**Garage Management System.**

The code can be complied in console using : >g++ Garage.cpp -o Garage

1. Introduction

To develop this mini project called "*Garage Management System*" I used C++ and Kate editor to write/edit code. As the name says this application is useful in garage for basic use like to show all Scheduled Appointments or to add, update and delete an appointment, then to check the total of how many cars were serviced in the garage and total pending cars and the total number of cars entered the garage. It also shows the employee details like Employee id, Employee Name, Birthdate, Joining Date..Etc. and can also use to maintain Garage inventory details. Each section i.e. Employee, Inventory, appointments have basic functionality of add, update & delete options. Text file is used to store the data and to populate the data on the screen. Three different text files are used like garage.txt, employee.txt, iv.txt (file to read Inventory details).We can also use .csv file with minimal code changes in display parameters.

To navigate through the Menu in this application I have made use of numbers for example: 1, 2, 3, 4. Etc. On any selection it will first ask the user for the file name and path. Then it will show the data in the file and it will prompt user if he/she wants to add or update or delete any records present in the file. In total number of cars menu it will only ask for filename and display the total number of cars, total number of cars serviced and total number of cars pending.

In Scheduled Appointments, Employee details and Inventory menu if the user wish to add or update the record the system will ask for specific details and it will write to the file. For example in Scheduled Appointments if a user select Add record it will ask for Booking id, Appointment date, Customer name, customer number, car details, status of the car and cost of repair.

The user is expected to be one of the Garage Member so he /she inputs the correct details in the correct fields.

2. Approach

Firstly, I have created a Main menu class in which i have created a function to upload the file which displays the data on the console on the selection of menu and after entering the required filename. Then I have created other classes like scheduled Appointments, Employee details (empdetails), inventory and for the total number of cars (totcars). In this inherited classes I have written three main function to add, update and delete. In all these functions I have created variables as per the column names to store the data by making use of in build getline function. In main class to make selection of menu i have made use of

*if-else* loop so that when the user inputs the number the system enters the specific *if-else* loop and calls the class function associated with that selection. I have also used flag to return to the main menu if the user is in anyone of the sub menu.

3. Issues Faced

Firstly I tried to develop my project using the sqlite3 database but later on I was not able to import the csv into the database easily, so I dropped the plan of using sqlite3 and I moved forward to use csv file and txt file.

Then my aim was to ask user to enter the file name only once but when the main class was finished the variable used to go out of scope. There are many ways to fix this but since there was less time and the other main functions need to be completed I chose to create a new variable in each class and function.

I wanted to create add, update and delete function generalized for all the three classes but then again the number of columns and variables used for those columns was different so i have to create these functions in each of the three class to work separately as per the classes their defined inside.

While using char\* pointer to read the filename it was working fine in some scenarios but sometimes it was giving segmentation fault error, to fix this i used the “char\* pointer = new char” so that it is stored in the heap memory and it will read the filename properly.

The main issue was while reading the file line by line and then by each column, the data was not getting separated by column at first. Then I used one variable to store each line which was present in the file and then I breaked it into different variable as per the columns using stringstream in build function.

4. Function Explanation.

UploadFile ():

The uploadfile function will take the filename as input from the user and will display the data using pointer to pointer array and *while* loop. The string array is used to store the data temporarily. Then each line is read and inserted in the array after which the loop is incremented to read the next line. Also the fstream functions like ifstream, is\_open and close is used to open the file to read and to close the file after successful reading.

The Addbooking(), Upload\_status(), delete\_record() functions are similar in all the three classes with different variables as per the columns used in that particular class.

Addbooking ():

In this function the user is asked to input the data by column and the data is fetch using getline function to store the data in the specific variable.

update\_status ():

In this function the data is read again by using the getline function and is stored in a specific variable later which is divided into columns using the stringstream inbuild function. Also while updating, each line is written to a new temp file as an updated line and later everything is written back to the main file which was given as input by the user.

delete\_record ():

The delete function works exactly as the update function mentioned above only the change in this function is that the line which the user wants to delete will not be written to the temp file and will be skipped so that the record is deleted from the original file.

Countcars ():

This function is only used in one class to count the total number of vehicles, the number of vehicles serviced and pending for servicing.

This function has one *while* loop to read the lines as per other functions and *if-else* loop in which I have compared the status of the cars like serviced and service pending.

If string status is equal to complete the counter name ccount will be incremented and if the string status is equal to pending the counter name pcount will be incremented and at the end once all the lines are read by the system it will display the counters. The same logic is used to calculate the Total number of cars.

5. Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Inputs** | **Expected behavior** | **Observed behavior** |
| Add record() | Select menu 1,3,4 and submenu 1  Enter all item data | New record added to the system | New record added |
| update\_status() | Select menu 1,3,4 and submenu 2  Enter the ID and if the id is present. Enter all item data | Record will be updated in the system. | record updated |
| delete\_record | Select menu 1,3,4 and submenu 3  Enter the ID and if the id is present. | Record deleted | Record deleted |
| Countcars | Select menu 2  Enter the filename. | To display the total number of cars, total number of completed cars and pending cars | displaying the total number of cars, total number of completed cars and pending cars |

To test the system I have used garage.txt, employee.txt, iv.txt the code can also be tested on csv files.

There are certain limitations in the system as mentioned below:

The Add(), Update(), Delete() functions fails to deliver basic validations like while system accepts NULL input without any error.

In update function as well if a string is not matched the system automatically gets redirected to main menu() without any message displayed to user and same happens in delete function.