**CONTACT MANAGEMENT SYSTEM**

## Advance Data Base

## 603-C

#### Wizards at work

A picture containing logo

Description automatically generated

Sacred Heart University

School of Computer Science & Engineering

The Jack Welch College of Business &Technology

Submitted To:

Dr. Reza Sadeghi

Spring 2022

### **Phase 4 Report of Contact Management System**

### **Team Name**

Wizards at Work

### **Team Members**

1. Prudhvi Sai Akhil Thumu thumup@mail. sacredheart.edu (Team Head)
2. Sandeep Yepuri yepuris@mail.sacredheart.edu (Team Member)
3. Nikhilender Reddy Baddam baddamn2@ mail.sacredheart.edu(Team Member)
4. Chetan Sai Tallamudi tallamudic@mail.sacredheart.edu (Team Member)

### **Roles of Team Members**

1. Prudhvi sai akhil Thumu.

I’m Prudhvi sai akhil Thumu, Graduate Computer Science student at Sacred Heart University.

Completed my Under graduation from Gitam University in India and, I acquired experience in IT industry as a cloud DevOps Engineer. My role in this project is to be Monitoring the team activities as a team lead, involving in the development of GUI, API and Database system for Contact management System and Creating Different Tables which are essential for the Contact Management System (CMS).

1. Sandeep Yepuri.

I’m Sandeep Yepuri, Graduate Computer Science student at Sacred Heart University.

Completed my Under graduation from KLU University in India and, I acquired experience in IT industry as a full stack developer. My role in this project to develop UI interface for contact management system and will share some work in API.

3.Chetan Sai Tallamudi.

I’m Chetan Sai Tallamudi, Graduate Computer Science student at Sacred Heart University. Completed my Under graduation from St Joseph in India. My role in this project to develop API in order to connect the GUI with backend database and will do some part of work in database.

1. Nikhilender Reddy Baddam.

I’m Nikhilender Reddy Baddam, Graduate Computer Science student at Sacred Heart

University. Completed my Under graduation from VNR VJIET institute of Technology

in India. I acquired experience in IT industry as a Python developer. My role in this project to develop the database with different tables for the Contact management System using the different tables in MySQL.

## Table of Contents

Table of Figures…………………………………………………………………………………………………...5

Project description………………………………………………………………………………………………...6-7

ER Diagram Description………………………………………………………………………………………….8-10

Database Description…………………………………………………………………………………………….11-14

Importing data, manipulating data, and optimizing database……………………………………………………14-20

Graphical User Interface………………………………………………………………………………………...21-25

GitHub Repository Address…………………………………………………………………………………….....26

References…………………………………………………………………………………………………………26

## Table of Figures

Figure1: ER Diagram……………………………………………………………………………………………....8

Figure2: EER Diagram……………………………………………………………………………………………..9

Figure3: Execution Plan……………………………………………………………………………………………19

Figure 4: CMS login Page………………………………………………………………………………………… 21

Figure 5: Admin Main menu Page…………………………………………………………………………………22

Figure 6: Admin Main menu Page…………………………………………………………………………………23

Figure 7: Action page to add a user………………………………………………………………………………..24

**Project Description**

Contact Management System (CMS) stores different types of information such as Telephone number, Mailing Address, Contact name, Phone number, Fax Number, Home number and Address. The CMS will store the user data in distinct SQL tables with different user types.

* CMS can ask the admin username and password to login and password should contain 8 characters. CMS can be able to change the admin user and Password.
* Admin user or root user can add new user to the CMS by adding the username and password to the database which cannot done by the normal user.
* Admin user can remove user from the CMS by removing username, password and any other related data.
* Every user can be able to add the contact information like first name, sur name, phone number, workplace number, workplace address, home address, zip code, fax number, email address, gender and age.
* Every user can be able to remove the contact information
* Every user can be able to edit the contact details
* CMS can display the information using various search options where users are able to search with contact number to get other details like name and mail address
* CMS graphical user interface (GUI) is very user friendly where it can show the warnings where the user is trying to put the contact information which is exist in tables.
* CMS GUI shows a beautiful welcome page
* CMS GUI can show all available options and functions to the end user
* CMS can show the reports in the tabular form
* CMS can be able to provide the exit functionality on the GUI

**ER Diagram Description:**

**Diagram

Description automatically generated**

**Figure 1:** ER Diagram for CMS

1. Partial Participation - User table
2. Total Participation – Customer table
3. Multivalued attribute – emails and contacts (single user may have multiple emails and multiple contacts)
4. Composite attribute – Name (F\_name, M\_name, L\_name)
5. Derived attribute – age of the user can be derived from the date of birth of the user.
6. Weak entities – Email table and contacts table

**Diagram

Description automatically generated**

**Figure 2:** EER Diagram for CMS

ER diagram for CMS consists of 10 tables which are stated below

1. Login (login\_id, login\_role\_id, login\_username, user\_password)
2. User (user\_id, user\_name, user\_mobile, user\_address, user\_email, user\_rolid, user\_perid)
3. Roles (role\_id, role\_name, role\_desc)
4. Address (add\_place, add\_desc,add\_zip)
5. Mobile (mobile\_des, mob\_num)
6. Permission (per\_id, per\_role\_id, per\_name, per\_module)
7. Contacts (con\_num, con\_des)
8. Company (company\_id, company\_name, company\_add, company\_num, company\_mail, company\_lev)
9. Email (per\_email,office\_mail)
10. Customer (cus\_id, cus\_name, cus\_mobile, cus\_email, cus\_add, cus\_comid, company\_com\_id, cus\_dob,cus\_age)

All the tables shown above are to implement the database for Contact Management System (CMS). Data which is going to store inside the CMS database is purely end user information who are working in different organizations and having different roles. For CMS in order to maintain the relationship between among the entities in the tables used one to many relationship and one to one relationship.

**Database Description:**

This Database is designed as per the EER model diagram which is shown above. All the tables and entities are created with the help SQL queries Containing Primary Keys, foreign keys and Null keys.

In all most all the tables Primary keys (Pkey) and Foreign Keys (Fkey) are defined which are having a unique value for each column.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Query** | **EER Model for Table** | **Description** | **Pkey** | **Fkey** |
| user | CREATE TABLE `user` (  user\_id int NOT NULL,  user\_name varchar(30) NOT NULL,  user\_mobile INT NOT NULL,  user\_email varchar(30) NOT NULL,  user\_address varchar(45) NOT NULL ,  role\_id INT NOT NULL,  per\_id INT NOT NULL,  login\_id int NOT NULL,  ssn\_num varchar(9) NOT NULL,  PRIMARY KEY (`user\_id`),  FOREIGN KEY (role\_id) REFERENCES ROLES(role\_id),  FOREIGN KEY (login\_id) REFERENCES login(login\_id),  FOREIGN KEY (per\_id) REFERENCES PERMISSION(per\_id)); |  | The purpose of User table is to manage the customer's data | Yes | Yes |
| customer | CREATE TABLE `customer` (  cus\_id int NOT NULL,  cus\_name varchar(45) NOT NULL,  cus\_mobile INT NOT NULL,  cus\_email varchar(45) NOT NULL,  cus\_dob varchar(45) NOT NULL,  cus\_age varchar(9) NOT NULL,  com\_id INT NOT NULL,  PRIMARY KEY (`cus\_id`),  FOREIGN KEY (com\_id) REFERENCES company(com\_id)); |  | The Purpose of customer table is to store the customer data. | Yes | Yes |
| manage | CREATE TABLE `manage` (  cus\_id int NOT NULL,  cus\_name varchar(45) NOT NULL,  email\_id varchar(45) NOT NULL,  add\_place varchar(45) NOT NULL,  com\_id INT NOT NULL,  mob\_num int NOT NULL,  user\_id INT NOT NULL,  FOREIGN KEY (cus\_id) REFERENCES customer(cus\_id),  FOREIGN KEY (email\_id) REFERENCES email(email\_id),  FOREIGN KEY (add\_place) REFERENCES address(add\_place),  FOREIGN KEY (com\_id) REFERENCES company(com\_id),  FOREIGN KEY (com\_id) REFERENCES company(com\_id),  FOREIGN KEY (mob\_num) REFERENCES contacts(mob\_num),  FOREIGN KEY (user\_id) REFERENCES user(user\_id)); |  | The purpose of manage table is for user to manage the customer data. | No | Yes |
| company | CREATE TABLE `company` (  com\_id int NOT NULL,  com\_name varchar(45) NOT NULL,  com\_email varchar(45) NOT NULL,  com\_add varchar(45) NOT NULL,  com\_num int NOT NULL,  PRIMARY KEY (`com\_id`)); |  | The purpose of company table is to add or manipulate the customer's company data. | Yes | No |
| permission | CREATE TABLE `permission` (  per\_id INT NOT NULL,  per\_role\_id varchar(45) NOT NULL,  per\_module varchar(45) NOT NULL,  per\_name varchar(45) NOT NULL,  PRIMARY KEY (`per\_id`)); |  | The purpose of Permissions table to assign different permissions to end user. | Yes | No |
| login | CREATE TABLE `login` (  login\_id int NOT NULL,  login\_username varchar(45) NOT NULL,  login\_password varchar(45) NOT NULL,  PRIMARY KEY (`login\_id`)); |  | The purpose of Login table is to store the login credentials like username and password etc | Yes | No |
| roles | CREATE TABLE `roles` (  role\_id INT NOT NULL,  role\_name varchar(45) NOT NULL ,  role\_desc varchar(45) NOT NULL ,  PRIMARY KEY (`role\_id`)); |  | The purpose of roles table is to assign the different roles to user like admin, developer etc. | Yes | No |
| address | CREATE TABLE `address` (  add\_place varchar(45) NOT NULL,  add\_desc varchar(45) NOT NULL,  PRIMARY KEY (`add\_place`)); |  | The purpose of address table is to add or manipulate the customer's address. | Yes | No |
| email | CREATE TABLE `email` (  email\_id varchar(45) NOT NULL,  office\_email varchar(45) NOT NULL,  PRIMARY KEY (`email\_id`)); |  | The purpose of email table is to add or manipulate the customer's email. | Yes | No |
| contacts | CREATE TABLE `contacts` (  mob\_num int NOT NULL,  con\_des varchar(45) NOT NULL,  PRIMARY KEY (`mob\_num`)); |  | The purpose of contacts table is to add or manipulate the customer's contacts. | Yes | No |
| mobile | CREATE TABLE `contacts` (  mob\_num int NOT NULL,  con\_des varchar(45) NOT NULL,  PRIMARY KEY (`mob\_num`)); |  | The purpose of mobile table is to add or manipulate the customer's contacts. | Yes | No |

**Importing data, manipulating data, and optimizing database:**

1. **Importing data:**

we followed the traditional method while importing the data into the tables without having the foreign keys in the beginning of tabular design.

First, we have created the tables which are not having foreign keys and after that we have created the other tables having the foreign keys.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Query** | **Description** |
| User | **Insert into user(user\_id,user\_name,user\_mobile,user\_email,user\_address,role\_id,per\_id,login\_id,ssn\_num)values**  **(1,'Sandeep',2,'sandeep214@gmail.com','186, lincoln',3,1,91222,'564776'),**  **(2,'Akhil',2,'akhndeep214@gmail.com','186, lincoln',3,1,91222,'564776'),**  **(3,'NIkhil',2,'nikhil@gmail.com','186, lincoln',3,1,91222,'564776'),**  **(4,'Chethan',4,'chethan14@gmail.com','186, lincoln',4,1,91222,'564776');** | Here we are inserting the values into the User table. Our application has different kinds of user and user will have different roles according to their permissions. |
| Login | **insert into login(login\_id,login\_username,login\_password) values('32354','sacreadheart','SHU@345'),**  **('90282','kites','Pru@345'),**  **('67533','university','Ak@6323'),**  **('63276','posst','mypass'),**  **('821629','kiytes','uni@1P'),**  **('91222','leosa','post%$'),**  **('78632','lionds','Peudjj@sj'),**  **('92329','school','Liqjs'),**  **('83728','hshes','Insjs'),**  **('656757','yrsyg','$^uyt');** | Here we are inserting the data into the login table. In this table we have information like login ids, username, and password. |
| Permission | **Insert into permission(per\_id, per\_role\_id, per\_module, per\_name)**  **values (1,2,'ALL','Access to all modules/pages'),(2,3,'Limited','Access to only some pages'),(3,4,'No Permission','No Access to any modules'),**  **(4,2,'hello','Access to all modules/pages'),**  **(5,3,'Limited','Access to only some pages'),**  **(7,8,'No Permission','No Access to any modules'),**  **(9,2,'ALL','Access to all modules/pages'),**  **(10,2,'Limited','Access to only some pages'),**  **(11,4,'No Permission','No Access to any modules');** | Here we are inserting the data into the permission table, and we have data like permission id, permission role, permission name. There will be different permissions for different users. |

1. **Manipulating the data:**

In this section we manipulated the data which exist in the tables with the help of ALTER and UPDATE commands.

Using ALTER command, we created a new column in the user table and updating the new column with some data with the help of UPDATE command.

**Table Name**: User

Before Manipulation of data:

Graphical user interface, application

Description automatically generated

Query with Alter:

Here we are adding the new column to the user table.

**Alter table user**

**add New\_SSN varchar(10);**

After Manipulation of data with Alter:

Graphical user interface, application

Description automatically generated

Query with Update:

Here we are assigning the value to the new column.

**Update user**

**set New\_SSN = '98273792'**

**where user\_id =2;**

After Manipulation of data with Update:

Graphical user interface, application

Description automatically generated

**Table Name**: User

Before Manipulation of data:

Graphical user interface, text, application

Description automatically generated

**Query:**

By using this command, we are renaming the role\_desc to role\_detail.

**Alter table roles**

**rename column role\_desc to role\_detail;**

**After Manipulation of data:**

**Graphical user interface, text, application

Description automatically generated**

**Query with update:**

By using this command, we are changing the roles.

**Update roles**

**set role\_name = 'Admin', role\_detail= 'Adminstrator'**

**where role\_id =4;**

After Manipulation of data with Update: **Graphical user interface, text, application

Description automatically generated**

**C**. **Optimizing the data:**

In the section of data optimization, we joined two different tables which are having the same column name (foreign key) with help of Inner Join and Group By command.

Using the count function, we counted the number of records presented for each user in the roles table and new column having the number of records.

Using Group by command, joined the existing table username and the newly created table number of roles.

User table**:**

Graphical user interface, application

Description automatically generated

Roles table:

Graphical user interface, text, application

Description automatically generated

Query:

Here we are evaluating by using the **GROUPBY** and **LEFT JOIN**.

**SELECT user.user\_name,COUNT(roles.role\_id) AS NumberOfRoles FROM roles**

**LEFT JOIN user ON roles.role\_id = user.role\_id**

**GROUP BY user\_name;**

Evaluated results**:**

**Table

Description automatically generated**

**Execution Plan:**

Below diagram represents the way how it joined the two different tables user and roles.

Diagram

Description automatically generated

Figure3: Execution Plan

**Query statistics:**

Checking the optimization by inserting the single instance and multiple instances.

Query:

**Insert into roles(role\_id, role\_name, role\_detail)**

**values**

**(5,'Admin','Adminstrator');**

**Execution time with one set of values:**

**Graphical user interface, text, application

Description automatically generated**

**Execution time with multiple set of values:**

**Graphical user interface, text, application

Description automatically generated**

**Graphical User Interface:**

1. **Login Page**

**Graphical user interface

Description automatically generated**

Figure 4: CMS login Page

Above figure illustrates the Login page of Contact Management System (CMS),

Which Contains the two login buttons for two respective users admin and user.

Admin have the privileges to create a user, alter the details of user and delete the user from CMS.

In order to login to the CMS dashboard, user should be created by admin and provides the username and password to the end user.

Once the user gets the Username and password, CMS allows the user to change the password as the per the user requirement and user should be able to add the required information.

**Code for Login page:**

**B. Main Menu Page:**

Chart, funnel chart

Description automatically generated

Figure 5: Admin Main menu Page

The above figure illustrates Admin main menu page containing two buttons to create a user and delete a user namely Add a new user and Delete a User.

After successful admin login, the admin main menu page is displayed where admin is allowed to create a new user which results in creating a new user in contact management system (CMS) and have the access to delete a user which results in deleting the existing user.

The user got deleted from contact management system (CMS) is no longer is able to login to the CMS dashboard.

Chart, funnel chart

Description automatically generated

Figure 6: User Main menu Page

The above figure illustrates User main menu page containing three buttons to create a user and delete a user namely View your contacts, add a contact, Delete contact.

After successful user login, the user main menu page is displayed where user is allowed to view a user which results in viewing the contacts in contact management system (CMS) and have the access to add a contact which results in adding the contacts.

User has a access to delete the contact which results in deleting the existing contacts.

The user got deleted from contact management system (CMS) is no longer can login to the CMS dashboard.

**Code for Main menu Page:**

**C. Action Pages:**

**Graphical user interface, application

Description automatically generated**

Figure 7: Action page to add a user

The above figure illustrates action page to add a user with admin in contact management system (CMS).

This page contains some textboxes where its required to enter the user information like user name, user mobile, user email, user address and user password.

Mainly there are two buttons Submit and Quit presented in this page. Once the admin user enters the valid details of the end user and clicks on submit will create a user and records the user details in the database.

The purpose of using the quit button is to terminate the user creation. Once admin clicks on the Quit button admin will navigated back to the admin main menu page.

**Code for Action Page:**

**GitHub Repository Address**

GitHub Repository address for the Contact Management System (CMS) is the following: <https://github.com/prudhviakhil619/contact-management-system>

**References**

* [https://monday.com/blog/project-management/contact-management-database/](https://monday.com/blog/project-management/contact-management-database/%20)
* <https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_groupby1>
* <https://www.sqlshack.com/query-optimization-techniques-in-sql-server-tips-and-tricks/>
* CMS logo: https://soffront.com/blog/features-free-contact-management-software/