# Technical Challenge

NAME : M A U S Y MARASINGHA

GITHUB : https://github.com/uthpalasajana/Data-Scraping-Project

Contents

[Technical Challenge 1](#_Toc211913492)

[1. Introduction 3](#_Toc211913493)

[2. System Functions 3](#_Toc211913494)

[Read and Extract Movie Names 3](#_Toc211913495)

[Dynamic URL Generation and Headless Navigation 3](#_Toc211913496)

[Web Data Extraction (Scraping) 3](#_Toc211913497)

[Data Store in JSON Format 4](#_Toc211913498)

[Fetch All Available TV Series 4](#_Toc211913499)

[Scheduled Execution 4](#_Toc211913500)

[3. Language and Technologies 5](#_Toc211913501)

[Primary Language 5](#_Toc211913502)

[Framework and Tools 5](#_Toc211913503)

[4. Dependences 6](#_Toc211913504)

[5. Folder Structure 6](#_Toc211913505)

[6. Assumptions and Concerns 7](#_Toc211913506)

[1. Website Stability 7](#_Toc211913507)

[2. Internet Connectivity 7](#_Toc211913508)

[3. Rate Limiting and Server Load 7](#_Toc211913509)

[4. Headless Mode Execution 7](#_Toc211913510)

[7. Error Handling 8](#_Toc211913511)

[8. Test Cases 9](#_Toc211913512)

[9. Edge Cases 10](#_Toc211913513)

[10. Future Improvements 10](#_Toc211913514)

[11.Conclusion 11](#_Toc211913515)

[12. References 11](#_Toc211913516)

# 1. Introduction

The Movie Data Scraper is a Spring Boot–based service that automatically extracts movie or TV show data from a public website called todaytvseries.one using Selenium WebDriver. It runs on a scheduler, scrapes data at fixed intervals (every 30 minutes), and saves structured results in JSON files

# 2. System Functions

The Movie Data Scraper system automates the process of collecting movie and TV series information from the todaytvseries.one website.

## Read and Extract Movie Names

* + The system reads a list of movie names from the movieInput.json file located in the resources directory.
  + These names are used as input parameters for the scraping process.

## Dynamic URL Generation and Headless Navigation

* + For each movie name, the system dynamically constructs the target URL by combining it with the base website address.
  + Selenium WebDriver operates in headless mode, enabling the scraping process to run in the background without launching a visible browser window.

## Web Data Extraction (Scraping)

* The scraper navigates to each movie’s dedicated webpage and extracts relevant details
  + - Movie Title
    - Poster Image URL
    - Genre or Type
    - Schedule or Release Time
    - Description
* Data is extracted using XPath selectors to accurately target specific elements on the webpage.

## Data Store in JSON Format

* + The extracted data for each movie is converted into structured JSON format.
  + Each movie’s details are stored in a separate JSON file under the output/ directory

## Fetch All Available TV Series

* + The system also visits the main TV series listing page on the website.
  + It extracts all available TV series names and their respective URLs.
  + These details are saved in a separate JSON file named allTvShows.json.

## Scheduled Execution

* + The entire process is automated using Spring Boot’s scheduling feature (@Scheduled).
  + The scraper runs at a fixed interval of every 30 minutes, ensuring that the collected data remains up-to-date.

# 3. Language and Technologies

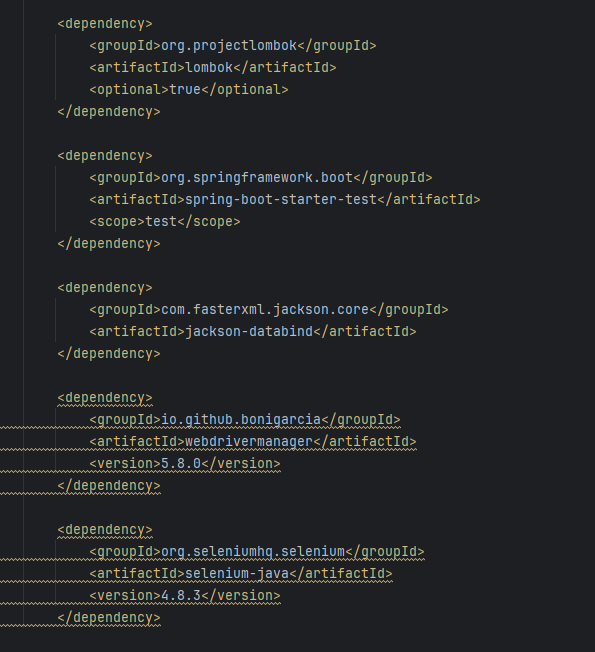
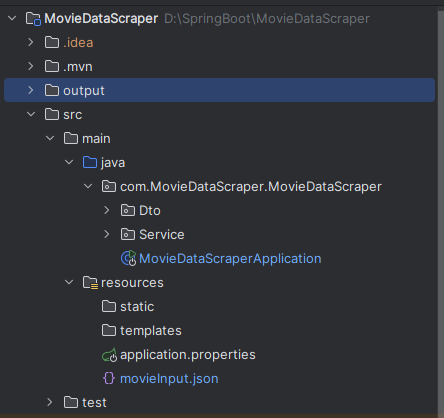
## Primary Language

The project is implemented using **Java**, The **Spring Boot** framework.Java was chosen for its strong ecosystem, scalability, and mature integration with Selenium for browser automation.  
Spring Boot simplifies the setup and management of scheduled services and JSON file handling.

## Framework and Tools

1. **Backend Framework – Spring Boot**
   * Provides a lightweight and production-ready framework.
   * Simplifies dependency management and configuration.
   * Used to build and schedule automated scraping tasks.
2. **Web Automation – Selenium WebDriver**
   * Enables browser-based automation for web scraping.
   * Interact with the webpage like a real user.
3. **Headless Browser – Google Chrome (Headless Mode)**
   * Execute browser automation without displaying a graphical interface.
   * Improves scraping speed and reduces system resource usage.
4. **Data Storage Format – JSON (Gson Library)**
   * Stores scraped data in a structured and lightweight format.
   * Gson library is used for writing and reading JSON data.
5. **Logging – SLF4J with Logback**
   * Provides efficient logging for tracking system activities and errors.
   * Helps in debugging, performance monitoring, and ensuring reliability.
6. **Scheduling – Spring Scheduler (@Scheduled)**
   * Automates the scraper to run periodically at fixed time intervals.
   * Ensures continuous and up-to-date data collection without manual intervention

# 4. Dependences

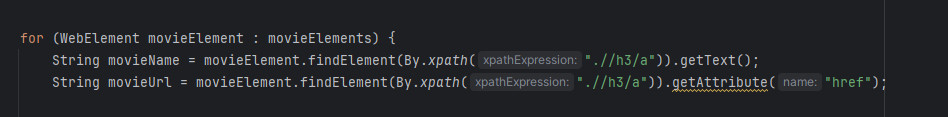


# 5. Folder Structure

# 6. Assumptions and Concerns

## Website Stability

* + It is assumed that the website’s HTML structure and XPaths remain consistent over time.



## Internet Connectivity

* + A stable internet connection is required for Selenium WebDriver to load and scrape pages.
  + Any interruption in connectivity may result in incomplete or failed data extraction.

## Rate Limiting and Server Load

* + The target website might limit or block frequent requests.
  + To prevent overloading the server, the scraper includes a controlled delay between requests.

A screen shot of a computer code

AI-generated content may be incorrect.

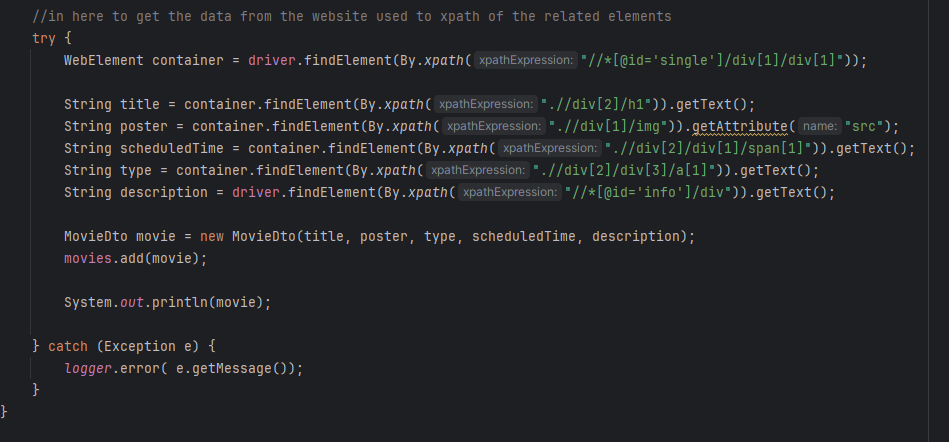
## Headless Mode Execution

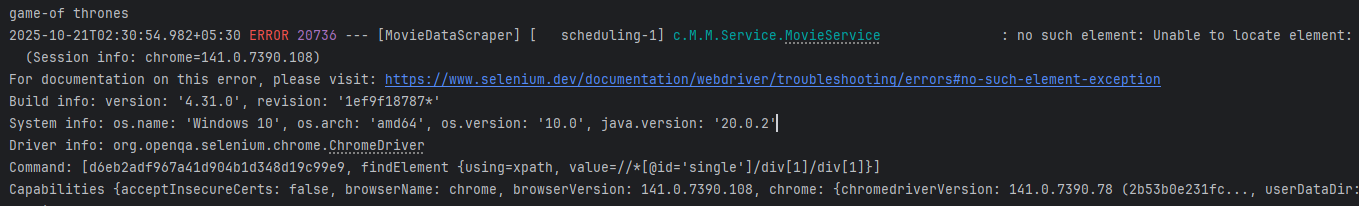
* + The scraper operates in Google Chrome’s headless mode to run in the background without opening a browser window.

A black background with text

AI-generated content may be incorrect.

# 7. Error Handling

* All scraping logic is wrapped in try-catch blocks to prevent the entire service from crashing due to a single failure.
  + Errors and exceptions are logged for debugging and monitoring to maintain reliability and robustness.



# 8. Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC ID** | **Scenario** | **Expected Result** | **Actual Result** | **Screen Shot** | **Status** |
| TC-01 | Application startup | Spring Boot application runs successfully, ChromeDriver initializes in headless mode | Application starts Successfully Chromedriver runs in headless mode |  | Pass |
| TC-02 | Read movie names from movieInput.json | JSON file is parsed correctly and movie names loaded into list | Movie names printed in console and processed |  | Pass |
| TC-03 | Scrape valid movie page | Movie details (title, poster, type, time, description) extracted correctly | Data printed and saved to JSON file |  | Pass |
| TC-04 | Scrape all TV shows | All available TV shows retrieved from web page | TV show list saved to allTvShows.json |  | Pass |
| TC-05 | Handle invalid movie name | System skips invalid movie without crash | Error logged, continues next movie |  | Pass |
| TC-06 | JSON file write | Data successfully written to output directory | Output files generated |  | Pass |
| TC-07 | Internet disconnection during run | Application runs successfully but no results scraping | Error logged as  No element found |  | failed |
| TC-08 | Website structure changes | Fails gracefully and logs missing element | Logged exception, program continues |  | Pass |

# 9. Edge Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Edge Case ID** | **Condition** | **Expected Behavior** | **Handling Mechanism** |
| EC-01 | movieInput.json file is empty | Program should not crash; simply log warning | Caught in try–catch, logs message “The JSON file contains no names  ” |
| EC-02 | Movie or TV show without description | JSON generate description field as empty | Adds "description": "" to JSON |
| EC-03 | Duplicate movie entries | Prevent scraping a duplicate entry | Caught in try–catch, logs message “Duplicate name found  ” then skip that entry |
| EC-04 | Output directory missing | log error | Handles via FileWriter exception logging |
| EC-05 | Extremly Large number of movies | Scrap only the loaded movies ai the first page | Uses clearData() after each movie to free memory |

# 10. Future Improvements

* Add dynamic scheduling using a JSON file
* Add Multi threading so that it can scrap several movies in the same time
* Save the data to a database instead of JSON files

# 11.Conclusion

The Movie Data Scraper automates movie and TV show data extraction using Spring Boot, Selenium, and headless Chrome. It runs on a fixed schedule, stores results in JSON, and includes error handling and logging. The system is efficient, reliable, and easily extendable for future enhancements like database storage or API integration.

# 12. References

* *Spring Initializr* (no date) *Spring Initializr*. Available at: https://start.spring.io (Accessed: October 21, 2025).
* Jaoude, E. (2021) *The README is the most important file in your GitHub projects*. Youtube. Available at: https://www.youtube.com/watch?v=5JoEB2YTlpw (Accessed: October 20, 2025).
* *(1817) YouTube* (no date). Youtube. Available at: https://www.youtube.com/watch?v=SfkXcYbxukI&list=LL&index=1&t=1573s (Accessed: October 21, 2025).
* *Lee, S. (2020) Web scraping dynamic websites with java and selenium. Youtube. Available at: https://www.youtube.com/watch?v=PF0iyeDmu9E&list=LL&index=3 (Accessed: October 20, 2025).*
* *Maven Repository: Search/Browse/Explore* (no date) *Mvnrepository.com*. Available at: https://mvnrepository.com/ (Accessed: October 21, 2025).