DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

MINI-PROJECT REPORT
(CS433P Programming Paradigms)

PHONE BOOK SYSTEM

BY

DHRUV MINESH PATEL (2262217)
ASHWIN SURESH (2262207)
PHILEMON T JOSEPH (2262242)

B. Tech in Computer Science Engineering (Data Science)

School of Engineering and Technology,
CHRIST (Deemed to be University),
Kumbalagodu, Bengaluru-560 074

March 2024

1. Introduction

The Phone Book System is a robust and user-friendly Java application designed to simplify the management of contacts. In our increasingly connected world, having an efficient way to store and retrieve contact information is essential for both personal and professional purposes. Whether you're keeping track of friends, family members, colleagues, or clients, the Phone Book System provides a reliable solution for organizing and accessing contact details with ease.

This intuitive application offers a range of features to streamline the process of managing contacts. Users can effortlessly add new contacts, search for existing ones, and navigate through the interface with simplicity and efficiency. With its clean and user-friendly design, the Phone Book System ensures a seamless experience for users of all levels of technical expertise.

Behind its sleek and intuitive interface, the Phone Book System leverages the power of Java's Swing library to deliver a visually appealing and responsive user experience. From the main window to the individual contact entry and search screens, every aspect of the application is meticulously crafted to enhance usability and productivity.

With the Phone Book System, gone are the days of flipping through endless pages of physical phone books or struggling to remember contact details. Whether you're on your desktop, laptop, or even a mobile device, the Phone Book System puts your entire contact list at your fingertips, allowing you to stay connected and organized wherever you go.

From its intuitive interface to its robust functionality, the Phone Book System is the ultimate solution for managing contacts effectively in today's fast-paced digital world. Whether you're a busy professional, a social butterfly, or someone who simply wants to keep their contacts organized, the Phone Book System is your go-to tool for staying connected and in control.

2. Design

- JFrame:

Represents the main window of the application.
Provides the overall structure for organizing and displaying GUI components.

- JPanel:

Used to organize and group components within the JFrame.
Helps in creating separate sections for adding contacts, searching contacts, and the home page.

- JTextField:

Provides an area for users to enter text such as name, phone number, and address.
Allows users to input contact details for adding or searching contacts.

- JTextArea:

Allows for displaying multiple lines of text.
Used here to display contact details when searching for contacts.

- JButton:

Enables users to perform actions such as adding contacts, searching for contacts, and returning to the home page.
Provides a way to trigger specific actions based on user input.

- JLabel:

Used to display text labels for input fields and buttons.
Provides labels to guide users on what information to input or what action to perform.

- ActionListener:

Interface used to handle events triggered by user interaction with buttons.
Defines methods to respond to button clicks and perform the desired actions.

Functionality:

- Add Contact:

Allows users to input contact details (name, phone number, address) and add them to the phone book.
Utilizes JTextFields for users to input contact information.
Upon clicking the "Add Contact" button, the ActionListener triggers the addition of the contact details to the phone book.

- Search Contact:

Enables users to search for existing contacts by name and displays their details if found.
Users input the name of the contact they wish to search for in a JTextField.
Upon clicking the "Search" button, the ActionListener searches for the contact in the phone book and displays its details if found.

- Return to Home Page:

Provides navigation to return to the home page from the add contact and search contact screens.
Users can easily navigate back to the main menu using the "Return to Home Page" button.
The ActionListener associated with this button switches the JPanel to display the home page.

- Exit:

Allows users to exit the application.
Upon clicking the "Exit" button, the ActionListener terminates the application.
File Handling:

- Contacts Storage:

Contacts are stored in a text file named "contacts.txt".
Each line in the text file represents a contact, with its name, phone number, and address separated by commas.
When a contact is added, its details are appended to the text file using FileWriter, BufferedWriter, and PrintWriter.

- Searching Contacts:

When searching for a contact, the application reads the text file line by line to find a match.
It splits each line into parts using the comma delimiter and compares the name with the search query.
If a match is found, the contact details (phone number and address) are displayed.

User Interface:

- Layout Organization:

The interface is designed using Swing components to provide a simple and intuitive user experience.
Components are organized into separate panels for adding contacts, searching contacts, and the home page.
This layout organization helps in maintaining clarity and ease of navigation for users.

- Navigation Buttons:

Navigation between panels is facilitated by buttons with appropriate action listeners.
Users can easily switch between the home page, add contact screen, and search contact screen using these buttons.

Error Handling:

- Invalid Input:

The application provides error handling for cases such as invalid input.
For example, if users enter non-numeric characters in the phone number field, an error message is displayed using a JOptionPane dialog.

- Non-existent Contact:

When searching for a contact that does not exist in the phone book, the application alerts users with a JOptionPane dialog.
This ensures that users are informed when their search query does not yield any results.

3. Implementation (Code)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class PhoneBookSystemFinal extends JFrame {
    private JPanel homePanel;
    private JPanel addContactPanel;
    private JPanel searchPanel;
    private JTextField nameField;
    private JTextField phoneNumberField;
    private JTextField addressField;
    private JTextArea displayArea;
    private JTextField displayNumberField;
    private JTextField displayAddressField;
    private JButton addContactButton;
    private JButton searchButton;
    private JButton exitButton;
    private JButton returnHomeButton;
    private JButton addEntryButton;
    private JButton searchReturnHomeButton;
    private JButton searchContactButton;
    private JTextField searchField;

    private static final String FILE_NAME = "contacts.txt";

    public PhoneBookSystemFinal() {
        setTitle("Phone Book System");
        setSize(400, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Initialize components
        initComponents();

        // Set home panel as default
        setContentPane(homePanel);
    }
```

```
private void initComponents() {
    homePanel = new JPanel();
    homePanel.setLayout(new GridLayout(4, 1));

    addContactPanel = new JPanel();
    addContactPanel.setLayout(new GridLayout(4, 2));

    searchPanel = new JPanel();
    searchPanel.setLayout(new GridLayout(6, 1));

    nameField = new JTextField();
    phoneNumberField = new JTextField();
    addressField = new JTextField();
    displayArea = new JTextArea(10, 30);
    displayNumberField = new JTextField();
    displayAddressField = new JTextField();
    addContactButton = new JButton("Add Contact");
    searchButton = new JButton("Search");
    exitButton = new JButton("Exit");

    addContactButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setContentPane(addContactPanel);
            revalidate();
            repaint();
        }
    });

    searchButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setContentPane(searchPanel);
            revalidate();
            repaint();
        }
    });

    exitButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });

    returnHomeButton = new JButton("Return to Home Page");
    addEntryButton = new JButton("Add");
    searchReturnHomeButton = new JButton("Return to Home Page");
    searchContactButton = new JButton("Search");
```

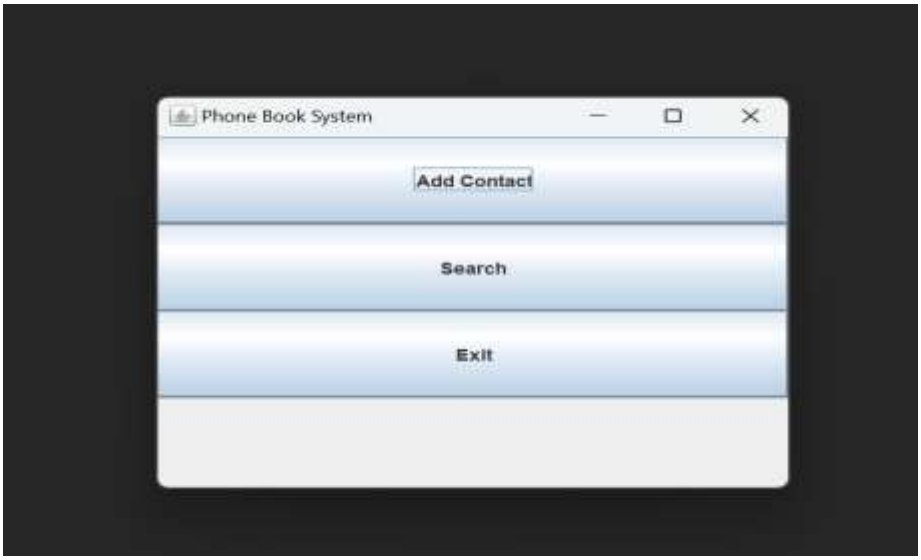
```
searchField = new JTextField();

addEntryButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String phoneNumber = phoneNumberField.getText();
        String address = addressField.getText();

        try (FileWriter fw = new FileWriter(FILE_NAME, true);
             BufferedWriter bw = new BufferedWriter(fw);
             PrintWriter out = new PrintWriter(bw)) {
            out.println(name + "," + phoneNumber + "," + address);
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        JOptionPane.showMessageDialog(null, "Contact added successfully!");
        nameField.setText("");
        phoneNumberField.setText("");
        addressField.setText("");
    }
});

searchContactButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String searchName = searchField.getText();
        displayArea.setText("");
        displayNumberField.setText("");
        displayAddressField.setText("");
        boolean found = false;
        try (BufferedReader br = new BufferedReader(new FileReader(FILE_NAME))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(",");
                if (parts.length >= 3 && parts[0].equalsIgnoreCase(searchName)) {
                    found = true;
                    displayNumberField.setText(parts[1]);
                    displayAddressField.setText(parts[2]);
                    break; // Exit loop after finding the first match
                }
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        if (!found) {
            JOptionPane.showMessageDialog(null, "Details not found.");
```

```
            }
        }
    });

    returnHomeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setContentPane(homePanel);
            revalidate();
            repaint();
        }
    });

    searchReturnHomeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setContentPane(homePanel);
            revalidate();
            repaint();
        }
    });

    addContactPanel.add(new JLabel("Name:"));
    addContactPanel.add(nameField);
    addContactPanel.add(new JLabel("Phone Number:"));
    addContactPanel.add(phoneNumberField);
    addContactPanel.add(new JLabel("Address:"));
    addContactPanel.add(addressField);
    addContactPanel.add(addEntryButton);
    addContactPanel.add(returnHomeButton);

    searchPanel.add(new JLabel("Enter Name to Search:"));
    searchPanel.add(searchField);
    searchPanel.add(new JLabel("Number:"));
    searchPanel.add(displayNumberField);
    searchPanel.add(new JLabel("Address:"));
    searchPanel.add(displayAddressField);
    searchPanel.add(searchContactButton);
    searchPanel.add(searchReturnHomeButton);

    homePanel.add(addContactButton);
    homePanel.add(searchButton);
    homePanel.add(exitButton);
    homePanel.add(new JLabel()); // Empty label for spacing
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
```

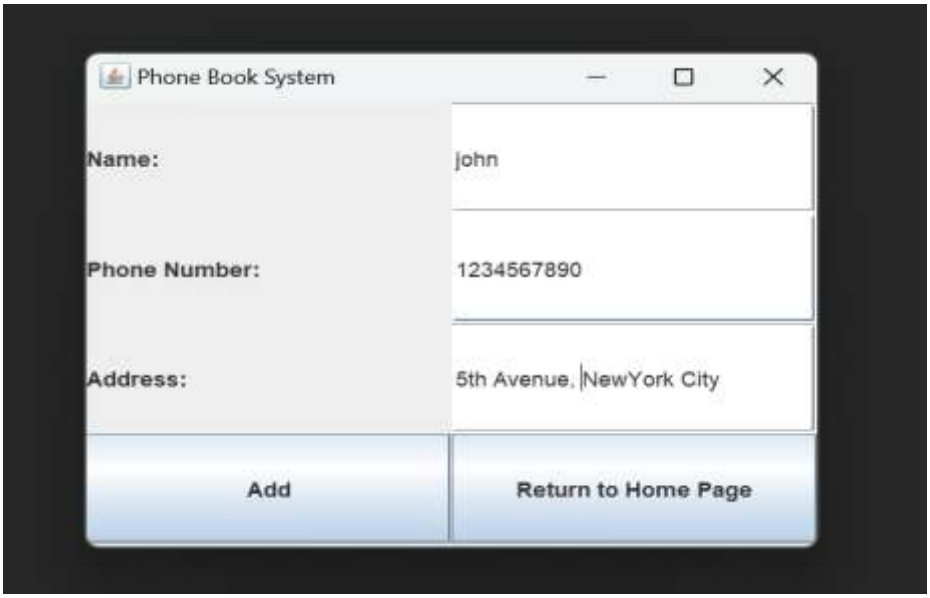
```
            public void run() {
                new PhoneBookSystem().setVisible(true);
            }
        });
    }
}
```

4. Results (Screenshots)

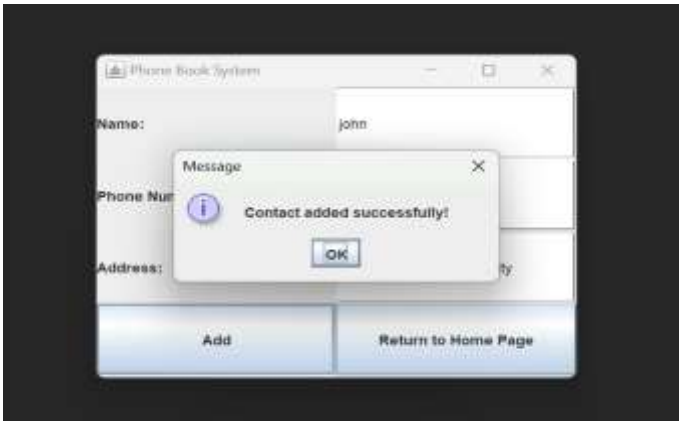
4.1 Home Page



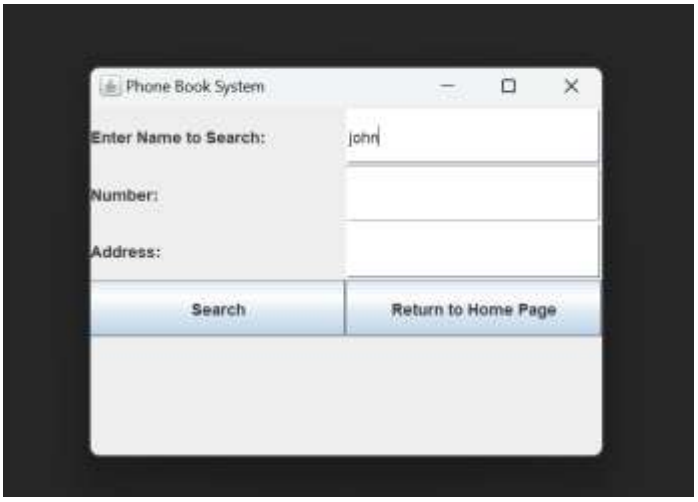
4.2 Entering Information



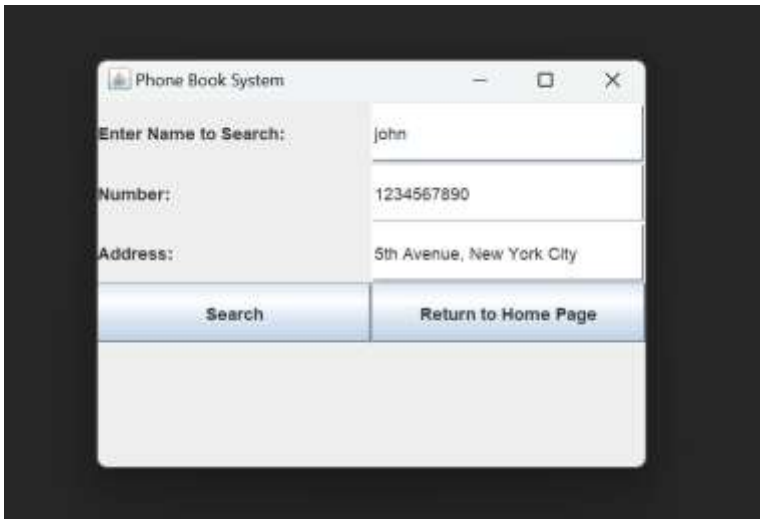
4.3 Information Added Successfully



4.4 Searching for number using name.



4.5 Information Derived.



5. Conclusion

In conclusion, the Phone Book System is a versatile and efficient Java application that offers a comprehensive solution for managing contacts. With its intuitive user interface and robust functionality, it provides users with the tools they need to organize and access their contact information with ease.

By leveraging the power of Java's Swing library, the Phone Book System delivers a visually appealing and responsive user experience, ensuring seamless navigation and interaction across the application. From adding new contacts to searching for existing ones, every feature is designed to streamline the process of managing contacts and enhance productivity.

Whether you're a busy professional, a social butterfly, or someone who simply wants to keep their contacts organized, the Phone Book System is the perfect tool for staying connected and in control. With its ability to store and retrieve contact details efficiently, it simplifies the task of managing contacts, allowing users to focus on what matters most – building and maintaining relationships.

In today's fast-paced digital world, having an efficient way to store and access contact information is essential for staying connected and organized. With the Phone Book System, users can easily keep track of their contacts and ensure that they are always just a click away. Whether you're at home, in the office, or on the go, the Phone Book System empowers you to stay connected and in control of your contacts.