**Proposal Details**

|  |  |
| --- | --- |
| Group Number | *G007* |
| Registration Number of Group Members | 2022-CS-07  2022-CS-37 |

**Proposal Details**

|  |  |
| --- | --- |
| ***Project*** |  |
| Proposed Project Title | WebBayScraper |
| Executive Summary | The project involves designing and implementing a web scraping script or tool to extract data from eBay's laptop listings page, including product details, prices, seller information, and other relevant attributes.Extracted data will be stored in a structured format, such as CSV or a database, to facilitate further analysis and reporting.  The scraped data will provide real-time information on laptop listings, helping businesses and consumers make informed decisions.Users can monitor price fluctuations over time for specific laptop models.  As there is no requirement for user registration, so we can just make this a view-only application for daily users, and admins can maintain the application and manage all the CRUD operations. But all other options like searching, sorting, etc can be done by the user.  UI would be interactive and user-friendly to give the user a great experience. While Scrapping data, we can also pause, start, resume and stop, with the progress bar showing the progress of tasks/ number of entities scrapped through UI.  **Sorting:**  Users can also see which video got the highest views and which got the lowest views. Similarly, it can be shown for other attributes. Hence, different sorting techniques/algorithms can be applied for that purpose and the choice will be of the user which algorithm they want to apply. Sorting can be applied on the base of any attribute.  **Searching:**  Users can search for any video based on any attribute. It can be the title of the video, the date published, or the channel name. In case of multiple outcomes, the user will get a preview of all. In this way, we could implement searching algorithms very effectively such that the user can have the option to search for a video according to his information from that many records.  **CRUD Operations:**   * Create: This is the equivalent of "Create" in CRUD. It involves collecting data from eBay's laptop listings page and storing it in a structured format (e.g., CSV file or database). Each new extraction session creates new data records. * Retrieve: After scraping and storing the data, you'll perform "Read" operations to access and retrieve specific pieces of information. This could include querying the data for laptop prices, seller ratings, or other details. * Update: Regularly update the scraped data to ensure it remains current and accurate. You may want to schedule periodic scrapes to refresh the dataset with the latest information. * Delete: While not as common as the other operations, you may need to delete or remove outdated or irrelevant data from your dataset to maintain data quality and reduce storage space. |
| ***Business Case*** |  |
| Outline the business need for the project | The business need for this project arises from the growing importance of data-driven decision-making in today's competitive e-commerce landscape. eBay is a massive online marketplace, and access to real-time data on product listings, pricing trends, and seller performance can provide businesses and individuals with a significant competitive advantage. The project aims to scrape eBay data and provide users with comprehensive insights to aid their decision-making processes. |
| End user of the product | The targeted audience for this project includes businesses, e-commerce entrepreneurs, researchers, and individual consumers who wish to access and analyze eBay data. They will benefit from detailed information about eBay listings, enabling them to make informed decisions about buying and selling products on the platform. |
| Motivation for Project | Businesses need to stay informed about market trends and competitor pricing on eBay to make informed product sourcing and pricing decisions. Sellers on eBay can gain a competitive edge by analyzing their competitors' listings and strategies. |
| State the level of impact expected should the project proceed and the implications of not proceeding | Users will be empowered to make data-driven decisions in the highly dynamic eBay marketplace, leading to better buying and selling strategies. Businesses can gain a competitive advantage by leveraging insights gathered from eBay data, which can result in improved profitability. |
| ***Technical Details*** |  |
| Name of Entity | Video (YouTube) |
| Attributes of Entity  (Minimum seven attributes/rows can be increased) | |  |  |  | | --- | --- | --- | | *Name* | *Data type* | *Description* | | Title | String | Name of the video | | No. of views | Integer | Total views on the video. | | No. of comments | Integer | Total comments on the video. | | No. of Likes | Integer | Total likes on the video. | | Channel Name | String | Name of the channel which published the video. | | Channel Subscribers | Integer | Total subscribers the channel has. | | Video Length | Integer (Seconds) | The actual length of the video. | | Date Published | Date Type Object | The date on which the video is published. | | URL | String | URL of the video. | |
| Sample of Scrapping Source |  |
| Gitlab Repository Link | [*https://gitlab.com/Dahmer\_Harman7/2022\_cs\_07\_37\_dsamidproject*](https://gitlab.com/Dahmer_Harman7/2022_cs_07_37_dsamidproject) |
| Sorting Algorithms |  |
| |  |  | | --- | --- | | **Algorithm Name** | **Description** | | Merge Sort | The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two halves and then they are combined in a sorted manner and it is being done recursively. | | Hybrid-Merge Sort | A hybrid sorting algorithm is a blending of two different sorting algorithms, typically, a divide-and-conquer algorithm, like merge-sort, combined with an incremental algorithm, like insertion-sort. | | Insertion Sort | Insertion sort is a simple sorting algorithm that works similarly to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed in the correct position in the sorted part. | | Bubble Sort | Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.In every iterate we will get largest element sorted at end. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high. | | Selection Sort | Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list. | | Counting Sort | Counting sort is a sorting algorithm that sorts the elements of an array by counting the number of occurrences of each unique element in the array. | | Radix Sort | Radix sort is just like count sort but in this we will create an array on the base of number system for decimal 10 size array it will sort elements on the base of individual digits. | | Quick Sort | A sorting technique that sequences a list by continuously dividing the list into two parts and moving the lower items to one side and the higher items to the other. It starts by picking one item in the entire list to serve as a pivot point. The pivot could be the first item or a randomly chosen one. | | Bucket Sort | Bucket Sort is a sorting algorithm that divides the unsorted array elements into several groups called buckets. Each bucket is then sorted by using any of the suitable sorting algorithms or recursively applying the same bucket algorithm. Finally, the sorted buckets are combined to form a final sorted array. | |  |  | | |
| Searching Algorithms | * **Liner Search**: Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found, otherwise the search continues till the end of the data set. It is the easiest search algorithm |
| Searching Filters for each data type | Searching can be done using the following filters:   * For attributes of string type: * **Contains**: Search for the keyword provided by the user within the product title, seller's name, or other relevant text attributes. * **Ends with**: Search for the keyword provided by the user at the end of the product title, seller's name, or other relevant text attributes. * **Starts with**: Search for the keyword provided by the user at the beginning of the product title, seller's name, or other relevant text attributes. * **Equals**: Search for the keyword provided by the user, and it must match exactly with the product title, seller's name, or other relevant text attributes. * For attributes of integer type: * **Equals**: Search for the exact numeric value provided by the user in attributes like price, shipping cost, seller rating, or any other relevant numeric attribute. * **Greater than** Search for values greater than the numeric figure provided by the user in attributes like price, shipping cost, seller rating, or any other relevant numeric attribute. * **Less than**: Search for values less than the numeric figure provided by the user in attributes like price, shipping cost, seller rating, or any other relevant numeric attribute.. * For attributes of Date type: * **On**: Search for items listed on the date entered by the user, based on attributes like listing date, last updated date, or any other relevant date attribute. * **Before**: Search for items listed before the date entered by the user, based on attributes like listing date, last updated date, or any other relevant date attribute. * **After**: Search for items listed after the date entered by the user, based on attributes like listing date, last updated date, or any other relevant date attribute.   (he user can select the attribute on which the search will be performed. For example, they can choose to search by product title, price, listing date, or any other relevant attribute.) |
| Multi-Level Sorting | Sort listings based on the number of views. Listings with the highest number of views appear first.If two or more listings have the same number of views, the second level of sorting takes place. In this case, we sort these listings based on the number of likes. Listings with the most likes among the ones with the same view count will be ranked higher. If there are still listings with the same view count and the same number of likes, the third level of sorting can be applied. This could involve sorting by the product title or a unique attribute like the URL of the listing. This ensures that even if view count and likes are identical, listings are presented in a definitive order. |
| Any other features | Users will have access to graphical analysis tools that provide visual insights into both individual eBay listings and entire seller channels. This can include charts, graphs, and visual representations of data to help users understand trends and patterns more easily.Clicking on the title of a specific eBay listing will redirect users to a dedicated page with comprehensive details about that specific listing. Users can also initiate searches for other eBay listings from this page, and the comparison of selected listings can be displayed using various filters and graphical representations.Users can select a particular seller's channel, and the platform will enable them to view comparisons of the videos within that channel using graphical representations. This feature allows users to gain insights into the seller's performance and content popularity within the channel. |
| ***Interfaces for your project*** |  |
| |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Side panel/bar | Navigation bar/ Menu | Having different options like admin login and more details. | | Sorting Algorithm Dropdown | Dropdown Menu | A drop-down menu has different sorting algorithms as options from which users can select anyone on whose base sorting will be done. | | Multi-Level Sorting Button | Button | When clicked, a new menu/screen will be opened so that the user selects different levels on whose basis | | Pause button | Button | A button to pause/start the process of scrapping. | | Tabular view of video attributes | Grid View | Contains information on all attributes of the video | | Progress bar of scrapping | Progress bar | A bar showing the progress made while scrapping the data and no. of videos scrapped so far. |  |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Home button | Button | A button to go back to the home page. | | Submit button | Button | A button to add the video. | | Different input fields for attributes | Input | Admin will enter the details for the new video. |  |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Side panel/bar | Navigation bar/ Menu | Having different options like admin login and more details. | | Video details section | Page Section | The area has details about the video and its attributes. | | Graphical Section | Page Section(Graphics) | This area will cover the options for the user to view the graphical comparison of one video with another. |  |  |  |  | | --- | --- | --- | | Add Level | Button | To add a level in Multi-level sorting | | Delete Level | Button | To delete a level in Multi-level sorting | | Col | Text Box | Represents the no of columns in multilevel sorting | | order | Text Box | To show order in multilevel sorting | | HeadingVideoAnalysis | TextBox | To Show the Heading | | Subs | TextBox | To Show No of subscribers |  |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Plain | Information Section | Welcome section. | | Input Fields | Input | Username and password input. | | Login Button | Button | Button to login | | |