Vivekanand Education Society's Institute Of Technology
Department Of Information Technology
DSA mini Project
A.Y. 2025-26

Title: Airport Check-in Counter Simulation

Sustainability Goal : Reduced Inequalities

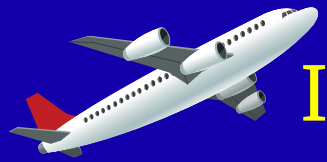
Domain:

Member: Shravani Mankar

Mentor Name: Kajal Jewani

Content

1. **Introduction to the Project**
2. **Problem Statement**
3. **Objectives of the Project**
4. **Requirements of the System (Hardware, Software)**
5. **Implementation**
6. **Conclusion**
7. **References (in IEEE Format)**



Introduction to Project

Introduction to Airport Check-in Simulation-

Airports handle thousands of passengers every day, and one of the most critical operations is the **check-in process**. This involves passengers arriving, waiting in queues, verifying their tickets, dropping off baggage, and finally receiving their boarding passes.

Why Simulation?

A simulation allows us to **model real-world systems** like an airport check-in counter in a controlled environment. By simulating passenger arrivals, queues, and processing, we can study how efficiently the system works and identify bottlenecks.

Problem Statement

Airports experience a high volume of passengers daily, making the check-in process one of the most critical and time-consuming tasks.
Passengers must queue at different counters, verify their identity, submit baggage, and collect their boarding passes. However, delays often occur due to **long queues, uneven counter allocation, and priority passengers** needing faster service.
In order to optimize this process, there is a need to design a **computer-based simulation of airport check-in** using concepts of **Data Structures and Algorithms (DSA)**. This simulation will model passengers arriving, joining queues, being served at counters, and leaving the system.

Objectives of the project

- To simulate the airport check-in process
- To implement core Data Structures
- To analyze system efficiency
- To demonstrate real-world applicability of DSA
- To provide a flexible simulation

DATA STRUCTURES

- Priority Queue
- Singly Linked List
- Structures (struct)
- Global Pointers (front and rear)
- Counter Variable (nextId)

Requirements of the system (Hardware, software)

Software

Programming Language:C
IDE:IntelliJ IDEA
Compiler / Runtime-program wiz online C compiler
Version Control :Git / GitHub
Simulation Output:Console-based for queue and passenger processing

Hardware

Processor: Intel i3
RAM: Minimum 4 GB
Storage: At least 1 GB free space
Display: Standard monitor
Input Devices: Keyboard and mouse
Operating System Support: Windows / macOS

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct {
    int id;
    char name[20];
    int priority;
} Passenger;
typedef struct Node {
    Passenger p;
    struct Node* next;} Node;
Node* front = NULL;
int nextId = 1;
int isEmpty() {
    return front == NULL;}
void enqueue(char name[], int priority) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        return; }
    newNode->p.id = nextId++;
    strcpy(newNode->p.name, name);
    newNode->p.priority = priority;
    newNode->next = NULL;
    if (isEmpty() || priority > front->p.priority) {
        newNode->next = front;
        front = newNode;}
        else {
        Node* temp = front;
        while (temp->next != NULL && temp->next->p.priority >= priority) {temp = temp
            ->next; }
        newNode->next = temp->next;
        temp->next = newNode;}
    printf("Passenger %s (ID: %d, Priority: %d) checked in.\n",
           name, newNode->p.id, newNode->p.priority);}
```

```c
void dequeue() {
    if (isEmpty()) {
        printf("No passengers in queue!\n");
        return;}
    Node* temp = front;
    printf("Passenger %s (ID: %d, Priority: %d) is being served.\n",
           temp->p.name, temp->p.id, temp->p.priority);
    front = front->next;
    free(temp);}
void displayQueue() {
    if (isEmpty()) {
        printf("Queue is empty!\n");
        return;}
    printf("Current Queue (Highest priority first):\n");
    Node* temp = front;
    while (temp != NULL) {
        printf("ID: %d, Name: %s, Priority: %d\n",
               temp->p.id, temp->p.name, temp->p.priority);
        temp = temp->next;}}
int main() {
    int choice, priority;
    char name[20];

    do {
        printf("\n--- Airport Check-in (Priority Queue) ---\n");
        printf("1. Add Passenger\n");
        printf("2. Serve Passenger\n");
        printf("3. Display Queue\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```c
    switch (choice) {
        case 1:
            printf("Enter passenger name: ");
            scanf("%s", name);
            printf("Enter priority
            (1=Low, 2=Medium, 3=High): ");
            scanf("%d", &priority);
            enqueue(name, priority);
            break;
        case 2:
            dequeue();
            break;
        case 3:
            displayQueue();
            break;
        case 4:
            printf("Exiting program...\n");
            break;
        default:
            printf("Invalid choice! Try again.\n");
    }
} while (choice != 4);

return 0;}
```

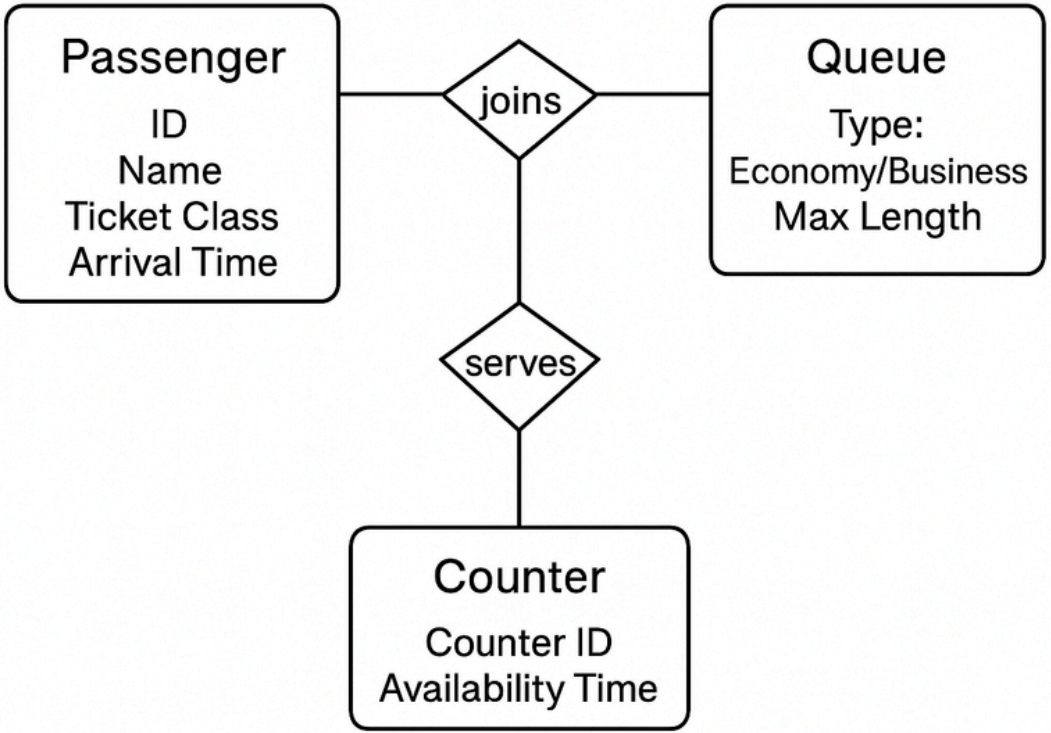
OUTPUT

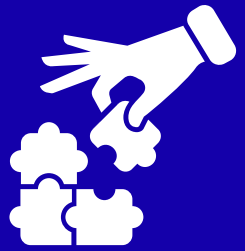
```
--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 1
Enter passenger name: Shravani
Enter priority (1=Low, 2=Medium, 3=High): 2
Passenger Shravani (ID: 1, Priority: 2) checked in.

--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 1
Enter passenger name: Leena
Enter priority (1=Low, 2=Medium, 3=High): 3
Passenger Leena (ID: 2, Priority: 3) checked in.
```

```
--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 1
Enter passenger name: Soham
Enter priority (1=Low, 2=Medium, 3=High): 1
Passenger Soham (ID: 3, Priority: 1) checked in.

--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 1
Enter passenger name: Jeet
Enter priority (1=Low, 2=Medium, 3=High): 2
Passenger Jeet (ID: 4, Priority: 2) checked in.
```

```
--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 2
Passenger Leena (ID: 2, Priority: 3) is being served.

--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 3
Current Queue (Highest priority first):
ID: 1, Name: Shravani, Priority: 2
ID: 4, Name: Jeet, Priority: 2
ID: 3, Name: Soham, Priority: 1

--- Airport Check-in (Priority Queue) ---
1. Add Passenger
2. Serve Passenger
3. Display Queue
4. Exit
Enter your choice: 4
Exiting program...


=== Code Execution Successful ===
```

ER diagram of the proposed system





Conclusion

The Airport Check-in Simulation demonstrates how queues can effectively manage passenger flow in a real-world scenario. Using C language and linked list–based queue implementation, the project shows how passengers arrive, wait, and are served in a fair and systematic manner. This project highlights the practical use of data structures in solving everyday problems, while keeping the model simple and extendable for future improvements.

References

- GeeksforGeeks, Queue Data Structure — https://www.geeksforgeeks.org/queue-data-structure
- TutorialsPoint, C Programming Language Tutorial — https://www.tutorialspoint.com/cprogramming
- Programiz, Data Structures and Algorithms in C — https://www.programiz.com/dsa