**Recent Call Tracker using Queue**

**A PROJECT REPORT**

***Submitted by***

**SUYASH PUNDIR, KARRI SRAVANKUMAR, VIKASH KUMAR MISHRA**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**Lovely Professional University, Punjab**

**APPENDIX II**

**Lovely Professional University, Punjab**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Recent Call Tracker using Queue” is** the Bonafide work of **“SUYASH PUNDIR, KARRI SRAVANKUMAR, VIKASH KUMAR MISHRA”** who carried out the project work under my supervision.

**ABSTRACT**

This project presents a real-time call tracking system developed using the C++ programming language. The system simulates incoming phone calls with randomly generated caller IDs and call durations. It maintains a time-based queue to monitor all calls received in the last 60 seconds. The call data, including timestamps and durations, is logged into a text file (calls\_log.txt). The project utilizes features from the Standard Template Library (STL), real-time processing with chrono and thread, and file handling. The system demonstrates how queue data structures can be applied to build efficient, lightweight, real-time applications. This project has potential applications in call centres, emergency response systems, and server-side analytics.

**APPENDIX III**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **CHAPTER NAME** | **PAGE NO.** |
|  | **Title Page** | **i** |
|  | **Bonafide Certificate** | **ii** |
|  | **Abstract** | ... |
|  | **Table of Contents** | **iii** |
|  | **Abstract** | … |
|  | **Table of Contents** | . . . |
|  | **List of Tables** | . . . |
|  | **List of Figures** | … |
|  | **List of Symbols, Abbreviations and Nomenclature** | . . . |
| 1 | Chapter 1 | **iv** |
| 2 | Chapter 2 | **v** |
| 3 | ……………… | … |
|  | Appendices | .. |
|  | References | **ix** |

**CHAPTER 1: INTRODUCTION**

**1.1 Background**

Tracking and managing real-time events like phone calls is essential in many industries. Traditional databases or bulk analytics tools often fall short in time-critical scenarios. Thus, the need for a fast and efficient real-time tracker arises.

**1.2 Objective**

The objective of this project is to:

* Track all incoming calls within the last 60 seconds.
* Generate random caller IDs and durations.
* Log all call activity to a persistent file.

**1.3 Technologies Used**

* Programming Language: C++
* Libraries: <iostream>, <queue>, <chrono>, <thread>, <fstream>

**CHAPTER 2: SYSTEM DESIGN & IMPLEMENTATION**

**2.1 Architecture Overview**

* Data Structure: std::queue``e is used to store active calls.
* Timing: std::chrono::steady\_clock ensures time accuracy.
* Call Struct: Contains callerId, timestamp, and duration.
* Logger: Each call is logged into calls\_log.txt.

**2.2 Call Struct Definition**

**struct Call {**

**int callerId;**

**steady\_clock::time\_point timestamp;**

**int duration; // in seconds**

**};**

**2.3 Class: RealTimeCallTracker**

**class RealTimeCallTracker {**

**queue<Call> callQueue;**

**ofstream logFile;**

**const int TIME\_WINDOW = 60;**

**...**

**};**

**2.4 Methods**

* **receiveCall(int id, int duration): Adds call to queue, logs it, and cleans up old ones.**
* **getRecentCallCount(): Returns number of calls in the last 60 seconds.**
* **cleanupOldCalls(): Removes outdated calls.**
* **logCall(): Writes formatted data to calls\_log.txt.**

**CHAPTER 3: OUTPUT & SAMPLE LOG**

**3.1 Sample Console Output**

Call from ID #5321 | Duration: 6s | Delay: 2s

Calls in last 60 seconds: 2

**3.2 Sample calls\_log.txt**

Caller ID: 5321, Time: 2025-07-13 19:45:33, Duration: 6s

Caller ID: 6143, Time: 2025-07-13 19:45:37, Duration: 3s

**CHAPTER 4: CONCLUSION & FUTURE WORK**

**4.1 Conclusion**

This project illustrates how real-time systems can be implemented using basic C++ STL components. It provides a simple yet effective model for event tracking in a time window, complete with file logging.

**4.2 Future Enhancements**

* Add GUI for real-time display.
* Store logs in CSV/JSON or push to a server.
* Track per-caller stats and aggregate durations.
* Use std::deque for better time complexity

**REFERENCES**

Bjarne Stroustrup (2013), “The C++ Programming Language”, 4th Edition, Addison-Wesley.

cppreference.com – C++ Standard Library documentation.

cplusplus.com – C++ tutorials and STL usage.

GeeksforGeeks – C++ chrono and queue examples.