

Turnitin Originality Report

Processed on: 17-Jan-2024 23:55 +08

ID: 2272558352

Word Count: 1616

Submitted: 2

DSA_project By Ariff Danish

Similarity Index

9%

Similarity by Source

Internet Sources: 1%
Publications: 0%
Student Papers: 8%

4% match (student papers from 20-Dec-2023)

[Submitted to Universiti Teknologi Malaysia on 2023-12-20](#)

2% match (student papers from 07-Mar-2023)

[Submitted to Johns Hopkins University on 2023-03-07](#)

1% match (student papers from 26-Sep-2021)

[Submitted to University of Sydney on 2021-09-26](#)

1% match (student papers from 22-Oct-2023)

[Submitted to Oklahoma State University on 2023-10-22](#)

1% match (student papers from 22-Aug-2011)

[Submitted to Higher Education Commission Pakistan on 2011-08-22](#)

1% match ()

[Nuzzo I., Caterino N., Novellino A., Occhiuzzi A.. "Computer-aided decision making for regional seismic risk mitigation accounting for limited economic resources", 'MDPI AG', 2021](#)

FACULTY OF COMPUTING 20232024/1 SECJ2013 – DATA STRUCTURES & ALGORITHMS SECTION 02 GROUP PROJECT : AIRLINE RESERVATION SYSTEM
LECTURER NAME : MRS LIZAWATI MI YUSUF GROUP NAME : GUSION NAME
MATRIC ID CHE MARHUMI BIN CHE AB RAHIM A22EC0147 MUHAMMAD ARIFF
DANISH BIN HASHNAN A22EC0204 MUHAMMAD IMAN FIRDAUS BIN
BAHARUDDIN A22EC0216 TABLE OF CONTENT 1.

Objective.....

3 2. Synopsis

3. Class Design

4. Problem

Analysis.....

6 5. Data Structure Implementation

7 1. Objective

The objective of this project is to develop our own model of the Airline Reservation System. This system's aim is to ease the process for booking/reserving purposes for customers, while for administrator or staff is to validate the customers request, update or remove the customer from the booking list, search for customers and get information of customers when displaying the list with the user-friendly menu. The main idea and implementation of this project is to develop the system using the data structure concept of queue and must be correctly applied using C++ language. In this project we implemented the queue using a linked list. • Customers successfully request for the booking(enqueue) • Customers can undo the request for reservation(dequeue) • Admin can validate or approve the request reservation or booking from customer(dequeue from request list then enqueue to booking list)

- Admin can view details of customers list
- Admin can remove customer from booking list(dequeue)
- Admin can search for customer using IC number
- System provides user-friendly menu driven

2. Synopsis the implementation of our airline reservation system, which has features for administrators and customers alike. Along with variables like name, destination, airline, booking date, and IC number, it has a Cust class that represents customers. The BookingQueue class, which uses linked nodes to construct a simple queue structure, is used by the programme to manage reservations. The system is managed by the BookingSystem class, which keeps track of two queues: reservationRequests for pending reservations and doneBooking for completed ones. The ability to approve reservations(queue and dequeue process), view all reservations, search for reservations by customer IC number, and remove(dequeue process) reservations are among the features that administrators can access. Conversely, customers have the ability to make reservations(queue process), cancel their most previous reservation(dequeue process), and return to the main menu. Booking data is taken from a file (booking.txt) by the programme to initialise data. It communicates with users via console I/O, manages queues via linked lists, and handles errors resulting from incorrect selections. The admin, customer, and programme exit options on the system's main menu are customised to each user role's needs. The system offers an extensive feature set for efficient reservation management with the ultimate goal of simulating the workflow of an airline reservation system.

3. Class Design PSEUDOCODE Here's a pseudocode for the main part of our program in project: `` `plaintext

```

class Customer attributes: name, destination, airline, day, month, year, IC
class BookingQueue class Node attributes: data (Customer), next (Node)
attributes: front (Node), rear (Node)
method enqueue(data) create Node with data if isEmpty() front = rear = newNode else rear.next = newNode rear = newNode
method dequeue() if not isEmpty() temp = front front = front.next delete temp
method isEmpty() return front == null
method getFront() if not isEmpty() return front.data else throw out_of_range("Queue is empty")
class BookingSystem attributes: doneBooking (BookingQueue), reservationRequests (BookingQueue)
method addBooking(booking) doneBooking.enqueue(booking)
method reserveBooking(booking) reservationRequests.enqueue(booking)
method cancelRecentReservation() if not reservationRequests.isEmpty() reservationRequests.dequeue()
print "Most recent reservation canceled." return true else print "No reservation requests to cancel." return false
method approveReservation() if not reservationRequests.isEmpty() process all reservations
print "All reservations approved." else print "No reservation requests to approve."
method searchByICNumber(icNumber) for each booking in doneBooking if booking.getIcNum() == icNumber display booking details return true
print "No bookings found for IC Number " + icNumber return false
method removeBookingByIC(icNumber) find and remove booking by IC number
print "Booking for IC Number " + icNumber + " removed."
method displayAllBookings() display all bookings
method displayRequestedBookings() display reservation requests
function extractData(bookingSystem) open "booking.txt" if file not found print "Cannot open file."
return else read lines from file extract and add data to bookingSystem
function customerMenu(bookingSystem) do display menu read user choice switch user choice
case 1: input customer details create Customer bookingSystem.reserveBooking(newCustomer)
print "Reservation request sent."
case 2: input IC number bookingSystem.cancelRecentReservation()
case 3: print "Returning to main menu."
while user choice is not 3 function adminMenu(bookingSystem) do display menu read user choice
switch user choice
case 1: bookingSystem.approveReservation()
case 2: bookingSystem.displayAllBookings()
case 3: print "Returning to main menu."
case 4: input IC number bookingSystem.searchByICNumber(IC number)
case 5: input IC number bookingSystem.removeBookingByIC(IC number)
while user choice is not 3
function main() create BookingSystem call extractData(bookingSystem) do display main menu
read user type choice switch user type choice
case 1: call adminMenu(bookingSystem)
case 2: call customerMenu(bookingSystem)
while user type choice is not 3 pause and return 0
4. Problem Analysis Airlines Reservation System needs to handle customer bookings and reservation efficiently to provide the best service thus need the reliable data transfer in and out. It needs to make sure that the data is accurate.

```

Apart from that, the interface for user and the admin also need to be simple and nice as this can improve usability and ability to handle problems especially in handling the reservation and booking made. Below are the problem that lead to the making of this project: - User interface - Error handling - Dynamic memory management for booking enqueue - Data reliability and accuracy

5. Data Structure Implementation

- Class BookingQueue The "BookingQueue" class represents a queue data structure. Specifically, it's an implementation of a queue using a linked list. The class includes the basic operations associated with a queue, such as enqueue (adding elements to the rear), dequeue (removing elements from the front), checking if the queue is empty, and retrieving the front element.
 - a) Enqueue Operation: The enqueue method is used to add a new booking to the queue. It takes a Cust object as a parameter and creates a new node with this data. If the queue is empty, both front and rear are set to the new node. Otherwise, the new node is added to the end of the queue, and the rear is updated.
 - b) Dequeue Operation: The dequeue method removes the front node from the queue. If the queue is not empty, it updates the front pointer to the next node, and the memory occupied by the front node is freed.
 - c) getFront Operation: This method returns a reference to the data (Cust object) stored in the front node of the queue. The if (!isEmpty()) condition ensures that the queue is not empty before attempting to access the front node. If the queue is not empty, it returns a reference to the data of the front node (front->data). If the queue is empty, it throws an out_of_range exception with the message "Queue is empty."
 - d) isEmpty Operation: This method check if the queue is empty by verifying if the front pointer is equal to nullptr. If front is nullptr, it means the queue is empty, and the method returns true. Otherwise, it returns false, indicating that the queue is not empty.
 - e) getFrontNode Operation: The getFrontNode method is used to obtain a pointer to the front node. This is utilized in various parts of the program, such as when approving reservations.
- Class BookingSystem In the BookingSystem class, the queue is implemented using two instances of the BookingQueue class: doneBooking and reservationRequests are private member variables of type BookingQueue. These variables represent two separate queues within the booking system.
 - doneBooking: This queue is used to store all the completed bookings (read from "booking.txt").
 - reservationRequests: This queue is used to store reservation requests made by customers.
 - a) addBooking Method: The addBooking method, completed bookings (read from "booking.txt") are added to the doneBooking queue.
 - b) reserveBooking Method: The reserveBooking method, reservation requests made by customers are added to the reservationRequests queue.
 - c) approveReservation The approveReservation method, if there are reservation requests, they are dequeued from the reservationRequests queue and then enqueued into the doneBooking queue.
 - d) displayQueueBookings Operation: displayQueueBookings is responsible for traversing the nodes of a given BookingQueue and displaying the booking information stored in each node. It's a way to visualize the contents of the queue.

6. User Guide

I. Admin/Staff

A. The main page of the Airline Reservation System.

B. Please enter the admin password before login.

C. If user choose '1', it will display the approvement of the customer booking

D. If the user chooses '2', it will display all the booking information.

E. If the user chooses '4', it will show a prompt for user to begin searching for information based on customer IC and display the related information if found. Else, it will display 'No bookings found for IC number: '.

F. If the user chooses '5', the system will ask the users to enter the IC number that will delete the booking. Then, its display the booking that was deleted is not in the list.

G. If user choose '3', the system will exit and end.

II. Customer

A. The main menu for customers.

B. If the user chooses '1', the system will ask the user to insert a new customer name, date of booking, destination, type of airline and IC number. If the details are valid, it will notify the user 'Reservation request sent successfully'.

C. If the user chooses '2', the system will delete the previous booking and notify with 'Most previous reservation canceled successfully'.

D. If the user choose '3', it will go back to the main menu.