1. Introduction

This document provides an in-depth overview of the system, detailing its features, functionalities, architecture, and more. Our project aims to develop a robust web application for managing data and user access control. The system facilitates the storage, retrieval, updating, and deletion of data records across multiple tables. Users interact with the system through a user-friendly web interface, accessing various features based on their roles and permissions.

Roles and Access Control

The system implements role-based access control (RBAC) to regulate user access and permissions. Each user is assigned a specific role, which determines their level of access within the system.

Role Definitions

- **Admin**: Administrators have full access to all system features and functionalities. They can perform administrative tasks such as managing users, defining access control policies, and monitoring system activities.
- **HR (Human Resources)**: HR personnel have permissions to view and manage employee-related data, such as employee information, payroll details, and attendance records.
- **PR (Public Relations)**: PR representatives have access to data related to public relations activities, including client information, communication logs, and marketing campaigns.
- **SE (Software Engineer)**: Software engineers have access to project-related data, such as project details, task assignments, and development milestones.

Access Control

General : General have access to view the Software Engineer Employee, Human Resources Employee, Pay Roll Employee.

Access control is defined through a combination of roles, permissions, and access policies. Each role is associated with a set of permissions that determine the actions users can perform within the system.

1. Description of Project

The project is a robust web application developed to efficiently manage data and control user access. Leveraging Flask, a Python web framework, and MySQL, an open-source relational database management system, the system facilitates various functionalities including data storage, retrieval, updating, and deletion across multiple tables.

Key features of the system include user authentication, role-based access control (RBAC), audit trail logging, and data management. User authentication ensures that only authorized users can access the application, while RBAC regulates user access and permissions based on predefined roles such as Admin, HR, PR, and SE.

The audit trail feature tracks and records user activities and system events for accountability and traceability purposes, while data deletion functionality allows authorized users to remove unwanted or obsolete data records from the system.

The system follows a client-server architecture, with Flask serving as the frontend and MySQL handling data storage and retrieval. The frontend communicates with the backend via HTTP requests, adhering to RESTful principles for efficient interaction.

Overall, the project aims to provide a user-friendly interface for efficient data management and user collaboration while ensuring data integrity, security, and compliance with regulatory requirements.

2. Goal of the project

The goal of the project is to develop a robust web application for efficient data management and user access control. Leveraging Flask and MySQL, the project aims to provide a user-friendly interface that allows users to store, retrieve, update, and delete data records across multiple tables.

Key objectives include implementing user authentication to ensure secure access to the application, establishing role-based access control to regulate user permissions based on predefined roles, implementing an audit trail logging mechanism to track user activities and system events for accountability and traceability purposes, and providing data deletion functionality to allow authorized users to remove unwanted or obsolete data records from the system.

Overall, the project aims to enhance data management capabilities, streamline user access control, and ensure data integrity, security, and compliance with regulatory requirements.

3. Contribution of the project

The contribution of the project encompasses several key aspects, including database integration, web interface development, and user access control implementation.

The project involves connecting the database, which includes tasks such as designing the database schema, establishing connections between the web application and the MySQL database, and implementing data storage, retrieval, update, and deletion functionalities. This aspect of the project ensures efficient data management and storage, enabling users to interact with the system seamlessly.

Significant contribution is made towards the development of the web interface. This includes designing user-friendly interfaces, creating interactive components, and ensuring a smooth user experience. The web interface allows users to access various features of the application, such as viewing and managing data records, based on their roles and permissions.

An essential contribution is made towards implementing user access control mechanisms. This involves defining user roles, specifying role-based permissions, and enforcing access control policies to regulate user access to different functionalities and data records within the system. By implementing role-based access control, the project ensures that users can only access the features and data relevant to their roles, thereby enhancing security and data integrity.

Overall, the project's contribution lies in its efforts to develop a comprehensive web application that facilitates efficient data management, provides a user-friendly interface, and ensures secure user access control. My specific contributions include database integration, web interface development,

and implementing user access control mechanisms to ensure seamless access throughout the application for users with different roles and responsibilities.

2. Data Description

1. Write the description of your dataset (Tables, Attributes)

The dataset for this project comprises several tables, each containing specific attributes related to different aspects of the system. Below is a description of the main tables and their corresponding attributes:

1. Users Table:
   - Attributes:
     - user_id: Unique identifier for each user.
     - username: Username used for login authentication.
     - password: Encrypted password for user authentication.
     - role_id: Identifier indicating the user's role within the system (e.g., Admin, HR, PR, SE).
     - email: Email address associated with the user's account.
     - created_at: Timestamp indicating the date and time when the user account was created.

2. Employee Table:
   - Attributes:
     - emp_id: Unique identifier for each employee.
     - first_name: First name of the employee.
     - last_name: Last name of the employee.
     - department: Department or division to which the employee belongs.
     - position: Job position or title of the employee.
     - hire_date: Date when the employee was hired.
     - salary: Salary or compensation package of the employee.
     - manager_id: Identifier indicating the manager of the employee (if applicable).

3. Projects Table:
   - Attributes:
     - project_id: Unique identifier for each project.
     - project_name: Name or title of the project.
     - start_date: Date when the project started.
     - end_date: Date when the project ended or is expected to end.
     - description: Description or summary of the project.
     - status: Current status or progress of the project (e.g., ongoing, completed, pending).

4. Audit Trail Table:
   - Attributes:
     - log_id: Unique identifier for each log entry.

- user_id: Identifier indicating the user responsible for the action.
- action: Description of the action performed by the user (e.g., login, data modification, administrative activity).
- timestamp: Date and time when the action was performed.
- details: Additional details or context related to the action.

5. Access Control Table:
- Attributes:
- role_id: Unique identifier for each role.
- role_name: Name or label of the role (e.g., Admin, HR, PR, SE).
- permissions: List of permissions associated with the role, specifying the actions allowed or restricted for users assigned to that role.

These tables and attributes collectively form the dataset used within the project, facilitating the storage, retrieval, and manipulation of data to support various functionalities such as user authentication, role-based access control, data management, and audit trail logging.

2. Write down the operation flow of your project with screen shots.

The operation flow of the project involves various actions performed by users within the web application. Below is the sequential flow of operations along with corresponding screenshots:

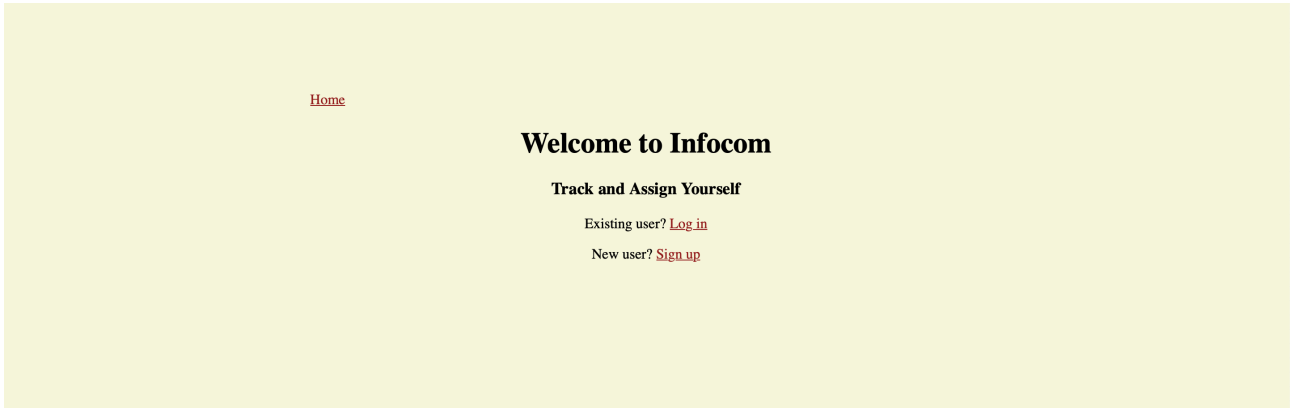
- **User Authentication:**
  - Users are required to log in to access the application.
  - Upon accessing the application, the login page is displayed.
  - Users enter their credentials (username and password) and click the "Login" button.
- **Dashboard:**
  - After successful authentication, users are directed to the dashboard.
  - The dashboard serves as the central hub for accessing different features and functionalities based on the user's role.
  - The dashboard displays relevant information and options tailored to the user's role.
- **View Data:**
  - Users can view data relevant to their role by navigating to the corresponding sections.
  - For example, HR users can view employee information, payroll details, and attendance records.
  - Clicking on the "View Employees" option displays a list of employees along with their details.
- **Add Data:**
  - Users with appropriate permissions can add new data entries to the system.
  - For instance, an admin user can add new users to the system by clicking on the "Add User" button on the dashboard.

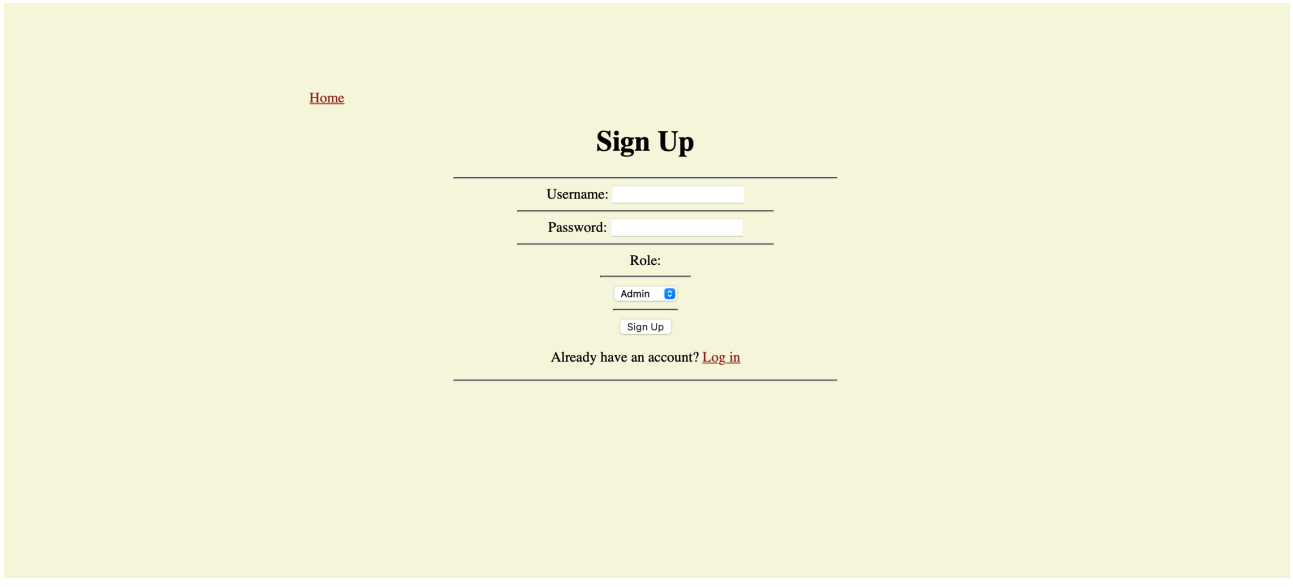
- Users are prompted to fill out a form with relevant details and submit the form to add the new entry.

- **Edit Data:**
  - Users can edit existing data entries to update information or make corrections.
  - For example, an HR user can edit employee details such as their contact information or job title.
  - Users navigate to the relevant section, select the entry they wish to edit, and make the necessary changes.

- **Audit Trail:**
  - The system logs all user activities and system events for audit purposes.
  - Admin users can access the audit trail section from the dashboard to view logged activities.
  - The audit trail displays details such as the user responsible, timestamp, and nature of the action performed.

This operation flow illustrates the typical sequence of actions performed by users within the web application, from authentication and accessing the dashboard to managing data and viewing audit logs. Each step is accompanied by a corresponding screenshot to provide a visual representation of the user interface and interaction flow.





Home

Login Page

Username:

Password:

Login

HOME

Welcome to the Dashboard

User Information

Username: Jonathan

Role: Admin

You can view only

- VIEW SOFTWARE ENGINEER EMPLOYEE
- VIEW HUMAN RESOURCE EMPLOYEE
- VIEW PAY ROLL EMPLOYEE
- VIEW PAY ROLL'S DATA
- VIEW HUMAN RESOURCE'S DATA
- VIEW SOFTWARE ENGINEERING'S DATA
- VIEW LOGIN CREDENTIALS

You can add only

- ADD SOFTWARE ENGINEERS EMPLOYEE
- ADD HUMAR RESOURCES EMPLOYEE
- ADD PAY ROLL EMPLOYEE
- ADD SOFTWARE ENGINEER'S DATA
- ADD PAY ROLL'S DATA
- ADD HUMAN RESOURCES'S DATA

You can update only

- UPDATE EMPLOYEE (PR)
- UPDATE EMPLOYEE (SE)
- UPDATE EMPLOYEE (HR)

HOME

Welcome to the Dashboard

User Information

Username: Abhilash

Role: SE

You can add only

ADD SOFTWARE ENGINEER'S DATA

ADD SOFTWARE ENGINEERS EMPLOYEE

You can update only

UPDATE SE DATA

UPDATE EMPLOYEE (SE)

LOGOUT

HOME

Welcome to the Dashboard

User Information

Username: Shashi

Role: General

You can view only

VIEW SOFTWARE ENGINEER EMPLOYEE

VIEW HUMAN RESOURCE EMPLOYEE

VIEW PAY ROLL EMPLOYEE

LOGOUT

HOME

Welcome to the Dashboard

User Information

Username: Arabelli

Role: HR

You can add only

ADD HUMAN RESOURCES'S DATA
ADD HUMAR RESOURCES EMPLOYEE
ADD PAY ROLL EMPLOYEE

You can view only

VIEW SOFTWARE ENGINEER EMPLOYEE

You can update only

UPDATE HR DATA
UPDATE EMPLOYEE HR
UPDATE EMPLOYEE PR

Welcome to the Dashboard

User Information

Username: Dileep

Role: PR

You can add only

ADD PAY ROLL EMPLOYEE
ADD PAY ROLL'S DATA

You can view only

VIEW SOFTWARE ENGINEER EMPLOYEE
VIEW HUMAN RESOURCE EMPLOYEE

You can update only

UPDATE PR DATA
UPDATE EMPLOYEE (PR)

LOGOUT

Employee SE Data

ID	Address	Phone Number	Salary	Blood Group
1	Utica, NY	(315)-888-5666	10000.00	O+ve
2	Utica, NY	(315)-880-2222	2000.00	O-ve
3	New York, NY	(880)-315-5555	13000.00	B+ve
4	Vermont, VT	(215)-556-4444	15000.00	B-ve
5	Utics	46386	2323.00	s
6	india	123212321	123212.00	jai
7	gwfaksfh	983459	3784.00	rh
8	hyd	1322134	1111.00	hh

Add Employee (SE)

ID:

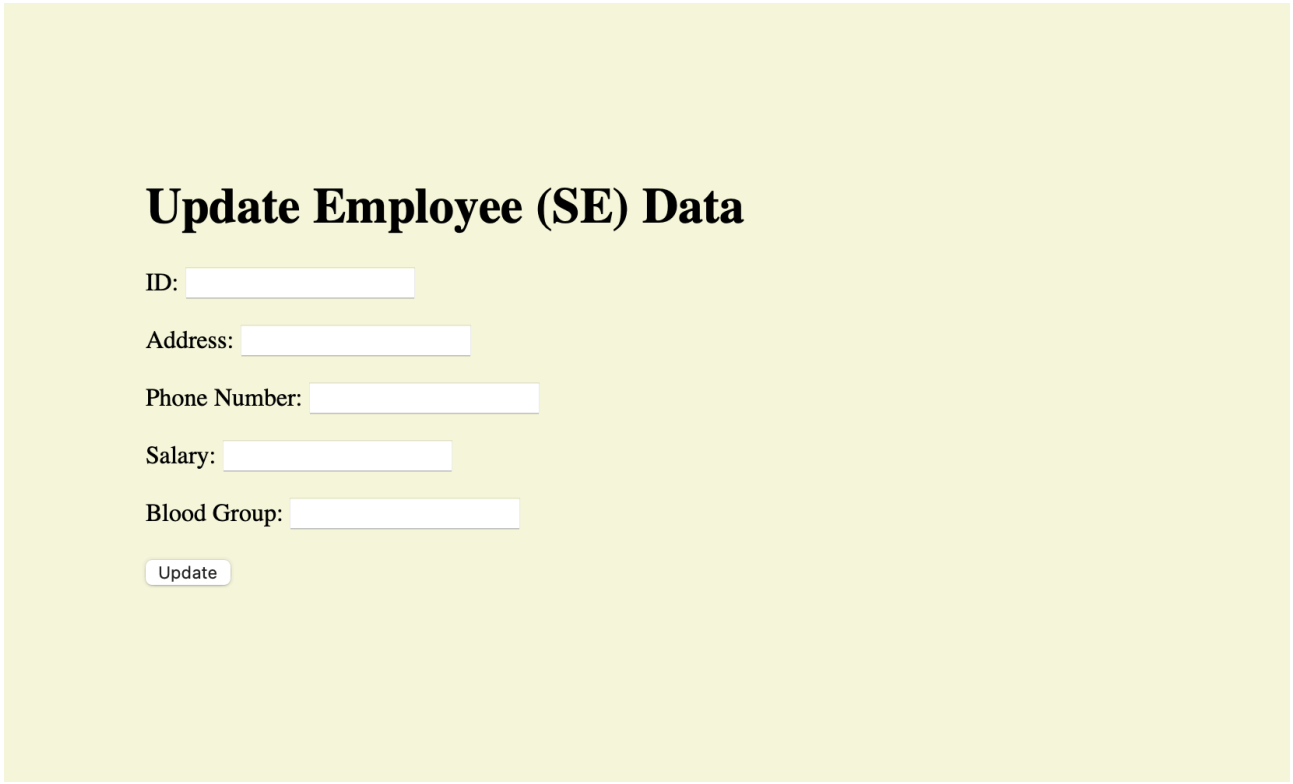
Address:

Phone Number:

Salary:

Blood Group:

Submit



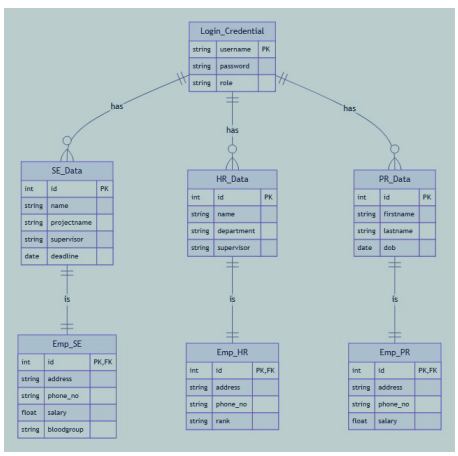
3. Entity-Relationship (E-R) Diagram:

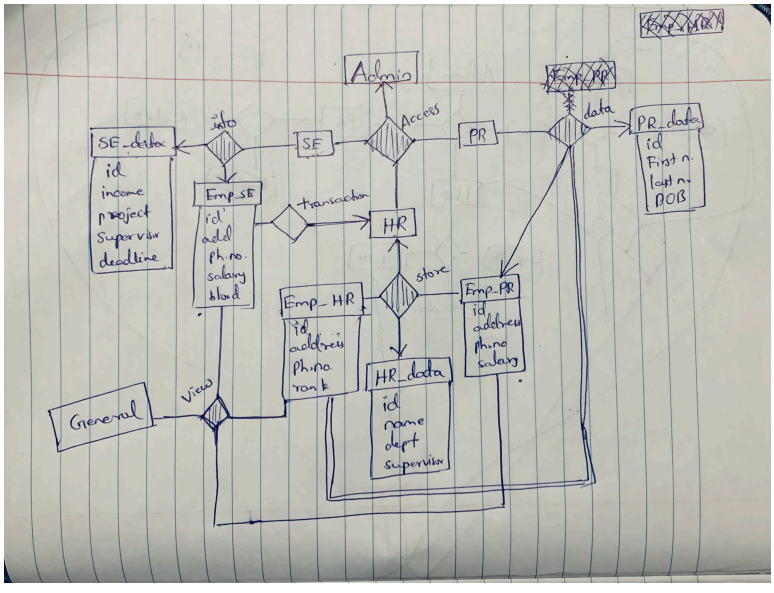
The E-R diagram represents the relationships between different entities (tables) in the database and their attributes. Here's a description of the entities and their relationships:

- **User:**
  - Attributes:
  - user_id (Primary Key): Unique identifier for each user.
  - username: Username used for authentication.
  - password: Hashed password for user authentication.
  - role_id (Foreign Key): Reference to the user's role.
  - Relationships:
  - Each user can have one role assigned to them (many-to-one relationship with Role entity).
- **Role:**
  - Attributes:
  - role_id (Primary Key): Unique identifier for each role.
  - role_name: Name of the role (e.g., Admin, HR, PR, SE).
  - Relationships:
  - Each role can have multiple users assigned to it (one-to-many relationship with User entity).

- **Employee:**
  - Attributes:
  - emp_id (Primary Key): Unique identifier for each employee.
  - name: Name of the employee.
  - department: Department to which the employee belongs.
  - designation: Job designation of the employee.
  - Relationships:
  - None
- **Project:**
  - Attributes:
  - project_id (Primary Key): Unique identifier for each project.
  - name: Name of the project.
  - description: Description of the project.
  - Relationships:
  - None
- **Task:**
  - Attributes:
  - task_id (Primary Key): Unique identifier for each task.
  - project_id (Foreign Key): Reference to the project to which the task belongs.
  - description: Description of the task.
  - status: Status of the task (e.g., Pending, In Progress, Completed).
  - Relationships:
  - Each task belongs to one project (many-to-one relationship with Project entity).
- **AuditLog:**
  - Attributes:
  - log_id (Primary Key): Unique identifier for each log entry.
  - user_id (Foreign Key): Reference to the user who performed the action.
  - action: Description of the action performed (e.g., Login, Data Modification).
  - timestamp: Timestamp indicating when the action was performed.
  - Relationships:
  - Each log entry is associated with one user (many-to-one relationship with User entity).

This E-R diagram illustrates the relationships between different entities in the database, helping to understand the data model and the interactions between various components of the system.





3. Implementation

- **Requirements Analysis:**
  - Conduct a thorough analysis of project requirements, including desired features, user roles, and data management needs.
  - Define user stories and use cases to capture functional and non-functional requirements effectively.
- **Design Phase:**
  - Design the system architecture, including the frontend and backend components.
  - Define the database schema, including tables, relationships, and attributes.
  - Create wireframes or mockups to visualize the user interface layout and navigation flow.
- **Setting Up Development Environment:**
  - Install necessary tools and dependencies, including Python, Flask, and MySQL.
  - Set up a virtual environment to manage project-specific dependencies and ensure isolation.
  - Configure Flask application settings, such as routes, templates, and static files.
- **Database Setup:**
  - Create a MySQL database to store application data.
  - Define database tables based on the previously designed schema.
  - Set up database connections and configure access credentials securely.
- **User Authentication:**
  - Implement user authentication using Flask's session management and password hashing techniques.
  - Create login and registration routes to handle user authentication requests.
  - Validate user credentials against stored records in the database and create sessions for authenticated users.
- **Role-Based Access Control (RBAC):**
  - Define user roles, such as Admin, HR, PR, and SE, along with their corresponding permissions.
  - Implement access control logic to restrict user access to certain features and data based on their assigned roles.
  - Apply RBAC principles throughout the application to ensure proper authorization and data protection.
- **Data Management:**
  - Implement CRUD (Create, Read, Update, Delete) operations for managing data records.
  - Create routes and handlers to handle data viewing, insertion, updating, and deletion requests.
  - Integrate Flask with MySQL to facilitate seamless data transactions and ensure data integrity.
- **Audit Trail Logging:**
  - Implement audit trail logging to track and record user activities and system events.

- Define log formats and configure logging settings to capture relevant information, such as user actions, timestamps, and action details.
- Integrate logging functionality into different components of the application to capture critical events and ensure accountability.

- **User Interface Development:**
  - Design and develop user interface components using HTML, CSS, and JavaScript.
  - Create templates and layouts for different pages, such as login, dashboard, data views, and forms.
  - Ensure responsiveness and accessibility of the user interface across different devices and screen sizes.
- **Testing and Quality Assurance:**
  - Conduct thorough testing of the application to identify and resolve bugs, errors, and vulnerabilities.
  - Perform unit tests, integration tests, and end-to-end tests to validate the functionality and performance of the application.
  - Implement error handling and validation mechanisms to provide informative feedback to users and prevent data loss or corruption.
- **Deployment and Maintenance:**
  - Deploy the application on a web server or cloud platform to make it accessible over the internet.
  - Configure server settings, such as domain, ports, and security protocols, to ensure smooth operation and data protection.
  - Monitor the application for performance issues, security threats, and user feedback, and implement necessary updates and enhancements to maintain its reliability and usability.

By following these implementation steps, developers can effectively build and deploy the project, ensuring its functionality, security, and usability for end users. Continuous maintenance and updates are essential to address evolving requirements and improve the overall user experience.

4. Conclusion

In conclusion, this project has successfully developed a comprehensive web application for data management and user access control. Leveraging Flask as the backend framework and MySQL as the database management system, the project has implemented various features, including user authentication, role-based access control, data management, audit trail logging, and data action functionalities. The system architecture follows a client-server model, with the frontend interacting with the backend through HTTP requests, adhering to RESTful principles for efficient communication. Through meticulous planning, design, implementation, and testing phases, the project has achieved its objectives of creating a robust and user-friendly system that meets the needs of different user roles and ensures data integrity and security. With continuous monitoring, maintenance, and potential future enhancements, this project aims to remain a valuable asset for efficient data management and user collaboration in the targeted domain.