PROJECT REPORT
WEB SCRAPPER

Omar ALLAL
Hamza BOUKHATEM
Skander RADHOUANE
Mohamed Ali BEN GHARBIA

Third deliverable: High Level Design of the proposed solution

Contents

Use case .....

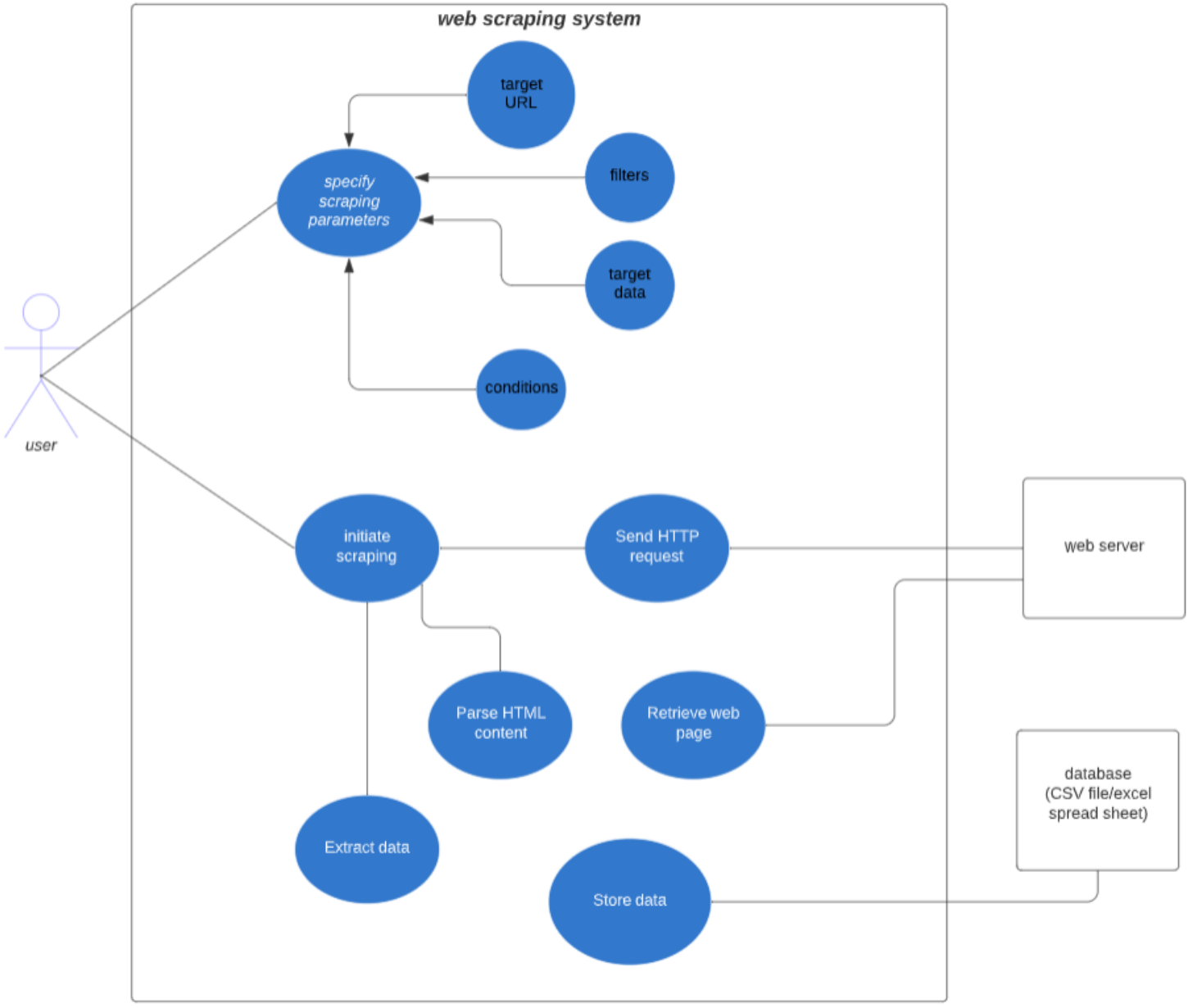
Functional requirements .....

Non-functional requirements .....

Assumptions.....

Exchanged messages/data.....

I. Use case


web scraping system
target URL
specify scraping parameters
filters
target data
conditions
user
initiate scraping
Send HTTP request
web server
Parse HTML content
Retrieve web page
database (CSV file/excel spread sheet)
Extract data
Store data

II. Functional requirements

- The system should be able to retrieve and extract data from the target website's web pages. This includes capturing text, images, links, and other relevant information based on specified criteria.
- The system should be capable of navigating through the target website's pages to access the desired data. This may involve following links or interacting with pagination.
- The system should be able to parse the retrieved HTML or other markup language to identify and extract specific data elements. This may involve using techniques like CSS selectors, XPath expressions, or regular expressions.
- The system should provide functionality to transform and clean the extracted data. This may include removing unnecessary whitespace, formatting dates or numbers, or converting data into a standardized format.
- The system should be able to handle various types of errors that may occur during the scraping process, such as connection errors, timeouts, or invalid HTML structures. It should provide appropriate error messages and graceful error handling mechanisms.
- The system should allow users to configure scraping parameters, such as specifying the target elements, defining filters or conditions, setting up scraping intervals, and selecting output formats.
- The system should provide logging and reporting features to track and record scraping activities. This includes logging successful and failed scraping attempts, capturing relevant metadata, and generating reports or summaries of scraping results.

III. Non-functional requirements (out of scope)

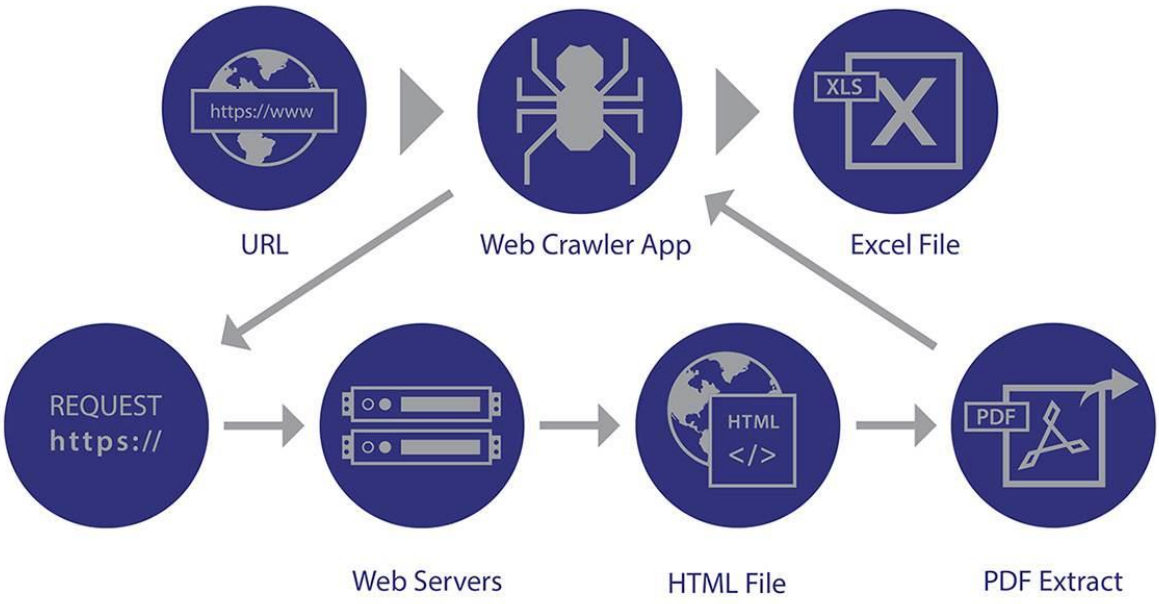
- The system should be able to efficiently process and extract data from web pages, providing fast response times and minimizing latency. It should be able to handle large volumes of data and perform optimally even with high traffic or complex web pages.
- The system should be designed to handle increased workloads and accommodate future growth. It should be able to scale horizontally or vertically to handle additional scraping tasks or larger datasets without a significant decrease in performance.
- The system should be reliable and robust, ensuring that data extraction processes are accurate and consistent. It should be able to handle errors gracefully and recover from failures or interruptions without losing data or compromising the overall system integrity.
- The system should have the ability to withstand and recover from unexpected failures, such as network outages, server errors, or changes in the target website's structure. It should employ techniques like retries, timeouts, and error handling mechanisms to ensure resilience in adverse conditions.
- The system should incorporate security measures to protect sensitive data and ensure secure communication between the system and the target website. It should support authentication and authorization mechanisms to restrict access to authorized users only.
- The system should be designed and implemented in a way that facilitates easy maintenance and future enhancements. It should have modular and well-documented code, allowing for easy troubleshooting, bug fixing, and extension of functionalities.
- The system should adhere to legal and ethical guidelines regarding web scraping. It should respect the terms of service of the target

website and comply with relevant data protection and privacy regulations.
- The system should provide a user-friendly interface and intuitive controls that enable users to configure scraping parameters, monitor scraping tasks, and access extracted data easily. It should be designed with user experience in mind to promote efficiency and productivity.

IV. Assumptions

- Availability and accessibility of web pages: The web scraping system assumes that the target website and the specific web pages being scraped are accessible and available during the scraping process. Changes in website structure or temporary unavailability can affect the scraping system's functionality.
- Consistent HTML structure: The system assumes that the HTML structure of the target website remains relatively consistent over time. If the website undergoes significant changes in its structure or markup, the scraping system may need to be adjusted or updated accordingly.
- Valid and well-formed HTML: The system assumes that the web pages being scraped contain valid and well-formed HTML code. In cases where the HTML is poorly structured or contains errors, the scraping system may encounter difficulties in accurately parsing and extracting the desired data.
- Network connectivity: The system assumes a stable and reliable internet connection for accessing the target website and retrieving web page content. Network disruptions or connectivity issues can impact the system's ability to perform scraping tasks.

V. Exchanged messages/data



- URL requests and responses: When a web scraper requests a web page, it sends a URL request to the server, which sends a response back containing the requested web page.
- HTML/XML documents: The web scraping engine processes the HTML or XML documents returned by the server to extract the desired data.
- Extracted data: The data extracted from the web pages is stored in the data storage component of the system.
- User input: Users provide input to the system through the user interface, specifying the websites to scrape, the data to extract, and other parameters of the scraping process.
- Error messages: If there are issues during the scraping process, the system may generate error messages that are displayed in the user interface or logged in a file.