



Software Requirements Specification
Version 1.0

InsightBot

Theme: Daily News Simplified

Project Name: InsightBot

Category: Data Science Dominion

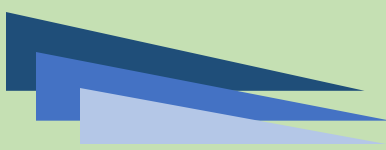


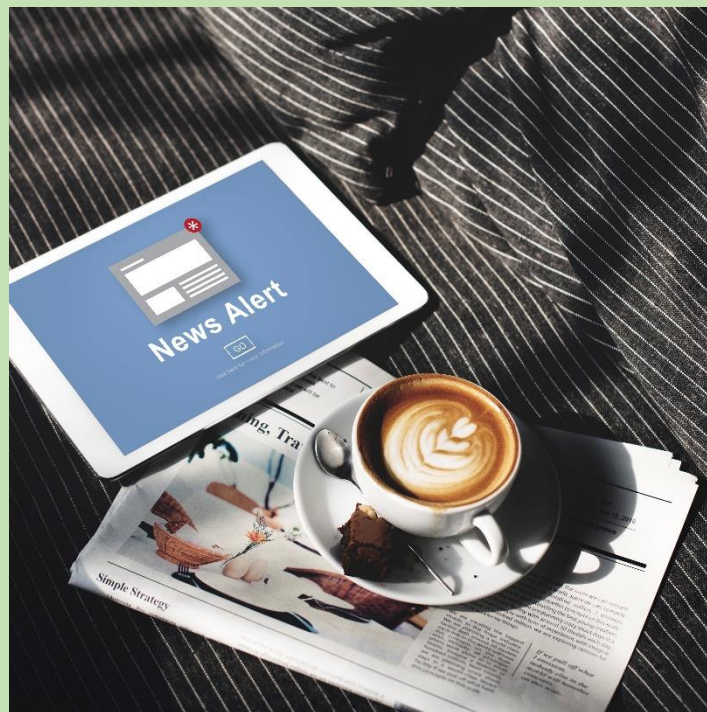
Table of Contents

1.1	Background and Necessity for the Application	2
1.2	Proposed Solution	3
1.3	Purpose of the Document.....	6
1.4	Scope of Project	6
1.5	Constraints	7
1.6	Functional Requirements	7
1.7	Non-Functional Requirements	10
1.8	Interface Requirements	11
1.8.1	Hardware	11
1.8.2	Software	11
1.9	Project Deliverables	12

1.1 Background and Necessity for the Application

In an age of information overload, people are bombarded with hundreds of news articles, blog posts, and updates daily across multiple platforms and in various formats. Navigating this sea of content to find clear, trustworthy, and relevant information can be time-consuming and overwhelming. To address this, **InsightBot** emerges as a smart, Data Science-driven solution that automatically extracts, filters, and simplifies news content into digestible summaries.

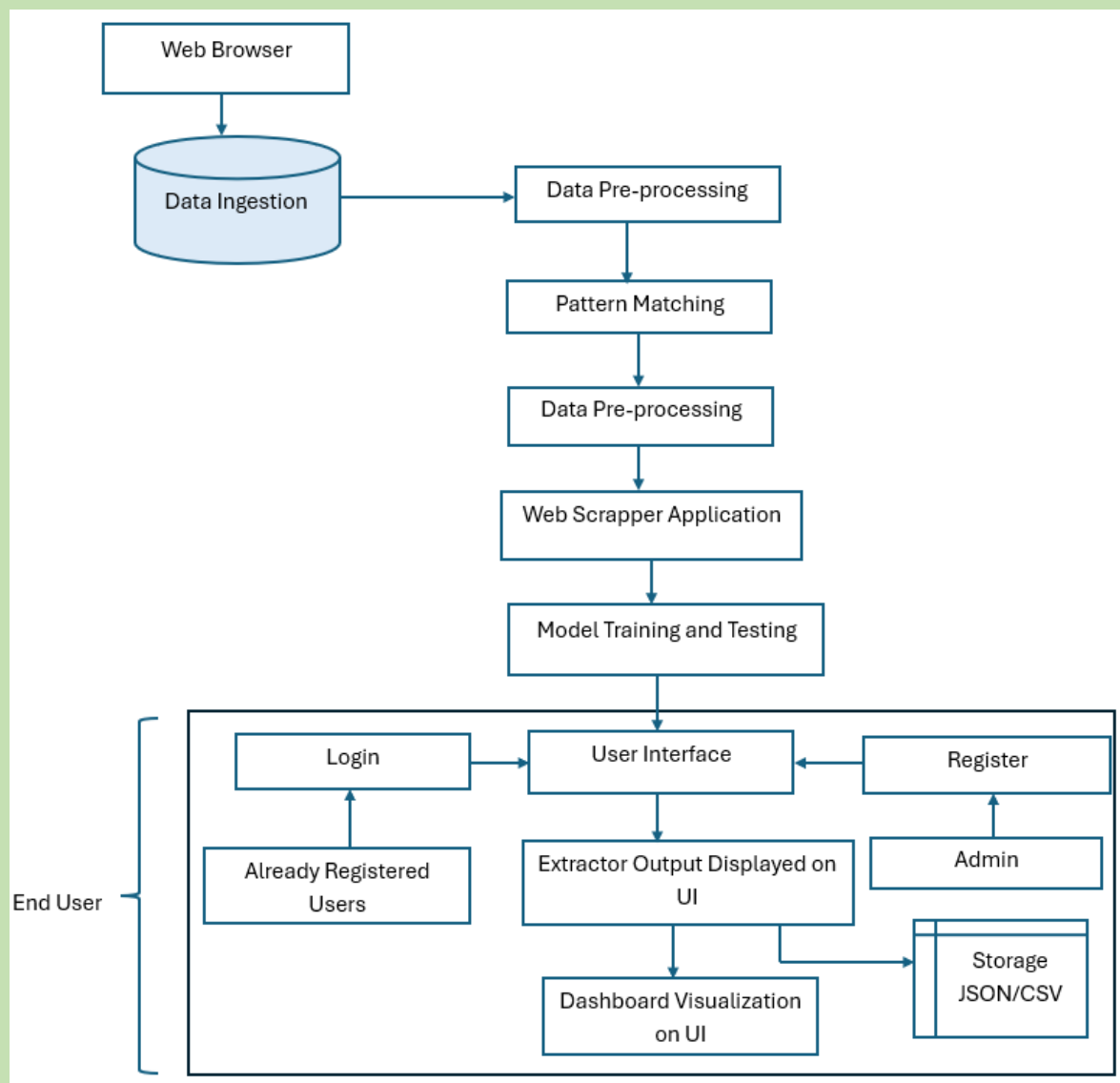
The necessity for such an application lies in the growing demand for personalized, real-time information delivery. By leveraging pattern mining techniques in Web scraping, **InsightBot** identifies structured elements such as titles and article bodies from diverse news Websites across multiple languages. This not only enables multilingual content processing, but also supports cross-regional awareness and media literacy. The integration of intelligent summarization and classification transforms raw Web data into user-friendly insights empowering readers, researchers, and professionals to stay informed with minimal effort.



1.2 Proposed Solution

To build a robust multilingual news simplification system, **InsightBot** applies a Data Science-driven approach focused on automated content extraction and pattern-based generalization. The methodology is designed to simplify daily news from multiple sources using lightweight and scalable techniques. The approach avoids complex browser automation and instead relies on pattern mining and structured HTML parsing to retrieve relevant information such as headlines and article content across various news Websites.

The sample architecture of the application can be as follows:



Sample Architecture of the Application



Dataset and Features

The dataset for this project is provided to you. The dataset consists of HTML page sources extracted from 40 multilingual news and blog Websites, covering English, Arabic, and Russian languages. These Websites were selected to represent diverse layouts and content styles commonly found in real-world news platforms. The dataset includes a mix of global and regional news outlets and blogs, each providing articles with varying structure, tone, and formatting.

Each record in the dataset contains unstructured HTML content parsed from Web pages, including key components such as article title, main body, publication date, language, and source Website URL.

Additionally, a separate set of 10 unseen Websites is used exclusively for testing purposes to evaluate the generalization ability of the rule-based scraping approach. These test sites help validate whether the system can effectively extract relevant content from websites it has not been trained on.

This dataset forms the basis for training a rule-driven content extraction system and visualizing key news insights through dashboards using Data Science techniques and Tableau.

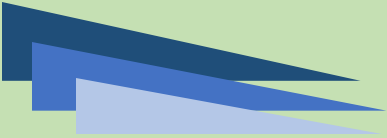
The project begins with data ingestion where a Web browser connects to news Websites and blog sources to collect HTML content. This raw data is stored in structured formats such as JSON or CSV, which serve as the storage foundation for the next stages.

The ingested content undergoes initial pre-processing, which includes cleaning, formatting, and isolating useful HTML tags. This step helps remove noise and prepare the data for further processing. Following this, a pattern matching module is employed to identify key visual and structural indicators on a Web page. These include the largest text blocks for headlines or paragraph blocks for article bodies, based on observations from 40 Websites across three languages (English, Arabic, and Russian).

After pattern matching, the data goes through another round of pre-processing to refine and normalize the extracted features. This ensures consistency before the Web scraper application runs. The scraper, guided by the earlier pattern rules, extracts article titles, publication dates, article bodies, and formats them into clean outputs.

The next phase is model training and testing, where the system uses these extracted samples from 40 Websites for training and evaluates generalization using data from 10 additional Websites.

Once the model is trained, it integrates with the User Interface (UI). Users can either log in (if already registered) or register via admin approval. The extractor's



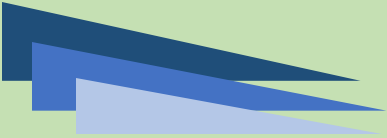
output is displayed on the UI in real-time, showing cleaned news content in a user-friendly layout.

Finally, dashboard visualization on the UI using tools such as Tableau allows users to view aggregated news trends, topic frequency, or language distribution. This completes the pipeline from data collection to meaningful insight presentation in an interactive graphical form.

Steps to build the model for **InsightBot** are as follows:

1. **Dataset Collection and Upload:** Backend ingestion or manual URL input.
2. **Preprocessing + Pattern Mining:** Identify large div text blocks and largest <h1>/<h2> titles.
3. **Training Model:** Use 40 Websites to train Document Object Model (DOM) pattern matching or rule-based logic.
4. **Testing Model:** Apply model to 10 Websites to verify extraction accuracy.
5. **Display on UI:** Extracted articles shown in clean template.
6. **Dashboard Visualization:** Use Tableau or embed charts (article count by domain, language, and date) via APIs.





1.3 Purpose of the Document

The purpose of this document is to present a trained and tested interactive model titled **InsightBot**.

This document explains the purpose and features of Data Science and the constraints under which it must operate. This document is intended for both stakeholders and developers of the application.

1.4 Scope of Project

The scope of this project is to implement a simplified news extraction system, **InsightBot**, that scrapes content from multiple news and blog Websites across different languages. The project includes designing and implementing data ingestion, preprocessing, and pattern-based content extraction. It also involves training and testing the extraction logic using multilingual Websites and displaying the extracted news titles and bodies on a UI.

However, advanced features such as sentiment analysis, fake news detection, multilingual translation, or deploying the solution as a live Web service fall outside the scope of this implementation.



1.5 Constraints

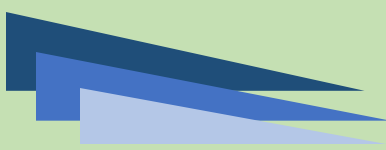
The constraints of this project include handling the structural variability of different news and blog Websites across multiple languages, that may affect the accuracy of pattern-based extraction.

Additionally, the project relies on publicly accessible Websites, meaning changes in HTML structure or access restrictions (such as CAPTCHA or anti-scraping mechanisms) can disrupt data collection. Limited language-specific NLP capabilities and the absence of labeled training data also restrict the precision of automated content extraction.





1.6 Functional Requirements

The **InsightBot** project focuses on building an intelligent content extraction and visualization system that scrapes news and blog data from multilingual Websites (English, Arabic, and Russian). It identifies and extracts key textual content such as headlines and article bodies and presents this information through a user-friendly interface with interactive dashboards. The solution emphasizes automated data



handling and pattern-based content extraction, allowing users to access structured and simplified news content.

Key functional requirements of this system are as follows:

- **i. Data Ingestion from Websites** - The system shall collect Web pages from 40 training Websites and 10 testing Websites in English, Arabic, and Russian using Python-based Web scraping tools/libraries such as BeautifulSoup or Scrapy. It must handle multilingual encoding and store raw HTML responses.
- ii. Content Parsing and Preprocessing** – The application shall parse raw HTML to extract clean and usable text. This involves removing unnecessary HTML tags, scripts, ads, and styles, and normalizing the text structure for pattern analysis.
- iii. Pattern-Based Content Extraction** - The system shall identify article titles and bodies using predefined structural patterns. These patterns include detecting the largest text blocks for headlines and the longest paragraph elements for article bodies, based on analysis from training Websites.
- **iv. Content Storage** – The extracted and structured data (titles and article bodies) shall be stored in JSON and CSV formats for further use in visualization and analysis.
- v. UI Development** – The system shall provide a simple user interface to allow users to browse the extracted news content in a clean and categorized format. The UI must support language filters and article browsing.
- vi. Dashboard Visualization:** The system shall integrate with Tableau Desktop to create dashboards displaying aggregated insights such as trending topics, frequency of keywords, language distribution, and article volume over time.

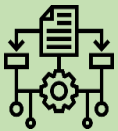


vii. **Automatic Scheduling-** The project shall include a mechanism to automate scraping at regular intervals (example, daily), ensuring content freshness and minimal manual intervention.

viii. **Extraction logic Evaluation-** The application shall validate the pattern-matching logic using a test set of 10 Websites and report performance based on accuracy in identifying correct titles and bodies.

The **InsightBot** application should also be able to implement following functionalities:

- **Multilingual Content Support** - The application shall support extraction from Websites in three languages. It must handle Unicode characters and apply language-agnostic extraction rules.
- **Language Toggle and Search Functionality-** If implemented, the UI should support switching between supported languages and allow keyword-based search within the extracted content.



1.7 Non-Functional Requirements

There are several non-functional requirements that should be fulfilled by the application which are as follows.



1. **Performance:** The system should process and extract data from a Website within a reasonable time (example, under five seconds per page), ensuring a smooth and responsive experience for the user.

2. **Scalable:** The application should be scalable to handle the addition of more Websites and multiple languages in the future without significant changes to the architecture.

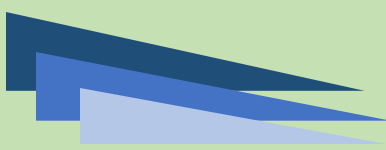


3. **Usable:** The user interface should be simple, intuitive, and accessible to users with minimal technical knowledge, allowing them to view extracted news articles and dashboard insights effortlessly.

4. **Reliable:** The system should maintain consistent performance over time, successfully scraping and displaying data even if minor structural changes occur on the source Websites.

5. **Maintainable:** The codebase should be modular and well-documented so that future updates to scraping logic, UI components, or language support can be implemented with ease.





These are the bare minimum expectations from the project. It is a must to implement the **FUNCTIONAL** and **NON-FUNCTIONAL** requirements given in this SRS.

Once they are complete, you can use your own creativity and imagination to add more features if required.

1.8 Interface Requirements

1.8.1 Hardware

Intel Core i5/i7 Processor or higher
8 GB RAM or higher
Color SVGA monitor
500 GB Hard Disk space
Mouse and Keyboard

1.8.2 Software

Technologies to be used:

1. **Data Store:** JSON/TXT/CSV
2. **Backend:** Apache Spark or Apache Hive, Flask, Django
3. **Database:** MongoDB, MySQL
4. **Programming/IDE:** Python 3.11.4 or higher, Jupyter Notebook, Anaconda 23.1.0 or higher, Google Collab
5. **Libraries:** BeautifulSoup, Scrapy, requests, regex, NLTK/BERT, Pandas, Matplotlib, seaborn, NumPy
6. **Visualization:** Tableau Desktop

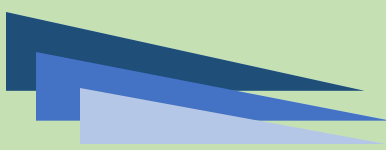


1.9 Project Deliverables

You will design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design Specifications
- Diagrams such as User Flow Diagram/User Journey Map
- Test Data Used in the Project
- Project Installation Instructions
- Link of GitHub for accessing the uploaded project code (Link should have public access)
- Detailed Steps to Execute the Project
- Link of Published Blog

The source code, including .ipynb files for Jupyter Notebook and Google Colab, should be shared via GitHub. Appropriate access permissions should be granted to users to allow testing for Jupyter Notebook and Google Colab. The consolidated project must be submitted on GitHub with a ReadMe.doc file listing assumptions (if any) made at your end.



Provide the GitHub URL where the project has been uploaded for sharing. The repository on GitHub should have public access. Documentation is a very important part of the project; hence, all crucial aspects of the project must be documented properly. Ensure that documentation is complete and comprehensive.

You should publish a blog of minimum 2000 words on any free blogging Website such as Blogger, Tumblr, Ghost, or any other blogging Website. The link of the published blog should be submitted along with the project documentation.

Do NOT copy content or code from GPTs or other AI tools, although you are permitted to use images generated by AI tools for any visual representation purposes. It is mandatory to mention such tools used in case you add any AI generated images.

Submit a video (.mp4 file) demonstrating the working of the application, including all the functionalities of the project. This is MANDATORY.

~~~ End of Document ~~~