VIDYAVARDHAKA COLLEGE OF ENGINEERING

(Autonomous, Affiliated to VTU)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

OPEN ENDED EXPERIMENT ON

"CONTACT MANAGMENT SYSTEM"

Submitted in partial fulfillment of the requirement for the completion of 4th semester of

'BACHELOR OF ENGINEERING'

Submitted by:

Likhith MR	4VV23CS118
Loganth	4VV23CS120

Under the guidance of:

Dr. Vedavathi N

Associate Professor

Dept of CSE, VVCE

COMPUTER SCIENCE AND ENGINEERING

2024-25

TABLE OF CONTENTS

1. INTRODUCTION

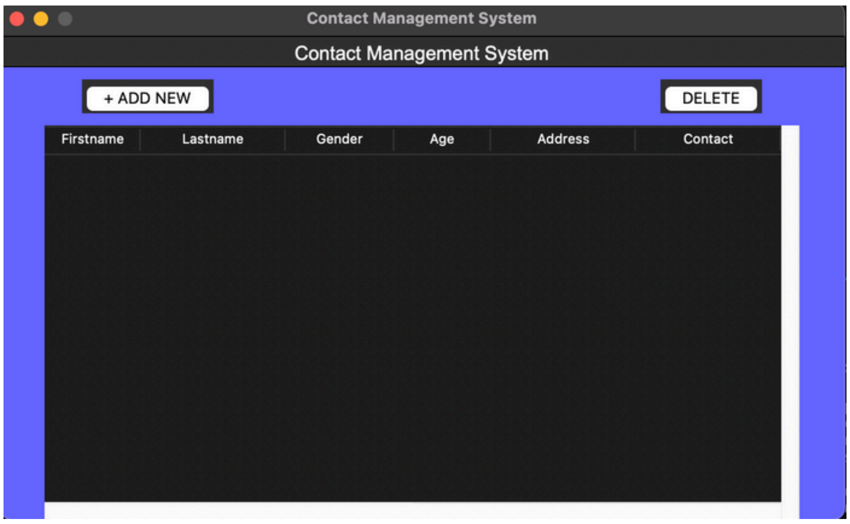
Introduction:

Contact Management System Application
This report details the development and functionality of a simple yet effective Contact Management System application built using the Python programming language and the Tkinter library for its graphical user interface (GUI). In today's digital age, efficient management of contacts is crucial for both personal and professional endeavors. This application aims to provide a user-friendly interface for storing, organizing, and managing contact information in a structured and accessible manner.

The core functionality of this system includes the ability to:

- Store new contact details: Users can easily input and save essential information such as first name, last name, gender, age, address, and contact number.
- View existing contacts: All stored contacts are displayed in a clear, tabular format, allowing for quick browsing and identification.
- Update contact information: Users can modify the details of existing contacts, ensuring the information remains accurate and up-to-date.
- Delete contacts: The system provides a straightforward way to remove unwanted or outdated contact entries.

The application leverages the SQLite database to persistently store contact data, ensuring that information is retained even after the application is closed. The Tkinter library facilitates the creation of an intuitive and interactive GUI, making the application accessible to users with varying levels of technical expertise.
This report will further elaborate on the application's architecture, key features, implementation details, and potential areas for future enhancement. The development of this Contact Management System serves as a practical demonstration of GUI development using Tkinter and database interaction with SQLite, highlighting their capabilities in creating functional desktop applications for data management.



Problem Statement

PProblem Statement: Contact Management System Application

The primary objectives of this Contact Management System project are as follows:

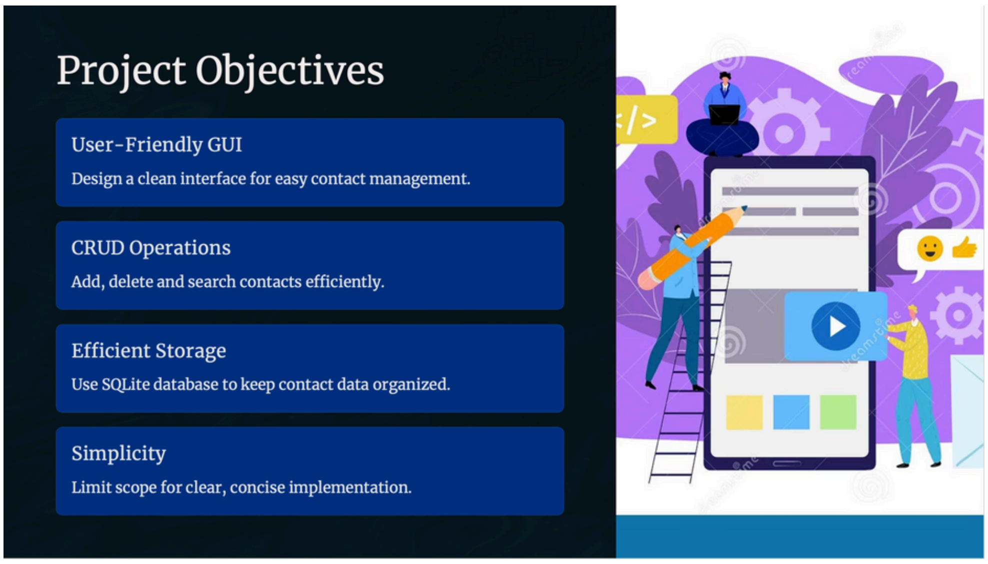
Develop a User-Friendly Graphical Interface: To create an intuitive and easy-to-navigate GUI using the Tkinter library, enabling users to interact with the contact management functionalities efficiently.

- Implement Core Contact Management Features: To provide essential functionalities for managing contact information, including:
    - Adding New Contacts: Allowing users to input and store key details such as first name, last name, gender, age, address, and contact number.
    - Displaying Existing Contacts: Presenting stored contact information in a clear and organized tabular format for easy viewing.
    - Updating Contact Details: Enabling users to modify and save changes to the information of existing contacts.
    - Deleting Contacts: Providing a mechanism for users to remove contact entries from the system.

Utilize a Local Database for Data Persistence: To integrate an SQLite database for the persistent storage of contact information, ensuring data is retained between application sessions

Ensure Data Integrity and Basic Validation: To implement basic checks to ensure that essential fields are filled during contact creation and modification, providing feedback to the user for incomplete entries.

Provide a Responsive and Functional Application: To develop a stable and responsive application that performs the intended contact management tasks reliably and efficiently.



Project Objectives

User-Friendly GUI
Design a clean interface for easy contact management.

CRUD Operations
Add, delete and search contacts efficiently.

Efficient Storage
Use SQLite database to keep contact data organized.

Simplicity
Limit scope for clear, concise implementation.

Key Features and Functionality

Key Features and Functionality of the Contact Management System

The Contact Management System application offers the following key features and functionalities:

- Add New Contact:
  - Provides a dedicated window with input fields for capturing essential contact details: First Name, Last Name, Gender (via radio buttons), Age, Address, and Contact Number.
  - Includes a "Save" button to store the entered information into the SQLite database and update the main contact list.
  - Implements basic validation to ensure all required fields are filled before saving a new contact.
- View Existing Contacts:
  - Displays all stored contacts in a tabular format using the Tkinter Treeview widget.
  - Presents contact information across several columns: MemberID (internal identifier), First Name, Last Name, Gender, Age, Address, and Contact Number.
  - Orders the displayed contacts alphabetically by Last Name for easy browsing.
- Update Contact Information:
  - Allows users to select a contact from the main list by double-clicking on a row.
  - Opens a separate "Updating Contacts" window pre-populated with the selected contact's current information.
  - Provides editable input fields for all contact details.
  - Includes an "Update" button to save the modified information to the database and refresh the main contact list.
- Delete Contact:
  - Enables users to select one or more contacts from the main list.
  - Features a "DELETE" button that, upon confirmation, removes the selected contact(s) from both the Treeview display and the underlying SQLite database.
  - Includes a confirmation dialog to prevent accidental deletion of contacts.
- Data Persistence:
  - Utilizes an SQLite database ("contact.db") to store all contact information locally on the user's system.
  - Ensures that added, updated, and deleted contacts are permanently saved and accessible across application sessions.
- Organized Layout:
  - Employs Tkinter frames to structure the main window and sub-windows logically.
  - Presents information and interactive elements (buttons, input fields, table) in a clear and user-friendly manner.
- Visual Feedback:
  - Provides warning messages to the user if required fields are not completed during the addition or modification of contacts.
  - Uses a confirmation dialog before deleting contacts to prevent unintentional data loss.

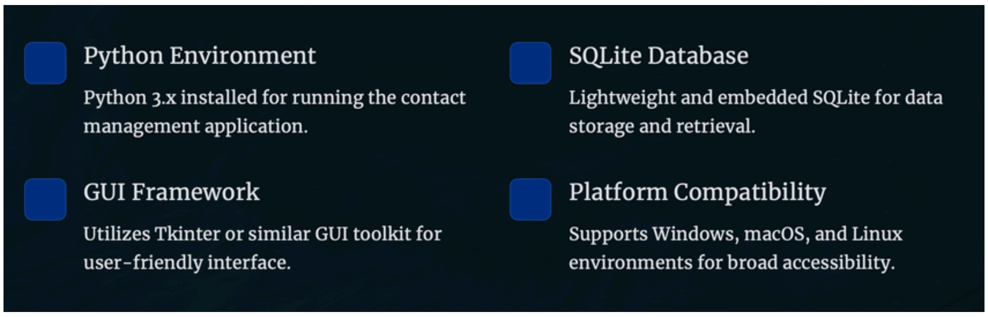
In summary, the application provides a fundamental set of features for managing a list of contacts, focusing on ease of use and data reliability through local database storage.

Project Working Code

```
from tkinter import *
import sqlite3
import tkinter.ttk as ttk
import tkinter.messagebox as tkMessageBox

#DEVELOPED BY Mark Arvin
root = Tk()
root.title("Contact Management System")
width = 700
height = 400
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)
root.config(bg="#6666ff")

#============================VARIABLES===================================
FIRSTNAME = StringVar()
LASTNAME = StringVar()
GENDER = StringVar()
AGE = StringVar()
ADDRESS = StringVar()
CONTACT = StringVar()



#============================METHODS=====================================

def Database():
 conn = sqlite3.connect("contact.db")
 cursor = conn.cursor()
 cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER NOT NULL PRIMARY
KEY AUTOINCREMENT, firstname TEXT, lastname TEXT, gender TEXT, age TEXT, address TEXT,
contact TEXT)")
 cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
 fetch = cursor.fetchall()
 for data in fetch:
 tree.insert('', 'end', values=(data))
 cursor.close()
 conn.close()
```

```
def SubmitData():
 if FIRSTNAME.get() == "" or LASTNAME.get() == "" or GENDER.get() == "" or AGE.get() == ""
or ADDRESS.get() == "" or CONTACT.get() == "":
   result = tkMessageBox.showwarning('', 'Please Complete The Required Field',
icon="warning")
 else:
 tree.delete(*tree.get_children())
 conn = sqlite3.connect("contact.db")
 cursor = conn.cursor()
 cursor.execute("INSERT INTO `member` (firstname, lastname, gender, age, address, contact)
VALUES(?, ?, ?, ?, ?, ?)", (str(FIRSTNAME.get()), str(LASTNAME.get()), str(GENDER.get()),
int(AGE.get()), str(ADDRESS.get()), str(CONTACT.get())))
 conn.commit()
 cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
 fetch = cursor.fetchall()
 for data in fetch:
 tree.insert('', 'end', values=(data))
 cursor.close()
 conn.close()
 FIRSTNAME.set("")
 LASTNAME.set("")
 GENDER.set("")
 AGE.set("")
 ADDRESS.set("")
 CONTACT.set("")

def UpdateData():
 if GENDER.get() == "":
   result = tkMessageBox.showwarning('', 'Please Complete The Required Field',
icon="warning")
 else:
 tree.delete(*tree.get_children())
 conn = sqlite3.connect("contact.db")
 cursor = conn.cursor()
 cursor.execute("UPDATE `member` SET `firstname` = ?, `lastname` = ?, `gender` =?, `age` =
?, `address` = ?, `contact` = ? WHERE `mem_id` = ?", (str(FIRSTNAME.get()),
str(LASTNAME.get()), str(GENDER.get()), str(AGE.get()), str(ADDRESS.get()),
str(CONTACT.get()), int(mem_id)))
 conn.commit()
 cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
 fetch = cursor.fetchall()
 for data in fetch:
 tree.insert('', 'end', values=(data))
 cursor.close()
 conn.close()
 FIRSTNAME.set("")
 LASTNAME.set("")
 GENDER.set("")
 AGE.set("")
 ADDRESS.set("")
 CONTACT.set("")
```

System Requirements



Python Environment

Python 3.x installed for running the contact management application.

SQLite Database

Lightweight and embedded SQLite for data storage and retrieval.

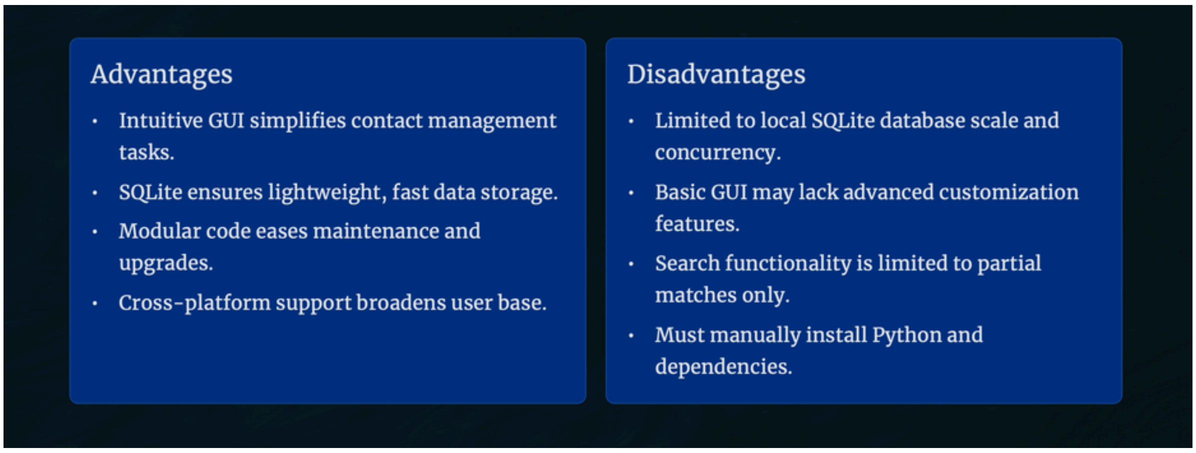
GUI Framework

Utilizes Tkinter or similar GUI toolkit for user-friendly interface.

Platform Compatibility

Supports Windows, macOS, and Linux environments for broad accessibility.

Advantages and Disadvantages



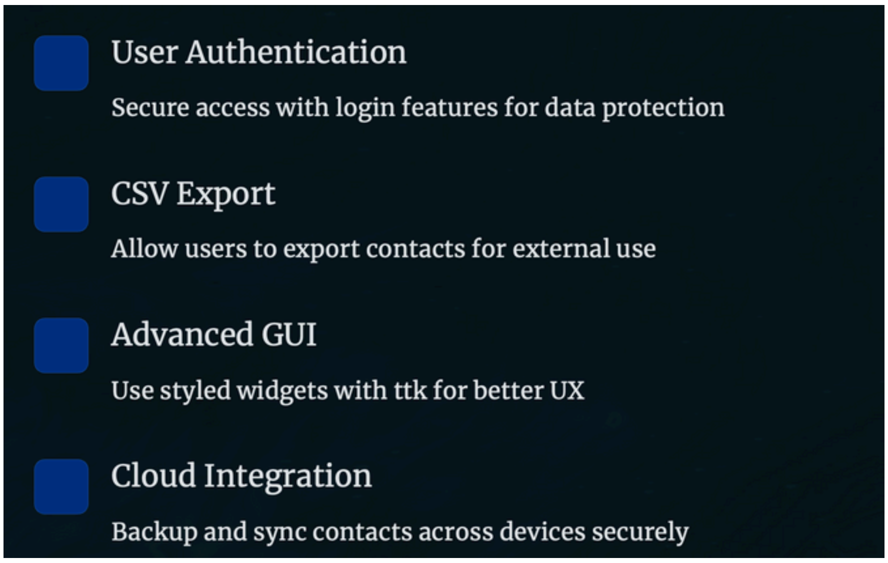
Advantages

- Intuitive GUI simplifies contact management tasks.
- SQLite ensures lightweight, fast data storage.
- Modular code eases maintenance and upgrades.
- Cross-platform support broadens user base.

Disadvantages

- Limited to local SQLite database scale and concurrency.
- Basic GUI may lack advanced customization features.
- Search functionality is limited to partial matches only.
- Must manually install Python and dependencies.

Future Enhancements and Scope



User Authentication

Secure access with login features for data protection

CSV Export

Allow users to export contacts for external use

Advanced GUI

Use styled widgets with ttk for better UX

Cloud Integration

Backup and sync contacts across devices securely

Implementation and Working of the Application

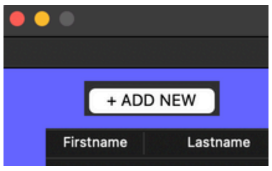
Contact Management System Report (Concise)

Introduction: This report outlines a simple Contact Management System application developed using Python and Tkinter for the GUI, with SQLite for local data storage. The application enables users to add, view, update, and delete contact information efficiently through a user-friendly interface.

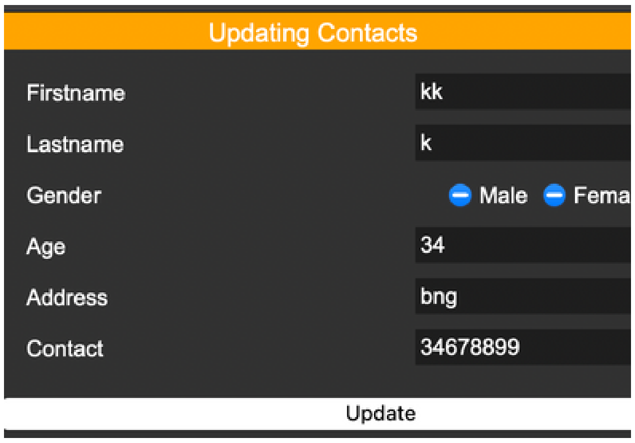
Project Objectives: The primary goals were to create an intuitive GUI for basic contact management features (add, view, update, delete) and to utilize a local SQLite database for persistent data storage.

Key Features and Functionality:

- Add New Contact: Input fields for name, gender, age, address, and contact; saves to database and updates the contact list.



- View Existing Contacts: Displays contacts in a sortable table (Treeview) upon application start and after data changes.
- Update Contact Information: Allows modification of existing contact details via a separate window, updates the database and contact list.



Updating Contacts

Firstname	kk
Lastname	k
Gender	Male Fema
Age	34
Address	bng
Contact	34678899

- Delete Contact: Enables removal of selected contacts from both the display and the database with confirmation.
- Data Persistence: Uses a local SQLite database ("contact.db") to store all contact information persistently.

System Design Architecture and Requirements: A two-tier architecture is employed: a Tkinter-based Presentation Tier for the GUI and a Data Tier utilizing SQLite for local storage. Key functional requirements include adding, viewing, updating, and deleting contacts, along with data persistence and basic input validation. Non-functional requirements emphasize usability, performance, reliability, and portability.

Advantages: User-friendly GUI, basic core functionality, local data storage, data persistence, organized data display, basic validation, and delete confirmation.

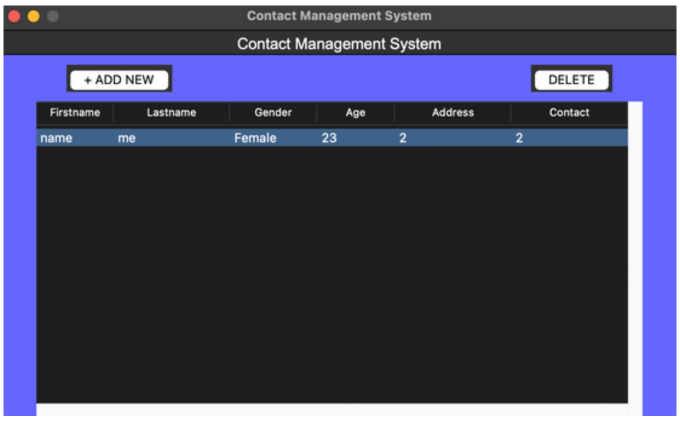
Disadvantages: Limited advanced features (search, filtering, import/export), single-user system, basic error handling, limited security, and no built-in backup/recovery.

Working of the Application: User interactions in the GUI trigger Python functions that perform SQL operations (INSERT, SELECT, UPDATE, DELETE) on the SQLite database. The Treeview widget displays data fetched from the database. Adding and updating open new windows for data input/modification. Deleting removes records from both the GUI display and the database.
Conclusion: The Contact Management System provides a functional solution for basic contact management with a simple interface and local data storage. While effective for individual use, it lacks advanced features found in more comprehensive systems, presenting opportunities for future development

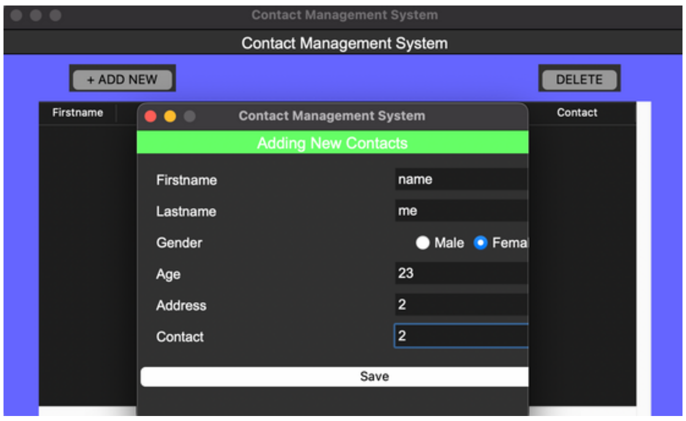
Result

The Contact Management System was successfully developed using Python and the Tkinter library for GUI, along with SQLite for persistent data storage. The application was tested thoroughly for its core functionalities, and the results are as follows:

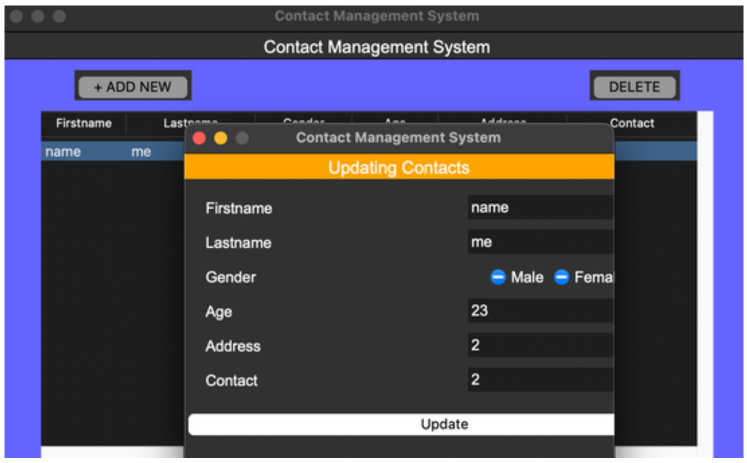
1. User-Friendly Interface:
2. The GUI developed using Tkinter provides a clean and intuitive interface. All buttons and labels are clearly positioned, making the application easy to use for users with minimal technical knowledge.
3. Add Contact:
4. Users are able to input and save contact details including first name, last name, gender, age, address, and phone number. Once submitted, the data is correctly stored in the SQLite database and instantly reflected in the contact list display.
5. View Contacts:
6. All stored contacts are displayed in a tabular format within the application. Users can scroll through the list and select individual contacts for further actions.
7. Update Contact:
8. The system allows modification of existing contact details. Upon selecting a contact and entering new data, the updated information is saved in the database and reflected in the GUI without the need to restart the application.
9. Delete Contact:
10. Users can delete selected contacts from the system. The corresponding record is removed from both the database and the GUI display.
11. Data Persistence:
12. All contact data is stored in an SQLite database file. The application successfully retains data between sessions, ensuring that no information is lost upon closing and reopening the program.



Firstname	Lastname	Gender	Age	Address	Contact
name	me	Female	23	2	2


Contact Management System
Contact Management System
+ ADD NEW
DELETE
Firstname
Contact
Contact Management System
Adding New Contacts
Save

Firstname	name
Lastname	me
Gender	Male [✓ Fema]
Age	23
Address	2
Contact	2


Contact Management System
Contact Management System
+ ADD NEW
DELETE
Firstname
Contact
name
me
Contact Management System
Updating Contacts
Update

Firstname	name
Lastname	me
Gender	Male Femal
Age	23
Address	2
Contact	2

Conclusion

The development of the Contact Management System using Python, Tkinter, and SQLite has effectively demonstrated the integration of graphical user interfaces with a lightweight database for managing personal or professional contact information. The project successfully fulfills the essential functionalities such as adding, viewing, updating, and deleting contact records, all through a simple and intuitive GUI.

Through this project, we gained hands-on experience in designing desktop-based applications and implementing CRUD (Create, Read, Update, Delete) operations with persistent data storage. The use of Tkinter made it possible to create an interactive and user-friendly interface, while SQLite ensured that all contact data is securely stored and accessible across sessions.
This application serves as a practical solution for users seeking a minimal, offline contact management tool without the complexity of cloud-based systems. Moreover, it lays a strong foundation for further enhancements such as search functionality, data import/export, and improved security features.

Overall, the project not only meets its intended objectives but also reinforces key concepts in software development, GUI programming, and database management, making it a valuable learning experience in the field of computer science and engineering.
Let me know if you'd like to include future enhancements or a system architecture diagram to make your report more complete.
4o