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

Title: LOST AND FOUND PROJECT USING DATA STRUCTURE ALGORITHM USING C

Sustainability Goal :To provide an efficient and eco-friendly digital solution for managing lost and found items

Domain:	Data Structures & Algorithms (DSA)
Member:	**PARTH RAHUL KHISMATRAO** **ROLL NO : 25** **D10B**


1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
THE GLOBAL GOALS
For Sustainable Development
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE AND JUSTICE STRONGINSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

Content

Introduction to Project

The **Lost and Found Application** is a mini project developed in **C language** as part of the **Data Structures and Algorithms (DSA)** course. The project addresses a common problem in colleges where students frequently misplace items such as ID cards, books, and accessories.

To solve this, we designed a **menu-driven program** that allows users to **report lost items, report found items, search, display, and delete records**. The application makes extensive use of **DSA concepts**:

- **Linked Lists** for dynamic storage of records,
- **Hash Tables** for fast searching, and
- **Binary Search Trees (BSTs)** for sorted retrieval.

This project demonstrates how theoretical DSA concepts can be applied in real-world scenarios, while also providing an efficient and eco-friendly digital solution for managing lost and found items in a college environment.

Problem Statement

In a college environment, students often misplace essential items such as ID cards, books, and electronic accessories. The existing manual system of handling lost and found items is a VESIT lost and found group which is **unorganized, time-consuming, and unreliable**.

There is a need for a **structured digital system** that can **record, search, and manage** lost and found items efficiently. By applying **Data Structures in C**, this project aims to provide a solution that ensures quick retrieval, organized storage, and easy tracking of lost and found records.

Objectives of the project

- To design a digital system for recording lost and found items.
- To implement **Linked Lists, Hashing, and Trees** for efficient data management.
- To enable quick **insertion, searching, and deletion** of records.
- To demonstrate how **DSA in C** can solve real-world college problems.

Requirements of the system (Hardware, software)

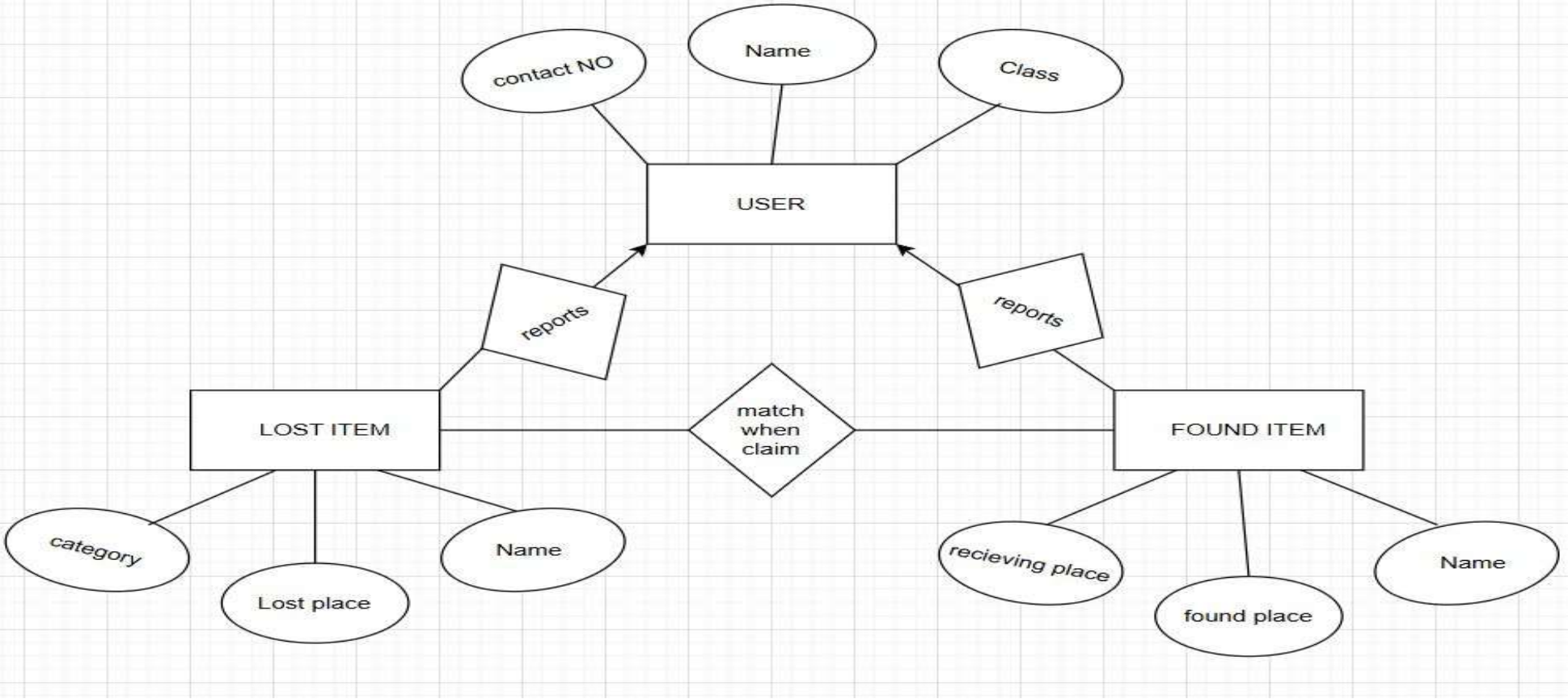
Hardware Requirements

- RAM: Minimum 2 GB (4 GB recommended)
- Storage: 100 MB free space

Software Requirements

- Operating System: Windows / Linux / macOS
- Compiler: GCC / Turbo C / Code::Blocks / Dev C++
- Language: C
- GitHub (for version control and submission)

ER diagram of the proposed system



DATA STRUCTURES & CONCEPTS USED

- **Linked List:** Dynamic storage of lost/found items.
- **Hash Table:** Fast searching of items by name or category.
- **Binary Search Tree (BST):** Sorted retrieval of items by name/date.
- **File Handling:** To store and retrieve records persistently.
- **Pointers & Structures in C:** Efficient record management.

Algorithm Explanation

Add Item:

1. Create a new node.
2. Insert into linked list.
3. Update hash table for quick lookup.

Search Item:

1. Generate hash key from item name.
2. Check hash table entry.
3. If found, return details; else, “Not found.”

Delete Item (Mark Returned):

1. Traverse linked list to find node by ID.
2. Remove node.
3. Update hash table / BST.

Time and Space Complex

Linked List:

- Insertion: O(1)
- Deletion: O(n)
- Traversal: O(n)

Hash Table:

- Search: O(1) average, O(n) worst case

BST:

- Insertion: O(log n) average, O(n) worst case
- Search: O(log n) average, O(n) worst case

Space Complexity: O(n) for storing items.

Front End

- **Interface:** Console-based (menu driven).
- **Features:**
  - Report Lost Item
  - Report Found Item
  - Search Item
  - Display All Items
  - Mark Item Returned
- **User Interaction:** Simple input/output using `printf` and `scanf`.

Implementation

Implemented in **C** using modular functions.

Data stored in **struct Item { … }**.

Core functions:

- `addLostItem()`
- `addFoundItem()`
- `searchItem()`
- `deleteItem()`
- `displayItems()`

GitHub used for version control and submission.

Gantt Chart

DAY 1: Problem study & system design

DAY 2: Data structure selection & coding core modules

DAY 3: Integration & testing

DAY 4: Documentation, PPT, GitHub upload

Test Cases

Test Case	Input	Expected Output	Result
1	Add Lost Item “ID Card”	Item added successfully	Pass
2	Search “Book”	Item details shown	Pass
3	Delete Item “Headphones”	Item removed	Pass
4	Display All	List of all items	Pass

Challenges and Solutions

Challenge: Efficient searching of items.

- **Solution:** Used hash table.

Challenge: Dynamic storage of varying items.

- **Solution:** Used linked list.

Challenge: Sorted output.

- **Solution:** Implemented BST.

Challenge: Persistent storage.

- **Solution:** Added file handling.

Future Scope

- User authentication (login system).
- Email/SMS notifications to owners.
- Graph-based system to track location of lost items.
- GUI or web application integration with college ERP.

Output Screenshots

```
========== COLLEGE LOST & FOUND SYSTEM ==========
1. Report Lost Item
2. Report Found Item
3. View All Lost Items
4. View All Found Items
5. Search by Category
6. Search by Keyword
7. Match Lost and Found Items
8. Save Data to File
9. Load Data from File
0. Exit
================================================
Enter your choice: 1

--- REPORT LOST ITEM ---
Item Name: bottle
Description: black ,tupperware

Select Category:
0. Electronics
1. Books
2. Clothing
3. Accessories
4. Stationery
5. Sports
```

TO UPLOAD LOST ITEM

```
========= COLLEGE LOST & FOUND SYSTEM =========
1. Report Lost Item
2. Report Found Item
3. View All Lost Items
4. View All Found Items
5. Search by Category
6. Search by Keyword
7. Match Lost and Found Items
8. Save Data to File
9. Load Data from File
0. Exit
=========================================
Enter your choice: 3

--- LOST ITEMS ---
ID    Name              Description               Category     Location          Date
=====================================================================
1000  bottle            black ,tupperware         Accessories                    22/09/2025

Total lost items: 1
```

TO CHECK LOST ITEMS

```
--- REPORT FOUND ITEM ---
Item Name: ID CARD
Description: PARTH KHI

Select Category:
0. Electronics
1. Books
2. Clothing
3. Accessories
4. Stationery
5. Sports
6. Keys
7. Bags
8. Jewelry
9. Others
Enter choice (0-9): 3
Found location: Your contact info: 989
Found item reported successfully! ID: 1000
```

TO REPORT FOUND ITEM

Conclusion

The **Lost and Found Application** successfully demonstrates the application of **DSA concepts in C**.
It provides an efficient way to manage lost and found records using **Linked Lists, Hash Tables, and BSTs**.

This project bridges the gap between **theoretical data structures** and **real-world problem solving**.

References

1. Reema Thareja, *Data Structures Using C*, Oxford University Press, 2014.

2. Ellis Horowitz, Sartaj Sahni, *Fundamentals of Data Structures in C*, 2nd Edition, 2008.

3. Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language*, Prentice Hall, 1988.

4. Class lecture notes and online resources.