67070503459 KITTIPHAT NOIKATE

67070503455 PHOO PHOO THIT

67070503402 GRITTAPOB CHUTITAS

A PROJECT SUBMITTED IN PROGRAMMING AND DATA STRUCTURES PROJECT

Project Title	Scam Call Analyzer	
Member(s)	Kittiphat Noikate	67070503459
	Phoo Phoo Phit	67070503455
	Grittapob Chutitas	67070503402

Program	Bachelor of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2024

Abstract

This project presents a Scam Call Analyzer system to help efficiently store, retrieve and manage reported phone numbers. The system is developed in C and organized into admin, user and main components. The hash table supports fundamental operations such as insertion, search, deletion, and full-table display. Chaining is implemented using linked lists to accommodate multiple entries at the same hash index, ensuring efficient and reliable storage even in the case of collisions. Risk scores are calculated dynamically based on **report frequency**, **geographic origin**, and **relationship connectivity**, with special emphasis on foreign numbers and non-SEA regions. Persistent data is stored in CSV format, allowing the system to resume state across sessions.

CONTENTS

TITLE **PAGE**

ABSTRACT I

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Approach 4

1.2 Objective 4

1.3 Scope 4

1.4 Expected Outcome 4

CHAPTER 2 SYSTEM DESIGN

2.1 Overview 5

2.2 Data Structures and Time Complexity 5-6

2.3 Risk Scoring and Relationships 6

CHAPTER 3 IMPLEMENTATION

3.1 Key Functionalities 7

3.2 Admin and User Module 7

CHAPTER 4 CORE PROGRAM VALIDATION

4.1 Code Walkthrough 7-8

4.2 Program Time/Space Complexity Table 9

4.3 Sample Test Case 9

CHAPTER 5 CONCLUSION

5.1 Real World Reflection 10

5.2 Team Responsibilities 10

5.3 References 10

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Approach :

Scam calls have been a common cyberthreat in today's digital world. These scams often involve phishing where callers pretend to be trustworthy individuals or companies to steal data from you. Many people are caught off guard and have no reliable way to verify unknown callers. To address this, we developed the **Scam Call Analyzer**, a command-line tool made with C that helps users report, analyze, and track suspicious phone numbers.

1.2 Objectives

The tool is designed to help users evaluate whether a phone number may be linked to scams. It enables:

- **User-admin collaboration**, where users report suspicious numbers and admins review and update the scam database.
- **Dynamic risk scoring**, calculated based on report frequency, origin country, and scam connectivity.
- **Graph-based relationship mapping**, revealing connections between scam numbers.
- **Activity logging**, which records all major system events for review and debugging.

1.3 Scope

The tool focuses on the Southeast Asian (SEA) region, recognizing phone numbers from countries like Thailand, Vietnam, Malaysia, and Indonesia. It collects phone numbers, risk scores, and report counts. Main features include real-time number lookups, admin review tools, relationship graph visualization, and CSV-based data storage. It is designed for everyday users and admins, operating fully through a command-line interface. However, it relies on user input and does not include a graphical UI or automated scam detection.

1.4 Expected Outcomes :

This project aims to provide a simple, effective solution for tracking scam calls. Users can contribute to improving the database by reporting unknown numbers, while admins ensure that data stays accurate and up to date. The logging system helps monitor usage and errors, which can guide future improvements to both functionality and usability.

CHAPTER 2 SYSTEM DESIGN

2.1 Overview

The Scam Call Analyzer is a terminal based tool made with C programming language designed to help users identify and report suspicious phone numbers. The system is divided into different components- the user, admin and the main logic components for a more orderly aesthetic and easier maintenance. Users can search and report for phone numbers, or view their risk level. On the other hand, Admins have tools to accept or reject user reports, analyze network connections between numbers, and maintain the integrity of the stored data. At its core the system works via a combination of hash tables for storage and retrieval of records and a graph structure to represent the relationships between different phone numbers.

2.2 Data Structures and Time Complexity

The Scam Call Analyzer uses three primary data structures which are hash table, graph, and queue. All three data structures are implemented using the C language.

1. **Hash Table**

   The **Hash table** is used to store data, this encompasses phone numbers, their associated risk score and the number of reports received. To handle collisions the tool uses separate chaining with linked lists for multiple entries to exist in the same hash table without colliding.

   The hash table is primarily used for fast lookups and storage of phone numbers. The time complexity of hash table for primary functions are Adding phone number to the table (insertion) , Looking up a phone number by its key (Search) , and Removing a phone number from the table (Deletion) are:

   - Average Case : **O(1)**
   - Worst Case: **O(n)**

2. **Graph**

   The **graph** is used in tracking the relationships between different phone numbers. With each node representing a phone number and edges between nodes representing a suspected connection.

   The graph is used to analyze the relationships between phone numbers. The key operations and their time complexities in graph module are

   - Adding a node (phone number) : **O(1)**
   - Adding an Edge (Relationship between phone numbers) : **O(1)**
   - Searching for a Relationship (Edge between two nodes) : **O(1)**
   - Graph Traversal (BFS or DFS) : **O(V+E)**

     V = Number of vertices

     E = Number of edges

3. **Queue**

   The **queue** is used to manage the order of processing tasks or phone numbers in the table. The data is handled in a **First-In-First-Out** order. The queue is an effective data structure to complete the tasks without overlapping or skipping steps.

   The queue is used to handle tasks and process phone numbers in a **First-In-First-Out (FIFO)** order. The time complexity for operations on a queue is

   - Enqueue (Adding an item to the queue) : **O(1)**
   - Dequeue (Removing an item form the queue) : **O(1)**

4. **CSV File Handling**

   The system utilizes a CSV file to save its data to ensure all information is recorded even when the program is closed. It also keeps a log of actions taken i.e reports being submitted or records being updated for future revisions.

   The CSV file is used to store records. The time complexity of reading from and writing to the CSV file can vary based on the size of the file:

   - Read (Getting data from the CSV file) : **O(n)**
   - Write (Saving data to the CSV file) : **O(n)**

   Here, n represents the number of entries (phone numbers and others) in the file.

2.3 Risk Scoring and Relationship

One of the key features of the Scam Call Analyzer is the risk scorer. This gives each phone number its associated risk score based on 3 main factors.

1. **Number of Reports**

   The more a user reports the same number the higher the risk score. The system can detect risky phone numbers that users report by this way.

2. **Geological Location**

   The program checks if it is a SEA number or foreign numbers. If it is a non-SEA number , the score is increased because scammers often use foreign numbers to avoid detection.

3. **Connection to other Suspected Numbers**

   If a number is linked to an existing number that was flagged as suspicious, the risk score would also be increased.

Alongside this the system uses graphs to uncover scam networks and identify risky numbers by finding links from two different numbers and linking them in a graph.

CHAPTER 3 IMPLEMENTATION

3.1 Key Functionalities

The Scam Call Analyzer provides essential tools for identifying and managing suspicious phone numbers. Users can search for numbers, view their risk scores, and report suspicious activity. Admins review these reports, update scam records, and define links between numbers to uncover scam networks.

3.2 Admin and User Module

The Scam Call Analyzer has two main modules, the Admin Module and The User module. Each module is designed with different responsibilities in mind.

3.2.1 Admin Module

The Admin Module gives the administrators various tools to manage the database.The admins have full control over the scam call database.Admins have these capabilities.

- Reviewing and Approving/Rejecting user-submitted reports
- Manually adding or updating recorded data
- Creating graph links between numbers to define relationships to identify scam networks
- Deleting phone number records that are invalid, duplicated, or error
- View formatted tables of scam record

3.2.2 User Module

The user module allows the public to interact with the Scam Call Analyzer through it. User module's functionalities are:

- Searching for a specific phone number
- Viewing a number's risk level
- Reporting a suspicious number
- Viewing the graph-based connection map

CHAPTER 4 CORE PROGRAM VALIDATION

4.1 Code Walkthough

4.1.1 Hash Table Lookup - Fast record lookup via hash index

```c
ScamRecord* hash_table_lookup(HashTable *table, const char *phone){

    if(!table || !phone) return NULL;

    unsigned int index = hash_function(phone);
    ScamRecord *current = table->buckets[index];

    while(current){
        if(strcmp(current->phone, phone) == 0){
            return current;
        }
        current = current->next;
    }
    return NULL;

}
```

This function searches for a phone number in the hash table using a hash index and traverses the linked list in that bucket. It enables constant-time average lookup: O(1).

4.1.2 Graph Add Edge - Create bidirectional links in scam graph

```
void graph_add_edge(GraphNode *nodes[], const char *a, const char *b){

    GraphNode *na = graph_get_node(nodes, a);
    GraphNode *nb = graph_get_node(nodes, b);
    if(!na || !nb) return;

    if(!already_connected(nodes, na->phone, nb->phone) && na->neighbor_count < MAX_NEIGHB
        na->neighbors[na->neighbor_count++] = nb;

    if(!already_connected(nodes, nb->phone, na->phone) && nb->neighbor_count < MAX_NEIGHB
        nb->neighbors[nb->neighbor_count++] = na;

}
```

This function creates a mutual connection between two phone numbers in the scam relationship graph, ensuring no duplication and respecting a neighbor limit.

4.1.3 Calculate Score - Core scam risk algorithm

```
float calculate_score(const char *phone, int report_count, int neighbor_count){

    float base;
    if(!Is_SEA_Country(phone)) return 1.0f;

    if(strncmp(phone, "+66", 3) == 0){
        if(strncmp(phone, "+662", 4) == 0) base = 0.5f + 0.05f * report_count;
        else                               base = 0.1f + 0.05f * report_count;
    }else if(strncmp(phone, "+855", 4)==0 || strncmp(phone, "+95", 3)==0 || strncmp(phone
        base = 0.7f + 0.05f * report_count;
    else
        base = 0.8f + 0.05f * report_count;

    float bonus = fminf(0.2f, 0.05f * neighbor_count);
    return fminf(1.0f, base + bonus);

}
```

This algorithm assigns a risk score to a phone number based on region, number of reports, and graph connections. Foreign numbers receive a maximum score of 1.0, while others increase based on user reports and relationships.

4.2 Program Time/Space Complexity Table

Operation	Time	Space	Description
Hash Insert / Look up	O(1) Avg case	O(n)	Worst case (chaining collision)
Graph Add Edge	O(1)	O(1)	Simple bidirectional edge insert
BFS / DFS Traversal	O(V + E)	O(V)	V = Number of nodes, E = Edges
CSV Read	O(n + e)	O(n + e)	Load scam into the data structures
CSV Write	O(n + e)	O(1)	Save all scam into the databases

4.3 Sample Test Case

4.3.1 Sample Database

Scam Number Record	Scam Number Edge	Pending Report
R, +66812345678, 0.60, 3	E, +66812345678, +66898765432	+66812345678 2025-06-04 12:00:00
R, +66898765432, 0.85, 5		

4.3.2 Sample User Test

- User search : 0812345678

Input	Output
Suspicious risk score percentage	60%
Report counting	3
Graph link to	+66898765432
Report to admin	Phone was reported into the pending file.

4.3.3 Sample Admin Test

- Admin accept : +66812345678 2025-06-04 12:00:00

Input	Output
Accept report	Report was moved into the scam record.

CHAPTER 5 CONCLUSION

5.1 Real World Reflection

This project shows how programming can be used to solve real world problems like scam calls. By applying the use of data structures such as hash tables we can create a tool that can prevent users from getting scammed, reminding us that simple tools can make a difference for public safety. This tool can also be further improved and integrated into larger scale usage, allowing for larger scam call centers to be caught.

5.2 Team Responsibilities

Member	**Role**	**Responsibilities**
Kittiphat Noikate	Structures & Storage	Data Structures: ● Hash Table ● Graph ● Queue Phone Formatting: ● Normalize Phone ● Several Code ● Calculate Score CSV Handler: ● Load Data ● Record Data ● Remove Pending Logging System System Architectures Implementation module
Grittapob Chutitas	Admin Command Line	Admin Command Line CRUD Operation: ● Add/Edit/Delete
Phoo Phoo Thit	User Command Line	User Command Line Interface Display Design UX/UI

5.3 References

[1] https://docs.mikelopster.dev/c/c-dsa/intro - Learn basic data structures
[2] https://www.geeksforgeeks.org/implementation-of-hash-table-in-c-using-separate-chaining/? - Implementation of Hash Table in C/C++
[3] Data structures using C, 2nd Ed. by Thareja, Reema (2014).pdf - C Data Structures book
[4] https://www.geeksforgeeks.org/applications-of-hashing/ - Hashing Implementation
[5] https://www.geeksforgeeks.org/graph-representations-using-set-hash/ - Graph representation