**PROJECT REPORT**

**Contact Management System**



Course: Data Structures & algorithms  
Program: BS Computer Science  
Instructor: Muhammad Umar Farooq

**Submitted by:**

* MINAL ABID - L1F23BSCS0160

**Contact Management System - Short Description**

**1. Introduction**

This report presents a detailed overview of the "Contact Management System" developed as a course project for the Data Structures subject. The purpose of this document is to provide a clear understanding of the project's objective, its real-world relevance, design methodology, and implementation using C++ with Object-Oriented Programming and data structure concepts.

**2. Project Objective**

The objective of this project is to design and implement a Contact Management System that allows users to store, retrieve, update, and delete contact records efficiently. It provides additional functionalities like undo/redo of actions and persistent storage through file handling.

**3. Real-World Significance**

Contact management is a fundamental functionality in many real-world applications such as mobile phones, email services, CRM software, and more. This system mimics a simplified yet functional version of such systems, emphasizing user operations, data consistency, and recoverability of actions.

**4. Why I Chose This Project**

I chose this project because it closely resembles real-life utility and allowed me to practically implement data structures like doubly linked lists, stacks, and queues. It was a great opportunity to strengthen my OOP skills, understand file operations in C++, and learn how to structure larger programs.

**5. Code Structure and Implementation**

The system is implemented using C++ and follows an object-oriented structure. Below are the primary components:

* **5.1 Data Structures and Algorithms Used**
  + Doubly Linked List: To store and manage the contact list in memory.
  + Stack: Used to implement undo/redo functionality.
  + Queue: To store operation history.
  + File Handling (fstream): To load and save contact data persistently.
* **5.2 Classes and Functions**

**➤ Class: Contact**

setName(string)  
 • setPhoneNumber(string)  
 • setEmail(string)  
 • setAddress(string)  
 • getName() const  
 • getPhoneNumber() const  
 • getEmail() const  
 • getAddress() const  
 • display() const  
 • enterdetails(name, phone, email, address)

**➤ Class: Node**

• Contains: Contact data, Node\* next, Node\* prev

**➤ Class: contactList**

addContact(name, phone, email, address)  
 • deleteContact(phone)  
 • deleteContactByName(name)  
 • updateContact(oldPhone, newName, newPhone, newEmail, newAddress)  
 • displayContacts()  
 • searchByName(name)  
 • searchByPhone(number)  
 • undoLastOperation()  
 • redoLastOperation()  
 • displayOperationHistory()  
 • isEmpty() const

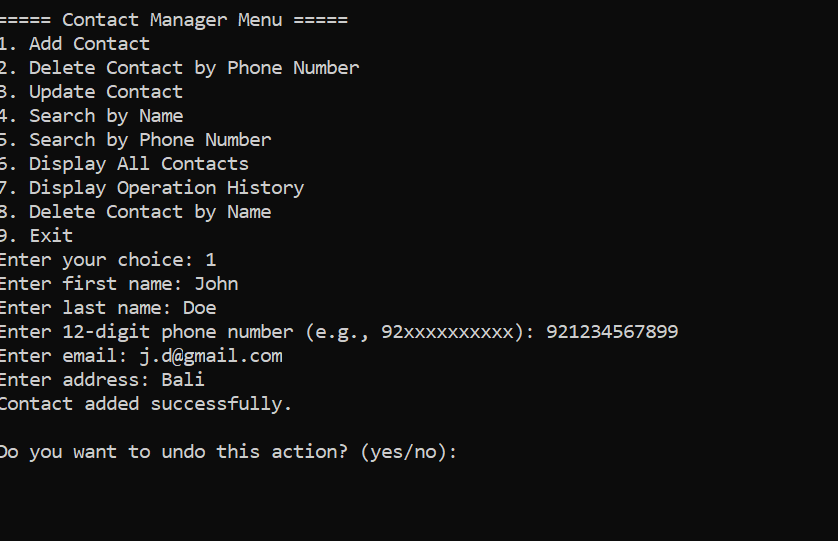
**➤ Helper Functions (outside class)**

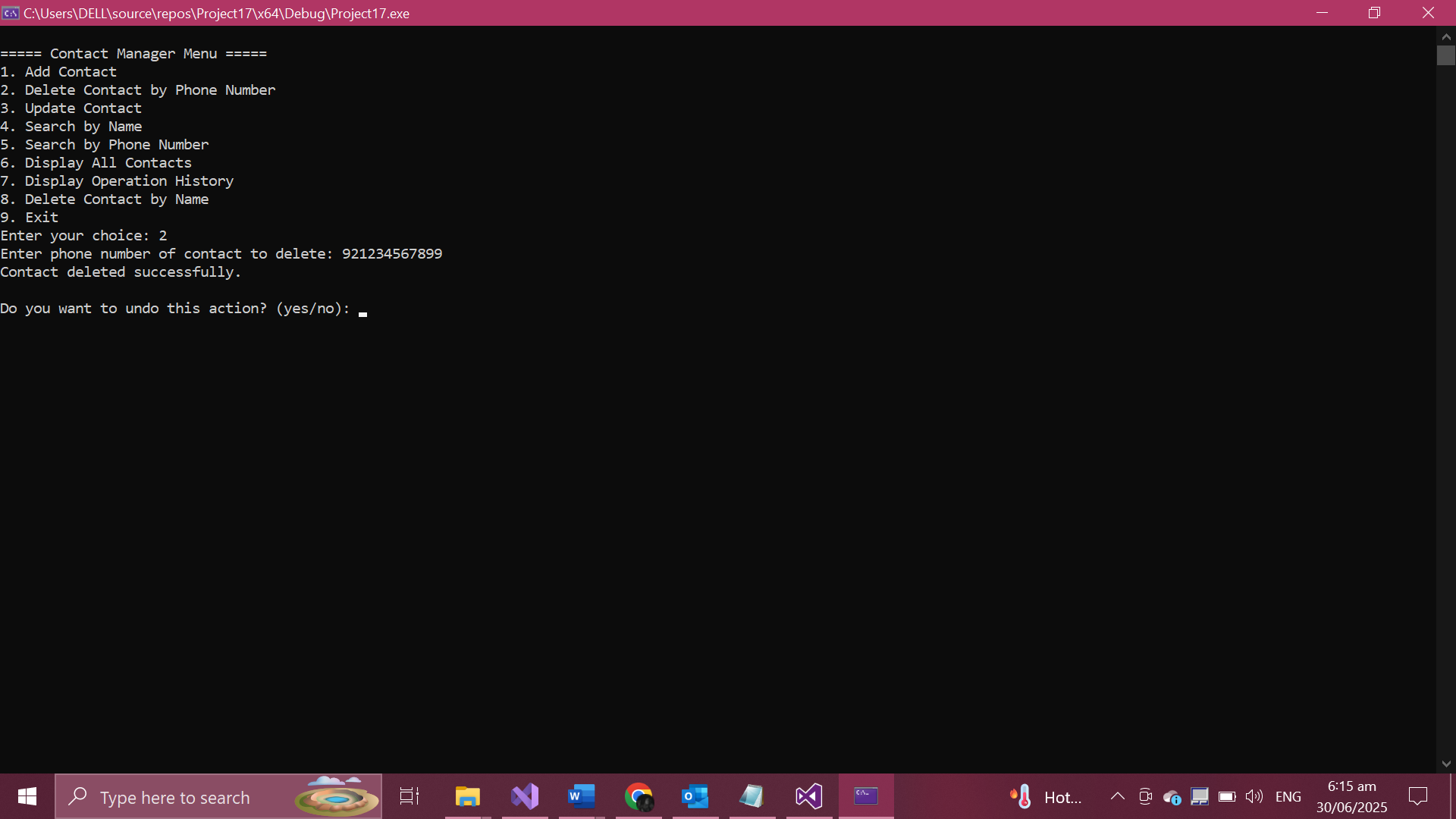
toLower(string)  
 • readContactFromFile(contactList&)  
 • updateFile(contactList)  
 • isValidPhoneNumber(string)

**6. Sample Outputs and Screenshots**

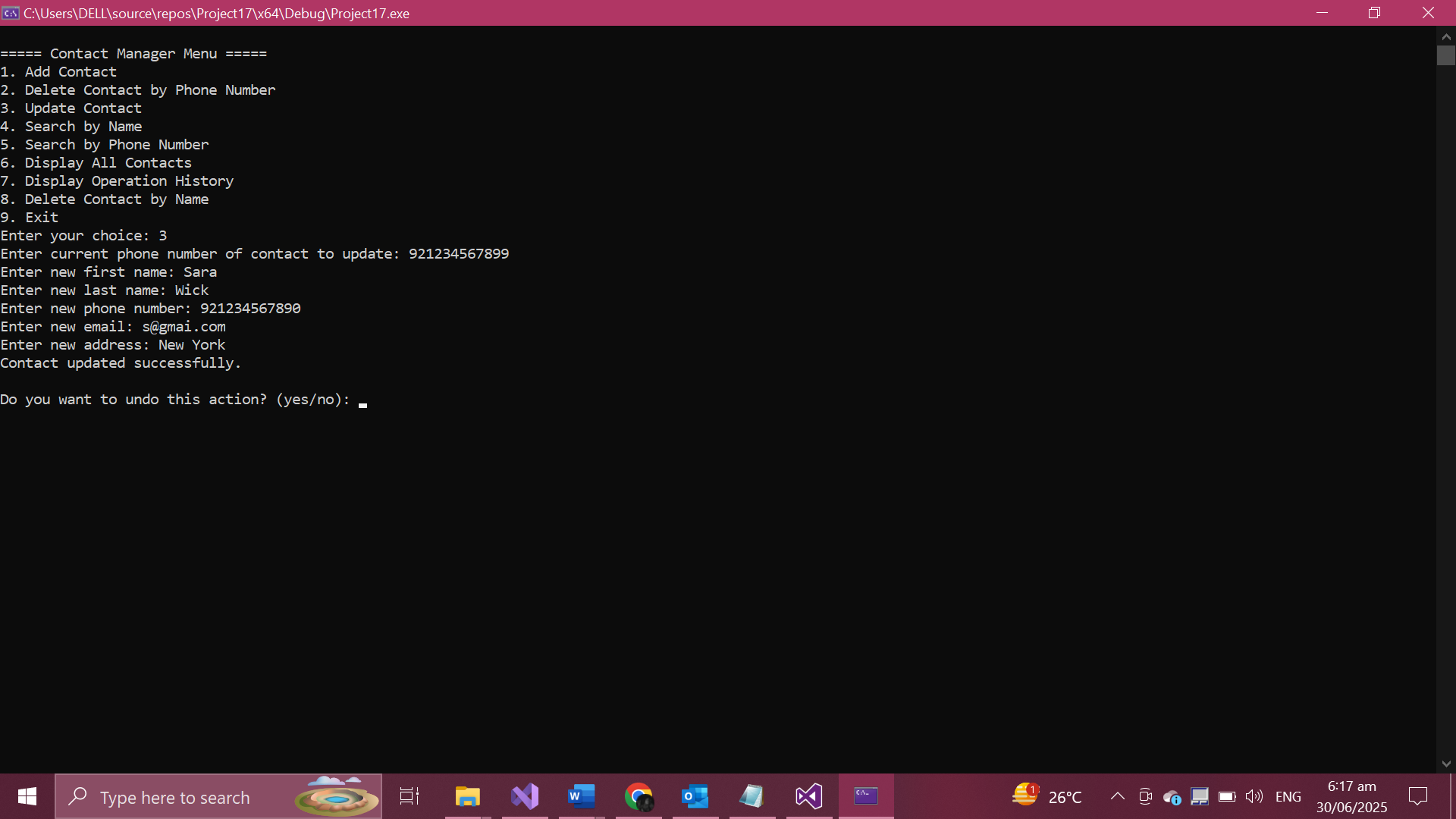
The following screenshots demonstrate the output of different functions executed from the menu-driven main interface.

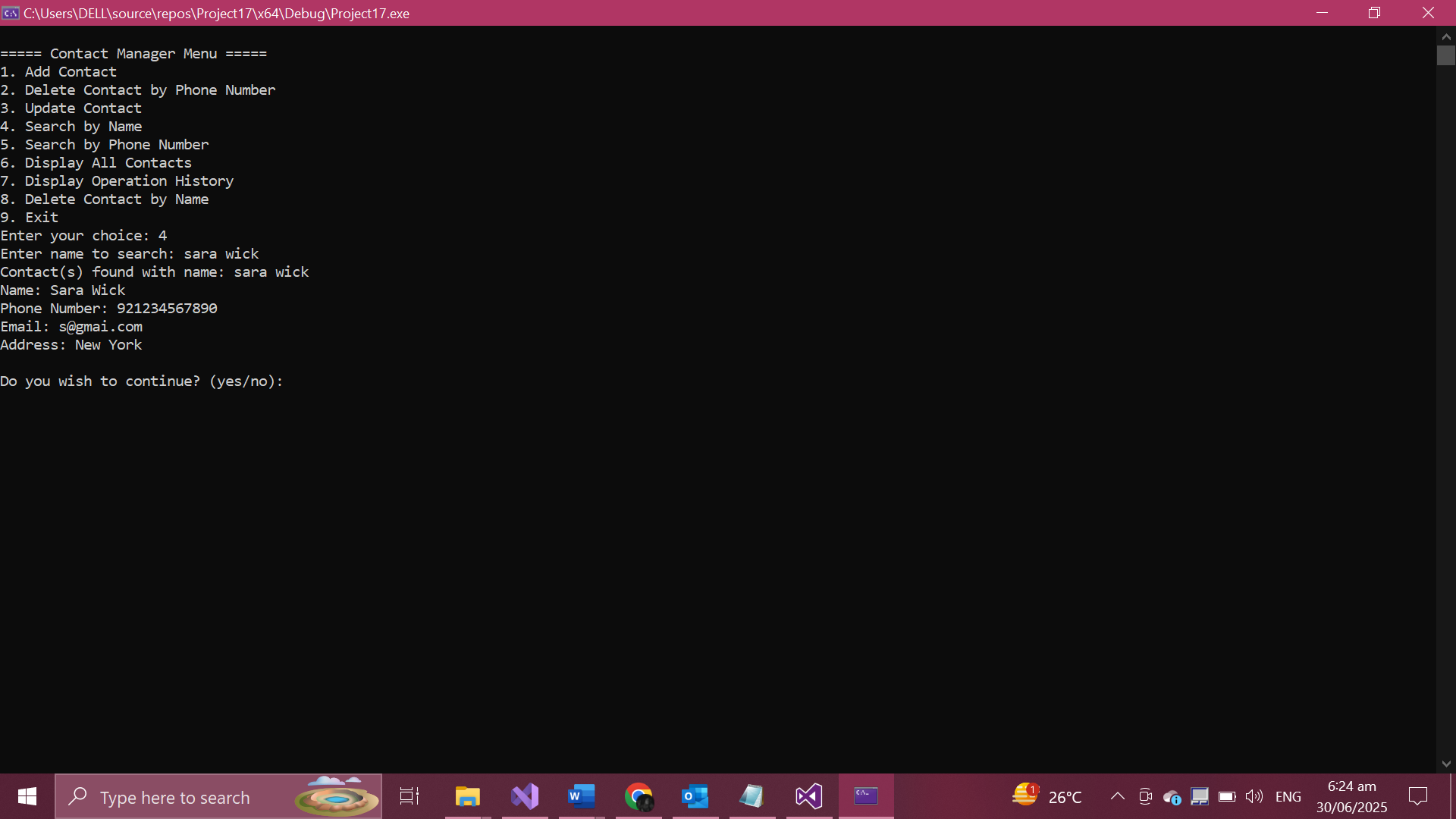
**• Add Contact**

**  
• Delete Contact**

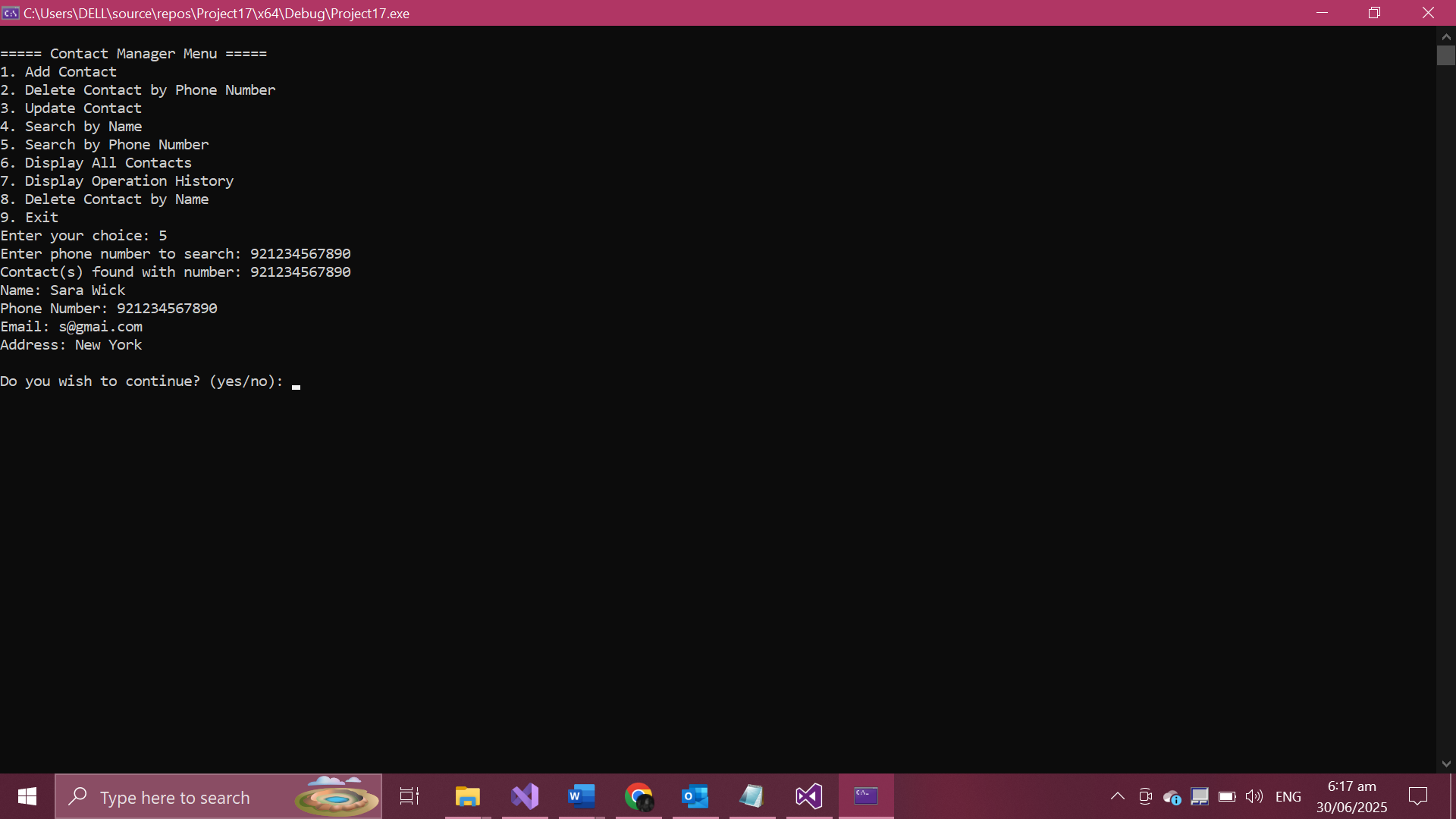
****

**• Update Contact**

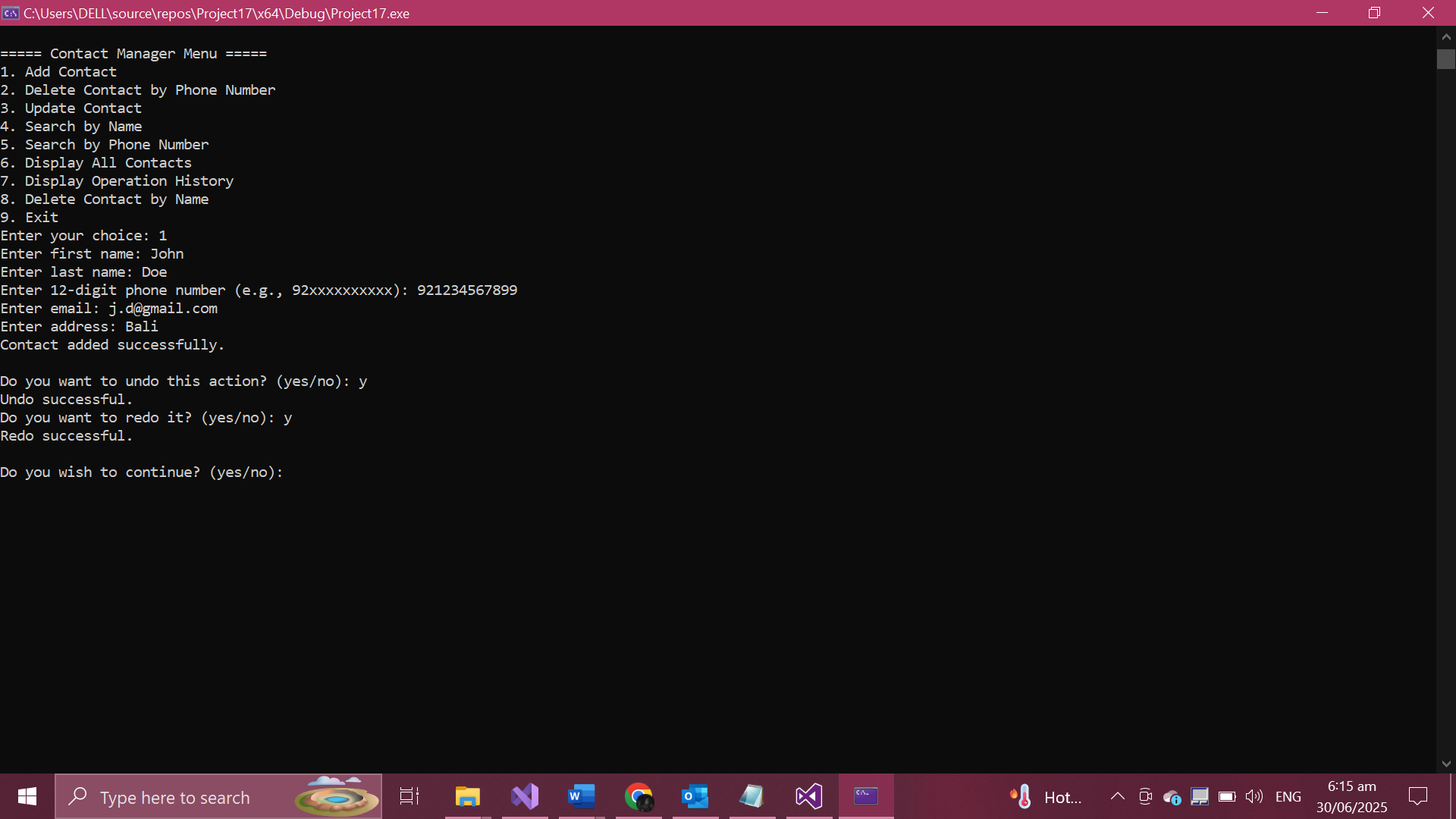
**  
• Search by Name**

****

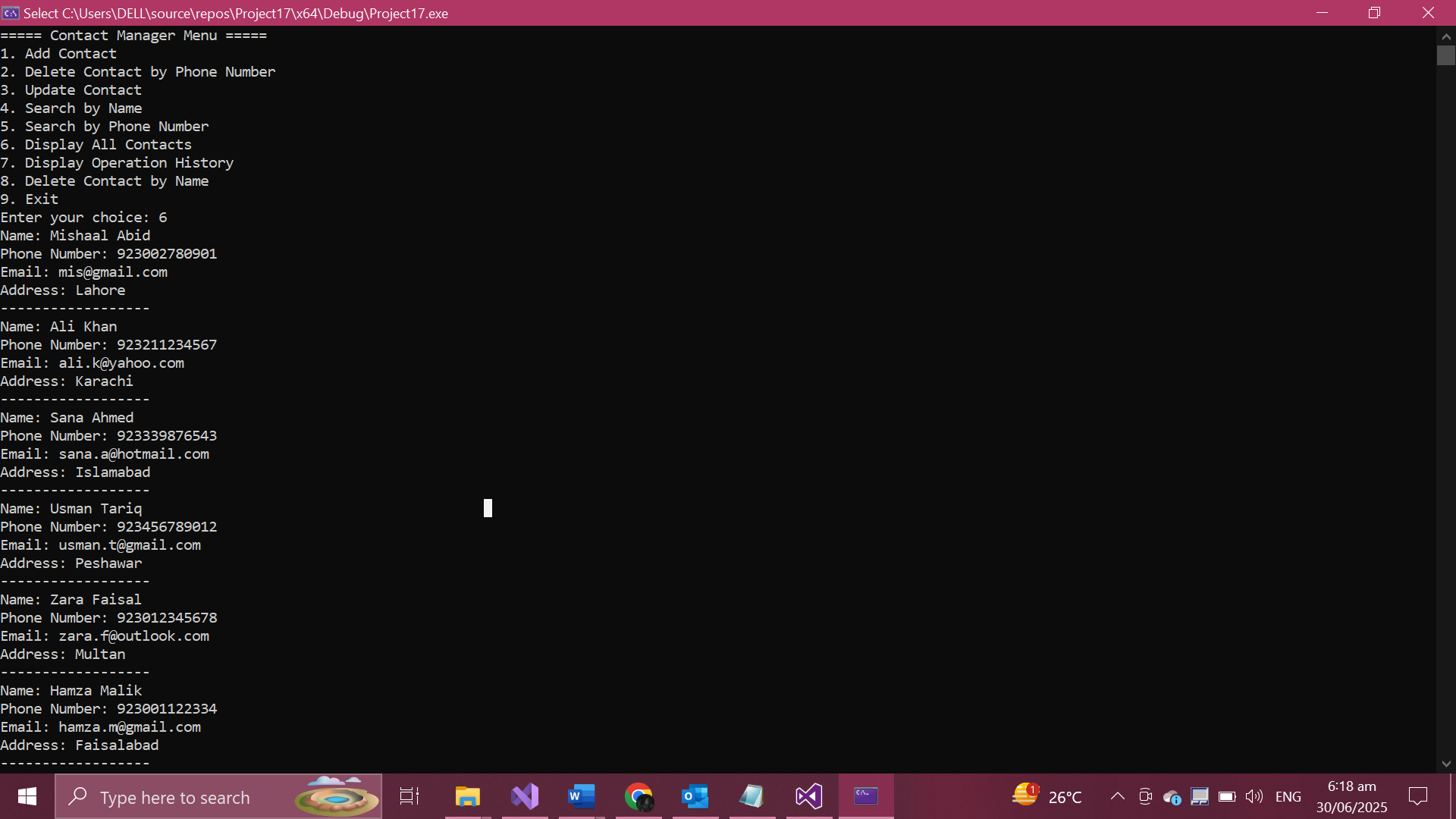
**• Search by Phone**

****

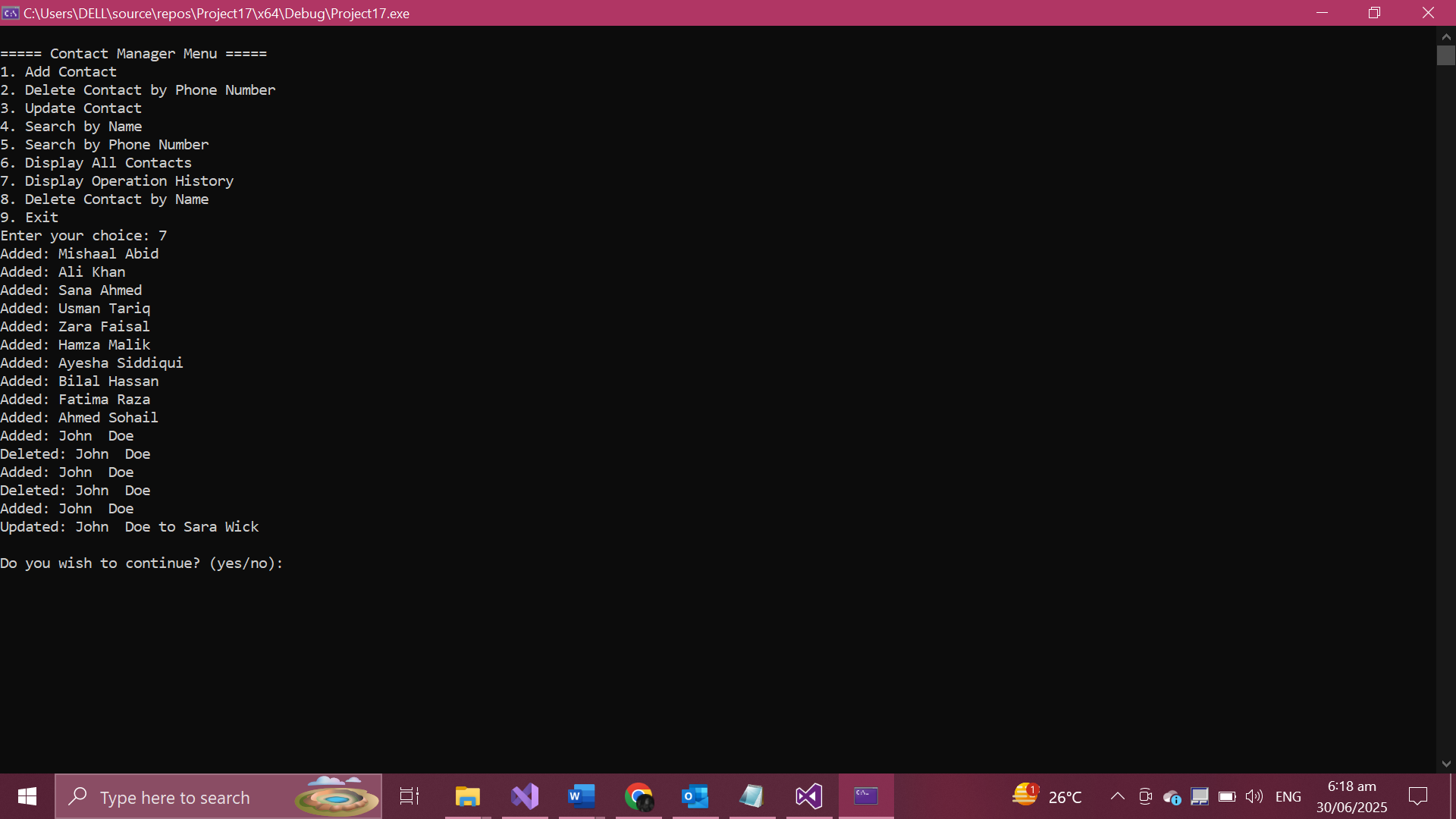
**• Undo/Redo**

****

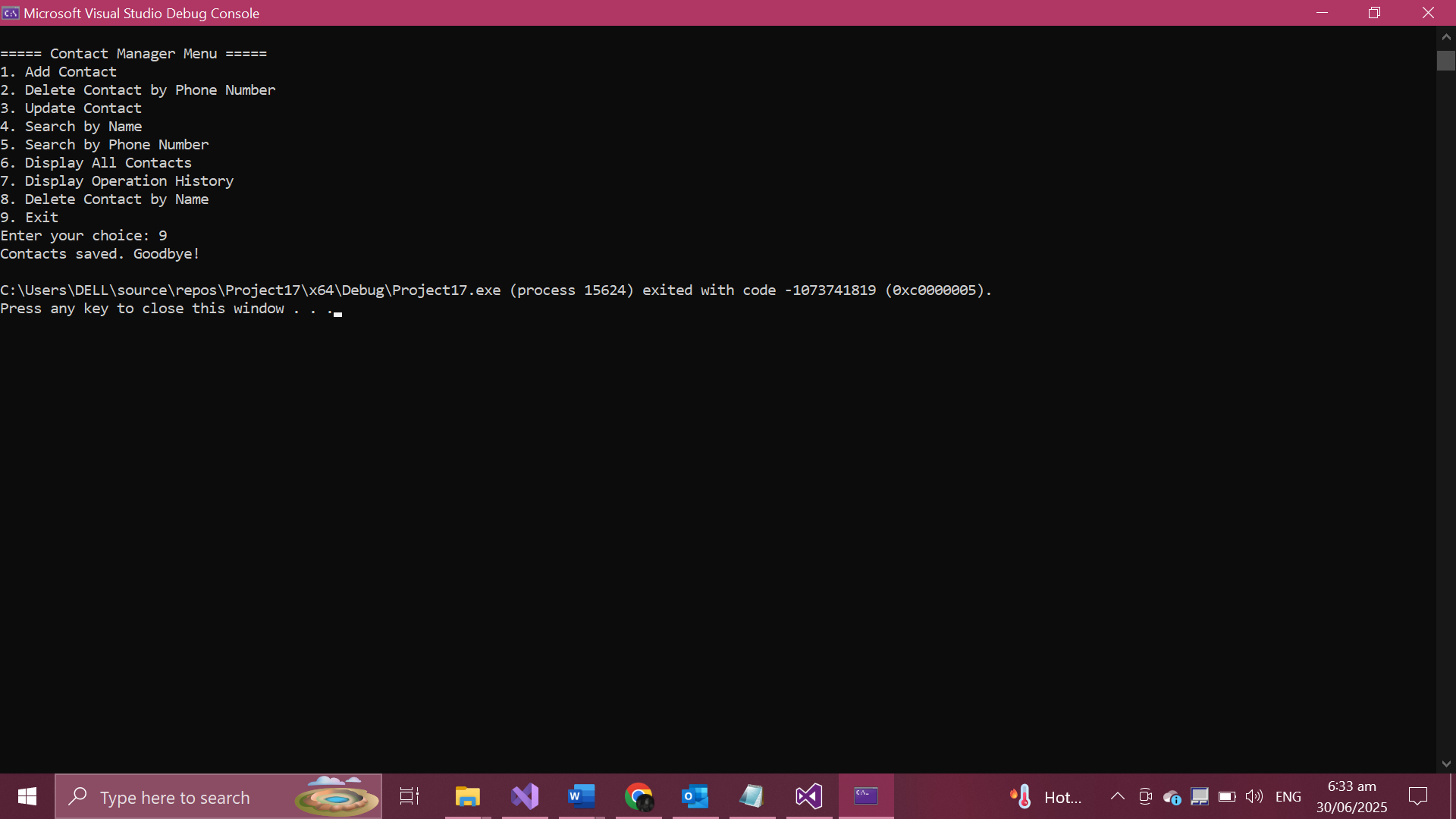
**• Display Contacts**

****

**• Operation History**

****

**• Exit/Save to File**



**7. Conclusion**

This project has successfully demonstrated the use of core data structures and object-oriented programming to solve a real-world problem. The Contact Management System is functional, user-friendly, and capable of handling operations with data consistency and recoverability. It was an enriching experience that deepened my understanding of C++ programming and structured software development**.**