



RESIDENTIAL LIFE

CS: 470 - Introduction to Database Management System



Group Members:

Aishwarya Iyer

Gayathree Iyer

Divya Neelima Putta

Sreekanth Sai Bhamidipati

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.

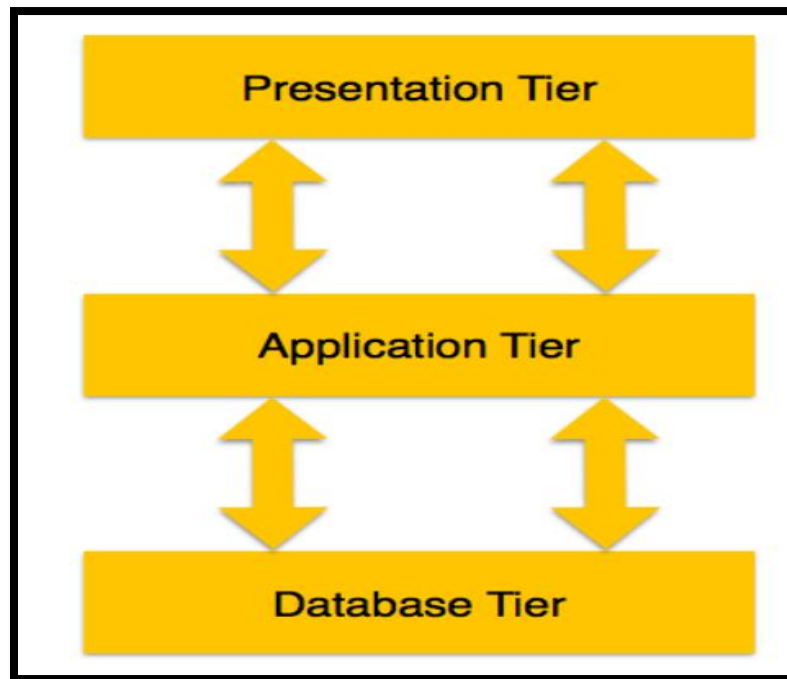
Application Tier: The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing. This layer is responsible for solving mission critical business problems.

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

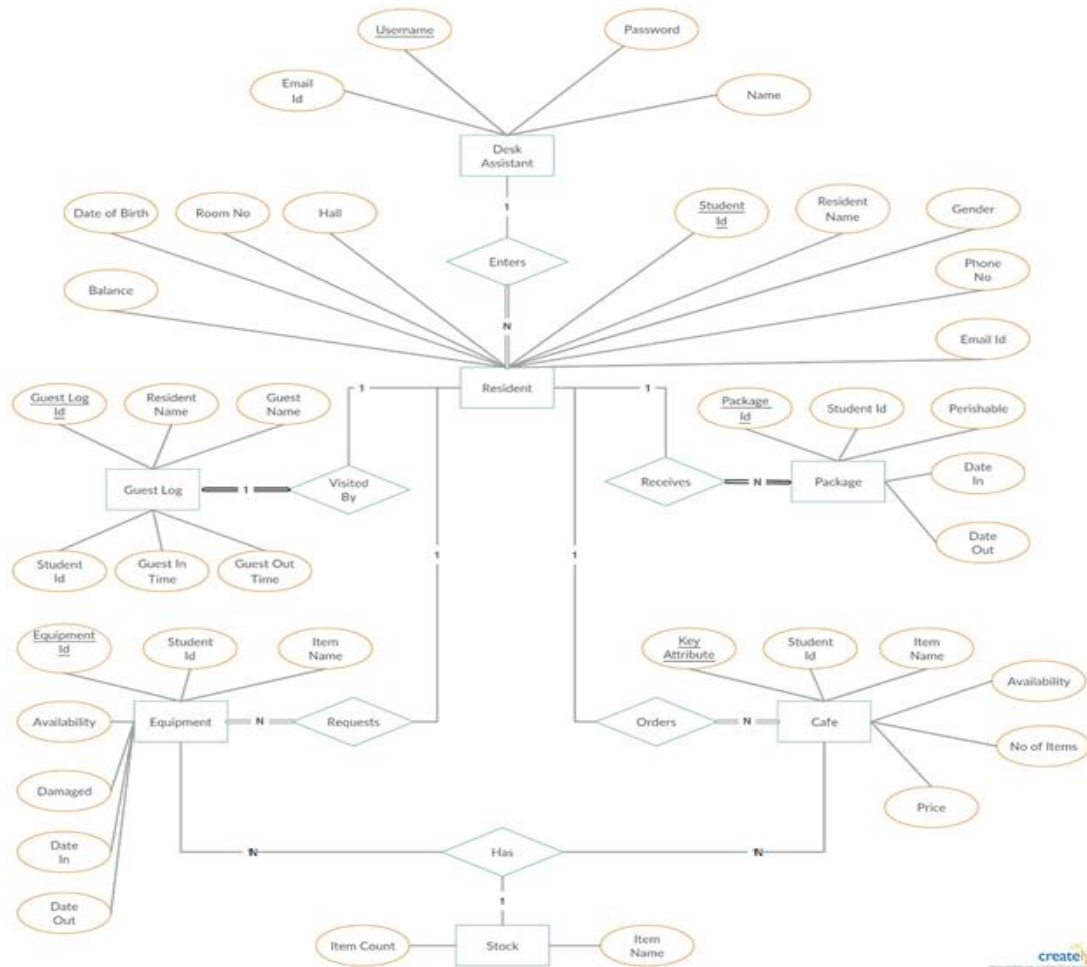
- Database: MySQL
- Language used in Business Logic Layer: C#
- Web page design: Windows Form Application

Database Architecture

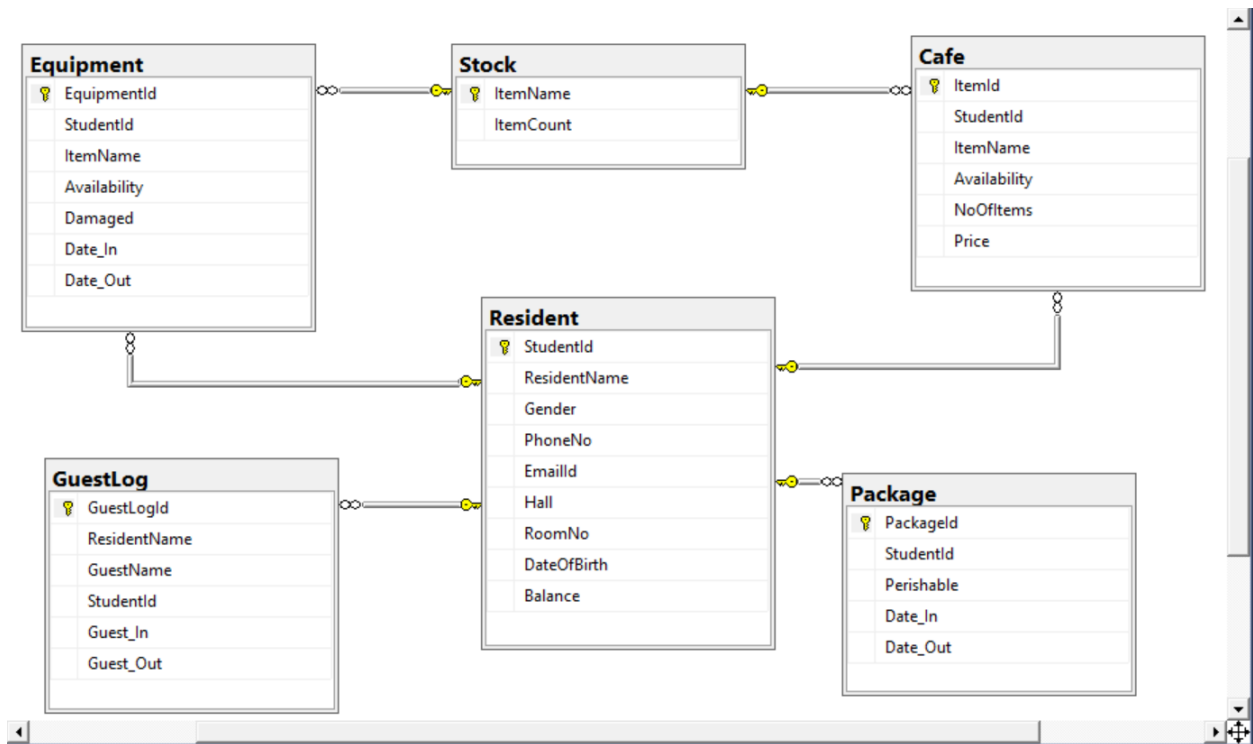


1. Presentation Tier – It has the Web Interface and end users work on this tier where they don't know anything about existence of database beyond this layer. (AngularJS, HTML5, CSS3)
2. Application Tier – It will contain the SQL Queries, and code to retrieve desired data from database and present it 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



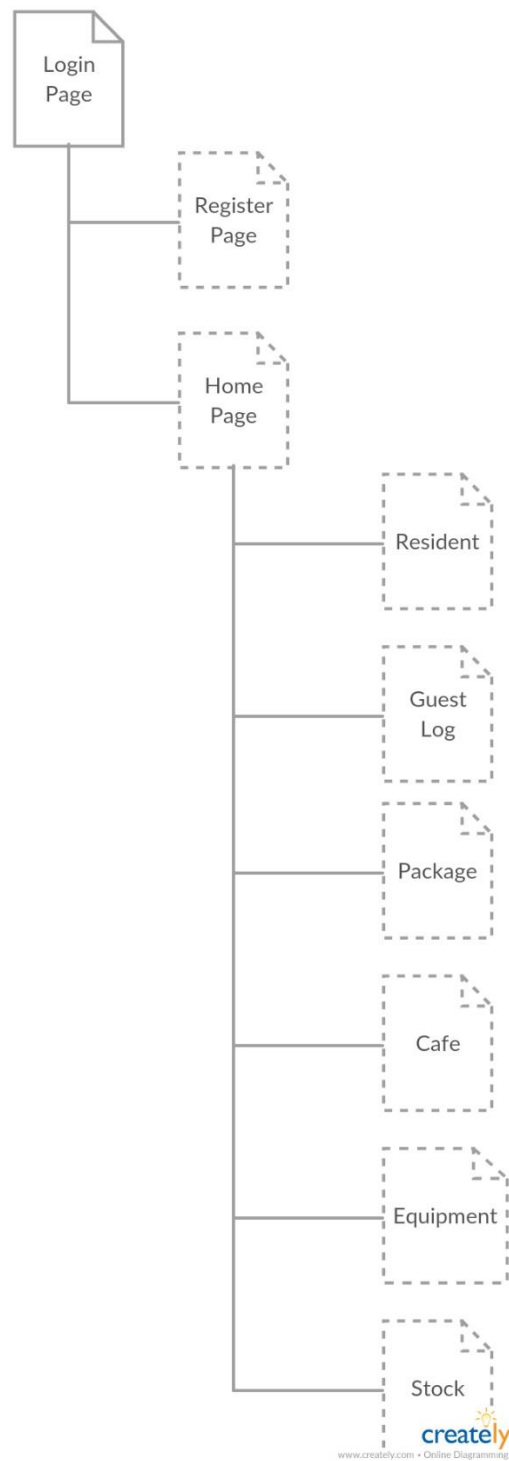
Data Schema



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	Packageld	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



Core Queries

```
-- 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)
);
```

```

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)
);

```



```

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:

```

//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:

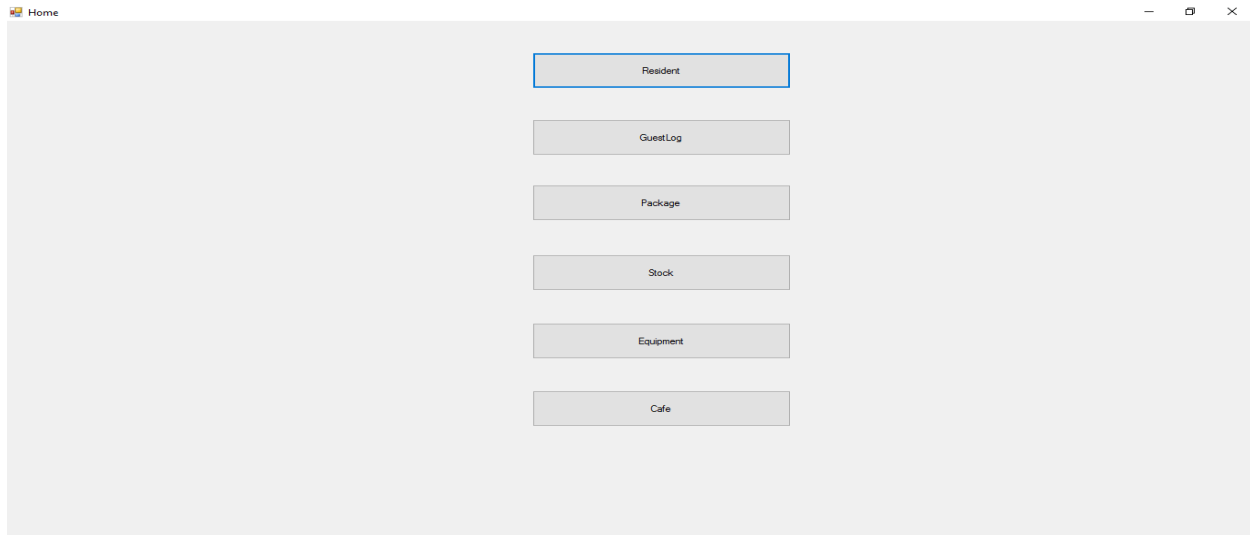
```
//Update Record
private void btn_Update_Click(object sender, EventArgs e)
{
    if (txt_Name.Text != "" &&
        txt_balance.Text != "" &&
        txt_gender.Text != "" &&
        txt_hall.Text != "" &&
        txt_room.Text != "" &&
        txt_balance.Text != "" &&
        txt_email.Text != "" &&
        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.

The screenshot shows a web application window titled 'Resident'. The form contains the following fields:

- Name:
- StudentID:
- PhoneNo:
- Hall:
- Email:
- RoomNo:
- Dob: (with a calendar icon)
- Balance:
- Gender:

There are three buttons at the bottom of the form: 'Insert', 'Update', and 'Delete'. A 'back' button is located to the right of the StudentID field.

Below the form is a table with the following data:

	StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
▶	12277893	Aishwarya 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	Denik 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.com	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 StudentID

PhoneNo Hall

Email RoomNo

Dob Balance

Gender

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
12277893	Aishwarya 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	Oak Place	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.com	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

PhoneNo Hall

Email RoomNo

Dob Balance

Gender

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishwarya 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	Oak Place	234	3/11/1992	500
16778898	Gayathree Iyer	F	8167856743	giyer@gmail.com	Oak Street	123	1/21/1995	500

RESIDENT (UPDATE)

Updating room number of John Canady from 125 to 126

Resident

Name: StudentID:

PhoneNo: Hall:

Email: RoomNo:

Dob: Balance:

Gender:

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishwarya 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.com	Oak Street	123	1/21/1995	500

Resident

Name: StudentID:

PhoneNo: Hall:

Email: RoomNo:

Dob: Balance:

Gender:

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	500
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	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.com	Oak Street	123	1/21/1995	500

Record Updated Successfully

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

Resident

Name StudentID

PhoneNo Hall

Email RoomNo

Dob Balance

Gender

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	500
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	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.com	Oak Street	123	1/21/1995	500

RESIDENT (DELETE)

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

Resident

Name StudentID

PhoneNo Hall

Email RoomNo

Dob Balance

Gender

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
1673452	Emily Jones	F	8167534456	emily@gmail.com	Johnson	342	6/12/1995	500
12277893	Aishwarya Iyer	F	8167456678	ai@gmail.com	Johnson	345	1/12/1995	500
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	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.com	Oak Street	123	1/21/1995	500

Record Deleted Successfully!

OK

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.

GuestLog

StudentId

GuestName

GuestTimeIn

GuestTimeOut

	GuestLogId	GuestName	StudentId	Guest_In	Guest_Out
▶	3	Quinton John	19554427	11/22/2017 4:24...	11/26/2017 4:24...
	4	Andrew Ong	16554432	11/26/2017 4:31...	11/26/2017 4:31...
	5	Caroline Park	16778763	11/25/2017 4:33...	11/26/2017 4:33...
	6	Terry Harbarger	15673454	11/20/2017 4:33...	11/26/2017 4:33...
	7	Chen Yun	15673454	11/23/2017 4:34...	11/26/2017 4:34...
*					

GUEST LOG (INSERT)

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

List of Student ID from Resident table.

GuestLog

StudentId

GuestName

GuestTimeIn

GuestTimeOut

	GuestLogId	GuestName	StudentId	Guest_In	Guest_Out
▶	3	Quinton John	19554427	11/22/2017 4:24...	11/26/2017 4:24...
	4	Andrew Ong	16554432	11/26/2017 4:31...	11/26/2017 4:31...
	5	Caroline Park	16778763	11/25/2017 4:33...	11/26/2017 4:33...
	6	Terry Harbarger	15673454	11/20/2017 4:33...	11/26/2017 4:33...
	7	Chen Yun	15673454	11/23/2017 4:34...	11/26/2017 4:34...
*					

12277893
18347702
16547687
19554427
16334562
16778896
16778896
15673454
16223345
16225564
17338893
16554432
16778763
16444432

Type here to search

Desktop 4:43 PM 11/26/2017

Guest-Sean Grube inserted for Resident with ID: 16547687

GuestLog

StudentId: 16547687

GuestName: Sean Grube

GuestTimeIn: 2017-11-21 16:42

GuestTimeOut: 2017-11-21 16:42

Back

Insert Update Delete

	GuestLogId	GuestName	StudentId	Guest_In	Guest_Out
▶	3	Quinton John	19554427	11/22/2017 4:24...	11/26/2017 4:24...
	4	Andrew Ong	16554432	11/26/2017 4:3...	
	5	Caroline Park	16778763	11/25/2017 4:33...	
	6	Terry Harbarger	15673454	11/20/2017 4:33...	
	7	Chen Yun	15673454	11/23/2017 4:34...	
*					

Record Inserted Successfully

OK

Guest-Sean Grube inserted successfully

Guest Time-out by default to current time

GuestLog

StudentId:

GuestName:

GuestTimeIn: 2017-11-26 16:45

GuestTimeOut: 2017-11-26 16:45

Back

Insert Update Delete

	GuestLogId	GuestName	StudentId	Guest_In	Guest_Out
▶	3	Quinton John	19554427	11/22/2017 4:24...	11/26/2017 4:24...
	4	Andrew Ong	16554432	11/26/2017 4:31...	11/26/2017 4:31...
	5	Caroline Park	16778763	11/25/2017 4:33...	11/26/2017 4:33...
	6	Terry Harbarger	15673454	11/20/2017 4:33...	11/26/2017 4:33...
	7	Chen Yun	15673454	11/23/2017 4:34...	11/26/2017 4:34...
	8	Sean Grube	16547687	11/21/2017 4:42...	11/21/2017 4:42...

GUEST LOG (UPDATE)

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

GuestLog

StudentId: 16547687

GuestName: Sean Grube

GuestTimeIn: 2017-11-21 16:42

GuestTimeOut: 2017-11-26 12:00

Back

Insert Update Delete

	GuestLogId	GuestName	StudentId	Guest_In	Guest_Out
	4	Andrew Ong	16554432	11/26/2017 4:31...	11/26/2017 4:31...
	5	Caroline Park	16778763	11/25/2017 4:33...	11/26/2017 4:33...
	6	Terry Harbarger	15673454	11/20/2017 4:33...	11/26/2017 4:33...
	7	Chen Yun	15673454	11/23/2017 4:34...	11/26/2017 4:34...
▶	8	Sean Grube	16547687	11/21/2017 4:42...	11/26/2017 12:00...
*					

Record Updated Successfully

OK

Guest time out updated successfully

GuestLog

StudentId:

GuestName:

GuestTimeIn: 2017-11-26 16:47

GuestTimeOut: 2017-11-26 16:47

Back

Insert Update Delete

	GuestLogId	GuestName	StudentId	Guest_In	Guest_Out
▶	3	Quinton John	19554427	11/22/2017 4:24...	11/26/2017 4:24...
	4	Andrew Ong	16554432	11/26/2017 4:31...	11/26/2017 4:31...
	5	Caroline Park	16778763	11/25/2017 4:33...	11/26/2017 4:33...
	6	Terry Harbarger	15673454	11/20/2017 4:33...	11/26/2017 4:33...
	7	Chen Yun	15673454	11/23/2017 4:34...	11/26/2017 4:34...
	8	Sean Grube	16547687	11/21/2017 4:42...	11/26/2017 12:00...

PACKAGE

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

Package

StudentId

Perishable

PackageDateIn

PackageDateOut

	PackageId	StudentId	Perishable	Date_In	Date_Out
▶	2	12277893	Not Perishable	11/20/2017	11/21/2017
	3	18347702	Perishable	11/21/2017	11/21/2017
	4	16547687	Not Perishable	11/20/2017	11/23/2017
	5	19554427	Not Perishable	11/24/2017	11/24/2017
	6	15673454	Perishable	11/25/2017	11/26/2017
*					

List of Student ID from Resident table.

Package

StudentId

Perishable

PackageDateIn

PackageDateOut

	PackageId	StudentId	Perishable	Date_In	Date_Out
▶	2	12277893	Not Perishable	11/20/2017	11/21/2017
	3	18347702	Perishable	11/21/2017	11/21/2017
	4	16547687	Not Perishable	11/20/2017	11/23/2017
	5	19554427	Not Perishable	11/24/2017	11/24/2017
	6	15673454	Perishable	11/25/2017	11/26/2017
*					

PACKAGE (INSERT)

Inserting a record: Package for Student ID: 16225564

Package

StudentId: 16225564

Perishable: Not Perishable

PackageDateIn: 2017-11-26

PackageDateOut: 2017-11-26

Insert Update Delete

	Packageld	StudentId	Perishable	Date_In	Date_Out
▶	2	12277893	Not Perishable	11/20/2017	11/21/2017
	3	18347702	Perishable	11/21/2017	
	4	16547687	Not Perishable	11/20/2017	
	5	19554427	Not Perishable	11/24/2017	
	6	15673454	Perishable	11/25/2017	
*					

Record Inserted Successfully

OK

Insertion successful

Package

StudentId:

Perishable:

PackageDateIn: 2017-11-26

PackageDateOut: 2017-11-26

Insert Update Delete

	Packageld	StudentId	Perishable	Date_In	Date_Out
▶	2	12277893	Not Perishable	11/20/2017	11/21/2017
	3	18347702	Perishable	11/21/2017	11/21/2017
	4	16547687	Not Perishable	11/20/2017	11/23/2017
	5	19554427	Not Perishable	11/24/2017	11/24/2017
	6	15673454	Perishable	11/25/2017	11/26/2017
	7	16225564	Not Perishable	11/26/2017	11/26/2017

PACKAGE (UPDATE)

Updating guest check out time

Package

StudentId: 12277893

Perishable: Not Perishable

PackageDateIn: 2017-11-20

PackageDateOut: 2017-11-26

Buttons: Insert, Update, Delete

Back

	Packageld	StudentId	Perishable	Date_In	Date_Out
▶	2	12277893	Not Perishable	11/20/2017	11/21/2017
	3	18347702	Perishable	11/21/2017	11/20/2017
	4	16547687	Not Perishable	11/20/2017	11/24/2017
	5	19554427	Not Perishable	11/24/2017	11/25/2017
	6	15673454	Perishable	11/25/2017	11/26/2017
	7	16225564	Not Perishable	11/26/2017	

Record Updated Successfully

OK

Guest check out time updated successfully

Package

StudentId:

Perishable:

PackageDateIn: 2017-11-26

PackageDateOut: 2017-11-26

Buttons: Insert, Update, Delete

Back

	Packageld	StudentId	Perishable	Date_In	Date_Out
▶	2	12277893	Not Perishable	11/20/2017	11/26/2017
	3	18347702	Perishable	11/21/2017	11/21/2017
	4	16547687	Not Perishable	11/20/2017	11/23/2017
	5	19554427	Not Perishable	11/24/2017	11/24/2017
	6	15673454	Perishable	11/25/2017	11/26/2017
	7	16225564	Not Perishable	11/26/2017	11/26/2017

STOCK

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

Stock

ItemName

ItemCount

Price

Back

Insert Update Delete

	ItemName	ItemCount	Price
▶	BoardGames	10	0
	Cart	8	0
	Chocolates	50	1
	Cookies	20	2.5
	Pizza	9	5.09
	VacuumCleaner	5	0

List of Stock list from equipment and cafe

Stock

ItemName

ItemCount

Price

Back

Insert Update Delete

	ItemName	ItemCount	Price
▶	BoardGames	10	0
	Cart	8	0
	Chocolates	50	1
	Cookies	20	2.5
	Pizza	9	5.09
	VacuumCleaner	5	0

Cart
BoardGames
KitchenItems
VacuumCleaner
PoolSupplies
Sandwich
Pizza
Drinks
Chocolates
Cookies

STOCK (INSERT)

Inserting stock: Pool supplies

Stock

ItemName: PoolSupplies

ItemCount: 4

Price: 0

Back

Insert Update Delete

	ItemName	ItemCount	Price
▶	BoardGames	10	0
	Cart	8	0
	Chocolates	50	1
	Cookies	20	2.5
	Pizza	9	5.09
	VaccumCleaner	5	0

Record Inserted Successfully

OK

Pool supplies inserted

Stock

ItemName:

ItemCount:

Price:

Back

Insert Update Delete

	ItemName	ItemCount	Price
▶	BoardGames	10	0
	Cart	8	0
	Chocolates	50	1
	Cookies	20	2.5
	Pizza	9	5.09
	PoolSupplies	4	0
	VaccumCleaner	5	0

STOCK (UPDATE)

Stock

ItemName: BoardGames

ItemCount: 15

Price: 0

Back

Insert Update Delete

ItemName	ItemCount	Price
BoardGames	10	0
Cart	8	0
Chocolates	50	1
Cookies	20	2.5
Pizza	9	5.09
PoolSupplies	4	0
VacuumCleaner	5	0

Record Updated Successfully

OK

Updating item count from 10 to 15

Stock

ItemName:

ItemCount:

Price:

Back


Insert Update Delete

ItemName	ItemCount	Price
BoardGames	15	0
Cart	8	0
Chocolates	50	1
Cookies	20	2.5
Pizza	9	5.09
PoolSupplies	4	0
VacuumCleaner	5	0

Item count updated successfully to 15

EQUIPMENT

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

 Equipment

StudentId	<input type="text"/>	EquipmentOut	<input type="text" value="2017-11-26 17:45"/>	<input type="button" value="Back"/>
ItemName	<input type="text"/>	EquipmentIn	<input type="text" value="2017-11-26 17:45"/>	
Availability	<input type="text"/>	Damaged	<input type="text"/>	

	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

Equipment

StudentId: 16444432
ItemName: VacuumCleaner
Availability: 5
EquipmentOut: 2017-11-22 17:45
EquipmentIn: 2017-11-24 17:45
Damaged: No

Back

Insert Update Delete

	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

Record Inserted Successfully

OK

Record inserted successfully

Equipment

StudentId:
ItemName:
Availability:
EquipmentOut: 2017-11-26 17:47
EquipmentIn: 2017-11-26 17:47
Damaged:

Back

Insert Update Delete

	EquipmentId	StudentId	ItemName	Date_Out	Date_In	Damaged
	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
	8	16444432	VaccumCleaner	11/22/2017 5:45...	11/24/2017 5:45...	No

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.

 Cafe

StudentId

ItemName

Availability

Price

	ItemId	StudentId	ItemName
▶	2	12277893	Cookies
	3	16547687	Chocolates
	4	16778898	Pizza
	5	16778763	Cookies
*			

- 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.

Cafe

StudentId: 16444432

ItemName: Pizza

Availability: 8

Price: 5.09

Back

Insert Delete

	ItemId	StudentId	ItemName
▶	2	12277893	Cookies
	3	16547687	Chocolates
	4	16778898	Pizza
	5	16778763	Cookies
*			

Record Inserted Successfully

OK

Record inserted successfully

Cafe

StudentId:

ItemName:

Availability:

Price:

Back

Insert Delete

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

ITEM PRICE DEDUCTED FROM BALANCE

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

Cafe

StudentId

ItemName

Availability

Price

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

Resident

Name StudentID

PhoneNo Hall

Email RoomNo

Dob Balance

Gender

StudentId	ResidentName	Gender	PhoneNo	EmailId	Hall	RoomNo	DateOfBirth	Balance
12277893	Aishwarya 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	Shrikant 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.com	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.

Equipment

StudentId: 16778896
ItemName: BoardGames
Availability: 11

EquipmentOut: 2017-11-20 18:04
EquipmentIn: 2017-11-22 15:04
Damaged: No

Back

Insert Update Delete

	EquipmentId	StudentId	ItemName	Date_Out	Date_In	Damaged
	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
	8	16444432	VaccumCleaner	11/22/2017 5:45...	11/24/2017 5:45...	No

Record Inserted Successfully

OK

Equipment

StudentId: 16554432
ItemName: BoardGames
Availability: 10

EquipmentOut: 2017-11-26 18:07
EquipmentIn: 2017-11-26 18:07
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
	11	16778896	BoardGames	11/20/2017 6:04...	11/22/2017 3:04...	No

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

```

33     }
34     DR2.Close();
35     con.Close();
36     DisplayData();
37 }
38 private void btn_Insert_Click(object sender, EventArgs e)
39 {
40     if (txt_iname.Text != "" &&
41         txt_id.Text != "" &&
42         txt_price.Text != "")
43     {
44         cmd = new SqlCommand("insert into dbo.Cafe(StudentId,ItemName) values (@id,@iname)", con);
45         cmd2 = new SqlCommand("update dbo.Stock set ItemCount= ItemCount-1 where ItemName=@iname", con);
46         cmd3 = new SqlCommand("update dbo.Resident set Balance=Balance-@price where StudentId=@id", con);
47         con.Open();
48         Int32 id = Int32.Parse(txt_id.Text);
49         float price = float.Parse(txt_price.Text);
50         cmd.Parameters.AddWithValue("@id", txt_id.Text);
51         cmd.Parameters.AddWithValue("@iname", txt_iname.Text);
52         cmd.Parameters.AddWithValue("@price", txt_price.Text);
53         cmd2.Parameters.AddWithValue("@iname", txt_iname.Text);
54         cmd3.Parameters.AddWithValue("@id", id);
55         cmd3.Parameters.AddWithValue("@price", price);
56         cmd.ExecuteNonQuery();
57         cmd2.ExecuteNonQuery();
58         cmd3.ExecuteNonQuery();
59         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:

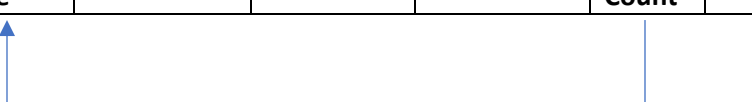
<u>Equipment ID</u>	Student ID	Item Name	Date_Out	Date_In	Damaged
1	16349876	Board Game, Vacuum Cleaner, Pool Supplies	2017/12/10	2017/14/10	No

Solution:

<u>Equipment ID</u>	Student ID	<u>Item Name</u>	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

<u>Equipment ID</u>	Student ID	Item Name	Date_Out	Date_In	Damaged	Item Count	Price



Solution:

<u>Item Name</u>	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>