AMRITA VISHWA VIDYAPEETHAM

CHENNAI CAMPUS

Second Year

B.Tech (Computer Science and Engineering)

Data Structure and Algorithms

Capstone Project Report

(From Academic year 2025-26)

Name: J.Darun Kumar
College Name: Amrita Vishwa Vidyapeetham
Roll No : CH.SC.U4CSE24111
Department: CSE B
Academic Year :2025-2026

Baggage Management System Using Queue

Efficient baggage management is essential for transportation systems involving multiple buses and stops. Traditional manual tracking methods often result in misplaced packages, delays, or overloading. This project implements a **Baggage Management System** using a **circular queue** in C, simulating multiple buses moving across multiple stops. The system allows users to load packages, process stops, and unload packages according to their destinations. The circular queue ensures **FIFO-based package handling**, efficient space reuse, and prevents overloading. The proposed system provides a **structured and reliable approach** to package management, ensuring all packages reach their intended destinations safely.

Introduction

Managing packages efficiently in a multi-bus, multi-stop scenario is crucial to ensure timely delivery and prevent mishandling. Manual tracking or linear storage methods often lead to delays and inefficiencies.

Problem Statement

Bus transportation systems face several challenges:

- Packages may be misplaced or left behind at intermediate stops.
- Overloading of buses can occur due to lack of tracking.
- Packages need to be efficiently handled without data loss.
- A structured, scalable system is required to manage multiple buses and stops simultaneously.

Objectives

- To implement a baggage management system using **circular queues**.
- To allow users to **load, transport, and unload packages** at multiple stops.
- To prevent bus overloading and maintain FIFO order for packages.
- To ensure packages are delivered to their respective stops efficiently.
- To simulate a **real-time bus stop delivery system** in software.

Review

Existing systems in logistics management often rely on databases or manual logs, which can be complex, slow, or prone to human error.

Queue-based data structures, especially **circular queues**, provide an efficient way to manage sequential tasks. They are widely used in buffering, scheduling, and real-time system simulations. The circular queue allows **constant-time enqueue and dequeue operations** and reuses memory efficiently, which is ideal for handling packages on buses.

Methodology

System Architecture:

The system has three main components:

1. **Bus Layer** – Each bus has a unique ID and a queue for packages.
2. **Queue Layer (Circular Queue)** – Manages package loading and unloading in FIFO order.
3. **Simulation Layer** – Moves buses through stops, processes unloading, and handles final stop delivery.

Data Structures Used:

- **Package Structure**: Contains package ID, owner name, and destination.
- **CircularQueue Structure**: Implements a fixed-size queue with front, rear, and size.
- **Bus Structure**: Contains bus ID and the package queue.

Core Functionalities:

Load Package

- Users input package details: owner name, destination stop, and bus number.
- Packages are added to the bus queue if space is available.

Process Stop

- Each bus stops sequentially from Stop 1 to the final stop.
- Packages destined for the current stop are unloaded.
- Packages not destined for the current stop are re-enqueued for later stops.

- At the final stop, all remaining packages are unloaded.

Queue Operations

- enqueue() – Adds package to the bus queue if not full.
- dequeue() – Removes package from the front of the queue.
- isFull() / isEmpty() – Check queue status to avoid errors.

Simulation Flow

- Buses move stop by stop.
- Packages are unloaded at their destination stops.
- Final stop unloads all remaining packages.

Advantages

- **Efficient Handling** – FIFO ensures packages are handled in order.
- **Prevents Overloading** – Circular queue checks prevent buses from exceeding capacity.
- **Scalable** – Can handle multiple buses and stops efficiently.
- **Avoids Package Loss** – Packages not unloaded at a stop are retained in the queue.
- **Space Reuse** – Circular queue efficiently reuses memory after dequeue operations.
- **User Friendly** – Menu-based system for easy input and simulation.
- **Structured Simulation** – Mimics real-life bus stop package delivery.

Conclusion

The project successfully implements a **Baggage Management System** using **circular queues** in C. The system ensures efficient package loading, transport, and unloading for multiple buses and stops. FIFO handling, queue size checks, and final stop processing make it reliable, structured, and user-friendly.

Future Scope

- Adding a **GUI interface** for real-time simulation visualization.
- Integration with **barcodes/RFID** for package tracking.
- Supporting dynamic queue resizing for larger or variable bus capacities.
- Extending the system for **logistics companies** with real-time delivery updates.

SOURCE CODE : [C language]

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

#define MAX 3        // max packages per bus
#define BUSES 5       // total buses
#define FINAL_STOP 4  // last stop

typedef struct {
    int id;
    int destination;
    char owner[50];
} Package;

typedef struct {
    Package items[MAX];
    int front, rear, size;
} CircularQueue;

typedef struct {
    int busId;
    CircularQueue queue;
} Bus;

// Queue functions
void initQueue(CircularQueue *q) {
    q->front = 0;
    q->rear = -1;
    q->size = 0;
}

int isFull(CircularQueue *q) {
    return q->size == MAX;
}

int isEmpty(CircularQueue *q) {
    return q->size == 0;
}

void enqueue(CircularQueue *q, Package p) {
    if (isFull(q)) {
        printf("Queue is full! Cannot add package %d.\n", p.id);
        return;
    }
    q->rear = (q->rear + 1) % MAX;
```

```
    q->items[q->rear] = p;
    q->size++;
}

Package dequeue(CircularQueue *q) {
    Package p = { -1, -1, "" };
    if (isEmpty(q)) {
        return p;
    }
    p = q->items[q->front];
    q->front = (q->front + 1) % MAX;
    q->size--;
    return p;
}

// Bus functions
void loadPackage(Bus *bus, int *pkgCounter) {
    if (isFull(&bus->queue)) {
        printf("Bus %d is full!\n", bus->busId);
        return;
    }
    Package p;
    p.id = ++(*pkgCounter);
    printf("Enter owner name: ");
    scanf("%s", p.owner);
    printf("Enter destination stop (1-3, final=4): ");
    scanf("%d", &p.destination);

    enqueue(&bus->queue, p);
}

void processStop(Bus *bus, int stop) {
    int size = bus->queue.size;
    printf("\n--- Bus %d at Stop %d ---\n", bus->busId, stop);

    for (int i = 0; i < size; i++) {
        Package p = dequeue(&bus->queue);
        if (p.destination == stop) {
            printf("Unloaded package %d for %s)\n",p.id, p.owner);
        } else {
            enqueue(&bus->queue, p); // keep for later stops
        }
    }

    if (stop == FINAL_STOP) {
        while (!isEmpty(&bus->queue)) {
            Package p = dequeue(&bus->queue);
            printf("Final Stop: Unloaded package %d (Owner: %s)\n",
```

```
                    p.id, p.owner);
        }
    }
}

int main() {
    Bus buses[BUSES];
    int pkgCounter = 0;
    int busNo, n;

    // Initialize buses
    for (int i = 0; i < BUSES; i++) {
        buses[i].busId = i + 1;
        initQueue(&buses[i].queue);
    }

    // Load packages initially
    printf("Enter number of packages to load: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("\nPackage %d:\n", i + 1);
        printf("Enter bus number (1-%d): ", BUSES);
        scanf("%d", &busNo);

        if (busNo >= 1 && busNo <= BUSES) {
            loadPackage(&buses[busNo - 1], &pkgCounter);
        } else {
            printf("Invalid bus number! Skipping package.\n");
        }
    }

    // Simulation: move each bus from stop 1 → FINAL_STOP
    for (int stop = 1; stop <= FINAL_STOP; stop++) {
        printf("\n=== Stop %d ===\n", stop);
        for (int i = 0; i < BUSES; i++) {
            processStop(&buses[i], stop);
        }
    }

    printf("\nSimulation complete. All buses are empty at the final stop.\n");
    return 0;
}
```

Output:

```
Enter number of packages to load: 10

Package 1:
Enter bus number (1-5): 1
Enter owner name: darun
Enter destination stop (1-3, final=4): 2

Package 2:
Enter bus number (1-5): 2
Enter owner name: kevin
Enter destination stop (1-3, final=4): 1

Package 3:
Enter bus number (1-5): 3
Enter owner name: vikky
Enter destination stop (1-3, final=4): 3

Package 4:
Enter bus number (1-5): 1
Enter owner name: prakash
Enter destination stop (1-3, final=4): 4

Package 5:
Enter bus number (1-5): 1
Enter owner name: aswath
Enter destination stop (1-3, final=4): 1

Package 6:
Enter bus number (1-5): 5
Enter owner name: nithis
Enter destination stop (1-3, final=4): 2

Package 7:
Enter bus number (1-5): 4
Enter owner name: halit
Enter destination stop (1-3, final=4): 2

Package 8:
Enter bus number (1-5): 5
Enter owner name: karthi
Enter destination stop (1-3, final=4): 3
```

```
Package 9:
Enter bus number (1-5): 4
Enter owner name: dhakshin
Enter destination stop (1-3, final=4): 4

Package 10:
Enter bus number (1-5): 5
Enter owner name: deepak
Enter destination stop (1-3, final=4): 2

=== Stop 1 ===

--- Bus 1 at Stop 1 ---
Unloaded package 5 for aswath)

--- Bus 2 at Stop 1 ---
Unloaded package 2 for kevin)

--- Bus 3 at Stop 1 ---

--- Bus 4 at Stop 1 ---

--- Bus 5 at Stop 1 ---

=== Stop 2 ===

--- Bus 1 at Stop 2 ---
Unloaded package 1 for darun)

--- Bus 2 at Stop 2 ---

--- Bus 3 at Stop 2 ---

--- Bus 4 at Stop 2 ---
Unloaded package 7 for halit)
```

```
--- Bus 5 at Stop 2 ---
Unloaded package 6 for nithis)
Unloaded package 10 for deepak)

=== Stop 3 ===

--- Bus 1 at Stop 3 ---

--- Bus 2 at Stop 3 ---

--- Bus 3 at Stop 3 ---
Unloaded package 3 for vikky)

--- Bus 4 at Stop 3 ---

--- Bus 5 at Stop 3 ---
Unloaded package 8 for karthi)

=== Stop 4 ===

--- Bus 1 at Stop 4 ---
Unloaded package 4 for prakash)

--- Bus 2 at Stop 4 ---

--- Bus 3 at Stop 4 ---

--- Bus 4 at Stop 4 ---
Unloaded package 9 for dhakshin)

--- Bus 5 at Stop 4 ---
```