**fProposer Details**

| Group Number | **Group 03** |
| --- | --- |
| Registration Number of Group Members | **2021-CS-02**  **2021-CS-11**  **2021-CS-35** |

**Proposal Details**

|  |  |
| --- | --- |
| ***Project*** |  |
| Proposed Project Title | **LIBRARY MANAGEMENT SYSTEM** |
| Executive Summary | Libraries are important meeting places in communities and are critical for providing public access to knowledge and information. Our solutions help libraries serve their communities and run more efficiently.  From small school libraries to country libraries, pop-up libraries to big city libraries and special libraries, we help our customers serve their patrons, giving librarians more time to pursue their important work of promoting reading, learning and creativity, supporting researchers and information seekers, and creating a welcoming space for library users.  Library Management System eliminates the hassle of keeping track of all the books and manually entering the data which makes it easier to organize the books and to ultimately run your business in an efficient manner. Using the software results in lower costs which is good for business. All the data is stored in a database and the users can easily access and categorize the books by utilizing the searching and sorting algorithms  The library management system is a software that is designed to manage all the functions of a library. It helps the librarian to maintain the database of new books and the books that are borrowed by members along with their due dates.  This system completely automates all your library’s activities. The best way to maintain, organize, and handle countless books systematically is to implement a library management system software.  A library management system is used to maintain library records. It tracks the records of the number of books in the library, how many books are issued, or how many books have been returned or renewed or late fine charges, etc.  You can find books in an instant, issue/reissue books quickly, and manage all the data efficiently and orderly using this system. Thus, the purpose of a library management system is to provide instant and accurate data regarding any type of book, thereby saving a lot of time and effort. |
| ***Business Case*** |  |
| Outline the business need for the project | What happens when someone makes a mistake in manually entering the data?  Could the time spent in manually entering and indexing the books be utilized in a more efficient manner? Like making plans for the staff or making the library more reader friendly.  What happens if there is a fire or a hazardous chemical on the premises of the library? It would have to close for a few days which could result in losing our customers.  If any of the aforementioned things do happen, pit could prove to be very damaging for business and may ruin the reputation of the library. Thus, implementing a Library Management System would solve 90% of these problems. The data will be secured and easy to handle in case of a wrong entry, this will save time which could then be used to maintain the physical conditions of the library or to go through the business plans.  It will offer 24-7 connectivity so that readers can access their books in a soft copy form in case of an On-sight emergency.  Thus, even a medium to small business would profit by implementing a library management system. |
| End user of the product | This system can be utilized by all libraries for example, Academic or National. Students and avid readers will benefit greatly from this. |
| Motivation for Project | To find out how data flows in a management system and to create something we’re proud of which could benefit others. |
| State the level of impact expected should the project proceed and implications of not proceeding | We will make sure if the implementation does have an impact on the surface level, we had made sure that the user is aware of it and all the necessary information will be provided in the user-manual. |
| ***Technical Details*** |  |
| Name of Entity | *Book* |
| Attributes of Entity  (Minimum seven attributes/rows can be increased) | |  |  |  | | --- | --- | --- | | *Name* | *Data Type* | *Description* | | Title | String | Name of the book | | Author | String | Book Publisher | | Price | Float | Price of the book | | Publish Year | String | Date the book was released on | | ISBN | Int | Book serial number | | Ratings | Float | Reviews of the books in terms of digits | | In Stock | Bool | Availability of the book | | Rating | Int | Rating of Book based on reviews. | |
| Sample of Scrapping Source | [*https://bit.ly/3CKEVn2*](https://bit.ly/3CKEVn2) |
| Gitlab Repository Link | [*https://bit.ly/3CKdWYM*](https://bit.ly/3CKdWYM) |
| Sorting Algorithms | ***Insertion Sort, Bubble Sort, Merge Sort, Selection Sort, Hybrid Merge Sort, Combo sort, Shell Sort, Strand Sort, Quick Sort, Counting Sort, Radix Sort, Bucket Sort, Postman Sort, Pigeon-hole Sort.*** |
| |  |  | | --- | --- | | **Algorithm Name** | **Description(Each algorithm in 2-3 lines)** | | Insertion Sort | Comparison-based stable sorting algorithm. Starts iteration from the 2nd element of array to end, and move each element to left where it fits if necessary | | Bubble Sort | Comparison-based stable sorting algorithm. It keeps comparing adjacent elements and swapping till no swapping is done in one complete iteration of main loop. | | Merge Sort | Comparison-based stable sorting algorithm. Recursively divides array into two parts till sub array cannot be further divided then merges them together in sorted order | | Selection Sort | Comparison-based unstable sorting algorithm. It finds the minimum element of array and swaps it with the starting value of loop till it reaches loop end. | | Hybrid Merge Sort | Comparison-based stable sorting algorithm. Combines merge sort with insertion in such a way that it switches between them depending on the efficiency of algorithms with accordance to elements. | | Comb Sort | Comparison-based unstable sorting algorithm. It’s an improved version of bubble sort. It first handles the small values on end of list (turtles) and big values on start of list (rabbits) and then apply bubble sort. | | Shell Sort | Comparison-based unstable sorting algorithm. It is similar to insertion. It starts by sorting elements far from each other and reduces the gap of elements to be compared after every main loop iteration. | | Strand Sort | Comparison-based stable sorting algorithm. It is recursive algorithm that creates sorted sub-arrays and merges them | | Quick Sort | Comparison-based unstable sorting algorithm. It divides array into two using a pivot such that values smaller to pivot on left and larger on right and we get sorted array after the recursive calls. | | Counting Sort | Linear stable sorting algorithm. Finds max and min of array, creates an array of size max – min and place each element at index (which is equal to its value) of new array. | | Radix Sort | Linear stable sorting algorithm. It creates ten buckets from 0 to 9. Then starts from least significant digit, places element according in bucket and progressively moves to more significant digits | | Bucket Sort | Linear stable sorting algorithm. It groups element into 10 buckets  Based on the place value and sorts them by other algorithms or by recursive calls | | Postman Sort | Linear stable sorting algorithm. It sorts from most significant digit to the least. Length of the largest value is used in process | | Pigeonhole Sort | Linear stable sorting algorithm. Same as Counting Sort, but it moves items twice, one on input array and other in bucket array | | |
| Searching Algorithms | ***Linear, Binary, Jump, Fibonacci*** |
| |  |  | | --- | --- | | **Algorithm Name** | **Description(Each algorithm in 2-3 lines)** | | Linear Search | Starts from beginning of array to end, searches each index of array . it stops at soon as it finds the data or reach end of array | | Binary Search | Divides a sorted array into half . Take the part that has the required value and then keeps on dividing it by comparing till it is left with only one element(Answer) | | Jump Search | Takes a sorted array and makes jump equal to square root of array length till it jump to an index having greater value then the required one then performs linear search on that array. | | Fibonacci Search | It also takes sorted array but divides into unequal part based on Fibonacci series to narrow down possible locations. | | |
| Searching Filters for each data type | A column can be searched by giving at least one input string and using filters:  Is equal to : Required data is exactly equal  Is not equal : Required data not exactly same as provided string  Contains : Required data is sub string of elements  Does not contain : Required data is not sub string of elements  Starts with : Required data is present at beginning of element  Ends with : Required data is at end of element  There will be another input string having same filters and results of both inputs can be compared by AND or OR operator to get desired result. |
| Multi-Level Sorting | Levels are added where array is sorted normally in required order in first level but it can happen that that column has multiple same values so at next level those recurring values are sorted based on another column and so on. |
| Any other features | A graph between time and number of entries scrapped per unit tie shall be displayed on the Scrapping window. Bonus Task *Any URL Scrapping.* |

|  |  |
| --- | --- |
| ***Interfaces for your project*** |  |
| |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Frames | Containers | To group widgets of same semantics. | | Graph | Image | To display ***Scrapp-time*** graph. | | Icons | Image | Icons for Menu. | | Labels | Display Widgets | To display headings and outputs. | | List | Item Widget | To display *failed attributes in scrapping.* | | Progress Bar | Display Widgets | To display progress of scrapping. | | Push Buttons | Buttons | To Start, Pause & Play and End Scrapping. | | Text Box | Input Widget | To Enter URL for Scrapping(*Bonus Task*). |      |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Combo Box | Input Widget | Multiple Combo Boxes are used to make users select column, algorithm and level of *Multi-Level Sorting* and operators & columns in *Multi-Level Searching.* | | Icons | Image | Icons for Menu. | | Labels | Display Widget | To display headings and outputs. | | List | Item Widget | To display levels in *Multi-Level Sorting.* | | Push Buttons | Button | To *Add level, Clear Filter* and *Search* push buttons. | | Table | Item Widgets | To display list of selected entities (Book and its attribute). | | Text Box | Input Widget | To take input for search strings. |      |  |  |  | | --- | --- | --- | | **UI Component Name** | **Type of UI component** | **Purpose of UI Component/Other details** | | Combo Box | Input Widget | To select Stock Status. | | Labels | Display Widget | To display attributes. | | Text Box | Input Widget | To take input for attribute strings. | | |