

|  |  |  |
| --- | --- | --- |
|  | **PES UNIVERSITY, Bangalore**  (Established under Karnataka Act No. 16 of 2013) | **UE18CS202** |
| **B.Tech, Sem III**  **Session : Aug-Dec, 2019**  **UE18CS202 – Data Structures** | | |

**REPORT**

**ON**

**“Restaurant Management System”**

**SECTION :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **SRN** | **Name** | **Contact No.** | **Email ID** | **Sign** |
|  | PES1201800125 | Tanishq | 8839168758 | tanishqvyas069@gmail.com |  |
|  | PES1201801948 | Bharath | 8197939334 | bhambore.bharath@gmail.com |  |
|  |  |  |  |  |  |

**Brief Description :**

**The project aims at implementing a Restaurant Management System using Linked Lists. We have used Linked lists and Multi Linked Lists (2D Linked Lists or in simple language, Linked list of Linked lists).**

**1) Files**

**index.c : client file**

**headers.h : Includes all the pre-processor directives and structure definitions.**

**printMenu.c : Includes functions to print the menu for Admin and Customer along with the common main menu.**

**adminFunctions.c : Includes all the function definitions related to admin's tasks.**

**customerFunctions.c : Includes all the function definitions related to customer's uses.**

**utilities.c : Includes supportive functions**

**2) Important functions :**

**#1 Admin**

**- void showMenu(menu\* head) : Function to print the menu card**

**- menu\* addDish(menu\* head) : Function to add a new dish to the menu (uses linked list internally)**

**- menu\* removeDish(menu\* head) : Function to remove an existing dish from menu**

**- int is\_New\_dish(menu\* head, int id, char\* name) : Function to check for existing dish (to avoid duplication)**

**#2 Customer**

**- records\* placeOrder(menu\* head, records\* record\_head, recordCount\* record\_count) : Function to place an order corresponding to a unique user (uses a stack internally)**

**- records\* cancelOrder(menu\* head, records\* record\_head, recordCount\* record\_count) : Function to cancel an existing order**

**- records\* addToExistingOrder(menu\* head, records\* record\_head, recordCount\* record\_count) : Function to add dishes to existing order**

**#3 Utils**

**- void clearScreen() : Function to clear output screen**

**- void delay(int number\_of\_seconds) : Function to delay the execution time**

**- void loginAsAdmin(menu\* head, records\* record\_head, recordCount\* record\_count) : Function to handle admin login**

**- menu\* buildInit() : Function to setup initial menu card**

**3) Advantages and used Cases :**

**The project application is robust and does not allow repeatition of records. It alows user to make changes in his order at will and keeps the admin functions password protected. No user is allowed to see the details of any other user but himself. Also the admin is given a complete control over the records and menu. This can be used in any medium sized restraunt and canbe expanded easily to be used in a large restaurant. Basic tweaks can also help usutilize this in a much better way by adopting functionalities of stacks, circular queues, etc.**

**4) Logic for implementation :**

**In order to implement the order storage we used linked list of linked lists. The main Linked list contains elements like customer name, bill and a poniter pointing to the head of another linked list which contains the order placed by the customer. At last we have a pointer to the next node in list of this records.**

**Eg with sime dummy values**

**NULl**

**^**

**|**

**Karan**

**200**

**\*-->Roti,1-->Vada,2-->Paneer,1-->NULL**

**^**

**|**

**Aman**

**450**

**\*-->Daal,4-->Paneer,2->>NULL**

**Where the node containing the datas for Aman is the head node for records.**

**We use stack pushing to insert the records into the records linked list and same goes for the data insertion for order nodes in the order linked list for each customer record. The search operations are performed on the records linked list by loooking for a nde with the unique customer name. Deletion and insertion are based on the unique customer name. Since no repetition of record is allowed while insertion (we perform a search and tell whether the data is already present or not) thus deletion just needs to compare the unique name.**

**The displaying of order takes place by displaying the elements of nodes then before proceeding to another node we print all the values in the order node then proceed to next node.**

**This basically completes the order handling.**

**In order to store menu records we used a singly linked list and used insertion at the end in order to store values in the menu linked list. We also perform search and delete operations on this in a way similar to above given way by looking for a unique pair of ID and Dish Name. In case the admin wishes to add a new dish then maintaining the uniqueness of the pair ID,Name we allow insertion in the end of the menu. Displaying the linked list is done using the usual methd till cur pointer reaches.**

**The update order is basically insertion at the end of a previously existing linked list of order.**

**The cancel order functionality is achieved by deleting the node corresponding to the entered customer name while maintaining the right head and next node pointers.**

**In order to add the functionality of login we use standard string comparison wherein the admin enters the username and password. These strings are then compared to the data strings corresponding to correct password and username. If both the entries are correct then person is given admin access.**

**In order to display all the menu we make use of switch case.**

**5) Learning and Scope of Improvement**

**The project helped us gain an indepth knowledge of operation with linked lists and all the associated operations like searching, deletion, insertion, updating, etc. It helped us understand that how can one implement stacks and basic queue insertions using linked lists. The project can be further improved by introducing a delay queue based on the priority of the customer and availibility of seats in the restaurant. Also it can be improved by inclusion of time dependant menu updation i.e. updating the menu linked based on what time it is after reading in data from csv files. The project can make use of a circular queue in order to implement availability in the reataurant.**