SERVICE TEAM DATBASE GUIDE

Abstract

This document lists the tools and methods that are used to create & maintain the Service Team database.

Author: Austin Stockwell

Date: 4/10/2018

Austin Stockwell

4/10/2018

Service Team Database Guide

This document lists the tools and methods that are used to create & maintain the Service Team database. The database serves as a platform to store vital information that can be used for tracking of the following:

- Service Team Employees
- Customers
- Products
- Parts
- Repair History (Service Requests)

The first step in creating the Service Team database was designing an "Entity-Relationship Diagram". This was done digitally using MySQL. Once the ER diagram was created, a feature of MySQL called "Forward Engineering" was used to generate the basic SQL statements used to create the database automatically. The generated statements included:

- Schema Creation
- Database Creation
- Table Creation
- Foreign & Unique Key Checks

NOTE: Although it is useful to use MySQL's "forward engineering" feature, it is recommended that the user has a basic understanding of the SQL language to modify/correct any problems that arise within the automatically generated code!

Austin Stockwell

4/10/2018

Service Team Database Guide

```
-- MySQL Workbench Forward Engineering
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `service_team` DEFAULT CHARACTER SET utf8 ;
USE `service_team` ;
```

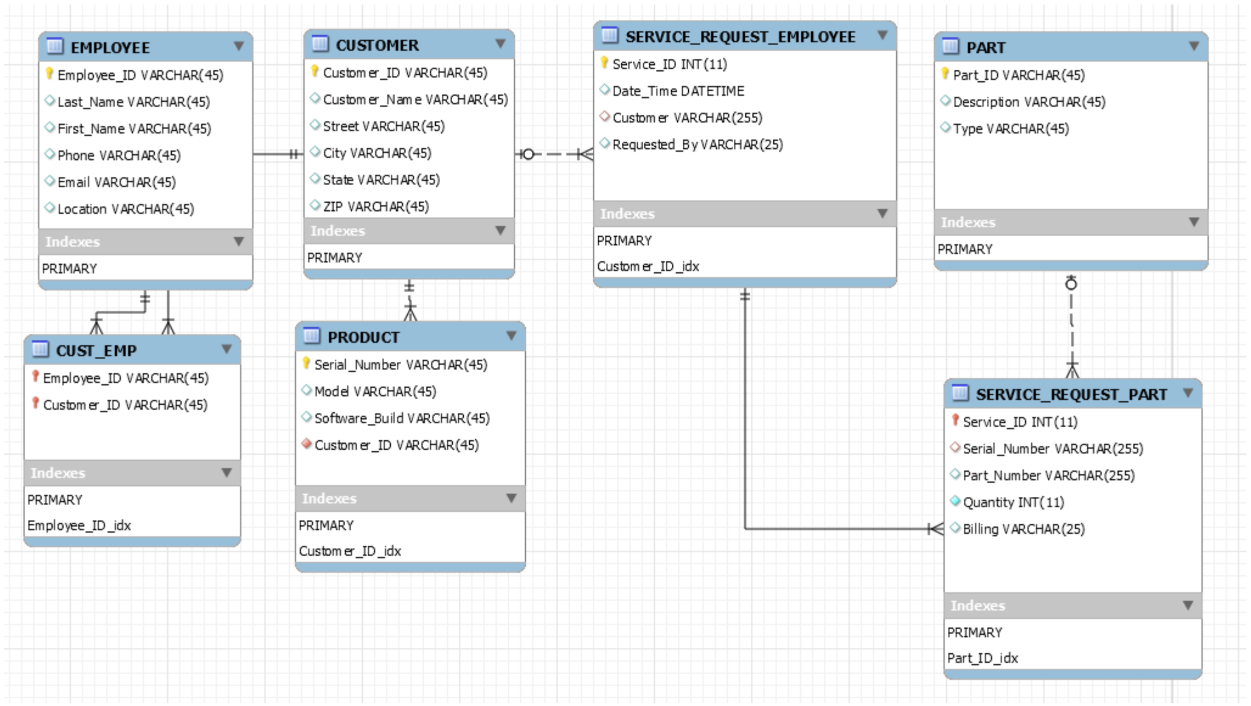


Fig 1 - ER Diagram created in MySQL.

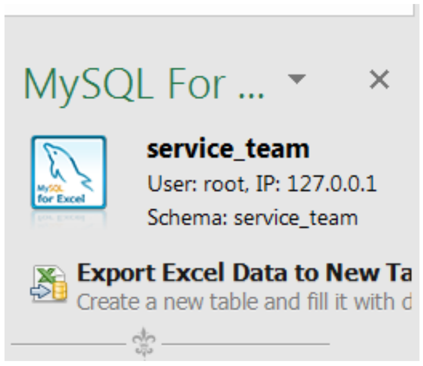
Once the tables and schema were created, it was necessary to enter data into the appropriate tables using INSERT commands. For example, the following statement inserted the employee named "Austin Stockwell" into the EMPLOYEE table, as well as important information about that employee.

```
-- -----------------------------------------------------

--  Inserts data into EMPLOYEE table –

-- -----------------------------------------------------

INSERT INTO EMPLOYEE

VALUES

('005', 'STOCKWELL', 'AUSTIN', '317-697-7664', 'austin.stockwell@mtdproducts.com',
'Indianapolis');
```

INSERT commands were also used to enter data into the PRODUCT, CUSTOMER, and PART tables. This was because these tables did not have many entries, and it was found that manual entry of data into these tables was adequate and was easily reviewable to maintain accuracy of data entry.

When dealing with larger datasets, it becomes important to find ways that avoid having to manually type SQL commands when entering data. For example, when creating the service team database, it was necessary to migrate over one thousand separate pieces of data from existing Excel worksheets into the MySQL database. Because this would require over one thousand separate INSERT commands if done manually, a feature of Excel was used that allowed data to be migrated into MySQL automatically, allowing the data to be entered into MySQL within seconds rather than hours (with 100% accuracy).

The 'MySQL for Excel' feature of Excel was used to automate the import of data from Excel into MySQL. The "MySQL for Excel" feature is located on the right-most side of the DATA menu within Excel. To create a new table in MySQL, the appropriate columns within the Excel worksheet were selected, and the "Export Excel Data to New Table" option was selected.


MySQL For ...
service_team
User: root, IP: 127.0.0.1
Schema: service_team
Export Excel Data to New Ta
Create a new table and fill it with d

Austin Stockwell

4/10/2018

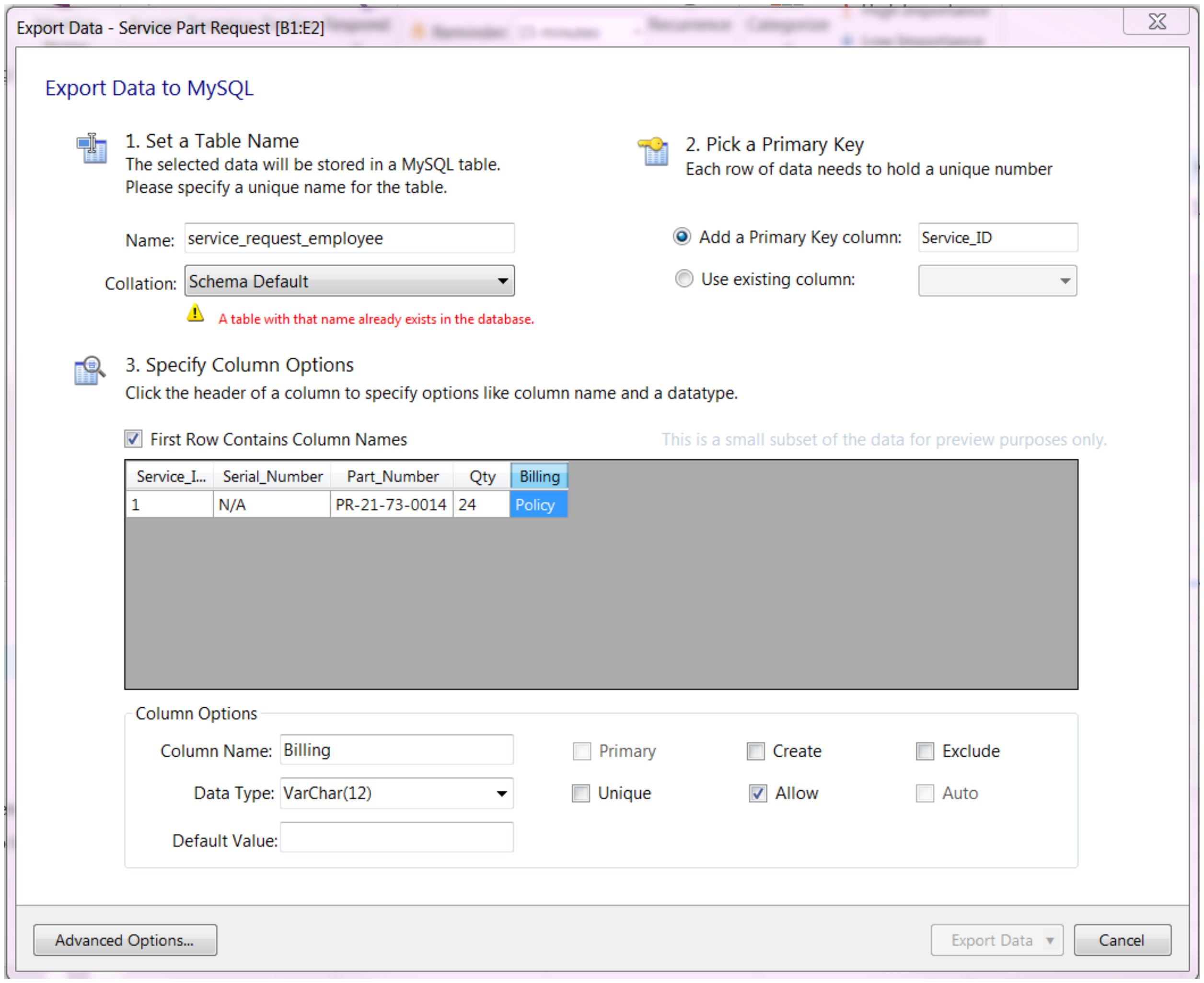
Service Team Database Guide

The following two tables were created:

a. SERVICE_REQUEST_EMPLOYEE (PARENT)

b. SERVICE_REQUEST_PART (CHILD)

It is important to note that to reduce update anomalies, the ***CASCADE ON UPDATE*** command was used. This allowed data in the SERVICE_REQUEST_PART (child table) to be deleted if the order with the corresponding Service_ID in the SERVICE_REQUEST_EMPLOYEE (parent table) was deleted.



Export Data - Service Part Request [B1:E2]

Export Data to MySQL

1. Set a Table Name
The selected data will be stored in a MySQL table.
Please specify a unique name for the table.

Name: service_request_employee

Collation: Schema Default

A table with that name already exists in the database.

2. Pick a Primary Key
Each row of data needs to hold a unique number

[✓ Add a Primary Key column:] Service_ID

Use existing column:

3. Specify Column Options
Click the header of a column to specify options like column name and a datatype.

[✓] First Row Contains Column Names

This is a small subset of the data for preview purposes only.

Service_I...	Serial_Number	Part_Number	Qty	Billing
1	N/A	PR-21-73-0014	24	Policy

Column Options

Column Name: Billing

Data Type: VarChar(12)

Default Value:

[ ] Primary [ ] Create [ ] Exclude

[ ] Unique [✓] Allow [ ] Auto

Austin Stockwell

4/10/2018

Service Team Database Guide

SQL Code:

```
/* AUTHOR: Austin Stockwell
   DATE: 4/10/2018
   SCOPE: This database is used for the SERVICE_TEAM
                        This database tracks employees, customers, products,
        parts, and service requests (repairs) of our products */


-- MySQL Workbench Forward Engineering
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `service_team` DEFAULT CHARACTER SET utf8 ;
USE `service_team` ;
-- -----------------------------------------------------
-- Table `mydb`.`EMPLOYEE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `service_team`.`EMPLOYEE`(
  `Employee_ID` VARCHAR(45),
  `Last_Name` VARCHAR(45) NULL,
  `First_Name` VARCHAR(45) NULL,
  `Phone` VARCHAR(45) NULL,
  `Email` VARCHAR(45) NULL,
  `Location` VARCHAR(45) NULL,
  PRIMARY KEY (`Employee_ID`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`CUSTOMER`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `service_team`.`CUSTOMER` (
  `Customer_ID` VARCHAR(45),
  `Customer_Name` VARCHAR(45) NULL,
  `Street` VARCHAR(45) NULL,
  `City` VARCHAR(45) NULL,
  `State` VARCHAR(45) NULL,
  `ZIP` VARCHAR(45) NULL,
  PRIMARY KEY (`Customer_ID`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`PRODUCT`
-- NOTE: CONSTRAINT was originally 'CUSTOMER_ID', which
--  gave an error.  To resolve error, CONSTRAINT
--  was changed to 'Cust_ID' (as shown below)
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `service_team`.`PRODUCT` (
  `Serial_Number` VARCHAR(45) NOT NULL,
  `Product` VARCHAR(45) NULL,
  `Model` VARCHAR(45) NULL,
  `Manufacture_Date` VARCHAR(45),
  `Customer_ID` VARCHAR(45) NOT NULL,
  `Hours` INT,
  `Software_Build` VARCHAR(45) NULL,
  PRIMARY KEY (`Serial_Number`),
  INDEX `Customer_ID_idx` (`Customer_ID` ASC),
  CONSTRAINT `Cust_ID`
    FOREIGN KEY (`Customer_ID`)
    REFERENCES `test_server`.`CUSTOMER` (`Customer_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`PART`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `service_team`.`PART` (
  `Part_ID` VARCHAR(45) NOT NULL,
  `Description` VARCHAR(75) NULL,
  `Type` VARCHAR(45) NULL,
  PRIMARY KEY (`Part_ID`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`CUST_EMP`
-- NOTE: Constraint was originally "CUSTOMER_ID"
--      was changed to "Cust_Ident" to avoid error!
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `service_team`.`CUSTUMER_EMPLOYEE` (
  `Employee_ID` VARCHAR(45) NOT NULL,
  `Customer_ID` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Customer_ID`, `Employee_ID`),
  INDEX `Employee_ID_idx` (`Employee_ID` ASC),
  CONSTRAINT `Emp_ID`
    FOREIGN KEY (`Employee_ID`)
    REFERENCES `test_server`.`EMPLOYEE` (`Employee_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Cust_ident`
    FOREIGN KEY (`Customer_ID`)
    REFERENCES `test_server`.`CUSTOMER` (`Customer_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`PRODUCT_SERVICE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `service_team`.`PRODUCT_SERVICE` (
  `Serial_Number` VARCHAR(45) NOT NULL,
  `Service_ID` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Serial_Number`, `Service_ID`),
  INDEX `Service_ID_idx` (`Service_ID` ASC),
  CONSTRAINT `Serial_Num`
    FOREIGN KEY (`Serial_Number`)
    REFERENCES `test_server`.`PRODUCT` (`Serial_Number`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Serv_ID`
    FOREIGN KEY (`Service_ID`)
    REFERENCES `test_server`.`SERVICE` (`Service_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`SERVICE_REQUEST_EMPLOYEE`
        /*AUTO GENERATED FROM EXCEL!!! */
-- -----------------------------------------------------
CREATE TABLE `service_request_employee` (
  `Service_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Date_Time` datetime DEFAULT NULL,
  `Customer` varchar(255) DEFAULT NULL,
  `Requested_By` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`Service_ID`)
) ENGINE=InnoDB AUTO_INCREMENT=837 DEFAULT CHARSET=utf8
-- -----------------------------------------------------
-- Table `mydb`.`SERVICE_REQUEST_PART`
/*AUTO GENERATED FROM EXCEL!!! */
-- -----------------------------------------------------
CREATE TABLE `service_request_part` (
  `Service_ID` int(11) NOT NULL AUTO_INCREMENT,
  `Serial_Number` varchar(255) DEFAULT NULL,
  `Part_Number` varchar(255) DEFAULT NULL,
  `Quantity` int(11) NOT NULL,
  `Billing` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`Service_ID`),
  KEY `Quantity_index` (`Quantity`),
  CONSTRAINT `Service_ID` FOREIGN KEY (`Service_ID`) REFERENCES `service_request_employee`
(`Service_ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=837 DEFAULT CHARSET=utf8
```

Austin Stockwell

4/10/2018

Service Team Database Guide

```
-- -----------------------------------------------------------

-- QUICK REFERENCE

-- -----------------------------------------------------------

SELECT * FROM EMPLOYEE;

SELECT * FROM CUSTOMER;

SELECT * FROM PART;

SELECT * FROM SERVICE;

SELECT * FROM SERVICE_LINE;

SELECT * FROM PRODUCT;

-- -------------------------------------------------------

-- INSERT EMPLOYEE DATA

-- -------------------------------------------------------

SELECT * FROM EMPLOYEE;

INSERT INTO EMPLOYEE

VALUES

('005', 'STOCKWELL', 'AUSTIN', '317-697-7664', 'austin.stockwell@mtdproducts.com', 'Indianapolis');
```

Austin Stockwell

4/10/2018

Service Team Database Guide

C# Database Entry Form

Because the average Service Team employee is not familiar with database operations, it was important to create a method of entering data into a MySQL database without having to familiarize the Service Team employee with MySQL or the SQL language. To do this, a C# program was created that achieves this goal.

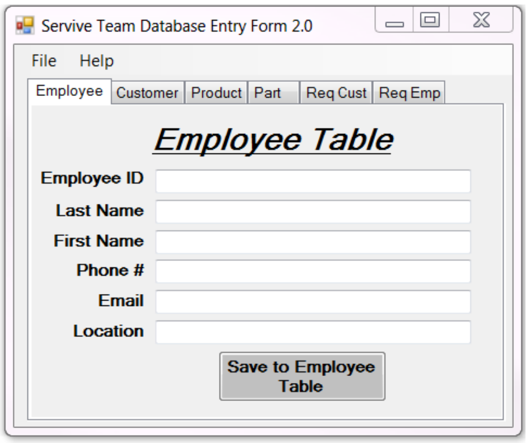
The program that was created utilizes the GUI controls that come with the C# language. The MySQL for Visual Studio library was also used to allow the application to interact with the database in real time (inserting data into the database).

Austin Stockwell

4/10/2018

Service Team Database Guide

C# Database Entry GUI


Servive Team Database Entry Form 2.0

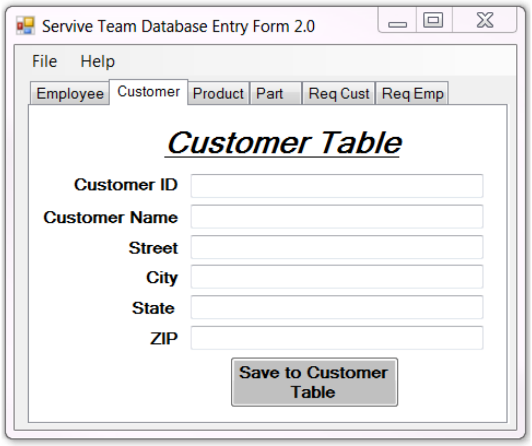
File Help

Employee | Customer | Product | Part | Req Cust | Req Emp

Employee Table

Employee ID
Last Name
First Name
Phone #
Email
Location

Save to Employee Table


Servive Team Database Entry Form 2.0

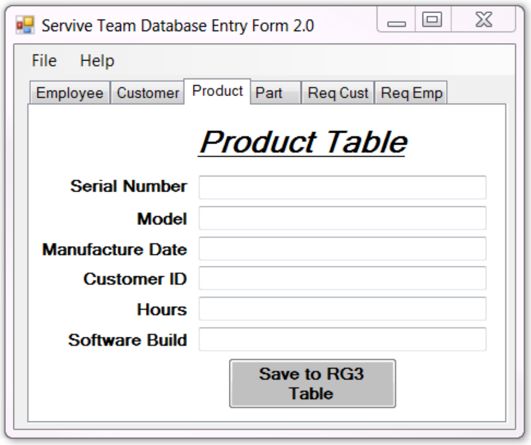
File Help

Employee | Customer | Product | Part | Req Cust | Req Emp

Customer Table

Customer ID
Customer Name
Street
City
State
ZIP

Save to Customer Table


Servive Team Database Entry Form 2.0

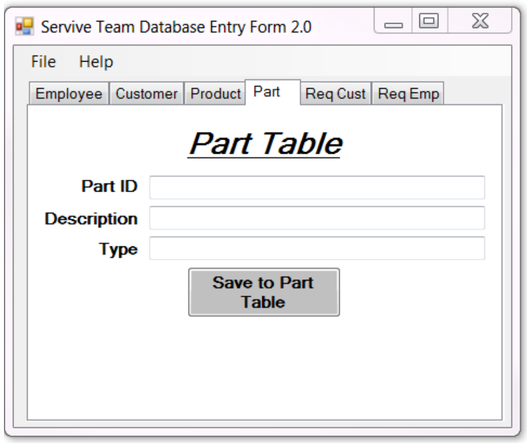
File Help

Employee | Customer | Product | Part | Req Cust | Req Emp

Product Table

Serial Number
Model
Manufacture Date
Customer ID
Hours
Software Build

Save to RG3 Table


Servive Team Database Entry Form 2.0

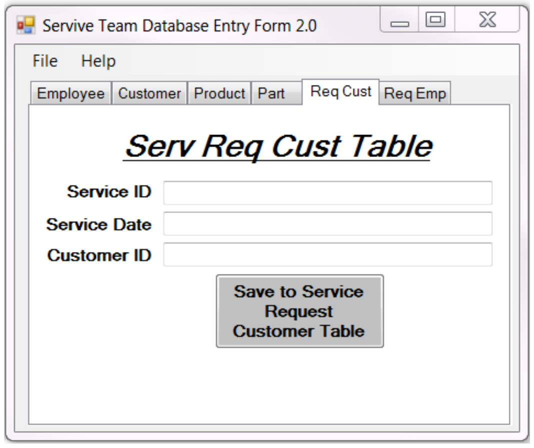
File Help

Employee | Customer | Product | Part | Req Cust | Req Emp

Part Table

Part ID
Description
Type

Save to Part Table


Servive Team Database Entry Form 2.0

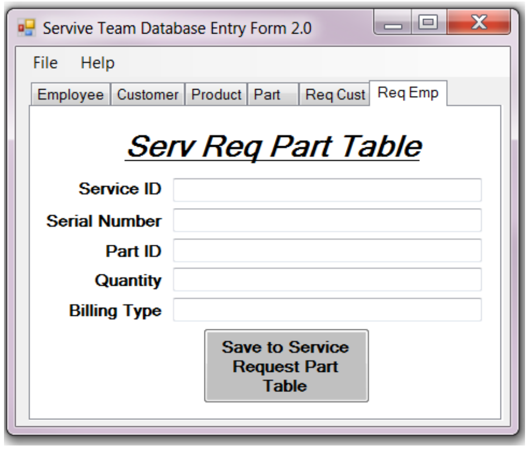
File Help

Employee | Customer | Product | Part | Req Cust | Req Emp

Serv Req Cust Table

Service ID
Service Date
Customer ID

Save to Service Request Customer Table


Servive Team Database Entry Form 2.0

File Help

Employee | Customer | Product | Part | Req Cust | Req Emp

Serv Req Part Table

Service ID
Serial Number
Part ID
Quantity
Billing Type

Save to Service Request Part Table

C# Application Code

```
// AUTHOR: AUSTIN STOCKWELL
// DATE: 4/10/2018
// DESCRIPTION: THIS PROGRAM IS USED TO ENTER DATA INTO THE SERVICE TEAM DATABASE.


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Service_Team_Database_Entry_Form_2._0
{
    public partial class Form1 : Form
    {
        MySqlConnection con = new MySqlConnection("server=localhost;user 
id=root;password=Supra777;database=service_team");
        public Form1()
        {
            InitializeComponent();
        }

        private void employeeTableSaveButton_Click(object sender, EventArgs e)
        {
            try
            {
                MySqlCommand cmd = new MySqlCommand("INSERT INTO 
EMPLOYEE(Employee_ID,Last_Name,First_Name,Phone,Email,Location)Values('" + 
employeeIDBox.Text + "','" + lastNameBox.Text + "','" + firstNameBox.Text + "','" 
+ phoneBox.Text + "','" + emailBox.Text + "','" + locationBox.Text + "')", con);
                con.Open();
                cmd.ExecuteNonQuery();
                MessageBox.Show("Save Success! You have entered:" + "\n" + "\n" + 
"Employee ID: " + employeeIDBox.Text + "\n" +
                                "Last Name: " + lastNameBox.Text + "\n" + "First 
Name: " + firstNameBox.Text + "\n" +
                                "Phone: " + phoneBox.Text + "\n" + "Email: " + 
emailBox.Text + "\n" + "Location: " + locationBox.Text);
                con.Close();
            }
            catch
            {
                MessageBox.Show("Error - Please try again!");
            }
        }

        private void customerTableSaveButton_Click(object sender, EventArgs e)
        {
            try
            {
                MySqlCommand cmd = new MySqlCommand("INSERT INTO 
CUSTOMER(Customer_ID,Customer_Name,Street,City,State,ZIP)Values('" + 
customerIDBox.Text + "','" + customerNameBox.Text + "','" + streetBox.Text + "','" 
+ cityBox.Text + "','" + stateBox.Text + "','" + zipBox.Text + "')", con);
                con.Open();
                cmd.ExecuteNonQuery();
                MessageBox.Show("Save Success! You have entered:" + "\n" + "\n" + 
"Customer ID: " + customerIDBox.Text + "\n" + 
                            "Customer Name: " + customerNameBox.Text + "\n" + 
"Street: " + streetBox.Text + "\n" + 
                            "City: " + cityBox.Text + "\n" + "State: " + 
stateBox.Text + "\n" + "ZIP: " + zipBox.Text);
                con.Close();
            }
            catch
            {
                MessageBox.Show("Error -- Please try again!");
            }
        }

        private void rg3TableSaveButton_Click(object sender, EventArgs e)
        {
            try
            {
                MySqlCommand cmd = new MySqlCommand("INSERT INTO 
RG3(Serial_Number,Model,Manufacture_Date,Customer_ID,Hours,Software_Build)Values('
" + serialNumberBox.Text + "','" + modelBox.Text + "','" + manufactureDateBox.Text 
+ "' , '" + rg3_CustomerIDBox.Text + "', '" + hoursBox.Text + "', '" + 
softwareBuildBox.Text + "')", con);
                con.Open();
                cmd.ExecuteNonQuery();
                MessageBox.Show("Save Success! You have entered:" + "\n" + "\n" + 
"Serial Number: " + serialNumberBox.Text + "\n" + 
                           "Model: " + modelBox.Text + "\n" + "Manufacture Date: " 
+ manufactureDateBox.Text + "\n" + 
                           "Customer ID: " + rg3_CustomerIDBox.Text + "\n" + 
"Hours: " + hoursBox.Text + "\n" + 
                           "Software Build: " + softwareBuildBox.Text);
                con.Close();
            }
            catch
            {
                MessageBox.Show("Error -- You must enter a valid Customer ID!");
            }
        }

        private void partTableSaveButton_Click(object sender, EventArgs e)
        {
            try
            {
                MySqlCommand cmd = new MySqlCommand("INSERT INTO 
PART(Part_ID,Description,Type)Values('" + partIDBox.Text + "','" + 
descriptionBox.Text + "','" + typeBox.Text + "')", con);
```

```
                con.Open();
                cmd.ExecuteNonQuery();
                MessageBox.Show("Save Success! You have entered:" + "\n" + "\n" +
"Part ID: " + partIDBox.Text + "\n" +
                           "Description: " + descriptionBox.Text + "\n" + "Type: "
+ typeBox.Text);
                con.Close();
            }
            catch
            {
                MessageBox.Show("Error -- Please try again!");
            }
        }

        private void serviceTableSaveButton_Click(object sender, EventArgs e)
        {
            try
            {
                MySqlCommand cmd = new MySqlCommand("INSERT INTO
SERVICE(Service_ID,Service_Date,Customer_ID)Values('" + serviceIDBox.Text + "','"
+ serviceDateBox.Text + "','" + service_CustomerIDBox.Text + "')", con);
                con.Open();
                cmd.ExecuteNonQuery();
                MessageBox.Show("Save Success! You have entered:" + "\n" + "\n" +
"Service ID: " + serviceIDBox.Text + "\n" +
                            "Service Date: " + serviceDateBox.Text + "\n" +
"Customer ID: " + service_CustomerIDBox.Text);
                con.Close();
            }
            catch
            {
                MessageBox.Show("Error -- Customer ID must match an existing
Customer ID");
            }
        }

        private void servicelineTableSaveButton_Click(object sender, EventArgs e)
        {
            try
            {
                {
                    MySqlCommand cmd = new MySqlCommand("INSERT INTO
SERVICE_LINE(Service_ID,Serial_Number,Part_ID,Quantity,Billing)Values('" +
serviceline_ServiceIDBox.Text + "','" + serviceline_SerialNumberBox.Text + "','" +
serviceline_PartIDBox.Text + "','" + quantityBox.Text + "','" + billingBox.Text +
"')", con);
                    con.Open();
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Save Success! You have entered:" + "\n" +
"\n" + "Service ID: " + serviceline_ServiceIDBox.Text + "\n" +
                            "Serial Number: " + serviceline_SerialNumberBox.Text +
"\n" + "Part ID: " + serviceline_PartIDBox.Text + "\n" +
                            "Quantity: " + quantityBox.Text + "\n" + "Billing: " +
billingBox.Text);
                    con.Close();
                }
            }
            catch
            {
                MessageBox.Show("Error -- Service ID, Serial Number, and Part ID 
must all match allready existing values in the system! Re-check your entry.");
            }
        }

        private void rg3Tab_Click(object sender, EventArgs e)
        {

        }
    }
}
```