Second Submission Report
Web Scraper Security Project

1. High-Level Concepts: Theory & Main Components

Main Components of a Web Scraper

Component	Function
HTTP Request Sender	Sends requests to target URLs to get webpage data
Response Handler	Receives and checks response status and handles errors
HTML Parser	Extracts specific data using BeautifulSoup, lxml, or regex
Scheduler (optional)	Controls timing of requests to avoid bans
Storage Manager	Saves data in formats like CSV, JSON, or databases
Logger	Tracks scraping progress and errors

Functional Flow Summary

- User inputs target URL
- Validate and sanitize input
- Send GET request using `requests`
- Receive HTML response
- Parse HTML with BeautifulSoup
- Store data securely (e.g., CSV, JSON)
- Log each step for transparency and debugging

2. High-Level Design of the Proposed Solution

Architecture Overview

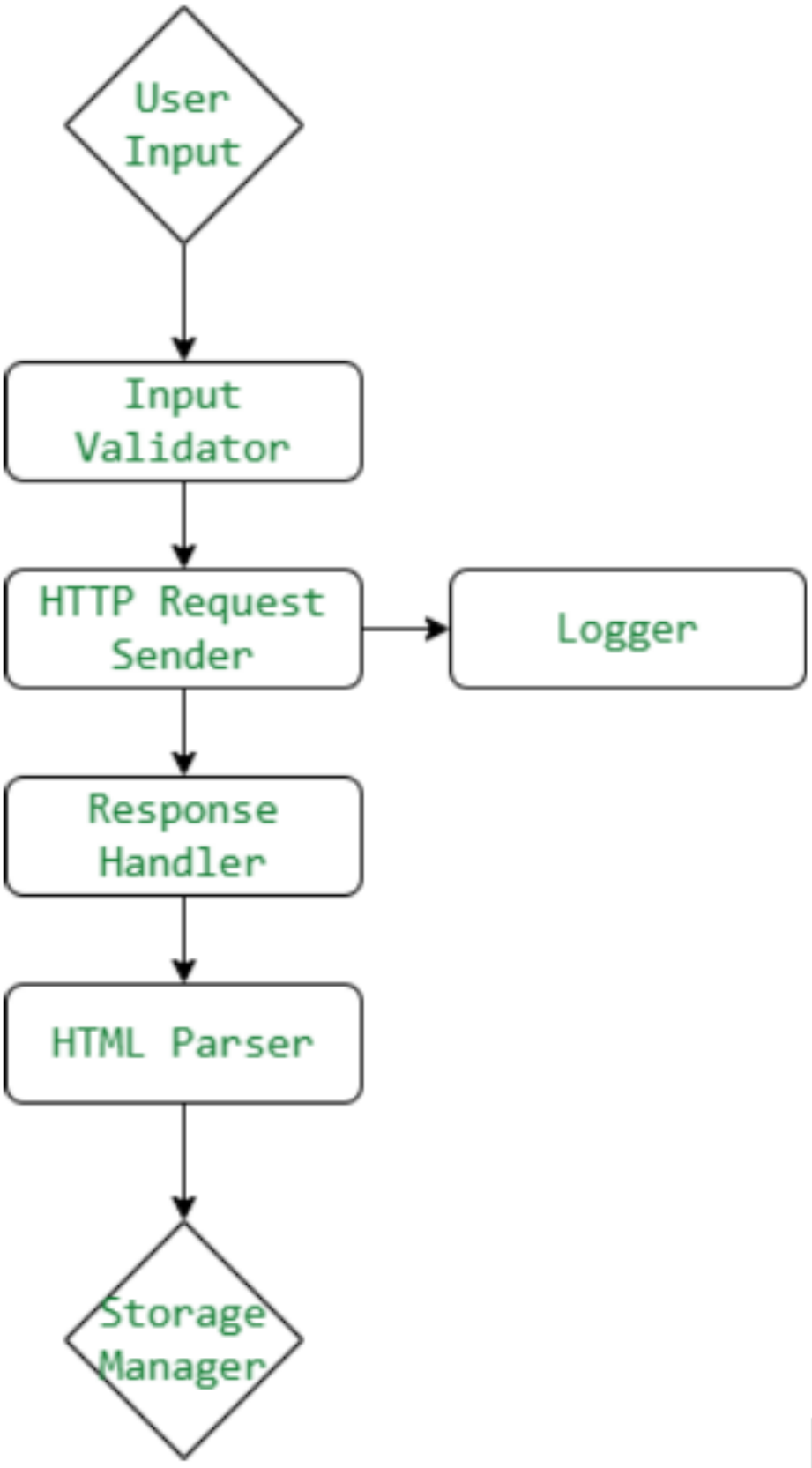

Figure 1: Architecture Overview of the Web Scraper System

Roles/Users

- **User:** Inputs the target URL and parameters
- **System (Backend):** Executes the scraper, processes data, logs activity

Exchanged Data

- **Input:** URLs, headers, keywords
- **Output:** Parsed data (e.g., product names, prices), logs, errors

3. Tools and Development Phases

Tools Used

- **IDE:** Visual Studio Code
- **Languages:** Python (scraper logic), JavaScript/HTML/CSS (frontend)
- **Libraries:**
  - `requests`, `BeautifulSoup` — core scraping
  - `logging` — process tracking
  - `pandas`, `json`, `csv` — storage
  - `flask` — API
- **Database:** Embedded in the website being scraped
- **Security:** Input sanitization, rate-limiting (via `time.sleep()`), and error handling

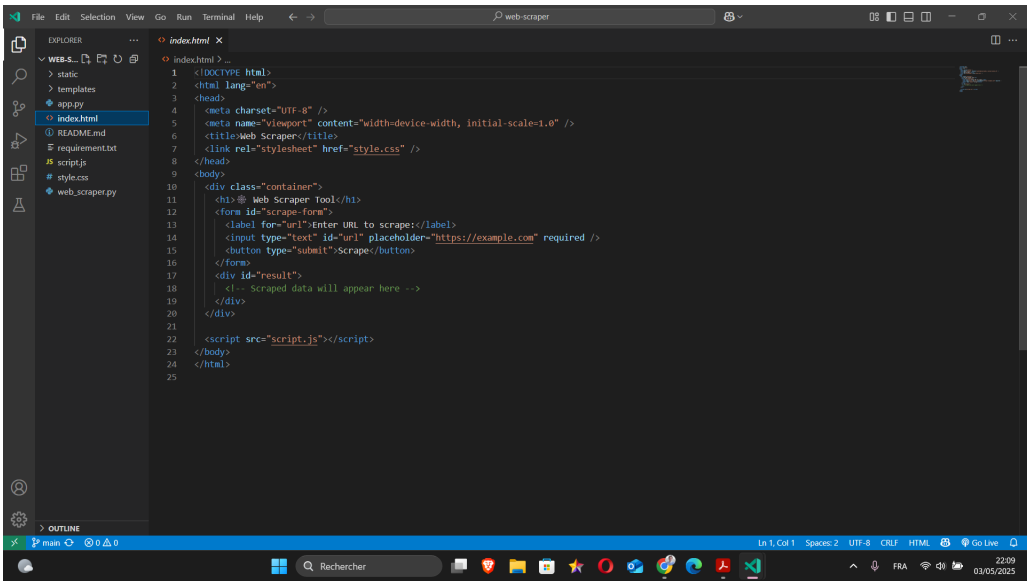

Figure 2: Security Enhancements in Web Scraper

Development Phases

Phase	Description
Phase 1 – Planning	Define scope, select target sites, clarify legality
Phase 2 – Core Scraper	Build scraping components using Python libraries

Phase 3 – Frontend UI	Simple HTML/CSS form to input URL and trigger scrape
Phase 4 – Storage & Logging	Format and store results; log all events
Phase 5 – Security Add-ons	Add error handling, rate-limiting, and URL checks
Phase 6 – Testing & Debug	Use mock sites and monitor scraper behavior

4. Advantages, Challenges, and Limitations

Advantages

- Automates repetitive data collection
- Useful for competitive analysis, research, news aggregation
- Flexible: supports static and dynamic content

Challenges & Limitations

- May violate websites' Terms of Service
- Bot detection mechanisms (CAPTCHAs, IP blocking)
- Scraping dynamic content (JavaScript-heavy pages) is complex
- Legal and ethical risks if scraping sensitive/personal data