

CMPT 2021: Algorithms
Assignment: Design of a Dynamic Ticketing System
Paula Frossard

This document aims to serve as a detailed report to explain the code, implementation and logic of the file “*CMPT2021_Paula_Frossard_Ticketing_System.ipynb*”

1. Introduction

As presented by the instructor, this project aims to “*Present a Dynamic Ticketing System that efficiently handles event ticket bookings, cancellations, and modifications while managing overbooked requests using a waitlist system. The solution should show the implementation of data structures such as dynamic arrays, hash tables, and priority queues to achieve these functionalities.*”

The program enables efficient seat booking, cancelling a ticket, modifying a ticket and waitlist system based on priority happening in case the event is sold out. It also allows for checking for tickets that were booked and their respective information, as well as increasing the number of available seats if needed. Implementation focuses on utilising the specified data structures properly. This specific implementation deals with 500 seats available initially.

2. Scenario

Famous band is performing at a 500-seat (initially) place in Edmonton. It has high demand, so a system is needed to make managing bookings easier. Customers should be able to choose seats, and in case of all seats already booked, they can be placed in a waitlist based on their priority level. Then, due to high demand, the event increases seats availability from 500 to 600. The system then adapts by adding 100 more seats.

3. System Components

3.1. Seat grid

With a dynamic array, the seat layout is a 20x25 grid (20 rows, 25 columns), starting with 500 available seats. Seats are labeled based on rows (A to T) and columns (1 to 25), making seat row 1(A), column 1(1) be A1, seat row 1(A), column 2(2) be A2, and so on. Each row is a list of seats (assigned as None for available and the ticket id in case it's booked)

3.2 Display

The method **show_seats** displays the layout of the seats. Icons are used to visually represent if a seat is empty or booked,  being for available and  for booked seat. The seat label is placed on the left of the visual indication (free/booked) of each seat.

3.3 Booking

The system checks if a seat is available with the method **seat_availability**, when booking a seat. If the seat is available, ticket id is generated and linked to this seat, together with user

information (for simplicity, we are currently only collecting Name and email from user and linking them to the user id), all in a dictionary.

When a seat is booked, it then goes from available to booked.

3.4 Storing data

The system contains a dictionary as a hash table for keeping the information related to the booking. The ticket id is the key, and the value is the dictionary mentioned earlier with user id, name, email, priority and seat. This facilitates the use of checking for information quickly.

3.5 Waitlist

When there are no available seats (all seats are booked), the user has the option of being added to a waitlist. Then, the **WaitlistQueue** class manages everything related to the waitlist, including the priority of each user (1 is for VIPs, 2 for special and 3 for regular).

When a cancellation happens, the waitlisted person with the highest priority in the waitlist is assigned to that seat. If there isn't anybody in the waitlist, the seat becomes available again.

3.6 Cancelling a ticket

cancel_booking method allows users to cancel their bookings when providing their ticket id. If the waitlist has a user listed, the highest priority one will be assigned to this ticket, otherwise, as mentioned earlier, the seat becomes available again.

3.7 Updating a ticket

Users can also modify their booking with the **modify_ticket** method. The only things they are allowed to change at the moment are their chosen seat and their priority level. This also updates the grid, making the old selection available and turning the new option (if available) as booked)

3.8 Increasing number of seats

In case all seats are booked, and the waitlist is getting more and more customers, the system can increase seat capacity. This is done by the method **increase_seating_capacity**, by adding new rows to the grid, and keeping the columns the same size.

To avoid errors, I added a limitation of seats based on the letters A to Z, so a possible improvement for systems that require constant increasing capacity would be to exchange letters by numbers, this way, we would always have 25 letters (being columns, A to Y), and an infinite possibility of rows to be increased, since they would be the numbers.

4. Features

Feature	Code	Type Function / Method / Variable / Data Structure	Description
Booking a seat	booking_seat()	Method	Books seat and stores booking with customer info
Generating ticket ID	generate_ticket_id()	Method	Creates unique ticket id based on event code, seat, and user id
Checking if there are	seat_availability()	Method	Checks if selected seat is available to be booked

available seats			
Seat display	show_seats()	Method	Shows seat grid with visual symbols for booked and available seats
Seat structure	seat_grid_create()	Method	Creates initial seat grid using dynamic array
Storing ticket info	self.ticket_info	Dictionary (hash table)	Stores ticket and user details using ticket id as key
Managing waitlist	WaitlistQueue	Class (priority queue)	Manages list of users waiting for an available seat based on priority level
Adding to waitlist	add_to_waitlist(), add_user()	Method	Adds users to waitlist with optional priority value (regular as default)
Viewing waitlist	display_waitlist()	Method	Displays all users in waitlist
Cancelling and reassigning	cancel_booking()	Method	Cancels a booking and reassigns seat if someone is waitlisted
Changing a booking	modify_ticket()	Method	Allows customer to change seat or priority level
Increasing seat capacity	increase_seating_capacity()	Method	Dynamically adds rows to seat grid to accept more bookings
Checking for ticket details	check_ticket_details()	Method	Retrieves and shows information about specific booking based on ticket id
Booking all seats	book_all_seats()	Method	Created to test waitlist methods: fills all seats with placeholder bookings to test use cases for waitlist
User friendly testing cases	run_ticketing_system()	Function	Runs a no code interface to test all functionalities of this system

5. Examples of interactions

5.1 Initial seating display

Select an option (1 to 11): 1	
A1	A2
B1	B2
C1	C2
D1	D2
E1	E2
F1	F2
G1	G2
H1	H2
I1	I2
J1	J2
K1	K2
L1	L2
M1	M2
N1	N2
O1	O2
P1	P2
Q1	Q2
R1	R2
S1	S2
T1	T2
A3	A4
B3	B4
C3	C4
D3	D4
E3	E4
F3	F4
G3	G4
H3	H4
I3	I4
J3	J4
K3	K4
L3	L4
M3	M4
N3	N4
O3	O4
P3	P4
Q3	Q4
R3	R4
S3	S4
T3	T4
A5	A6
B5	B6
C5	C6
D5	D6
E5	E6
F5	F6
G5	G6
H5	H6
I5	I6
J5	J6
K5	K6
L5	L6
M5	M6
N5	N6
O5	O6
P5	P6
Q5	Q6
R5	R6
S5	S6
T5	T6
A7	A8
B7	B8
C7	C8
D7	D8
E7	E8
F7	F8
G7	G8
H7	H8
I7	I8
J7	J8
K7	K8
L7	L8
M7	M8
N7	N8
O7	O8
P7	P8
Q7	Q8
R7	R8
S7	S8
T7	T8
A9	A10
B9	B10
C9	C10
D9	D10
E9	E10
F9	F10
G9	G10
H9	H10
I9	I10
J9	J10
K9	K10
L9	L10
M9	M10
N9	N10
O9	O10
P9	P10
Q9	Q10
R9	R10
S9	S10
T9	T10
A11	A12
B11	B12
C11	C12
D11	D12
E11	E12
F11	F12
G11	G12
H11	H12
I11	I12
J11	J12
K11	K12
L11	L12
M11	M12
N11	N12
O11	O12
P11	P12
Q11	Q12
R11	R12
S11	S12
T11	T12
A13	A14
B13	B14
C13	C14
D13	D14
E13	E14
F13	F14
G13	G14
H13	H14
I13	I14
J13	J14
K13	K14
L13	L14
M13	M14
N13	N14
O13	O14
P13	P14
Q13	Q14
R13	R14
S13	S14
T13	T14
A15	A16
B15	B16
C15	C16
D15	D16
E15	E16
F15	F16
G15	G16
H15	H16
I15	I16
J15	J16
K15	K16
L15	L16
M15	M16
N15	N16
O15	O16
P15	P16
Q15	Q16
R15	R16
S15	S16
T15	T16
A17	A18
B17	B18
C17	C18
D17	D18
E17	E18
F17	F18
G17	G18
H17	H18
I17	I18
J17	J18
K17	K18
L17	L18
M17	M18
N17	N18
O17	O18
P17	P18
Q17	Q18
R17	R18
S17	S18
T17	T18
A19	A20
B19	B20
C19	C20
D19	D20
E19	E20
F19	F20
G19	G20
H19	H20
I19	I20
J19	J20
K19	K20
L19	L20
M19	M20
N19	N20
O19	O20
P19	P20
Q19	Q20
R19	R20
S19	S20
T19	T20
A21	A22
B21	B22
C21	C22
D21	D22
E21	E22
F21	F22
G21	G22
H21	H22
I21	I22
J21	J22
K21	K22
L21	L22
M21	M22
N21	N22
O21	O22
P21	P22
Q21	Q22
R21	R22
S21	S22
T21	T22
A23	A24
B23	B24
C23	C24
D23	D24
E23	E24
F23	F24
G23	G24
H23	H24
I23	I24
J23	J24
K23	K24
L23	L24
M23	M24
N23	N24
O23	O24
P23	P24
Q23	Q24
R23	R24
S23	S24
T23	T24

Select an option (1 to 11): 1

Legend: = Available seat = Booked seat

5.2 Customer booking seat A1

TicketingSystem Menu. What would you like to do?

1. View seating map
2. Book a seat
3. Join the waitlist
4. See the waitlist
5. Check if all tickets are sold
6. (Testing) Booking all seats
7. Make a cancellation
8. Modify reservation
9. Increase seating capacity
10. Check ticket details
11. Exit

Select an option (1 to 11): 2

Enter the letter corresponding to your seat row (A to T): A

Enter the number corresponding to your seat column (1-25): 1

Enter your name: Mary

Enter your email: mary@email.com

Seat A1 booked! Ticket ID is EDM25-6e88-A1

5.3 Visualising seat booked

Select an option (1 to 11): 1

A1	<input checked="" type="checkbox"/>	A2	<input type="checkbox"/>	A3	<input type="checkbox"/>	A4	<input type="checkbox"/>	A5	<input type="checkbox"/>	A6	<input type="checkbox"/>	A7	<input type="checkbox"/>	A8	<input type="checkbox"/>	A9	<input type="checkbox"/>	A10	<input type="checkbox"/>	A11	<input type="checkbox"/>	A12	<input type="checkbox"/>	A13	<input type="checkbox"/>	A14	<input type="checkbox"/>	A15	<input type="checkbox"/>	A16	<input type="checkbox"/>	A17	<input type="checkbox"/>	A18	<input type="checkbox"/>	A19	<input type="checkbox"/>	A20	<input type="checkbox"/>	A21	<input type="checkbox"/>	A22	<input type="checkbox"/>	A23	<input type="checkbox"/>	A24	<input type="checkbox"/>	A25	<input type="checkbox"/>
B1	<input type="checkbox"/>	B2	<input type="checkbox"/>	B3	<input type="checkbox"/>	B4	<input type="checkbox"/>	B5	<input type="checkbox"/>	B6	<input type="checkbox"/>	B7	<input type="checkbox"/>	B8	<input type="checkbox"/>	B9	<input type="checkbox"/>	B10	<input type="checkbox"/>	B11	<input type="checkbox"/>	B12	<input type="checkbox"/>	B13	<input type="checkbox"/>	B14	<input type="checkbox"/>	B15	<input type="checkbox"/>	B16	<input type="checkbox"/>	B17	<input type="checkbox"/>	B18	<input type="checkbox"/>	B19	<input type="checkbox"/>	B20	<input type="checkbox"/>	B21	<input type="checkbox"/>	B22	<input type="checkbox"/>	B23	<input type="checkbox"/>	B24	<input type="checkbox"/>	B25	<input type="checkbox"/>
C1	<input type="checkbox"/>	C2	<input type="checkbox"/>	C3	<input type="checkbox"/>	C4	<input type="checkbox"/>	C5	<input type="checkbox"/>	C6	<input type="checkbox"/>	C7	<input type="checkbox"/>	C8	<input type="checkbox"/>	C9	<input type="checkbox"/>	C10	<input type="checkbox"/>	C11	<input type="checkbox"/>	C12	<input type="checkbox"/>	C13	<input type="checkbox"/>	C14	<input type="checkbox"/>	C15	<input type="checkbox"/>	C16	<input type="checkbox"/>	C17	<input type="checkbox"/>	C18	<input type="checkbox"/>	C19	<input type="checkbox"/>	C20	<input type="checkbox"/>	C21	<input type="checkbox"/>	C22	<input type="checkbox"/>	C23	<input type="checkbox"/>	C24	<input type="checkbox"/>	C25	<input type="checkbox"/>
D1	<input type="checkbox"/>	D2	<input type="checkbox"/>	D3	<input type="checkbox"/>	D4	<input type="checkbox"/>	D5	<input type="checkbox"/>	D6	<input type="checkbox"/>	D7	<input type="checkbox"/>	D8	<input type="checkbox"/>	D9	<input type="checkbox"/>	D10	<input type="checkbox"/>	D11	<input type="checkbox"/>	D12	<input type="checkbox"/>	D13	<input type="checkbox"/>	D14	<input type="checkbox"/>	D15	<input type="checkbox"/>	D16	<input type="checkbox"/>	D17	<input type="checkbox"/>	D18	<input type="checkbox"/>	D19	<input type="checkbox"/>	D20	<input type="checkbox"/>	D21	<input type="checkbox"/>	D22	<input type="checkbox"/>	D23	<input type="checkbox"/>	D24	<input type="checkbox"/>	D25	<input type="checkbox"/>
E1	<input type="checkbox"/>	E2	<input type="checkbox"/>	E3	<input type="checkbox"/>	E4	<input type="checkbox"/>	E5	<input type="checkbox"/>	E6	<input type="checkbox"/>	E7	<input type="checkbox"/>	E8	<input type="checkbox"/>	E9	<input type="checkbox"/>	E10	<input type="checkbox"/>	E11	<input type="checkbox"/>	E12	<input type="checkbox"/>	E13	<input type="checkbox"/>	E14	<input type="checkbox"/>	E15	<input type="checkbox"/>	E16	<input type="checkbox"/>	E17	<input type="checkbox"/>	E18	<input type="checkbox"/>	E19	<input type="checkbox"/>	E20	<input type="checkbox"/>	E21	<input type="checkbox"/>	E22	<input type="checkbox"/>	E23	<input type="checkbox"/>	E24	<input type="checkbox"/>	E25	<input type="checkbox"/>
F1	<input type="checkbox"/>	F2	<input type="checkbox"/>	F3	<input type="checkbox"/>	F4	<input type="checkbox"/>	F5	<input type="checkbox"/>	F6	<input type="checkbox"/>	F7	<input type="checkbox"/>	F8	<input type="checkbox"/>	F9	<input type="checkbox"/>	F10	<input type="checkbox"/>	F11	<input type="checkbox"/>	F12	<input type="checkbox"/>	F13	<input type="checkbox"/>	F14	<input type="checkbox"/>	F15	<input type="checkbox"/>	F16	<input type="checkbox"/>	F17	<input type="checkbox"/>	F18	<input type="checkbox"/>	F19	<input type="checkbox"/>	F20	<input type="checkbox"/>	F21	<input type="checkbox"/>	F22	<input type="checkbox"/>	F23	<input type="checkbox"/>	F24	<input type="checkbox"/>	F25	<input type="checkbox"/>
G1	<input type="checkbox"/>	G2	<input type="checkbox"/>	G3	<input type="checkbox"/>	G4	<input type="checkbox"/>	G5	<input type="checkbox"/>	G6	<input type="checkbox"/>	G7	<input type="checkbox"/>	G8	<input type="checkbox"/>	G9	<input type="checkbox"/>	G10	<input type="checkbox"/>	G11	<input type="checkbox"/>	G12	<input type="checkbox"/>	G13	<input type="checkbox"/>	G14	<input type="checkbox"/>	G15	<input type="checkbox"/>	G16	<input type="checkbox"/>	G17	<input type="checkbox"/>	G18	<input type="checkbox"/>	G19	<input type="checkbox"/>	G20	<input type="checkbox"/>	G21	<input type="checkbox"/>	G22	<input type="checkbox"/>	G23	<input type="checkbox"/>	G24	<input type="checkbox"/>	G25	<input type="checkbox"/>
H1	<input type="checkbox"/>	H2	<input type="checkbox"/>	H3	<input type="checkbox"/>	H4	<input type="checkbox"/>	H5	<input type="checkbox"/>	H6	<input type="checkbox"/>	H7	<input type="checkbox"/>	H8	<input type="checkbox"/>	H9	<input type="checkbox"/>	H10	<input type="checkbox"/>	H11	<input type="checkbox"/>	H12	<input type="checkbox"/>	H13	<input type="checkbox"/>	H14	<input type="checkbox"/>	H15	<input type="checkbox"/>	H16	<input type="checkbox"/>	H17	<input type="checkbox"/>	H18	<input type="checkbox"/>	H19	<input type="checkbox"/>	H20	<input type="checkbox"/>	H21	<input type="checkbox"/>	H22	<input type="checkbox"/>	H23	<input type="checkbox"/>	H24	<input type="checkbox"/>	H25	<input type="checkbox"/>
I1	<input type="checkbox"/>	I2	<input type="checkbox"/>	I3	<input type="checkbox"/>	I4	<input type="checkbox"/>	I5	<input type="checkbox"/>	I6	<input type="checkbox"/>	I7	<input type="checkbox"/>	I8	<input type="checkbox"/>	I9	<input type="checkbox"/>	I10	<input type="checkbox"/>	I11	<input type="checkbox"/>	I12	<input type="checkbox"/>	I13	<input type="checkbox"/>	I14	<input type="checkbox"/>	I15	<input type="checkbox"/>	I16	<input type="checkbox"/>	I17	<input type="checkbox"/>	I18	<input type="checkbox"/>	I19	<input type="checkbox"/>	I20	<input type="checkbox"/>	I21	<input type="checkbox"/>	I22	<input type="checkbox"/>	I23	<input type="checkbox"/>	I24	<input type="checkbox"/>	I25	<input type="checkbox"/>
J1	<input type="checkbox"/>	J2	<input type="checkbox"/>	J3	<input type="checkbox"/>	J4	<input type="checkbox"/>	J5	<input type="checkbox"/>	J6	<input type="checkbox"/>	J7	<input type="checkbox"/>	J8	<input type="checkbox"/>	J9	<input type="checkbox"/>	J10	<input type="checkbox"/>	J11	<input type="checkbox"/>	J12	<input type="checkbox"/>	J13	<input type="checkbox"/>	J14	<input type="checkbox"/>	J15	<input type="checkbox"/>	J16	<input type="checkbox"/>	J17	<input type="checkbox"/>	J18	<input type="checkbox"/>	J19	<input type="checkbox"/>	J20	<input type="checkbox"/>	J21	<input type="checkbox"/>	J22	<input type="checkbox"/>	J23	<input type="checkbox"/>	J24	<input type="checkbox"/>	J25	<input type="checkbox"/>
K1	<input type="checkbox"/>	K2	<input type="checkbox"/>	K3	<input type="checkbox"/>	K4	<input type="checkbox"/>	K5	<input type="checkbox"/>	K6	<input type="checkbox"/>	K7	<input type="checkbox"/>	K8	<input type="checkbox"/>	K9	<input type="checkbox"/>	K10	<input type="checkbox"/>	K11	<input type="checkbox"/>	K12	<input type="checkbox"/>	K13	<input type="checkbox"/>	K14	<input type="checkbox"/>	K15	<input type="checkbox"/>	K16	<input type="checkbox"/>	K17	<input type="checkbox"/>	K18	<input type="checkbox"/>	K19	<input type="checkbox"/>	K20	<input type="checkbox"/>	K21	<input type="checkbox"/>	K22	<input type="checkbox"/>	K23	<input type="checkbox"/>	K24	<input type="checkbox"/>	K25	<input type="checkbox"/>
L1	<input type="checkbox"/>	L2	<input type="checkbox"/>	L3	<input type="checkbox"/>	L4	<input type="checkbox"/>	L5	<input type="checkbox"/>	L6	<input type="checkbox"/>	L7	<input type="checkbox"/>	L8	<input type="checkbox"/>	L9	<input type="checkbox"/>	L10	<input type="checkbox"/>	L11	<input type="checkbox"/>	L12	<input type="checkbox"/>	L13	<input type="checkbox"/>	L14	<input type="checkbox"/>	L15	<input type="checkbox"/>	L16	<input type="checkbox"/>	L17	<input type="checkbox"/>	L18	<input type="checkbox"/>	L19	<input type="checkbox"/>	L20	<input type="checkbox"/>	L21	<input type="checkbox"/>	L22	<input type="checkbox"/>	L23	<input type="checkbox"/>	L24	<input type="checkbox"/>	L25	<input type="checkbox"/>
M1	<input type="checkbox"/>	M2	<input type="checkbox"/>	M3	<input type="checkbox"/>	M4	<input type="checkbox"/>	M5	<input type="checkbox"/>	M6	<input type="checkbox"/>	M7	<input type="checkbox"/>	M8	<input type="checkbox"/>	M9	<input type="checkbox"/>	M10	<input type="checkbox"/>	M11	<input type="checkbox"/>	M12	<input type="checkbox"/>	M13	<input type="checkbox"/>	M14	<input type="checkbox"/>	M15	<input type="checkbox"/>	M16	<input type="checkbox"/>	M17	<input type="checkbox"/>	M18	<input type="checkbox"/>	M19	<input type="checkbox"/>	M20	<input type="checkbox"/>	M21	<input type="checkbox"/>	M22	<input type="checkbox"/>	M23	<input type="checkbox"/>	M24	<input type="checkbox"/>	M25	<input type="checkbox"/>
N1	<input type="checkbox"/>	N2	<input type="checkbox"/>	N3	<input type="checkbox"/>	N4	<input type="checkbox"/>	N5	<input type="checkbox"/>	N6	<input type="checkbox"/>	N7	<input type="checkbox"/>	N8	<input type="checkbox"/>	N9	<input type="checkbox"/>	N10	<input type="checkbox"/>	N11	<input type="checkbox"/>	N12	<input type="checkbox"/>	N13	<input type="checkbox"/>	N14	<input type="checkbox"/>	N15	<input type="checkbox"/>	N16	<input type="checkbox"/>	N17	<input type="checkbox"/>	N18	<input type="checkbox"/>	N19	<input type="checkbox"/>	N20	<input type="checkbox"/>	N21	<input type="checkbox"/>	N22	<input type="checkbox"/>	N23	<input type="checkbox"/>	N24	<input type="checkbox"/>	N25	<input type="checkbox"/>
O1	<input type="checkbox"/>	O2	<input type="checkbox"/>	O3	<input type="checkbox"/>	O4	<input type="checkbox"/>	O5	<input type="checkbox"/>	O6	<input type="checkbox"/>	O7	<input type="checkbox"/>	O8	<input type="checkbox"/>	O9	<input type="checkbox"/>	O10	<input type="checkbox"/>	O11	<input type="checkbox"/>	O12	<input type="checkbox"/>	O13	<input type="checkbox"/>	O14	<input type="checkbox"/>	O15	<input type="checkbox"/>	O16	<input type="checkbox"/>	O17	<input type="checkbox"/>	O18	<input type="checkbox"/>	O19	<input type="checkbox"/>	O20	<input type="checkbox"/>	O21	<input type="checkbox"/>	O22	<input type="checkbox"/>	O23	<input type="checkbox"/>	O24	<input type="checkbox"/>	O25	<input type="checkbox"/>
P1	<input type="checkbox"/>	P2	<input type="checkbox"/>	P3	<input type="checkbox"/>	P4	<input type="checkbox"/>	P5	<input type="checkbox"/>	P6	<input type="checkbox"/>	P7	<input type="checkbox"/>	P8	<input type="checkbox"/>	P9	<input type="checkbox"/>	P10	<input type="checkbox"/>	P11	<input type="checkbox"/>	P12	<input type="checkbox"/>	P13	<input type="checkbox"/>	P14	<input type="checkbox"/>	P15	<input type="checkbox"/>	P16	<input type="checkbox"/>	P17	<input type="checkbox"/>	P18	<input type="checkbox"/>	P19	<input type="checkbox"/>	P20	<input type="checkbox"/>	P21	<input type="checkbox"/>	P22	<input type="checkbox"/>	P23	<input type="checkbox"/>	P24	<input type="checkbox"/>	P25	<input type="checkbox"/>
Q1	<input type="checkbox"/>	Q2	<input type="checkbox"/>	Q3	<input type="checkbox"/>	Q4	<input type="checkbox"/>	Q5	<input type="checkbox"/>	Q6	<input type="checkbox"/>	Q7	<input type="checkbox"/>	Q8	<input type="checkbox"/>	Q9	<input type="checkbox"/>	Q10	<input type="checkbox"/>	Q11	<input type="checkbox"/>	Q12	<input type="checkbox"/>	Q13	<input type="checkbox"/>	Q14	<input type="checkbox"/>	Q15	<input type="checkbox"/>	Q16	<input type="checkbox"/>	Q17	<input type="checkbox"/>	Q18	<input type="checkbox"/>	Q19	<input type="checkbox"/>	Q20	<input type="checkbox"/>	Q21	<input type="checkbox"/>	Q22	<input type="checkbox"/>	Q23	<input type="checkbox"/>	Q24	<input type="checkbox"/>	Q25	<input type="checkbox"/>
R1	<input type="checkbox"/>	R2	<input type="checkbox"/>	R3	<input type="checkbox"/>	R4	<input type="checkbox"/>	R5	<input type="checkbox"/>	R6	<input type="checkbox"/>	R7	<input type="checkbox"/>	R8	<input type="checkbox"/>	R9	<input type="checkbox"/>	R10	<input type="checkbox"/>	R11	<input type="checkbox"/>	R12	<input type="checkbox"/>	R13	<input type="checkbox"/>	R14	<input type="checkbox"/>	R15	<input type="checkbox"/>	R16	<input type="checkbox"/>	R17	<input type="checkbox"/>	R18	<input type="checkbox"/>	R19	<input type="checkbox"/>	R20	<input type="checkbox"/>	R21	<input type="checkbox"/>	R22	<input type="checkbox"/>	R23	<input type="checkbox"/>	R24	<input type="checkbox"/>	R25	<input type="checkbox"/>
S1	<input type="checkbox"/>	S2	<input type="checkbox"/>	S3	<input type="checkbox"/>	S4	<input type="checkbox"/>	S5	<input type="checkbox"/>	S6	<input type="checkbox"/>	S7	<input type="checkbox"/>	S8	<input type="checkbox"/>	S9	<input type="checkbox"/>	S10	<input type="checkbox"/>	S11	<input type="checkbox"/>	S12	<input type="checkbox"/>	S13	<input type="checkbox"/>	S14	<input type="checkbox"/>	S15	<input type="checkbox"/>	S16	<input type="checkbox"/>	S17	<input type="checkbox"/>	S18	<input type="checkbox"/>	S19	<input type="checkbox"/>	S20	<input type="checkbox"/>	S21	<input type="checkbox"/>	S22	<input type="checkbox"/>	S23	<input type="checkbox"/>	S24	<input type="checkbox"/>	S25	<input type="checkbox"/>
T1	<input type="checkbox"/>	T2	<input type="checkbox"/>	T3	<input type="checkbox"/>	T4	<input type="checkbox"/>	T5	<input type="checkbox"/>	T6	<input type="checkbox"/>	T7	<input type="checkbox"/>	T8	<input type="checkbox"/>	T9	<input type="checkbox"/>	T10	<input type="checkbox"/>	T11	<input type="checkbox"/>	T12	<input type="checkbox"/>	T13	<input type="checkbox"/>	T14	<input type="checkbox"/>	T15	<input type="checkbox"/>	T16	<input type="checkbox"/>	T17	<input type="checkbox"/>	T18	<input type="checkbox"/>	T19	<input type="checkbox"/>	T20	<input type="checkbox"/>	T21	<input type="checkbox"/>	T22	<input type="checkbox"/>	T23	<input type="checkbox"/>	T24	<input type="checkbox"/>	T25	<input type="checkbox"/>

5.3 Modifying reservation (seat and priority)

TicketingSystem Menu. What would you like to do?

1. View seating map
2. Book a seat
3. Join the waitlist
4. See the waitlist
5. Check if all tickets are sold
6. (Testing) Booking all seats
7. Make a cancellation
8. Modify reservation
9. Increase seating capacity
10. Check ticket details
11. Exit

Select an option (1 to 11): 8

Enter your Ticket id: EDM25-69ae-a1

Would you like to change your seat? (Yes or No) yes

Enter the letter corresponding to your new seat row (A to T): a

Enter the number corresponding to your new seat column (1-25): 5

Would you like to change the priority? (Yes or No) yes

Enter new priority (1, 2 or 3): 2

Seat updated from a1 to a5.

User priority updated to 2

6. User friendly interface

To facilitate for test cases and practice more, a menu interface was created/ It runs in a loop and allows for choosing between a range of actions to be done in the system.

```
▶ run_ticketing_system()  
***  
TicketingSystem Menu. What would you like to do?  
1. View seating map  
2. Book a seat  
3. Join the waitlist  
4. See the waitlist  
5. Check if all tickets are sold  
6. (Testing) Booking all seats  
7. Make a cancellation  
8. Modify reservation  
9. Increase seating capacity  
10. Check ticket details  
11. Exit  
Select an option (1 to 11): [ ]
```

6.1 Menu command

```
▶ run_ticketing_system()
```

This function starts the interactive loop for users to perform different actions

7. Use of Gen AI

Following are the screenshots of the interaction with OpenAi ChatGPT

ChatGPT 4.0 ▾

Share

I am implementing a ticketing system, and in this part we are asked to make use of your capabilities to improve the system. I will paste my code here. Please analyse it and let me know about any possible improvements. Do not make any improvements before discussing it with me.

Some of the improvements suggested by the instructor were group booking, student discount, senior discount, refund tickets. Let me know what you think of them or if I would have better gain by working on other improvements first. Keep into consideration I would be interested in the group booking, or the same person booking a couple of seats under the same user id.

Thanks for sharing your code, Paula! You've done a really solid job structuring it. It's clear, modular, and your comments make it easy to follow. Now let's dive into the analysis and feature suggestions based on your interest and what could bring the most value to users.

✓ Your Priority: Group Booking

You mentioned you're especially interested in group booking or allowing the same person to book multiple seats under the same user ID. That's a **great idea to tackle next**. Here's why:

- It's a **natural extension** of your current logic without being overly complex.
- You already have the seat booking logic well modularized, so adapting it for multiple seats is straightforward.
- It would provide a **strong UX improvement**, especially for families, friends, or colleagues booking together.
- You can easily tie multiple seats to a **single ticket group ID or user ID** for reference.

💡 Analysis of Other Suggested Features

Feature	Description	Implementation Effort	User Value	Notes
Student Discount	Apply discount based on user type	Low to Medium	Medium	Needs UI input for user type and a field for price or discount
Senior Discount	Same as above	Low to Medium	Medium	Similar structure as student discount
Refund Tickets	Allow users to request refunds after booking	Medium	High	You've already got a <code>cancel_booking()</code> function, so refund logic could piggyback on that
Group Booking	Book multiple seats in one go	Medium	High	Offers strong UX value and is aligned with your current interests



Recommendations

Given your goals and the current state of your code, here's a **suggested implementation priority**:

1. **Group Booking** – You're interested, it fits naturally with your logic, and improves UX.
 2. **Refunds** – You've got cancellation logic already, just add a refund confirmation flow (could even simulate refund logic for now).
 3. **Discounts (Student/Senior)** – More related to pricing, which isn't heavily handled yet in your system. Might make more sense after pricing logic is added.
-

Initial Questions for You

To move forward with group booking, could you clarify a few things?

1. Should all seats in the group booking **belong to the same user ID**, or would each seat still need its own individual ticket ID?
2. Would you want to **automatically assign adjacent seats** if available, or let the user pick multiple manually?
3. Should you allow **booking failure if not all selected seats are available**, or partial bookings with a warning?

Once I know what you'd prefer, we can brainstorm the best approach together.



Great, answering your questions, 1. yes, they will belong to the same user ID, and I would like a limitation of 5 tickets per user. They can book their tickets in different moments, for instance, booking only one ticket at first, and then booking another one. So I might need some sort of identification before booking a ticket.
2. yes, first ask how many tickets user wants to purchase, and if they are booking more than one ticket at once, suggest adjacent seats, but they would have to confirm their preferences or change specific ones if they want.
3. partial bookings with a warning, or asking for the user to select other ones before continuing, would be a good option.

Perfect, Paula! That gives me a clear direction for what you need. Here's a breakdown of how we can structure the **Group Booking** upgrade with your preferences in mind:

🔧 Key Adjustments to Make

1. Track Bookings by User ID

- You'll need to store user IDs and their associated tickets.
- Introduce a `user_registry` dictionary:

```
python
```

Copy

Edit

```
self.user_registry = {} # key: user_id, value: List of ticket_ids
```



2. Identify Returning Users

2. Identify Returning Users

- Before booking, ask:
"Are you a new user or returning user?"
 - New → generate new UUID.
 - Returning → prompt for existing `user_id` (show email or ticket lookup helper).

3. Enforce Ticket Limit

- Before booking, check if the user already has tickets:

```
python
```

Copy

Edit

```
if len(self.user_registry[user_id]) >= 5:  
    print("You cannot book more than 5 tickets.")
```

4. Multi-Seat Booking Logic

- If a user books more than one seat at once:
 - Ask: “Do you want adjacent seats?”
 - Find and suggest adjacent seats if available.
 - Let them confirm or change the proposed seats before booking.
- If seats are not available:
 - Offer alternative groupings or continue with a warning (e.g., “Only 3 of the 5 selected seats are available. Proceed or pick others?”).

5. Update `booking_seat()` or Add `booking_multiple_seats()`

- You can either:
 - Modify your existing function to handle both single/multiple seats, or
 - Create a new method specifically for multi-seat bookings to keep things clean.



💡 Optional Enhancements (Future Ideas)

- Provide a lookup by `email` or `name` for returning users to avoid pasting UUIDs.
- Implement a `temporary hold` system for selected seats (e.g., “held for 5 mins”) to prevent race conditions if extended online.
- Print a summary: “*You booked 3 seats: A5, A6, A7.*”

Would you like help drafting the structure for:

- `booking_multiple_seats()` and the adjacent seat suggestion logic?
- Or should we start by modifying the user tracking and ticket limit first?

Let me know which step you’d like to tackle next, and I’ll guide you through!



Provide me only with the booking by group method implementation at first, together with the user identification and everything related to that discussed.

Absolutely! Here's the **group booking method** implementation along with the necessary **user identification logic and data structure setup**. This includes:

- A way to **track users and their booked tickets**
- A method to **suggest adjacent seats**
- A `group_booking` method that:
 - Identifies the user (new or returning)
 - Enforces a **5-ticket limit per user**
 - Suggests adjacent seats if booking multiple tickets
 - Falls back to manual seat selection if needed



Add this to your `__init__` method:

python

Copy Edit

```
self.user_registry = {} # key: user_id, value: List of ticket_ids
```

Suggest adjacent seats method:

python

Copy Edit

```
def suggest_adjacent_seats(self, num_tickets):  
    for r in range(self.rows):  
        for c in range(self.columns - num_tickets + 1):  
            seats = [f"{string.ascii_uppercase[r}]{c + i + 1}" for i in range(num_tickets)]  
            if all(self.seat_availability(seat) for seat in seats):  
                return seats  
    return []
```

Group booking method:

```
python
```

```
def group_booking(self, name, email, num_tickets, user_id=None, priority=3):
    """
    Allows group booking with user ID tracking, 5-ticket limit per user,
    adjacent seat suggestion, and fallback to manual selection.
    """

    # User identification
    if user_id is None:
        user_id = str(uuid.uuid4())
        print(f"New user created. Your User ID is: {user_id}")
    elif user_id not in self.user_registry:
        self.user_registry[user_id] = []

    # Check user ticket limit
    if len(self.user_registry.get(user_id, [])) + num_tickets > 5:
```

Copy Edit

Full code in Google colab

Let me know when you're ready and I can show you how to:

- Integrate this with your main menu
- Add helper functions for user ID lookup
- Or continue with refund/discounts!

Would you like the `booking_seat` method updated as well to better handle user ID registration inside this flow?

Yes, I am ready.

And yes, please update everything inside this part to adapt to the changes:

```
import uuid #to give unique user and ticket id
import string #to ↓ ascii uppercase to be uses in seat codes
```

I asked it to adjust everything inside my first cell in colab. (Just so I won't touch my code and mix with it, everything done by AI, will be under the AI part in colab only.)

```
</> Ticketing System Group Booking

1 import uuid #to give unique user and ticket id
2 import string #to get ascii uppercase to be used in seat codes
3
4 v class TicketingSystem:
5 v   def __init__(self, event_code, rows=20, columns=25):
6     self.event_code = event_code
7     self.rows = rows
8     self.columns = columns
9     self.grid = self.seat_grid_create()
10    self.ticket_info = {}
11    self.user_registry = {} # user_id: list of ticket_ids
12    self.waitlist = WaitlistQueue()
13
14 v   def seat_grid_create(self):
```

Your `TicketingSystem` class has now been updated with:

- `user_registry` to track user bookings by user ID
- An enhanced `booking_seat()` method that supports user re-identification
- A new `group_booking()` method with adjacent seat suggestions and manual fallback

Everything is structured to enforce the 5-ticket limit per user and allow bookings across sessions. Let me know when you're ready to update the interactive menu or continue with features like refunds or discounts!



I would like you to provide me with test cases to experiment with these, so I can test them

Great choice! Here are **simple and clear test cases** you can run directly after creating an instance of your `TicketingSystem`. These help you test:

1. A new group booking
2. Reusing the same user ID for additional bookings
3. Enforcing the 5-ticket limit
4. Booking suggested adjacent seats
5. Falling back to manual seat entry

1. Create system and make a group booking for 3 seats (new user)

```
python
```

 Copy  Edit

```
system = TicketingSystem("EDM25")

# Booking 3 seats (will auto-generate a user_id and suggest adjacent seats)
system.group_booking(name="Alice", email="alice@example.com", num_tickets=3)
```

When asked:

- If suggested adjacent seats are shown, type `YES` to test auto-booking.
 - If you type `NO`, it'll ask you to enter 3 seat codes manually.
-

2. Book 2 more seats under the same user ID (revisit)

Get Alice's user ID from the output in step 1, then use it here:

```
python
```

 Copy  Edit

```
# Replace 'your-user-id-here' with the real ID printed in step 1
system.group_booking(name="Alice", email="alice@example.com", num_tickets=2, user_id="your-user-id")
```

This brings her total to 5 tickets.

3. Try to exceed 5-ticket limit

```
python
```

 Copy  Edit

```
# This should be rejected
system.group_booking(name="Alice", email="alice@example.com", num_tickets=1, user_id="your-user-id")
```

Expected: "You cannot book more than 5 tickets per user."

⌚ 4. Test adjacent seat suggestion directly

You can call this just to check what would be suggested:

```
python
```

Copy Edit

```
print(system.suggest_adjacent_seats(3))
```

If you want to manually book those:

```
python
```

Copy Edit

```
adjacent = system.suggest_adjacent_seats(3)
if adjacent:
    for seat in adjacent:
        system.booking_seat(seat, name="Bob", email="bob@example.com")
```

🛠 5. Test manual selection fallback

Start a booking but type `no` when prompted about adjacent seats.

```
python
```

Copy Edit

```
system.group_booking(name="Charlie", email="charlie@example.com", num_tickets=2)
```

When prompted:

- Suggested seats → type `no`
- Then manually enter 2 seat codes (make sure they're available like "A20", "A21").

Let me know if you'd like help wrapping these into a proper `unittest` test suite or automate inputs using mocks for learning or grading purposes!

Explain to me what a unittest is and how beneficial it would be for me

Great question, Paula!

What is `unittest`?

`unittest` is Python's built-in testing framework that allows you to **automate the testing of your code**. Instead of manually typing inputs and checking the results every time, you can write test scripts that run automatically and verify if the code behaves as expected.

Think of it as a **safety net** — like setting up sensors that beep if something goes wrong in your program after changes or improvements.

Why is `unittest` beneficial for you?

Here's why it's especially useful in your case:

1. Confidence in your changes

When you add new features like group booking, user IDs, refunds, discounts — you can run your tests and make sure nothing else broke.

2. Save time

Instead of running your full app and manually checking edge cases, tests can run in milliseconds.

3. Catch bugs early

Say you change something in `booking_seat()` and accidentally break how `group_booking()` works. A test will fail and tell you exactly where the issue is.

4. Maintain complex systems

As your system grows, it becomes harder to remember all the rules (like max 5 tickets per user). `unittest` helps you keep track of what should happen in each scenario.

5. Show professionalism

If you ever share this project (e.g., portfolio, GitHub, a job), having tests in place shows your maturity as a developer — especially in areas involving logic like ticketing or payments.

