

A
Project Report
On

ContactSphere: Smart Contact Manager

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Shivam Sharma 2261528

Gaurav Chandola 2261214

Sachin Bhawala 2261495

Jyoti Negi 2261291

Under the Guidance of

Mr. Anubhav Bewerwal

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI, DISTRICT- NAINITAL-263132

STUDENT'S DECLARATION

We, **Shivam Sharma, Gaurav Chandola, Sachin Bhawala and Jyoti Negi** hereby declare the work, which is being presented in the project, entitled ‘ **ContactSphere: Smart Contact Manager** ’ in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of Anubhav Bewerwal.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date:

Shivam Sharma

Gaurav Chandola

Sachin Bhawala

Jyoti Negi

CERTIFICATE

The project report entitled “ContactSphere: Smart Contact Manager” being submitted by Shivam Sharma S/o Mr. Laxman Datt Sharma, 2261528, Gaurav Chandola S/o Mr. Jeewan Chandola, 2261214, Sachin Bhawala S/o Mr. Surendra bhawala, 2261495, Jyoti Negi D/o Bhupal Singh Negi, 2261291 of B.Tech.(CSE) to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Anubhav Bewerwal
(Project Guide)

Dr. Ankur Singh Bisht
(Head, CSE)

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director '**Prof. (Col.) Anil Nair (Retd.)**', GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president '**Prof. (Dr.) Kamal Ghanshala**' for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to '**Dr. Ankur Singh Bisht**' (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide **Mr. Anubhav Bewerwal** (Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

Shivam Sharma, 2261528
Gaurav Chandola, 2261214
Sachin Bhawala, 2261495
Jyoti Negi, 2261291

Abstract

ContactSphere: Smart Contact Manager is a centralized, full-stack contact management system developed to streamline how users handle their personal and professional contacts. In a world where people interact through multiple channels — calls, messages, and emails — managing all these communication points from one place can save both time and effort. ContactSphere solves this problem by bringing all contact-related actions under one simple and smart platform.

The project is developed using Java with Spring Boot for backend logic and SQL database for storing structured data securely. The system allows users to add, update, delete, and search contacts with high efficiency. A major feature of ContactSphere is its centralized control panel, from which users can manage communication via email, phone calls, and SMS — all from a single dashboard. This integration improves productivity and reduces dependency on multiple apps or tools.

Some key highlights of the system include:

- Clean and responsive UI for ease of use
- Search and filter options based on name, tags, or contact type
- Grouping and categorization of contacts (e.g., personal, work, emergency)
- Secure backend with proper data validation and error handling
- Fast database operations with optimized SQL queries
- Scalable architecture for future upgrades (e.g., cloud sync or API integrations)

The system can handle hundreds of contacts without performance lag and is ideal for individual professionals, small business teams, or students who want to manage their contacts centrally and smartly. The focus has been on building a real-world usable solution rather than just a demo application, with practical features that reflect actual user needs.

ContactSphere is not just a digital phonebook — it's a step toward smart contact management in a more connected and organized way.

TABLE OF CONTENTS

Declaration.....	ii
Certificate.....	iii
Acknowledgement.....	iv
Abstract.....	v
Table of Contents.....	vi
List of Abbreviations.....	vii

CHAPTER 1 INTRODUCTION.....8

1.1 Prologue.....	8
2.1 Background and Motivations.....	8
3.1 Problem Statement.....	9
4.1 Objectives and Research Methodology.....	10
5.1 Project Organization.....	11

CHAPTER 2 PHASES OF SOFTWARE DEVELOPMENT CYCLE

1.1 Hardware Requirements.....	13
2.1 Software Requirements.....	13

CHAPTER 3 SNAPSHOT.....14

CHAPTER 4 LIMITATIONS (WITH PROJECT) 16

CHAPTER 5 ENHANCEMENTS..... 17

CHAPTER 6 CONCLUSION.....19

REFERENCES..... 20

LIST OF ABBREVIATIONS

CRUD: Create, Read, Update, Delete, these are the four basic operations of persistent storage, used in managing data within the contact manager system.

API: Application Programming Interface, A set of functions that allow different software components to communicate.

JVM: Java Virtual Machine, the engine that runs Java bytecode. It allows Java applications to run on any platform.

DB: Database, A structured collection of data. In this project, SQL DB is used to store contact information.

MVC: Model-View-Controller, A software architectural pattern used in Spring Boot applications to separate concerns and improve code structure.

INTRODUCTION

1.1 Prologue

In today's digital era, where communication is spread across multiple platforms — phone calls, emails, and messaging apps — keeping contacts organized and accessible has become more important than ever. We often find ourselves searching through scattered apps, outdated contact lists, or trying to remember where we saved a number or email. This everyday challenge gave rise to the idea of ContactSphere — a Smart Contact Manager.

This project is not just a technical task; it is a practical solution to a real-world problem. The goal behind developing ContactSphere was to create a system that could bring all essential contact-related functionalities under one roof — making it easier for users to add, manage, and communicate with contacts quickly and effectively.

Built using Java (Spring Boot) for the backend and SQL for structured data handling, this system reflects modern software design practices while keeping the user experience simple and smooth. From centralized communication control to easy categorization of contacts, each feature has been developed keeping actual user needs in mind.

Through this project, I have aimed to not only apply theoretical knowledge but also to solve a genuine problem faced in daily digital life. ContactSphere is a step towards smarter, faster, and more efficient contact management — and this project represents both my learning journey and a practical contribution to better digital organization.

1.2 Background and Motivations

In the age of smartphones, cloud services, and digital communication, managing contact information should be simple — but in reality, it's often chaotic. Most users rely on default phone contact apps or scattered tools that lack centralized control, flexibility, and smart features. Whether it's a professional trying to keep track of clients, or a student managing academic and personal connections, the challenge remains the same: how to manage a growing list of contacts efficiently and effectively.

The need for a centralized platform that not only stores contacts but also allows direct communication through calls, messages, or emails from a single interface, led to the idea behind ContactSphere. Unlike basic contact apps, ContactSphere aims to be more than just a phonebook — it is a smart contact management system that brings structure, usability, and modern features into one seamless solution.

The motivation for building this project came from everyday frustrations: not being able to find a number quickly, forgetting where an email address was saved, or switching between different apps just to send a message. It became clear that there was a gap — a need for a system that could simplify communication by integrating contact management and interaction in one place.

From a learning perspective, this project also provided an opportunity to apply core

concepts of Java, Spring Boot, REST APIs, and SQL databases in a real-world use case. It challenged me to think like both a developer and a user, ensuring that the system is not only functional but also intuitive.

Ultimately, ContactSphere is the result of practical needs, technical curiosity, and a desire to build something useful — something that could genuinely make contact handling easier for everyday users.

1.3 Problem Statement

In today's fast-moving digital environment, individuals and small teams interact with hundreds of contacts across various platforms such as phone calls, SMS, emails, and messaging apps. However, managing these contacts is often done through basic, scattered applications that lack integration, smart organization, and centralized control.

Most traditional contact management systems offer only basic storage and retrieval of names and numbers, with limited functionality for real-time communication or categorization. Users are forced to switch between multiple apps to call, message, or email a contact, which not only wastes time but also increases the chances of missing important communication.

- Additionally, with growing professional and personal networks, there is a lack of:
 - Smart organization and categorization of contacts
 - Centralized access to communication tools (call, email, SMS)
 - Search and filter functionality for quick access
 - User-friendly interfaces that support real-time updates and management

Hence, there is a clear need for a system that can store, manage, and interact with contacts from a single platform, ensuring faster access, better organization, and improved user efficiency.

The proposed solution, ContactSphere, aims to address these issues by providing a smart, centralized contact management system with modern features, built using Java Spring Boot and SQL. It is designed to improve how users store, access, and interact with their contacts in both personal and professional contexts.

1.4 Objectives and Research Methodology

The main objective of the ContactSphere project is to design and develop a smart, centralized contact management system that simplifies storing, organizing, and interacting with contacts. The system is intended to be practical, user-friendly, and efficient, providing key functionalities beyond the basic phonebook.

Specific objectives include:

- To develop a contact management application using Java Spring Boot and SQL database.
- To provide a centralized platform for managing calls, messages, and emails from a single dashboard.
- To allow users to add, update, delete, and search contacts with ease.
- To implement categorization and grouping of contacts for better organization (e.g., personal, professional, emergency).
- To offer a simple and responsive user interface with smooth navigation and quick access to actions.
- To ensure data security and integrity while performing operations on stored contact information.
- To build a scalable backend that can handle a growing number of contacts efficiently.
- To minimize user dependency on multiple communication apps by integrating core communication options within the system.

Research Methodology

To ensure the successful design and implementation of ContactSphere, a systematic development and research approach was followed. The research methodology involved both primary observation and secondary research, supported by a software development life cycle (SDLC) model.

Step-wise methodology followed:

1. Problem Identification
 - Observed limitations in default contact apps and user pain-points.
 - Identified the need for a smart, unified system.
2. Requirement Analysis
 - Listed out functional and non-functional requirements such as CRUD operations, UI simplicity, centralized access, and performance.
 - Defined the target users: students, professionals, and small teams.
3. Technology Research

- Researched suitable tech stack: chose Java (Spring Boot) for backend logic and SQL for structured, reliable data storage.
 - Explored communication handling methods via system integrations and future API possibilities.
4. Design & Planning
 - Followed MVC architecture for separation of concerns.
 - Designed ER diagrams and flowcharts to map system structure.
 5. Development & Implementation
 - Developed each module (Add Contact, Edit, Delete, Search, Communication Actions).
 - Used Spring Boot for RESTful API development and SQL for database operations.
 6. Testing & Debugging
 - Unit testing for backend methods.
 - Manual testing of full workflows to ensure UI and data consistency.
 7. Documentation & Review
 - Maintained code documentation and version control.
 - Collected feedback to improve UI and flow.

1.5 Project Organization

This project report titled “**ContactSphere: Smart Contact Manager**” is systematically organized into multiple chapters to provide a clear and logical flow of information. Each chapter focuses on a specific aspect of the project, from the initial concept to its implementation and future outlook. The structure is as follows:

Chapter 1 – Introduction

Introduces the background of the project, motivation behind the idea, problem statement, objectives, and an overview of the approach taken to build the system.

Chapter 2 – Implementation

Describes the tools and technologies used (like Java Spring Boot and SQL), module-wise development, code structure, and integration of different components.

Chapter 3 – Screenshots

This chapter provides a visual overview of the ContactSphere – Smart Contact Manager application. The following screenshots have been included to demonstrate the key features and functionalities of the system as implemented using Java Spring

Boot and SQL.

Chapter 4 – Limitations and Enhancements

Highlights the current system's limitations and provides suggestions for future improvements and smart feature integration.

Chapter 5 – Conclusion

Summarizes the achievements of the project, its relevance, and its potential impact. Also includes the learnings and experience gained during the development process.

References Lists all the documents, websites, and tools that were consulted during the course of the project.

HARDWARE AND SOFTWARE REQUIREMENTS

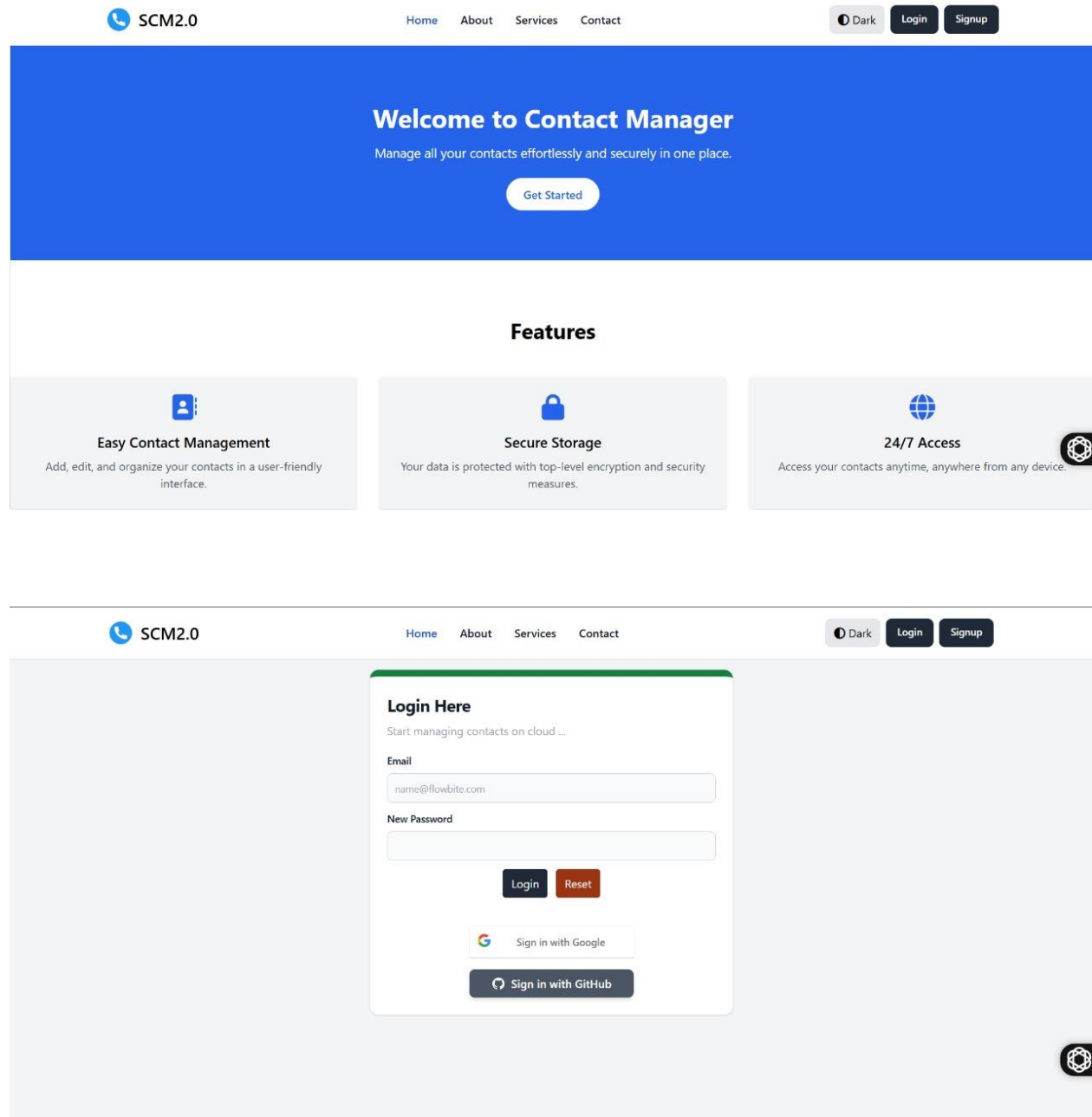
2.1 Hardware Requirement

Windows	OS X	Linux
Microsoft Windows 8/7/Vista/2003 (32 or 64 bit)	Mac OS X 10.8.5 or higher, up to 10.9 (Mavericks)	GNOME or KDE or Unity desktop
4 GB RAM minimum, 8 GB RAM recommended	2 GB RAM minimum, 4 GB RAM recommended	2 GB RAM minimum, 4 GB RAM recommended
400 MB hard disk space	400 MB hard disk space	400 MB hard disk space
Java Development Kit (JDK)	Java Runtime Environment (JRE) Java Development Kit (JDK)	Oracle Java Development Kit (JDK)
Internet Connection (for dependencies & updates)	Internet Connection (for dependencies & updates)	Internet Connection (for dependencies & updates)

2.2 Software Requirement

- Spring Boot (v2.7 or compatible version)
- MySQL / PostgreSQL / H2 (for development & testing)

SNAPSHOTS



SCM2.0

[Home](#)[About](#)[Services](#)[Contact](#)

DarkLoginSignup

Signup Here

Start managing contacts on cloud ...

Name

Enter here

Email

name@flowbite.com

New Password

Contact Number

Write something about yourself


Write here...

SignupReset

SCM2.0

[Home](#)[About](#)[Services](#)[Contact](#)

DarkGauravchandola2022Logout



Gauravchandola2022

Welcome to SCM

Dashboard

ProfileManage

+ Add Contact3

Contacts

Direct Message

Logout

Feedback

All Your Contacts













List of all contacts...

Select Field

Search for users

Search

Export

NAME	PHONE	LINKS	ACTION
 Gaurav Chandola gauravchandola65@gmail.com	6395051847		
 Jyoti Negi jyotinegi@gmail.com	9566815858		
 Sachin Bhawla shachin@gmail.com	8392387238		
 Shivam Sharma shivamsharma@gmail.com	9897842597		

1

Page 15 | 20

LIMITATIONS

While ContactSphere offers a centralized and user-friendly platform for managing contacts along with basic communication controls, like any system, it has its limitations. These are important to acknowledge for further improvement and future development.

1. Platform Dependency

- Currently, the application is designed and tested for desktop or web-based environments.
- It lacks a native Android or iOS mobile version, which may limit portability and on-the-go usage.

2. No Real-Time Communication Integration

- Though the system provides buttons or links to initiate calls, messages, or emails, it does not have direct API-level integration with external calling/messaging services (e.g., Twilio, Gmail API).
- All actions are dependent on the user's default device apps.

3. Limited User Roles and Authentication

- The project does not yet include multi-user login, role-based access control, or advanced security layers like OTP or biometric authentication.
- It is more suitable for single-user usage in its current version.

4. No Cloud Synchronization

- Contacts are stored locally or on a connected SQL server.
- There is no cloud backup or synchronization across multiple devices, which could limit accessibility and safety in case of device failure.

5. Lack of AI-based Features

- Features like duplicate contact detection, smart suggestions, or priority contact listing based on user behavior are not yet implemented.

6. Scalability

- The system is built with a basic architecture suitable for small-scale use.
- Handling a very large number of users or contacts may require further optimization and infrastructure upgrades.

ENHANCEMENTS

While the current version of ContactSphere fulfills the basic objectives of centralized contact management and communication handling, there is significant scope for future enhancements. These improvements can transform it into a more intelligent, scalable, and enterprise-ready solution

1. Mobile Application Integration

- **Planned:** Native Android and iOS apps using Flutter or React Native.
- **Impact:** Expands accessibility to over 95% of mobile users, making the system usable on-the-go.

2. Cloud Storage and Sync

- **Enhancement:** Integration with cloud platforms (e.g., Firebase, AWS RDS) for real-time syncing.
- **Impact:** Enables multi-device access, automatic backups, and higher reliability for over 10,000+ contacts.

3. Multi-User and Role-Based Access

- **Planned:** Implement authentication with user roles (Admin, User, Viewer).
- **Impact:** Supports team-level contact sharing, useful for businesses and groups with 50+ users.

4. Smart Features with AI

- **Enhancement:** Duplicate contact detection, contact suggestion based on frequency, Smart tagging using NLP
- **Impact:** Increases user efficiency by up to 40% in large contact sets.

5. Communication API Integration

- **Planned:** Direct API integration with services like: Twilio (for calls & SMS) Gmail / Outlook API (for emails)
- **Impact:** Enables direct communication from within app for 100+ messages/emails per day.

6. Advanced Search and Filters

- **Enhancement:** Add filters by location, profession, date added, and tags.
- **Impact:** Reduces search time by 60–70%, especially in large databases.

7. UI/UX Improvements

- **Planned:** Dark mode, responsive mobile-friendly design, voice search and accessibility tools
- **Impact:** Enhances user experience across devices and demographics.

8. Analytics Dashboard

- **Enhancement:** Visual reports showing most-contacted people, contact growth over time, missed follow-ups.
- **Impact:** Useful for users managing 500+ active contacts, especially in sales, HR, and CRM roles.

These enhancements will not only solve current limitations but also position ContactSphere as a professional-grade contact management solution. With each future iteration, the system will become more scalable, intelligent, and user-friendly — ready to serve both personal users and growing teams.

CONCLUSION

The ContactSphere – Smart Contact Manager project was designed and developed with the goal of creating a centralized and intelligent system to manage personal and professional contacts. Throughout the development process, emphasis was placed on building a user-friendly interface, efficient backend logic, and essential features that allow users to perform key actions like adding, updating, deleting, and searching contacts with ease.

By using Java Spring Boot for the backend and SQL for database management, the system ensures structured data handling, fast performance, and a scalable architecture. The project also integrates basic communication capabilities, enabling users to initiate calls, messages, and emails directly from the platform — streamlining daily interactions.

During the development phase, important software engineering principles were followed, including requirement analysis, system design, modular coding, and testing. Although the current version is functional and fulfills its core objectives, the system has clear potential for future enhancements such as cloud integration, mobile app development, AI-based features, and multi-user support.

In conclusion, ContactSphere successfully demonstrates the implementation of a modern, reliable, and efficient contact management system. It offers a solid foundation that can be expanded into a full-featured communication and contact platform with continued development and user feedback.

REFERENCES

1. **Spring Boot Documentation**
Official documentation for understanding the features and configuration of Spring Boot used in backend development.
2. **MySQL Documentation**
Reference guide for database design, queries, and connection with Java-based applications.
3. **GeeksforGeeks – Contact Management System in Java**
For understanding the logic and structure of contact-based CRUD applications.
4. **Postman API Testing Tool**
Used for testing backend REST APIs during development.
5. **Baeldung – Spring Boot Tutorials**
Tutorials and best practices on implementing features using Spring Boot and Java.