# CONTACT MANAGEMENT SYSTEM –JAVA ABSTRACT

The Contact Management System (CMS) is a Java-based console application designed to efficiently store, manage, and retrieve contact details. The system allows users to add, view, search, update, and delete contact records, ensuring organized and easily accessible contact data.

The CMS provides an interactive menu-driven interface, making it simple to use and extend. It can serve as a foundational model for real-world applications such as phonebooks, CRM tools, or employee directories.

# SOURCE CODE:

import java.util.ArrayList; import java.util.Scanner; class Contact {

private String name;

private String phoneNumber; private String email;

public Contact(String name, String phoneNumber, String email) { this.name = name;

this.phoneNumber = phoneNumber; this.email = email;

}

public String getName() { return name;

}

public String getPhoneNumber() { return phoneNumber;

}

public String getEmail() { return email;

}

public void setName(String name) { this.name = name;

}

public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber;

}

public void setEmail(String email) { this.email = email;

}

public String toString() {

return String.format("%-20s %-15s %-25s", name, phoneNumber, email);

}

}

public class ContactManagementSystem {

private static final ArrayList<Contact> contacts = new ArrayList<>(); private static final Scanner sc = new Scanner(System.in);

public static void main(String[] args) { int choice;

do {

System.out.println("\n===== CONTACT MANAGEMENT SYSTEM

=====");

System.out.println("1. Add Contact"); System.out.println("2. Display All Contacts"); System.out.println("3. Search Contact"); System.out.println("4. Update Contact");

System.out.println("5. Delete Contact"); System.out.println("6. Exit"); System.out.print("Enter your choice: "); while (!sc.hasNextInt()) {

System.out.print("Invalid input. Enter a number: "); sc.next();

}

choice = sc.nextInt(); sc.nextLine();

switch (choice) {

case 1 -> addContact(); case 2 -> displayContacts(); case 3 -> searchContact(); case 4 -> updateContact(); case 5 -> deleteContact();

case 6 -> System.out.println("Exiting... Goodbye!");

default -> System.out.println("Invalid choice. Please try again.");

}

} while (choice != 6);

}

private static void addContact() { System.out.print("Enter name: "); String name = sc.nextLine().trim();

System.out.print("Enter phone number: "); String phoneNumber = sc.nextLine().trim();

if (!phoneNumber.matches("\\d{10}")) { System.out.println("Invalid phone number! Must be 10 digits."); return;

}

for (Contact c : contacts) {

if (c.getPhoneNumber().equals(phoneNumber)) { System.out.println("Contact with this number already exists!"); return;

}

}

System.out.print("Enter email: "); String email = sc.nextLine().trim();

if (!email.contains("@") || !email.contains(".")) { System.out.println("Invalid email format!"); return;

}

contacts.add(new Contact(name, phoneNumber, email)); System.out.println("Contact added successfully!");

}

private static void displayContacts() { if (contacts.isEmpty()) {

System.out.println("No contacts to display."); return;

}

System.out.println("\n---- CONTACT LIST ");

System.out.printf("%-20s %-15s %-25s%n", "Name", "Phone Number", "Email");

System.out.println(" ");

for (Contact c : contacts) { System.out.println(c);

}

}

private static void searchContact() {

System.out.print("Enter name or phone number to search: "); String query = sc.nextLine().trim().toLowerCase();

boolean found = false;

for (Contact c : contacts) {

if (c.getName().toLowerCase().contains(query) || c.getPhoneNumber().equals(query)) {

if (!found) {

System.out.println("\n---- SEARCH RESULTS ");

System.out.printf("%-20s %-15s %-25s%n", "Name", "Phone Number", "Email");

System.out.println("

--");

found = true;

}

System.out.println(c);

}

}

if (!found) {

System.out.println("No contact found matching: " + query);

}

}

private static void updateContact() {

System.out.print("Enter name or phone number of contact to update: "); String query = sc.nextLine().trim().toLowerCase();

ArrayList<Contact> matchedContacts = new ArrayList<>(); for (Contact c : contacts) {

if (c.getName().toLowerCase().contains(query) || c.getPhoneNumber().equals(query)) {

matchedContacts.add(c);

}

}

if (matchedContacts.isEmpty()) {

System.out.println("No contact found matching: " + query); return;

}

Contact contactToUpdate = selectContact(matchedContacts, "update"); if (contactToUpdate == null) return;

System.out.print("Enter new name (leave blank to keep unchanged): "); String name = sc.nextLine().trim();

if (!name.isEmpty()) contactToUpdate.setName(name);

System.out.print("Enter new phone number (leave blank to keep unchanged): ");

String phone = sc.nextLine().trim(); if (!phone.isEmpty()) {

if (phone.matches("\\d{10}")) { for (Contact c : contacts) {

if (c != contactToUpdate CC c.getPhoneNumber().equals(phone)) { System.out.println("Another contact already has this number!"); return;

}

}

contactToUpdate.setPhoneNumber(phone);

} else {

System.out.println("Invalid phone number! Must be 10 digits."); return;

}

}

System.out.print("Enter new email (leave blank to keep unchanged): "); String email = sc.nextLine().trim();

if (!email.isEmpty()) {

if (email.contains("@") CC email.contains(".")) { contactToUpdate.setEmail(email);

} else {

System.out.println("Invalid email format! Keeping old email.");

}

}

System.out.println("Contact updated successfully!");

}

private static void deleteContact() {

System.out.print("Enter name or phone number of contact to delete: "); String query = sc.nextLine().trim().toLowerCase();

ArrayList<Contact> matchedContacts = new ArrayList<>(); for (Contact c : contacts) {

if (c.getName().toLowerCase().contains(query) || c.getPhoneNumber().equals(query)) {

matchedContacts.add(c);

}

}

if (matchedContacts.isEmpty()) {

System.out.println("No contact found matching: " + query); return;

}

Contact contactToDelete = selectContact(matchedContacts, "delete"); if (contactToDelete == null) return;

System.out.print("Are you sure you want to delete this contact? (y/n): "); String confirm = sc.nextLine().trim().toLowerCase();

if (confirm.equals("y")) { contacts.remove(contactToDelete); System.out.println("Contact deleted successfully!");

} else {

System.out.println("Deletion cancelled.");

}

}

private static Contact selectContact(ArrayList<Contact> matchedContacts, String action) {

if (matchedContacts.size() == 1) return matchedContacts.get(0); System.out.println("Multiple contacts found:");

for (int i = 0; i < matchedContacts.size(); i++) { System.out.println((i + 1) + ". " + matchedContacts.get(i));

}

System.out.print("Enter the number of the contact to " + action + ": "); while (!sc.hasNextInt()) {

System.out.print("Invalid input. Enter a number: "); sc.next();

}

int choice = sc.nextInt(); sc.nextLine();

if (choice < 1 || choice > matchedContacts.size()) { System.out.println("Invalid selection!");

return null;

}

return matchedContacts.get(choice - 1);

}

}

# OUTPUT:

===== CONTACT MANAGEMENT SYSTEM =====

1. Add Contact
2. Display All Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Exit

Enter your choice: 1

Enter name: Rahul Sharma

Enter phone number: 9876543210 Enter email: [rahul@gmail.com](mailto:rahul@gmail.com) Contact added successfully!

# ===== CONTACT MANAGEMENT SYSTEM =====

1. Add Contact
2. Display All Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Exit

Enter your choice: 1 Enter name: Priya Verma

Enter phone number: 9998887776 Enter email: [priya@outlook.com](mailto:priya@outlook.com) Contact added successfully!

# ===== CONTACT MANAGEMENT SYSTEM =====

1. Add Contact
2. Display All Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Exit

Enter your choice: 2

# ---- CONTACT LIST ----

Name Phone Number Email

Rahul Sharma 9876543210 [rahul@gmail.com](mailto:rahul@gmail.com) Priya Verma 9998887776 [priya@outlook.com](mailto:priya@outlook.com)

# ===== CONTACT MANAGEMENT SYSTEM =====

Enter your choice: 3

Enter name or phone number to search: priya

# ---- SEARCH RESULTS ----

Name Phone Number Email

Priya Verma 9998887776 [priya@outlook.com](mailto:priya@outlook.com)

# ===== CONTACT MANAGEMENT SYSTEM =====

Enter your choice: 4

Enter name or phone number of contact to update: rahul Enter new name (leave blank to keep unchanged): Rahul S. Enter new phone number (leave blank to keep unchanged):

Enter new email (leave blank to keep unchanged): [rahulsharma@gmail.com](mailto:rahulsharma@gmail.com) Contact updated successfully!

# ===== CONTACT MANAGEMENT SYSTEM =====

Enter your choice: 2

# ---- CONTACT LIST ----

Name Phone Number Email

Rahul S. 9876543210 [rahulsharma@gmail.com](mailto:rahulsharma@gmail.com) Priya Verma 9998887776 [priya@outlook.com](mailto:priya@outlook.com)

# ===== CONTACT MANAGEMENT SYSTEM =====

Enter your choice: 4

Enter name or phone number of contact to update: rahul Enter new name (leave blank to keep unchanged): Rahul S. Enter new phone number (leave blank to keep unchanged):

Enter new email (leave blank to keep unchanged): [rahulsharma@gmail.com](mailto:rahulsharma@gmail.com) Contact updated successfully!

# ===== CONTACT MANAGEMENT SYSTEM =====

Enter your choice: 2

# ---- CONTACT LIST ----

Name Phone Number Email

Rahul S. 9876543210 [rahulsharma@gmail.com](mailto:rahulsharma@gmail.com) Priya Verma 9998887776 [priya@outlook.com](mailto:priya@outlook.com)

# ===== CONTACT MANAGEMENT SYSTEM =====

Enter your choice: 6 Exiting... Goodbye!

# CONCLUSION:

The Contact Management System (CMS) developed in Java is a robust, user- friendly application that demonstrates the practical use of Object-Oriented Programming (OOP) concepts, collections, and input validation. It allows users to efficiently add, view, search, update, and delete contacts, providing flexibility by enabling updates and deletions through either contact name or phone number.

Through this project, key programming concepts such as encapsulation, array lists, exception handling, and menu-driven interfaces were applied to create a

fully functional and reliable system. Error handling ensures data integrity, prevents duplicates, and maintains proper formatting for phone numbers and emails.

This CMS can serve as a foundational model for real-world applications such as phonebooks, customer directories, or CRM systems, and it can be easily extended to include additional features like file storage, GUI integration, or cloud database connectivity in the future.

Overall, the project highlights how structured programming, careful validation, and user-centric design combine to create a practical and maintainable software solution.