Animal Training Management System

By:Raja Kumail Ali Nasir

Registeration:241201009



**Content**

Page 3: Problem statement and Requirements

Page 4: ERdiagram

Page 5: Relational Schema

Page 6-10: Database design

Page 11: DDL and DML code snippit

Page 12-13: Application interface(ui) and features

Page 13-14: Testing

Page 15: Chalenges and learning

Page 15: Conclusion and Future improvement

**Problem Statement**

There is an organization responsible for training animals.

They are struggling in managing the organization.

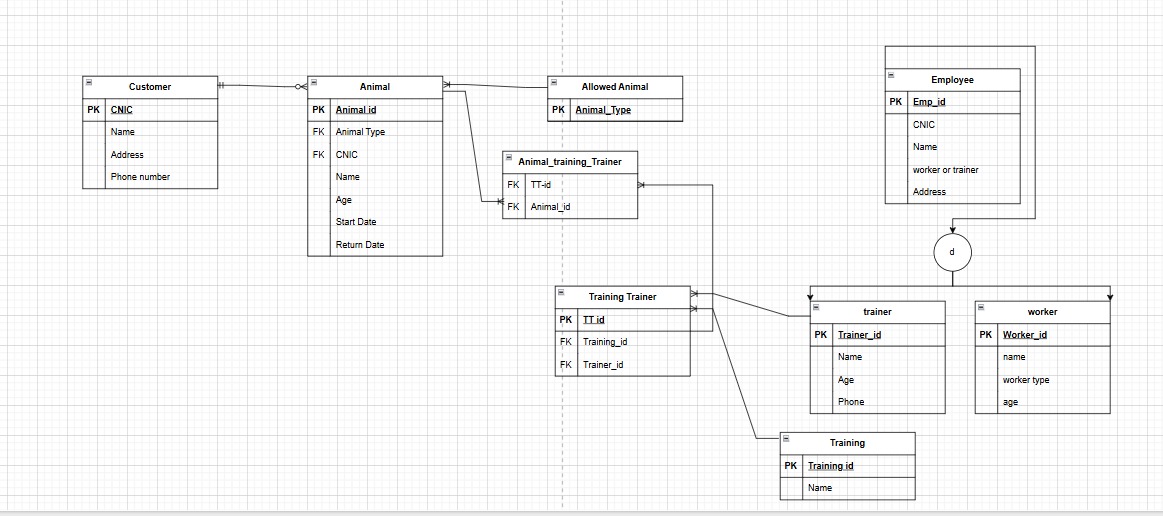
Mistakes like missing customers, missing animals , not know what trainer offers what type of training and many other things are common occurances.

**Requirements**

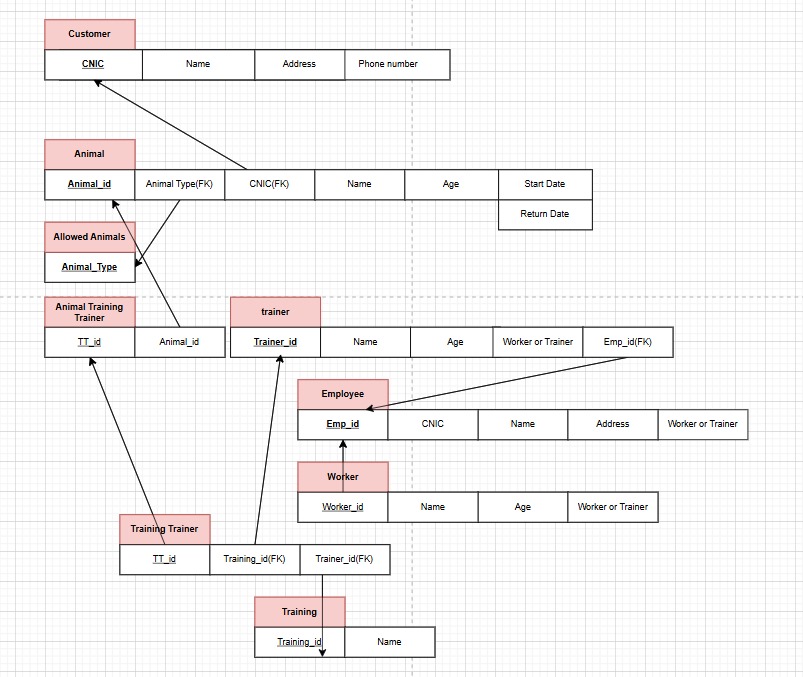
To create an application that works with sql server, c++ and has a UI

The application should be able to insert update delete search and other processes from the SQL server

**ER Diagram**



**Relational schema**



Database Design

**Tables**

We have several "tables" to store different types of information. Think of each table like a spreadsheet.

* **Customer Table**
  + **Purpose:** Stores information about our customers.
  + **Key Info:**
    - CNIC: Customer's ID (Main ID for this table)
    - Name: Customer's full name
    - Address: Where they live
    - PhoneNumber: How to call them
* **Allowed\_Animal Table**
  + **Purpose:** Lists the types of animals we train (e.g., Dog, Cat).
  + **Key Info:**
    - Animal\_Type: The type of animal (Main ID for this table)
* **Employee Table**
  + **Purpose:** Stores general information about everyone who works here.
  + **Key Info:**
    - Emp\_id: Employee's unique ID number (Main ID for this table)
    - CNIC: Employee's national ID (must be unique)
    - Name: Employee's full name
    - WorkerOrTrainer: Tells if they are a "Worker" or "Trainer"
    - Address: Where they live
* **Trainer Table**
  + **Purpose:** Specific details for employees who are Trainers.
  + **Key Info:**
    - Trainer\_id: Employee ID (links to Emp\_id in Employee table - Main ID here)
    - Age: Trainer's age
    - Phone: Trainer's phone number
* **Worker Table**
  + **Purpose:** Specific details for employees who are general Workers.
  + **Key Info:**
    - Worker\_id: Employee ID (links to Emp\_id in Employee table - Main ID here)
    - WorkerType: What kind of worker (e.g., "Kennel Staff")
    - Age: Worker's age
* **Training Table**
  + **Purpose:** Lists all the training programs we offer.
  + **Key Info:**
    - Training\_id: Unique ID for the training program (Main ID for this table)
    - Name: Name of the training (e.g., "Basic Obedience")
* **Animal Table**
  + **Purpose:** Information about each animal brought in for training.
  + **Key Info:**
    - Animal\_id: Unique ID for the animal (Main ID for this table)
    - AnimalType: Type of animal (links to Allowed\_Animal table)
    - CNIC: Owner's ID (links to Customer table)
    - Name: Animal's name
    - Age: Animal's age
    - StartDate: When training started
    - ReturnDate: When training ends
* **Training\_Trainer Table (Linking Table)**
  + **Purpose:** Shows which trainers can teach which training programs.
  + **Key Info:**
    - TT\_id: Unique ID for this specific trainer-training link (Main ID)
    - Training\_id: Links to the Training table
    - Trainer\_id: Links to the Trainer table
* **Animal\_Training\_Trainer Table (Linking Table)**
  + **Purpose:** Shows which animal is taking which training with which trainer.
  + **Key Info:**
    - Animal\_id: Links to the Animal table (Part of Main ID)
    - TT\_id: Links to the Training\_Trainer table (Part of Main ID)

**Relationships**

* A **Customer** can have many **Animals**. An **Animal** belongs to one **Customer**.
  + (Linked by Customer.CNIC and Animal.CNIC)
* An **Animal** is of an **Allowed\_Animal** type.
  + (Linked by Allowed\_Animal.Animal\_Type and Animal.AnimalType)
* An **Employee** can be a **Trainer** OR a **Worker**.
  + (Trainer.Trainer\_id is the same as Employee.Emp\_id. Worker.Worker\_id is the same as Employee.Emp\_id)
* A **Trainer** can teach many **Trainings**. A **Training** can be taught by many **Trainers**.
  + The Training\_Trainer table links them.
* An **Animal** can be in many **Training sessions** (a specific training taught by a specific trainer). A **Training session** can have many **Animals**.
  + The Animal\_Training\_Trainer table links them using the TT\_id from Training\_Trainer.

**Constraints**

* **Main IDs (Primary Keys):** Each item in a table (like a customer or an animal) gets a unique ID so we can find it easily.
* **Links (Foreign Keys):** These are IDs in one table that point to a Main ID in another table. This is how we connect information (e.g., an animal record has the owner's CNIC to link to the customer).
* **Keeping Data Clean:**
  + If a customer is deleted, their animals are also deleted.
  + If an employee who is a trainer is deleted, their trainer-specific info and their training assignments are also deleted.
  + This helps prevent having "orphan" records (like an animal with no owner).

**SQL Scripts**

**DDL**

CREATE TABLE Customer (

CNIC VARCHAR(20) PRIMARY KEY,

Name VARCHAR(255),

Address VARCHAR(255),

PhoneNumber VARCHAR(20)

);

The above is a code snipit of creating the customer table

**DML**

bool ui\_insertCustomer(const CustomerData& customer) {

    if (customer.cnic.empty() || customer.name.empty()) { statusMessage = "Error: CNIC and Name cannot be empty."; return false; }

    std::string sql = "INSERT INTO Customer (CNIC, Name, Address, PhoneNumber) VALUES ('" +

                      sanitizeSqlString(customer.cnic) + "', '" + sanitizeSqlString(customer.name) + "', '" +

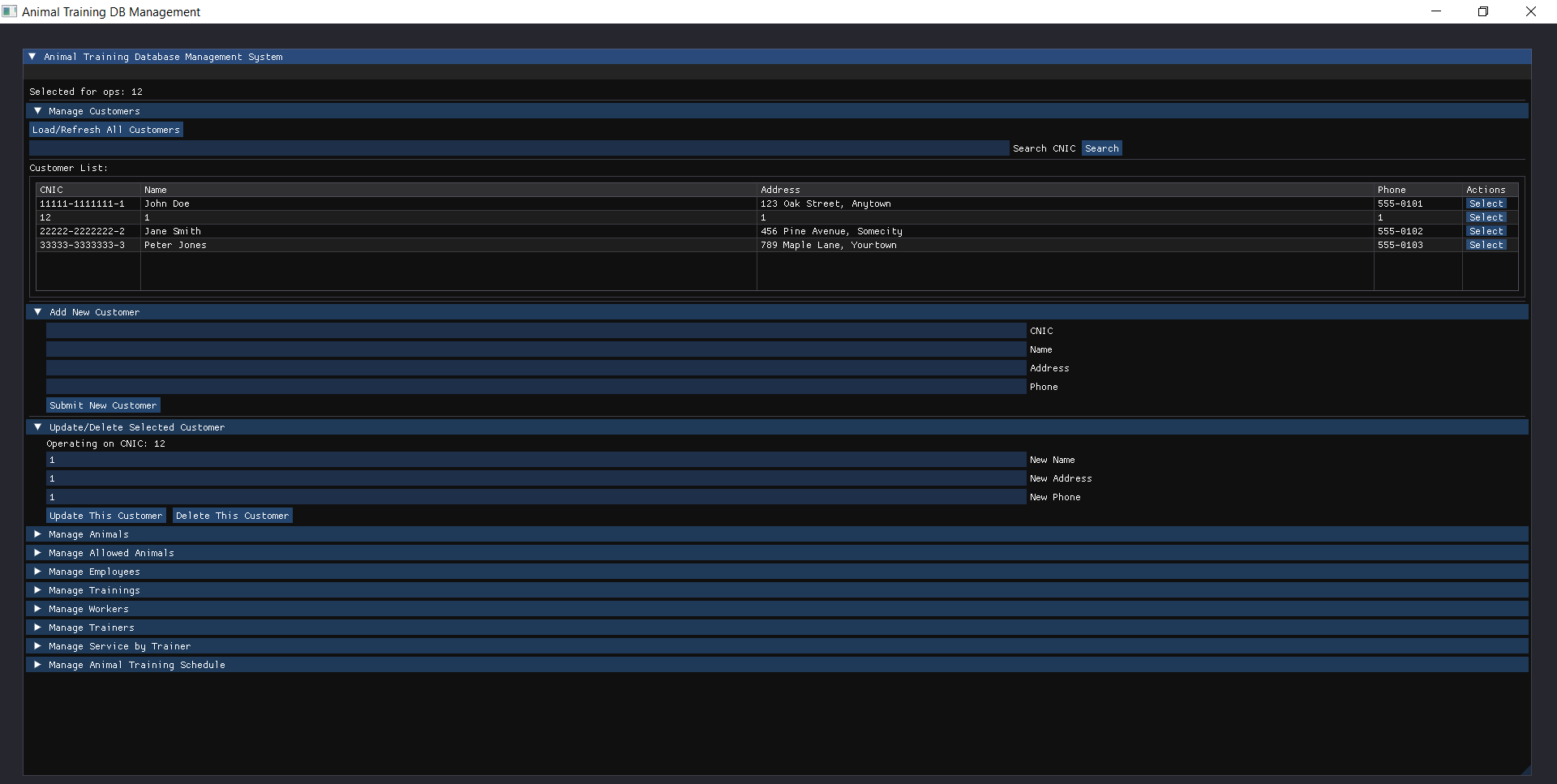
                      sanitizeSqlString(customer.address) + "', '" + sanitizeSqlString(customer.phoneNumber) + "');";

    return executeNonQuery(sql);

}

**The above is code snippet of insertion into customer table**

**Application interface**

****

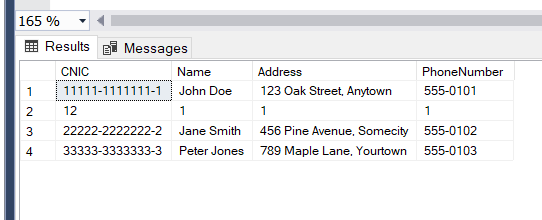
The UI has multiple multiple drop down menus with each entity(table) having its own respective drop down menu along with adding(insertion),updation and deletion.

The menu can be bnavigated using both mouse and keyboard.

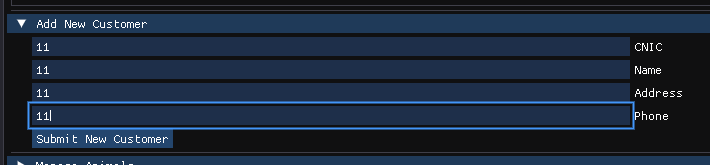
**Testing**

Below is default select\*from customer

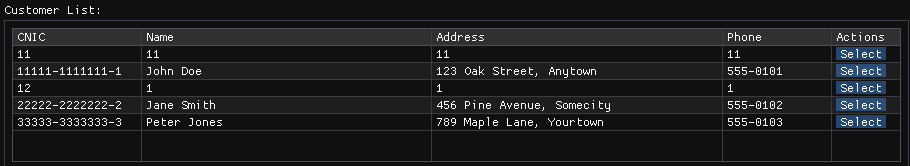


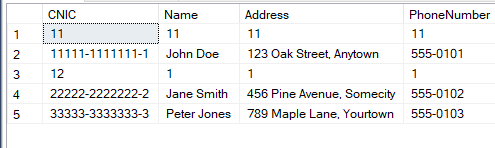


Then we insert 11,11,11,11 as a test input



we then get





**Challenges & Learnings**

-The main 2 challenges were to learn how to join sql and c++ and then to show the answer on a user interface

-The problem was solved with the help of Google Gemini 2.5 and videos from youtube

**Conclusion & Future Improvements**

In **conclusion** the application I created is targeted towards the administrator/manager not a general employee.

**Future development** could be to create a separate interface for customer, regular employees and use a password system to select type of person that logs in and the range of their actions .