M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

A Project Report

on

# AIRPORT CHECK IN-DESK STIMULATOR

Submitted in partial fulfilment of requirements for the award of the course

of

**CGA1121 – DATA STRUCTURES**

Under the guidance of

**Mrs. K. MAKANYADEVI M.E.,**

**Assistant Professor/CSE**

Submitted By

**RASMITHA.R** **(927623BIT095)**

**DEPARTMENT OF FRESHMAN ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**
(Autonomous)

**KARUR – 639 113**

MAY 2024

M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

**M. KUMARASAMY COLLEGE OF ENGINEERING**

**(Autonomous Institution affiliated to Anna University, Chennai)**

**KARUR – 639 113**

- **BONAFIDE CERTIFICATE**

Certified that this project report on **"AIRPORT CHECK IN-DESK STIMULATOR"** is the bonafide work of **RASMITHA.R (927623BIT095)** who carried out the project work during the academic year 2023- 2024 under my supervision.

Signature

**Mrs. P. KAYALVIZHI M.E.,**

**SUPERVISOR,**

Department of Computer Science and Engineering,

M. Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Signature

**Dr. K.CHITIRAKALA, M.Sc., M.Phil.,Ph.D.,**

**HEAD OF THE DEPARTMENT,**

Department of Freshman Engineering,

M. Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

## DEPARTMENT OF INFORMATION TECHNOLOGY

### VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

### MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

### VISION OF THE DEPARTMENT

To create groomed, technically competent and skilled intellectual IT professionals to meet the current challenges of the modern computing industry.

### MISSION OF THE DEPARTMENT

- To ensure the understanding of fundamental aspects of Information Technology.
- Prepare students to adapt to the challenges of changing market needs by providing an environment.
- Build necessary skills required for employ ability through career development training to meet the challenges posed by the competitive world.

### PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

**PEO 1:** Graduates will be able to solve real world problems using learned concepts pertaining to Information Technology domain.

**PEO 2:** Encompass the ability to examine, plan and build innovative software Products and become a successful entrepreneur.

**PEO 3:** Graduates will be able to carry out the profession with ethics, integrity, leadership and social responsibility.

**PEO 4:** Graduates will be able to pursue post-graduation and succeed in academic and research careers.

## PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

1. **PSO1: Professional Skills:** Comprehend the technological advancement and practice professional ethics and the concerns for societal and environmental well-being.
2. **PSO 2: Competency Skills:** Design software in a futuristic approach to support current technology and adapt cutting-edge technologies.
3. **PSO 3: Successful career:** Apply knowledge of theoretical computer science to assess the hardware and software aspects of computer systems.

# ABSTRACT


- This project presents an Airport Check-In Desk Simulator developed using the C programming language. The simulator aims to replicate the operations of an airport check-in desk, allowing users to understand and analyze the efficiency of different check-in desk configurations and strategies. The program simulates various scenarios such as passenger arrivals, check-in procedures, luggage handling, and boarding processes. Through the implementation of queue management, time-based events, and statistical analysis, the simulator provides valuable insights into optimizing check-in desk operations to enhance passenger experience and minimize wait times. This abstract outlines the objectives, methodology, and key features of the Airport Check-In Desk Simulator, offering a comprehensive overview of its functionality and potential applications.

# ABSTRACT WITH POs AND PSOs MAPPING

| ABSTRACT | POs MAPPED | PSOs MAPPED |
|---|---|---|
| This project introduces a comprehensive phone directory application designed for efficient contact management, implemented using a doubly linked list data structure. The application enables users to seamlessly add, update, and delete contacts, providing a dynamic and user-friendly interface. Leveraging the advantages of a doubly linked list, the system ensures swift insertion, deletion, and bidirectional traversal, accommodating the evolving nature of contact lists. Exception handling mechanisms enhance the robustness of the application, addressing unforeseen errors and ensuring stability. The user interface is intuitively designed, allowing users to display, search, edit, and delete contacts effortlessly. The project showcases technical proficiency in data structure implementation, emphasizing the significance of user-centric design and error management. | **PO1(2)**<br>**PO2(3)**<br>**PO3(2)**<br>**PO4(2)**<br>**PO5(3)**<br>**PO6(1)**<br>**PO7(3)**<br>**PO8(2)**<br>**PO9(3)**<br>**PO10(3)**<br>**PO11(2)**<br>**PO12(2)** | **PSO1(3)**<br>**PSO2(2)**<br>**PSO3(2)** |

Note: 1- Low, 2-Medium, 3- High

**SUPERVISOR** **HEAD OF THE DEPARTMENT**

# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

**The Airport Check-In Desk Simulator project is a software endeavor aimed at replicating and analyzing the operations of an airport check-in desk using the C programming language and the Queue (Q) data structure. Airports are bustling hubs of activity, where efficient check-in processes are crucial for ensuring smooth operations and passenger satisfaction. By simulating various scenarios and configurations, this project provides a platform for understanding, optimizing, and improving check-in desk operations.**

## 1.2 **Objective**

Check-in is one among those services passenger received at airport where passenger is provided with a boarding pass, luggage tag for his or her checked luggage.

In check-in process, passenger has to check his or her tickets for the confirmation of travelling to his or her destination. To get this service, passengers often have to stand in queues in front of a traditional check-in counter during the check-in period.

## 1.3 Data Structure Choice

- ✓ **Queue:** A queue data structure can be used to manage the line of passengers waiting for check-in. Each passenger can be represented as an element in the queue, and the queue follows the First-In-First-Out (FIFO) principle.

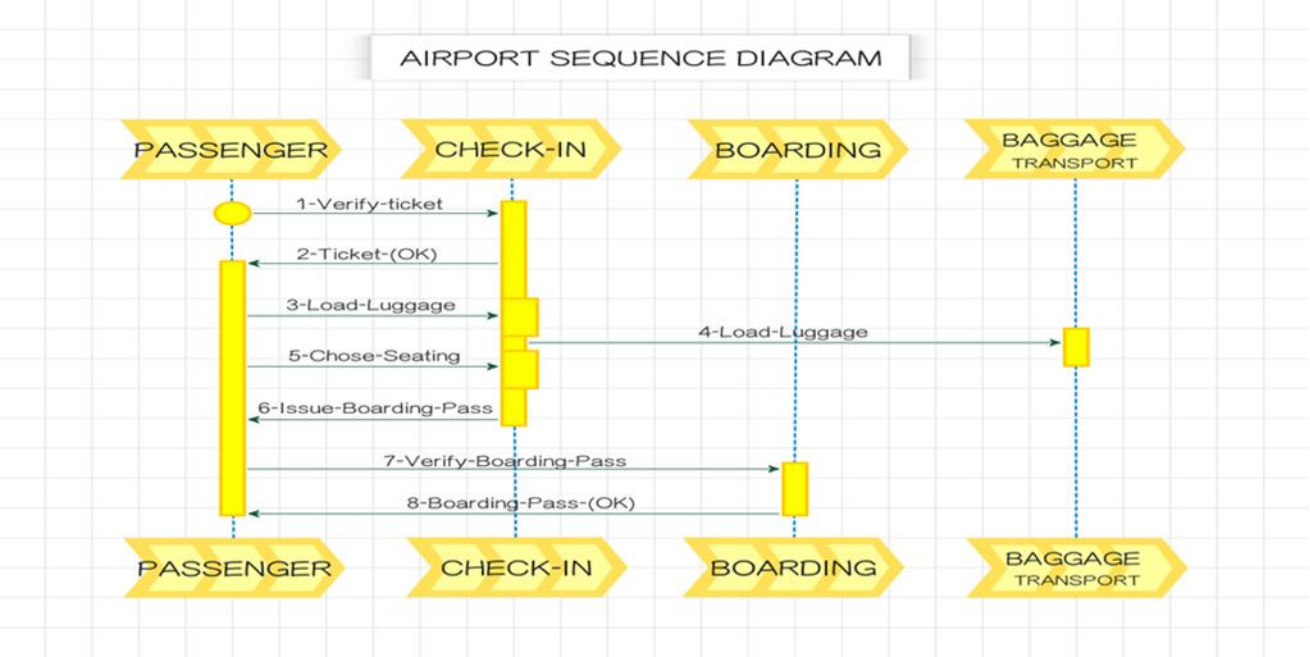
- ✓ **Graph:** A graph data structure can be employed to model flight connections and routes. Nodes represent airports, and edges represent flights between airports.
- ✓ **Stack:** A stack data structure can be used for managing temporary data during check-in processes. For example, a stack can be used to store luggage boarding passes temporarily before final processing.

# CHAPTER 2
# PROJECT METHODOLOGY

## 2.1 Methodology

Objects represent passenger move in a process, elements represent check-in desk in process. There are five types of element in this paper consist of entrances: this is where passengers enter a process, buffers: this is where passenger can stand in line at check – in desk, work stations: this is where check-in desk is performed on passenger, decision points: this is where a passenger goes in one of two directions, exits: this is where objects leave a process. The third building block, statistical distributions, is discussed in the next section. When a SimQuick simulation begins, a “simulation clock” starts in the computer and runs for the designated duration of the simulation. While this clock is running, a series of events sequentially takes place. There are three types of events in SimQuick: the arrival of passenger at an entrance, the departure of passenger from an exit, and the finish of staff on passenger at a work station. Whenever an event occurs, SimQuick moves objects from element to element as much as possible.

## 2.2 Block Diagram



# CHAPTER 3

# MODULES

## 3.1 PASSENGER ARRIVAL:

Passenger arrival varies during different hours of the day and during different days. Maximum numbers of customers are seen at the Airport between 08:00 am to 09:00 pm from Monday to Sunday, ie. passengers arraival is to be verified

3.1 AIRPORT PASSENGER CHECK-IN:

It is the first thing which airline passenger must do on reporting for his flight at any report or town terminal is to check in that is register his baggage, and exchange his flight for a boarding pass. As at any moment, passengers may be checking in for several different flights, a number of manned service positions must be provided. The study describes an operational research study which was carried out to determine the best type of check – in system to adopt at certain stations.

**BASED ON QUEUE METHOD:**

**Airport Simulation Based on Queuing Model Using ARENA The check-in process for passengers using the airport takes a lot of time for departure and a considerable delay is delayed until the total boarding time.**

**This way of arranging the passengers as the one by one helps to Identify and make the check in process quickly.**

### 3.2 **OPTIMIZATION :**

**The Statistics Module tracks performance metrics. The User Interface Module provides interaction and visualization capabilities. The Configuration Module allows users to customize simulation parameters. The Logging Module records simulation events, while the Optimization Module (optional) enhances efficiency based on performance data. This modular architecture ensures effective simulation of airport check-in desk operations with flexibility and scalability.**

### 3.3 FEATURES:

**This abstract outlines the objectives, methodology, and key features of the Airport Check-In Desk Simulator, offering a comprehensive overview of its functionality and potential applications.**

**airport check-in desk, allowing users to understand and analyze the efficiency of different check-in desk configurations and strategies. The program simulates various scenarios such as passenger arrivals, check-in procedures, luggage handling, and boarding processes.**

# CHAPTER 4

# RESULTS AND DISCUSSION

### 4.1 Results

**1. Passenger Class*: Represents individual passengers.**

**2. CheckInQueue*: Manages the queue of passengers.**

**3. CheckInDesk*: Simulates the check-in process, with a configurable processing time.**

**4. Event Scheduler: Handles the arrival and processing of passengers, ensuring they are checked in one by one.**

**5. Main Function: Initializes and starts the simulation.**

### 4.1.1 PASSENGER ARRIVAL

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

// Passenger structure
struct Passenger {
    char name[50];
    bool ticket_verified;
    bool luggage_loaded;
    bool
boarding_pass_issued;
    bool
boarding_pass_verified;
    struct Passenger* next;
};

// Check-in desk structure
struct CheckInDesk {
    struct Passenger*
front;
    struct Passenger* rear;
};
```

### 4.1.2 CHECK IN VERIFICATION

```c
// Check-in desk structure
struct CheckInDesk {
    struct Passenger*
front;
    struct Passenger* rear;
};

// Function to initialize
the check-in desk
void
initialize_check_in_desk(st
ruct CheckInDesk* desk) {
    desk->front = NULL;
    desk->rear = NULL;
}

// Function to check if the
queue is empty
bool is_empty(struct
CheckInDesk* desk) {
    return desk->front ==
NULL;
}
```

### 4.1.3 PASSENGER TICKET VERIFICATION

```
// Function to verify
ticket
void verify_ticket(struct
Passenger* passenger) {
    printf("Verifying
ticket for passenger:
%s\n", passenger->name);

passenger->ticket_verified
= true;
}

// Function to load luggage
void load_luggage(struct
Passenger* passenger) {
    printf("Loading luggage
for passenger: %s\n",
passenger->name);

passenger->luggage_loaded =
true;
}
```

### 4.1.4 BORDING

```
// Function to check if
passenger is ready to board
bool
is_boarding_pass_ok(struct
Passenger* passenger) {
    return
passenger->ticket_verified
&&
passenger->luggage_loaded
&&
passenger->boarding_pass_is
sued &&
passenger->boarding_pass_ve
rified;
}
```

## 4.2 Discussion

**An airport check-in desk simulator mimics the processpassengers undergo when checking in for a flight. This simulation helps in understanding the workflow, resource allocation, and potential bottlenecks in the check-in process. By using data structures such as queues, the simulation can effectively manage the order in which passengers are processed, ensuring a first-in-first out (FIFO) approach.**

**EXPLANATION :**

- ✓ **Passenger Structure :**

**Represents each passenger with relevant attributes, such as name and various stages of the check-in process (ticket verification, luggage loading, etc.).**

- ✓ **Check-In Desk Structure:**

**Manages the queue of passengers, using a linkedlist to simulate the FIFO order.**

- ✓ **Queue Operations:**

**Functions to enqueue (add) and dequeue (remove) passengers, ensuring they are processed in the correct order.**

- ✓ **Check-In Operations*:**

**Functions to simulate each step of the check-in process (e.g., verifying tickets, loading luggage, issuing boarding passes).**

# CHAPTER 5
# CONCLUSION

The airport check-in desk simulator provides a foundational understanding of how queues can be used to manage processes in a structured and efficient manner. By simulating the check-in process, it offers insights into potential improvements and helps in planning and resource management, ensuring a smoother experience for both passengers and staff. This simulator highlights the importance of systematic passenger handling and can be a valuable tool for training, capacity planning, and process optimization in real-world airport operations. By extending the simulation with more complex features and interactive elements, it can further enhance its practical applications and effectiveness.

## REFERENCES


1. **Books on Data Structures and Algorithms**
   - "Data Structures and Algorithm Analysis in C" by Mark Allen Weiss.
   - "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.

2. **Academic Papers and Theses**
   - "Queueing Theory and Applications: A Tutorial" by D. Gross and C.M. Harris.
   - Research papers on queue management and simulation in airport operations available in journals like the "Journal of Air Transport Management" or "Transportation Research Part A: Policy and Practice."

3. **Online Tutorials and Courses**
   - [GeeksforGeeks: Queue Data Structure](https://www.geeksforgeeks.org/queue-data-structure/)
   - [Coursera: Data Structures and Algorithm Specialization by UC San Diego](https://www.coursera.org/specializations/data-structures-algorithms)


These resources will provide comprehensive information on data structures, queue management, simulation techniques, and C programming,

# APPENDIX

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

// Passenger structure
struct Passenger {
    char name[50];
    bool ticket_verified;
    bool luggage_loaded;
    bool boarding_pass_issued;
    bool boarding_pass_verified;
    struct Passenger* next;
};
```

```
// Check-in desk structure
struct CheckInDesk {
    struct Passenger* front;
    struct Passenger* rear;
};

// Function to initialize the check-in desk
void initialize_check_in_desk(struct CheckInDesk* desk) {
    desk->front = NULL;
    desk->rear = NULL;
}

// Function to check if the queue is empty
bool is_empty(struct CheckInDesk* desk) {
    return desk->front == NULL;
}

// Function to enqueue a passenger
void enqueue(struct CheckInDesk* desk, const char* name) {
    struct Passenger* new_passenger = (struct Passenger*)malloc(sizeof(struct
Passenger));
    if (new_passenger == NULL) {
        printf("Memory allocation failed.\n");
        exit(EXIT_FAILURE);
    }
    snprintf(new_passenger->name, sizeof(new_passenger->name), "%s",
name);
    new_passenger->ticket_verified = false;
    new_passenger->luggage_loaded = false;
```

```
    new_passenger->boarding_pass_issued = false;
    new_passenger->boarding_pass_verified = false;
    new_passenger->next = NULL;

    if (is_empty(desk)) {
        desk->front = new_passenger;
        desk->rear = new_passenger;
    } else {
        desk->rear->next = new_passenger;
        desk->rear = new_passenger;
    }
}

// Function to dequeue a passenger

struct Passenger* dequeue(struct CheckInDesk* desk) {
    if (is_empty(desk)) {
        printf("Queue is empty.\n");
        exit(EXIT_FAILURE);
    }
    struct Passenger* temp = desk->front;
    desk->front = desk->front->next;
    if (desk->front == NULL) {
        desk->rear = NULL;
    }
    return temp;
}

// Function to verify ticket
```

```
void verify_ticket(struct Passenger* passenger) {
    printf("Verifying ticket for passenger: %s\n", passenger->name);
    passenger->ticket_verified = true;
}

// Function to load luggage
void load_luggage(struct Passenger* passenger) {
    printf("Loading luggage for passenger: %s\n", passenger->name);
    passenger->luggage_loaded = true;
}

// Function to issue boarding pass
void issue_boarding_pass(struct Passenger* passenger) {
    printf("Issuing boarding pass for passenger: %s\n", passenger->name);
    passenger->boarding_pass_issued = true;
}

// Function to verify boarding pass
void verify_boarding_pass(struct Passenger* passenger) {
    printf("Verifying boarding pass for passenger: %s\n", passenger->name);
    passenger->boarding_pass_verified = true;
}

// Function to check if passenger is ready to board
bool is_boarding_pass_ok(struct Passenger* passenger) {
    return passenger->ticket_verified && passenger->luggage_loaded &&
passenger->boarding_pass_issued && passenger->boarding_pass_verified;
}
```

```
int main() {
    struct CheckInDesk desk;
    initialize_check_in_desk(&desk);

    // Enqueue passengers
    enqueue(&desk, "John");
    enqueue(&desk, "Jane");
    enqueue(&desk, "Alice");

    // Process passengers
    while (!is_empty(&desk)) {
        struct Passenger* current_passenger = dequeue(&desk);
        verify_ticket(current_passenger);
        load_luggage(current_passenger);
        issue_boarding_pass(current_passenger);
        verify_boarding_pass(current_passenger);

        if (is_boarding_pass_ok(current_passenger)) {
            printf("%s is ready to board the flight.\n", current_passenger->name);
        } else {
            printf("%s is not ready to board the flight.\n", current_passenger-
>name);
        }

        free(current_passenger);
    }

    return 0;
}
```