Capgemini

[redacted]

High-Level Design & Low-Level Design

Index

1. Introduction

The introduction of the software requirement specification provides an overview of the entire software. The entire SRS with overview description purpose, scope, tools used and basic description. The aim of this document is to gather, analyse and give an in-depth insight into the complete Contacts Management application by defining the problem statement in detail. The detailed requirements of the contacts management application are provided in this document.

1.1 Intended Audience:

The target audience set for this project can be identified as an admin who manages multiple users and the users manage multiple contacts and automate the process of managing the contacts which includes personal and business contacts of users.

1.2 Project Purpose:

Contacts Management is a project that helps us understand the basic concepts of functions, file handling, and data structure. The automatic contacts management application will get users data as well as the contacts data of each user. It will create a list of personal and business contacts of all users. We can maintain a database of users and contacts by adding, updating, deleting and displaying the main menu of this contacts management application.

1.3 Key Project Objectives:

a. Allows the admin to add users

b. Allow the users to add their contacts

c. Updating the contacts information of a user

d. Deleting the contacts information of a user

e. Displaying personal contacts of all users

f. Displaying business contacts of all users

1.4 Project scope:

This project aims to create a contacts management application which takes the personal and business contacts information of multiple users such as first name, last name, mobile number, email-id in common for both type of contacts and for personal contacts, additional information like address, WhatsApp mobile number, birthdate, emergency contact and for business contacts, information like company name, designation and

company website as inputs and manages the information like updating and deleting the contacts of the respective users and as well as listing the personal and business contacts information separately. The application will have an admin who has the option to add users along with the default passwords which can be changed if required by the users, such that they can login and manage their contacts later.

2. Design Overview: -

2.1 Design Objectives:

1. Add users to user database
2. Add contacts information to contacts database
3. Update the contacts information of a user
4. Delete the contacts information of a user
5. Display personal contacts of all users
6. Display business contacts of all users

2.2 Design Alternative:

We have used standard template library (STL) to store contacts information i.e., contact_id, first name, last name, mobile number, email_id, address, WhatsApp mobile number, birth date, emergency contact, company name, designation and company website.

2.3 User Interface Paradigms:

The Contacts Management application provides an option to users to add, update, delete and display their personal and business contacts.

2.4 Validation:

- Contact_id should not be blank and duplication is not allowed and characters aren't allowed in the contact_id
- We check for the validity of the contacts information; it should not contain more than the required fields

3. SYSTEM ARCHITECTURE:

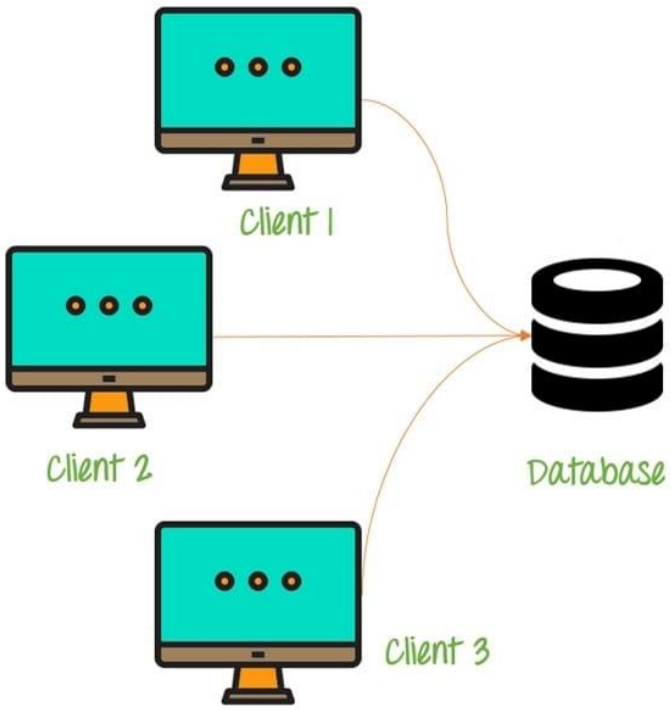
3.1. Database Architecture

The architecture used in this system comprises of the database architecture. It is a representation of the database management system design, wherein you can design, develop, implement and maintain the database. This architecture allows dividing the database into different components that can be independently modified, changed, replaced and altered as required for the system.

The database architecture is divided into three tiers namely,

- 1 - Tier Architecture
- 2 - Tier Architecture
- 3 - Tier Architecture

Our system is based on the Tier 1 model of the database architecture. In this type of model, the database is directly available to the user, the user can directly access the database and all of its contents. Which enables the user to directly interact and execute operations?



Some of the characteristics of Database Architecture are:

Self-Describing Nature of a Database System:

- One of the most fundamental characteristics of the database approach is that the database system contains not only the database itself but also an entire definition or description of the database structure and constraints also known as metadata of the database.

Isolation between Data, Programs and Data Abstraction:

- In a traditional file processing system, the structure of database knowledge files is embedded within the application programs, so any changes to the structure of a file may require changing all programs that access that file.

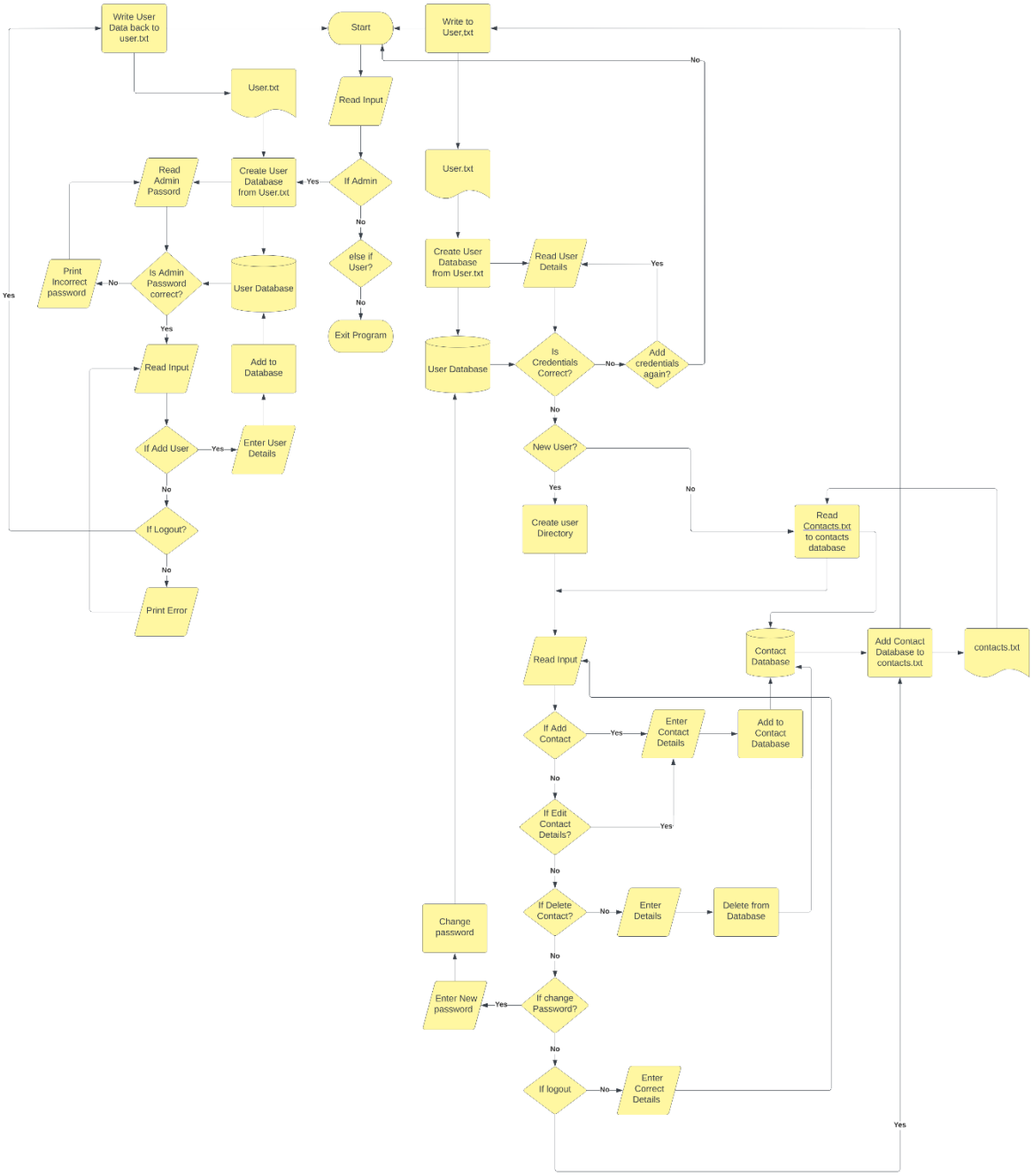
Support for Multiple Views of the Data:

- A database sometimes has many users, each of whom may require a special perspective or view of the database.

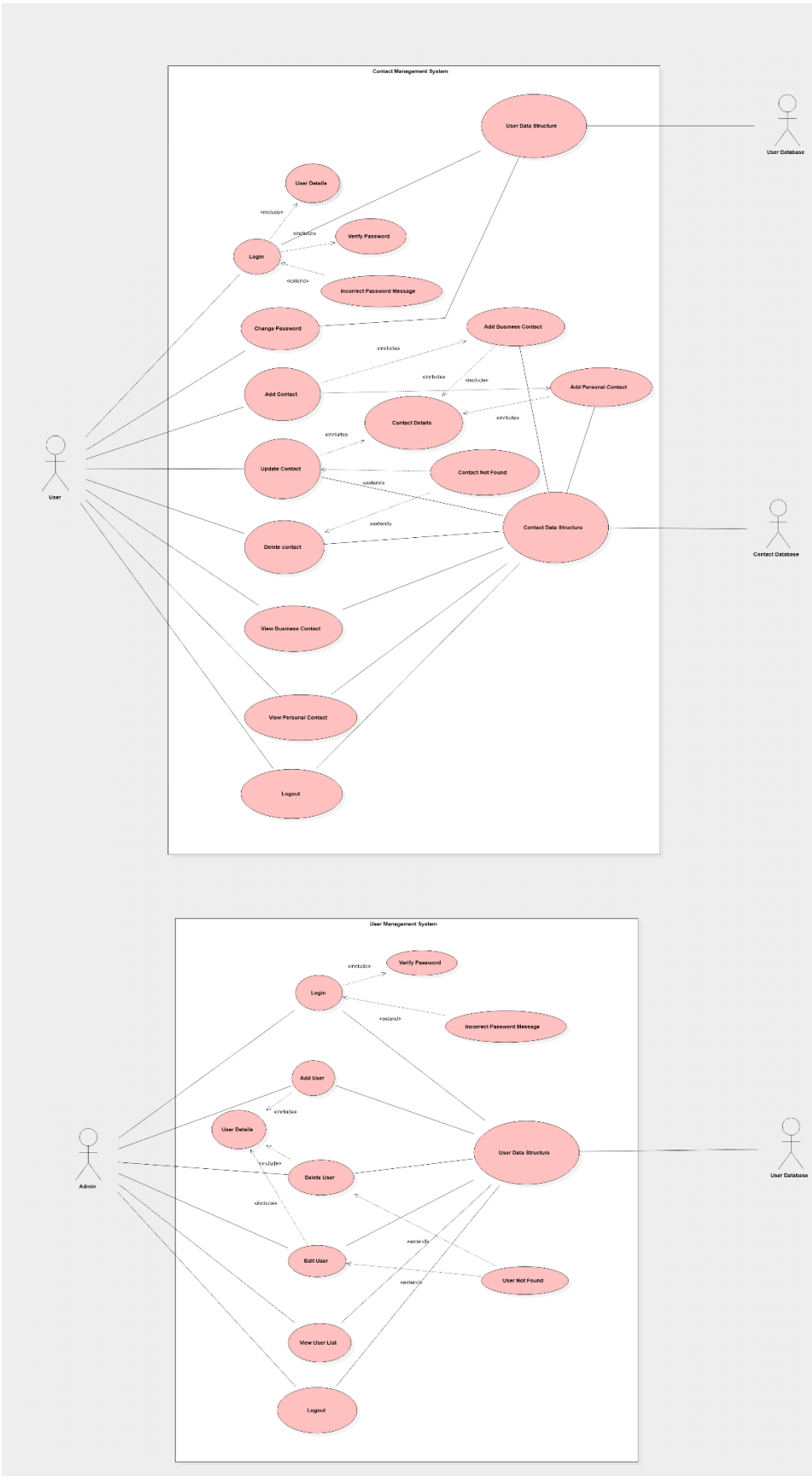
Sharing of knowledge and Multi-user Transaction Processing:

- A multi-user DBMS, as its name implies, must allow multiple users to access the database at an equivalent time or concurrently.

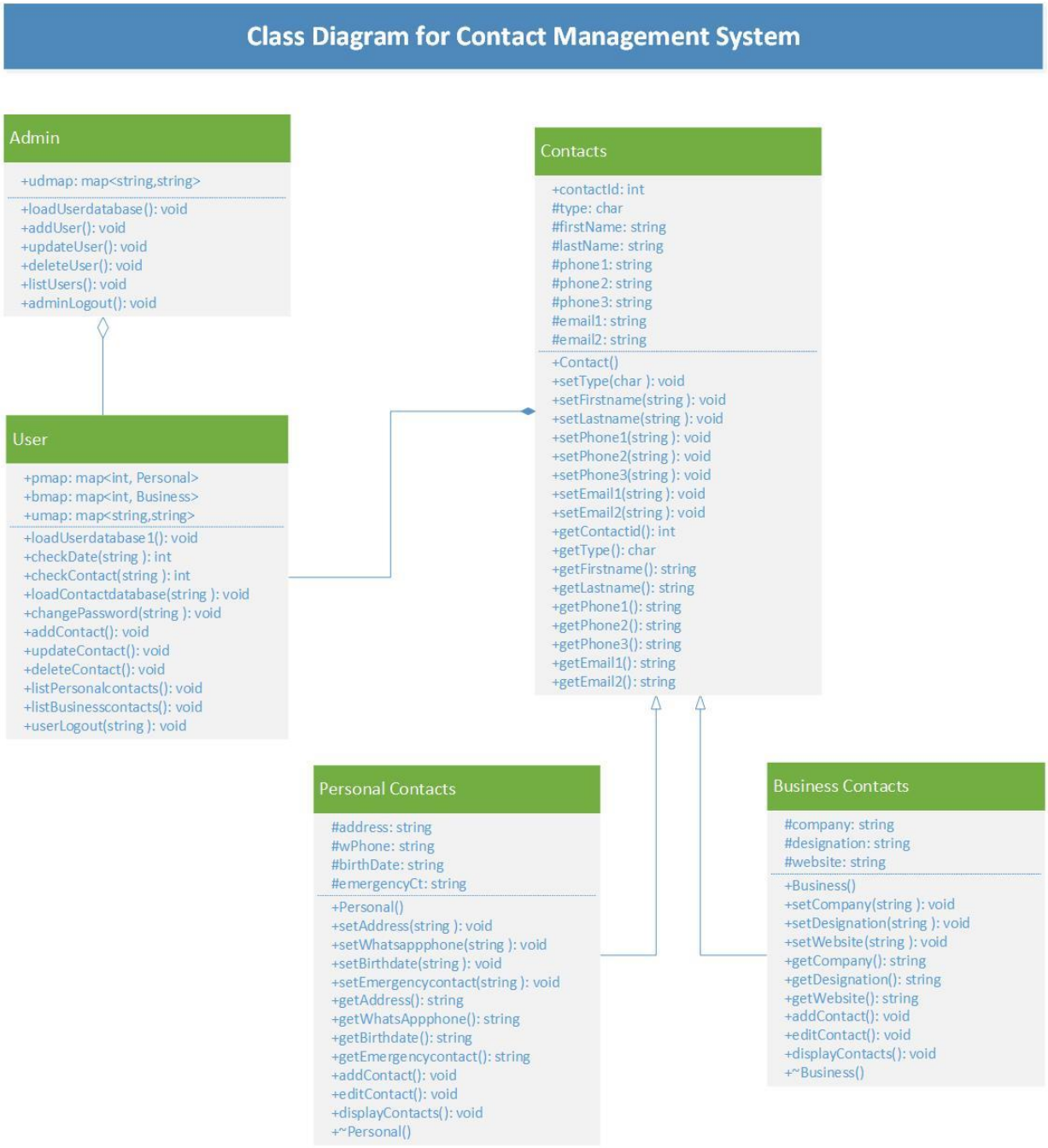
4. DETAILED SYSTEM DESIGN:



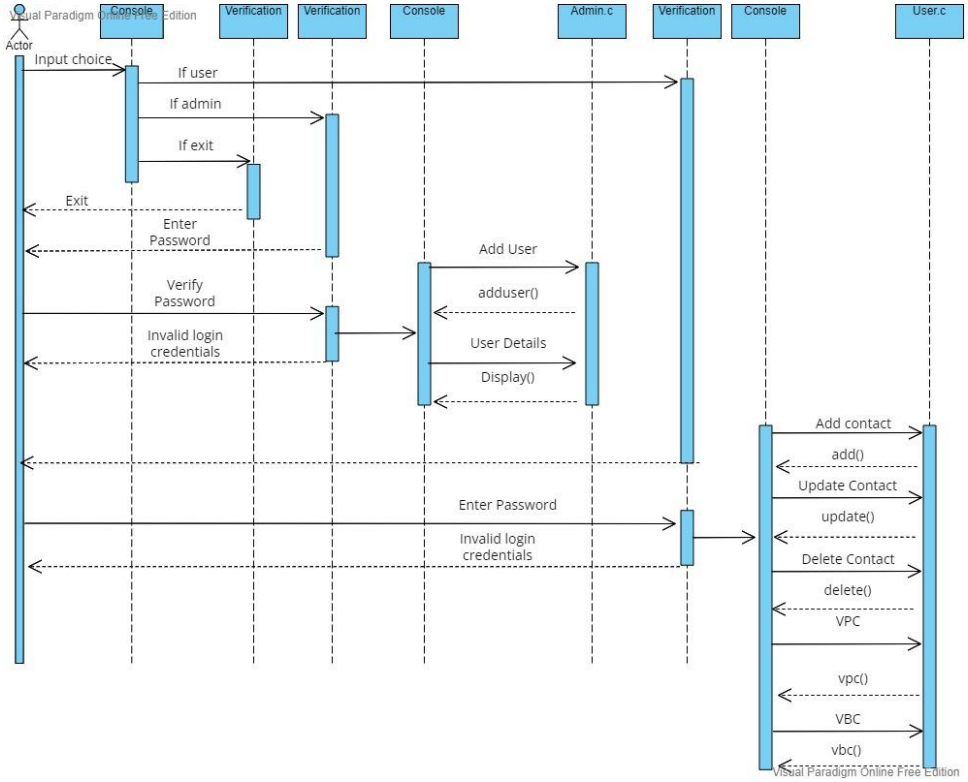
4.1 Flow Chart of the application



4.2 Usecase Diagram



4.3 Class diagram



4.4 Sequence Diagram

5. TOOLS REPORT

5.1 Valgrind

```
==126333== Memcheck, a memory error detector
==126333== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==126333== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==126333== Command: ./contactsManagement.exe
==126333==
admin
Admin
4
5
y
samson
samson
4
^C==126333==
==126333== Process terminating with default action of signal 2 (SIGINT)
==126333==    at 0x4B1F84E: read (read.c:26)
==126333==    by 0x4AB1899: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:517)
==126333==    by 0x4AB2B01: _IO_default_uflow (genops.c:362)
==126333==    by 0x494996C: __gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char> >::underflow() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==126333==    by 0x495754E: std::istream::sentry::sentry(std::istream&, bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==126333==    by 0x495777E: std::istream::operator>>(int&) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==126333==    by 0x10C75F: deleteContact() (in /home/cg8luser7/project_mod/bin/contactsManagement.exe)
==126333==    by 0x10A8B6: user(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >) (in /home/cg8luser7/project_mod/bin/contactsManagement.
exe)
==126333==    by 0x10A543: main (in /home/cg8luser7/project_mod/bin/contactsManagement.exe)
==126333==
==126333== HEAP SUMMARY:
==126333==     in use at exit: 480 bytes in 5 blocks
==126333==   total heap usage: 42 allocs, 37 frees, 131,189 bytes allocated
==126333==
==126333== LEAK SUMMARY:
==126333==    definitely lost: 0 bytes in 0 blocks
==126333==    indirectly lost: 0 bytes in 0 blocks
==126333==      possibly lost: 0 bytes in 0 blocks
==126333==    still reachable: 480 bytes in 5 blocks
==126333==         suppressed: 0 bytes in 0 blocks
==126333== Reachable blocks (those to which a pointer was found) are not shown.
==126333== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==126333==
==126333== For lists of detected and suppressed errors, rerun with: -s
==126333== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

6. Requirements Traceability Matrix (RTM)

Req	Design Mapping	Code Mapping	UT Mapping	IT Mapping
CM01		menu-driven contacts management application to manage contacts of multiple users	Test_Case-1,Test_Case-2,Test_Case-3,Test_Case-4,Test_Case-5,Test_Case-6,Test_Case-7,Test_Case-8,Test_Case-9,Test_Case-10	INPUT_002
CM02		common attributes for both personal and business contacts		
CM03		specified attributes for both personal and business contacts		
CM04	3.1.1	addUser(), deleteUser(), editUser(), listUsers()		INPUT_001
CM05	3.1.2	addPesronalcontact(), addBusinesscontact(), updateContact(), deleteContact(), listPersonalcontacts(), listBusinesscontacts()	Test_Case-11,Test_Case-12,Test_Case-13,Test_Case-14,Test_Case-15,Test_Case-16,Test_Case-17,Test_Case-18,Test_Case-19,Test_Case-20,Test_Case-21,Test_Case-22,Test_Case-23,Test_Case-24,Test_Case-25,Test_Case-26,Test_Case-27,Test_Case-28,Test_Case-29,Test_Case-30	INPUT_004,INPUT_005
CM06	3.1.3	loadContactdatabase()		
CM07	3.1.4	checkContact(), checkDate()	Test_Case-31,Test_Case-32,Test_Case-33	INPUT_003
CM08	3.1.3	loadContactdatabase()		
CM09		password encryption and decryption		