Postal Package Queue System: Project Layout

Below is a comprehensive project structure for a **Postal Package Queue System**, tailored for managing packages (not just customers) at a postal counter. This system handles queue management based on package type, priority, and entry time—featuring search, sorting, and future extensibility.

1. Introduction

Postal package counters face frequent congestion and delays. An efficient queue system helps prioritize urgent parcels, manages packages by entry time, and allows quick lookup—improving operational workflow.

2. Team Members

SN	NAME	ROLL NO.	ENROLLEMENT NO.	THEIR CONTRIBUTION
1.	HAARISH .A. KHAN PATHAN	71	2403031460059	Input, search functions
2.	KARANSINH RAJPUROHIT	20	2403031460892	Display module and Sorting algorithms
3.	YUVRAJ GUPTA	41	2403031460892	Main driver and declarations
4.	KARNATI VENKATA JASWANTH REDDY	68	2403031460215	Sorting algorithms
5.	MADI REDDY PAVAN TEJ REDDY	59	2403031460937	System integration tests

3. Objective

Create a **C program** that:

- Maintains a queue of package records (name, type, priority, entry time).
- Sorts queue by package priority (express, standard, etc.) and entry time.

- Searches for package info by sender name or package ID.
- Displays current queue status.

4. Problem Statement

Design a Postal Package Queue System that:

- Stores each package's sender name, type (letter, parcel, money order, etc.), priority, and entry time.
- Allows sorting queue by priority (express first), then entry time.
- Enables searching for packages by sender name or package ID.

5. System Requirements

- **Software**: C Compiler (Turbo C++, Dev C++), or modern text editor
- **Hardware**: Standard PC/Laptop, 2GB+ RAM, basic I/O accessories

6. Data Structures Used

- **Structure Array**: Store package details (name, ID, type, priority, entry time)
- **Sorting Algorithms**: Bubble sort or selection sort
- **Searching**: Linear search or binary search (if sorted)

7. Modules and Description

Input Module	**Accepts package info (name, ID, type, priority, entry time)**
MODULE	**DESCRIPTION**
Display Module	**Shows list of packages with position, type, and timings**
Sort Module	**Sorts queue by package priority and entry time**
Search Module	**Searches by sender name or package ID**

8. Algorithms

A. Sorting by Priority & Entry Time

c

```
void sortByPriorityTime(Package p[], int n) {
 Package temp;
```

```
 for (int i = 0; i < n - 1; i++) {
 for (int j = 0; j < n - i - 1; j++) {
 if ((p[j].priority < p[j + 1].priority) ||
 (p[j].priority == p[j + 1].priority &&
 p[j].entryTime > p[j + 1].entryTime)) {
 temp = p[j];
 p[j] = p[j + 1];
 p[j + 1] = temp;
 }
 }
 }
}
```

B. Linear Search by Sender Name/Package ID

c

```
int searchBySender(Package p[], int n, char targetName[]) {
 for (int i = 0; i < n; i++) {
 if (strcasecmp(p[i].senderName, targetName) == 0) {
 return i;
 }
 }
 return -1;
}
```

c

```
int searchByID(Package p[], int n, int targetID) {
 for (int i = 0; i < n; i++) {
 if (p[i].packageID == targetID) {
```

```
 return i;
 }
 }
 return -1;
}
```

9. Sample Input/Output

Input:

- Enter number of packages: 4
- Package 1: Sender Name: Amit, Type: Parcel, Priority: 2, Entry Time: 965

Output (sorted):

- Express parcels first, then by earliest entry.
- Search example: Enter Sender Name: Amit → found at position 2, details displayed.

10. Test Cases

Test Case	Input	Sort Type	Output Type
1	**4 packages**	**Priority**	**Sorted: Highest priority first**
2	**4 packages**	**Entry Time**	**Sorted: Earliest entry first**
3	**"Amit"**	**Search Name**	**Shows position & package details**
4	**Invalid ID**	**Search ID**	**Output: Not found message**

11. Future Enhancements

- Add file saving/loading for package records
- Partial/regex search of sender/package type
- Detailed package tracking (status, location)
- Integration with barcode scanning
- GUI for real-time queue display

- Mobile notifications for ready packages

12. Conclusion

This *Postal Package Queue System* enables methodical package management at postal counters. By sorting parcels by urgency and arrival, and enabling quick search, it helps optimize customer service and accuracy in package handling.

13. References

- Data Structures by Jashwanth Sir
- C Programming resources
- Code examples from online forums
- ByteXL-App