NoCap: Fact Checking with AI

Team
Anthony Ciero aciero2022@my.fit.edu
Thomas Chamberlain tchamberlain2023@my.fit.edu
Joshua Pechan jpechan2023@my.fit.edu
Varun Doddapaneni vdoddapaneni2023@my.fit.edu

Advisor
Marius Silaghi msilaghi@fit.edu

Requirements Document

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to describe the “NoCap” project, a fact checking AI tool. This document also lists out the requirements needed such as functional requirements, interface requirements, and performance requirements. This document is for users of the NoCap application so they can better understand the specifications and features of the website.

1.2 Scope

The goal of NoCap is to create a website where users can input a piece of text or a link to any article and our AI will read the article and check for true and false information. The AI will also give every article an authenticity score for users to understand how trustworthy an article is, with help from authenticity graphical representation. Our website will then see if this article is already in our database and if not, will store it along with its authenticity score based on the publisher. This will allow other users to view articles trustworthiness before reading.

1.3 Definitions, acronyms and abbreviations

AI: Artificial intelligence
API: Application Programming Interface
URL: Uniform Resource Locator
AWS: Amazon Web Services
LLM: Large Language Model
CI/CD: Continuous Integration/Continuous Deployment
WCAG: Web Content Accessibility Guidelines
IEEE: Institute of Electrical and Electronics Engineers

1.4 References

IEEE Standard for Software Requirement Specifications

1.5 Overview

This document further explains the product perspective and functionalities, user characteristics, constraints, assumptions/dependencies, and specific requirements, namely functional, interface, and performance specifications

2. Overall Description

2.1 Product perspective

NoCap is a website that allows users to paste an article URL, where our AI will read the article and give the article an authenticity score and report. The articles can then be saved into the database and aggregate scores for the publishers will be public to view. This allows new users to see other articles and publishers that may be spreading misinformation.

2.2 Product Function

2.2.1 URL Detection

The website will allow the user to paste a URL into a box. Users will submit this URL via a button.

2.2.2 Raw Text Detection

The user will be able to input raw text instead of an article URL. Users will submit via the same button.

2.2.3 Text Detection with AI

The AI will read the given article and determine which sections of the article are fact or fiction.

2.2.4 Authenticity Score and Report

The AI will then give the article an authenticity rating out of 100, 1 being the lowest. It will also give a small report on which sections it believes to be true or false.

2.2.5 Add Article to Database

After checking and rating the article, if the article has not been checked before, add the article to a list based on the publisher.

2.2.6 Authenticity Graphical Representation

Based on the articles and authenticity reports in our database, users can view a bar graph showing which words or topics are believed to be true or false and how often they show up across the articles.

2.2.7 Publisher Aggregate Scores

Publishers themselves will be given an authenticity score by averaging all the authenticity scores of the articles in the database.

2.2.8 Viewing Publisher & Article Scores

Users will be able to view all the articles published sorted by publishing company and will be able to see which articles are considered more false or more true based on the highest true score starting at the top.

2.3 User characteristics

The main users are people who are looking for information about a topic and want to verify that information as true. This includes “news

consumers” who want to identify misinformation/biased reporting and “casual readers” who want to quickly fact check a source.9

2.4 Constraints

2.4.1 Website Internet Connection

- Since the system depends on cloud-based AI models and real-time API requests and database searches, it cannot function offline. Users must have an active internet connection to submit text/URLs and receive authenticity scores

2.4.2 API Dependency (rate limits and pricing)

- The platform depends on external APIs for AI-driven text analysis. These services have rate limits that may restrict the number of requests per second/minute, as well as cost implications per request. Both need to be factored into system design, caching strategies, and usage policies to avoid service interruptions or excessive operating expenses.

2.4.3 Response Speed

- Users expect near-instantaneous results. System performance is constrained by the time required for external API calls, backend processing, and network latency. To maintain usability, the system must optimize response speed through efficient request handling, load balancing, and caching where possible.

2.4.4 Scalability

- The system must be able to handle increased user traffic as adoption grows. Scalability will be constrained by cloud infrastructure costs. Proper use of AWS auto-scaling and serverless functions will be necessary to meet peak demand without significant downtime.

2.4.4 Processing Limits

- Each API call and backend service has limits on the size and complexity of data it can process. Full-article analysis requires reading, understanding, and scoring with potentially thousands of words per request. This creates constraints around processing speed, as the system must balance depth of analysis with quick turnaround times. There will need to be many optimizations to ensure comprehensive reports in reasonable time which could include summarization, parallel processing, and optimized prompt design.

2.5 Assumptions and dependencies

- The user has a device to open the website
- The user has an available wifi connection

- The user has a URL to the article they want to check
- The user has surface level experience with how to work their device and navigate to a website.

3. Specific Requirement

3.1 Functional Requirements

3.1.1 Amazon Bedrock Integration

- Used to handle AI model interface. The system will send text or URLs to the AI model hosted on Bedrock, which will process and return insights like authenticity scores and reasons for them. This ensures scalability and avoids building in-house model hosting

3.1.2 Text/URL entry

- Users can input either raw text or a webpage url. The system will parse the input, run it through the analysis pipeline and return an authenticity score.

3.1.3 Authenticity Score

- Each input will be given a calculated authenticity score which will represent the likelihood of reliability and misinformation. This helps users quickly gauge trustworthiness at a glance but also allows for more in depth analysis.

3.1.4 Publisher Aggregate Score

- Beyond individual articles, the database will store and maintain a record of publishers and compute an aggregate score based on all the submitted content. This provides users context on the historical credibility of a source.

3.1.5 Publisher Aggregate Ranking

- The website displays a list of major publishers/news sources (CNN, BBC, Reuters, etc) and ranks them based on the aggregate scores of each publisher. This will be sorted from high (most reliable) to low (least reliable).

3.1.6 Chrome Extension

- A browser extension will allow users to analyze articles directly on the web without switching to the website. They would only need to visit a site and the chrome extension would provide an authenticity score and a snippet of the reason behind it.

3.1.7 Database

- A database that stores scores for saved articles. This database will be referred to whenever a user inputs an URL. The URL is checked with the database, and if it already exists, that score is used.

3.1.8 WCAG Accessibility Guidelines met

- The website meets the WCAG AA standards of web accessibility, ensuring handicapped individuals can make use of the website. This includes tabability, accessible color contrasts, and easily readable text.

3.1.9 GitHub CI/CD pipeline

- Application has CI/CD pipeline from the GitHub repo, allowing any changes to be automatically deployed.

3.2 Interface Requirements

3.2.1 Home Page

- The entry point for users with quick access to text and URL input, navigation, and recent analyses. This will act as a dashboard for the application.

3.2.2 Rating Page

3.2.2.1 Overall Score

- A visual indicator on top of the numerical score.
  Red for 0-50 (Likely False)
  Yellow for 51-75 (Questionable)
  Green for 76-100 (Reliable)

3.2.2.2 Details of the Analysis

- For each prompt, the page shows justification for why it was scored the way it was. Summarizing claims in the data and aligning it with publicly available data and seeing if the claims align or are lacking cited sources. Additionally it should point out biased words such as “disastrous” or “catastrophic”.

3.2.3 Aggregate Page

- Provides an overview of publishers and their historical credibility. This page aggregates authenticity scores across multiple articles from the same publisher, showing a cumulative score. It can also include trends over time comparisons between publishers, and filtering/search features.

3.2.4 Extension Page

- A lightweight version of the analysis interface designed for the chrome extension. It will show the authenticity score and a summary directly in the browser without requiring the user to visit the main site. There will be a link to the main site if more detail is requested. This page will prioritize speed and clarity since it is being used in an in-context setting.

3.2.5 Multi-Browser Compatibility

- The website should be fully functional and visually consistent across all browsers and devices.

3.3 Performance Requirements

3.3.1 Speed

- Pages load under 5 seconds

- Analysis time: Ratings generated in under 30 seconds
- Article checking: Checking if an article has been rated before should happen under 10 seconds

3.3.2 Scalability(early version)

- Be able to handle ~100 requests an hour

3.3.3 Reliability

- If the API fails, the website should return an error message similar to (“Unable to analyze article, try again later”).

3.3.4 Usability

- The website should be compliant with the WCAG 2.1 AA guidelines