PROJECT AND TEAM INFORMATION

Project Title:

Lost and Found portal for college using C

Student /Team Information

|  |  |
| --- | --- |
| Team name: | Byte Storm |
| Team member 1(Leader)  ( Negi : Antriksh : 241601259: [antrikshnegi2007@gmail.com](mailto:antrikshnegi2007@gmail.com) ) |  |
| Team member 2:  ( Tiwari : Ayush : 241601311 : [ayush.tiwari0618@gmail.com](mailto:ayush.tiwari0618@gmail.com) ) | A person wearing glasses and a white shirt  AI-generated content may be incorrect. |
| Team member 3:  ( Thapa : Kriti : 241602072 :  [kritithapa609@gmail.com](mailto:kritithapa609@gmail.com) ) |  |
| Team member 4 :  ( Chauhan : Rohit : 241601103 :  [Rc9634708@gmail.com](mailto:Rc9634708@gmail.com) ) |  |

Proposal Description:

Motivation:

In colleges, students and faculty frequently misplace important belongings such as ID cards, books, keys, wallets, and electronic devices. Currently, these lost and found cases are managed inefficiently through word-of-mouth or physical notice boards, making item recovery difficult and slow.

Our goal is to create a centralized Lost & Found Portal using C programming that allows students and staff to report, search, and retrieve lost & found items efficiently. By implementing Linked Lists, Hash Tables, and Binary Search Trees (BST), the system will enable quick and organized item searching, reducing the hassle of finding lost objects.

# State of the Art / current solution:

Currently, lost items are managed using manual methods such as:

1. Physical Bulletin Boards – Where people put up notices about lost or found items.
2. Word-of-Mouth & Social Media – Asking around or posting in student groups.

3. College Admin Desk – Some colleges maintain a register of lost & found items.

However, these methods are slow, unreliable, and lack proper organization, leading to many unclaimed items. Our system provides a fast, digital, and structured way to manage lost & found cases.

# Project Goals and Milestones:

Goals:

Build a Lost & Found Portal using C and file handling.

Provide a CLI menu to report, view, search, and claim items.

Store all data in a simple text file (lost\_found.txt).

Make it easy to deploy on any college computer without requiring internet or server setup.

Milestones:

The project began with a requirement analysis phase, where we identified the core needs—such as offline usage, ease of reporting, and file-based storage. Once the goals were clear, we moved on to designing the structure of the program and planning the user interface. The development phase involved writing modular C functions for reporting items, viewing the list, searching using keywords, and claiming found items. After implementing the core features, we integrated file handling using a text file (lost\_found.txt) to store item data persistently.

With the basic features completed, the project entered the testing and debugging phase, where the functionality is being verified using sample inputs and edge cases. Next, we will focus on writing the final documentation, creating a demo video, and preparing the presentation for submission. The project is on track for timely completion and ready for demonstration on local college systems.

# Project Approach:

Project Approach

Modular Programming in C: Separate functions for each feature (report, view, search, claim).

Text File Storage: Data saved as pipe-separated values in lost\_found.txt.

Simple Search: Use strstr() to match keywords during search.

Offline Ready: Runs without internet or network, ideal for lab computers.

User-Friendly CLI: Menu-based interaction for easy use by students and staff.

Implementation Steps:

Implementation Steps

Plan Features

Decide on key actions: report, view, search, and claim items.

Design File Format

Use a simple text file with pipe-separated fields.

Set Up Menu

Create a command-line menu to navigate the features.

Write Functions

Implement C functions for each feature using file handling.

Test Functionality

# System Architecture:

Main Components:

System Architecture

The Lost & Found Portal uses a simple three-layer architecture:

* User Interface Layer
* Command-line based interface
* Users interact through a menu to report, view, search, or claim items.
* Input is taken using standard C functions like scanf().
* Application Logic Layer
* Handles core functionality with modular C functions.
* Includes logic for adding new entries, displaying items, searching by keyword, and deleting claimed items.
* Manages formatting and internal control flow.
* Data Storage Layer
* Stores all item records in a plain text file (lost\_found.txt).
* Each line represents one item, formatted using delimiters (e.g., | symbol).
* Uses standard file handling functions (fopen(), fprintf(), fgets(), etc.) to read/write data.
* The architecture is designed to run offline, without requiring any network, server, or database, making it ideal for use on individual college computers.

# Assumptions

The system will be used locally on a single machine.

Users interact only through a text-based interface.

The text file (lost\_found.txt) remains intact and correctly formatted.

All inputs are assumed to be safe and valid (no validation for SQL/file injection).

# References:

The C Programming Language – Brian Kernighan & Dennis Ritchie

GeeksforGeeks: File Handling in C

Stack Overflow (for practical C syntax & logic)

Personal debugging and testing on Linux/Windows terminal

# Internal Instructor / Mentor

Signature