PROBLEM DEFINITION

Our project is based on the airline reservation system management. It deals with the following objectives....

- It aims in providing the end user with the following menu options-
    - ➢ Flight Input
    - ➢ Customer Input
    - ➢ Check number of seats booked under given flight
    - ➢ Check number of tickets cancelled in a given month
    - ➢ Check number of tickets booked in a given month
    - ➢ Check number of tickets booked between two given cities
    - ➢ Cancel booked ticket
    - ➢ Check flight details
    - ➢ Check Customer details
    - ➢ Check booked ticket details
    - ➢ Quit
- Our project also aims to be user friendly and hence it becomes mandatory for us to show the end user only the details that he/she requires.
- We aim in giving the final output of the program with proper inundations and attractive formatting.
- We also aim to store the data entered by the end user for later reference using the concept of file handling.
- We aim to implement the advanced concepts of object oriented programming (C++) like data abstractions and data hiding.
- We strive to manage the concept of data consistency throughout our project (like checking for proper values of date and time!)
- On the whole the objective of our project is to satisfy the requirements of our end users to the maximum possible extent...

Flight input

- When details of a particular flight are to be entered in the database the airlines management can make use of the option 1 in the main menu.
- Here indirectly *write_flig*ht *()* is invoked.
- The function waits for the end user to enter the correct password.
    - If the password is correct, the Flight number is to be entered first.
        - Each flight's details from the file *flight.dat* is read and checked if already the flight number exists.

    1. If it exists then the message *"FLIGHT NUMBER ALREADY EXISTS.PLEASE RE-ENTER DIFFERENT NUMBER."* is displayed and a new flight number is to be entered.
    2. **A)** Else other details like *flight name, origin city, destination city, flight capacity, flight type* and *working condition* is to be entered. If invalid flight *capacity* or *flight type* or *working condition* is entered then a message is displayed asking the user to correct the particular detail.

       B) The *timing* of the flight also has to be entered for *morning* and *evening slots*. Input is taken in the format *hh: __, mm: __, ss: __*. Here also if invalid time is entered then a message is displayed asking the user to correct the timing.
    - Else the message "*operation illegal"* is displayed.

Flight details

- When the customer wants to know if a particular flight is available he/she can choose option 8 in the main menu.
- Here the *case 8:* in the *switch* case(main function) is executed.
- The end user has to enter the flight number.
- It reads the details of all the flights under the file *flight.dat* and checks if the flight number entered by the user matches with any of the flight details.
    1. If it matches, the function *flight_output* is invoked using the object Cf and the details of the particular flight are displayed

2. Else the message *"Does not exist"* is displayed.

Quit

- When the end user chooses option 11 the exit() function is executed and the program run is terminated.

Ticket details

➢ When option 10 is chosen by the end user, case 10 of the switch case (main function) is executed.
➢ The flight number, seat number, date and time slot of the ticket is to be entered by the user.
➢ If invalid date or time is entered then a message stating "*Invalid date"* or "*Invalid slot"* is displayed asking the user to correct the date or time.
➢ All the details of the tickets booked are read from the file "*ticket.dat"* and if the details that are input, matches with any of the ticket details in the file the details of that ticket are displayed.
➢ Cancelled ticket which has been replaced will not be displayed.
➢ If such a ticket does not exist then the message"*Does not exist"* is displayed.

Customer details

➢ When option 9 is chosen by the user, case 9 of the switch case (main function) is executed.
➢ The email id of the customer is to be entered by the user.
➢ All the details of the customers are read from the file "*customer.dat*" and if the email id entered by the user matches with any of the customer details in the file the details of that customer are displayed.
➢ If such an email id does not exist then the message "*DOES NOT EXIST*" is displayed.

View files

- ➢ This menu option leads the end user to a page which gives them 4 more sub menu options-

  1. Flight.dat
  2. Customer.dat
  3. Ticket.dat
  4. Cancelled.dat

- ➢ The user is asked to choose one particular option. This option leads the user further to a page where the user is able to read the relevant details stored in the given file in an attractive gridded format.

- ➢ The above menu option uses the function view_files. This function runs on a switch case basis and each case (1-4) opens up the relevant file and reads the data from it and displays only the required information. The default case as usual displays the user an “Invalid choice”.

- ➢ This main menu option enables the program users to go through the files of the program in a much easier manner as compared to the opening up of actual .dat files.

Cancellation:

This option is for the customer to cancel a booked ticket.

- The customer has to enter the date of cancellation.
- If the cancellation date is one day before the booked date then 90% of the money is returned back If 2to 6 days before the booked date then 85% of the money is returned .If 7 or more days before then only 80% of the money is returned else if cancelled after the date of journey 50% of the money is returned.
- The ticket details which are cancelled is stored in cancelled.dat

Customer input:

This option allows the user to become a customer of our airlines and book tickets.

- If the user is already a customer then he/she is requested to enter credit card number. By matching the entered card number assigning process would start .An error message is reported if the card number does not match with any of the customer's details.
- If a new customer, then he/she is requested to enter his/her personal details like first name, last name, age, address, phone no.,etc.
- Phone no. should have exactly 10 digits .Age entered should be less than 100.If conditions not satisfied an error message is reported.
- The details accepted from the user are stored in a separate file customer.dat.
- Now for ticket booking the customer is requested to enter starting and ending place.
- As per the choice of the customer the available flights are displayed and the customer is allowed to choose the flight using flight number .An error message for invalid flight number.
- And now the customer has to select the slot of travel.
- Now choice for the class available in the flight selected is displayed .An error message is reported for an invalid class .If the flight full under the wanted class the customer is allowed to select another class.
- The customer is given choice for the seats (aisle, middle, and window).
- The detail for travel date is retrieved.
- Now the file ticket.dat is read, if a ticket is cancelled for the same details of the flight then this seat is allotted for the new customer .Otherwise a fresh new seat is allotted.
- The ticket for the customer is now assigned .The ticket details is stored in a separate file Ticket .dat.
- The ticket for the customer is displayed which consists of the date, time, flight number, seat number, etc.

COUNTING SEATS BOOKED IN A PERTICULAR FLIGHT

When the end user chooses the option 3 in the main menu, it takes him to a fresh page. The function count_seats_of_flights() is invoked.

- ➢ The user has to give an input quoting the flight number for which he desires to find the number of seats booked
- ➢ Input is taken using the inbuilt function gets() .
- ➢ A counter variable is declared and initialised to 0.
- ➢ The file “tickets.dat” is opened. This file stores all information regarding tickets booked or cancelled in various flights.
- ➢ The records read are stored in a temporary variable CT(object of class TICKET_MASTER).
- ➢ The function retfno() ,a member of ticket_master is invoked and the value it returns is compared to the value entered by the user.
- ➢ If this value matches, further checking is done weather the value in “cancellation” ,also a member of ticket_master is equal to “n” .The cancellation variable keeps track of weather the user has cancelled his ticket or not. It holds a value ”n” as long as the user doesn’t cancel his ticket
- ➢ If both the above conditions are satisfied, the counter variable is incremented
- ➢ This process of reading record, checking the condition and then incrementing the counter is carried on as long as the end of file is not reached.
- ➢ Finally, the number of tickets booked is displayed as the value stored in counter variable.

NUMBER OF TICKETS CANCELLED IN A PERTICULAR MONTH

When the end user goes in for the choice 4 in the main menu, it takes him to a fresh page. The function no_of_tickets_cancelled().

- The user is asked to enter the month and year for which he desires to know the number of tickets cancelled.
- A counter variable is declared and initialised with the value 0.
- The value entered are checked to lie within the range of 1 to 12(since the Gregorian calendar contains only 12 month cycles)
- If the value entered by the user is not within the above mentioned dates, an error message saying “invalid month” is displayed.
- Similarly, the year entered by the user is checked to lie within the range of 1900 and 2050
- The file “cancelled.dat” , which contains information regarding cancelled tickets is opened.
- Records present in this file are read and stored in the variable CH (object of class TICKET_MASTER).
- The function retcdate() ,a member of ticket master, is invoked and its value is stored in a temporary variable “d” of type date. The value of elements m, y (signifying month and date) of “d” is compared to the values entered by the user.the value returned by retcancel() is also checked to be equal to “Y”. This signifies that the user has cancelled the ticket.
- If all three values match, counter variable is incremented.
- This process of reading records, checking conditions and incrementing the counter continues until the end of the file is reached.
- Finally the number of tickets cancelled in a particular month is displayed to be the variable counter

COUNTING THE NUMBER OF TICKETS BOOKED IN A PERTICULAR MONTH

When the end user goes in for the option 5 in the main menu, it takes him to a new page. The function count_tickets_booked_in_a_month().

- ➢ The user is asked to enter a month and year.
- ➢ A counter variable is declared and initialized with the value 0.
- ➢ The value entered for the month by the user is checked to lie within the range of 1 to 12(since the Gregorian calendar consists of only 12 month cycles).Similarly the value of year is checked to be within the range of 1900 to 2050 .
- ➢ If any of the above values do not lie in the specified ranges, an error message is displayed and the user is asked to re-enter appropriate values.
- ➢ The file “tickets.dat” which contains all information regarding tickets is opened.
- ➢ Record are read from this file one by one and stored in the temporary variable CH (object of class TICKET_MASTER).
- ➢ The function retddate(),a member of TICKET_MASTER is invoked and the value returned by it is stored in a temporary variable d of type date.
- ➢ The values of members m and y(corresponding to month and year respectively) of d are compared with the values entered by the user. The value returned by retcancel() (member of TICKET_MASTER) is checked to be equal to “y”. This variable keeps track of weather the user has cancelled his ticket or not.
- ➢ If all of the above conditions are satisfied, the counter variable is incremented by one.
- ➢ This process of reading records, checking conditions, and incrementing the counter goes on until the end of file is reached.
- ➢ The number of tickets booked in that particular month is then displayed to be the value in the counter variable.

TO COUNT THE NUMBER OF TICKETS BOOKED BETWEEN ANY TWO GIVEN CITIES

When the user goes in for the option 6 in the main menu, he is taken to a fresh page.

The function reports() is invoked.

- ➢ The user is asked to enter the starting place and ending place.
- ➢ The input given by the user is stored in the string variables spp and dpp.
- ➢ A counter variable is declared and initialized as 0.
- ➢ Record are read from this file one by one and stored in the temporary variable CH (object of class TICKET_MASTER).
- ➢ The file “tickets.dat” containing all information about the tickets is opened.
- ➢ Records are read from the file and stored in the temporary variable CH(object of class ticket _master)
- ➢ The values of the members returned by the function retdplace() and the function retsplace() (both members of TICKET_MASTER)are compared respectively to dpp and spp using the inbuilt function strcmp().The value returned by retcancel() is also checked to be equal to “n”. This variable keeps track of weather the user has cancelled his ticket or not.
- ➢ If all the above conditions are satisfied, the counter variable is incremented by 1.
- ➢ This process of reading records, checking conditions. And incrementing counter is carried on until the end of file is reached.
- ➢ Finally the number of tickets booked between two cities is displayed to be the value in the variable counter.

FUTURE ENHANCEMENTS

The following enhancements could be made to our project to improvise and perfect it....

- ➢ We can provide a separate account for each and every registered customer.
- ➢ We can make the project more presentable by maintaining a link between all our customers with the respective tickets they have booked.
- ➢ We can input the run time date for cancellation date by including headers like <time.h> to make the project more real.
- ➢ Details of flights with poor working conditions are left to go a waste as there is no function provided to modify its status.
- ➢ A huge amount of variables having static memory have been used. Instead of this dynamic memory allocation using pointers could be preferred in the future.
- ➢ We can make the program terminate by default if null value is being entered in place of any required detail using exit function.
- ➢ We can also make the program to terminate by default if there is a data type mismatch in the data entered by the user using exit function.
- ➢ More menu options could be included to check the reports of different flights in a given month.
- ➢ Separate menu could be maintained for the staff members of the airlines, where their salary updates, time table and a lot more could be displayed.
- ➢ Formatting of output like font size, text color, background color perfect spacing, etc could be enhanced using graphics.
- ➢ We can even provide attractive cover designs for every page opened by our project menu using graphics.

BIBLIOGRAPHY

http://www.google.com.

http://www.iCBSE.com

http://cbse-sample-papers.blogspot.com