University of Windsor

TEAM LANCER

Car Rental Price Analysis

Group 5

~By: Gravin Patel, Smit Patel, Ayush Patel, Timil Kumar Prajapati, Sanket Kothiya
MAC, UWindsor.

Team Members

Group 05

Gravin Patel
110156150
Group Leader

Smit Patel
110156386

Ayush Patel
110137866

Timil Prajapati
110157182

Sanket Kothiya
110156177

Members

INTRODUCTION

The idea of the project is to analyze and understand car rental prices to get the best deal for the user:

- Choose at least three websites to crawl (e.g. Zoom.com, carrental.ca, expedia.com, etc.).
- Specify the details of your search (e.g. Name, Price, Rating, Image, Car Specification, Seat Capacity, Passenger Limit, etc).
- Implement required features (see Final Project Information.pdf, Instructions).

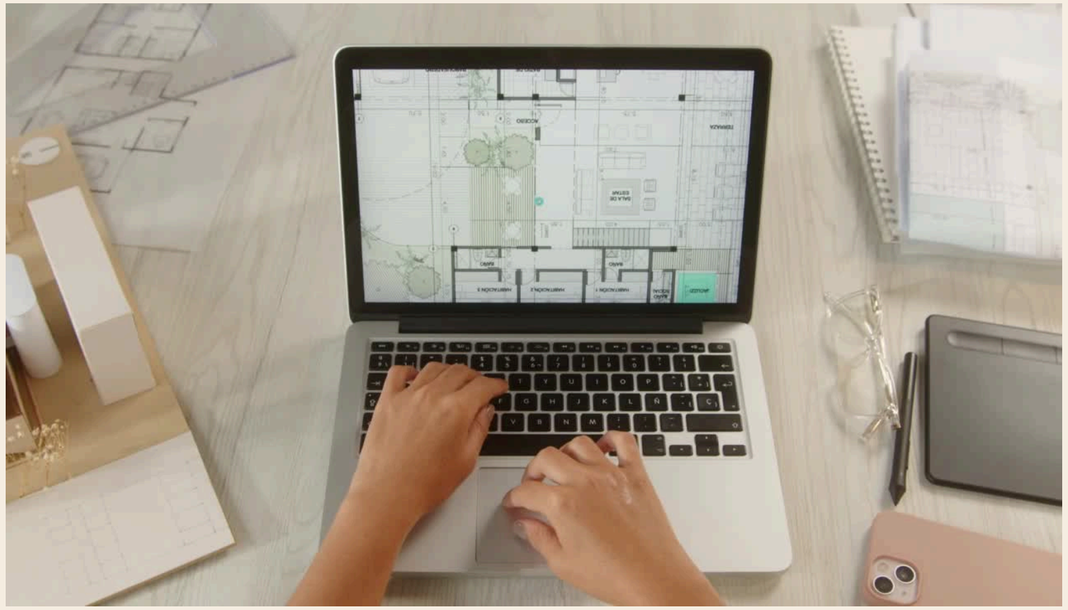
AGENDA

- PROJECT TASK
- TASK MANAGEMENT
- TASK DESCRIPTION
- REFERENCES
- CONCLUSION

PROJECT TASK

List of tasks included

- Data Validation
- Pattern Finder using Regex
- Web Crawler
- HTML Parser
- Spell Checker
- Word Completion
- Word Frequency Count in File
- Search Frequency
- Page Ranking
- URL Extractor
- Rating based Sorting with QuickSort



TASK MANAGEMENT

Gravin Patel

1

- Page Ranking [PageRanking.java]
- HTML parser and Web Crawler [Main.java]
- Final Documentation

Smit Patel

2

- Word Frequency in File [WordFrequencyInFile.java]
- Find pattern using RegEx [UserManager.java]
- Rating based Sorting
- Integration with Command Line Interface [IntegratedFinalfile.java]

Ayush Patel

3

- Search Frequency [SearchQuery.java]
- Web Crawler [Main.java]
- Presentation
- Final Documentation

Timil Prajapati

4

- Spell Checker [SpellChecker.java]
- Data Validation [UserManager.java]
- Integration with Command Line Interface

Sanket Kothiya

5

- Word Completion with Tries [WordCompletionTries.java]
- URL Extractor [URLExtractor.java]
- CSV Management

TASK DESCRIPTION

Task-1: Data Validation

Data validation for email and password ensures that user inputs meet specific criteria for security and usability. Email validation checks for a standard format, including valid characters and a domain. Password validation enforces strength by requiring a mix of upper and lower case letters, digits, and special characters, enhancing protection against unauthorized access.

Task-2: Pattern Finder using Regex

A Pattern Finder using Regex identifies and extracts specific text patterns efficiently. It searches for sequences like email addresses or phone numbers, enabling precise data retrieval and manipulation for streamlined processing and analysis.

Task-3: Web Crawler

A web crawler for a car rental website automatically extracts data like car listings, Name, Price, Rating, Image, Car Specification, Seat Capacity, Passenger Limit. It systematically navigates web pages, collecting relevant information to provide comprehensive and up-to-date datasets for analysis and comparison.

Task-4: HTML Parser

An HTML parser processes HTML documents to extract and manipulate data. It converts HTML content into a structured format, allowing easy access to elements like tags, attributes, and text for tasks such as web scraping and data extraction.

TASK DESCRIPTION [CONT]

Task-5: Spell Checker

In this task, we aim to create an efficient spell checker using hashing techniques and sorting algorithms. We'll build a vocabulary from CSV/text files and store it with cuckoo hashing for fast lookups. To suggest alternative words for misspellings, we'll implement an edit distance algorithm and sort the suggestions using merge sort, ensuring users receive the most relevant corrections first.

Task-6: Word Completion using Tries

In this task, we'll build a word completion feature using a Trie. By loading a vocabulary into the Trie, we'll enable fast retrieval of words that start with a given prefix, providing users with quick and relevant suggestions.

TASK DESCRIPTION [CONT]

Task-7: Word Frequency Counter in CSV

This task involves counting word frequencies in text files using the Boyer-Moore algorithm for efficient matching. Words are extracted from CSV/TXT files and counted with a hash table. The Boyer-Moore algorithm speeds up the process by reducing unnecessary comparisons, making it ideal for large text files

Task-8: Search Frequency

This task involves tracking and displaying word search frequencies using self-balancing trees. We'll handle user search queries and store them in an AVL or red-black tree, updating counts with each search. A log of queries will be maintained to display the top searches to users.

TASK DESCRIPTION [CONT]

Task-9: Page Ranking

This task focuses on developing a page ranking system using specific sorting algorithms. We'll parse web page content to extract keywords, store their frequencies in a self-balancing tree, and sort them by frequency using quick sort and Boyer-Moore algorithm. Page ranks will be calculated based on keyword frequencies, tracking and displaying the highest-ranked pages.

Task-10: URL Extraction

This task aims to find and extract URLs from a given text. We will create a regular expression to identify URLs and implement a method to extract all URLs from the text, providing a list of the found URLs.

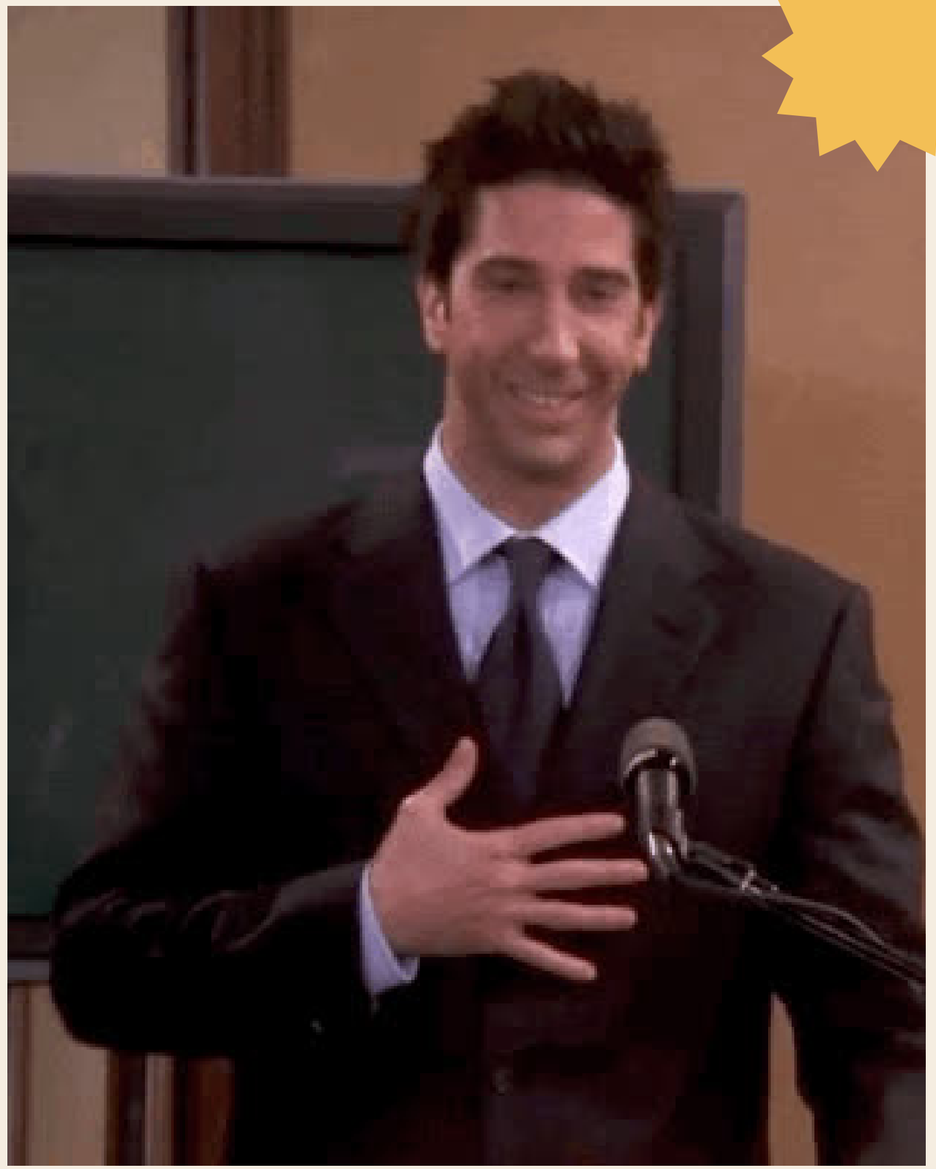

DEMO

CONCLUSION

Through this project, we have developed a comprehensive set of text processing tools that utilize a range of algorithms and data structures, including hashing, tries, self-balancing trees, and various sorting techniques. The tasks have provided us with practical experience in implementing and optimizing these algorithms for spell checking, word completion, frequency counting, search frequency tracking, and page ranking. Moreover, the validation and extraction features have enhanced our skills in using regular expressions for data extraction. This project has not only strengthened our algorithmic and programming capabilities but also prepared us for tackling complex problems in software development and data analysis.

REFERENCE LIST

1. Regular Expressions in Java.
2. A study on Page Ranking Algorithm.
3. Data Structure and Algorithms in Java.
4. Selenium WebDriver Quick Starter.
5. Optimize Boyer Moore Algorithm Using Parallel Processing.



Thank You