RESIDENTIAL LIFE

CS: 470 - Introduction to Database Management System

Group Members:

Aishvwarya Iyer

Gayathree Iyer

Introduction

This project is about the Residential Life information of University of Missouri-Kansas City.

Every university has a residential facility. Students live on campus to get better grades, build academic connections, increase their opportunities to meet new friends and utilize campus resources more efficiently.

Residential Life includes various database information like the 'Resident', 'Package', 'Equipment', 'Guests', 'Cafe', 'Stock'.

Tables

Following are the requirements for this project:

- 'Residents' which includes all the details about the residents. These details include Student ID, Resident Name, Phone number, email address, gender, Hall, Room number and balance.
- 'Guests' which includes details like the ResidentID, Guest name, Date In and Date Out.
- 'Equipment' database includes the resident name, equipment name (pool supplies, cart, kitchen supplies, board games, etc.), the date on which the resident checks out the equipment and the date on which the resident checks in the equipment.
- 'Package' database includes the resident's name, is perishable, the date on which the package arrives at the location, and the date on which the resident picks up the package.
- 'Café' in the dorms is for the residents. The database includes Item ID, Item name, Item price.
- 'Stock' table includes the item name, item count and item price.

Architecture

Three tier architecture is best suited for this project. It is a client–server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. A three tier architecture is a programming model that enables the distribution of the functionality of the application across three independent systems, namely:

Presentation Tier: Presentation tier is the application's top most level. It is responsible for providing the application's user interface. It uses the graphical User Interface for smart client interaction and web based technologies for browser based interaction.

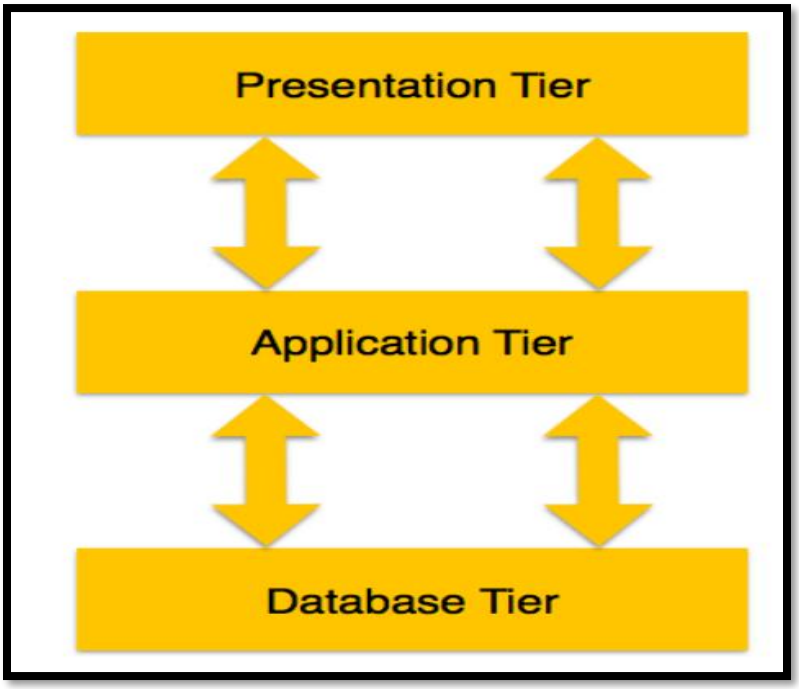
Application Tier: The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing. This layer is responsible for solving mission critical business problems.

Data Tier: Houses database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic. Information is stored and retrieved in this layer.

Platform

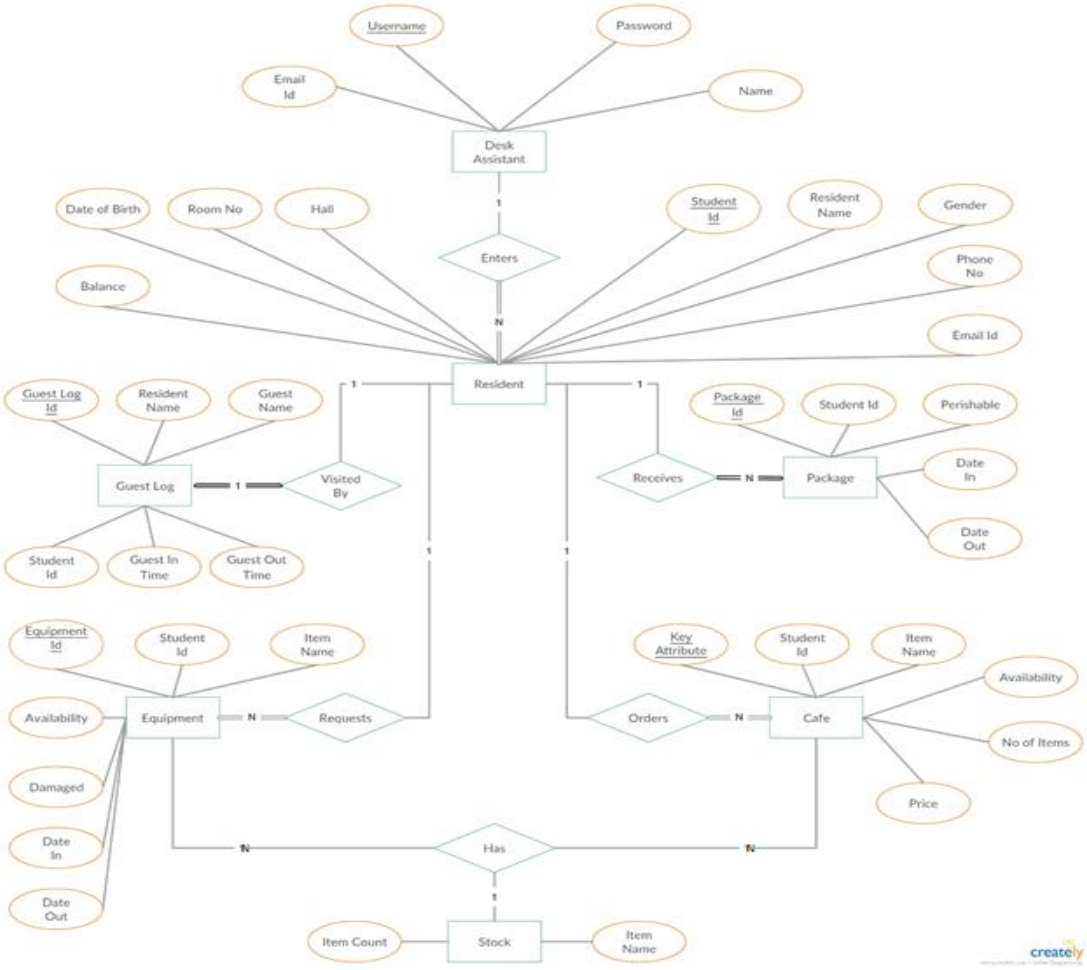
- Database: MySQL
- Language used in Business Logic Layer: C#
- Web page design: Windows Form Application

Database Architecture

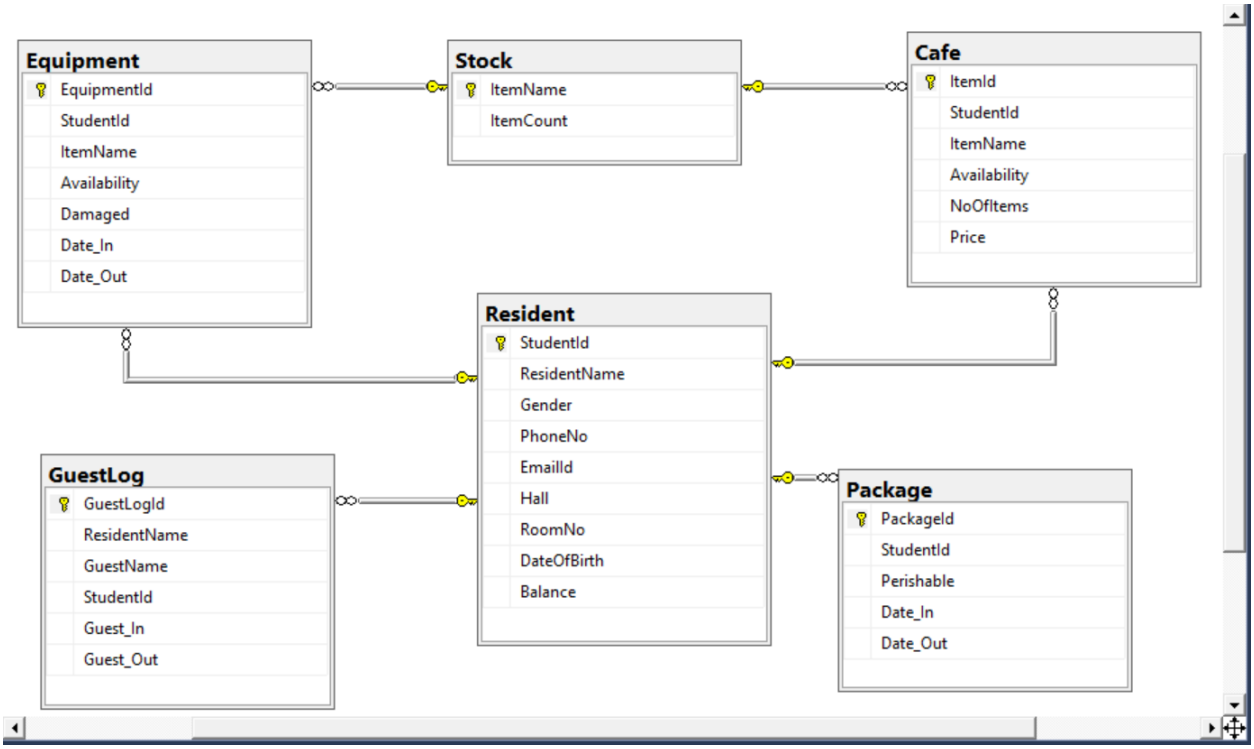


1. Presentation Tier – It has the Web Interface and end users work on this tier where they don't know anything about existence of database beyond this layer. (AngularJS, HTML5, CSS3)
2. Application Tier – It will contain the SQL Queries, and code to retrieve desired data from database and present it into Presentation Tier. (JAVA)
3. Database Tier – It consist of database which contains tables, triggers, and stored procedures related to residential life management system. (SQL SERVER 2017)

E-R Diagram


Username
Password
Email Id
Name
Desk Assistant
1
Enters
N
Date of Birth
Room No
Hall
Student Id
Resident Name
Gender
Phone No
Balance
Email Id
Resident
1
1
Guest Log Id
Resident Name
Guest Name
Guest Log
1
Visited By
Student Id
Guest In Time
Guest Out Time
Package Id
Student Id
Perishable
Receives
N
Package
Date In
Date Out
1
1
Equipment Id
Student Id
Item Name
Availability
Equipment
N
Requests
Damaged
Date In
Date Out
Key Attribute
Student Id
Item Name
Availability
Orders
N
Cafe
No of Items
Price
N
Has
N
1
Item Count
Stock
Item Name
creately

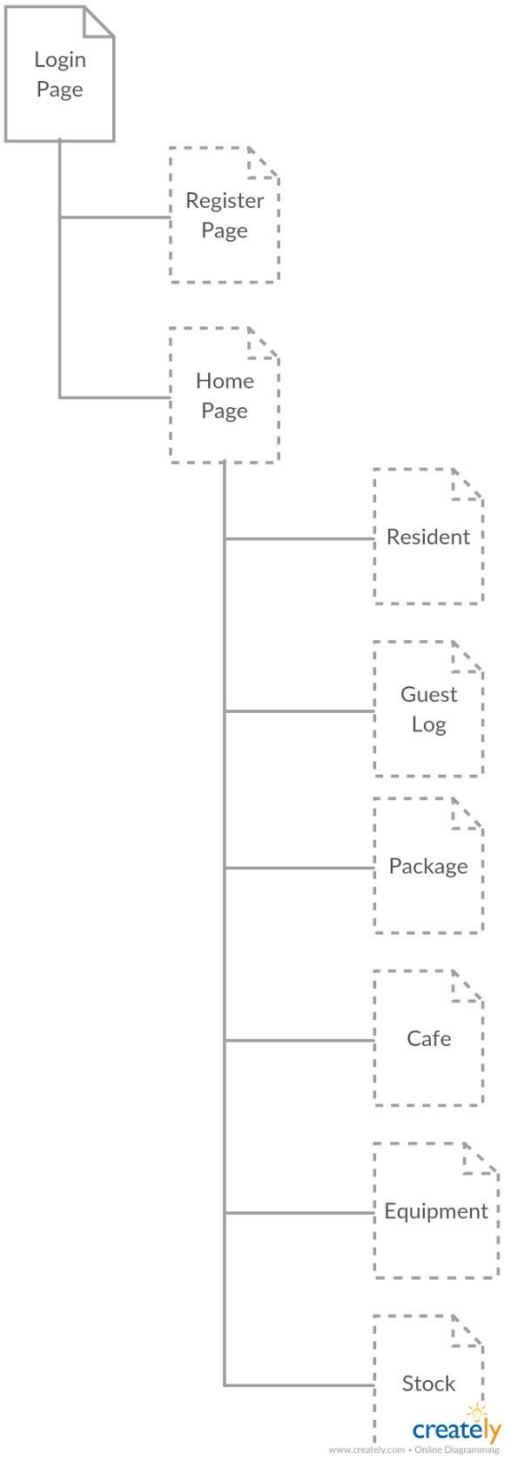
Data Schema



Anticipated Constraints

	Desk Assistant	Resident	Guest Log	Package	Equipment	Cafe	Stock
NOT NULL	Email Id Password Name	ResidentName Gender Email Id Hall RoomNo DateOfBirth Balance	ResidentName GuestName Guest_In Guest_Out	Perishable Date_In Date_Out	StudentId ItemName Damaged Date_In Date_Out	StudentId ItemName NoOfItems Price	
UNIQUE	Email Id	Email Id					
PRIMARY KEY	Username	StudentId	GuestLogId	PackageId	EquipmentId	ItemId	ItemName
FOREIGN KEY			StudentId	StudentId	StudentId ItemName	StudentId ItemName	
CHECK		Gender DateOfBirth			Availability	Availability NoOfItems Price	ItemCount
DEFAULT			Guest_In	Date_In	Date_In	Date_In	ItemCount

Site Maps


Login Page
Register Page
Home Page
Resident
Guest Log
Package
Cafe
Equipment
Stock
creately
www.creately.com • Online Diagramming

Core Queries

```sql
-- Create Database
Create database ResidentialLife
-- Create Table DeskAssistant
USE ResidentialLife
CREATE TABLE dbo.DeskAssistant (
    Username varchar(255) PRIMARY KEY,
    EmailId varchar(255) NOT NULL UNIQUE,
    Password varchar(255) NOT NULL,
    Name varchar(255) NOT NULL
);

DROP TABLE dbo.DeskAssistant
-- Create Table Resident
USE ResidentialLife
CREATE TABLE dbo.Resident
(
    StudentId INT,
    ResidentName VARCHAR(255) NOT NULL,
    Gender VARCHAR(255) NOT NULL,
    CHECK(Gender IN ('M','F','O')),
    PhoneNo VARCHAR(15),
    EmailId VARCHAR(255) NOT NULL UNIQUE,
    Hall varchar(255) NOT NULL,
    RoomNo INT NOT NULL,
    DateOfBirth DATE NOT NULL,
    CHECK(DATEDIFF(YEAR,DateOfBirth,GETDATE())>16),
    Balance DECIMAL NOT NULL,
    CHECK(Balance>0),
    PRIMARY KEY(StudentId)
);

DROP TABLE dbo.Resident
-- Create Table GuestLog
USE ResidentialLife
CREATE TABLE dbo.GuestLog(
    GuestLogId INT IDENTITY(1,1) PRIMARY KEY,
    ResidentName VARCHAR(255) NOT NULL,
    GuestName VARCHAR(255) NOT NULL,
    StudentId INT NOT NULL,
    Guest_In DATETIME NOT NULL DEFAULT GETDATE(),
    Guest_Out DATETIME NOT NULL,
    FOREIGN KEY (StudentId) REFERENCES dbo.Resident(StudentId)
);
```

```sql
DROP TABLE dbo.GuestLog
-- Create Table Package
USE ResidentialLife
CREATE TABLE dbo.Package(
    PackageId INT IDENTITY(1,1) PRIMARY KEY,
    StudentId INT NOT NULL,
    Perishable BIT NOT NULL,
    Date_In DATE NOT NULL DEFAULT GETDATE(),
    Date_Out DATE NOT NULL,
    FOREIGN KEY (StudentId) REFERENCES dbo.Resident(StudentId)
);

DROP TABLE dbo.Package
-- Create Table Equipment
USE ResidentialLife
CREATE TABLE dbo.Equipment(
    EquipmentId INT IDENTITY(1,1) PRIMARY KEY,
    StudentId INT NOT NULL,
    ItemName VARCHAR(255) NOT NULL,
    Availability INT,
    CHECK(Availability>0),
    Damaged BIT NOT NULL,
    Date_In DATETIME NOT NULL DEFAULT GETDATE(),
    Date_Out DATETIME NOT NULL,
    FOREIGN KEY (StudentId) REFERENCES dbo.Resident(StudentId),
    FOREIGN KEY (ItemName) REFERENCES dbo.Stock(ItemName)
);

DROP TABLE dbo.Equipment
-- Create Table Cafe
USE ResidentialLife
CREATE TABLE dbo.Cafe(
    ItemId INT IDENTITY(1,1) PRIMARY KEY,
    StudentId INT NOT NULL,
    ItemName VARCHAR(255) NOT NULL,
    Availability INT,
    CHECK(Availability>0),
    NoOfItems INT NOT NULL,
    CHECK(NoOfItems>0),
    Price DECIMAL NOT NULL,
    CHECK(Price>0),
    FOREIGN KEY (StudentId) REFERENCES dbo.Resident(StudentId),
    FOREIGN KEY (ItemName) REFERENCES dbo.Stock(ItemName)
);
```

```sql
DROP TABLE dbo.Cafe
-- Create Table Stock
USE ResidentialLife
CREATE TABLE dbo.Stock(
   ItemName VARCHAR(255) PRIMARY KEY,
   ItemCount INT DEFAULT 0
);

DROP TABLE dbo.Stock
```

INSERT RECORD:

```csharp
//Insert Data
private void btn_Insert_Click(object sender, EventArgs e)
{
    if (txt_Name.Text != "" &&
        txt_balance.Text != "" &&
        txt_gender.Text != "" &&
        txt_hall.Text != "" &&
        txt_room.Text != "" &&
        txt_balance.Text != "" &&
        txt_email.Text != "" &&
        txt_dob.Text != "")
    {
        cmd = new SqlCommand("insert into dbo.Resident(StudentID,ResidentName,Gender,PhoneNo,EmailId,Hall,RoomNo,DateOfBirth,Ba
        "values(@id,@name,@gender,@phone,@email,@hall,@room,@dob,@balance)", con);
    con.Open();
    cmd.Parameters.AddWithValue("@id", txt_id.Text);
    cmd.Parameters.AddWithValue("@name", txt_Name.Text);
    cmd.Parameters.AddWithValue("@phone", txt_Phoneno.Text);
    cmd.Parameters.AddWithValue("@email", txt_email.Text);
    cmd.Parameters.AddWithValue("@hall", txt_hall.Text);
    cmd.Parameters.AddWithValue("@room", txt_room.Text);
    cmd.Parameters.AddWithValue("@dob", Convert.ToDateTime(txt_dob.Text));
    cmd.Parameters.AddWithValue("@gender", txt_gender.Text);
    cmd.Parameters.AddWithValue("@balance", txt_balance.Text);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("Record Inserted Successfully");
    DisplayData();
    ClearData();
```

UPDATE RECORD:

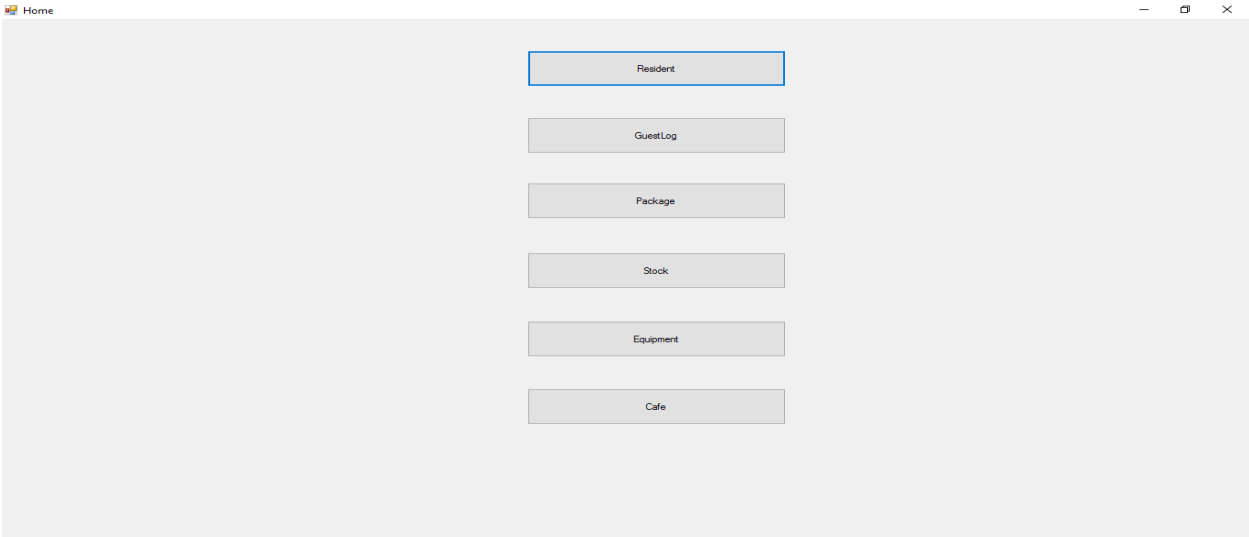
```
//Update Record
private void btn_Update_Click(object sender, EventArgs e)
{
    if (txt_Name.Text != "" &&
       txt_balance.Text != "" &&
       txt_gender.Text != "" &&
       txt_hall.Text != "" &&
       txt_room.Text != "" &&
       txt_balance.Text != "" &&
       txt_email.Text != "" &&
       txt_dob.Text != "")
    {
        cmd = new SqlCommand("update dbo.Resident set StudentId=@id,ResidentName=@name,Gender=@gender,PhoneNo=@phone," +
        "EmailId=@email,Hall=@hall,RoomNo=@room,DateOfBirth=@dob,Balance=@balance where StudentID=@id", con);
    con.Open();
    cmd.Parameters.AddWithValue("@id", txt_id.Text);
    cmd.Parameters.AddWithValue("@name", txt_Name.Text);
    cmd.Parameters.AddWithValue("@phone", txt_Phoneno.Text);
    cmd.Parameters.AddWithValue("@email", txt_email.Text);
    cmd.Parameters.AddWithValue("@hall", txt_hall.Text);
    cmd.Parameters.AddWithValue("@room", txt_room.Text);
    cmd.Parameters.AddWithValue("@dob", Convert.ToDateTime(txt_dob.Text));
    cmd.Parameters.AddWithValue("@gender", txt_gender.Text);
    cmd.Parameters.AddWithValue("@balance", txt_balance.Text);
    cmd.ExecuteNonQuery();
    MessageBox.Show("Record Updated Successfully");
    con.Close();
    DisplayData();
    ClearData();
```

DELETE RECORD:

```
//Delete Record
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (ID != 0)
    {
        cmd = new SqlCommand("delete dbo.Resident where StudentId=@id", con);
    con.Open();
    cmd.Parameters.AddWithValue("@id", txt_id.Text);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("Record Deleted Successfully!");
    DisplayData();
    ClearData();
}
    else
    {
        MessageBox.Show("Please Select Record to Delete");
    }
```

Working Model

HOMEPAGE

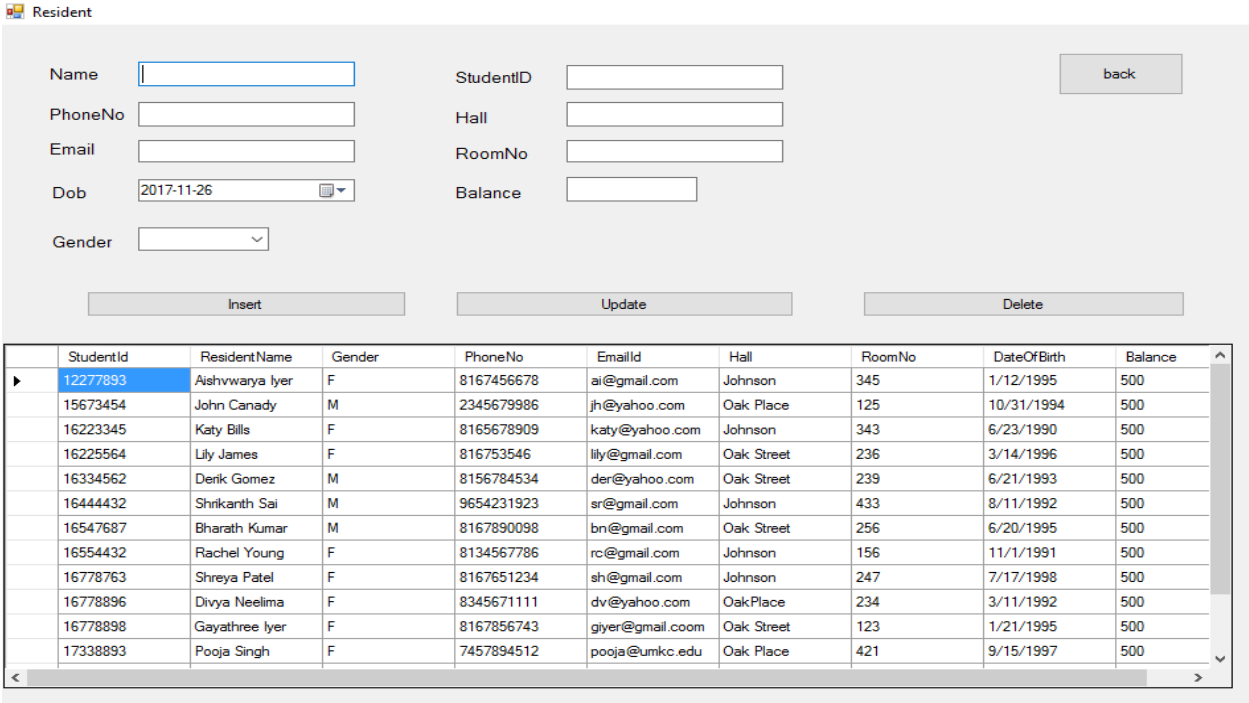


The Home Page includes buttons to Resident, Guest Log, Package, Stock, Equipment, Café.

RESIDENT

The 'Resident' page includes details about the residents residing in the university dorms.

It includes StudentID, Resident name, gender, phone number, email id, Hall, Room No., Date of birth and balance.



StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
12277893	Aishvwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	500
15673454	John Canady	M	2345679986	jh@yahoo.com	Oak Place	125	10/31/1994	500
16223345	Katy Bills	F	8165678909	katy@yahoo.com	Johnson	343	6/23/1990	500
16225564	Lily James	F	816753546	lily@gmail.com	Oak Street	236	3/14/1996	500
16334562	Derik Gomez	M	8156784534	der@yahoo.com	Oak Street	239	6/21/1993	500
16444432	Shrikanth Sai	M	9654231923	sr@gmail.com	Johnson	433	8/11/1992	500
16547687	Bharath Kumar	M	8167890098	bn@gmail.com	Oak Street	256	6/20/1995	500
16554432	Rachel Young	F	8134567786	rc@gmail.com	Johnson	156	11/1/1991	500
16778763	Shreya Patel	F	8167651234	sh@gmail.com	Johnson	247	7/17/1998	500
16778896	Divya Neelima	F	8345671111	dv@yahoo.com	OakPlace	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.coom	Oak Street	123	1/21/1995	500
17338893	Pooja Singh	F	7457894512	pooja@umkc.edu	Oak Place	421	9/15/1997	500

RESIDENT (*INSERT*)

Inserting a new record

Resident

Name: Emily Jones
StudentID: 1673452
back
PhoneNo: 8167534456
Hall: Johnson
Email: emily@gmail.com
RoomNo: 342
Dob: 1995-06-12
Balance: 500
Gender: F

Insert | Update | Delete

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
12277893	Aishvwarya Iyer	F	8167456678	ai@gmail.com	Johnson		95	500
15673454	John Canady	M	2345679986	jh@yahoo.com	Oak Pla		994	500
16223345	Katy Bills	F	8165678909	katy@yahoo.com	Johnson		90	500
16225564	Lily James	F	816753546	lily@gmail.com	Oak Stre		96	500
16334562	Derik Gomez	M	8156784534	der@yahoo.com	Oak Stre		93	500
16444432	Shrikanth Sai	M	9654231923	sr@gmail.com	Johnson		92	500
16547687	Bharath Kumar	M	8167890098	bn@gmail.com	Oak Street	256	6/20/1995	500
16554432	Rachel Young	F	8134567786	rc@gmail.com	Johnson	156	11/1/1991	500
16778763	Shreya Patel	F	8167651234	sh@gmail.com	Johnson	247	7/17/1998	500
16778896	Divya Neelima	F	8345671111	dv@yahoo.com	OakPlace	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.coom	Oak Street	123	1/21/1995	500
17338893	Pooja Singh	F	7457894512	pooja@umkc.edu	Oak Place	421	9/15/1997	500

Record Inserted Successfully

OK

Record inserted successfully

Resident

Name
StudentID
back
PhoneNo
Hall
Email
RoomNo
Dob: 2017-11-26
Balance
Gender

Insert | Update | Delete

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishvwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	500
15673454	John Canady	M	2345679986	jh@yahoo.com	Oak Place	125	10/31/1994	500
16223345	Katy Bills	F	8165678909	katy@yahoo.com	Johnson	343	6/23/1990	500
16225564	Lily James	F	816753546	lily@gmail.com	Oak Street	236	3/14/1996	500
16334562	Derik Gomez	M	8156784534	der@yahoo.com	Oak Street	239	6/21/1993	500
16444432	Shrikanth Sai	M	9654231923	sr@gmail.com	Johnson	433	8/11/1992	500
16547687	Bharath Kumar	M	8167890098	bn@gmail.com	Oak Street	256	6/20/1995	500
16554432	Rachel Young	F	8134567786	rc@gmail.com	Johnson	156	11/1/1991	500
16778763	Shreya Patel	F	8167651234	sh@gmail.com	Johnson	247	7/17/1998	500
16778896	Divya Neelima	F	8345671111	dv@yahoo.com	OakPlace	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.coom	Oak Street	123	1/21/1995	500

RESIDENT (*UPDATE*)

Updating room number of John Canady from 125 to 126

Resident

Name: John Canady
StudentID: 15673454
back
PhoneNo: 2345679986
Hall: Oak Place
Email: jh@yahoo.com
RoomNo: 126
Dob: 1994-10-31
Balance: 500
Gender: M

Insert | Update | Delete

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishvwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	500
15673454	John Canady	M	2345679986	jh@yahoo.com	Oak Place	125	10/31/1994	500
16223345	Katy Bills	F	8165678909	katy@yahoo.com	Johnson	343	6/23/1990	500
16225564	Lily James	F	816753546	lily@gmail.com	Oak Street	236	3/14/1996	500
16334562	Derik Gomez	M	8156784534	der@yahoo.com	Oak Street	239	6/21/1993	500
16444432	Shrikanth Sai	M	9654231923	sr@gmail.com	Johnson	433	8/11/1992	500
16547687	Bharath Kumar	M	8167890098	bn@gmail.com	Oak Street	256	6/20/1995	500
16554432	Rachel Young	F	8134567786	rc@gmail.com	Johnson	156	11/1/1991	500
16778763	Shreya Patel	F	8167651234	sh@gmail.com	Johnson	247	7/17/1998	500
16778896	Divya Neelima	F	8345671111	dv@yahoo.com	OakPlace	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.coom	Oak Street	123	1/21/1995	500

Resident

Name: John Canady
StudentID: 15673454
back
PhoneNo: 2345679986
Hall: Oak Place
Email: jh@yahoo.com
RoomNo: 126
Dob: 1994-10-31
Balance: 500
Gender: M

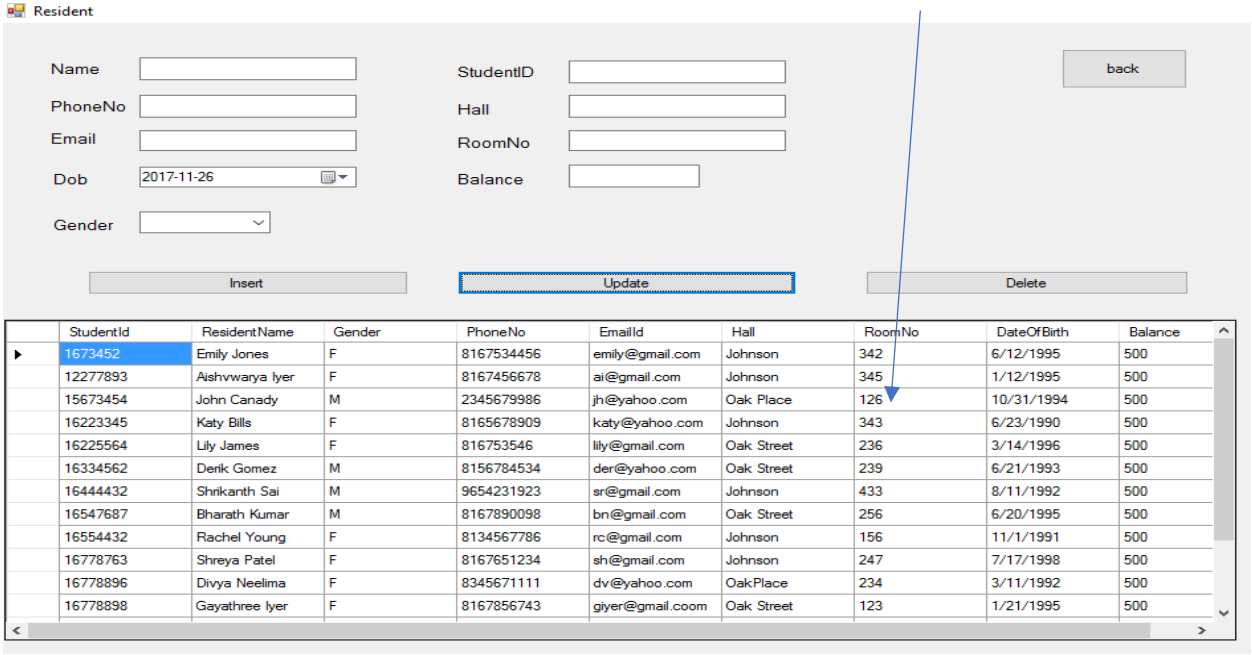
Insert | Update | Delete

Record Updated Successfully

OK

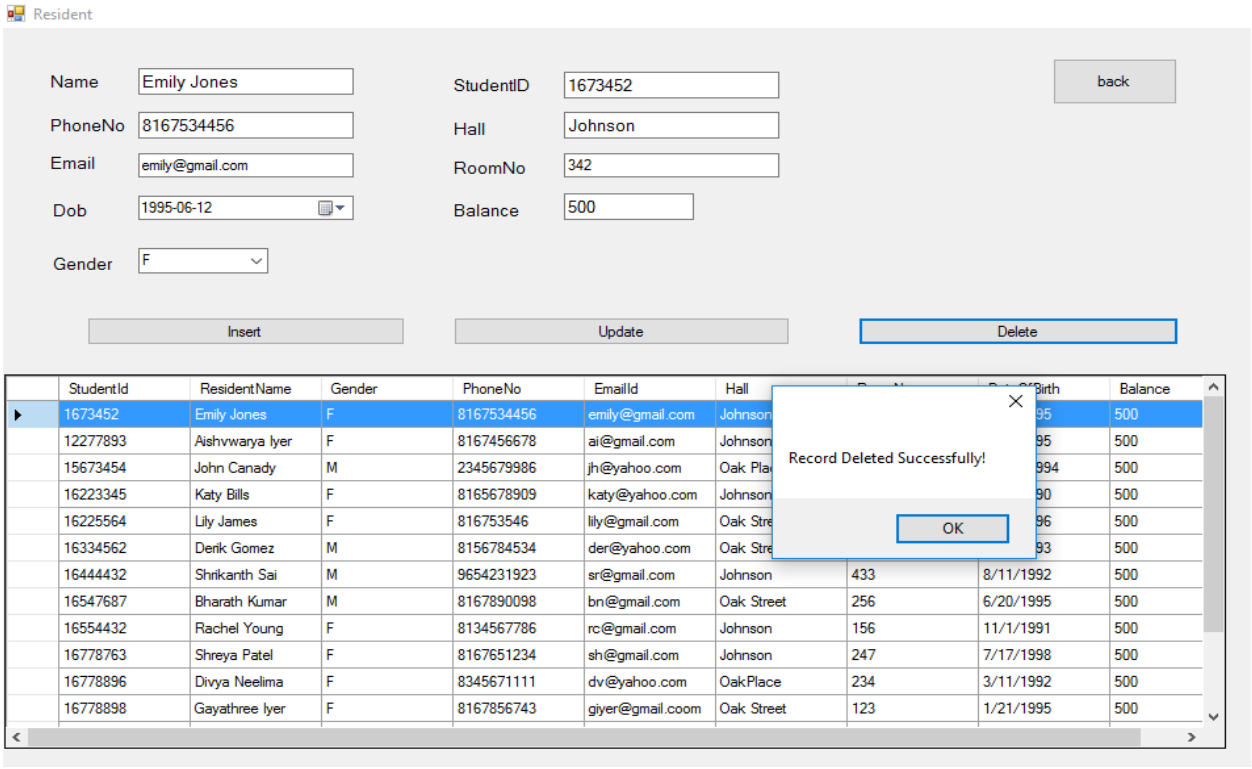
StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson		95	500
12277893	Aishvwarya Iyer	F	8167456678	ai@gmail.com	Johnson		95	500
15673454	John Canady	M	2345679986	jh@yahoo.com	Oak Pla		994	500
16223345	Katy Bills	F	8165678909	katy@yahoo.com	Johnson		90	500
16225564	Lily James	F	816753546	lily@gmail.com	Oak Stre		96	500
16334562	Derik Gomez	M	8156784534	der@yahoo.com	Oak Stre		93	500
16444432	Shrikanth Sai	M	9654231923	sr@gmail.com	Johnson	433	8/11/1992	500
16547687	Bharath Kumar	M	8167890098	bn@gmail.com	Oak Street	256	6/20/1995	500
16554432	Rachel Young	F	8134567786	rc@gmail.com	Johnson	156	11/1/1991	500
16778763	Shreya Patel	F	8167651234	sh@gmail.com	Johnson	247	7/17/1998	500
16778896	Divya Neelima	F	8345671111	dv@yahoo.com	OakPlace	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.coom	Oak Street	123	1/21/1995	500

John Canady's room number updated successfully from 125 to 126



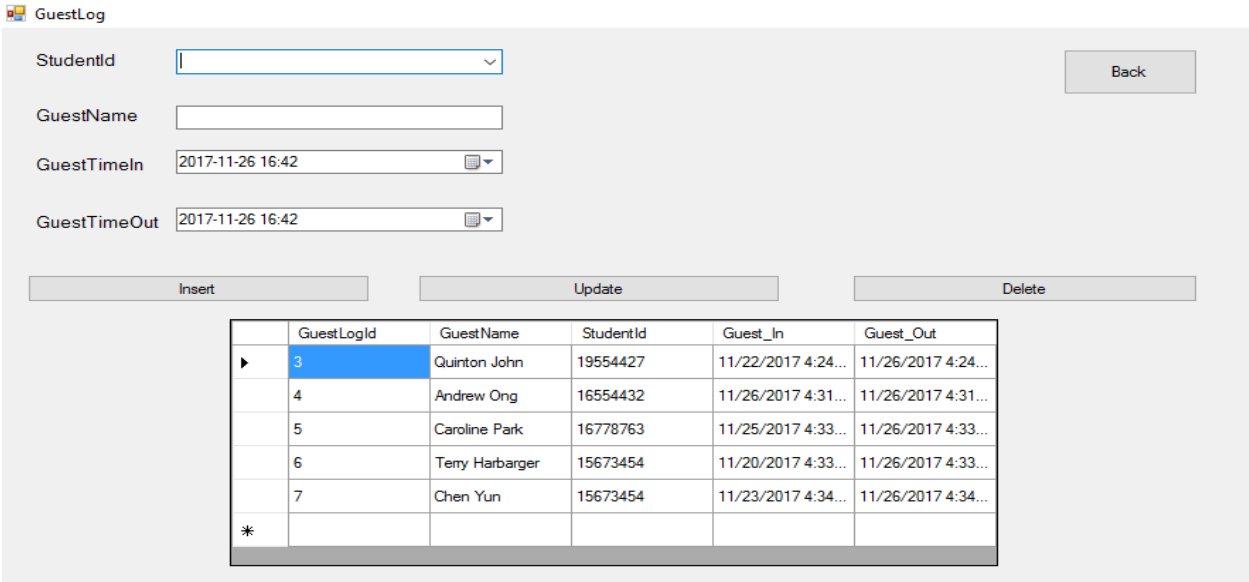
RESIDENT *(DELETE)*

If a student is not a resident anymore, we will delete his/her record from the table.



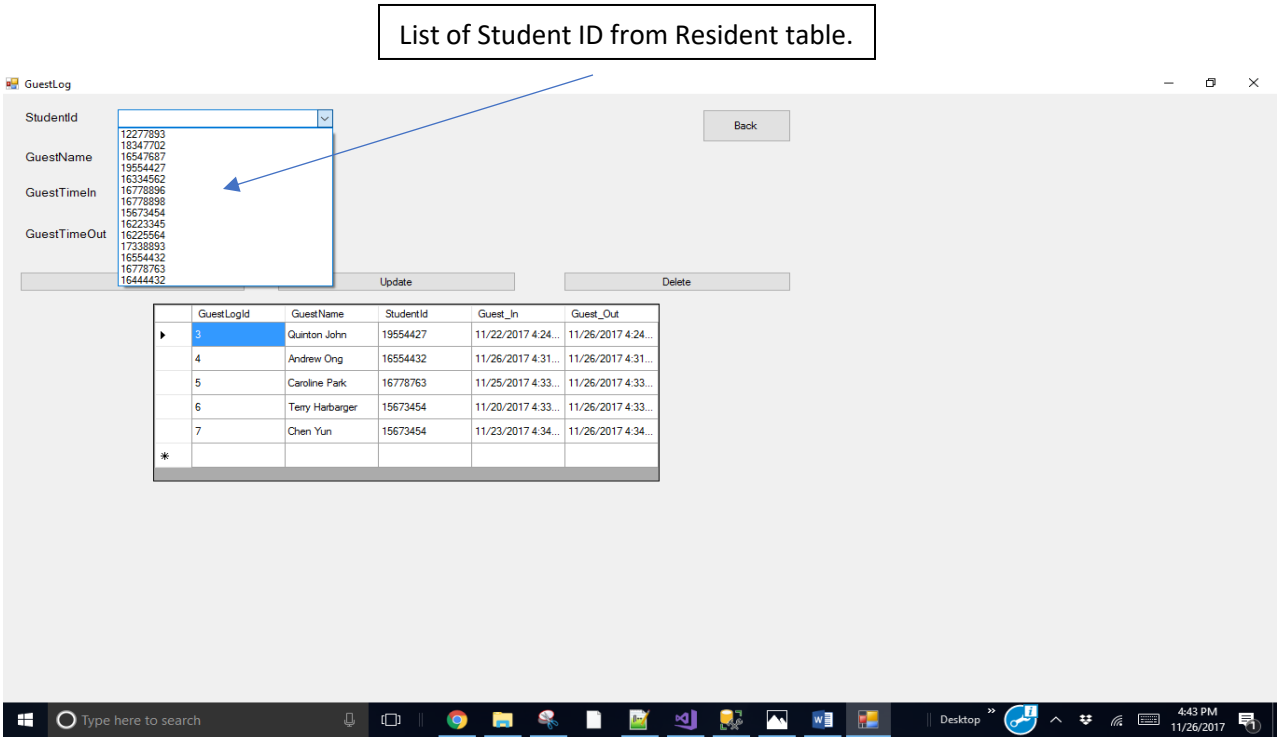
GUEST LOG

The Guest Log helps to manage the guest check-ins and check-outs. It includes GuestLog ID, Guest name, Resident's Student ID, Guest in time and Guest out time.

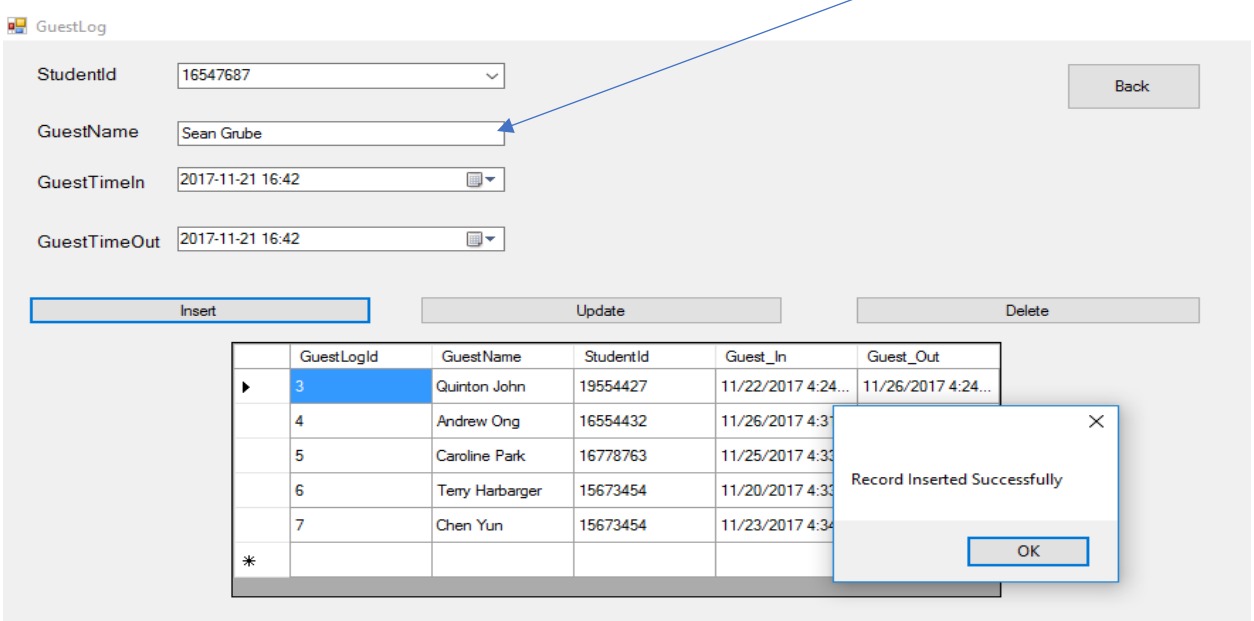


GUEST LOG (*INSERT*)

When inserting a new record, Student ID will be listed from the Resident table.

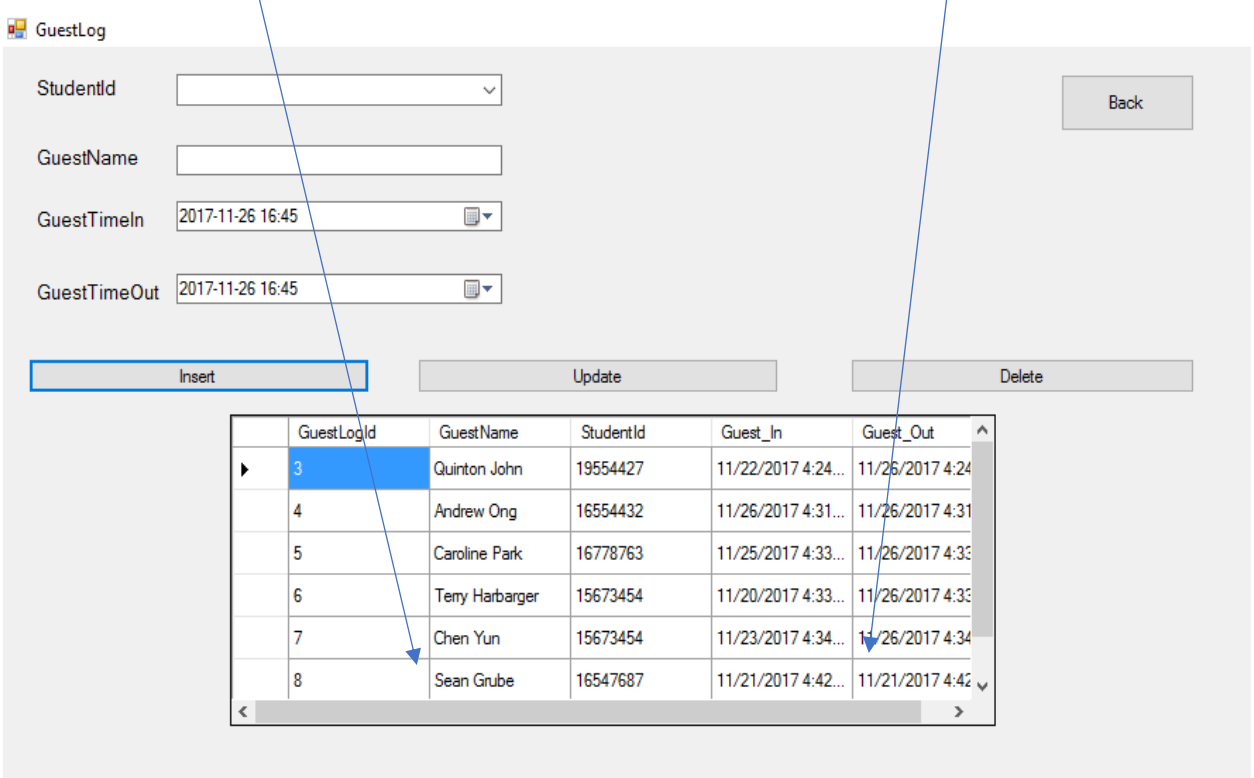


Guest-Sean Grube inserted for Resident with ID: 16547687



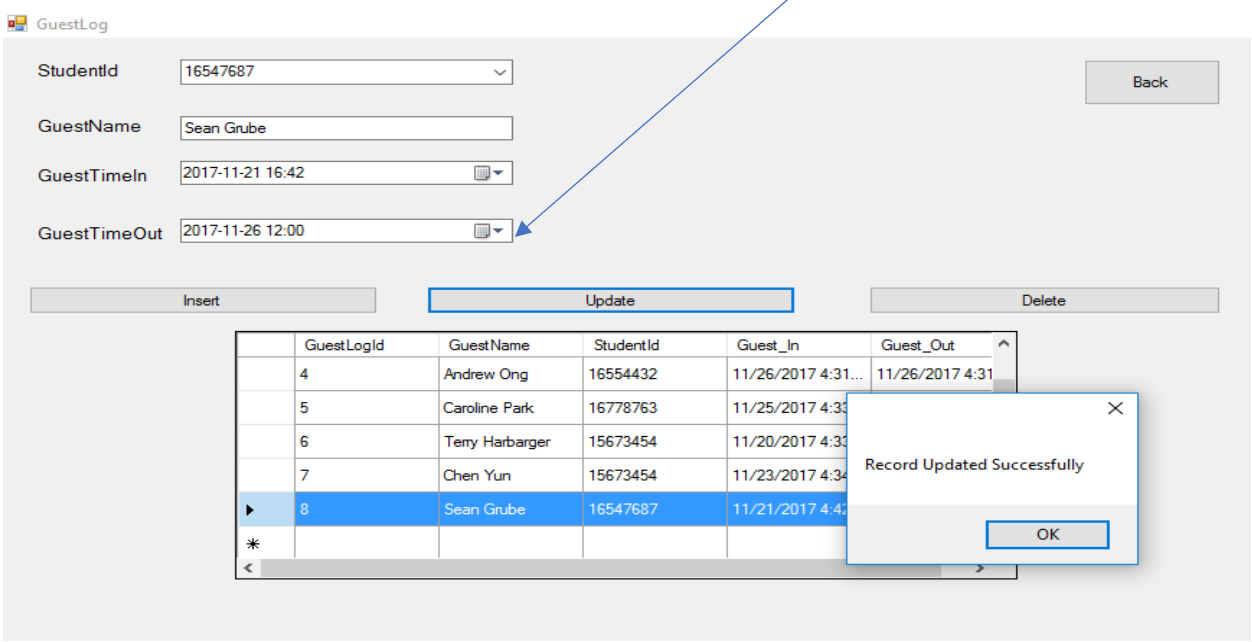
Guest-Sean Grube inserted successfully

Guest Time-out by default to current time

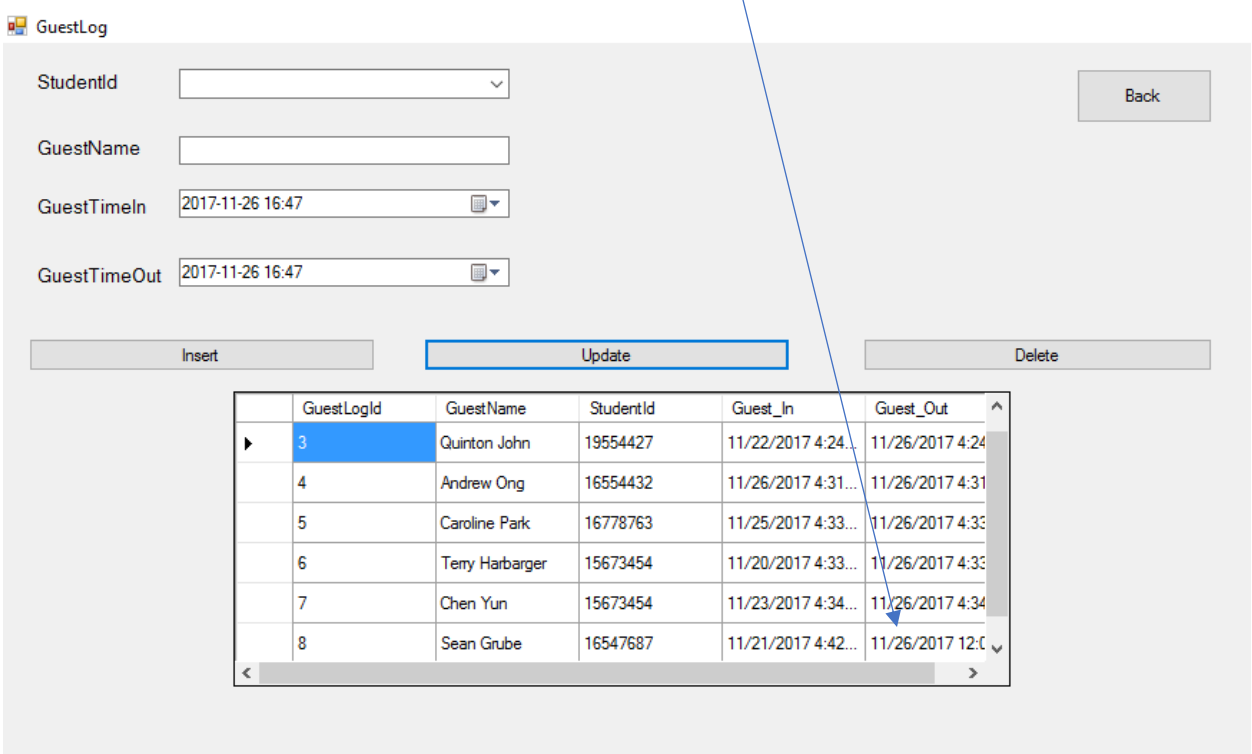


GUEST LOG (*UPDATE*)

When the guest leave, we will update the guest time out

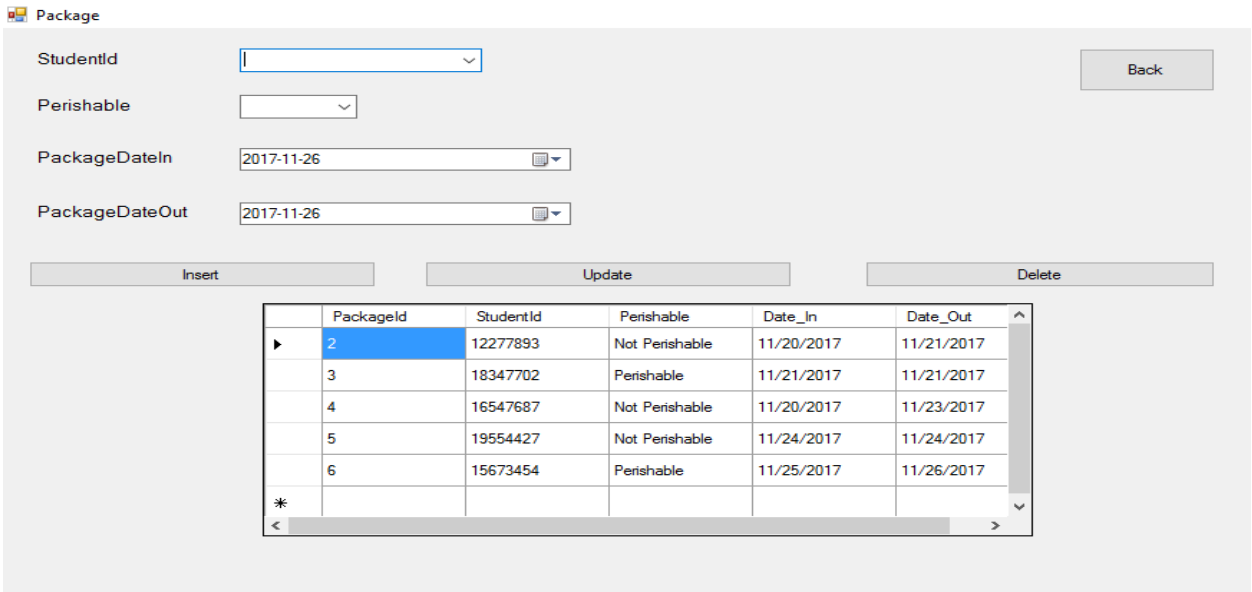


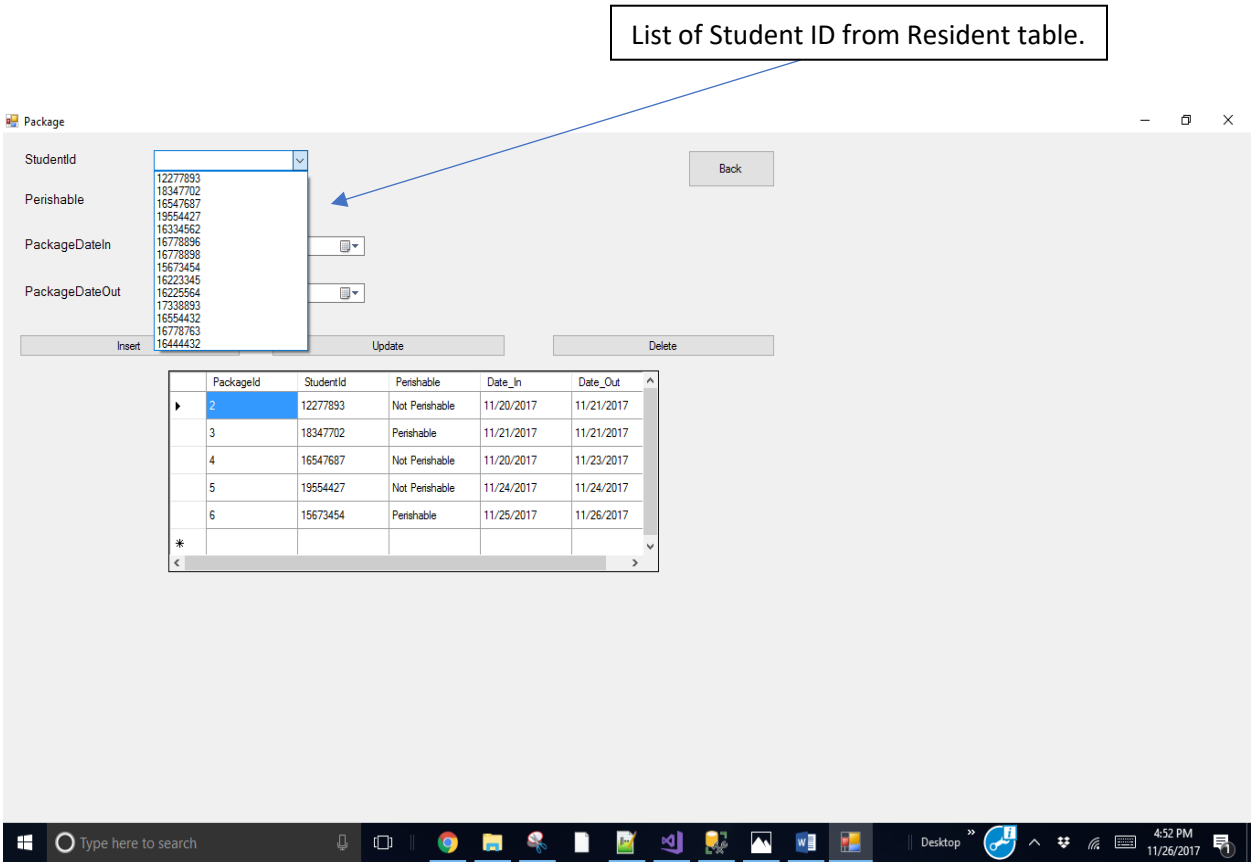
Guest time out updated successfully



PACKAGE

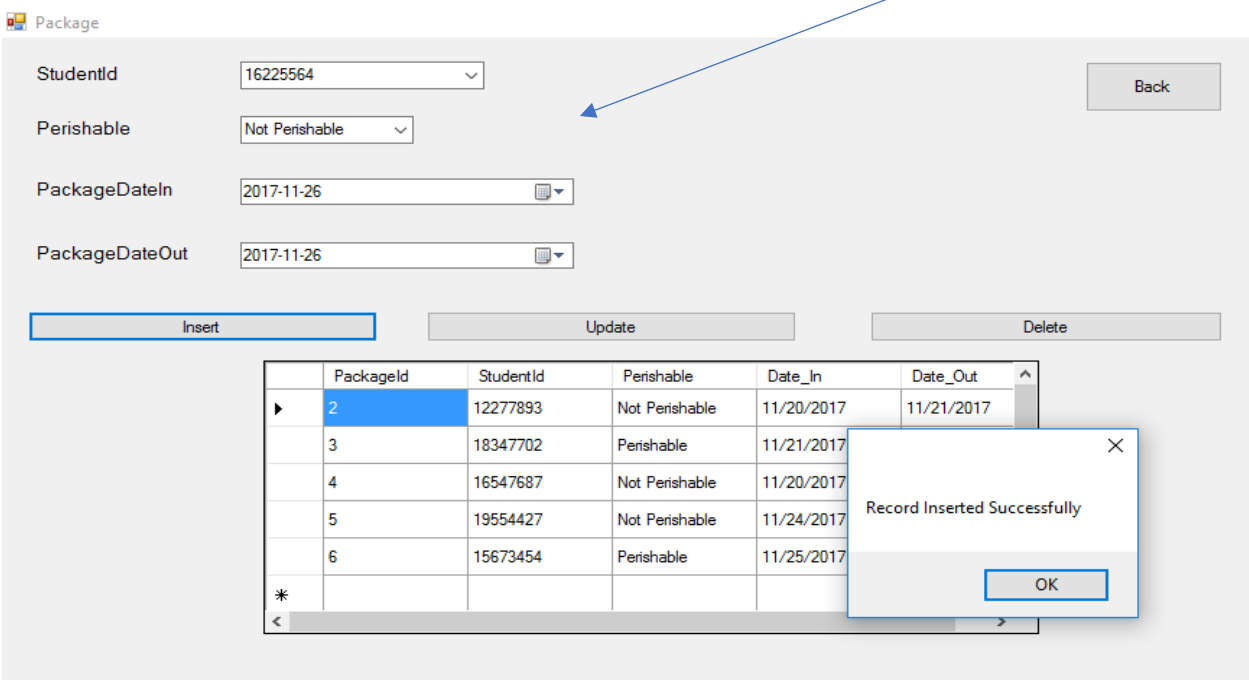
The Package Log helps to keep track of what packages are received by the resident and when the residents pick their package.



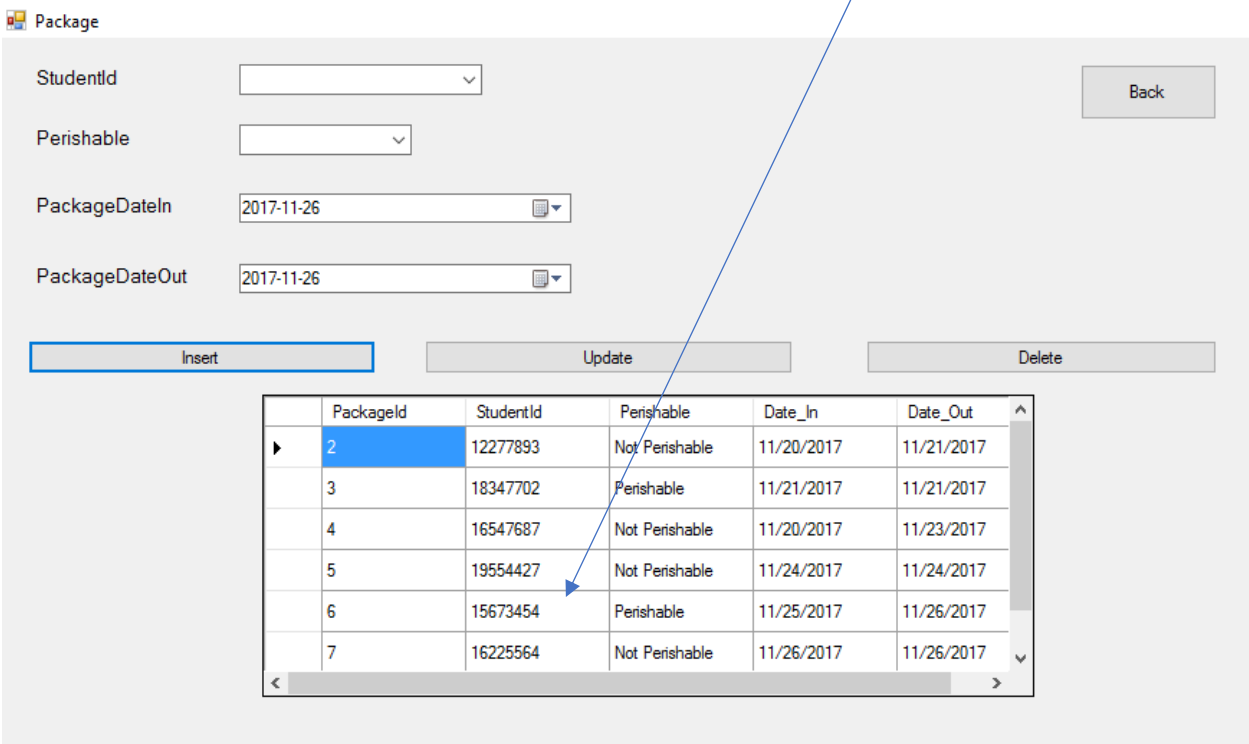


PACKAGE (*INSERT*)

Inserting a record: Package for Student ID: 16225564

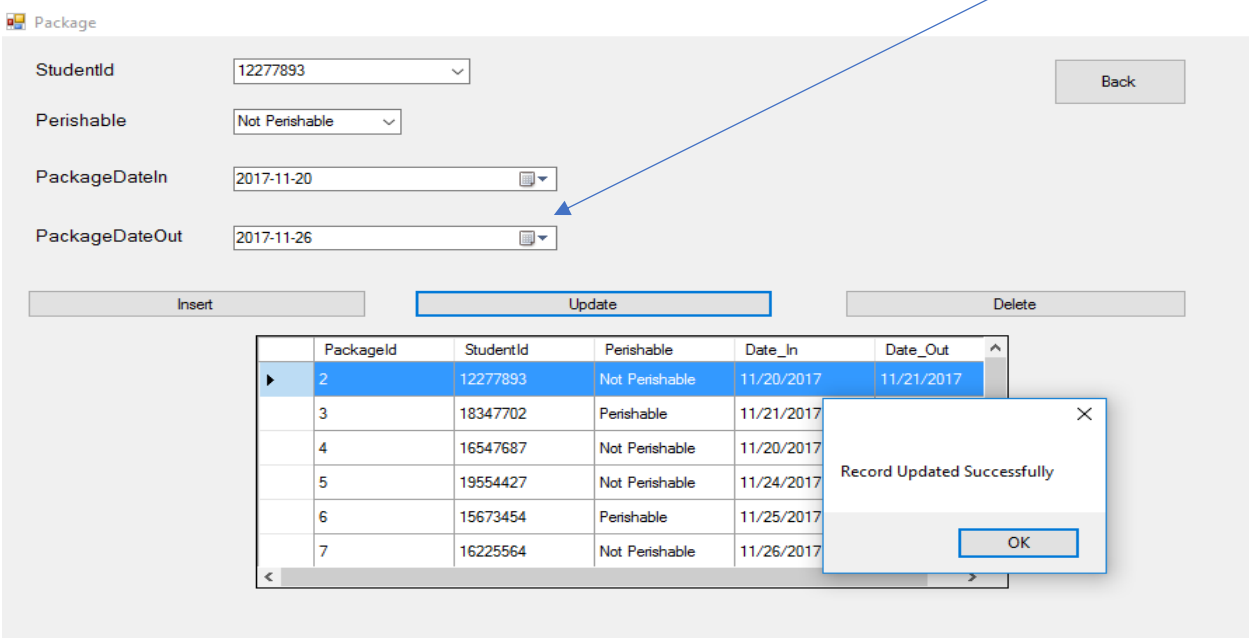


Insertion successful

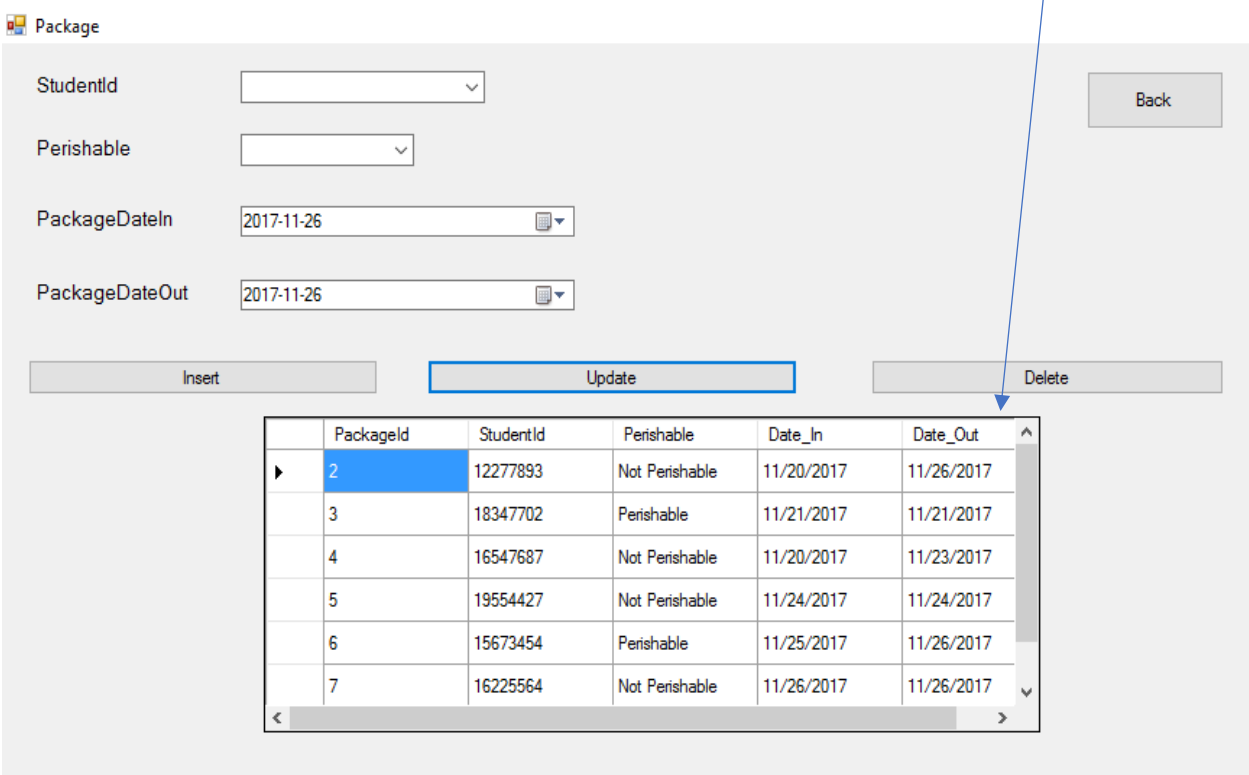


PACKAGE (*UPDATE*)

Updating guest check out time

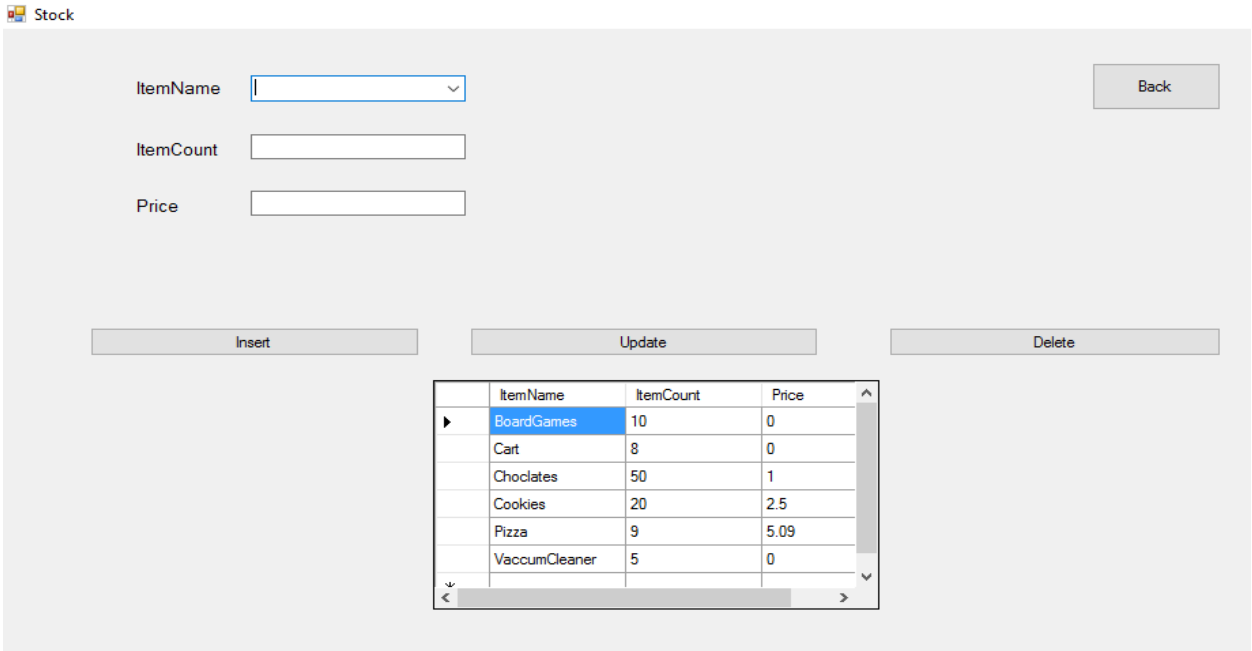


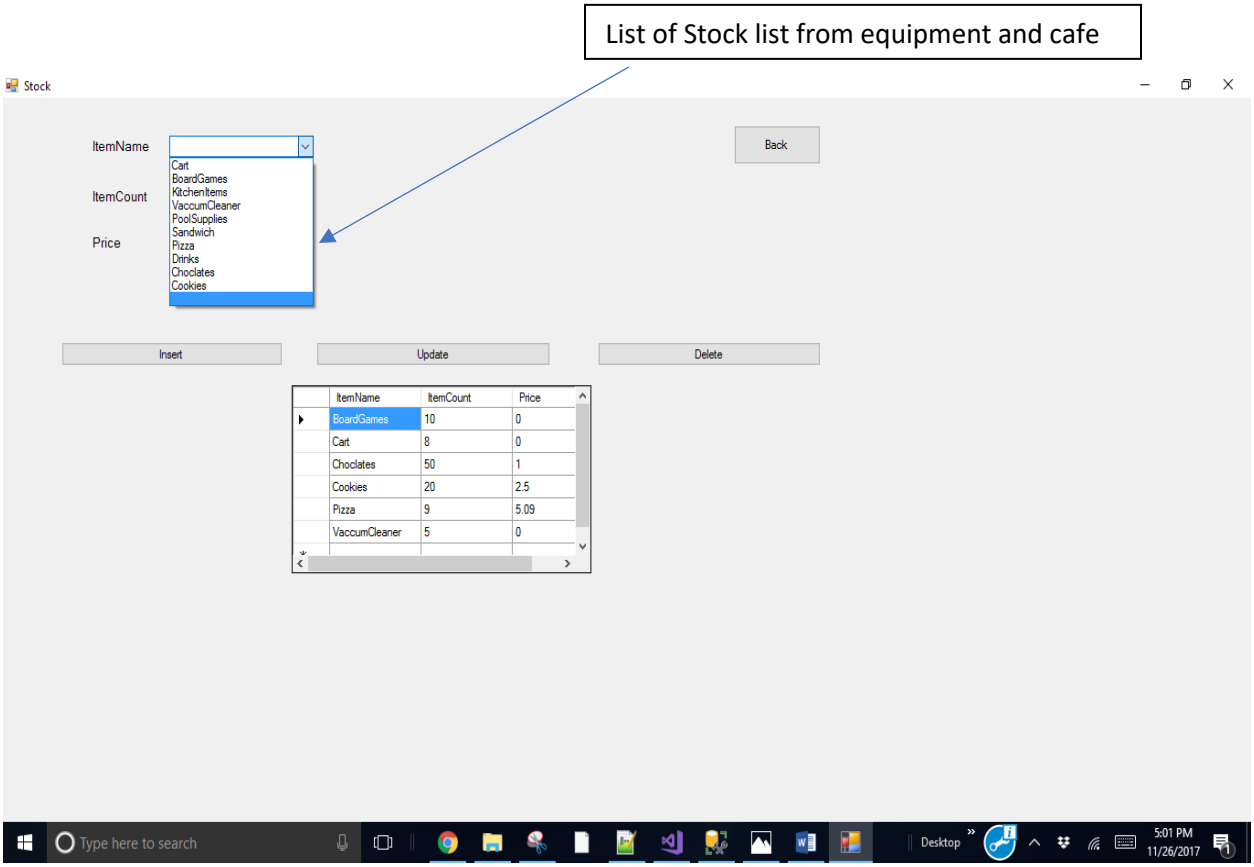
Guest check out time updated successfully



STOCK

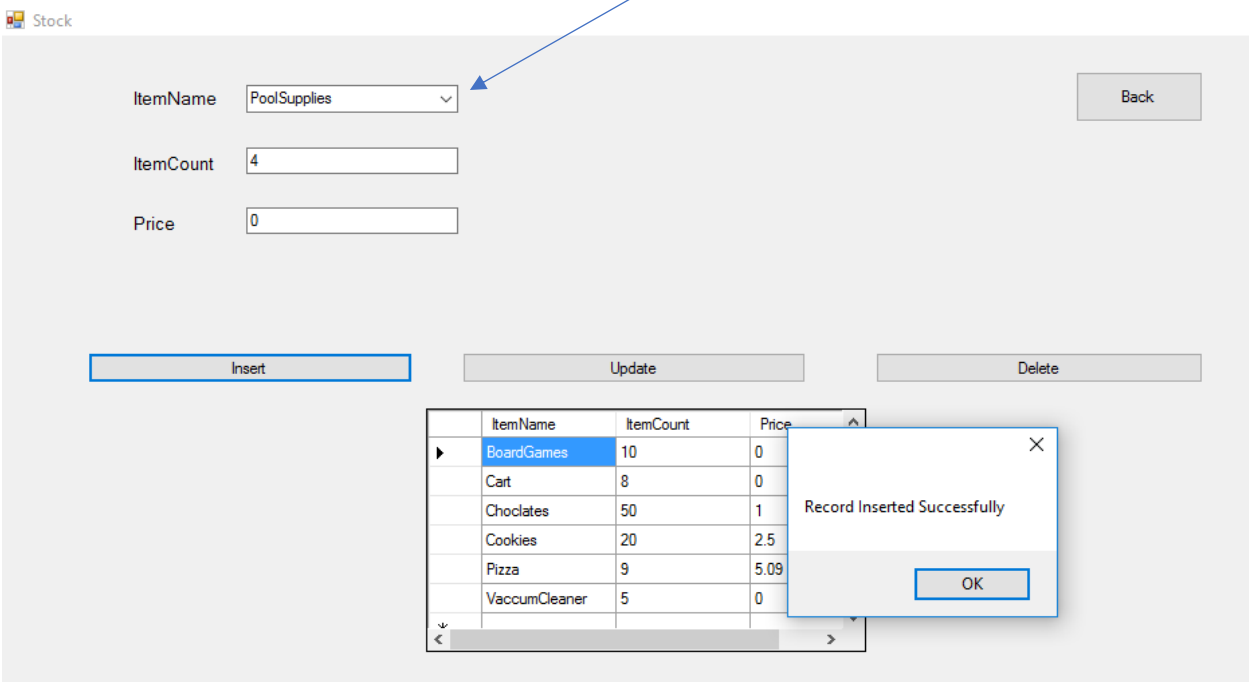
The Stock table is used to maintain the stock for Café and Equipment table. It includes Item name, count and price.



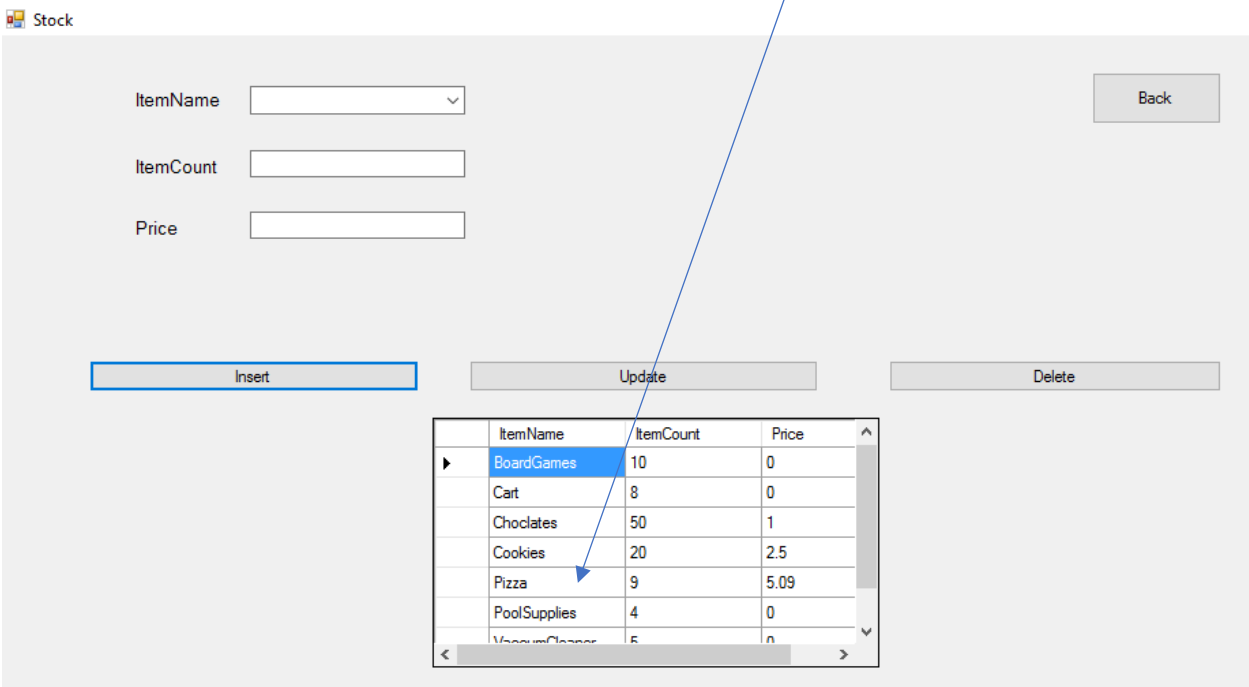


STOCK (*INSERT*)

Inserting stock: Pool supplies

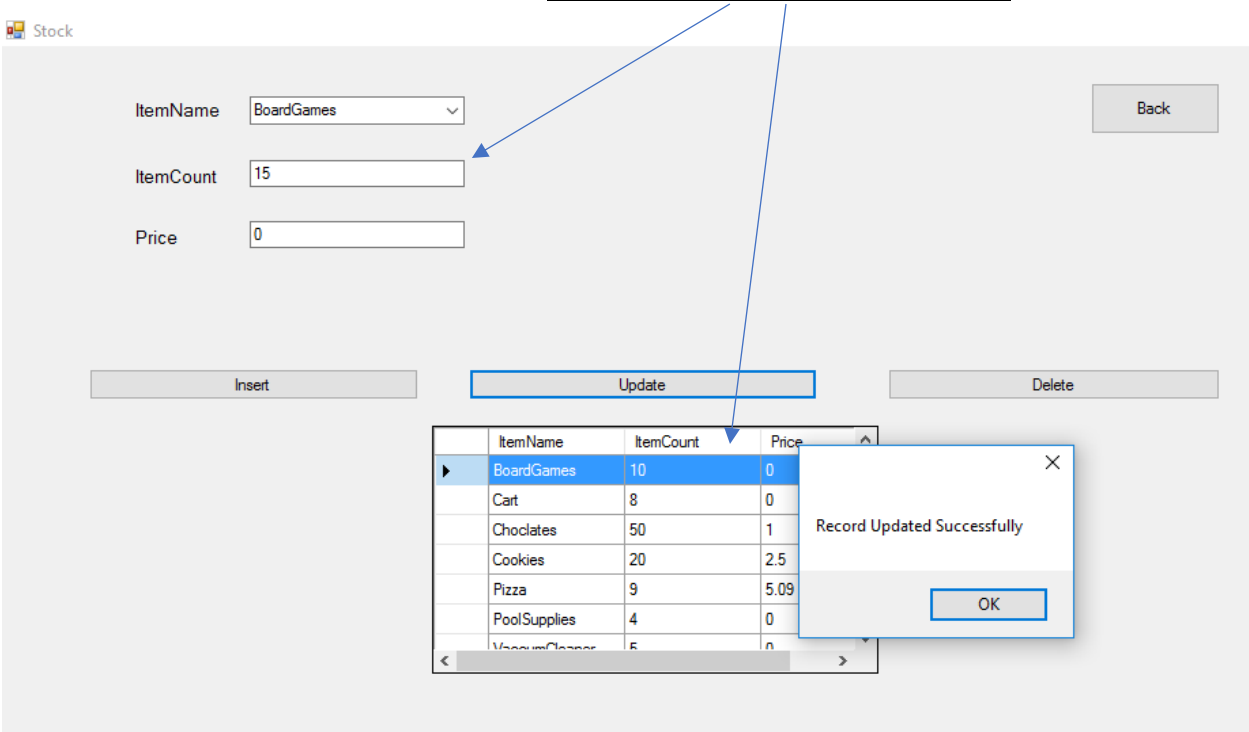


Pool supplies inserted

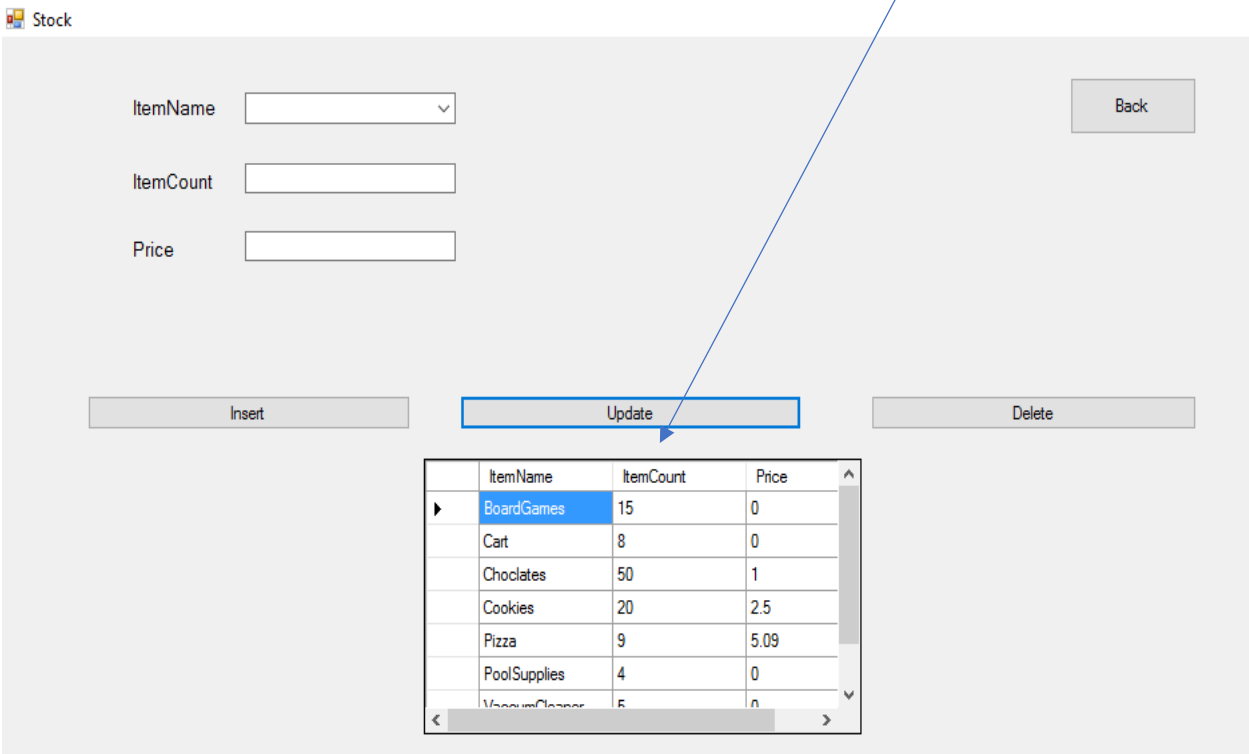


STOCK (*UPDATE*)

Updating item count from 10 to 15

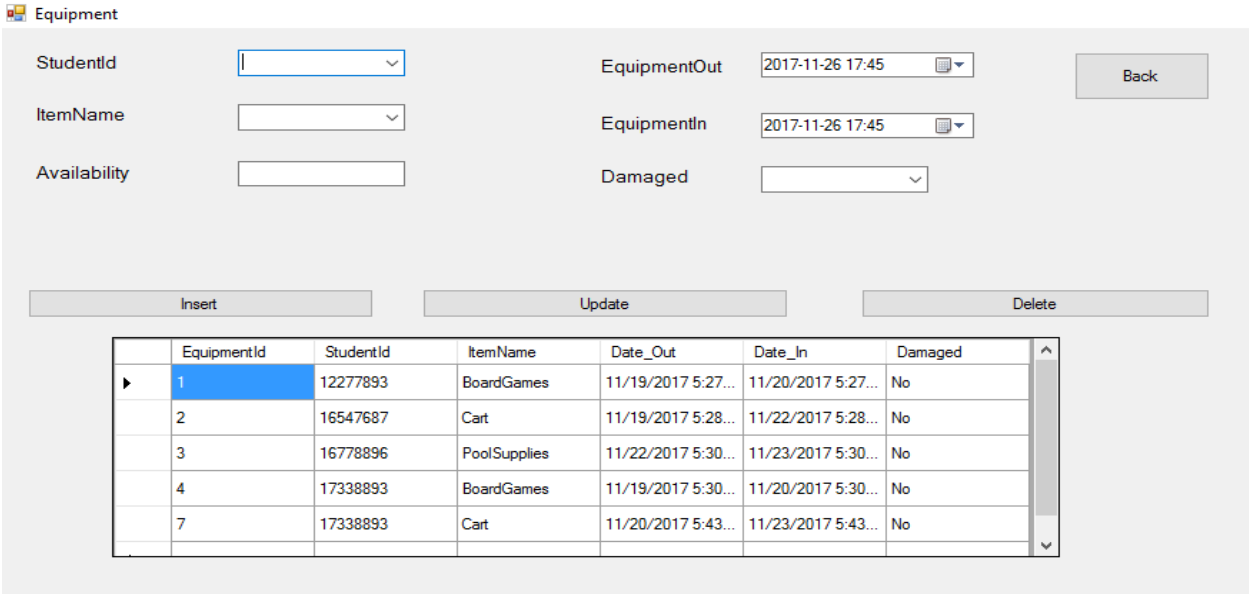


Item count updated successfully to 15



EQUIPMENT

The Equipment table maintains the equipment checked-in and checked-out by the resident.



EquipmentId	StudentId	ItemName	Date_Out	Date_In	Damaged
1	12277893	BoardGames	11/19/2017 5:27...	11/20/2017 5:27...	No
2	16547687	Cart	11/19/2017 5:28...	11/22/2017 5:28...	No
3	16778896	PoolSupplies	11/22/2017 5:30...	11/23/2017 5:30...	No
4	17338893	BoardGames	11/19/2017 5:30...	11/20/2017 5:30...	No
7	17338893	Cart	11/20/2017 5:43...	11/23/2017 5:43...	No

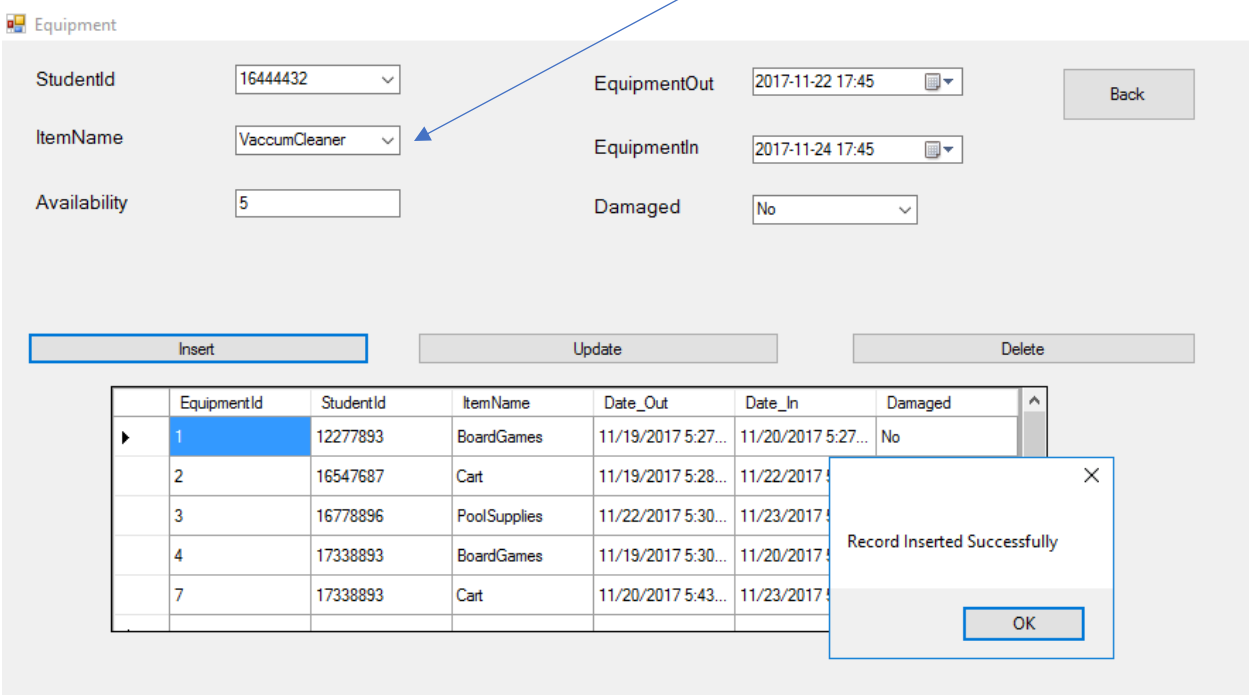
- The Equipment that the residents can check-out are Cart, Board games, Pool supplies, Vacuum Cleaner.

- Inserting these details in the database allows to keep track of which equipment is checked-out by the resident and when the resident check-in the equipment back. Also, whether the equipment turned in is damaged or not.

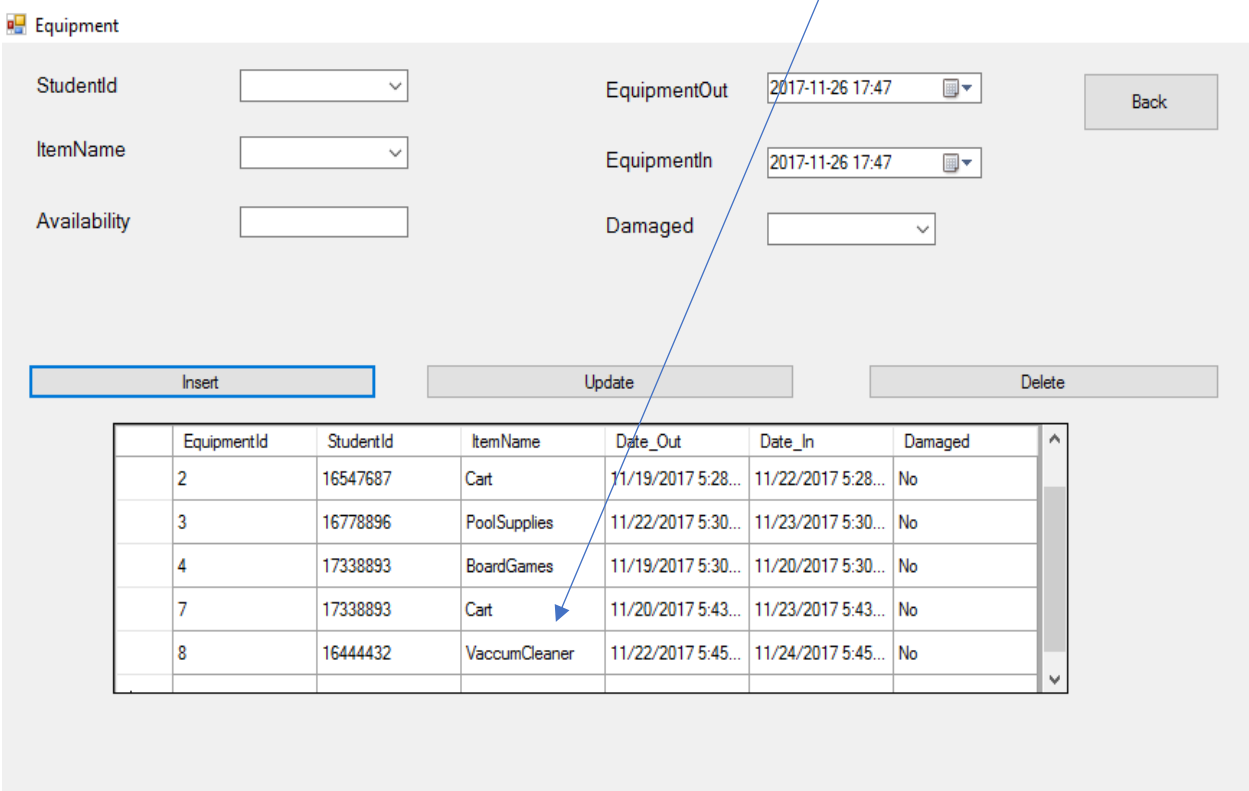
- The date-in value is default value when the record is inserted. The date-in can then be updated to reflect the value of the date when the equipment is checked back in.

EQUIPMENT (*INSERT*)

Resident with Student ID checks out an equipment



Record inserted successfully



EQUIPMENT (*UPDATE*)

Equipment check-out date updated to 2017-11-24 from default value

Equipment

StudentId 16554432

ItemName BoardGames

Availability 11

EquipmentOut 2017-11-20 17:47

EquipmentIn 2017-11-24 15:47

Damaged No

Back

Insert | Update | Delete

EquipmentId	StudentId	ItemName	Date_Out	Date_In	Damaged
3	16778896	PoolSupplies	11/22/2017 5:30...	11/23/2017 5:30...	No
4	17338893	BoardGames	11/19/2017 5:30...	11/20/2017 5:30...	No
7	17338893	Cart	11/20/2017 5:43...	11/23/2017 5:43...	No
8	16444432	VaccumCleaner	11/22/2017 5:45...	11/24/2017 5:45...	No
9	16554432	BoardGames	11/20/2017 5:47...	11/26/2017 5:47...	No

Record Updated Successfully

OK

Equipment check-out date updated successfully

Equipment

StudentId

ItemName

Availability

EquipmentOut 2017-11-26 17:51

EquipmentIn 2017-11-26 17:51

Damaged

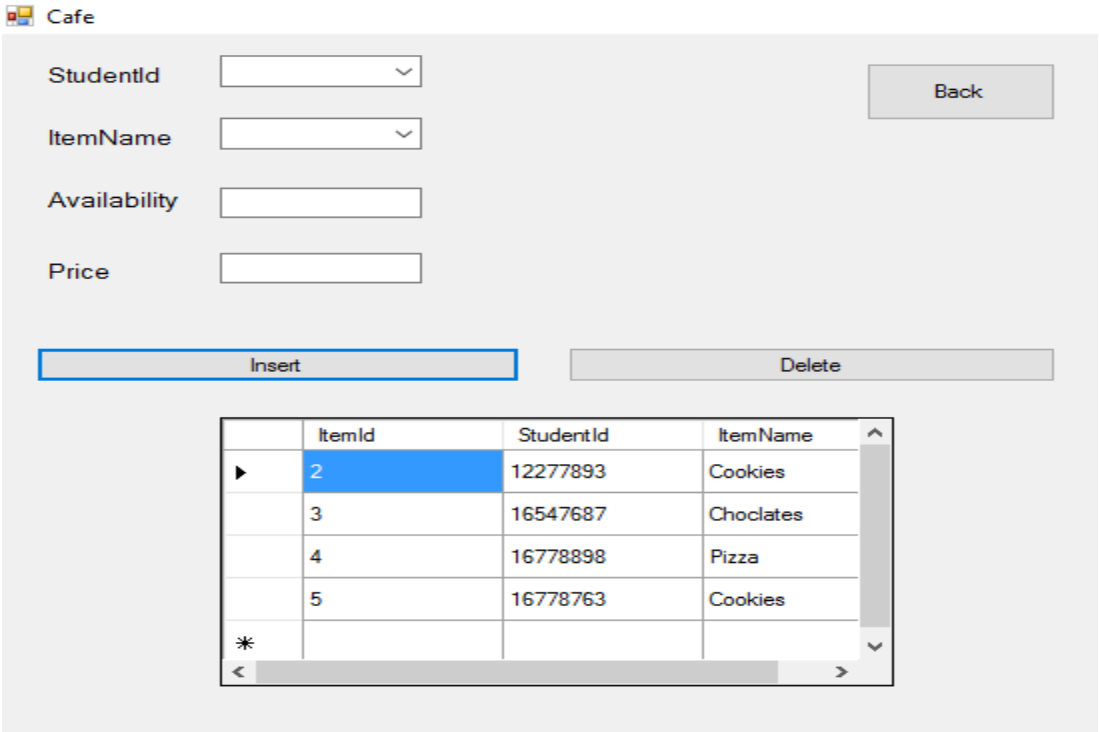
Back

Insert | Update | Delete

EquipmentId	StudentId	ItemName	Date_Out	Date_In	Damaged
3	16778896	PoolSupplies	11/22/2017 5:30...	11/23/2017 5:30...	No
4	17338893	BoardGames	11/19/2017 5:30...	11/20/2017 5:30...	No
7	17338893	Cart	11/20/2017 5:43...	11/23/2017 5:43...	No
8	16444432	VaccumCleaner	11/22/2017 5:45...	11/24/2017 5:45...	No
9	16554432	BoardGames	11/20/2017 5:47...	11/24/2017 3:47...	No

CAFÉ

The Café table maintains the items ordered by the resident.

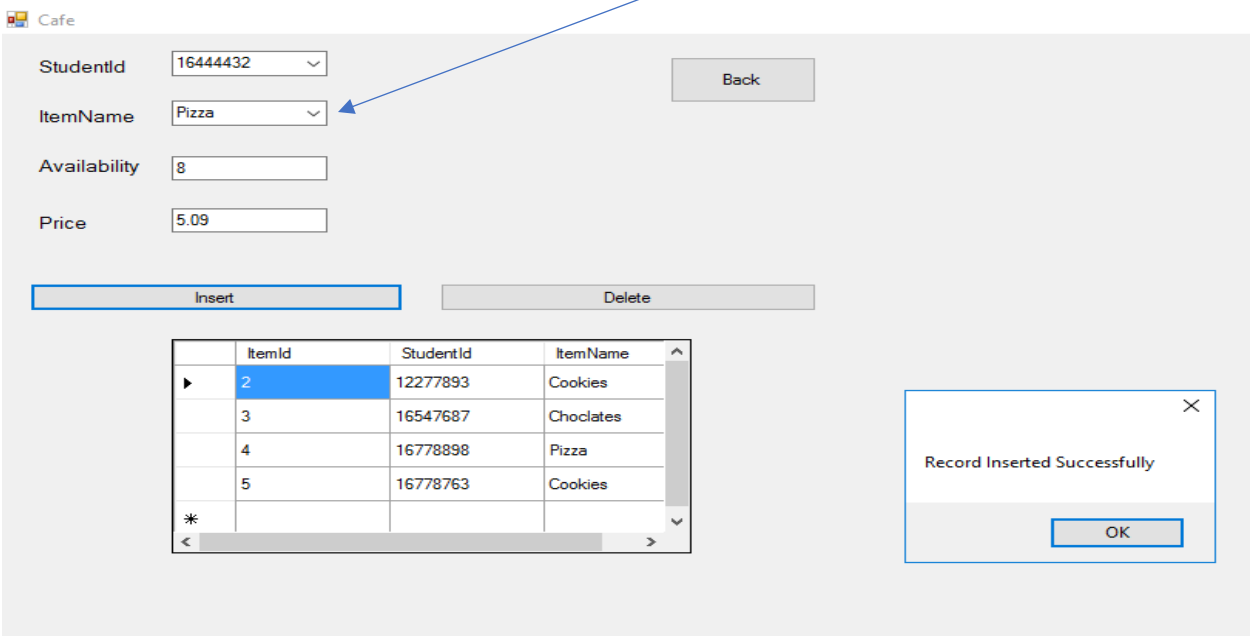


- When the resident orders an item, the availability of that item reduces by 1.

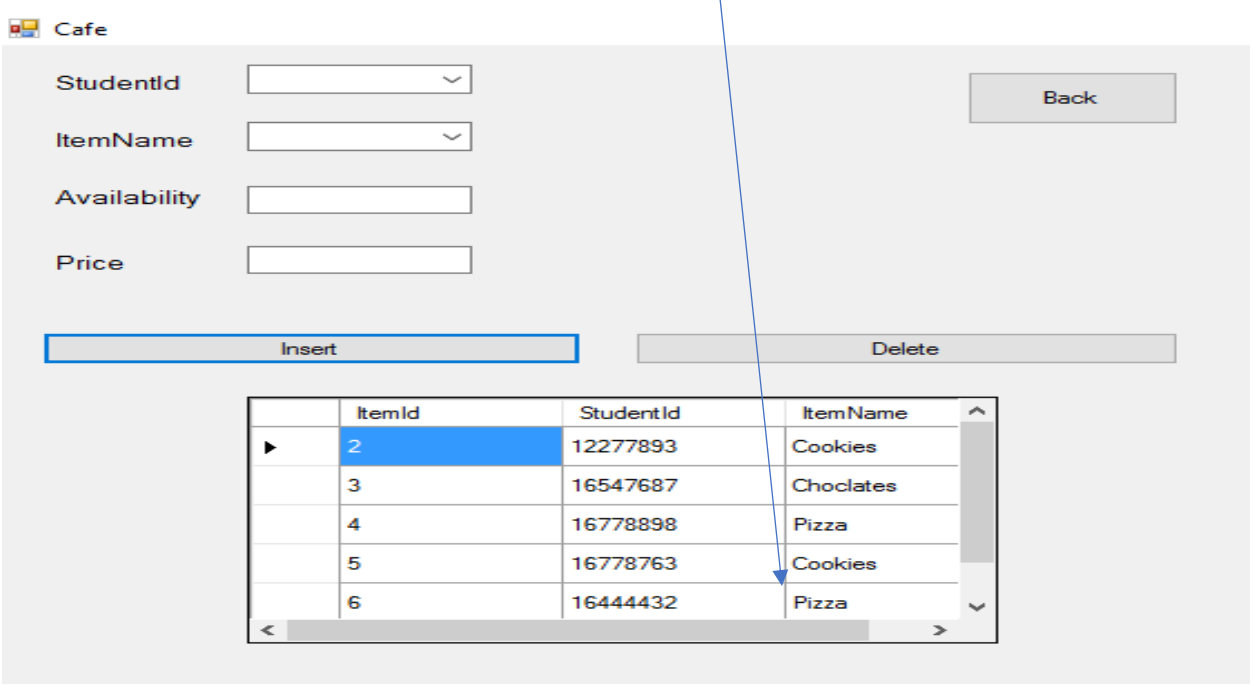
- The price of that item will be deducted from the resident's balance.

CAFE (*INSERT*)

Resident orders a Pizza from café. The price will be deducted from his balance.



Record inserted successfully



ITEM PRICE DEDUCTED FROM BALANCE

When resident orders from café, the item price will be deducted from the resident

Cafe

StudentId

ItemName

Availability

Price

Back

Insert

Delete

ItemId	StudentId	ItemName
2	12277893	Cookies
3	16547687	Choclates
4	16778898	Pizza
5	16778763	Cookies
6	16444432	Pizza

Resident

Name

PhoneNo

Email

Dob 2017-11-26

Gender

StudentID

Hall

RoomNo

Balance

back

Insert

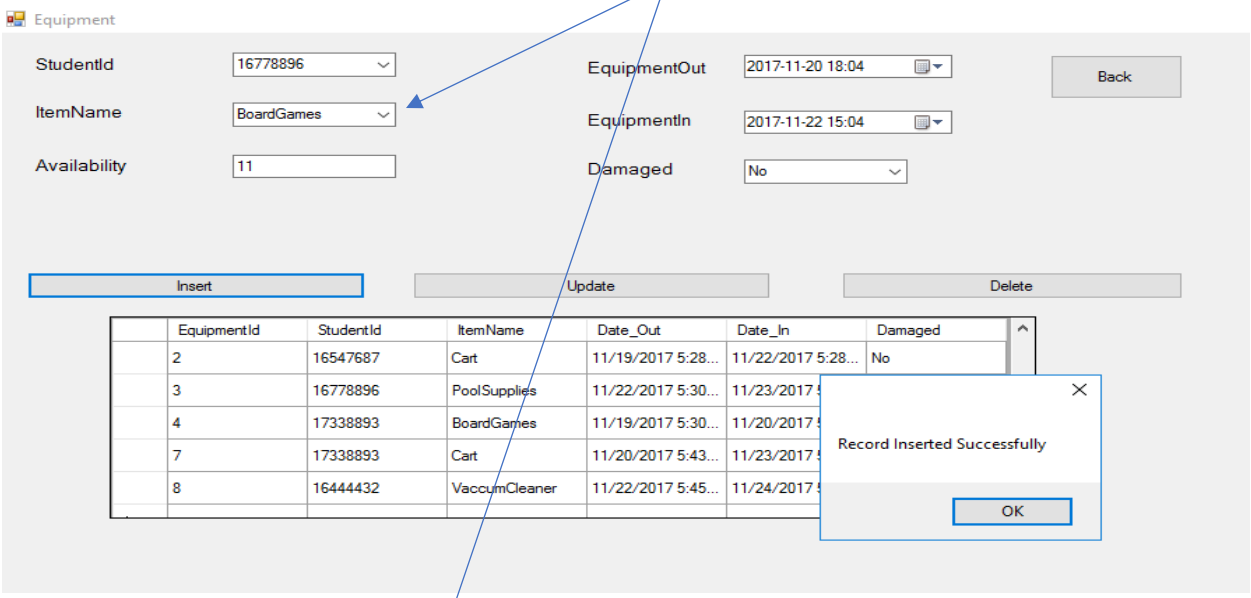
Update

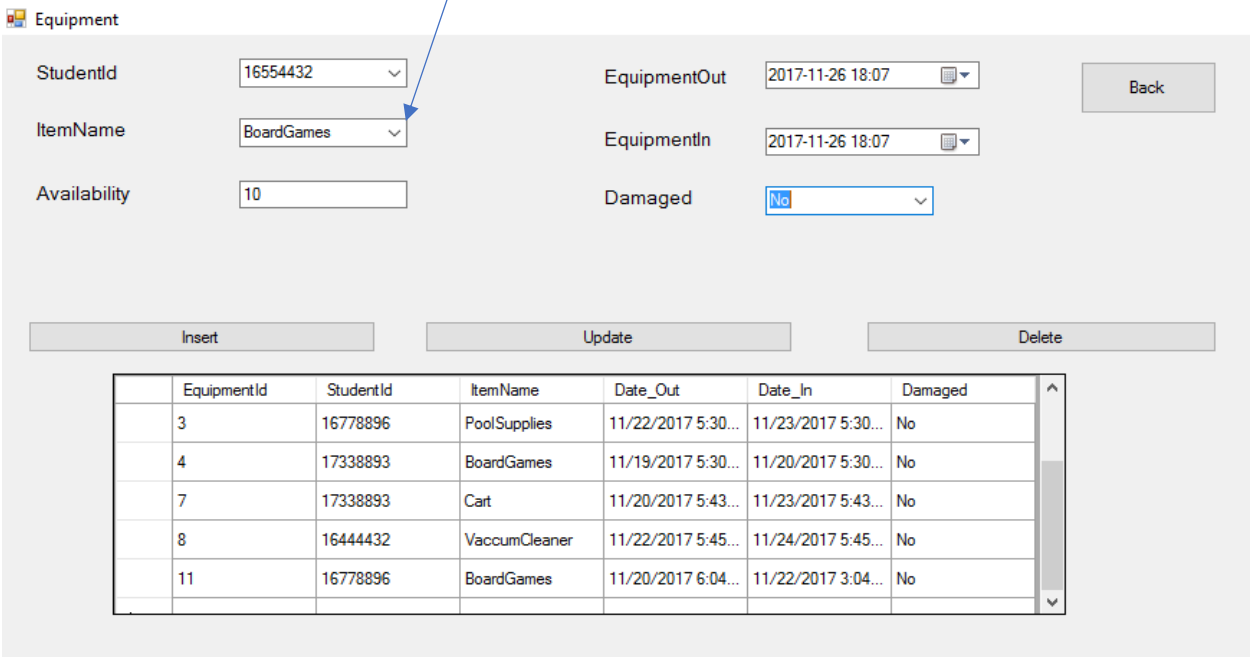
Delete

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
12277893	Aishvwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	497.5
15673454	John Canady	M	2345679986	jh@yahoo.com	Oak Place	126	10/31/1994	500
16223345	Katy Bills	F	8165678909	katy@yahoo.com	Johnson	343	6/23/1990	500
16225564	Lily James	F	816753546	lily@gmail.com	Oak Street	236	3/14/1996	500
16334562	Derik Gomez	M	8156784534	der@yahoo.com	Oak Street	239	6/21/1993	500
16444432	Shrikanth Sai	M	9654231923	sr@gmail.com	Johnson	433	8/11/1992	494.90999984
16547687	Bharath Kumar	M	8167890098	bn@gmail.com	Oak Street	256	6/20/1995	499
16554432	Rachel Young	F	8134567786	rc@gmail.com	Johnson	156	11/1/1991	500
16778763	Shreya Patel	F	8167651234	sh@gmail.com	Johnson	247	7/17/1998	497.5
16778896	Divya Neelima	F	8345671111	dv@yahoo.com	OakPlace	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.coom	Oak Street	123	1/21/1995	494.90999984
17338893	Pooja Singh	F	7457894512	pooja@umkc.edu	Oak Place	421	9/15/1997	500

AVAILABILITY REDUCED WHEN AN EQUIPMENT IS CHECKED-OUT

When a resident checks-out a board game, the availability shows 11. After insertion of the record is complete, the next time when a resident checks-out board game, availability shows 10.





- Code written to reflect deduction of the item price in the resident's balance.

```
        }
        DR2.Close();
        con.Close();
        DisplayData();
    }
    private void btn_Insert_Click(object sender, EventArgs e)
    {
        if (txt_iname.Text!=""&&
            txt_id.Text != ""&&
            txt_price.Text!="")
        {
            cmd = new SqlCommand("insert into dbo.Cafe(StudentId,ItemName) values (@id,@iname)", con);
            cmd2 = new SqlCommand("update dbo.Stock set ItemCount= ItemCount-1 where ItemName=@iname", con);
            cmd3 = new SqlCommand("update dbo.Resident set Balance=Balance-@price where StudentId=@id", con);
            con.Open();
            Int32 id = Int32.Parse(txt_id.Text);
            float price = float.Parse(txt_price.Text);
            cmd.Parameters.AddWithValue("@id", txt_id.Text);
            cmd.Parameters.AddWithValue("@iname", txt_iname.Text);
            cmd.Parameters.AddWithValue("@price", txt_price.Text);
            cmd2.Parameters.AddWithValue("@iname", txt_iname.Text);
            cmd3.Parameters.AddWithValue("@id", id);
            cmd3.Parameters.AddWithValue("@price", price);
            cmd.ExecuteNonQuery();
            cmd2.ExecuteNonQuery();
            cmd3.ExecuteNonQuery();
            con.Close();
```

- ((Inner Join of Café and Stock when ItemName is same in both the tables) Inner join of resident when café and resident have same student id).

```
ter("select c.* from dbo.Cafe as c inner join dbo.Stock as s on c.ItemName=s.ItemName inner join dbo.resident as r on c.StudentId=r.StudentId", con)
```

- Code written to list Student ID in the guest log table.

```
public GuestLog()
{
    InitializeComponent();
    con.Open();
    SqlCommand cmd = new SqlCommand("select StudentId from dbo.Resident", con);
    SqlDataReader DR = cmd.ExecuteReader();
    while (DR.Read())
    {
        txt_id.Items.Add(DR[0]);
    }
    txt_gtimein.Value = DateTime.Now ;
    txt_gtimeout.Value = DateTime.Now;
    con.Close();
    DisplayData();
}
```

Normalization

1 NF

Equipment ID	Student ID	Item Name	Date_Out	Date_In	Damaged

Item Name can have multiple values like Cart, Board Games, Pool Supplies, Vacuum Cleaners. This violates 1 NF.

Example:

Equipment ID	**Student ID**	**Item Name**	**Date_Out**	**Date_In**	**Damaged**
1	16349876	Board Game, Vacuum Cleaner, Pool Supplies	2017/12/10	2017/14/10	No

Solution:

Equipment ID	**Student ID**	**Item Name**	**Date_Out**	**Date_In**	**Damaged**
1	16349876	Board Game	2017/12/10	2017/14/10	No
1	16349876	Vacuum Cleaner	2017/12/10	2017/14/10	No
1	16349876	Pool Supplies	2017/12/10	2017/14/10	No

2 NF

Equipment ID	**Student ID**	**Item Name**	**Date_Out**	**Date_In**	**Damaged**	**Item Count**	**Price**

Solution:

Item Name	Item Count	Price

Deficiencies

- When a resident checks out an equipment the date-in value will be the default value, i.e. current date-time.
- When the resident checks-in the equipment back, the record must be updated to the actual date-in value.
- Similarly, when a guest checks-in, the date-out value will be default value: current date-time. This must be updated when the guest checks-out of the residence hall.
- This is because C# does not allow to insert null values in the date time picker in Windows Form Application.

Future Scope

Null value should be allowed in equipment date-in and guest check-out fields.

Reference

https://dev.mysql.com/doc/refman/5.7/en/tutorial.html

https://www.youtube.com/watch?v=yPu6qV5byu4

https://www.tutorialspoint.com/csharp/

https://www.youtube.com/watch?v=lisiwUZJXqQ