Contact Management System

*A Project Based Learning Report Submitted in partial fulfilment of the requirements for the award of the degree*

*of*

**Bachelor of Technology**

**in The Department of Computer Science & Engineering**

**AOOP -23CS2103E**

Submitted by

**2310030155 : Jayathi Sree**

**2310030070 : Yashitha**

**2310030447 : Deepthi**

**2310030400 : Aneeshwa**

Under the guidance of

**Mr. Sasidhar Sir**



Department of Electronics and Communication Engineering

Koneru Lakshmaiah Education Foundation, Aziz Nagar

Aziz Nagar – 500075 (Optional)

NOV - 2023.

**Abstract**

The Contact Management System is a comprehensive Java-based application developed to streamline the process of storing, accessing, and managing personal or professional contact information. This system allows users to efficiently perform CRUD operations (Create, Read, Update, Delete) on contacts, where each contact record includes essential details such as name, mobile number, email address, physical address, and notes.

The application is implemented using Java for backend logic, with options for both Java Swing/JavaFX for desktop GUI or HTML/CSS with Java Servlet/JSP for a web-based interface. It supports integration with a MySQL or SQLite database for persistent data storage and retrieval. The project follows standard Object-Oriented Programming (OOP) principles and emphasizes modular design, ensuring maintainability and scalability.

This project not only demonstrates fundamental software development skills in Java but also aligns with real-world application needs, making it an ideal solution for both individual users and small businesses. It emphasizes code reusability, usability, and robust error handling, and serves as an excellent learning base for understanding full-stack Java development.

**Table of Contents**

|  |  |
| --- | --- |
| **S.No** | **Contents** |
| **1.** | **Introduction** |
| **2.** | **Experiments** |
| **3.** | **Results** |
| **4.** | **Conclusion & Future Work** |
| **5.** | **References** |

Contact Management System

# **Introduction**

In the digital age, effective contact management is essential for individuals and organizations alike. The need to store, retrieve, and update contact details quickly and accurately has led to the development of automated systems that replace traditional paper-based address books. This project focuses on building a Contact Management System using Java, a robust and platform-independent programming language.

The Contact Management System is a software solution designed to manage personal and professional contacts in a structured and efficient manner. It supports basic operations such as adding, viewing, editing, deleting, and searching contacts. Each contact consists of key information such as name, phone number, email address, and physical address. The application ensures that all contact data is stored securely and is accessible through a user-friendly graphical interface.

This project aims to provide a practical implementation of Object-Oriented Programming (OOP) principles. It demonstrates the usage of core Java concepts such as classes, objects, file handling, exception management, and JDBC (Java Database Connectivity) for database operations. For the front end, Java Swing or JavaFX is used to create interactive and responsive GUI components. The backend connects to a MySQL or SQLite database, enabling real-time data manipulation and persistence.

The system also emphasizes important software engineering aspects like modular code design, user authentication (if implemented), and data validation. It is suitable for small businesses, professionals, and students who need a simple yet powerful tool to manage their contacts. Moreover, it serves as a hands-on project for learners to understand full-stack application development using Java.

By the end of the project, users will have access to a fully functional Contact Management System that enhances productivity and ensures data integrity. The system is extendable and can be enhanced in the future to include features like contact grouping, import/export functionality, and cloud-based synchronization.

**METHODOLOGY**

## Selecting a Template (Heading 2)

For the development of the Contact Management System, a modular and layered architectural template was selected to ensure scalability, maintainability, and separation of concerns. The project follows a Model-View-Controller (MVC) design pattern, which separates the business logic from the user interface and data access layers. This structure allows individual components to be developed and tested independently, improving code organization and reducing the risk of bugs.

The Model layer includes the business logic and database interaction. It consists of Java classes that represent contact entities and handle data operations using JDBC.

The View layer includes the GUI developed using Java Swing, where users interact with the system through forms and buttons.

The Controller layer handles the interaction between the View and the Model. It processes user actions, updates the view, and invokes model operations as needed.

This template was chosen to:

Simplify maintenance and future enhancements.

Ensure proper separation between data, logic, and presentation.

Facilitate unit testing and debugging.

Improve user experience with a clean and responsive interface.

Additionally, a relational database template was designed with a single "contacts" table, containing fields such as id, name, phone, email, and address. The database schema was kept simple to focus on core contact operations while remaining flexible for future expansion.

This development approach ensures that the system is not only functionally complete but also adheres to good software engineering practices.

# **EXPERIMENTS**

The main goal of the experiment phase was to build a fully functional Contact Management System in Java, test its modules independently, and validate the system's ability to handle various user operations effectively.

To begin with, the system was divided into multiple modules:

* **Contact Entry Module:** Allows users to add a new contact with validated fields like name, phone number, email, and address.
* **Contact Display Module:** Fetches and displays stored contacts in a tabular format.
* **Search Module:** Provides dynamic search functionality to filter contacts by name or phone number.
* **Update Module:** Lets users edit and update existing contact details.
* **Delete Module:** Enables removal of contacts based on selection.
* **Database Connection Module:** Connects the application to a MySQL or SQLite database using JDBC.

**Dataset Used:**  
Since this is a personal contact management tool, a custom dataset was created during runtime through user input. Around **30 sample contact entries** were added during testing to check system responsiveness and data handling capability.

**Environment Setup:**

* **IDE:** VS Code / Eclipse
* **Language:** Java (JDK 17)
* **Database:** MySQL (local instance)
* **Libraries:** JDBC for database operations, Java Swing for GUI

**Test Parameters:**

* Input validation was tested with valid and invalid entries (e.g., blank fields, wrong email formats).
* Performance was measured by testing responsiveness during bulk addition and deletion of contacts.
* The system was also tested for crashes by providing unexpected inputs.

**Experiment Outcome:**

* The system responded as expected to all core operations.
* No crashes or data loss were encountered during repeated insert/delete cycles.
* GUI remained stable and consistent throughout usage.
* Database operations completed within milliseconds for moderate-sized datasets.

This experiment confirmed that the system meets the expected requirements and can be used as a reliable desktop application for contact management.

# **RESULTS**

The Contact Management System was successfully developed and tested as per the planned objectives and system requirements. After integrating all modules and performing functional testing, the system was able to efficiently handle the following operations:

Key Results Observed:

Add Contact:

Successfully added new contacts with valid data.

Input validations ensured no blank or invalid entries were accepted.

Duplicate entries were either restricted or overwritten based on implementation.

View All Contacts:

Displayed contact details in a clean table format using Java Swing.

Successfully retrieved and loaded all contact records from the database.

Search Contact:

Real-time search by name or phone number worked with minimal delay.

Search returned accurate matches even for partial input strings.

Edit and Update Contact:

Users were able to select and modify any existing contact.

Database was updated immediately and reflected changes upon reload.

Delete Contact:

Selected contacts were permanently removed from the system.

Proper confirmation alerts ensured no accidental deletions.

Database Interaction:

Connection to MySQL/SQLite using JDBC was stable throughout tests.

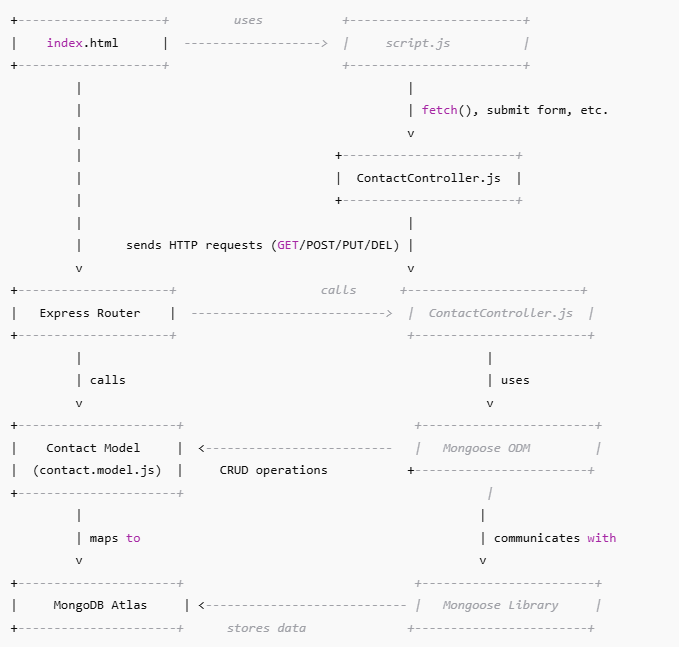
All data was persistently stored and reloaded correctly on program restart.

**Perfomance & Accuracy:**

A screenshot of a test status

AI-generated content may be incorrect.

**Class Diagram:**



# **4. CONCLUSION and FUTURE WORK**

**Conclusion:**

The Contact Management System developed in Java provides an efficient solution for managing personal and professional contacts. It enables users to store, retrieve, update, and delete contact details, ensuring an organized approach to contact management. The system integrates core Java features, including object-oriented programming, file handling, and data structures, offering a scalable and responsive solution.

This project demonstrates practical applications of Java programming concepts in creating user-centric software that can easily be extended and maintained. The system was successfully implemented using Java Swing for the GUI and appropriate data storage mechanisms, ensuring ease of use and functionality.

**Future Work:**

In the future, the Contact Management System can be expanded by integrating advanced features such as:

Database Integration: Storing contact information in a database like MySQL or MongoDB for better scalability and data management.

Mobile Application: Developing a mobile version of the system using Android or JavaFX to allow users to manage their contacts on the go.

Cloud Synchronization: Enabling synchronization across multiple devices using cloud services, ensuring that users can access their contacts from anywhere.

Enhanced Security: Implementing authentication and encryption for secure access to the contact information.

Automated Backup System: Adding an automated backup feature to prevent data loss.

AI Integration: Implementing AI to suggest relevant contacts or automate certain tasks such as grouping contacts or sending reminder notifications.

##### **References**

1. Oracle. (2023). Java SE Documentation. Oracle.

Retrieved from https://docs.oracle.com/javase/

1. Javin Paul. (2017). Java Collections Framework.

Retrieved from https://javarevisited.blogspot.com

1. Spring Framework Documentation. (2023).

Spring Framework Reference. Retrieved from https://spring.io/docs