



CONTACT MANAGEMENT SYSTEM



A PROJECT REPORT

Submitted by

JAYASURYA L (2303811710421066)

in partial fulfillment of requirements for the award of the course

CGB1221-DATABASE MANAGEMENT SYSTEMS

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE- 2025

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**CONTACT MANAGEMENT SYSTEM**” is the bonafide work of **JAYASURYA L(2303811710421066)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



SIGNATURE

Mrs.A.DELPHIN CAROLINA RANI,
M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.



SIGNATURE

Mr.P.MATHESWARAN, M.E.,Ph.D.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

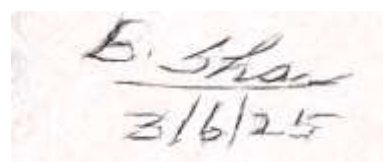
K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “ **CONTACT MANAGEMENT SYSTEM** ” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1221 – DATABASE MANAGEMENT SYSTEMS**.

Signature



JAYASURYA L

Place: Samayapuram

Date:03.06.2025

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr.P.MATHESWARAN, M.E.,Ph.D.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render my sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express my special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- ❑ Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- ❑ Be an institute with world class research facilities
- ❑ Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Contact Management System is a comprehensive web-based application designed to streamline the storage, organization, and retrieval of contact information for users. The proposed architecture integrates a clear separation of concerns between the frontend, backend, and database layers, ensuring modularity and scalability. At the frontend, users interact with an intuitive interface that facilitates seamless navigation and efficient user authentication through secure login and registration modules. Upon successful authentication, users access a centralized dashboard that serves as the hub for all contact management activities. Core functionalities of the dashboard include adding new contacts, editing existing contact details, deleting contacts, searching for specific contacts, and grouping contacts for better organization. These features empower users to manage their contact data with ease and precision. The backend, implemented using PHP, acts as the intermediary between the frontend and the database. It processes user requests, enforces business logic, and ensures secure data handling. All contact-related operations are executed through robust database interaction mechanisms that guarantee data integrity and consistency. The database layer, typically powered by MySQL, is responsible for persistently storing user credentials, contact information, and group associations. Efficient data retrieval and storage are achieved through optimized SQL queries, supporting both the storage and fetching of data as required by user actions.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD DATABASE MANAGEMENT APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>The proposed architecture for the Contact Management System is structured to ensure efficient and secure handling of contact information. The system begins with a frontend interface that provides user authentication, allowing users to either log in or register for an account. Once authenticated, users are granted access to a centralized dashboard where they can perform core operations such as adding, editing, deleting, searching, and grouping contacts. All actions initiated from the dashboard are processed by the backend, which serves as the intermediary between the user interface and the database. The backend is responsible for executing database interactions, including storing new data and fetching existing data as required. This layered approach, with clear separation between frontend, backend, and database interaction, enhances the system's security, scalability, and maintainability while delivering a seamless user experience for managing contacts.</p>	<p>PO1 -3</p> <p>PO2 -3</p> <p>PO3 -3</p> <p>PO4 -3</p> <p>PO5 -3</p> <p>PO6 -3</p> <p>PO7 -3</p> <p>PO8 -3</p> <p>PO9 -3</p> <p>PO10 -3</p> <p>PO11-3</p> <p>PO12 -3</p>	<p>PSO1 -3</p> <p>PSO2 -3</p> <p>PSO3 -3</p>

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	Viii
	LIST OF FIGURE	Xi
	LIST OF ABBREVIATION	Xii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 SQL and Database concepts	3
2	PROJECT METHODOLOGY	8
	2.1 Proposed Work	8
	2.2 Block Diagram	9
3	MODULE DESCRIPTION	10
	3.1 User Authentication Module	10
	3.2 Dashboard Module	10
	3.3 Contact Management Module	11
	3.4 Group Management Module	11
	3.5 Import And Export Module	12
	3.6 Database Interaction Module	12
4	CONCLUSION & FUTURE SCOPE	13
5	APPENDIX A SOURCE CODE	14
	APPENDIX B SCREENSHOTS	31
	REFERENCES	34

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
2.1	BLOCK DIAGRAM	9

LIST OF ABBREVIATIONS

CMS	-	Contact Management sysstem
UI	-	User Interface
CRUD	-	Create,Read,Update,Delete
CSV	-	Comma-Separated values
SQL	-	Structured Query Language
GUI	-	Graphical User Interface
ID	-	Identifier

CHAPTER 1

INTRODUCTION

1. Objective

The proposed architecture for the Contact Management System, as illustrated in the diagram, is organized into distinct layers to ensure efficiency, security, and scalability. The process begins at the frontend, where users interact with the system through a user authentication module, providing options for login and registration. Once authenticated, users gain access to a centralized dashboard that serves as the main interface for managing contacts. The dashboard enables users to perform essential operations such as adding, editing, deleting, searching, and grouping contacts, offering a comprehensive suite of contact management features. All user actions on the dashboard are processed by the backend, which acts as a bridge between the frontend and the database. The backend is responsible for handling business logic, validating user requests, and ensuring secure communication with the database. The database layer is tasked with persistent storage and retrieval of contact data, managed through robust database interaction mechanisms. These mechanisms support both storing new data and fetching existing records, maintaining data integrity and consistency.

2. Overview

The Contact Management System is designed as a robust web application that streamlines the management of personal and professional contacts. The architecture is structured into distinct layers, beginning with a user-friendly frontend that handles user authentication, allowing secure login and registration. Once authenticated, users are directed to a central dashboard, which serves as the core interface for managing contacts. The dashboard offers essential functionalities such as adding, editing, deleting, searching, and grouping contacts, ensuring users can efficiently organize and access their information. To enhance usability, the system now includes features for

importing contacts from CSV files and exporting the current contact list as a CSV file, facilitating easy data migration and backup. All dashboard operations are securely routed to the backend, which is responsible for processing user requests and enforcing business logic. The backend communicates directly with the database, where all contact and user data is stored and managed. Database interactions are optimized for both storing new records and fetching existing data, ensuring data integrity and quick access. The modular separation between frontend, backend, and database layers not only improves maintainability but also allows for future scalability and feature expansion. Security is a priority throughout the system, with careful handling of user sessions and data validation. The import feature allows users to bulk add contacts using a standard CSV format, while the export feature enables users to download their contacts for offline use or transfer to other systems. These enhancements make the system highly adaptable for both individual and organizational needs. The overall design ensures a seamless, efficient, and secure contact management experience, leveraging modern web technologies and best practices. With this architecture, users benefit from a professional interface, reliable data management, and practical tools for comprehensive contact organization.

1.3 SQL and Database

Concepts

1.3.1 Database Connection

The proposed architecture for the Contact Management System is designed to deliver an efficient, secure, and user-friendly platform for managing contact information. The system architecture is divided into three main layers: frontend, backend, and database. At the frontend, users interact with the application through a clear interface that facilitates user authentication, allowing individuals to securely log in or register. Once authenticated, users are directed to a centralized dashboard, which serves as the main hub for all contact management operations. The dashboard provides essential functionalities such as adding new contacts, editing existing details, deleting unwanted contacts, searching for specific entries, and grouping contacts for better organization. All user actions performed on the dashboard are seamlessly routed to the backend, which acts as the intermediary between the user interface and the database. The backend is responsible for processing user requests, applying business logic, and ensuring that only valid operations are executed. It communicates directly with the database layer, where all user and contact data is securely stored. The database interaction component manages both storing new data and fetching existing records, supporting the system's core operations with reliability and speed.

1.3.2 CRUD Operations

The proposed architecture for the Contact Management System is designed to provide a clear, efficient, and scalable solution for managing contact information in both personal and organizational contexts. The system is

structured into distinct layers, beginning with the frontend, which serves as the primary interface for user interaction. At the entry point, users are presented with authentication options, enabling them to securely log in or register for an account. This authentication process ensures that only authorized users can access and manage contact data, thereby maintaining privacy and security. Once authenticated, users are directed to a centralized dashboard that acts as the core hub for all contact management activities. The dashboard offers a suite of essential features, including the ability to add new contacts, edit existing contact details, delete unwanted contacts, search for specific entries, and group contacts for better organization. Each action initiated from the dashboard is seamlessly routed to the backend, which is responsible for processing user requests and enforcing business logic. The backend acts as the intermediary between the frontend interface and the database layer, ensuring that all operations are executed securely and efficiently. When a user adds or updates a contact, the backend validates the input data and communicates with the database to store or modify the relevant records. Deletion requests are similarly processed, with the backend ensuring that only authorized deletions are permitted. The search and grouping functionalities are powered by optimized queries that fetch relevant data from the database, providing users with instant access to the information they need.

The database layer is central to the architecture, providing persistent storage for all user, contact, and group data. Database interactions are managed through a dedicated component that handles both the storage of new data and the retrieval of existing records. This ensures data integrity, consistency, and quick access, even as the system scales to accommodate more users and contacts. The architecture also incorporates robust error handling and data validation mechanisms, minimizing the risk of data loss or corruption. By separating the frontend, backend, and database interaction layers, the architecture promotes modularity, making it easier to maintain, update, and expand the system in the future. Features such as import/export of contacts, advanced search, and integration with external services can be added with minimal disruption to existing functionality. Security is a key consideration

throughout the system, with measures in place to protect user credentials, prevent unauthorized access, and safeguard sensitive contact information.

1.3.3 Data Validation

Before performing database operations, the system validates user inputs to ensure data integrity. For instance, it checks that phone numbers contain exactly 10 digits and that email addresses match a specific regex pattern. This validation prevents the insertion of incorrect or malformed data into the database, maintaining the quality and reliability of the stored information. The validation functions are invoked before executing SQL operations, ensuring that only valid data is processed. If the validation fails, appropriate error messages are displayed to the user, prompting them to correct the input. This approach enhances user experience by providing immediate feedback on data entry errors. Additionally, it reduces the likelihood of data inconsistencies and errors in the database. By ensuring that only valid data is entered, the system maintains the integrity and accuracy of the contact information. This validation process is an essential aspect of data management, contributing to the overall reliability and usability of the application.

1.3.4 Parameterized Queries

To safeguard against SQL injection attacks, the system employs parameters. This approach enhances security by ensuring that user inputs are treated as data, not executable code. By using parameterized queries, the application prevents malicious inputs from altering the intended SQL commands. This method not only secures the application but also improves code readability and maintainability. It separates SQL logic from data, making the code less prone to errors and easier to understand. Additionally, parameterized queries can enhance performance by allowing the database to optimize query execution plans. Overall, this practice is a fundamental aspect of secure and efficient database interaction.

1.3.5 Error Handling

The system incorporates error handling mechanisms to manage potential issues during database operations. By using try-except blocks, the application can catch exceptions such as connection errors or SQL execution failures. This approach allows for graceful error handling, providing informative messages to users and preventing application crashes. For example, if a connection to the database cannot be established, an error message is displayed, and the application exits safely. Similarly, if an SQL query fails, the error is caught, and the user is notified without disrupting the application's flow. This robust error handling ensures that the application remains stable and user-friendly, even in the face of unexpected issues. It also aids in troubleshooting by providing clear error messages that can guide developers in diagnosing and resolving problems. Incorporating comprehensive error handling is essential for building resilient applications that can handle various runtime scenarios effectively.

1.3.6 SQL Execution

The proposed architecture for the Contact Management System is thoughtfully designed to ensure efficiency, security, and scalability in managing contact information. The system is divided into distinct layers, starting with the frontend, which serves as the user's primary interface. User authentication is handled at this stage, allowing individuals to securely log in or register for an account before accessing the system's features. Once authenticated, users are directed to the dashboard, which functions as the central hub for all contact management activities. The dashboard offers essential functionalities such as adding new contacts, editing existing contact details, deleting contacts, searching for specific entries, and grouping contacts for better organization. Each action initiated on the dashboard is seamlessly routed to the backend, which is responsible for processing user requests and enforcing business logic. The backend acts as a mediator between the frontend and the database, ensuring that all operations are executed securely and efficiently. The database layer provides persistent storage for all user and contact data, while the database interaction component manages both storing new data and fetching existing records as needed. This interaction ensures data integrity, consistency, and quick access to information. The architecture also incorporates robust error handling and validation mechanisms, minimizing the risk of data loss or unauthorized access. By separating the frontend, backend, and database layers, the system promotes modularity, making it easier to maintain, update, and scale as user needs evolve. Features such as advanced search, contact grouping, and secure authentication enhance the user experience and make the system adaptable for both personal and organizational use. Overall, this architecture delivers a comprehensive, user-friendly, and secure platform for contact management, empowering users to efficiently organize and access their contacts in a reliable digital environment.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Contact Management System project centers on developing a robust, secure, and user-friendly platform for managing contact information. The system's architecture is divided into several key modules, each responsible for specific functionalities that collectively deliver a seamless user experience. The frontend module is designed to provide an intuitive interface, allowing users to interact with the system efficiently. User authentication is a critical component, ensuring that only registered users can access the dashboard and perform operations on their contacts. This authentication process includes both login and registration features, with secure handling of user credentials. Once authenticated, users are directed to the dashboard, which serves as the operational core of the application. The dashboard offers a suite of essential features, including the ability to add new contacts, edit existing contact details, delete unwanted contacts, search for specific entries, and group contacts for better organization. Each dashboard action is linked to clear user interface elements, making the system accessible even to those with minimal technical expertise. All user actions on the dashboard are routed to the backend, which processes requests and applies the necessary business logic. The backend is responsible for validating user inputs, enforcing access controls, and preparing data for storage or retrieval. It acts as a secure intermediary between the user interface and the database layer, ensuring that operations are performed reliably and efficiently. The database module is tasked with the persistent storage of all user, contact, and group data. It is designed to handle both the storage of new information and the retrieval of existing records, supporting the system's core CRUD (Create, Read, Update, Delete) operations.

2.2 Block Diagram

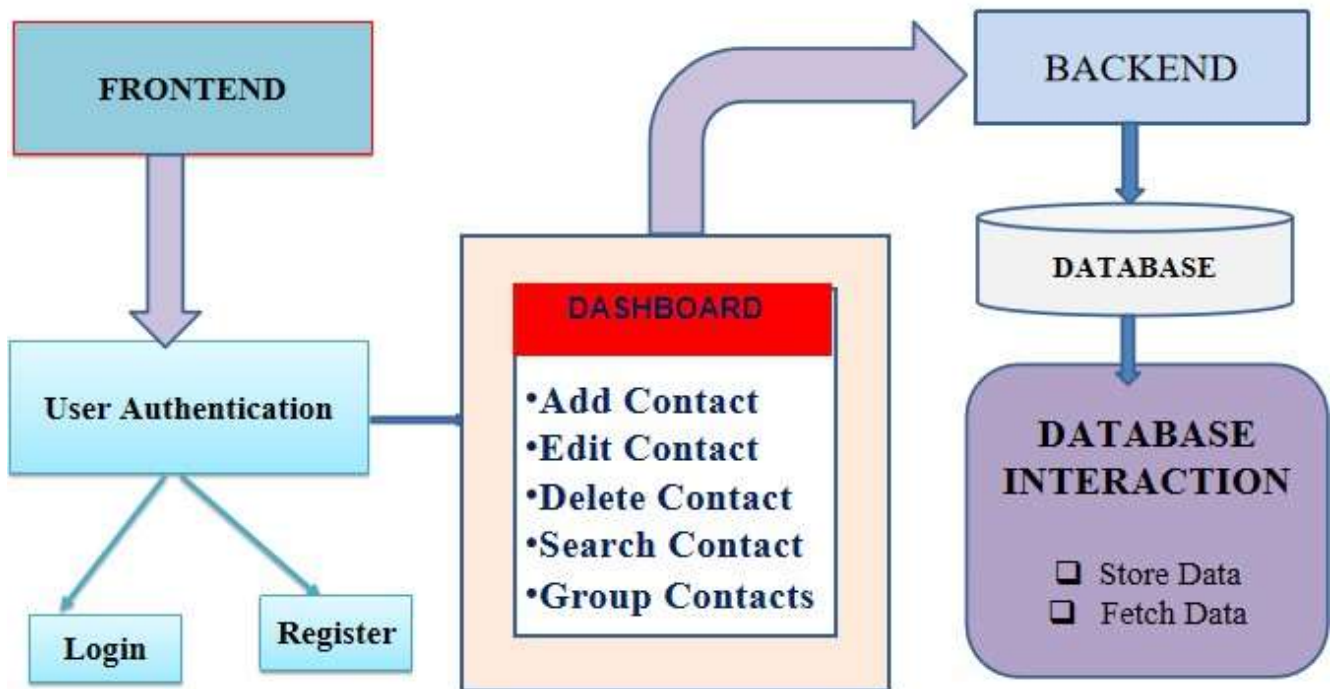


Fig 2.1 BLOCK DIAGRAM

CHAPTER 3

MODULE DESCRIPTION

3.1 User Authentication Module

The User Authentication module is the entry point for all users accessing the system. It manages secure login and registration processes, ensuring that only authorized users can reach the dashboard and other functionalities.

This module validates user credentials, prevents unauthorized access, and maintains user sessions throughout their interaction with the platform. By incorporating features like password hashing and input validation, it safeguards sensitive user information. The module also handles error messages and feedback for incorrect login attempts or registration issues. Integration with the frontend ensures a seamless user experience, while backend logic enforces security protocols.

3.2 Dashboard Module

The Dashboard module serves as the central hub for all contact management activities after successful authentication. It provides users with an organized and intuitive interface to access core features such as adding, editing, deleting, searching, and grouping contacts. The dashboard displays an overview of the user's contact list and any relevant statistics or notifications. It is designed for ease of navigation, allowing users to quickly perform actions or access different modules. The dashboard also integrates import and export options for handling contact data in CSV format, making bulk operations straightforward. Real-time updates and feedback are provided for each user action, ensuring clarity and responsiveness. By centralizing all major functionalities, the dashboard enhances productivity and streamlines workflow. It acts as the main interaction point between the user and the backend logic.

3.3 Contact Management Module

The Dashboard module serves as the central hub for all contact management activities after successful authentication. It provides users with an organized and intuitive interface to access core features such as adding, editing, deleting, searching, and grouping contacts. The dashboard displays an overview of the user's contact list and any relevant statistics or notifications. It is designed for ease of navigation, allowing users to quickly perform actions or access different modules. The dashboard also integrates import and export options for handling contact data in CSV format, making bulk operations straightforward. Real-time updates and feedback are provided for each user action, ensuring clarity and responsiveness. By centralizing all major functionalities, the dashboard enhances productivity and streamlines workflow. It acts as the main interaction point between the user and the backend logic. The design supports scalability, allowing for the addition of new features without disrupting existing workflows. Overall, the Dashboard module is the operational heart of the system.

3.4 Group Management Module

The Group Management module enhances organizational capabilities by allowing users to create, edit, and delete contact groups. Users can assign contacts to groups, making it easier to manage large lists and segment contacts for specific purposes, such as marketing or internal communication. The module provides interfaces for adding new groups with custom names and updating or removing existing ones as needed. It ensures that group-related operations are reflected across the system, maintaining consistency in contact categorization. The backend logic validates group data and manages relationships between contacts and groups in the database. This module supports bulk actions, such as moving multiple contacts to a group or deleting a group and reassigning its members. Group Management improves efficiency, especially for users handling extensive contact lists. It also supports advanced filtering and reporting based on group membership.

3.5 Import And Export Module

The Import/Export module streamlines the process of bulk data handling within the Contact Management System. Users can import contacts from CSV files, allowing for quick population of the contact database from external sources. The module parses uploaded CSV files, validates data, and adds new contacts while handling duplicates or errors gracefully. Export functionality enables users to download their entire contact list or filtered selections as CSV files, facilitating backups, migrations, or sharing. The module ensures compatibility with common spreadsheet applications for ease of use. Error handling and user feedback are integrated to guide users through the import/export process. Backend logic manages the actual data insertion or extraction, maintaining data integrity throughout. This module is especially valuable for organizations transitioning from other systems or managing large-scale contact updates. It enhances the system's flexibility and adaptability to user needs. Overall, the Import/Export module supports efficient data management and improves user productivity.

3.6 Database Interaction Module

The Database Interaction module is the backbone of all data operations in the system. It manages connections to the database, executes SQL queries, and ensures secure storage and retrieval of contact, group, and user information. This module handles all CRUD operations, supporting both individual and bulk data transactions. Security measures such as prepared statements and input sanitization are implemented to prevent SQL injection and data breaches. The module is optimized for performance, supporting fast query execution even as the dataset grows. It also manages database schema updates and migrations as the system evolves. Error handling mechanisms provide detailed logs for debugging and maintenance. The module abstracts database complexities from other parts of the system, promoting modularity and ease of development. It supports transaction management for complex operations like batch imports. Consistent data integrity and reliability are maintained through robust validation and rollback features.

CHAPTER 4

CONCLUSION AND FUTURE ENHANCEMENT

4.1 CONCLUSION

In conclusion, the proposed architecture for the Contact Management System offers a well-structured and systematic approach to managing contacts efficiently. By dividing the system into distinct layers—frontend, backend, and database interaction—the design ensures clear separation of concerns, which enhances maintainability and scalability. The dashboard serves as the operational core, empowering users with essential features such as adding, editing, deleting, searching, and grouping contacts. All user actions are seamlessly routed to the backend, which processes requests and communicates with the database for data storage and retrieval. The database interaction layer is optimized for both storing new information and fetching existing data, ensuring reliability and data integrity. This architecture supports robust error handling and validation, minimizing the risk of data loss or unauthorized access. Overall, the Contact Management System delivers a secure, user-friendly, and scalable solution that meets the needs of both individuals and organizations.

4.2 FUTURE ENHANCEMENT

In the future, the Contact Management System can be enhanced by integrating advanced search and filtering options, such as searching by custom tags or recent activity. Adding email and SMS integration can enable direct communication with contacts from within the dashboard. Support for importing and exporting contacts in various formats, like Excel and vCard, would improve data portability. Incorporating a contact history or activity log would help track changes and interactions over time. A responsive mobile application could be developed for on-the-go access. Integration with cloud storage solutions would facilitate secure backups and synchronization across devices. AI-powered suggestions for duplicate contact merging or smart group creation could further streamline management. Multi-language support would make the system accessible to a broader audience.

APPENDIX A (SOURCE CODE)

```
#===== DATABASE=====

<?php
$host = 'localhost';
$db = 'contact_management';
$user = 'root';
$pass = "";

try {
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
#=====REGISTER=====

<?php
session_start();

include 'includes/db.php';

$error = "";
$success = "";

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = trim($_POST['username']);
    $email = trim($_POST['email']);
    $password = $_POST['password'];
    $confirm_password = $_POST['confirm_password'];
```

```

if ($password !== $confirm_password) {
$error = "Passwords don't match!";
} else {
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

try {
$stmt = $pdo->prepare("INSERT INTO users (username, email, password) VALUES (?, ?, ?)");
if ($stmt->execute([$username, $email, $hashed_password])) {
$success = "Registration successful! You can now login.";
}
} catch (PDOException $e) {
$error = "Username or email already exists!";
}
}
}
?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Register - Contact Management System</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
<div class="container mt-5">
<div class="row justify-content-center">
<div class="col-md-6">
<div class="card">

```

```

<div class="card-header text-center">
<h3>Register</h3>
</div>
<div class="card-body">
<?php if ($error): ?>
<div class="alert alert-danger"><?php echo $error; ?></div>
<?php endif; ?>
<?php if ($success): ?>
<div class="alert alert-success"><?php echo $success; ?></div>
<?php endif; ?>

<form method="POST">
<div class="mb-3">
<label for="username" class="form-label">Username</label>
<input type="text" class="form-control" id="username" name="username" required>
</div>
<div class="mb-3">
<label for="email" class="form-label">Email</label>
<input type="email" class="form-control" id="email" name="email" required>
</div>
<div class="mb-3">
<label for="password" class="form-label">Password</label>
<input type="password" class="form-control" id="password" name="password" required>
</div>
<div class="mb-3">
<label for="confirm_password" class="form-label">Confirm Password</label>
<input type="password" class="form-control" id="confirm_password"
name="confirm_password" required>
</div>
<button type="submit" class="btn btn-primary w-100">Register</button>
</form>
<div class="text-center mt-3">

```

```

<a href="login.php">Already have an account? Login here</a>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</body>
</html>

```

```

#=====LOGIN=====
<?php
session_start();
include 'includes/db.php';

$error = "";

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = trim($_POST['username']);
    $password = $_POST['password'];

    $stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
    $stmt->execute([$username]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['username'] = $user['username'];
        header("Location: dashboard.php");
        exit();
    }
}

```

```

    } else {
$error = "Invalid username or password!";
    }
}
?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login - Contact Management System</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
<div class="container mt-5">
<div class="row justify-content-center">
<div class="col-md-6">
<div class="card">
<div class="card-header text-center">
<h3>Login</h3>
</div>
<div class="card-body">
<?php if ($error): ?>
<div class="alert alert-danger"><?php echo $error; ?></div>
<?php endif; ?>

<form method="POST">
<div class="mb-3">
<label for="username" class="form-label">Username</label>
<input type="text" class="form-control" id="username" name="username" required>

```

[illegible]

#=====DASHBOARD=====

```
<?php
session_start();
include 'includes/db.php';

if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

$user_id = $_SESSION['user_id'];
$message = "";

// Handle CSV Import
if (isset($_POST['import_csv'])) {
    if (isset($_FILES['csv_file']) && $_FILES['csv_file']['error'] == 0) {
        $file = fopen($_FILES['csv_file']['tmp_name'], 'r');
        // Skip header row
```

```

fgetcsv($file);
$imported = 0;
while (($row = fgetcsv($file)) !== false) {
    $name = isset($row[0]) ? trim($row[0]) : "";
    $email = isset($row[1]) ? trim($row[1]) : "";
    $phone = isset($row[2]) ? trim($row[2]) : "";
    $address = isset($row[3]) ? trim($row[3]) : "";
    $group_id = isset($row[4]) && $row[4] !== " ? $row[4] : null;

    if ($name != "") {
        $stmt = $pdo->prepare("INSERT INTO contacts (user_id, group_id, name, email, phone,
address) VALUES (?, ?, ?, ?, ?, ?)");
        $stmt->execute([$user_id, $group_id, $name, $email, $phone, $address]);
        $imported++;
    }
}
fclose($file);
$message = "Imported contacts imported successfully!";
} else {
    $message = "Error uploading CSV file.";
}
}

// Handle contact operations
if ($_SERVER['REQUEST_METHOD'] == 'POST' && !isset($_POST['import_csv'])) {
    if (isset($_POST['add_contact'])) {
        $name = trim($_POST['name']);
        $email = trim($_POST['email']);
        $phone = trim($_POST['phone']);
        $address = trim($_POST['address']);
        $group_id = $_POST['group_id'] ?: NULL;

        $stmt = $pdo->prepare("INSERT INTO contacts (user_id, group_id, name, email, phone,
address) VALUES (?, ?, ?, ?, ?, ?)");
        if ($stmt->execute([$user_id, $group_id, $name, $email, $phone, $address])) {
            $message = "Contact added successfully!";
        }
    }

    if (isset($_POST['update_contact'])) {
        $contact_id = $_POST['contact_id'];
        $name = trim($_POST['name']);
        $email = trim($_POST['email']);
        $phone = trim($_POST['phone']);
        $address = trim($_POST['address']);
        $group_id = $_POST['group_id'] ?: NULL;

        $stmt = $pdo->prepare("UPDATE contacts SET name=?, email=?, phone=?, address=?,

```



```

group_id=? WHERE id=? AND user_id=?");
if ($stmt->execute([$name, $email, $phone, $address, $group_id, $contact_id, $user_id])) {
    $message = "Contact updated successfully!";
}
}

if (isset($_POST['add_group'])) {
    $group_name = trim($_POST['group_name']);
    $stmt = $pdo->prepare("INSERT INTO contact_groups (user_id, group_name) VALUES (?, ?)");
    if ($stmt->execute([$user_id, $group_name])) {
        $message = "Group added successfully!";
    }
}

// Handle delete operations
if (isset($_GET['delete'])) {
    $contact_id = $_GET['delete'];
    $stmt = $pdo->prepare("DELETE FROM contacts WHERE id=? AND user_id=?");
    if ($stmt->execute([$contact_id, $user_id])) {
        $message = "Contact deleted successfully!";
    }
}

// Handle search
$search = isset($_GET['search']) ? trim($_GET['search']) : "";
$group_filter = isset($_GET['group_filter']) ? $_GET['group_filter'] : "";

// Get contacts with search and filter
$sql = "SELECT c.*, g.group_name FROM contacts c
LEFT JOIN contact_groups g ON c.group_id = g.id
WHERE c.user_id = ?";
$params = [$user_id];

if ($search) {
    $sql .= " AND (c.name LIKE ? OR c.email LIKE ? OR c.phone LIKE ?)";
    $search_param = "%$search%";
    $params[] = $search_param;
    $params[] = $search_param;
    $params[] = $search_param;
}

if ($group_filter) {
    $sql .= " AND c.group_id = ?";
    $params[] = $group_filter;
}

```

```

$sql .= " ORDER BY c.name";
$stmt = $pdo->prepare($sql);
$stmt->execute($params);
$contacts = $stmt->fetchAll();

// Get groups for dropdown
$stmt = $pdo->prepare("SELECT * FROM contact_groups WHERE user_id = ? ORDER BY
group_name");
$stmt->execute([$user_id]);
$groups = $stmt->fetchAll();

// Get contact for editing
$edit_contact = null;
if (isset($_GET['edit'])) {
    $edit_id = $_GET['edit'];
    $stmt = $pdo->prepare("SELECT * FROM contacts WHERE id=? AND user_id=?");
    $stmt->execute([$edit_id, $user_id]);
    $edit_contact = $stmt->fetch();
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Dashboard - Contact Management System</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"
rel="stylesheet">
<link href="https://cdn.datatables.net/1.11.5/css/dataTables.bootstrap5.min.css"
rel="stylesheet">
</head>
<body>
<!-- Navigation -->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<div class="container">
<a class="navbar-brand" href="#">
<i class="fas fa-address-book"></i> Contact Manager
</a>
<div class="navbar-nav ms-auto">
<span class="navbar-text me-3">Welcome, <?php echo
htmlspecialchars($_SESSION['username']); ?></span>
<a class="btn btn-outline-light btn-sm" href="logout.php">
<i class="fas fa-sign-out-alt"></i> Logout
</a>
</div>

```

```

</div>
</nav>

<div class="container mt-4">
<?php if ($message): ?>
<div class="alert alert-success alert-dismissible fade show" role="alert">
<?php echo $message; ?>
<button type="button" class="btn-close" data-bs-dismiss="alert"></button>
</div>
<?php endif; ?>

<div class="row">
<!-- Add/Edit Contact Form -->
<div class="col-md-4">
<div class="card">
<div class="card-header">
<h5><i class="fas fa-user-plus"></i> <?php echo $edit_contact ? 'Edit Contact' : 'Add New Contact'; ?></h5>
</div>
<div class="card-body">
<form method="POST">
<?php if ($edit_contact): ?>
<input type="hidden" name="contact_id" value="<?php echo $edit_contact['id']; ?>">
<?php endif; ?>

<div class="mb-3">
<label for="name" class="form-label">Name *</label>
<input type="text" class="form-control" id="name" name="name"
value="<?php echo $edit_contact ? htmlspecialchars($edit_contact['name']) : "; ?>" required>
</div>

<div class="mb-3">
<label for="email" class="form-label">Email</label>
<input type="email" class="form-control" id="email" name="email"
value="<?php echo $edit_contact ? htmlspecialchars($edit_contact['email']) : "; ?>">
</div>

<div class="mb-3">
<label for="phone" class="form-label">Phone</label>
<input type="text" class="form-control" id="phone" name="phone"
value="<?php echo $edit_contact ? htmlspecialchars($edit_contact['phone']) : "; ?>">
</div>

<div class="mb-3">
<label for="address" class="form-label">Address</label>
<textarea class="form-control" id="address" name="address" rows="3"><?php echo
$edit_contact ? htmlspecialchars($edit_contact['address']) : "; ?></textarea>
</div>

```

```

<div class="mb-3">
<label for="group_id" class="form-label">Group</label>
<select class="form-select" id="group_id" name="group_id">
<option value="">No Group</option>
<?php foreach ($groups as $group): ?>
<option value="<?php echo $group['id']; ?>"
<?php echo ($edit_contact && $edit_contact['group_id'] == $group['id']) ? 'selected' : ''; ?>>
<?php echo htmlspecialchars($group['group_name']); ?>
</option>
<?php endforeach; ?>
</select>
</div>

<button type="submit" name="<?php echo $edit_contact ? 'update_contact' : 'add_contact';
?>"
class="btn btn-primary w-100">
<i class="fas fa-save"></i> <?php echo $edit_contact ? 'Update Contact' : 'Add Contact'; ?>
</button>

<?php if ($edit_contact): ?>
<a href="dashboard.php" class="btn btn-secondary w-100 mt-2">
<i class="fas fa-times"></i> Cancel
</a>
<?php endif; ?>
</form>
</div>
</div>

<!-- Add Group Form -->
<div class="card mt-3">
<div class="card-header">
<h6><i class="fas fa-layer-group"></i> Add New Group</h6>
</div>
<div class="card-body">
<form method="POST">
<div class="input-group">
<input type="text" class="form-control" name="group_name" placeholder="Group name"
required>
<button type="submit" name="add_group" class="btn btn-outline-primary">
<i class="fas fa-plus"></i>
</button>
</div>
</form>
</div>
</div>
</div>

```

```

<!-- Contacts List -->
<div class="col-md-8">
<div class="card">
<div class="card-header">
<h5><i class="fas fa-list"></i> Your Contacts</h5>
</div>
<div class="card-body">
<!-- Import/Export CSV -->
<form      action="dashboard.php"      method="post"      enctype="multipart/form-data"
style="display:inline-block; margin-right:10px;">
<input type="file" name="csv_file" accept=".csv" required>
<button type="submit" name="import_csv" class="btn btn-success btn-sm">Import
CSV</button>
</form>
<a href="export_csv.php" class="btn btn-info btn-sm">Export CSV</a>

<!-- Search and Filter -->
<form method="GET" class="row mb-3 mt-3">
<div class="col-md-6">
<input type="text" class="form-control" name="search"
placeholder="Search contacts..." value="<?php echo htmlspecialchars($search); ?>">
</div>
<div class="col-md-4">
<select class="form-select" name="group_filter">
<option value="">All Groups</option>
<?php foreach ($groups as $group): ?>
<option value="<?php echo $group['id']; ?>"
<?php echo ($group_filter == $group['id']) ? 'selected' : ''; ?>>
<?php echo htmlspecialchars($group['group_name']); ?>
</option>
<?php endforeach; ?>
</select>
</div>
<div class="col-md-2">
<button type="submit" class="btn btn-primary w-100">
<i class="fas fa-search"></i>
</button>
</div>
</form>

<!-- Contacts Table -->
<div class="table-responsive">
<table id="contactsTable" class="table table-striped table-hover">
<thead class="table-dark">
<tr>
<th>Name</th>
<th>Email</th>
<th>Phone</th>

```

```

<th>Group</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<?php foreach ($contacts as $contact): ?>
<tr>
<td><?php echo htmlspecialchars($contact['name']); ?></td>
<td><?php echo htmlspecialchars($contact['email']); ?></td>
<td><?php echo htmlspecialchars($contact['phone']); ?></td>
<td>
<?php if ($contact['group_name']): ?>
<span class="badge bg-info"><?php echo htmlspecialchars($contact['group_name']);
?></span>
<?php else: ?>
<span class="text-muted">No Group</span>
<?php endif; ?>
</td>
<td>
<a href="?edit=<?php echo $contact['id']; ?>" class="btn btn-sm btn-warning">
<i class="fas fa-edit"></i>
</a>
<a href="?delete=<?php echo $contact['id']; ?>"
class="btn btn-sm btn-danger"
onclick="return confirm('Are you sure you want to delete this contact?')">
<i class="fas fa-trash"></i>
</a>
<button class="btn btn-sm btn-info" data-bs-toggle="modal"
data-bs-target="#viewModal<?php echo $contact['id']; ?>">
<i class="fas fa-eye"></i>
</button>
</td>
</tr>

<!-- View Contact Modal -->
<div class="modal fade" id="viewModal<?php echo $contact['id']; ?>" tabindex="-1">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title">Contact Details</h5>
<button type="button" class="btn-close" data-bs-dismiss="modal"></button>
</div>
<div class="modal-body">
<p><strong>Name:</strong> <?php echo htmlspecialchars($contact['name']); ?></p>
<p><strong>Email:</strong> <?php echo htmlspecialchars($contact['email']); ?></p>
<p><strong>Phone:</strong> <?php echo htmlspecialchars($contact['phone']); ?></p>
<p><strong>Address:</strong> <?php echo nl2br(htmlspecialchars($contact['address']));
?></p>

```

```

<p><strong>Group:</strong>      <?php      echo      $contact['group_name']      ?
htmlspecialchars($contact['group_name']) : 'No Group'; ?></p>
</div>
</div>
</div>
</div>
<?php endforeach; ?>
</tbody>
</table>
</div>
</div>
</div>
</div>
</div>
</div>

```

```

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="https://cdn.datatables.net/1.11.5/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.11.5/js/dataTables.bootstrap5.min.js"></script>
<script>
$(document).ready(function() {
$('#contactsTable').DataTable({
"pageLength": 10,
"responsive": true
});
});
</script>
</body>
</html>

```

```

#=====LOGOUT =====

```

```

<?php
session_start();
session_destroy();
header("Location: login.php");
exit();
?>

```

```
#=====EXPORT CSV=====
```

```
<?php
session_start();
include 'includes/db.php';

if (!isset($_SESSION['user_id'])) {
header("Location: login.php");
exit();
}

$user_id = $_SESSION['user_id'];

$stmt = $pdo->prepare("SELECT name, email, phone, address, group_id FROM contacts
WHERE user_id = ?");
$stmt->execute([$user_id]);
$contacts = $stmt->fetchAll(PDO::FETCH_ASSOC);

header('Content-Type: text/csv; charset=utf-8');
header('Content-Disposition: attachment; filename=contacts_export.csv');

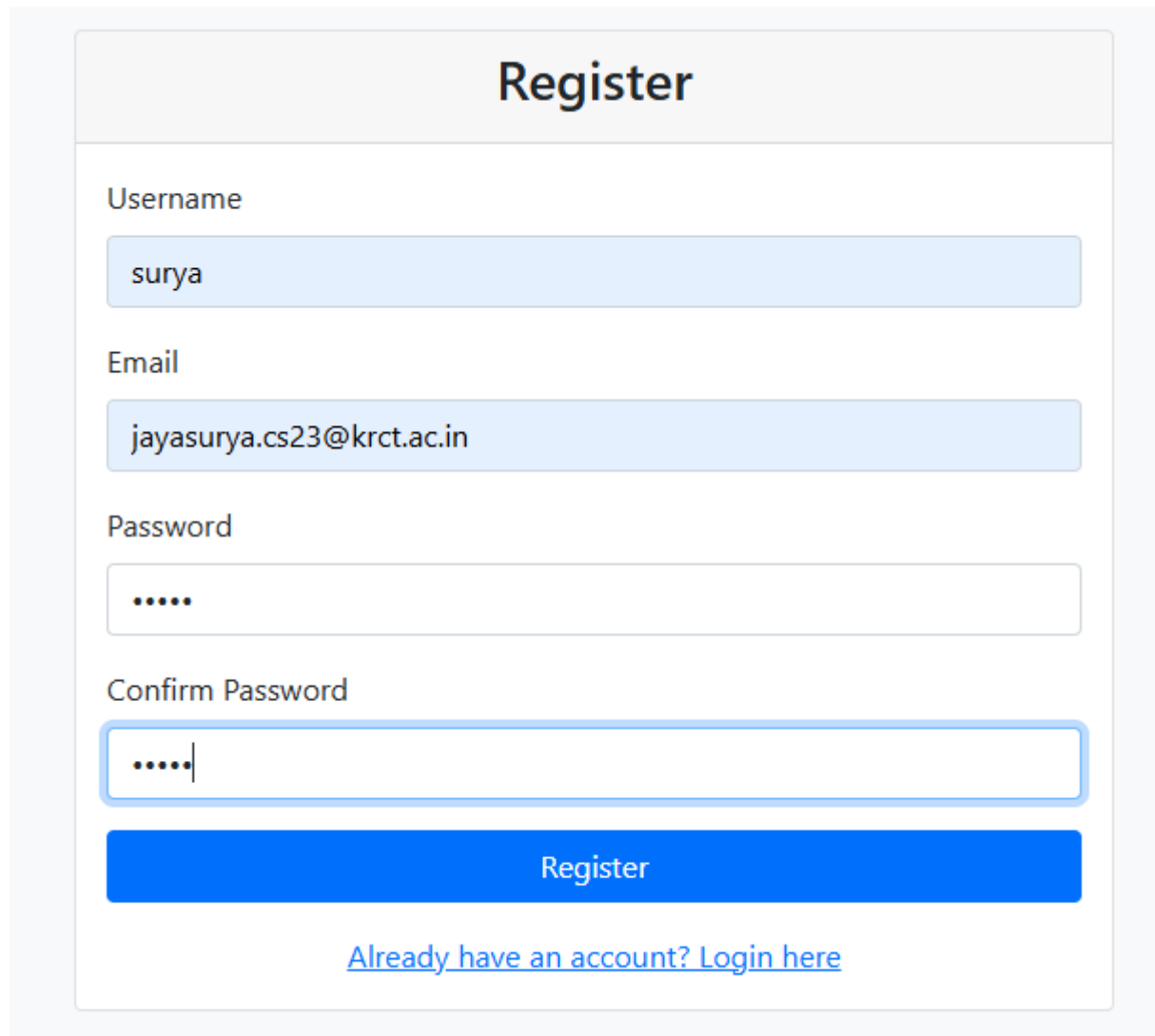
$output = fopen('php://output', 'w');

fputcsv($output, ['Name', 'Email', 'Phone', 'Address', 'GroupID']);

foreach ($contacts as $contact) {
$row = [
isset($contact['name']) ? $contact['name'] : "",
isset($contact['email']) ? $contact['email'] : "",
isset($contact['phone']) ? $contact['phone'] : "",
isset($contact['address']) ? $contact['address'] : "",
isset($contact['group_id']) ? $contact['group_id'] : ""
];
fputcsv($output, $row);
}

fclose($output);
exit();
```


APPENDIX B
(SCREENSHOTS)



The screenshot shows a web registration form with a light gray background. At the top, a white header box contains the title "Register" in bold black text. Below the header, the form is organized into sections. The "Username" section has a light blue input field containing the text "surya". The "Email" section has a light blue input field containing the email address "jayasurya.cs23@krct.ac.in". The "Password" section has a white input field with five black dots representing masked characters. The "Confirm Password" section has a white input field with five black dots and a vertical cursor line at the end. Below these fields is a prominent blue button with the text "Register" in white. At the bottom of the form, there is a blue hyperlink that reads "Already have an account? Login here".

Register

Username

surya

Email

jayasurya.cs23@krct.ac.in

Password

.....

Confirm Password

.....|

Register

[Already have an account? Login here](#)

Login

Username

surya

Password

.....

Login

[Don't have an account? Register here](#)

+ Add New Contact

Name *

Email

Phone

Address

Group

No Group

Add Contact

+ Add New Group

Group name

+

☰ Your Contacts

Choose File No file chosen

Import CSV

Export CSV

Search contacts...

All Groups



Show 10 entries

Search:

Name	Email	Phone	Group	Actions
surya	jayasurya.cs23@krct.ac.in	6383270849	boys	
surya	jayasurya.cs23@krct.ac.in	6383270849	boys	
surya	jayasurya.cs23@krct.ac.in	6383270849	boys	
surya	jayasurya.cs23@krct.ac.in	6383270849	boys	
surya	jayasurya.cs23@krct.ac.in	6383270849	boys	

Showing 1 to 5 of 5 entries

Previous 1 Next

DATABASE

The screenshot displays the phpMyAdmin web interface. On the left, a sidebar shows a database structure with a tree view containing 'New', 'contact_management', 'contacts', 'contact_groups', 'users', 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', and 'test'. The main panel shows the 'users' table selected. At the top, a status bar indicates 'Showing rows 0 - 1 (2 total. Query took 0.0012 seconds.)'. Below this, the SQL query 'SELECT * FROM `users`' is entered. A toolbar with icons for 'Show all', 'Number of rows' (set to 25), 'Filter rows' (with a search box), and 'Sort by key' (set to None) is visible. The 'Extra options' section shows a table with two rows of data:

		id	username	email	password	created_at
<input type="checkbox"/>	Edit	1	sunya	skysunya.cs23@knd.ac.in	Sunya@436	2025-05-26 22:14:00
<input type="checkbox"/>	Edit	2	karti	brad511a@gmail	\$y\$101aW5hpsC2MLOQJFkGwFRHMcPOYwbnAkwW8h...	2025-05-26 22:20:31

Below the table, there are buttons for 'Check all', 'With selected', 'Edit', 'Copy', 'Delete', and 'Export'. The 'Query results operations' section includes buttons for 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'. At the bottom, there is a 'Bookmark this SQL query' section with a 'Label' input field and a checkbox for 'Let every user access this bookmark'.

REFERENCE

1.Books:

- Database System Concepts by Abraham Silberschatz, Henry Korth - To understand relational database design, normalization, and data handling essential for contact storage.
- "HTML and CSS: Design and Build Websites" by Jon Duckett – for creating user-friendly interfaces for contact management systems.

1.Websites:

- GeeksforGeeks – Concepts on databases, GUI, and object-oriented programming.
- W3Schools – For SQL queries and database concepts.
- TutorialsPoint – Php basics