| **Data Structures and Algorithms** |
| --- |
| [Project-Google 2.0] |
| **Course Project Report** |

| **School of Computer Science and Engineering**  **2021-22** |
| --- |

**Contents**

| **Si. No.** | **Topics** |
| --- | --- |
| 1. | Course and Team Details |
| 2. | Introduction |
| 3. | Problem Definition |
| 4. | Functionalities |
| 5. | Tools and Techniques |
| 6. | Learning and Takeaway |
| 7. | References |

**1. Course and Team Details**

**1.1 Course details**

| **Course Name** | Data Structures and Algorithms  (Theory and Lab) |
| --- | --- |
| **Course Code** | 20ECSC205 and 19ECSP201 |
| **Semester** | III |
| **Division** | B |
| **Year** | 2021-22 |
| **Instructor** | Prakash Hegade |

**1.2 Team Details**

| **Si. No.** | **Roll No.** | **Name** |
| --- | --- | --- |
| 1. | 215 | Sagar Kabadi |
| 2. | 222 | Smitha K V |
| 3. | 207 | Siri K V |
| 4. | 209 | Rohan Kurdekar |

**2. Introduction**

DATA STRUCTURE AND ALGORITHM –

Algorithms help to break down a complex problem into several simple steps so that the computation is essentially understood. This particular course develops the capability of a student in writing algorithms for any given task and allocate a suitable data structure and analyse the efficiency of the same.

After the completion of this course , one can eventually build several design solutions to real world problems using many data structures and algorithms. This course contains a lot of case studies where students work as a team .Which indirectly helps in cultivating good leadership and teamwork qualities.

This project undoubtedly is one of the most integral assessments of our learning throughout the course. This projects questions and demands our implementation skills and optimization capability In solving an issue.

**3. Problem Statement**

A program that simulates than imitates the functionalities of **Google** .A browser which has various functions like search engine which gives users option to search anything with the feature of **AUTOCOMPLETION-**To make searching easier, Mail application providing benefits of sending texts to people, Maps helping the users in calculating distance to reach various places and a Calendar which solves the issue of remembering important dates for the user by providing a function to search important dates and even save the important dates.

**4. Functionalities**

| **SI. No.** | **Function Name** | **Description** | **DS and Algorithm Used** |
| --- | --- | --- | --- |
|  | Load file | Loads the data from the file into appropriate data structure. | Doubly linked list. |
|  | Createll | Adds a new node at the end of the list. | Doubly linked list. |
|  | bubble sort | Sorts the doubly linked list. | Bubble sort algorithm.  on Doubly linked list |
|  | display | Displays the data stored in the doubly linked list. |  |
|  | Link open | Opens the link that the user selects from the options given. | Doubly linked list. |
|  | history | Creates the user search history. | Doubly linked list. |
|  | createldll | Creates doubly linked list and stores the links opened by the user. | Doubly linked list. |
|  | Delete searched | Gives an option to delete a particular link and thus deletes it from the history. | Doubly linked list. |
|  | Sign up | Function reads the datas of the new user and stores in the file. Begins with reading the data from file named "id.txt"  which contains an integer value stored in it indicationg number ids already stored.  After creating the username  and password of the user it stores the drtails in file "signup.txt".  Function also makes sure that user enters valid userid of form [abc@xyz.com] and insists to reenter if not entered  in mentioned syntax. | Structure array |
|  | Sign in | Function reads mailid and password from the user.Allows accces to mail and to send messages only if login is succesfull  checks if the entered mailid matches or is present in (signu) structure if present it captures the index location  in 'count' variable.[ count represents or gives the index number at which signed in user details are present in signu  strucrure array. | structure array |
|  | recovery | Reads the mailid from the user. Asks to enter new passwrd and updates the value in "signup.txt"as well by calling  "printinfile" function |  |
|  | printinfile | This function is called from rcovery function  In this function we print the user detials in signup.txt when any user changes their password |  |
|  | Mail open | Gives the logged in user chance to open inbox or to compose mail,etc. |  |
|  | inbox | Displays the mailids of those who have sent the messages to the user and gives the option to view the selected messages. |  |
|  | sendmails | Displays the user the messages they have sent [ WE HAVENT STORED THIS DATA IN FILE SO ONCE PROGRAM IS CLOSED THIS DATA IS LOST] |  |
|  | Compose | reads the mailid of the user of whom the message has to be delivered then searches and stores the index number in "flag" variable where the  details of user whom the message has to be delivered is present in "signu" structure array created above.  Then the message is copied in the inbox data type inside structure array of selected user. |  |
|  | Loadsignupdata | We have stored the details of the users who signup into the file named as "signup.tct". When program is ran these details are loaded into structure  array signu. This way details are not lost. | Structure array |
|  | storemsgdata | In this function we store the inbox detials of the users into the file named as "loaddata.txt". | Structure array |
|  | loadmsgdata | In this function Inbox detials of the users stored in "loaddata.txt" is loaded into their respective data locations in signu structure array | Structure array |
|  | Trie insertion | To Attain 'AUTO COMPLETE' Feature we have used "TRIE DATA STRUCTURE".  In this function we read the data from searchdata structure array and insert inside trie data structure | Trie data structure |
|  | print\_trie | This functions prints our trie. | Trie data structure |
|  | printintosuggestions | We check if the read string is present inside trie or not | Trie data structure |
|  | islastnode | This Function is used to check if it is the last node of trie or string searching | Trie data structure |
|  | append | In this function We append the half searched string with the topics present inthe brute.txt file that matches the half entered string. | Strings |
|  | print | In this Function we print the character or string by completing half read string | String |
|  | selectchoice | In this function we give user choise to select from the options available from auto complete results. | 2d Character array |
|  | loadEvents | This functions loads the events name and date from file eventList to struct Events |  |
|  | isLeap | This functions returns 1 if the year given is leap else returns 0 |  |
|  | isValidDate | This function returns 1 if the given date is valid (dd , mm , yyyy) else returns 0 |  |
|  | dayNumber | This funtions returns day number for given date if day is monday if returns 1 , for tuesday it returns 2 |  |
|  | getMonthName | This function returns the name of month for given month number jan for 1 , feb for 2 and so on |  |
|  | getDayName | This funtion returns day name for a given int , returns monday for 1 , tuesday for 2 and so on. |  |
|  | addevent | This functions adds the event date , month and name to structure event and sorts it in correct order. | A variation of Bubble sort algorithm |
|  | printEvents | This funtion print the all the events in the events in correct order |  |
|  | sortEvents | This functions sorts the events in events structure  In correct fashion | A variation of Bubble sort algorithm |
|  | findDay | This function prints the day name for the given date |  |
|  | printCalendar | This function prints the calender for the given year  the range of valid years is declared globally |  |
|  | deleteevent | This function deletes the events from the events structure of calender and make sure the other events are in correct order |  |
|  | numberOfDays | This funtions returns the number of days in the given month and year |  |
|  | printNearbyCities | This funtioss prints the nearby city for given city and range input of city name and range has been taken in this function  itself and makes call for dfs for rest of the work | DFS |
|  | dfs | This funtion is used to detect the cities nearby given city and range | Depth first search algorithm |
|  | ClearHistory | This function is used clear the history of search in maps structure |  |
|  | printHistory | This function prints the history present in search of maps |  |
|  | printCities | This function prints the cities which are used in our function |  |
|  | FindThepathbetweenTwocities | This funtions prints the path and distance between cities | Kruskals Algo |
|  | shortestpath | This function is helper function of [FindThepathbetweenTwocities] | Kruskals algo |
|  | addtoHistory | This funtion adds the search to history [Maps] |  |
|  | createAdjancecyMatrix | This funtions creates adjacnecy matrix and loads the correct data into it using edge structure |  |
|  | LoadGraph | This function load the data of graph in to structure |  |
|  | LoadCityNames | This functions Loads the city name from cities to city datails structure |  |
|  | PrintCityDeatails | This funtion prints the city details |  |
|  | printgraph | This funtion prints the Graph |  |

**5. Tools and Techniques**

**5.1 Data Structures and Algorithms**

**5.1.1 DATA STRUCTURES USED-**

Various Data structures are used in the project. Basic Data structures like **Arrays, Strings**

**and Structure** has been used to store some basic information. Some more Complex Data structures used are-

**1-Doubly-Linked-List:** Has been used to store links in the creation of browser functions. as it would be convenient to go back and forward from the present opened webpage easily.

**2-Graph**: As we have Maps. Dealing with Graphs Data structure becomes necessary to store the paths and distances.

**3-Trie**: As we tried to implement the autocomplete feature. Trie data structure was necessary as it is the special data structure used to store strings that can be visualised like a graph. [Even Google uses the same]

**5.1.2 ALGORITHMS USED-**

Various Algorithms like **Bubble sort, Union Find Algorithm, Kruskal’s Algorithm, DFS[DEPTH FIRST SEARCH]** have been used.

Sorting Algorithms has been used in Browser to sort links based on priority, Used in calendar, and even to sort the edges in Kruskal’s. A variation of bubble sort has been used for sporting events in the correct manner.

-DFS has been used to traverse the graph.

**5.2 Project Statistics**

| **Si. No.** | **Measure** | **Value** |
| --- | --- | --- |
|  | Total Functions in Project | 52 |
|  | Total number of lines of code  (Including comments, newlines etc.) | 2018 |
|  | Number of Errors | 0 |
|  | Number of Warnings | 0 |
|  | Team Satisfaction about Project | 100 |

**6. Learning and Takeaway**

There have been many learnings from the project. It's only when we try to implement the projects on our own, we learn things better. Practical exposure has a major role in gaining excellence in any course.

We as a team learnt about many new things while working on the project. DIVIDE AND CODE was the key so that each one of us in the team would have an equal contribution.

We learnt to implement new codes, Learnt to explore the Internet for various references required during various instances.

Lastly, we majorly learnt about the uses and applications of various Data structure and Algorithms in Real life.

**7. References**

[1]https://nivethithaj.blogspot.com/2016/06/implementation-of-gmail-using-c-program.html

[2]https://gist.github.com/aulisius/37ad1b797b88e3d9b8c9

[3]<https://www.geeksforgeeks.org/print-calendar-for-a-given-year/?ref=lbp>

**~\*~\*~\*~\*~\*~\*~\*~**