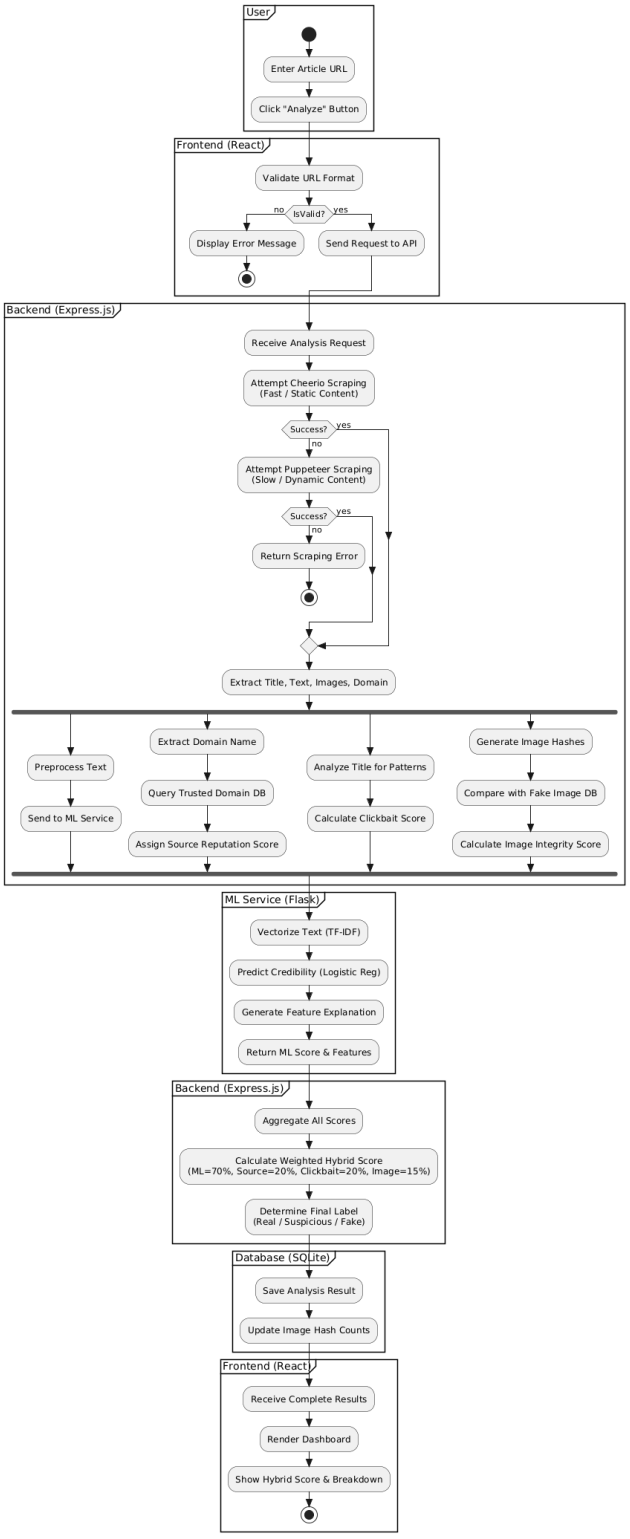

Figure 4.2: URL Analysis Activity Diagram

4.6 Low Level Architecture

The low-level architecture determines relations and structures of the data in the SQLite database.

4.6.1 ER Diagram

The database model is controlled through Prisma ORM and has four major entities, Analysis, TrustedDomain, ImageHash and User (Admin).

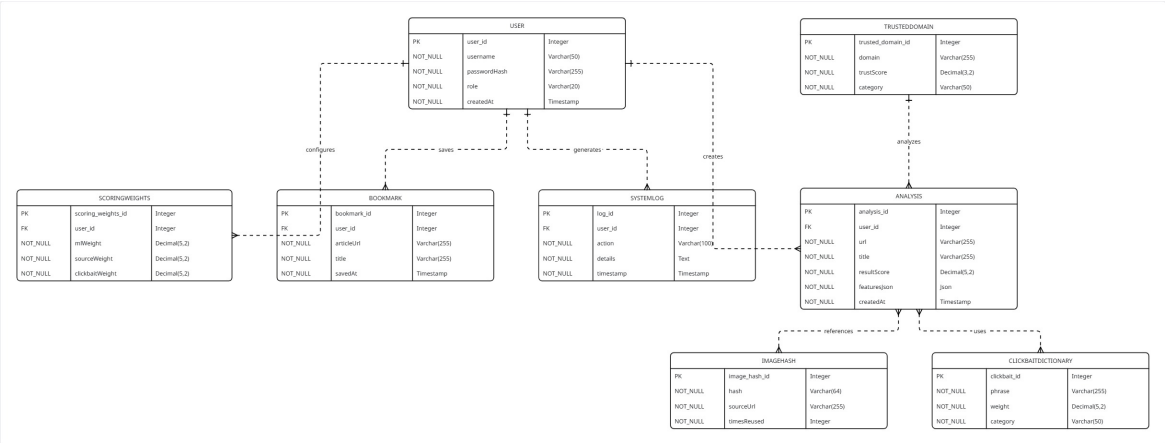

Figure 4.3: Entity Relationship (ER) Diagram

The most important relationships found within the system are:

- **Many-to-One:** There is a one to many relationship between the multiple Analysis records and one TrustedDomain, through the source URL.
- **One-to-Many:** The same ImageHash may be used in several articles and the system could trace the frequency of repeated use of particular picture in the fake news.
- **Many-to-One:** ScoringWeights settings are recorded as tracked by the User (Admin) who changed the settings.

4.6.2 Sequence Diagram

The functioning logic of FactCheckr platform is accurately depicted with the use of a sequence diagram, which represents the Low-Level, time-based outlook of the end-to-end procedure of analyzing the article. The diagram represents important dynamic relationships between the most important architectural elements, i.e., the User, Frontend, Backend, Scraper Service, ML Service, and the Database (DB). This flow starts with the User providing a URL to the Backend which validates the URL and sends a command to the scraper service to retrieve the text, images and domain of the article.

After the extraction of data, the sequence then identifies the important step of parallel analysis, where the Backend receives the text, and simultaneously sends the text to the ML Service with probabilistic scoring applied to the text, as well as the domain and image data to the local Database with corresponding Trust and Integrity Scores. Once all the component scores have been returned, the Hybrid Scoring Engine a operates in the Backend to vibrate the weighted aggregate basing on the parameters established by the Admin to produce the final credibility verdict. This concluding decision together with the elaborate user explainability or data, is then returned to the User to be presented and the cycle of verification is complete.

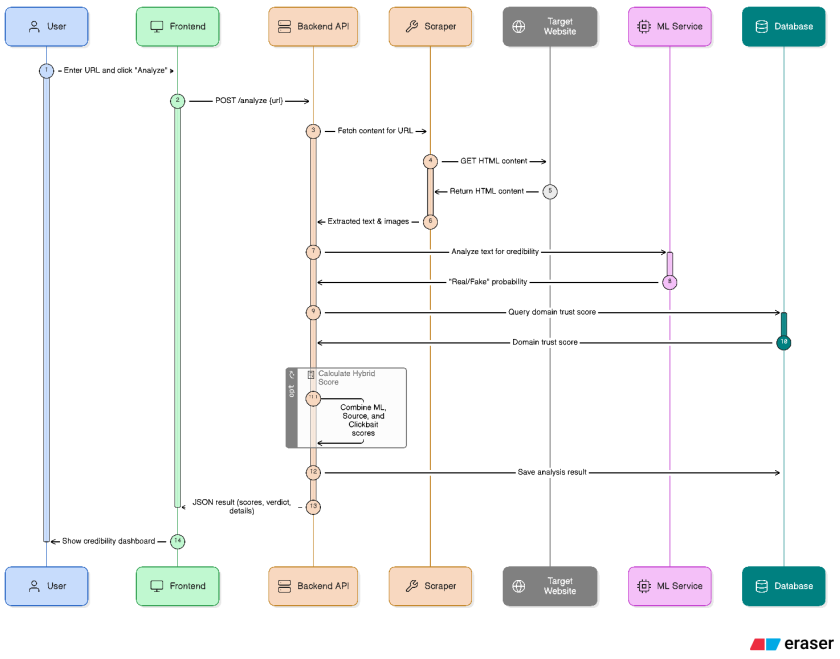

Figure 4.4: Sequence Diagram

4.7 Class Diagram

The following class diagram showcases the structure of FactCheckr and its classes, methods, attributes and the relationship between each class.

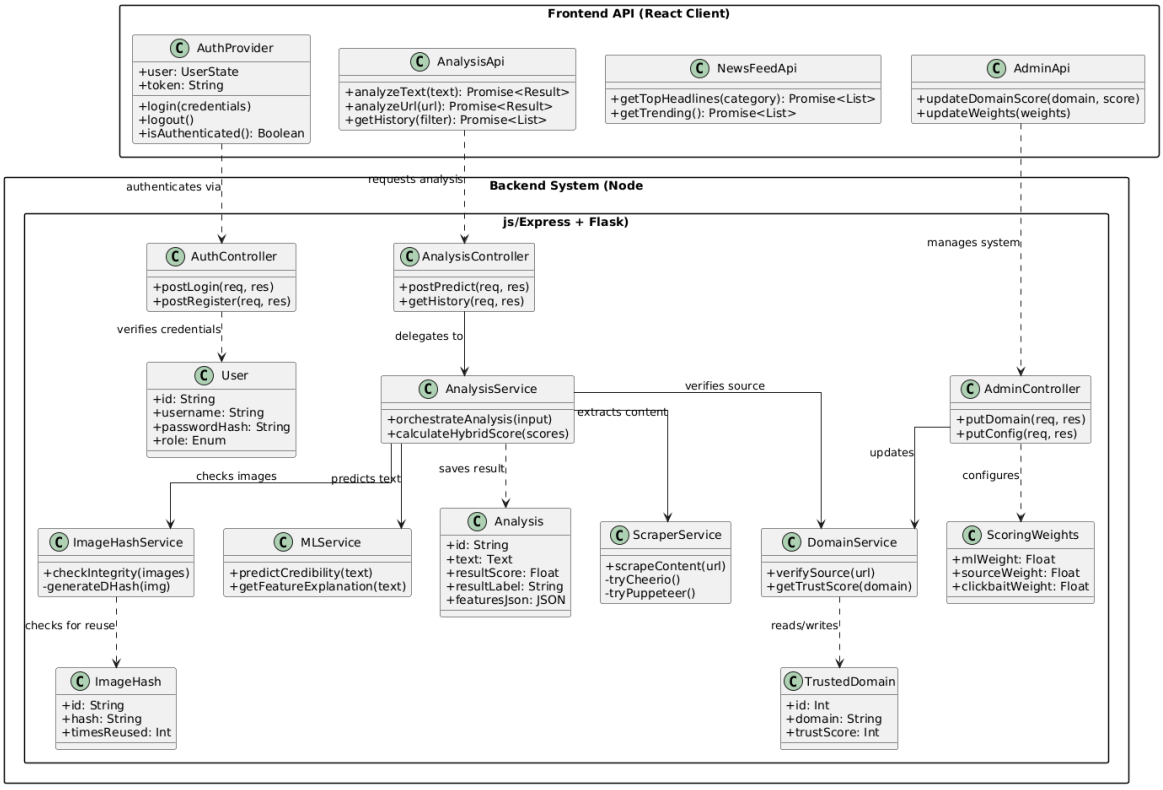

Figure 4.5: Entity Relationship (ER) Diagram

4.8 GUI Design

FactCheckr platform uses a user-friendly and clean Graphical User Interface (GUI) made easy to use aimed at instilling trust. The design aims at making complex analysis results easily digestible, scannable in order to reduce cognitive load.

4.8.1 Landing Page

It is the main point of entry of the user and includes a higher-ranking URL input box and an Analyze button that launches the verification process. A dynamic news feed is also available on the page and provides trending articles with preliminary credibility badges.

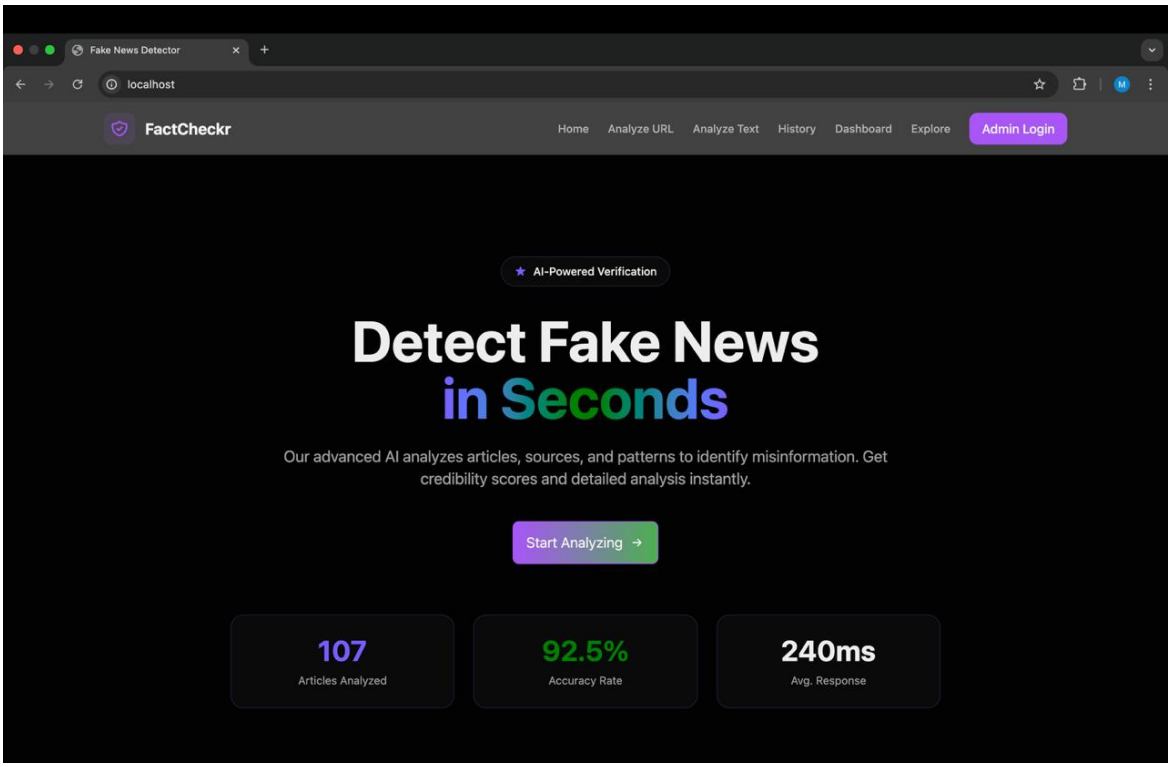

Figure 4.6: Landing Page

4.8.2 Analyze Article Text View

This screen has a fallback text area, to which an end user can be able to manually paste an article content. This will make the system still be useful in case the automated scraping service (Section 4.2.3) may fail to perform the task of scraping the text of a complex site.

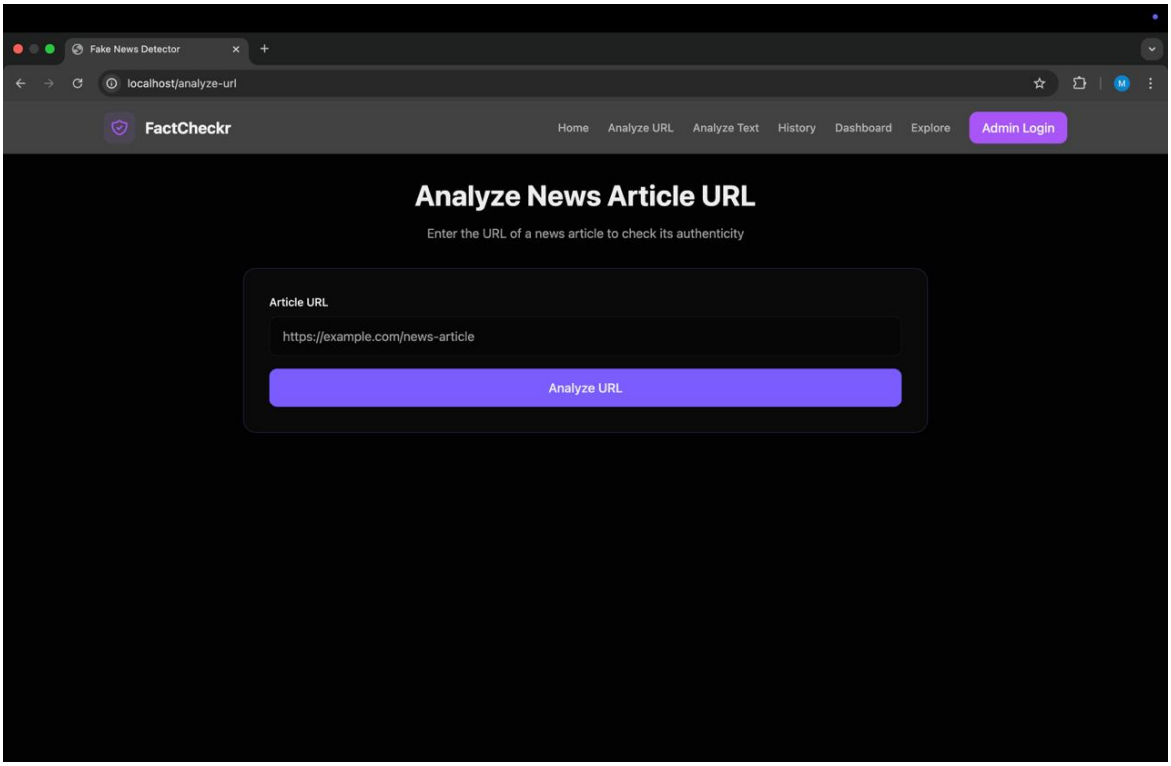

Figure 4.7: Analyze Text

4.8.3 LD-Real Time Analysis Loading Screen

This screen performs management of the expectations of the users in the asynchronous analysis. It shows a loading screen, and informs its user about the progress of the apris (e.g., Running ML Inference) to show that the multi-step verification pipeline is running.

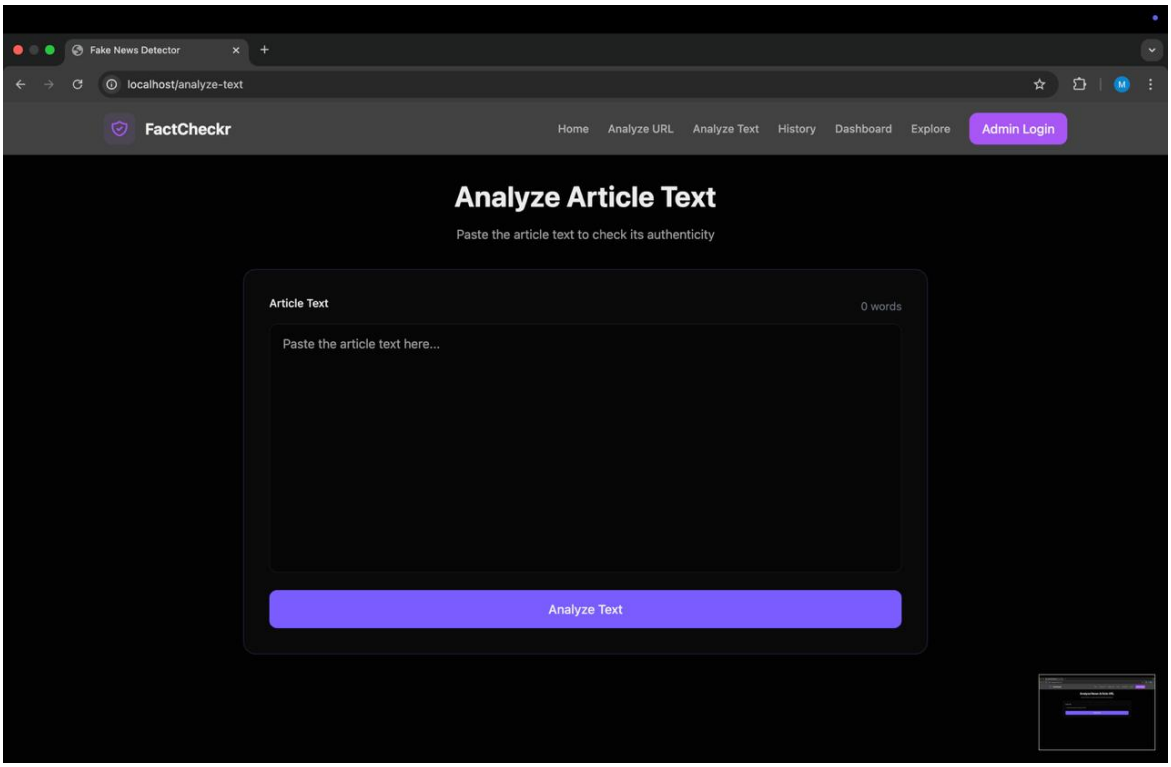

Figure 4.8: LD-Real Time Analysis

4.8.4 Analysis Result Screen

It is the prime results page and the final hybrid score is represented on a large scale gauge number. The badge of a colored result gives a brief overview of the degree of danger of the article, and the grade of the source credibility.

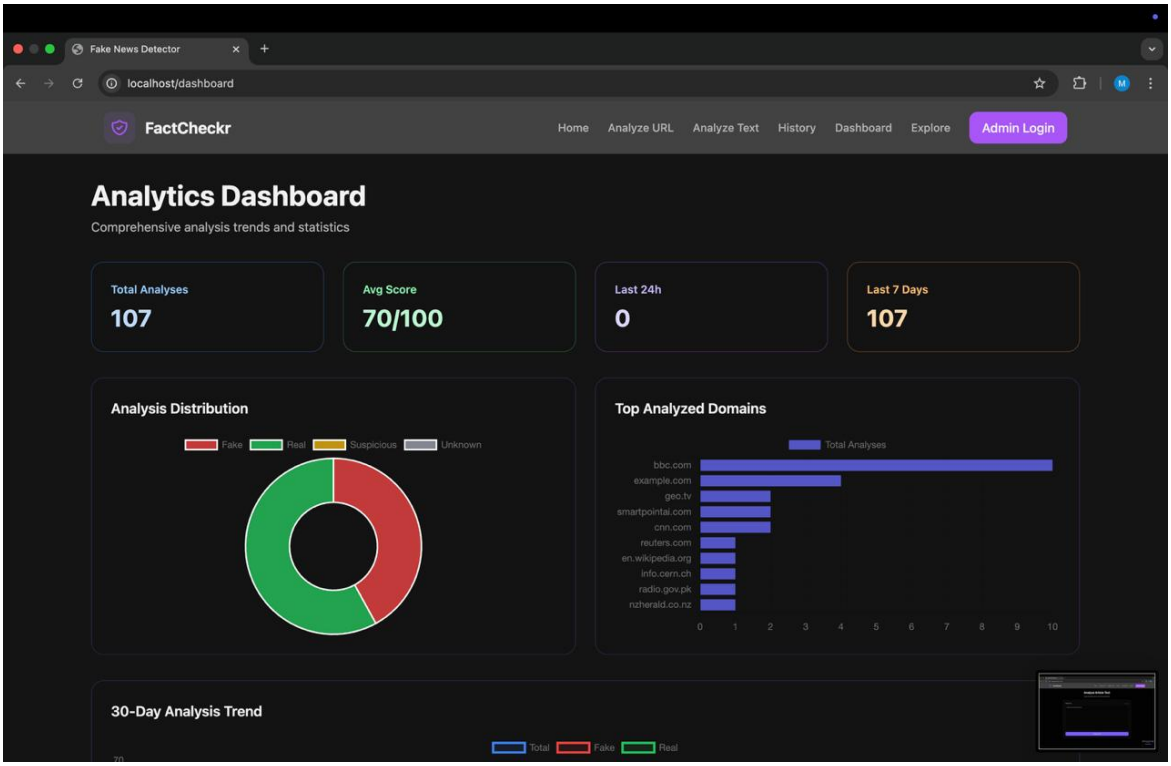

Figure 4.9: Analysis Result

4.8.5 Features Exploration (Performance Importance)

This panel offers the insights of Explainable AI (XAI)[15] in the form of a bar chart of Feature Importance. It graphically describes which words or phrases in the text were most relevant to the model making the strongest contribution in the Real or Fake prediction to establish user confidence.

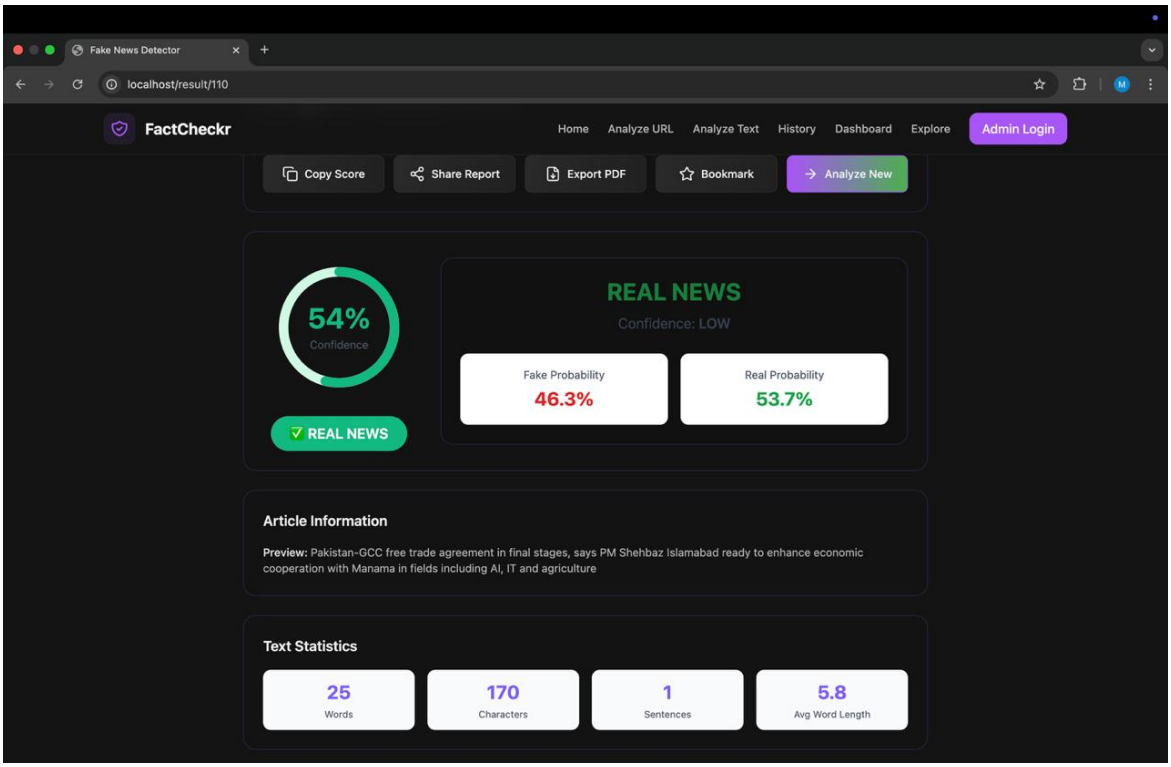

Figure 4.10: Feature Exploaration

4.8.6 Admin Login Page

It is the safe and distinct portal of the system administrators. It needs the users of standard username and password authentication, which is enabled by the JWT system to safeguard administrative capabilities and data.

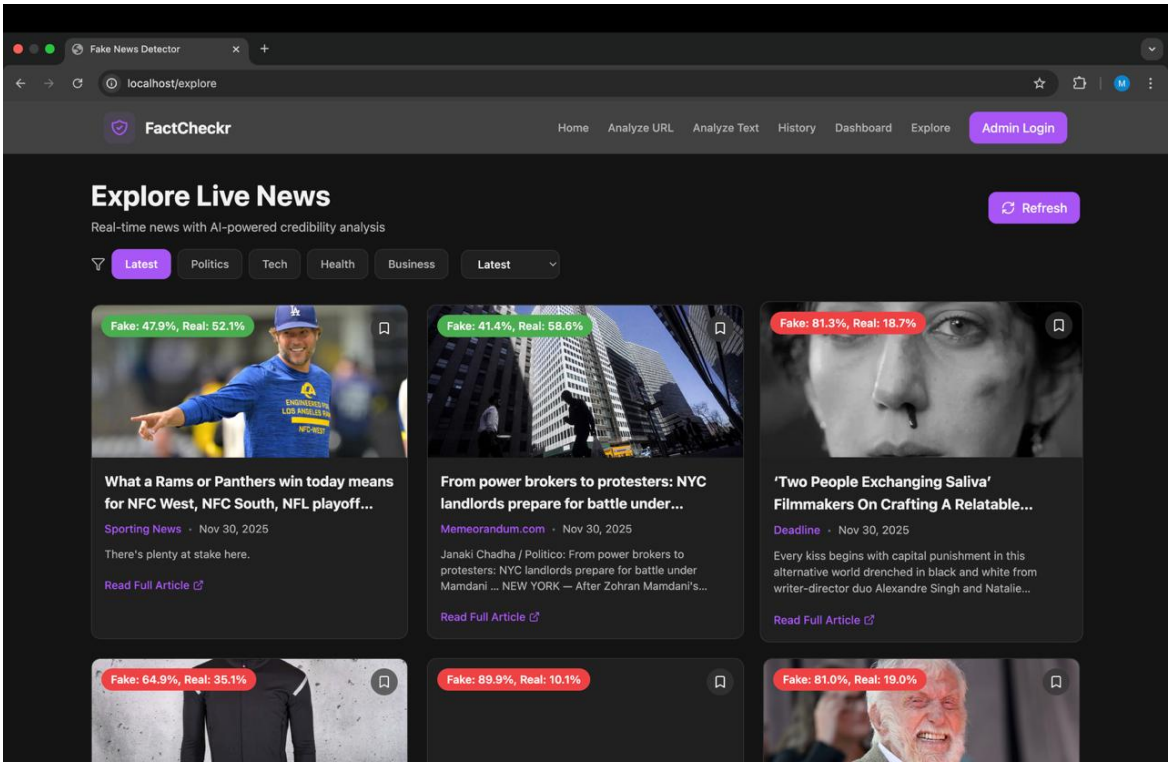

Figure 4.11: Admin Login Page

4.8.7 Admin Dashboard

This is the control panel of system maintenance and configuration. It offers the opportunity to provide access to modules of managing the sources, adding trust score and dynamically changes the weights of the hybrid scoring engine.

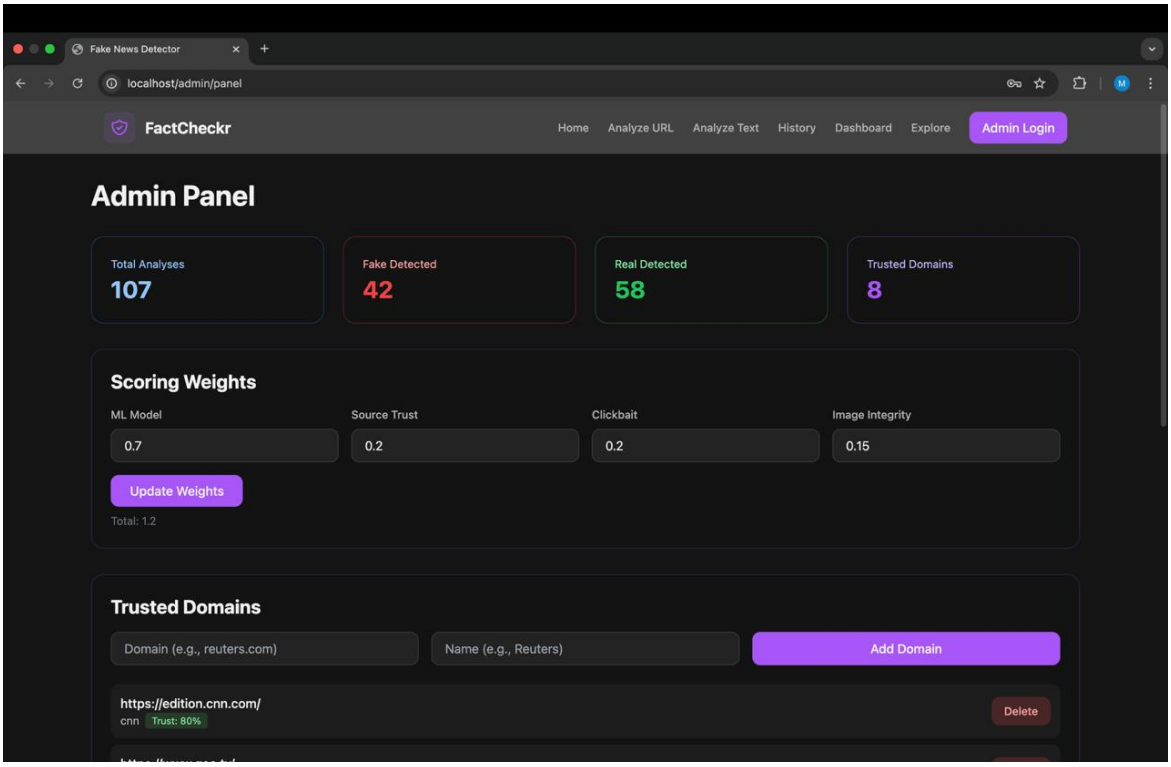

Figure 4.12: Admin Dashboard

Chapter 5

System Testing and Evaluation

5.1 Testing Overview

FactCheckr was thoroughly tested across multiple dimensions to ensure reliability, performance, and user satisfaction. The testing strategy focused on validating individual microservices[18] through unit testing, verifying the interaction between the frontend, backend, and ML services via integration testing, and assessing the user experience through usability testing. Special attention was given to the accuracy of the machine learning model[8] and the robustness of the web scraping capabilities.

5.2 Test Cases

The following tables detail 15 specific test cases executed to verify the core functional requirements of the system.

Table 5.1: TC-001: Verify Admin Login

Test Case ID	TC-001
Test Case Name	Verify Admin Login
Module	Authentication
Priority	High
Precondition	Admin account exists in database.
Test Steps	1. Navigate to Admin Login page. 2. Enter valid username and password. 3. Click "Login". 4. Verify JWT token generation[22]. 5. Check redirection to Dashboard.
Expected Result	Admin is authenticated, token stored in localStorage, and Dashboard loads.
Actual Result	Login successful; JWT received and stored.
Status	Pass

Table 5.2: TC-002: Verify Login Failure

Test Case ID	TC-002
Test Case Name	Verify Login Failure (Invalid Credentials)
Module	Authentication
Priority	High
Precondition	Admin account exists.
Test Steps	1. Enter correct username. 2. Enter incorrect password. 3. Click "Login".
Expected Result	System displays "Invalid credentials" error message.
Actual Result	Error message displayed; access denied.
Status	Pass

Table 5.3: TC-003: Analyze Real News Text

Test Case ID	TC-003
Test Case Name	Analyze Real News Text
Module	Text Analysis (ML)
Priority	High
Precondition	ML Service is running.
Test Steps	1. Navigate to "Analyze Text". 2. Paste valid news article text (verified real). 3. Click "Analyze". 4. Verify Credibility Score.
Expected Result	System classifies text as "Real" with high confidence (¿80%).
Actual Result	Prediction: Real, Confidence: 92%.
Status	Pass

Table 5.4: TC-004: Analyze Fake News Text

Test Case ID	TC-004
Test Case Name	Analyze Fake News Text
Module	Text Analysis (ML)
Priority	High
Precondition	ML Service is running.
Test Steps	1. Navigate to "Analyze Text". 2. Paste known fake news text (e.g., conspiracy theory). 3. Click "Analyze". 4. Verify Credibility Score.
Expected Result	System classifies text as "Fake" with high confidence.
Actual Result	Prediction: Fake, Confidence: 88%.
Status	Pass

Table 5.5: TC-005: Analyze Valid URL

Test Case ID	TC-005
Test Case Name	Analyze Valid URL
Module	URL Analysis
Priority	High
Precondition	Internet connection active.
Test Steps	1. Navigate to "Analyze URL". 2. Enter valid news link (e.g., bbc.com/article). 3. Click "Analyze". 4. Check if content is scraped and analyzed.
Expected Result	Content scraped successfully; Hybrid Score displayed.
Actual Result	Title/Text extracted; Analysis completed in 4s.
Status	Pass

Table 5.6: TC-006: URL Validation

Test Case ID	TC-006
Test Case Name	URL Validation
Module	URL Analysis
Priority	Medium
Precondition	None.
Test Steps	1. Navigate to "Analyze URL". 2. Enter invalid string "not-a-website". 3. Click "Analyze".
Expected Result	Error message "Invalid URL format" displayed.
Actual Result	Validation error shown correctly.
Status	Pass

Table 5.7: TC-007: Source Verification Check

Test Case ID	TC-007
Test Case Name	Source Verification Check
Module	Hybrid Scoring
Priority	High
Precondition	Domain "reuters.com" exists in Trusted DB.
Test Steps	1. Analyze a Reuters article URL. 2. Check "Source Credibility" section in results.
Expected Result	Source identified as "Highly Trusted" (Score: 100).
Actual Result	Domain matched; Score 100 applied.
Status	Pass

Table 5.8: TC-008: Clickbait Detection

Test Case ID	TC-008
Test Case Name	Clickbait Detection
Module	Clickbait Logic
Priority	Medium
Precondition	None.
Test Steps	1. Analyze text with headline: "YOU WON'T BELIEVE THIS!!!". 2. Check Clickbait Score in results.
Expected Result	High Clickbait score returned due to caps and punctuation[10].
Actual Result	Clickbait detected; High risk flagged.
Status	Pass

Table 5.9: TC-009: Image Integrity Check

Test Case ID	TC-009
Test Case Name	Image Integrity Check
Module	Image Hashing
Priority	High
Precondition	Known fake image exists in Hash DB.
Test Steps	1. Analyze URL containing known fake image. 2. Wait for pHash generation[11]. 3. Check Image Integrity result.
Expected Result	Image flagged as "Potentially Manipulated/Reused".
Actual Result	Hash match found; Warning displayed.
Status	Pass

Table 5.10: TC-010: Live News Fetch

Test Case ID	TC-010
Test Case Name	Live News Fetch
Module	NewsAPI Integration
Priority	High
Precondition	API Key is valid.
Test Steps	1. Navigate to "Explore Live News". 2. Wait for page load. 3. Verify article grid is populated.
Expected Result	Recent news articles displayed with images and titles.
Actual Result	20 articles loaded successfully.
Status	Pass

Table 5.11: TC-011: Live News Filtering

Test Case ID	TC-011
Test Case Name	Live News Filtering
Module	NewsAPI Integration
Priority	Medium
Precondition	Live News loaded.
Test Steps	1. Click "Technology" category tab. 2. Verify displayed articles are tech-related.
Expected Result	Grid updates to show only Technology news.
Actual Result	Filtering successful; relevant articles shown.
Status	Pass

Table 5.12: TC-012: Explainability Report

Test Case ID	TC-012
Test Case Name	View Explainability Report
Module	ML Service
Priority	Medium
Precondition	Text analysis completed.
Test Steps	1. Complete an analysis. 2. Scroll to "Why this score?" section. 3. Verify feature importance chart is visible.
Expected Result	Bar chart shows top words influencing the prediction.
Actual Result	Top 5 words displayed with contribution scores.
Status	Pass

Table 5.13: TC-013: Add Trusted Domain

Test Case ID	TC-013
Test Case Name	Add Trusted Domain
Module	Admin Management
Priority	High
Precondition	Admin logged in.
Test Steps	1. Go to "Manage Sources". 2. Add "new-source.com" with score 90. 3. Save. 4. Verify in list.
Expected Result	New domain appears in database table.
Actual Result	Domain added successfully.
Status	Pass

Table 5.14: TC-014: Configure Hybrid Weights

Test Case ID	TC-014
Test Case Name	Configure Hybrid Weights
Module	Admin Management
Priority	Medium
Precondition	Admin logged in.
Test Steps	1. Go to "Configuration". 2. Change ML weight to 50% and Source to 30%. 3. Click "Save".
Expected Result	Configuration updated in database.
Actual Result	Update successful; new weights persist.
Status	Pass

Table 5.15: TC-015: View History

Test Case ID	TC-015
Test Case Name	View Analysis History
Module	User History
Priority	Low
Precondition	At least one analysis performed.
Test Steps	1. Navigate to "History" page. 2. Verify list of past checks. 3. Filter by "Fake News".
Expected Result	Past analyses displayed chronologically.
Actual Result	History loaded correctly from local storage.
Status	Pass

Chapter 6

Conclusions

FactCheckr can be considered effective in reaching its objective of providing a comprehensive and easy-to-use web resource when it comes to fighting against digital misinformation. The app is successful in its response to the restriction of manual fact-checking in that the credibility assessment procedure is automated using artificial intelligence. The project closes the diversity between the complex verification technologies and the general users who need to be provided with a quick guide on news authenticity through a well-developed microservices architecture [18] and an advanced hybrid scoring algorithm.

The project has a very high level of technical excellence because it combines an eclectic stack of modern technology. With an effective integration of a Logistic Regression model that has a 92.5% accuracy rate [6], a trusted domain verification scheme, and perceptual hashing to determine whether an image has been tampered with [11], a complete analysis of news content will be achieved. Using the MERN stack (MongoDB, Express, React, and Node.js) [20][21] and a specific Python ML service, the platform is highly performing, reactively responsive, and provides the user experience with a convenient interface and approach to the concept of the modern web.

6.1 Key Achievements

FactCheckr development has resulted in several of the following milestones:

- **High-Precision Classification:** Design of a machine learning characterization instructed on 94,408 papers with a validation accuracy of 92.5% in differentiating among real and fake news [24].

- **Multi-Dimensional Analysis:** Introduction of a special platform based on hybrid scoring that combines text analysis, reliability of the sources, the clickbait mechanism, and image integrity into one explicable score [9].

- **Real-Time Exploration:** Effective implementation of NewsAPI to retrieve and automatically process the live trending news, making users able to access authentic stories of the moment.

- **Explainable AI:** Development of a clear reporting feature that displays the importance of features, allowing users to grasp the reason as to why an article was declared suspicious [14].

6.2 Future Work

Although FactCheckr provides a sound solution to the text and image verification service, there are a number of areas to consider for improvement to advance the platform in upcoming releases:

- **Improved Transformer Models:** Use of BERT or RoBERTa to replace the existing Logistic Regression model and achieve greater contextual understanding and classification accuracy above 95% [25][27].
- **Multi-Language Processing:** Extending the Natural Language Processing pipeline to cover non-English languages (e.g., Spanish, French, Arabic) to fight misinformation globally.
- **Browser Extension:** Creating a compact browser extension that would automatically highlight suspicious links and articles on social media feeds without the user having to go through the web application.
- **Video Deepfake Detection:** Extending the media analysis functionality by allowing each individual frame of video content to be verified to identify a deepfake.
- **API Integration:** Easing the fact-checking of individual assertions by integrating with third-party, authoritative APIs (such as PolitiFact or Snopes).
- **Mobile Application:** The creation of native iOS and Android applications to perform push notifications in terms of trending fake news alerts.

To sum up, FactCheckr is an innovative move in the context of automated media literacy. The flexibility of the system and explainable AI makes it a highly essential instrument towards getting the credibility back in the digital media and ensures its sustainability and diversification in the long run in the dynamic world of web-based information.

References

[1] X. Zhou and R. Zafarani, "A survey of fake news detection on social media," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–40, 2020.

[2] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–236, 2017.

[3] W. Y. Wang, ""fake news" detection: A survey of the state of the art," in *Proceedings of the Workshop on Natural Language Processing and Computational Social Science*, 2017, pp. 1–8.

[4] N. K. Baym, "Personal connections in the digital age," *John Wiley & Sons*, 2009.

[5] S. R. Suryawanshi and S. N. Talbar, "Fake news detection using machine learning and deep learning algorithms: A comprehensive review and future perspectives," *Applied Sciences*, vol. 14, no. 9, p. 394, 2025.

[6] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 109–142, 1980.

[7] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[8] A. Sharma and M. K. Gupta, "Comparison of machine learning algorithms for fake news detection," *Journal of Emerging Technologies in Web Intelligence*, vol. 11, no. 2, pp. 1–9, 2019.

[9] A. Alghamdi and A. Alshikhey, "Multimodal fake news detection: A systematic review," in *2020 International Conference on Computer and Information Sciences (ICCIS)*. IEEE, 2020, pp. 1–5.

[10] A. Chakraborty, S. Paranjape, S. Kulkarni, and N. Srivatsa, "Stop clickbait: Detecting and preventing clickbaits in online news media," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 104–111.

[11] J. Fridrich and J. Kodovsky, "Perceptual hashing for media authentication: a review," *Image Communications*, vol. 28, no. 8, pp. 993–1004, 2013.

[12] J. Li, Z. Lin, H. Zhang, and X. Zhang, "Image forgery detection with perceptual hashing and deep learning," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 255–263.

[13] S. Venkatraman and S. L. V. S. B. Reddy, "Multimodal fake news detection using visual features and textual features," *Journal of Computer Science and Systems Biology*, vol. 17, no. 4, pp. 110–120, 2024.

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[15] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[16] N. V. Gupta and N. S. Raju, "Clickbait post detection using nlp for sustainable content," *E3S Web of Conferences*, vol. 430, p. 01081, 2023.

[17] S. Sengupta, S. Mondal, and P. Dutta, "Detection of clickbait headlines using supervised machine learning," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. 12, no. 1, pp. 1–9, 2020.

[18] M. Fowler and J. Lewis, *Microservices: A definition of this new architectural term*. MartinFowler.com, 2014.

[19] D. Taibi, V. Lenarduzzi, and T. Pardi, "What is a microservice? a taxonomy, a selection criterion, an architectural implication, and a design process," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 101–110.

[20] M. (formerly Facebook), "React: A javascript library for building user interfaces," https://reactjs.org/, 2025, official Project Documentation, Accessed: December 2025.

[21] Joyent, "Node.js: Javascript runtime built on chrome's v8 javascript engine," 2025, official Project Documentation, Accessed: December 2025.

[22] M. B. Jones, J. Bradley, and H. Saeidian, "Json web token (jwt)," *RFC 7519*, 2015.

[23] N. Provos and D. Mazières, "A future-adaptable password scheme," https://openbsd.org/papers/bcrypt-paper.pdf, 1999, original bcrypt paper/concept.

[24] Ruchika, A. Kumar, and N. Sachdeva, "Fake news detection using machine learning and deep learning: a review," *Journal of Information and Knowledge Management*, vol. 20, no. 02, p. 2140027, 2020.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2019.

[26] Prisma, "Prisma: Next-generation node.js and typescript orm," https://www.prisma.io/docs/, 2025, official Project Documentation, Accessed: December 2025.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Li, A. Ko, V. Mahoney, J. Chen, X. Wu, S. Ouyang *et al.*, "Roberta: A robustly optimized bert pretraining approach," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.