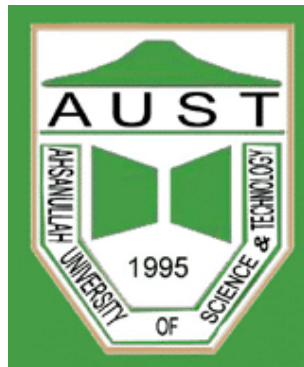


B.Sc Engineering Thesis Paper
On
**“Designing and Interfacing a Hospital-Based
Database System”**
(A Case Study of BIRDEM)



Department of Computer Science & Engineering
Ahsanullah University of Science & Technology
Dhaka, Bangladesh.

A thesis paper submitted in partial fulfillment of the requirements
for the Degree of B.Sc Engineering (Computer Science &
Engineering)

Date: - July 12, 2010

Session: - Spring '10

**“Designing and Interfacing a Hospital-Based
Database System”
(A Case Study of BIRDEM)**

Submitted By:-

- | | |
|---------------------|--------------|
| 1. Syed Mahboob Nur | 06.02.04.013 |
| 2. Jahid Hasan | 06.02.04.036 |
| 3. Kazi Sumaiya | 06.02.04.042 |
| 4. Tasfia Rahman | 06.02.04.044 |

In Partial Fulfillment for the Degree of
B.Sc Engineering in Computer Science & Engineering
Ahsanullah University of Science & Technology.

Certification

We hereby, proclaim that the thesis on "Designing and Interfacing a Hospital-Based Database System (A Case Study of BIRDEM)" was conducted under the supervision of Ms. Rosina Surovi Khan.

We also declare that neither this nor any part thereof has been submitted elsewhere for the award of any degree.

Approved By:

Dr. S. M. Abdullah Al-Mamun

Professor & Head of the Department

Dept of C.S.E , AUST

Submitted By:

Syed Mahboob Nur

Jahid Hasan

Supervised By:

Ms. Rosina Surovi Khan

Assistant Professor

Dept of C.S.E, AUST

Kazi Sumaiya Mona

Tasfia Rahman

CONTENTS AT A GLANCE

PREFACE

ABSTRACT

1. Introduction

2. Designing the Database System

3. Interfacing the Database System using
.NET framework

4. Conclusion and Future work

Acknowledgement

Starting by the name of Almighty Allah.....

Authors would like to express their sincere and hearty gratitude and profound indebtedness to their respectful teacher Ms Rosina Surovi Khan, Assistant Professor, AUST, for her constant timely and appropriate guidance, helpful advice, invaluable assistance and endless patience throughout the progress of their work, without which the work could not have been completed.

Authors also acknowledge with hearty thanks to all the members of the BIRDEM hospital for their important information and cooperation.

Finally, authors acknowledge all cooperation of their friends, who helped them through giving their important time, their knowledge and their best advice.

Special thanks to our parents and elders for their help and support.

Table of Contents

PREFACE	ix
ABSTRACT	x
1. INTRODUCTION.....	1
2. DESIGNING THE DATABASE SYSTEM	4
2.1 Determining Entities and Attributes.....	4
2.2 Entity Relationship Diagram	7
2.3 Relational Model	9
2.3.1 Relational Tables' Descriptions	13
2.3.2 Explanation of Relational Model	26
2.4 Relational Database Design	34
2.4.1 Functional Dependency	34
2.4.2 Normalization	35

2.5 Implementation in SQL Server	49
2.5.1 Creation of Tables and Insertion of data ...	51
2.5.2 Sample Data Values of Tables	53
2.6 Complex Queries	60
 3. INTERFACING THE DATABASE SYSTEM USING .NET FRAMEWORK	 64
3.1 Research on Interface Design Guidelines	64
3.2 FRONT END Design	74
3.2.1 Forms' Design	75
3.2.2 Relating Interface Design Guidelines to Front End Design	93
3.3 Security feature of FRONT END	96
3.4 Implementation of Insert, Delete, Update buttons & Search Option	107
3.5 Usage of DLL file	113

4. CONCLUSION & FUTURE WORK	115
4.1 Conclusion	115
4.2 Future Work	115
4.2.1 Gridline View Features	116
4.2.2 Trigger Features	129
REFERENCES.....	133
APPENDIX	135

PREFACE

Our thesis is about Designing and Interfacing a Hospital-Based Database System. It forms a basic entity of the management of a Hospital. Hence, it is very important for the system to be reliable, user friendly, and should be properly functional for a long time without cropping up of any errors.

To start with the system study we visited Bangladesh Institute of Research and Rehabilitation for Diabetes, Endocrine and Metabolic Disorders (BIRDEM). We saw their system, studied it and tried to develop a better system. **Our system is an automated system for Hospital Management.** This gave us the idea of the different fields that ought to be in a Hospital Management System such as patient registration, his/her advance payment, the records, the details etc. and also how a software system can make the work easy both for the hospital staff and the patients. Moreover, the evaluation helped us to arrive at the conclusion that the automated software is far more superior to the manual ones.

ABSTRACT

Our motive is to develop a software that is very much user friendly and easy to gather information in a very short time. We try to make our software reliable and comfortable.

As our thesis paper is on Designing and Interfacing a Hospital Management System (A Case Study of BIRDEM) we divide our work into two basic parts Designing part and Interfacing Part.

® We give a flow chart on our work division in THESIS OVERVIEW part.

Chapter 1 → Introduction

In this chapter we discuss the definition of Database and its usefulness. We also describe the reason to take HOSPITAL MANAGEMENT SYSTEM as our thesis work.

Chapter 2 → Designing the Database System

In this chapter we describe the entities and attributes. We draw the Entity Relationship Diagram (ERD) and Tables. We determine the attributes of tables and its data types. We also find functional dependencies and normalize all the tables. Then we implement our database in SQL Server and finally we execute some complex queries on the system.

Chapter 3 → Interfacing the Database System using .Net Framework.

We made a research on Interface Design Guidelines and designed our front end in C#. We applied some of the guidelines in our front end.

We control our software security using C#. We Insert Delete, Update and Search data from the database in our software. We used a DLL file so that we

can easily access to any Operating System and we don't need to load our database.

Chapter 4 → Conclusion and Future Work.

We tried to Save, Delete and Update data using Data Grid view and we also tried to use Trigger in SQL Server but we cannot complete them. So we include it as a part of future work.

CHAPTER 1

INTRODUCTION

❖ What is a Database?

- A Database is a collection of records which are stored on a computer; a database organizes the data according to database models such as a relational model. [1]

❖ Why do we need Databases?

- Databases collect items on which the user can carry out various operations such as viewing, navigating, creating tables, and searching. Databases can be seen as a symbolic form of the computer age. [2]

We use databases for these reasons. Such as,

1. We use database because we can easily manipulate, edit or delete data.
2. Data are kept organized in a database so we can easily retrieve data.
3. Easy to find out desired data.
4. Data are secured.

❖ Advantages of Database

- ✓ Reduced Data Redundancy.
- ✓ Reduced updating errors and increased consistency.
- ✓ Greater data integrity and independence from applications programs.

- ✓ Improved data access to users through use of host and query languages.
- ✓ Improved data security.
- ✓ Reduced data entry, storage, and retrieval costs.
- ✓ Facilitated development of new application programs. [3]

In our thesis *Designing and Interfacing a Hospital-Based Database System (A case study of BIRDEM)* we can see two basic parts.

- ✓ **Designing &**
- ✓ **Interfacing**

Our Thesis Teacher Ms. Rosina Surovi Khan decided that we have to complete the design part in semester 4/1 and interfacing part in semester 4/2. In the introductory class of the thesis our respected madam suggested to select a specific database system to work on.

Choosing Hospital Management System for our thesis

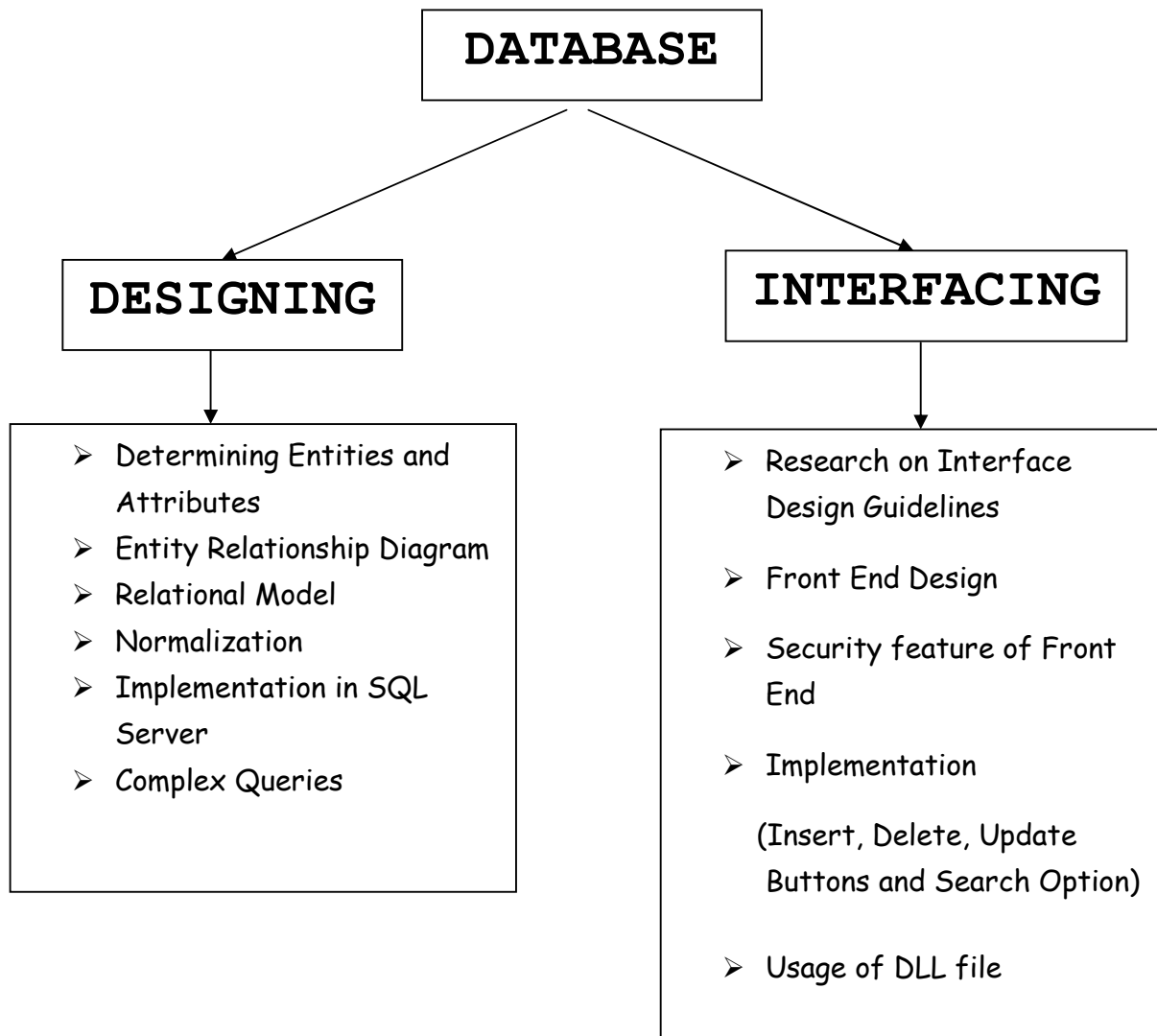
We study and select three systems at first. The systems were

- Banking System
- Computer Sales Management System
- Hospital Management System

We saw the demos of the respective systems from different sources and all the group members decided to do the thesis on Hospital Management System (A Case Study of BIRDEM) because the system is less complex and easy to study. Most Banking Systems and Computer Sales Management Systems are controlled using online based software where users can access from any part of the country. **But we are determined to make desktop based software.** So we decided

to choose Hospital Management System based on a Case Study of BIRDEM. We try our best to make the system efficient and user friendly with the help of our database and front end software.

Thesis Overview



DESIGNING THE DATABASE SYSTEM

2.1 Determining Entities and Attributes

❖ Entity

- ✓ An *entity* is something that has a distinct, separate existence, though it need not be a material existence. In particular, abstractions and legal fictions are usually regarded as entities. In general, there is also no presumption that an entity is animate. Entities are used in system developmental models that display communications and internal processing of, say, documents compared to order processing.
- ✓ An entity could be viewed as a set containing subsets.
- ✓ A DBMS *entity* is either a thing in the modeled world or a drawing element in an Entity Relationship Diagram(ERD) .[4]

❖ Attribute

- ✓ An *attribute* is a specification that defines a property of an object, element, or file. It may also refer to or set the specific value for a given instance of such.
- ✓ Attributes should more correctly be considered metadata. It is frequently and generally a property of an entity.
- ✓ An attribute of an object usually consists of a name and a value; of an element, a type or class name; of a file, a name and extension.[5]

❖ Data Type

- ✓ A *data type* (or *datatype*): In programming, a classification identifying one of various types of data, as floating-point, integer, or Boolean, stating the possible values for that type, the operations that can be done on that type, and the way the values of that type are stored.[6]

We think our best and determine the entities and attributes for our Database System. The Entities and Attributes are given below.

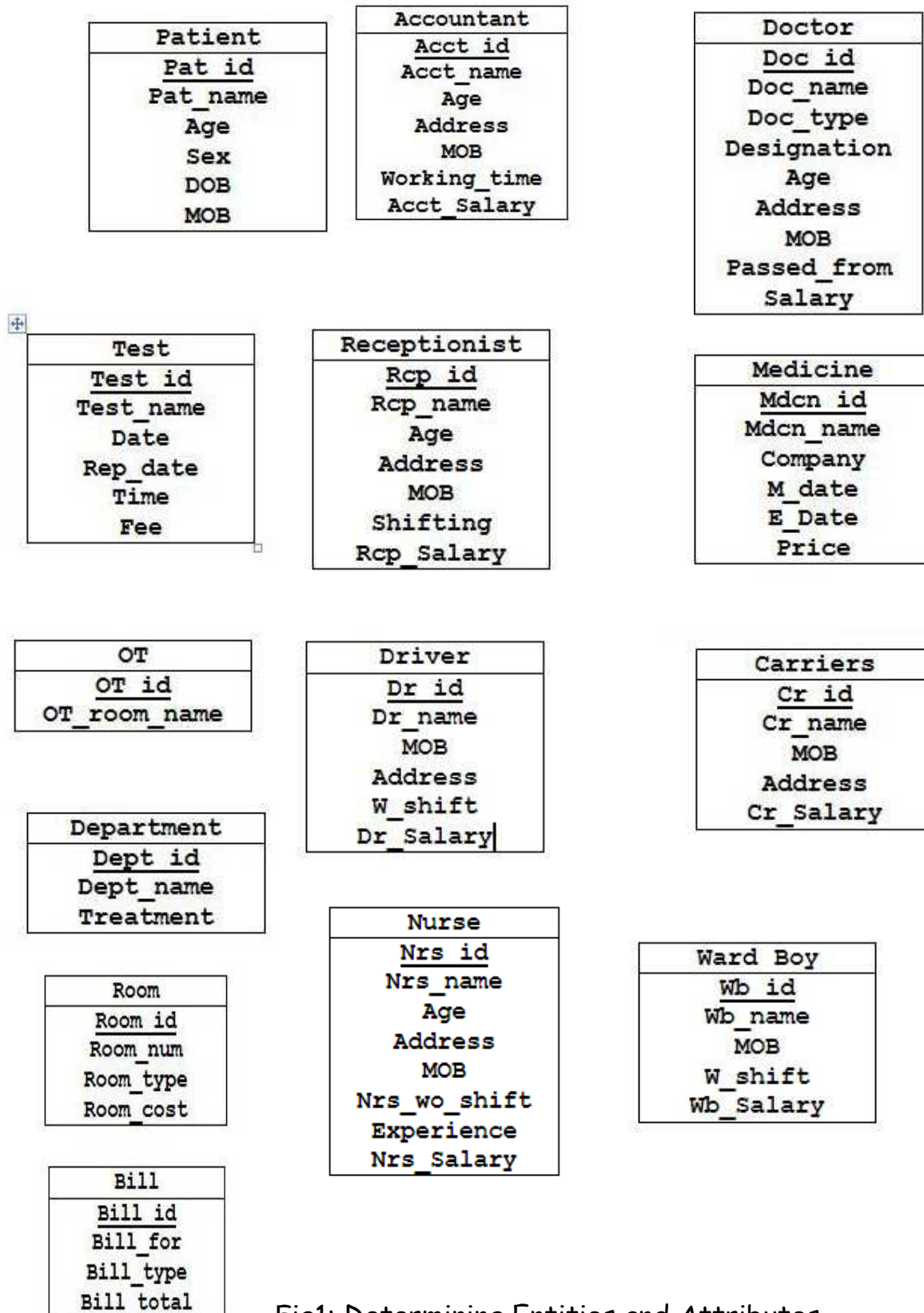


Fig1: Determining Entities and Attributes.

2.2 Entity Relationship Diagram (ERD):

We draw the Entity Relationship Diagram (ERD) very carefully and efficiently for the whole system of BIRDEM.

We were able to cover all probable information of BIRDEM in our ERD.

The ERD is given below:

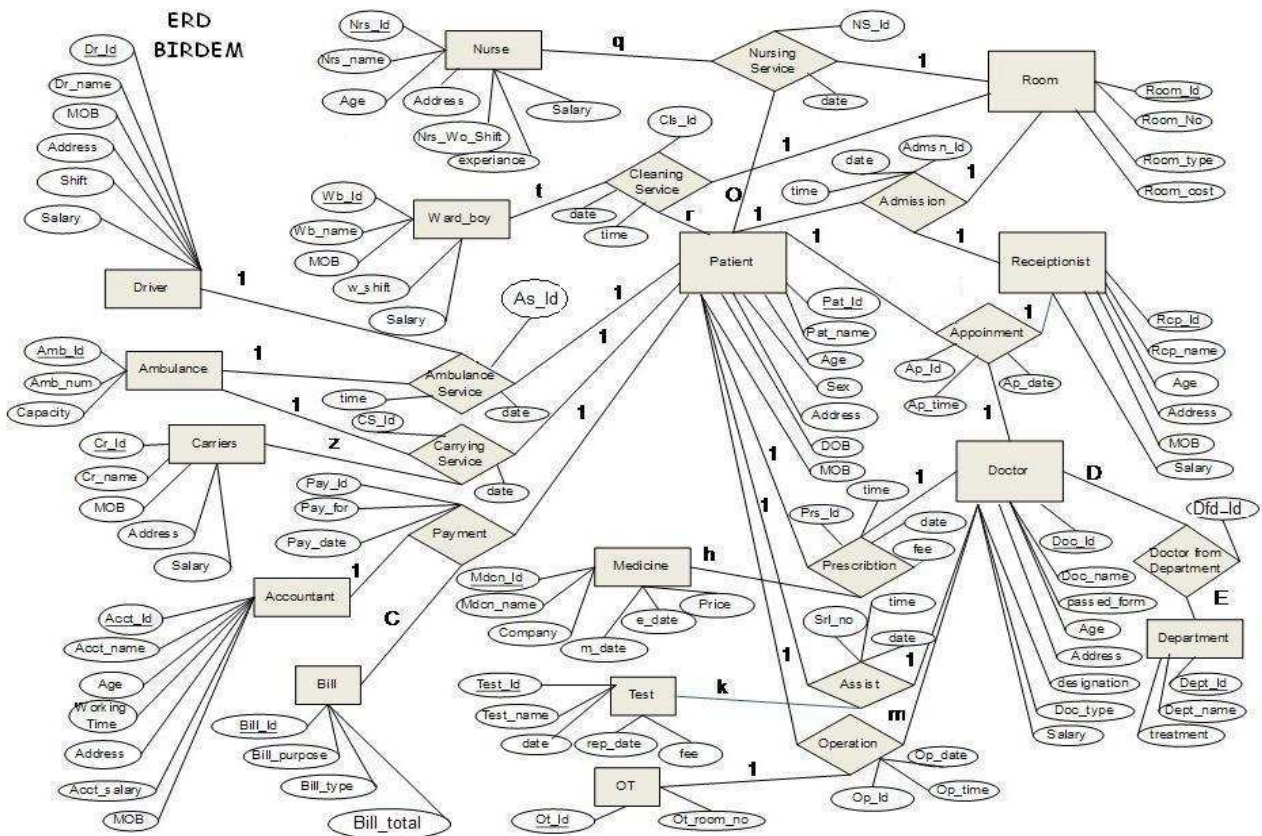


Fig2: Entity Relationship Diagram (ERD).

2.3 Relational Model:

After completing the ERD successfully we made the relational model (table schemas) taking into account all the entities and the relationships.

Patient Table:-

<u>Pat_id</u>	Pat_name	Age	Sex	Address	DOB	MOB

Room Table:-

<u>Room_id</u>	Room_No	Room_type	Room_cost

Receptionist Table:-

<u>Rcp_id</u>	Rcp_name	Age	Address	MOB	shifting	salary

Admission Table:-

This is a junction table between Patient, Receptionist & Room tables.

<u>Admsn_id</u>	Pat_id	Room_id	Rcp_id	date	time

Doctor Table:-

<u>Doc_id</u>	Doc_name	Doc_type	Designation	Age	Address	MOB	Passed_from	Salary

Appointment Table:-

This is a junction table between Patient, Receptionist & Doctor tables.

<u>Ap_id</u>	Pat_id	Doc_id	Rcp_id	apnmt_date	apnmt_time

Bill Table:-

<u>Bill_id</u>	Bill_for	Bill_type	Bill_total

Accountant Table:-

<u>Acct_id</u>	Acct_name	Age	Address	MOB	Working_time	Acct_salary

Payment Table:-

This is a junction table between Patient, Bill & Accountant Tables.

<u>Pay_id</u>	Bill_for	Pat_id	Acct_id	Pay_type	Pay_date

Medicine Table:-

<u>Mdcn_id</u>	Mdcn_name	Company	m_date	e_date	price

Prescription Table: -

This is a junction table between Patient, Doctor & Medicine tables.

<u>Prs_id</u>	Doc_id	Mdcn_id	Pat_id	date	Fee

Test Table:-

<u>Test_id</u>	Test_name	date	rep_date	fee

Assist Table:-

This is a junction table between Patient, Doctor & Test tables.

<u>Serial_no</u>	Pat_id	Doc_id	Test_id	time	date

OT Table:-

<u>Ot_id</u>	Ot_room_no

Operation Table:-

This is a junction table between Patient, Doctor & OT tables.

<u>Op_id</u>	Doc_id	Pat_id	Ot_id	Op_date	Op_time

Department Table:-

<u>Dept_id</u>	Dept_name	treatment

Doctor_from_Department Table:-

This is a junction table between Doctor & Department tables.

<u>Dfd_id</u>	<u>Doc_id</u>	<u>Dept_id</u>

Nurse Table:-

<u>Nrs_id</u>	Nrs_name	Age	Address	MoB	Nrs_wo_shift	experience	Salary

Nursing_Service Table:-

This is a junction table between Patient, Room & Nurse tables.

<u>Ns_id</u>	Pat_id	Nrs_id	<u>Room_id</u>

Ward Boy Table:-

<u>Wb_id</u>	wb_name	MoB	w_shift	Salary

Cleaning Service Table:-

This is a junction table between Patient, Room & Ward Boy tables.

<u>Cls_id</u>	Pat_id	Wb_id	<u>Room_id</u>

Driver Table:-

<u>Dr_id</u>	Dr_name	Mob	Address	Shift	Salary

Ambulance Table:-

<u>Amb_id</u>	Amb_num	Capacity

Ambulance Service Table:-

This is a junction table between Patient, Driver & Ambulance tables.

<u>As_id</u>	Pat_id	Dr_id	<u>Amb_id</u>

Carriers Table:-

<u>Cr_id</u>	Cr_name	MOB	Address	Salary

Carrying Service Table:-

This is a junction table between Patient, Ambulance & Carriers tables.

<u>CS_id</u>	<u>Cr_id</u>	<u>Amb_id</u>	<u>Pat_id</u>

2.3.1 Relational Tables' Descriptions

Patient table

Attributes	Data type	Comments
Pat_id	int	Unique id for a Patient
Pat_name	varchar(20)	Patient's Name
Age	int	Patient's Age
Sex	varchar(20)	Patient is Male or Female
Address	varchar(20)	Patient's Address
Dob	varchar(20)	Date of Birth
Mob	int	Mobile Number

Room table

Attributes	Data type	Comments
Room_id	int	Unique id for a Room
Room_no	varchar(20)	Room number
Room_type	varchar(20)	Room is VIP or Normal
Room_cost	int	Cost of the Room

Receptionist table

Attributes	Data type	Comments
Rcp_id	int	Unique id for a Receptionist
Rcp_name	varchar(20)	Receptionist's name
Age	int	Receptionist's age
Address	varchar(20)	Receptionist's Address
MOB	int	Mobile Number
Shifting	varchar(20)	Receptionist working shift
Salary	int	Salary a Receptionist gets

Admission table

Attributes	Data type	Comments
Admsn_id	int	Unique id for an Admission
Pat_id	int	Unique id for a Patient
Room_id	int	Unique id for a Room
Rcp_id	int	Unique id for a Receptionist
Date	varchar(20)	Date of Admission

Doctor table:

Attributes	Data type	Comments
Doc_id	int	Unique id for a Doctor
Doc_name	varchar(20)	Doctor's name
Doc_type	varchar(20)	Doctor's specialty
Age	int	Doctor's age
Address	varchar(20)	Doctor's address
Mob	int	Mobile Number
Designation	varchar(20)	Doctor's designation
Passed_from	varchar(20)	Doctor is passed from which medical college
Salary	int	Salary of a doctor

Appointment table

Attributes	Data type	Comments
Apnmt_id	int	Unique id for an Appointment
Pat_id	int	Unique id for a Patient
Doc_id	int	Unique id for a Doctor
Rcp_id	int	Unique id for a Receptionist
Apnmt_date	varchar(20)	Date of an Appointment

Bill table

Attributes	Data type	Comments
Bill_id	int	Unique id for a Bill
Bill_for	varchar(20)	Purpose of the bill
Bill_type	varchar(20)	Bill either in Cash or Check
Bill_total	int	Total amount

Accountant table

Attributes	Data type	Comments
Acct_id	int	Unique id for an Accountant
Acct_name	varchar(20)	Accountant's Name
Age	int	Accountant's age
Address	varchar(20)	Accountant's Address
Mob	int	Mobile Number
Acct_salary	int	Salary of an Accountant

Payment table

Attributes	Data type	Comments
Pay_id	int	Unique id for a Payment
Bill_id	int	Unique id for a Bill
Pat_id	int	Unique id for a Patient
Acct_id	int	Unique id for an Accountant
Pay_type	varchar(20)	Payment in Cash or Check
Pay_date	varchar(20)	Date of Payment

Medicine table

Attributes	Data type	Comments
Mdcn_id	int	Unique id for a Medicine
Mdcn_name	varchar(20)	Medicine's Name
company	varchar(20)	Medicine's Company
M_date	varchar(20)	Manufacture Date
E_date	varchar(20)	Expire Date
price	int	Price of the Medicine

Prescription table

Attributes	Data type	Comments
Prs_id	int	Unique id for a Prescription
Doc_id	int	Unique id for a Doctor
Mdcn_id	int	Unique id for a Medicine
Pat_id	int	Unique id for a Patient
Date	varchar(20)	Date of the Prescription
Time	varchar(20)	Time of the Prescription
Fee	varchar(20)	Prescription Fees

Test table

Attributes	Data type	Comments
Test_id	int	Unique id for a Test
Test_name	varchar(20)	Name of the Test
Date	varchar(20)	Date of Test
Rep_date	varchar(20)	Date of the Report
Fee	int	Test Fees

Assist table

Attributes	Data type	Comments
Serial_no	int	Unique id for an Assisted Test directed to a Patient by a Doctor
Pat_id	int	Unique id for a Patient
Doc_id	int	Unique id for a Doctor
Test_id	int	Unique id for a Test
Date	varchar(20)	Date of the Assisted Test
Time	varchar(20)	Time of the Assisted Test

OT table

Attributes	Data type	Comments
Ot_id	int	Unique id for an Operation Theater (OT)
Ot_room_no	varchar(20)	OT Room Number

Operation table

Attributes	Data type	Comments
Op_id	int	Unique id for an Operation
Doc_id	int	Unique id for a Doctor
Pat_id	int	Unique id for a Patient
Ot_id	int	Unique id for an OT
Op_date	varchar(20)	Date of the Operation
Op_time	varchar(20)	Time of the Operation

Department table

Attributes	Data type	Comments
Dept_id	int	Unique id for a Department
Dept_name	varchar(20)	Department's name
treatment	varchar(20)	Treatments of a patient conducted in a Department

Doctor_from_Department table

Attributes	Data type	Comments
Dfd_id	int	Unique id for a DoctorsfromDepartment junction table
Doc_id	int	Unique id for a Doctor
Dept_id	int	Unique id for a Department

Nurse table

Attributes	Data type	Comments
Nrs_id	int	Unique id for a Nurse
Nrs_name	varchar(20)	Nurse's Name
Age	int	Nurse's age
Address	varchar(20)	Nurse's Address
Mob	int	Mobile Number
Nrs_wo_shift	varchar(20)	Nurse working Shift example morning,day,evening,night
Experience	varchar(20)	Nurse's Experience
salary	int	Salary of a Nurse

Nursing_Service table

Attributes	Data type	Comments
Ns_id	int	Unique id for a Nursing Service
Pat_id	int	Unique id for a Patient
Nrs_id	int	Unique id for a Nurse
Room_id	int	Unique id for a Room
Date	varchar(20)	Date of Nursing Service
Time	varchar(20)	Time of Nursing Service

Ward_boy table

Attributes	Data type	Comments
Wb_id	int	Unique id for a Ward Boy
Wb_name	varchar(20)	Ward Boy's Name
Mob	int	Mobile Number
W_shift	varchar(20)	Working shift of a Ward Boy
salary	int	Salary of a Ward boy

Cleaning_Service table

Attributes	Data type	Comments
Cls_id	int	Unique id for a Cleaning Service
Pat_id	int	Unique id for a Patient
Wb_id	int	Unique id for a Ward Boy
Room_id	int	Unique id for a Room
Date	varchar(20)	Date of Cleaning Service
Time	varchar(20)	Time of Cleaning Service

Driver table

Attributes	Data type	Comments
Dr_id	int	Unique id for a Driver
Dr_name	varchar(20)	Driver's Name
mob	int	Mobile Number
address	varchar(20)	Driver's Address
Shift	varchar(20)	Working shift of a Driver
salary	int	Salary of a Driver

Ambulance table

Attributes	Data type	Comments
Amb_id	int	Unique id for an Ambulance
Amb_num	varchar(20)	Ambulance's Number
Capacity	int	Capacity of an Ambulance

Ambulance_Service table

Attributes	Data type	Comments
As_id	int	Unique id for an Ambulance Service
Pat_id	int	Unique id for a Patient
Dr_id	int	Unique id for a Driver
Amb_id	int	Unique id for an Ambulance
Date	varchar(20)	Date of the Ambulance Service
Time	varchar(20)	Time of the Ambulance Service

Carriers table

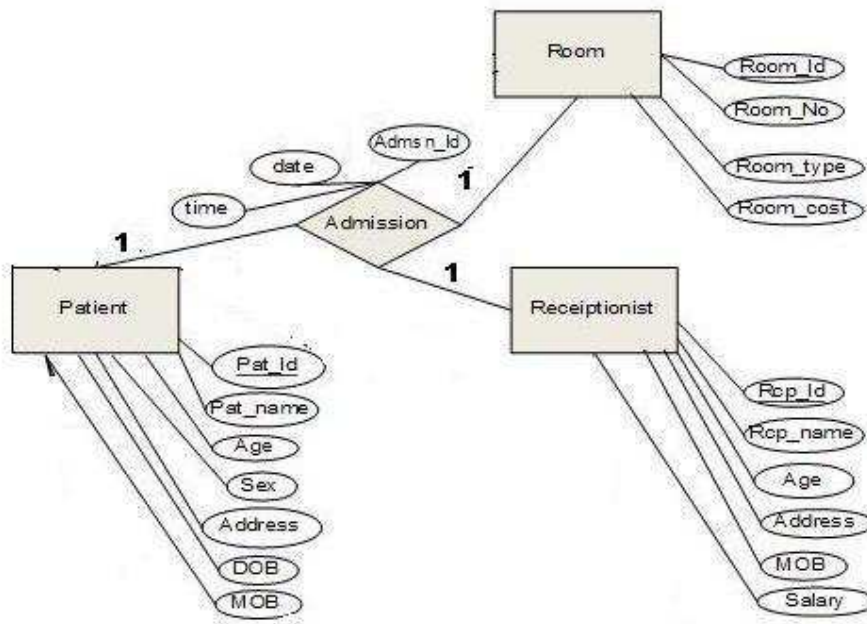
Attributes	Data type	Comments
Cr_id	int	Unique id for a Carrier who will carry patients inside the hospital's premises from the ambulance.
Cr_name	varchar(20)	Carrier's Name
Mob	int	Mobile Number
Address	varchar(20)	Carrier's Address
Salary	int	Salary of a Carrier

Carrying_Service table

Attributes	Data type	Comments
Cs_id	int	Unique id for a Carrying Service
Cr_id	int	Unique id for a Carrier
Amb_id	int	Unique id for an Ambulance
Pat_id	int	Unique id for a Patient
Date	varchar(20)	Date of the Carrying Service
Time	varchar(20)	Time of the Carrying Service

2.3.2 Explanation of Relational Model

Relationship between Receptionist, Patient and Room Entities in the ER Model:



- 1 Receptionist can admit 1 Patient in 1 Room in a certain date and time.
- 1 Receptionist can admit in 1 Room 1 Patient in a certain date and time.
- In 1 Room, 1 Patient is admitted by 1 Receptionist in a certain date and time.

So the relationship is a Ternary Relationship named *Admission* (in the diamond) with cardinality ratio from Patient to Receptionist to Room as 1 to 1 to 1.

Relational model for Receptionist, Patient and Room Entities:

Receptionist, Patient and Room Entities become Receptionist, Patient and Room tables.

Patient Table:-

<u>Pat_id</u>	Pat_name	Age	Sex	DOB	MOB	Address

Room Table:-

<u>Room_id</u>	Room_No	Room_type	Room_cost

Receptionist Table:-

<u>Rcp_id</u>	Rcp_name	Age	Address	MOB	shifting	salary

The junction *Admission* also becomes a table.

Admission Table:-

<u>admsn_id</u>	Pat_id	Room_id	Rcp_id	Date	time

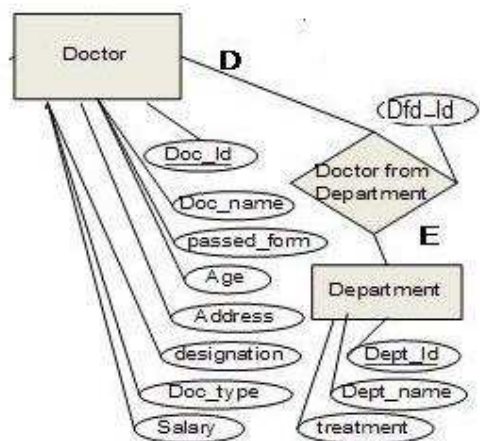
- Primary Key of the Patient Table goes to Admission Table as Foreign Key.
- Primary Key of the Room Table goes to Admission Table as Foreign Key.
- Primary Key of the Receptionist Table goes to Admission Table as Foreign Key.

Since the Cardinality Ratio from Patient to Receptionist to Room is 1 to 1 to 1,

admsn_id is a Primary key in the Admission Table. Pat_id from Patient Table, Room_id from Room Table and Rcp_id from Receptionist Table become Foreign Keys in the Admission Table.

In a similar way, as cardinality ratio for Receptionist_Patient_Doctor relationship is 1 to 1 to 1, Receptionist, Patient and Doctor entities become separate tables along with a junction Appointment table which has Rcp_id, Pat_id and Doc_id as foreign keys. Similar logic applies to Patient_Ambulance_Driver relationship with cardinality ratio 1 to 1 to 1.

Relationship between Doctor and Department Entities in the ER Model:



- 1 Doctor can be from 1 or Many Departments.
- 1 Department may have 1 or Many Doctors.

So it is a Many to Many relationship named *Doctor from Department* (in the diamond).

Relational model for Doctor and Department Entities:

Doctor and Department Entities become Doctor and Department tables.

Doctor Table:-

<u>Doc_id</u>	Doc_name	Doc_type	Designation	Age	Address	MOB	Passed_from	Salary

Department Table:-

<u>Dept_id</u>	Dept_name	treatment

The junction table *Doctor from Department* also becomes a table.

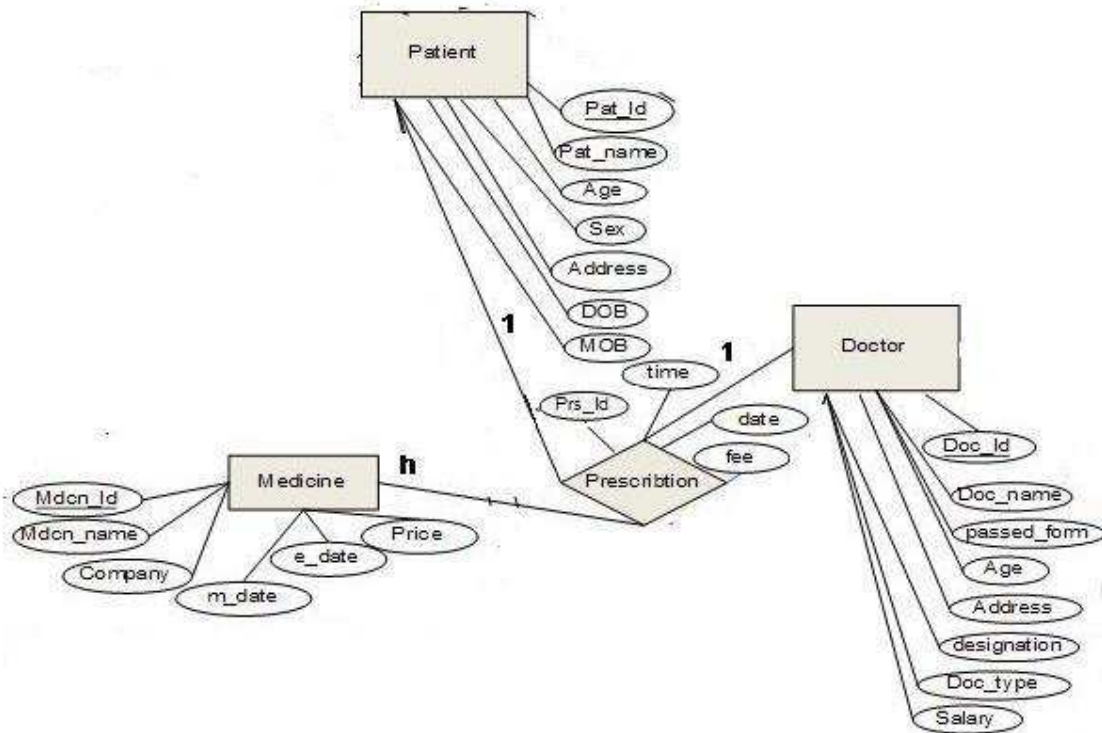
Doctor_from_Department Table:-

<u>Dfd_id</u>	<u>Doc_id</u>	<u>Dept_id</u>

- Primary Key of the Doctor Table goes to Doctor_from_Department Table as part of Primary Key.
- Primary Key of the Department Table goes to Doctor_from_Department Table as part of Primary Key.

Since the Cardinality Ratio from Doctor to Department is Many to Many, Dfd_id is a part of Primary key in the Doctor_from_Department Table. Doc_id from Doctor Table and Dept_id from Department Table become parts of Primary Key in the Doctor_from_Department Table.

Relationship between Patient, Doctor and Medicine Entities in the ER Model:



- 1 Doctor gives 1 patient 1 or more medicine.
- 1 patient takes 1 medicine prescribed by 1 doctor.
- 1 medicine is prescribed by 1 doctor to 1 patient.

So the relationship is a Ternary Relationship named *Prescription* (in the diamond) with a Cardinality Ratio from Patient to Doctor to Medicine 1 to 1 to Many.

Relational model for Patient, Doctor and Medicine Entities:

Patient, Doctor and Medicine Entities become Patient, Doctor and Medicine tables.

Patient Table:-

<u>Pat_Id</u>	Pat_name	Age	Sex	DOB	MOB	Address

Doctor Table:-

<u>Doc_id</u>	Doc_name	Doc_type	Designation	Age	Address	MOB	Passed_from	Salary

Medicine Table:-

<u>Mdcn_id</u>	Mdcn_name	company	m_date	e_date	price

Prescription Table: -

This is a junction table between Patients, Doctor & Medicine Table.

<u>Prs_id</u>	<u>Doc_id</u>	<u>Mdcn_id</u>	<u>Pat_id</u>	date	fee

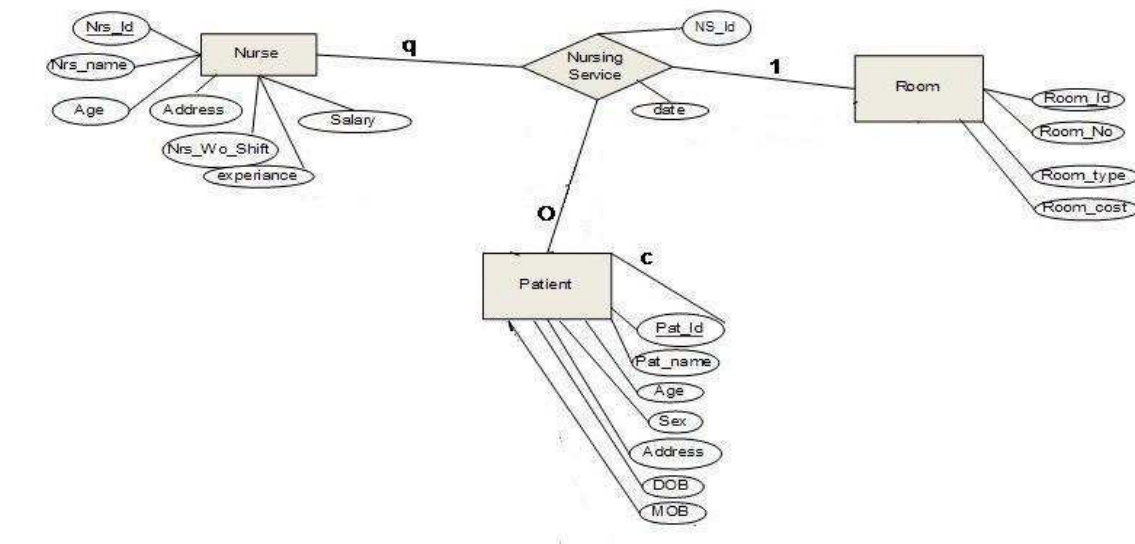
- Primary Key of the Patient Table goes to Prescription Table as Foreign Key.
- Primary Key of the Doctor Table goes to Prescription Table as Foreign Key.
- Primary Key of the Medicine Table goes to Prescription Table as part of Primary Key.

Since the Cardinality Ratio from Patient to Doctor to Medicine 1 to 1 to M, Prs_id is a Primary key in the Prescription Table. Pat_id from Patient Table,

Doc_id from Doctor Table and **Mdcn_id** from Medicine Table become Foreign Keys in the Admission Table.

In a similar way relational tables have been designed for Patient-Doctor-Test, Patient-OT-Doctor, Patient-Bill-Accountant relationships with cardinality ratio 1 to 1 to M. Similar logic applies for Patient-Ambulance-Carrier relationship with cardinality ratio 1 to 1 to M.

#Relationship Between Patient,Room & Nurse Entities in the ER Model :-



- 1 room is fixed for 1 Patient to provide nursing service for 1 or Many nurses in a certain date.
- 1 patient receives nursing service from 1 Nurse in 1 Room in a certain date.
- 1 nurse can render proper services in 1 room to many patients in a certain date.

So it is a Ternary Relationship named Nursing Services (in the diamond) with cardinality Ratio from Room to Nurse to Patient 1 to M to M.

#Relational model between Patient, Nurse and Room Entities:-

Patient Table:-

<u>Pat_id</u>	Pat_name	Age	Sex	DOB	MOB	Address

Room Table:-

<u>Room_id</u>	Room_No	Room_type	Room_cost

Nurse Table:-

<u>Nrs_id</u>	Nrs_name	Age	Address	Mob	Nrs_wo_shift	experience	Salary

Nursing Service Table:-

This is a junction table between Patient, Room and Nurse Table.

<u>Ns_id</u>	<u>Pat_id</u>	<u>Nrs_id</u>	<u>Room_id</u>

- Primary Key of the Patient Table goes to Nursing Service Table as part of Primary Key.
- Primary Key of the Nurse Table goes to Nursing Service Table as part of Primary Key.
- Primary Key of the Room Table goes to Nursing Service Table as Foreign Key.

Since the Cardinality Ratio from Room to Patient to Nurse is 1to M to M. Ns_id is a Primary key in the Nursing Service Table. Pat_id from Patient Table, Nrs_id

from Nurse Table become parts of Primary Key in the Nursing Service Table. Room_id from Room Table becomes Foreign Key in the Nursing Service Table. In a similar way relational tables are created for Patient-Room-Wardboy relationship with cardinality ratio 1 to M to M.

2.4 Relational Database Design

Relational databases are the most commonly used database today. It uses the table to structure information so that it can be readily and easily searched through.

To make a Relational database design we have to be clear about two parts:

1. Functional Dependency
2. Normalization

2.4.1 Functional Dependencies

Definition of functional dependencies:

Given a relational schema $R(A_1, A_2, \dots, A_n)$ and $X, Y \{A_1, \dots, A_n\}$.

Then $X \rightarrow Y$ means that for every extension of R , the following holds:

R contains no two tuples that are equal in all values of X but differ in at least one value of Y .

(Pronunciation: "X determines Y functionally" "Y is functionally dependent of X").

Example:

Student (matNr, name):

$\{\text{matNr}\} \rightarrow \{\text{name}\}$

Definition of *full* functional dependencies:

Prerequisites as in Definition 1.

Y is said to be fully functionally dependent of X, if there is no proper subset $X' \subset X$,

Where $X' \rightarrow Y$.

Notation: $X \Rightarrow Y$.

Example:

A University Database:-

Class (classId, room, day, pName)

$\{classId, room\} \rightarrow \{pName\}$

$\{classId, day, pName\} \rightarrow \{room\}$

$\{classId\} \Rightarrow \{pName\}$

$\{classId\} \Rightarrow \{room\}$ [7]

2.4.2 Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

It has mainly two goals:-

- ✓ First goal: eliminate redundant data

For example, storing the same data in more than one table

- ✓ Second Goal: ensure data dependencies make sense

For example, only storing related data in a table

Benefits of Normalization:

- Less storage space
- Quicker updates
- Less data inconsistency
- Clearer data relationships

- Easier to add data
- Flexible Structure

Bad database designs results in:

- Redundancy: inefficient storage.
- Anomalies: data inconsistency, difficulties in maintenance.[7]

1NF, 2NF, 3NF, BCNF are some of the early forms in the list that address this problem.

First Normal Form (1NF)

Definition:

A relation is in first normal form if it contains only simple, atomic values for attributes, no sets. Example:

Name	Offspring		Place
	Child	Age	
Muller	Christa	12	Stuttgart
	Peter	10	
	Iris	9	
Schmidt	Martin	17	Trier
	Rainer	18	

The value of an attribute can be a relation by itself.

=> Operations in the model are much more complicated

=> In order to keep the model simple: 1NF

Ways to normalize the above relation:

First attempt:

Person (name, place, child1, child2, child3)

=> Not good. Reason: either not enough available columns for some data records (How many children can a person have??) Or, if there are enough columns to provide for all thinkable cases, waste of much space (many NULL values).

Second attempt:

Person:-

<u>pName</u>	place
Muller	Stuggart
Schmidt	Trir

Child:-

<u>pName</u>	<u>chName</u>	age
Muller	Christa	12
Muller	Peter	10
Muller	Iris	9
Schmidt	Martin	17
Schmidt	Rainer	18

Advantage:

This requires just the right amount of space that is actually needed.

Disadvantage:

It requires an additional table. pName is redundantly stored.

Second Normal Form (2NF)

Definitions:

Definition of second normal form (simple version):

A relation is in 2NF, if it is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key of the relation.

Definition of second normal form (extended version):

A relation is in 2NF, if it is in 1NF and every non-candidate-key attribute is fully functionally dependent on every candidate key.

Example:-

A University Database:

TA (matNr, classId, sName, hours, taSalary)

Full functional dependencies:

{matNr, classId} => {hours}

{matNr, classId} => {taSalary}

{matNr} => {sName}

TA (matNr, classId, sName, hours, taSalary)

Student (matNr, sName)

=> TA is not in 2NF

Redundancy since the name is repeated for every occurrence of the same Matrikel Number.

Solution:

Move the dependency $\{\text{matNr}\} \Rightarrow \{\text{name}\}$ to a separate relation.

\Rightarrow Relation "Student"

Third Normal Form (3NF)

Definition:-

A functional dependency $X \rightarrow Y$ in a relation R is called a transitive dependency, if R contains a set of attributes, Z for which holds:

. A chain Exists.

. $X \rightarrow Z \rightarrow Y$

. Y is not a part of primary key

. Z is not a super key and

. $X \rightarrow Z \rightarrow Y$

Y is then called transitively dependent on X via Z.

Definition of Third Normal Form:

A Relation is in 3NF, if it is in 2NF and no non primary key attributes is transitively dependent on the primary key.

Example:-

TA (matNr, classId, hours, taSalary)

Functional dependencies:

$\{\text{matNr}, \text{classId}\} \Rightarrow \{\text{hours}\}$

$\{\text{matNr}, \text{classId}\} \Rightarrow \{\text{taSalary}\}$

Assumption:

$\{\text{hours}\} \Rightarrow \{\text{taSalary}\}$

There is the following transitive dependency:

$\{\text{matNr}, \text{classId}\} \Rightarrow \{\text{hours}\} \Rightarrow \{\text{taSalary}\}$

Since taSalary is not an attribute in a candidate key and hours is not a superkey, TA is not in 3NF.

There is unnecessary redundancy since taSalary is repeated for each occurrence of the same value of hours.

Solution:

Move the dependency $\{\text{hours}\} \Rightarrow \{\text{taSalary}\}$ to a separate relation.

Example:

TANew (matNr, classId, hours) and TASalary (hours, taSalary).

Boyce Coded Normal Form (BCNF)

A relation R is in 3NF relation and for a dependency $X \rightarrow A$ from an attributes set X to an attributes A holds that,

- ✓ X is not a super key
- ✓ In addition, A is a part of a primary key
- ✓ Then this relation is not also in BCNF.

In all other cases, 3NF and BCNF are identical.

BCNF is a little stronger than 3NF. In most cases, relations in 3NF are also in BCNF.

The alternative definition of BCNF shows in comparison to the 3NF definition how the two differ: in BCNF, X must always be a super key; in 3NF it does not need to be a super key if A is part of a candidate key.

- ✓ A relation is in BCNF, if and only if, every determinant is a candidate key.

- ✓ No part of the primary key is Fully Functional Dependent on the non primary key.

Example:-

Relation Speedlimits (town, streetSegment, postcode, speed)

Full functional dependencies:

- {town, streetSegment} => {postCode}
- {town, streetSegment} => {speed}
- {postCode} => {town}
- {postcode, streetSegment} => {speed}

Candidate keys:

- (town, streetSegment)
- (postCode, streetSegment)

Speedlimits is in 3NF:

- 1NF by definition

2NF since all non-primary-key attributes are fully functionally dependent on the primary Key. For the extended definition: speed is the only attribute that is not part of a Candidate key, and it is fully functionally dependent not only on the primary key, but also on the other candidate.

- 3NF since the only non-candidate-key attribute is speed, and the only transitive Dependencies ending in speed would be from one of the keys to the other and then to speed. However, transitive dependencies where the middle set is a candidate key do not violate the definition of 3NF.

But BCNF is violated:

The problematic dependency is from an attribute (postcode) which is not a superkey to a part (town) of the primary key.

<u>town</u>	<u>streetSegment</u>	postcode	speed
Stuttgart	A-Str	70000	30
Stuttgart	B-Str	70000	30
Stuttgart	C-Str	70000	50
Stuttgart	D-Str	71234	70

Redundancy: postCode implies the town => unnecessary repetition

Transforming to BCNF:

1. Attempt:

Speedlimit (town, streetSegment, speed)

Codes (postCode, town)

Schema is now in BCNF.

- The dependency {town, streetSegment} => {postCode} is no longer recognizable.

2. Attempt:

Speedlimit (town, streetSegment, speed)

PostCodes (streetsegment, postCode)

→ BCNF

But:

- The dependency {town, streetSegment} => {postCode} is again not recognizable.
- The decomposition is lossy again!

3. Attempt:

Speedlimit (postCode, streetSegment, speed)

Codes (postCode, town)

Now both relations are in BCNF, and the decomposition is lossless.

However, the dependencies {town, streetSegment} => {postCode} and {town, street-

Segment} => {speed} are lost.

It is possible to show:

- A relation that is not in BCNF can always be losslessly decomposed towards BCNF.
- A lossless decomposition into BCNF that preserves all dependencies does not always exist. [7]

In our thesis we will try Normalize all the relational tables.

FULFILMENT OF NORMAL FORMS:

Room Table:-

<u>Room_id</u>	Room_no	Room_type	Room_cost

$\{Room_id\} \Rightarrow \{Room_no\}$

Functional Dependency Exist

2 different room no's do not correspond to the same Room_id.

$\{Room_id\} \Rightarrow \{Room_type\}$

Functional Dependency Exist

2 different room types' do not correspond to the same Room_id

$\{Room_id\} \Rightarrow \{Room_cost\}$

Functional Dependency Exist

2 different room cost's do not correspond to the same Room_id

Relation : (Room_id, Room_No, Room_type, Room_cost)

Full Functional Dependencies:

$\{Room_id\} \Rightarrow \{Room_no\}$

$\{Room_id\} \Rightarrow \{Room_type\}$

$\{Room_id\} \Rightarrow \{Room_cost\}$

1NF:-

Attributes do not have sub attributes.

So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key.

So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF.

BCNF:-

No part of the primary key is Fully Functional Dependent on the non primary keys. So the relation is in BCNF.

Bill Table:-

<u>Bill_id</u>	Bill_for	Bill_type	Bill_ total

$\{Bill_id\} \Rightarrow \{Bill_for\}$ Functional Dependency Exist.

2 different Bill_for's do not correspond to the same Bill_id.

$\{Bill_id\} \Rightarrow \{Bill_type\}$ Functional Dependency Exist.

2 different Bill_type do not correspond to the same Bill_id.

$\{Bill_id\} \Rightarrow \{Bill_total\}$ Functional Dependency Exist.

2 different Bill total do not correspond to the same Bill_id.

Relation : (Bill_id, Bill_for, Bill total, Bill_type)

Full Functional Dependency:

$\{Bill_id\} \Rightarrow \{Bill_for\}$

$\{Bill_id\} \Rightarrow \{Bill_type\}$

$\{Bill_id\} \Rightarrow \{Bill_total\}$

1NF:-

Attributes do not have sub attributes.

So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key.

So the relation is in 2NF

3NF:-

No chain Exists.

So the relation is in 3NF.

BCNF:-

No part of the primary key is Fully Functional Dependent on the non primary key. So the relation is in BCNF.

In a similar way Bill, Doctor, Accountant, Receptionist, Driver, Ambulance, Carriers, OT, Medicine, Test, Department and Nurse Tables fulfill all the normal forms.

JUNCTION TABLES:

Admission Room Table:-

This is a junction table between Patient, Room, and Receptionist Table

<u>Admsn_id</u>	Room_id	Pat_id	Rcp_id	Date	Time

Full Functional Dependencies:

{admsn_id} => {Room_id} Functional Dependency Exist

{admsn_id} => {Rcp_id} Functional Dependency Exist

{adsn_id} => {Date} Functional Dependency Exist

{admsn_id} => {Time} Functional Dependency Exist

{admsn_id} => {Pat_id} Functional Dependency Exist

1NF:-

Attributes do not have sub attributes.

So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key.

So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF.

BCNF:-

No part of the primary key is Fully Functional Dependent on the non primary keys. So the relation is in BCNF.

In a similar way Ambulance Service and Appointment Tables fulfill all the normal forms.

Prescription Table:-

This is a junction table between Patient, Medicine & Doctor Table.

<u>Prs_id</u>	Doc_id	<u>Mdcn_id</u>	Pat_id	Date	Fees	Time

Full Functional Dependencies:

$\{Prs_id, Mdcn_id\} \Rightarrow \{Doc_id\}$	Functional Dependency Exist
$\{Prs_id, Mdcn_id\} \Rightarrow \{Pat_id\}$	Functional Dependency Exist
$\{Prs_id, Mdcn_id\} \Rightarrow \{Date, Fees, Time\}$	Functional Dependency Exist

Relation: (Prs_id, Mdcn_id, Doc_id, Pat_id, Date, Fees, Time)

$\{Prs_id, Mdcn_id\} \Rightarrow \{Doc_id\}$

$\{Prs_id, Mdcn_id\} \Rightarrow \{Pat_id\}$

$\{Prs_id, Mdcn_id\} \Rightarrow \{Date\}$

$\{Prs_id, Mdcn_id\} \Rightarrow \{Time\}$

$\{Prs_id, Mdcn_id\} \Rightarrow \{Fees\}$

1NF:-

Attributes do not have sub attributes.

So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key.

So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF.

BCNF:-

No part of the primary key is Fully Functional Dependent on the non primary keys. So the relation is in BCNF.

In a similar way Assist, Carrying Service, Cleaning Service, Operation and Nursing Service tables fulfill all normal forms.

VIOLATION OF NORMAL FORM:

Payment Table:-

This is a junction table between Patients, Bill & Accountant tables.

<u>Pay_id</u>	Pat_id	<u>Bill_id</u>	Acct_id	Pay_type	Pay_date

For Payment relation, the following functional dependencies exist:

$\{Pay_id\} \Rightarrow \{Pay_Type, Pay_date, Pat_id\}$

Two different patient ids, payment dates and payment types cannot correspond to the same payment id. So Pay_Type, Pay-date and Pat_id are fully functionally dependent on Pay_id.

$\{Bill_id\} \Rightarrow \{Acct_id, Pat_id\}$

Similarly two different accountant ids and patient ids cannot correspond to the same bill id. So Acct_Id and Pat_id are fully functionally dependent on Bill_id.

Based on the above functional dependencies:

The relation is in 1NF.

The relation is not in 2NF because all non-primary keys are not fully functionally dependent on the primary key (Pay_id, Bill_id). So we split the relation to make it 2NF.

Payment1 (Pay_id, Pay_Type, Pay_date, Pat_id)

Payment2 (Bill_id, Acct_id, Pat_id)

The relations are now in 2NF.

3NF:

There is no chain.

So the relations are in 3NF.

BCNF:

No Part of the primary key (Pay_Id, Bill_Id) is fully functionally dependent on any non primary key. So the relations are in BCNF.

2.5 Implementation in SQL Server:

After Normalization, we implemented our Database in SQL Server. There were 27 tables and each of them was connected accurately in the SQL Server's Entity Relationship Diagram. Then we entered the data in the corresponding database tables.

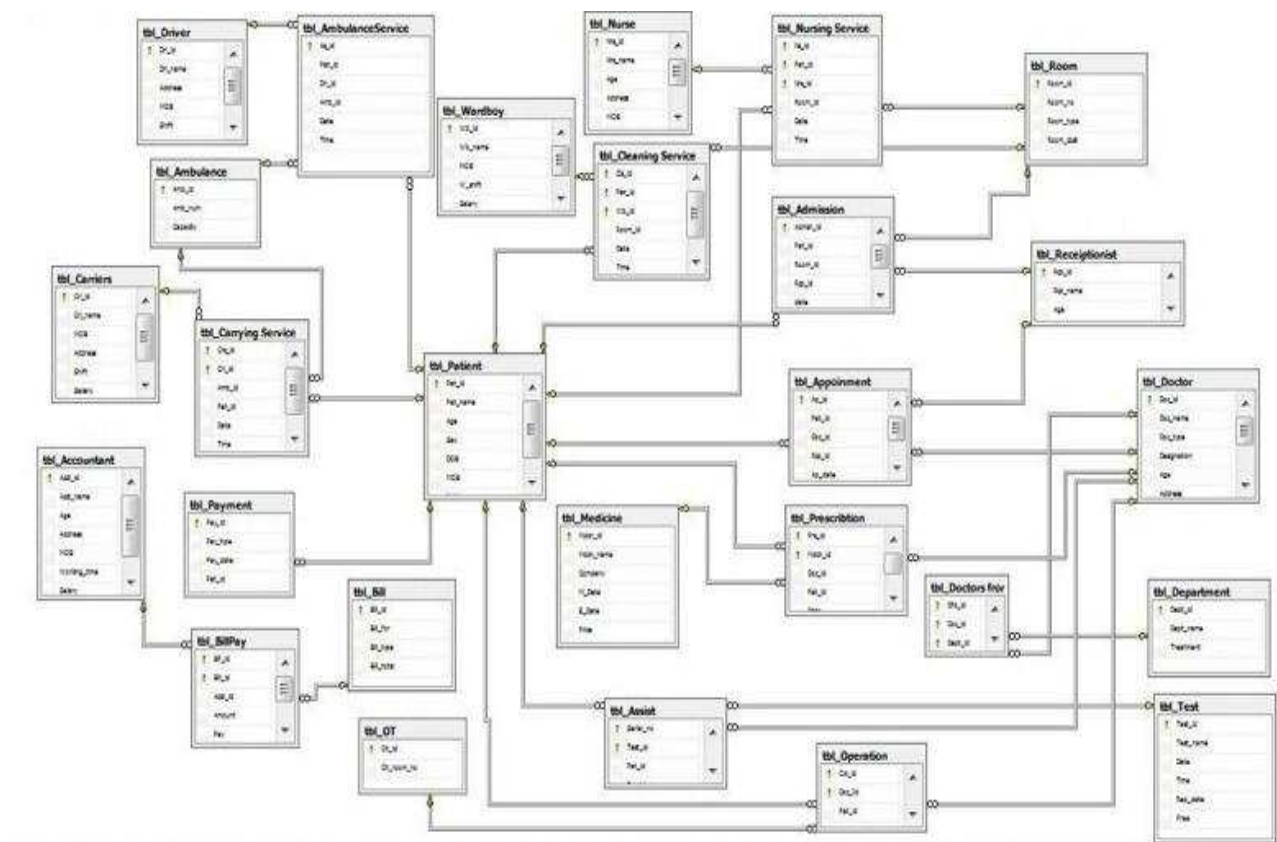


Fig: Relational model Implementation on SQL Server.

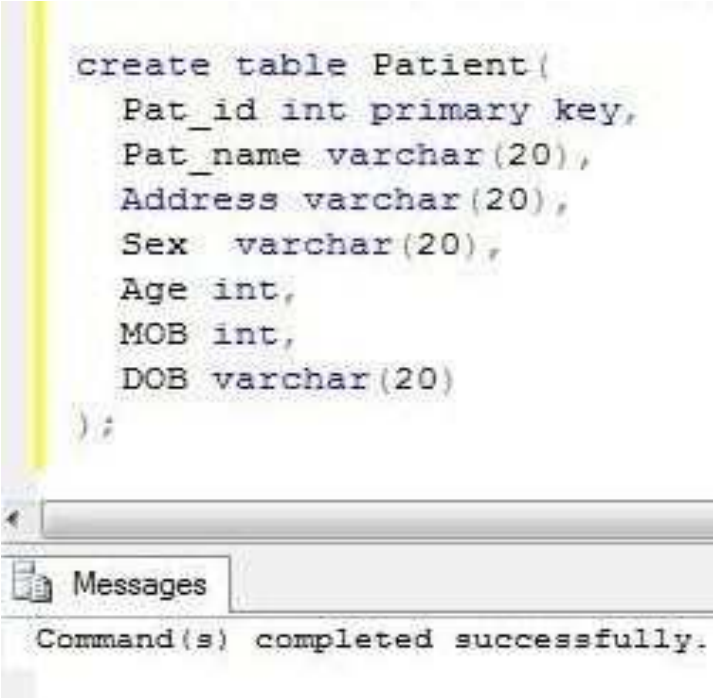
2.5.1 Creation of Tables and Insertion of Data:

In our thesis we create tables and insert data using SQL server and SQL Language.

MAIN TABLE

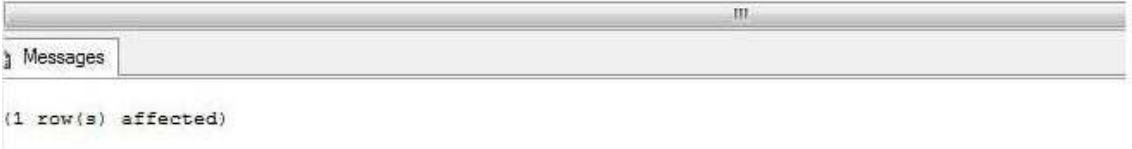
Create Patient Table

```
create table Patient(  
    Pat_id int primary key,  
    Pat_name varchar(20),  
    Address varchar(20),  
    Sex varchar(20),  
    Age int,  
    MOB int,  
    DOB varchar(20)  
);
```

The image is a screenshot of a SQL Server Enterprise Manager window. The top pane shows a SQL script with the following text: `create table Patient(
 Pat_id int primary key,
 Pat_name varchar(20),
 Address varchar(20),
 Sex varchar(20),
 Age int,
 MOB int,
 DOB varchar(20)
);`. The bottom pane, titled "Messages", displays the message "Command(s) completed successfully." in a monospaced font. The window has a standard Windows-style title bar and a scroll bar on the left side of the script pane.

Insert Values into Patient Table

```
insert into Patient values(1,'Moni','Uttara','Female',20,01914564987,'12/05/10')
```



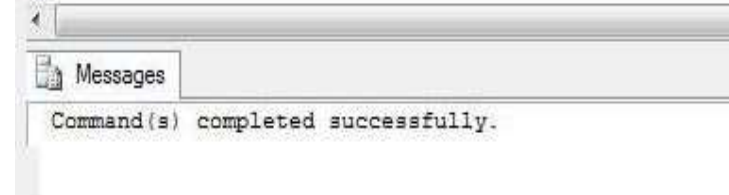
The screenshot shows a database window with a 'Messages' tab. Below the tab, the text '(1 row(s) affected)' is displayed, indicating that the insert operation was successful.

In this way we create all the main tables and insert data in them.

Junction Table

Create Admission Table

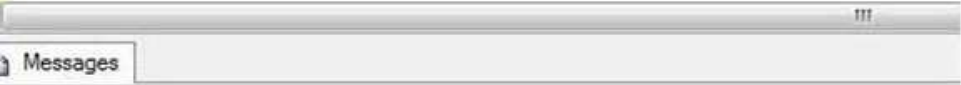
```
create table Admission
(
    AD_id int primary key,
    Ad_date varchar(20),
    Pat_name varchar(20) references Patient,
    Rcp_name varchar(20)
    references Receptionist,
    Room_no varchar(20) references Room,
);
```



The screenshot shows a database window with a 'Messages' tab. Below the tab, the text 'Command(s) completed successfully.' is displayed, indicating that the create operation was successful.

Insert Values into Admission Table

```
insert into Admission values (2,1,2,3,'06/05/10','11 A.M.')
```



(1 row(s) affected)

In this way we create all the junction tables and insert data in them.

2.5.2 Sample Data values of Tables

Patient table

Pat_id	Pat_name	Age	Sex	DOB	MOB	Address
1	Moni	20	Female	12/1/80	16245658	Uttara
2	Karim	30	Male	12/12/80	17254698	Mirpur
3	Mamun	10	Male	10/10/200	NULL	Elephantroad
4	Rimi	18	Female	1/10/92	NULL	Mirpur
5	Kamal	50	Male	1/1/1960	19245698	Uttara

Room table

Room_id	Room_no	Room_type	Room_cost
1	R-101	Normal	2000
2	R-102	Normal	2000
3	R-103	Normal	2000
4	R-205	VIP	4000
5	R-206	VIP	4000

Receptionist table

Rcp_id	Rcp_name	Age	Sex	Address	MOB	Shift	Salary
1	Raj	20	Mail	Mirpur	1502235235	Morning	6000
2	Rasel	25	Mail	Mohammadpur	912545682	Evening	6000
3	Mir Karim	30	Mail	Kazipara	1912456352	Night	6000

Admission room table

Admsn_id	Pat_id	Room_id	Rcp_id	date
1	1	1	1	10/01/09
2	2	1	1	10/01/06
3	3	2	2	11/01/06
4	4	2	2	12/01/09
5	5	3	1	10./01/09

Doctor table

Doc_id	Doc_name	Doc_type	Designation	Age	Address	MOB	Passed_from	Salary
1	Selima	Gyanicologist	A ssistant Professor	50	Dhanmondi	1815745685	DMC	20000
2	Rahat	Orthopedist	A ssistant Professor	45	MIrpur	1715871496	DMC	20000
3	Mohammed	Heart Specialist	A ssistant Professor	50	Dhanmondi	1679584562	DMC	20000
4	Kibria	Medicine	A ssistant Professor	40	Mohammedpur	1897545632	SMC	18000
5	Rahima	Medicine	A ssistant Professor	45	Mirpur	1912654578	KMC	18000

Appointment table

Ap_id	Pat_id	Doc_id	Rcp_id	Ap_date
1	1	1	1	10/01/09
2	2	1	2	11/01/09
3	3	1	1	12/01/09
4	4	4	2	13/01/09
5	5	3	2	14/04/09

Bill table

Bill_id	Bill_for	Bill_type	Bill_total
1	Doctor Fee	Cash	500
2	Test Fee	Cash	800
3	Test	Cash	600
4	Doctor Fee	Cash	500
5	Medicine cost	Cash	100

Accountant table

Acct_id	Acct_name	Age	Address	MOB	Working_time	Salary
1	Rajib	55	Mohammadpur	1954568125	Morning	15000
2	Rajin	50	Mirpur	1715845789	Evening	15000
3	Monir	45	Mirpur	1815645362	Night	15000

BillPay table

BP_id	Bill_id	Acct_id	Amount	Pay	Due
1	1	1	800	200	600
2	2	1	600	600	0
3	3	2	600	300	300
4	4	3	600	500	100
5	5	2	800	800	0

Payment Table

Pay_id	Pay_type	Pay_date	Pat_id
1	Cash	12/05/09	1
2	Cash	13/05/09	2
3	Cheque	14/05/098	3
4	Cash	15/05/09	4
5	Cash	16/05/09	5

Medicine table

Mdcn_id	Mdcn_name	Company	M_Date	E_Date	Price
1	Napa	Glaxo	12/12/08	8/11/11	15
2	Omidon	Incepta	12/1/08	12/01/10	30
3	Ace	Incepta	12/1/08	12/1/10	15
4	Yamadin	Glaxo	1/12/10	1/12/12	30
5	Zymet	Glaxo	1/11/09	1/11/10	35

Prescription table

Prs_id	Mdcn_id	Doc_id	Pat_id	Date	Time	Fees
1	1	1	1	12/05/09	8.15AM	500
2	1	2	2	13/05/09	9.00 AM	500
3	2	3	3	16/05/09	12.00PM	500
4	2	2	2	17/05/09	1.00PM	500
5	5	5	5	20/05/09	9.00AM	500

Test table

Test_id	Test_name	Date	Rep_date	Free
1	Blood	12/05/09	14/05/09	500
2	Urine	13/05/06	14/05/09	300
3	X-Ray	14/05/09	15/05/09	800
4	Ultra Sono	15/05/09	16/05/09	400
5	Engiogram	15/05/09	16/05/09	600

Assist table

Serial_no	Test_id	Pat_id	Doc_id	Date	Time
1	1	1	1	12/05/09	9.00AM
2	1	2	1	13/05/09	10.00AM
3	2	2	2	14/05/09	11.00 AM
4	3	3	3	15/05/09	11.00AM
5	5	1	1	16/05/09	12.00PM

OT table

Ot_id	Ot_room_no
1	R-200
2	R-300
3	R-400
4	R-500
5	R-600

Operation table

Op_id	Doc_Id	Pat_id	Ot_id	Op_date	Op_time
1	1	1	1	12/05/09	12.00PM
2	2	1	1	12/05/09	12.00PM
3	3	1	1	12/05/09	12.00PM
4	4	2	2	13/05/09	1.00 PM
5	4	4	4	14/05/09	2.00PM

Department table

Dept_id	Dept_name	Treatment
1	Orthopedics	Bones
2	Burning	Minimize Burn
3	Gynae	Pregnensi and s...
4	Cardiology	HEART
5	Medicine	Give proper med...

Doctor_form_department table

Dfd_id	Doc_id	Dept_id
1	1	1
2	2	1
3	3	3
4	4	4
5	5	5

Nurse table

Nrs_id	Nrs_name	Age	Address	MOB	W_Shift	Experience	Salary
1	Shori	30	Mirpur	18156245225	Morning	5	7000
2	Sima	25	Mirpur	17251478562	Evening	3	6000
3	Koli	25	Mohammadpur	19152846822	MOrning	2	4000
4	Kakoly	22	Mirpur	11919171822	Night	6	7000
5	Prity	32	Uttara	18145879625	Night	8	9000

Nursing service table

Ns_id	Pat_id	Nrs_id	Room_id	Date	Time
1	1	1	1	12/05/09	3.00PM
2	2	1	1	12/05/09	3.05PM
3	3	2	2	12/05/09	6.00PM
4	4	2	2	12/05/09	6.00PM
5	5	3	3	13/05/09	7.00PM

Ward_boy table

Wb_id	Wb_name	MOB	W_shift	Salary
1	Das	15585658222	Morning	2000
2	Kader	16752845621	Morning	2000
3	Saber	17895122121	Evening	2000
4	Salam	19145658452	Evening	2000
5	Shafi	15525252352	Night	2000

Cleaning service table

Cls_id	Pat_id	Wb_id	Room_id	Date	Time
1	1	1	1	12/05/09	6.00PM
2	2	2	1	12/05/09	6.00PM
3	3	3	2	13/05/09	8.00AM
4	4	3	3	14/05/09	8.00PM
5	5	4	4	16/05/09	7.00PM

Driver table

Dri_id	Dri_name	Address	MOB	Shift	Salary
1	Karim	mirpur	1234	MORning	6000
2	Kader	mirpur	23456	Evening	6000
3	Kasem	mirpur	456897	Night	6000
4	Kamal	rampoura	456321	Evening	6000
5	Dipon	rampura	12345698	Night	6000

Ambulance table

Amb_id	Amb_num	Capacity
1	17-1232	8
2	17-1233	6
3	17-1234	6
4	17-1235	8
5	17-1235	8

Ambulance service table

As_id	Pat_id	Dri_id	Amb_id	Date	Time
1	1	1	1	12/05/09	11.00AM
2	1	2	2	13/05/09	12.00PM
3	2	2	2	14/05/05	1.00AM
4	4	3	3	15/05/05	12.00AM
5	5	5	5	20/05/09	10.00AM

Carriers table

Cri_id	Cri_name	MOB	Address	Shift	Sakary
1	Jobbar	159876252	Mirpur	Morning	5000
2	Jamal	165874522	Mohammadpur	Morning	5000
3	Karim	174569856	Mirpur	Morning	5000
4	Arif	181478546	Mohammadpur	Evening	5000
5	Atik	175469859	Mirpur	Evening	5000

Carrying service table

Crs_id	Cri_id	Amb_id	Pat_id	Date	Time
1	1	1	1	12/05/09	12.00AM
2	2	1	1	12/05/09	12.00AM
3	3	2	2	13/05/09	11.00AM
4	4	2	2	10/05/09	12.00AM
5	5	2	2	10/05/09	12.00AM

2.6 Complex Queries

After completing the implementation we retrieved different information from the system by joining 2 or more tables of the system. Sample Examples are given below:

Question 1

Which tests are suggested by doctor Selima to which Patients?

Query 1:

```
select Pat_name, Doc_name, Test_name from tbl_Patient, tbl_Doctor, tbl_Test ,  
tbl_Assist where Doc_name='Selima' and tbl_Doctor.Doc_id = tbl_Assist.Doc_id and  
Tbl_Patient.Pat_id = Tbl_Assist.Pat_id and tbl_Test.Test_id = tbl_Assist.Test_id
```

Output:

	Pat_name	Doc_name	Test_name
1	Moni	Selima	Blood
2	Karim	Selima	Blood
3	Moni	Selima	Engiogram

Question 2

Which doctors prescribed which medicine to patient Mamun?

Query 2:

```
select Pat_name, Doc_name, Mdcn_name from tbl_Patient, tbl_Doctor, tbl_Medicine,  
tbl_Prescription where Pat_name = 'Mamun' and tbl_Patient.Pat_id = tbl_Prescription.Pat_id  
and tbl_Doctor.Doc_id = tbl_Prescription.Doc_id and tbl_Medicine.Mdcn_id =  
tbl_Prescription.Mdcn_id
```

Output 2:

	Pat_name	Doc_name	Mdcn_name
1	Mamun	Mohammed	Omidon

Question 3:

Which Doctors are from which Department and they passed from which college and got salaries below 20000 taka?

Query 3

```
select Doc_name,Passed_from,Dept_name from tbl_Doctor, tbl_Department,
tbl_DFD where Salary <20000 and tbl_Doctor.Doc_id =
tbl_DFD.Doc_id and tbl_Department.Dept_id =
tbl_DFD.Dept_id
```

Output 3:

	Doc_name	Passed_from	Dept_name
1	Kibria	SMC	Cardiology
2	Rahima	KMC	Medicine

Question-4

Which doctor conducted the Urine Test for which Patient at 11.00 AM?

Query -4

```
select pat_name,doc_name from tbl_Patient,tbl_Doctor,tbl_Test,tbl_Assist where
tbl_Assist.Time='11.00AM' and tbl_Test.Test_name='Urine' and
tbl_Patient.Pat_id=tbl_Assist.Pat_id and tbl_Doctor.Doc_id=tbl_Assist.Doc_id
```

Output 4:-

	pat_name	doc_name
1	Mamun	Mohammed

Question 5:

Which Patient is carried by which driver in Ambulance serial no 5?

Query -5:

```
select Pat_name,Dri_name from tbl_Patient,tbl_Driver,tbl_AmbulanceService where Amb_id =  
5 and tbl_Patient.Pat_id=tbl_AmbulanceService.Pat_id and tbl_Driver.Dri_id =  
tbl_AmbulanceService.Dri_id
```

Output 5 :-

	Pat_name	Dri_name
1	Kamal	Dipon

Question 6:

In which time receptionist Rasel appointed patient Kamal to Doctor Selima?

Query 6:

```
Select pat_name,Ap_time from tbl_patient,tbl_Receptionist,tbl_Appointment,tbl_Doctor where  
Rcp_name='Rasel'and Doc_name='Selima'and pat_name='kamal' and  
tbl_patient.pat_id=tbl_Appointment.pat_id and tbl_Receptionist.Rcp_id=tbl_Appointment.Rcp_id
```

Output 6:

	pat_name	Ap_time
1	Kamal	8.00PM

INTERFACING THE DATABASE SYSTEM USING .NET FRAMEWORK

3.1 Research on Interface Design Guidelines

❖ User Interface

User interface should be designed to match the skills, experience and expectations of its anticipated users. System users often judge a system by its interface rather than its functionality.

❖ Objectives

- To suggest some general design principles for user interface design.
- To explain different interaction styles and their use.
- To explain when to use graphical and textual information presentation.
- To explain the principal activities in the user interface design process.
- To introduce usability attributes and approaches to system evaluation.[8]

❖User Interface Design Principle

Principle	Description
User familiarity	The interface should use terms and concepts which are drawn from the experience of the people who will make most use of the system.
Minimal Surprise	Users should never be surprised by the behavior of a system.
Recoverability	The interface should include mechanisms to allow users to recover from errors.
User Guidance	The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.
User diversity	The interface should provide appropriate interaction facilities for different types of system users.

❖User Interface Design Guidelines

1. Consistency

- It is known as ("Principle of least astonishment").
- Certain aspects of an interface should behave in consistent ways at all times for all screens
- Terminology should be consistent between screens
- Icons should be consistent between screens
- Colors should be consistent between screens of similar function.[9]

2. Simplicity

- Break complex tasks into simpler tasks
- Break long sequences into separate steps
- Keep tasks easy by using icons, words etc.
- Use icons/objects that are familiar to the user. [9]

3. Match between system and the real world

- The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.
- Follow real-world conventions, making information appear in a natural and logical order.[9]

4. Human Memory Limitations

- Organize information into a small number of "chunks"
- Try to create short linear sequences of tasks
- Don't flash important information onto the screen for brief time periods
- Organize data fields to match user expectations, or to organize user input (e.g. auto formatting phone numbers)
- Provide cues/navigation aids for the user to know where they are in the software or at what stage they are in an operation
- Provide reminders, or warnings as appropriate
- Provide ongoing feedback on what is and/or just has happened
- Let users recognize rather than recall information
- Minimize working memory loads by limiting the length of sequences and quantity of information - avoid icon mania![9]

5. Display issues

- Maintain display inertia - make sure the screen changes little from one screen to the next within a functional task situation
- Organize screen complexity
- Eliminate unnecessary information
- Use concise, unambiguous wording for instructions and messages
- Use easy to recognize icons
- Use a balanced screen layout - don't put too much information at the top of the screen - try to balance information in each screen quadrant
- Use plenty of 'white space' around text blocks - use at least 50% white space for text screens
- Group information logically
- Structure the information rather than just presenting a narrative format (comprehension can be 40% faster for a structured format).[9]

6. Error prevention

- Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.[9]

7. Help and documentation:

- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.

- Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.[9]

8. System messages:

- Provide user-centered wording in messages (e.g. "there was a problem in copying the file to your disk" rather than "execution error 159")
- Avoid ambiguous messages (e.g. hit 'any' key to continue - there is no 'any' key and there's no need to hit a key, reword to say 'press the return key to continue')
- Avoid using threatening or alarming messages (e.g. fatal error, run aborted, kill job, catastrophic error)
- Use specific, constructive words in error messages (e.g. avoid general messages such as 'invalid entry' and use specific phrases such as 'please enter your name')
- Make the system 'take the blame' for errors (e.g. "illegal command" versus "unrecognized command").[9]

9. Attention

- Use attention grabbing techniques cautiously (e.g. avoid overusing 'blinks' on web pages, flashing messages, bold colors etc.)
- Don't use more than 4 different font sizes per screen
- Use serif or sans serif fonts appropriately as the visual task situation demands.
- Don't use all uppercase letters - use and uppercase/lowercase mix
- Don't overuse audio or video
- Use colors appropriately and make use of expectations (e.g. don't have an OK button colored red! use green for OK, yellow for 'caution, and red for 'danger' or 'stop')

- Don't use more than 4 different colors on a screen
- Don't use blue for text (hard to read), blue is a good background color.
- Don't put red text on a blue background
- Use high contrast color combinations
- Use colors consistently
- Use only 2 levels of *intensity* on a single screen
- On text screens don't use more than 3 fonts on a single screen. [9]

10. Anthropomorphization

- Don't anthropomorphize (i.e. don't attribute human characteristics to objects) - avoid the "Have a nice day" messages from your computer. [9]

11. Choose specific fonts, font sizes and font characteristics to represent certain types of information

With the proliferation of high resolution display devices, designers no longer need to be as concerned about the technical problems associated with what types of fonts and font characteristics are used on the monitor. Using a particular font in a particular location or for a particular portion of a program can aid users when searching for screens that contain the type of information they are searching for. Font characteristics such as bold, italic, and underlining can be used to designate key words that are hot or active. [10]

12. Provide selectable areas to allow users to access information

Some possible selectable areas to consider are buttons and hot text within a text field. The location of these elements on the screen will depend on the available screen real estate and the function of the selectable areas. It is recommended that the placement of selectable areas be tested with users to find out what is the optimal location for them. The selectable area will be a control element for users to access information. The control chosen will depend on the task to be done. Be consistent in implementing particular controls for particular functions. [10-15]

13. Provide visual effects to give users visual feedback that their choices have been made and registered by the program

Buttons, icons, and menus can be highlighted or animated to show users that a choice has been made. Keep the highlighting or animation simple. The duration of a highlight or animation should be long enough to be registered visually by the users, but short enough so that users are not waiting for an animation to be over so that they can get to the information they want.

Visual effects, such as wipes, fades, and zooms may be used to indicate access to a particular piece of information. The use of these visual effects should be consistent. Do not use them simply because they are available, but rather use them to indicate a particular action of the program. Additionally, be consistent in the use of a visual effect. If wipes are used when clicking on a right arrow, use them throughout the program. If zoom outs are used when clicking on a menu item, then use zoom INS when returning to the menu. Above all, make the visual effect have meaning and be consistent with its use throughout the program. [10-15]

❖ Human Factors in Interface Design

□ Limited Short-term memory

- People can instantaneously remember about 7 items of information. If you present more than this, they are more liable to make mistakes.

□ People make mistakes

- When people makes mistakes and systems go wrong, inappropriate alarms and messages can increase stress and hence the likelihood of more mistakes.

□ People are different

- People have a wide range of physical capabilities. Designers should not just design for their own capabilities.

□ People have different interaction preferences

- Some people like picture and some like text. [16]

Sample of Interfaces

Here we show some samples of Interfaces:

SAMPLE-1

The screenshot shows a DOS-based interface titled "Q & A Report Writer". The main window displays "ADULT PROBATION DATA" with a list of fields on the left and a corresponding data entry area on the right. The fields include: LAST NAME, COURT NAME, TRUE NAME, ADDRESS, CITY, DATE OF BIRTH, SEX, HEIGHT, ALIAS OR NICK, IDENTIFY MARKS, PROBATION OFF, EMPLOYER, CONTACT PERSON, CRUSE NUMBER, OFFENSE, LENGTH OF SENT, LENGTH SUS SENT, MODIF DATE, and DOC REL DATE. The data entry area includes fields for FIRST NAME, MIDDLE IN, FILE NUMBER, TRANSFER, HOME PHONE, STATE, ZIP, SOCIAL SECURITY #, EYES, HAIR, WORK PHONE, PHONE, COURT INFORMATION, SENTENCING DATE, DURATION OF PROB, PROBATION ON DATE, and PROBATION OFF DATE. The bottom of the window shows a status bar with "ADULT.DTF", "Retrieve Spec", and "Page 1 of 10".

DOS-Based Q&A[16]

SAMPLE-2

The screenshot shows a Windows-based search form titled "Search Board (Newer) : Form". The form has a heading "Begin Search" and contains several input fields for search criteria: Last Name, First Name, True Name, Court Name, Date of Birth, Social Security Number, Address, City, State, and Zip. There is also a "Status" dropdown menu currently set to "Active". A "Search" button and a "Reset Search Criteria" button are located at the bottom right of the form.

Begin Search Form Created with MS Access [16]

Tippecanoe County Probation Department has made DOS

Based Q&A and A Search Form using Access as shown in Sample-1 and Sample-2.

SAMPLE-3:



Connection Dialog Box using VB.Net[16]

#Usability Attributes:

Attribute	Description
Learnability	How long does it take a new user to become productive with the system?
Speed of operation	How well does the system response match the user's work practice?
Robustness	How tolerant is the system of user error?
Recoverability	How good is the system at recovering from user errors?
Adaptability	How closely is the system tied to a single model of work?

Source: [16]

Summary

We can say that we have to design interfaces clearly and efficiently according to the user choice. A poorly designed interface can cause a user to make catastrophic errors. Poor user interface design is the reason why so many software systems are never used.

3.2 FRONT END Design

Introduction:

Front end and **Back End** are generalized terms that refer to the initial and the end stages of a process. The front end is responsible for collecting input in various forms from the user and processing it to conform to a specification the back end can use. The front end is an interface between the user and the back end.

- ❖ The separation of software systems into front and back ends simplifies development and separates maintenance.
- ❖ For major computer subsystems, a graphical file manager is a front end to the computer's file system. The front end faces the user and the backend launches the programs of the operating system in response.[17]

We have completed the **backend design** using **SQL Server** and now we have designed the **front end** using **.NET Framework/(C#)**.

3.2.1 FORMS DESIGN:

Front end Forms Design includes

- ☐ **Login Form**

- ☐ **Form Menu**

- ☐ **Admin Part**

- ✓ **Accountant Form**
- ✓ **Receptionist Form**
- ✓ **Nurse Form**
- ✓ **Room Form**
- ✓ **Ward boy Form**
- ✓ **Ambulance Form**
- ✓ **Carrier Form**
- ✓ **Driver Form**
- ✓ **Bill Form**
- ✓ **Admission Form**
- ✓ **Appointment Form**
- ✓ **Ambulance Service Form**
- ✓ **Carrying Service Form**
- ✓ **Nursing Service Form**
- ✓ **Cleaning Service Form**
- ✓ **Payment Form**

□ Medical Part

- ✓ Patient Form
- ✓ Doctor Form
- ✓ Department Form
- ✓ Medicine Form
- ✓ Test Form
- ✓ Operation Theater Form
- ✓ Doctor's from Department Form
- ✓ Prescription Form
- ✓ Assist Form
- ✓ Operation Form

□ Search Option

□ Login Form:

This form comes at the very beginning of the software:



The screenshot shows a Windows-style window titled "LOGIN". Inside the window, there is a header section with a circular logo on the left and the text "Diabetic Association of Bangladesh" on the right. Below the header, there is a login form with two input fields: a dropdown menu and a text box. The text "Password" is visible below the text box. At the bottom of the window, there are two buttons: "Login" and "Cancel".

Fig: Login page



When Designation and password will match we can switch to the Form Menu.

□ **Form Menu:**



In this form we can see a menu strip and there are many menu options like Entry, Search, View, Tools, Windows, Help and other icons.

Example: In Patient Form which comes under Medical Part of Entry menu bar, we can enter the new patient data.

In Search option under Search menu bar we can retrieve information of different tables of our choice according to Search criteria.

ADMINISTRATION PART

The way we enter data in the administration forms is given below.



Room Form:

Hospital Management(BIRDEM) - [frmRoom]

Entry Search View Tools Windows Help

Room_Id

Room Number

Room Type

Room Cost

	Room_id1	Room_no1	Room_type1	Room_cost1	ID
1	R-101	Normal	2500	0	
2	R-102	Normal	2000	0	
3	R-103	Normal	2000	0	
4	R-205	VIP	4000	0	
6	R-105	Normal	500	0	
7	R-798	VIP	8870	0	
8	R-106	Normal	7000	0	

AddNew Save Update Delete Cancel

Status

Bill Form:

Bill Id

Bill For

Bill Type

Bill Total

Bill_id1	Bill_for1	Bill_type1	Bill_total1	ID
1	Doctor	Cash	500	0
2	Medicine	Cash	800	0
3	Test	Cash	600	0
4	Operation	Cash	500	0
5	Test	Cash	6000	0
7	Operation	Cash	500	0

AddNew **Save** **Update** **Delete** **Cancel**

Status

Accountant Form:

Hospital Management(BIRDEM) - [frmAccountant]

Entry Search View Tools Windows Help

Accountant Id

Accountant Name

Mobile

Shift

Salary

	Acct_jd1	Acct_name1	Address1	MOB1	Salary1
▶	1	Rajib	Mohammedpur	1718765860	15000
	2	Rajin	Mirpur	1913133760	15000
	3	Monir	Mirpur	1718765862	15000
	4	Sadek	Kazipara	1913133768	34000
	5	Dulal	Framget	1718765868	23000

Status

Receptionist Form:

Hospital Management(BIRDEM) - [frmReceptionist]

Entry Search View Tools Windows Help

Receptionist_Id

Receptionist_Name

Mobile

Shift

Salary

	Rcp_jd1	Rcp_name1	Address1	MOB1	Sex1	Salary1
▶	1	Raj	Mirpur	1718765866	Male	6000
	2	Rasel	Mohamedpur	1718765867	Male	6000
	3	Mir Karim	Kazipara	1718765868	Male	6000
	4	Saaif	Ultra	1718765861	Male	7000
	5	Zayed	Framget	1718765862	Male	6000

Status

Driver Form

frmDriver

Driver_Id
Driver_Name
Address
Mobile
Salary

	Dri_id1	Dri_name1	Address1	MOB1	Salary1
▶	1	Karim	mirpur	1712346789	6000
	2	Kader	mirpur	1712345665	6000
	3	Kasem	mirpur	1712456897	6000
	4	Kamal	rampoura	1913456321	6000
	5	Dipon	rampura	12345698	6000

AddNew
 Save
 Update
 Delete
 Cancel

Ambulance Form

Hospital Management(BIRDEM) - [frmAmbulance]

Entry Search View Tools Windows Help

Ambulance_Id
Ambulance Number
Ambulance Capacity

	Amb_id1	Amb_num1	Capacity1	ID
▶	1	17-1232	8	0
	2	17-1233	6	0
	3	17-1234	6	0
	4	17-1235	8	0

AddNew
 Save
 Update
 Delete
 Cancel

Status

Carriers Form

Hospital Management(BIRDEM) - [frmCarriers]

Entry Search View Tools Windows Help

Carriers_Id

Carriers_Name

Address

Mobile

Salary

	Cri_id1	Cri_name1	Address1	MOB1	Salary1
▶	1	Jobbar	Mirpur	1718765860	5000
	2	Jamal	Mohhamdpur	1718765861	5000
	3	Karim	Mirpur	1718765862	5000
	4	Arif	Mohhamdpur	1718765863	5000

Status

Nurse Form

Hospital Management(BIRDEM) - [frmNurse]

Entry Search View Tools Windows Help

Nurse_Id

Nurse_Name

Address

Mobile

Shift

Exprience

Salary

	Nrs_id1	Nrs_name1	Address1	MOB1	W_shift1	Exprience1
▶	1	Nipa	Mirpur_1	1718765860	Moming	5_years
	2	Nila	Mirpur_10	1718765861	Night	6_years
	3	Umi	Mirpur_2	1718765862	Evening	10_years
	4	Shimu	Kazipara	1718765863	Moming	1_years
	5	Fatema	Mirpur	1718765864	Evening	3_years

Status

Ward boy Form

Hospital Management(BIRDEM) - [frmWardboy]

Entry Search View Tools Windows Help

Wardboy_Id
Wardboy_Name
Address
Mobile
Salary

	Wb_id1	Wb_name1	Address1	MOB1	Salary1
▶	1	Das	Dhaka	1718765860	2000
	2	Kader	Khulna	1913133760	2000
	3	Saber	Comilla	1913133765	2000
	5	Shafi	Kazipara	1670520296	2000

AddNew Save Update Delete Cancel

Status

Admission Form

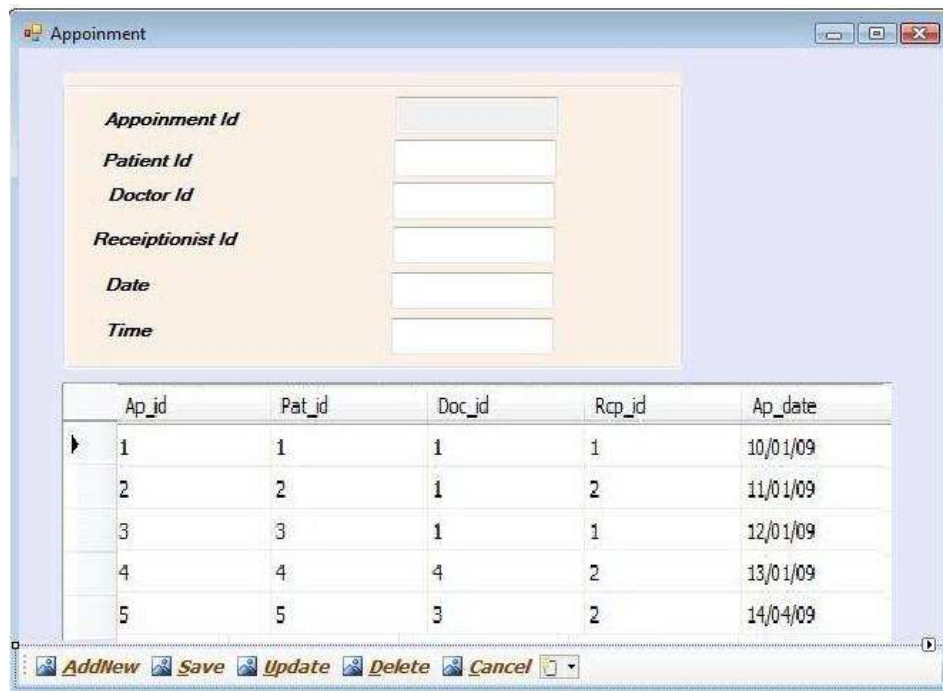
Admission

Admission Id
Patient Id
Room Id
Receptionist Id
Date
Time

	Admsn_id	Pat_id	Room_id	Rcp_id	date
▶	1	1	1	1	10/01/09
	2	2	1	1	10/01/06
	3	3	2	2	11/01/06
	4	4	2	2	12/01/09
	5	5	3	1	10./01/09

AddNew Save Update Delete Cancel

Appointment Form

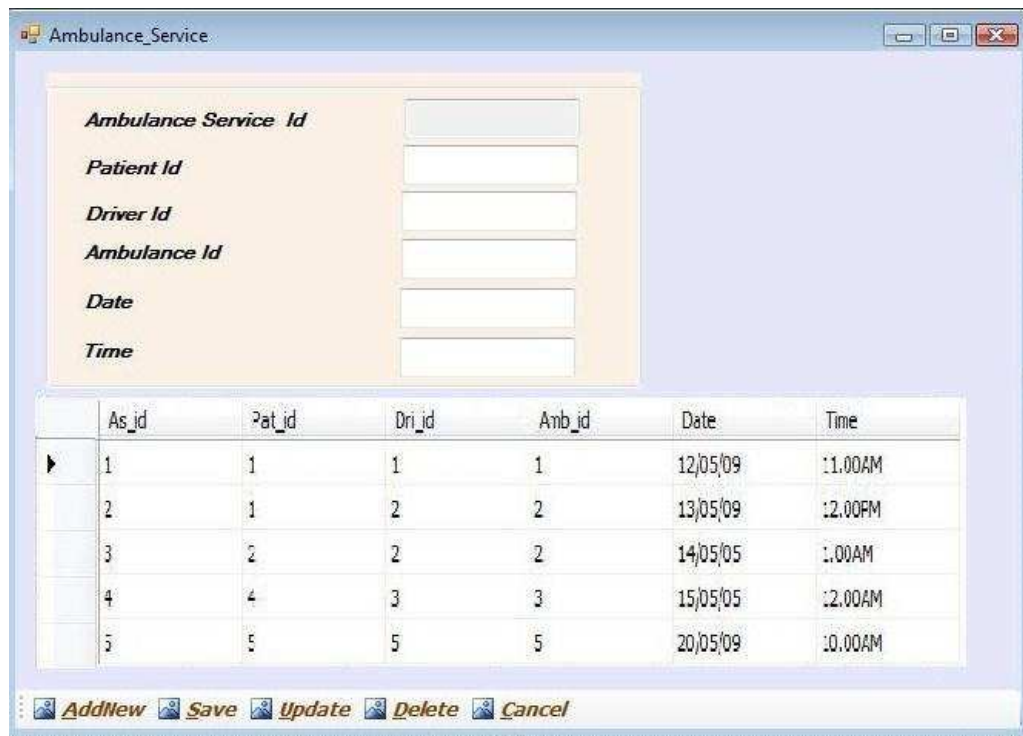


The Appointment Form is a software window titled "Appointment". It features a form area with input fields for "Appointment Id", "Patient Id", "Doctor Id", "Receptionist Id", "Date", and "Time". Below the form is a table with 5 rows and 5 columns. The columns are labeled "Ap_id", "Pat_id", "Doc_id", "Rcp_id", and "Ap_date". The table contains the following data:

Ap_id	Pat_id	Doc_id	Rcp_id	Ap_date
1	1	1	1	10/01/09
2	2	1	2	11/01/09
3	3	1	1	12/01/09
4	4	4	2	13/01/09
5	5	3	2	14/04/09

At the bottom of the window is a toolbar with buttons: "AddNew", "Save", "Update", "Delete", "Cancel", and a dropdown menu.

Ambulance Service Form



The Ambulance Service Form is a software window titled "Ambulance_Service". It features a form area with input fields for "Ambulance Service Id", "Patient Id", "Driver Id", "Ambulance Id", "Date", and "Time". Below the form is a table with 5 rows and 7 columns. The columns are labeled "As_id", "Pat_id", "Dri_id", "Amb_id", "Date", and "Time". The table contains the following data:

As_id	Pat_id	Dri_id	Amb_id	Date	Time
1	1	1	1	12/05/09	11.00AM
2	1	2	2	13/05/09	12.00PM
3	2	2	2	14/05/05	1.00AM
4	4	3	3	15/05/05	12.00AM
5	5	5	5	20/05/09	10.00AM

At the bottom of the window is a toolbar with buttons: "AddNew", "Save", "Update", "Delete", "Cancel", and a dropdown menu.

Nursing Service Form

Nursing_Service

Nursing Service Id

Patient Id

Nurse Id

Room Id

Date

Time

	Ns_id	Pat_id	Nrs_id	Room_id	Date	Time
▶	1	1	1	1	12/05/09	3.00PM
	2	2	1	1	12/05/09	3.05PM
	3	3	2	2	12/05/09	6.00PM
	4	4	2	2	12/05/09	6.00PM
	5	5	3	3	13/05/09	7.00PM

AddNew
 Save
 Update
 Delete
 Cancel

Cleaning Service Form

Cleaning_Service

Cleaning Service Id

Patient Id

Ward boy Id

Room Id

Date

Time

	Cls_id	Pat_id	Wb_id	Room_id	Date	Time
▶	1	1	1	1	12/05/09	6.00PM
	2	2	2	1	12/05/09	6.00PM
	3	3	3	2	13/05/09	8.00AM
	4	4	3	3	14/05/09	8.00PM
	5	5	4	4	16/05/09	7.00PM

AddNew
 Save
 Update
 Delete
 Cancel

Carrying Service Form

Carrying_Service

Carrying Service Id

Carrier Id

Room Id

Patient Id

Date

Time

	Crs_id	Cri_id	Amb_id	Pat_id	Date	Time
▶	1	1	1	1	12/05/09	12.00AM
	2	2	1	1	12/05/09	12.00AM
	3	3	2	2	13/05/09	11.00AM
	4	4	2	2	10/05/09	12.00AM
	5	5	2	2	10/05/09	12.00AM

Payment Form

Payment

Payment Id

Pay_type

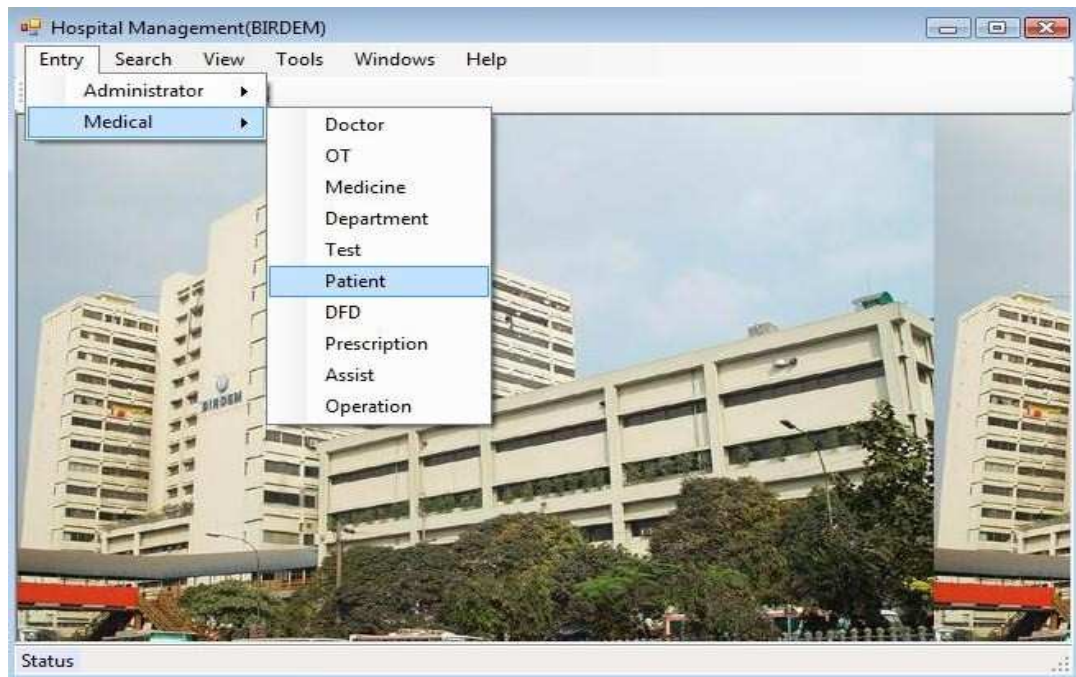
Pay_date

Patient Id

	Pay_id	Pay_type	Pay_date	Pat_id
▶	1	Cash	12/05/09	1
	2	Cash	13/05/09	2
	3	Cheque	14/05/098	3
	4	Cash	15/05/09	4
	5	Cash	16/05/09	5

MEDICAL PART

The way we enter data in the Medical forms is given below.



Doctor Form

The screenshot shows the 'Hospital Management(BIRDEM) - [frmDoctor]' application window. The menu bar includes 'Entry', 'Search', 'View', 'Tools', 'Windows', and 'Help'. The form contains several input fields for doctor information:

- Doctor_Id
- Doctor_Name
- Doctor_Type
- Designation
- Address
- MOB
- Passed_From
- Salary

Below the form is a table displaying a list of doctors:

	Doc_id1	Doc_name1	Doc_type1	Designation1	Address1	MOB1
▶	1	Selima	Gynocologist	Assitent Professor	Dhanmondi	1913133760
	2	Rahat	Arthopetics	Assitent Professor	Mirpur	1913133761
	3	Mohammed	Heart Specialist	Assitent Professor	Dhanmondi	1913133762
	4	Kibria	Medicine	Assitent Professor	Mohammedpur	1913133763
	5	Rehman	Medicine	Assitent Professor	Mohammedpur	1913133764

OT Form

Hospital Management(BIRDEM) - [frmOT]

Entry Search View Tools Windows Help

Operation_Id
Ot_room_no

	Ot_id1	Ot_room_no1	ID
▶	1	R-200	0
	2	R-300	0
	3	R-400	0
	4	R-500	0
	5	R-600	0
	6	R-400	0

AddNew Save Update Delete Cancel

Status

Medicine Form

Hospital Management(BIRDEM) - [frmMedicine]

Entry Search View Tools Windows Help

Medicine_Id
Medicine_Name
Company
Manufacture_Date
Expire_Date
Price

	Mdcn_id1	Mdcn_name1	Company1	M_date1
▶	1	Napa	Glaxo	12/12/08
	2	Omidon	Incepta	12/1/08
	3	Ace	Incepta	12/1/08

Status

Department Form

Hospital Management(BIRDEM) - [frmDepartment]

Entry Search View Tools Windows Help

Department_Id
Department_Name
Treatment

	Dept_id1	Dept_name1	Treatment1	ID
▶	1	Arthopedics	Bones	0
	2	Buming	Minimize Bum	0
	3	Gayni	Pregnensi and so...	0

Status

Test Form

Hospital Management(BIRDEM) - [frmTest]

Entry Search View Tools Windows Help

Test_Id

Test_Name

Date

Report_Date

Fees

	Test_id1	Test_name1	Date1	Rep_date1	Fees1
▶	1	Blood	12/05/09	14/05/09	800
	2	Urine	13/05/06	14/05/09	900
	3	X-Ray	14/05/09	15/05/09	400

AddNew Save Update Delete Cancel

Status

Patient Form

Hospital Management(BIRDEM) - [frmPatient]

Entry Search View Tools Windows Help

Patient_Id

Patient_Name

Address

Date Of Birth

Mobile

Age

Sex

	Pat_id1	Pat_name1	Address1	DOB1	MOB1	Age1
▶	1	Moni	Uttara	12/1/80	1718765860	20
	2	Karim	Mirpur	12/12/80	1718765861	30
	3	Mamun	Elephantroad	10/10/200	1718765862	10
	4	Rimi	Mirpur	1/10/92	1718765863	18

AddNew Save Update Delete Cancel

Status

DFD Form

DFD

Dfd Id

Doctor Id

Department Id

	Dfd_id	Doc_id	Dept_id
▶	1	1	1
	2	2	1
	3	3	3
	4	4	4
	5	5	5

... AddNew Save Update Delete Cancel

Prescription Form

Prescription

Prescription Id

Medicine Id

Doctor Id

Patient Id

Date

Time

Fees

	Prs_id	Mdcn_id	Doc_id	Pat_id	Date	Time
▶	1	1	1	1	12/05/09	8.15AM
	2	1	2	2	13/05/09	9.00 AM
	3	2	3	3	16/05/09	12.00PM
	4	2	2	2	17/05/09	1.00PM
	5	5	5	5	20/05/09	9.00AM

... AddNew Save Update Delete Cancel

Assist Form

Assist

Serial No

Test Id

Patient Id

Doctor Id

Date

Time

	Serial_no	Test_id	Pat_d	Doc_id	Date	Time
▶	1	1	1	1	12/05/09	9.00AM
	2	1	2	1	13/05/09	10.00AM
	3	2	2	2	14/05/09	11.00 AM
	4	3	3	3	15/05/09	11.00AM
	5	5	1	1	16/05/09	12.00PM

Operation Form

Operation

Operation Id

Doctor Id

Patient Id

OT Id

Date

Time

	Cp_id	Doc_Id	Pat_id	Ot_id	Op_date	Op_time
▶	1	1	1	1	12/05/09	12.00PM
	2	2	1	1	12/05/09	12.00PM
	3	3	1	1	12/05/09	12.00PM
	4	4	2	2	13/05/09	1.00 PM
	5	4	4	4	14/05/09	2.00PM

Search Form

We can search the data in the way given below



The screenshot shows the 'Hospital Management(BIRDEM) - [Search_information]' window. The title bar reads 'Hospital Management(BIRDEM) - [Search_information]'. The menu bar includes 'Entry', 'Search', 'View', 'Tools', 'Windows', and 'Help'. Below the menu bar, there are several icons. The main area contains a 'Search Options' section with a label 'Id_Number' and a dropdown menu showing '2'. Below this, there is a 'Room' dropdown menu. A 'Search' button is located to the right of the 'Id_Number' dropdown. Below the search options, there is a table with the following columns: 'Room_id', 'Room_no', 'Room_type', and 'Room_cost'. The first row of the table is highlighted in blue and contains the values '2', 'R-102', 'Normal', and '2000'. Below the table, there is a large empty space. At the bottom left, there is a 'Status' label.

	Room_id	Room_no	Room_type	Room_cost
▶	2	R-102	Normal	2000
*				

In this form we can search different information of our software according to search criteria.

3.2.2 Relating Interface Design Guidelines to our Front end Design:

In our front end we refer to the User Interface Guidelines that we researched

1. Match between system and the real world

- **The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.[9]**
 - ➔ In our system we tried to make it more users friendly and familiar to the user, so that it should speak the user's language.
- **Follow real-world conventions, making information appear in a natural and logical order.[9]**
 - ➔ To match between real world and the system we tried to arrange all the information of our system appears in a natural and logical order.

2. Help and Documentation

- **Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.[9]**
 - ➔ To follow this guideline we tried to make our information list small and easy to search.

3. Attention

- **Don't use more than 4 different font sizes per screen.[9]**

→ In our front ends we use a single font (Comic Sans MS) in 4 Different sizes.

- **Don't use all uppercase letters - use and uppercase/lowercase mix.[9]**

→ If we use all uppercase or lowercase letters it is not so comfortably visible for users. That's why we have followed the instructions and mixed the upper and lower cases.

- **Don't overuse audio or video.**

→ We do not use any audios or videos in the forms.

- **Use colors appropriately and make use of expectations (e.g. don't have an OK button colored red! use green for OK, yellow for 'caution, and red for 'danger' or 'stop').**

→ We do not use buttons in red color in our front end, we use system color in the button and it looks good with the background color.

- **Don't use more than 4 different colors on a screen.**

→ We use two different colors in our front ends. The colors are Lavender (as background), Linen (in the Groupbox).

- **Don't use blue for text (hard to read), blue is a good background color.**
 - ➔ We use blue as our front end background and black as text color.
- **Don't put red text on a blue background.**
 - ➔ Red is not matchable on a blue background .To follow the guidelines we don't use it in our system.
- **Use italic, underlining, bold, inverse video or other markers sparingly.**
 - ➔ We use italic and bold in our form texts.
- **Use colors consistently.[9]**
 - ➔ We tried to use colors consistently.

Summary

We try our level best to follow the guidelines which were very helpful for us in our form design. We hope a user friendly and efficient interface has been developed.

3.3 Security feature of FRONT END

- ✓ Security has to be compared to related concepts: Safety, continuity, reliability. The key difference between security and reliability is that security must take into account the actions of people attempting to cause destruction.

Here, we discuss about security for any Computer Software System. To start this topic we must have to know about Computer system security and Database Security. [26]

❖ **Computer System Security**

- The term computer system security means the collective processes and mechanisms by which sensitive and valuable information and services are protected from publication, tampering or collapse by unauthorized activities or untrustworthy individuals and unplanned events respectively.
- Computer security is critical in almost any technology-driven industry which operates on computer systems. Computer security can also be referred to as computer safety. [19]
- Database security includes the system, processes, and procedures that protect a database from unintended activity.
- Data security is the means of ensuring that data is kept safe from corruption and that access to it is suitably controlled.
 - ✓ Data security helps to ensure privacy.
 - ✓ Helps in protecting personal data.

To control and work with the Database Security we need an administrator. [20]

❖ Features of Database Administrator:-

- ✓ Database administrators work with database management software and determine ways to store, organize, analyze, use, and present data.
- ✓ Identify user needs and set up new computer databases. Database administrators must integrate data from old systems into a new system.
- ✓ Test and coordinate modifications to the system when needed. [21]

An organization's database administrator ensures the performance of the system, understands the platform on which the database runs, and adds new users to the system.

Our Software is about BIRDEM Hospital Management System. After comprehending the importance of security we try to secure our system from any type of unintended activity.

In our security panel there are 3 types of members.

1. Administrator,
2. Receptionist,
3. Accountant.



Fig : Login page of our software.



Fig: When we run our software we can see 3 options



Fig: Administrator option is selected and password is entered



Fig: The password is matched

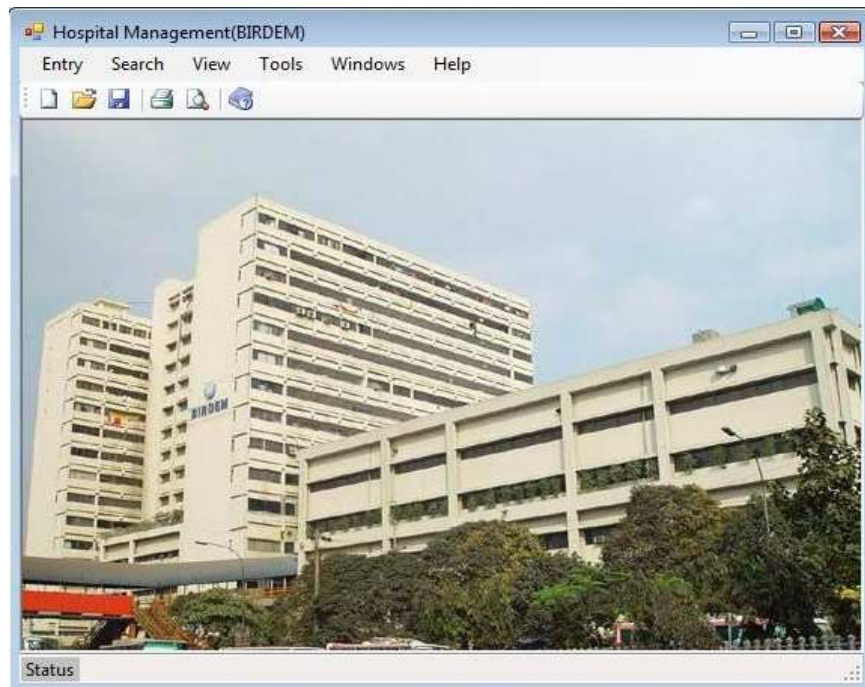


Fig: When the password is matched we can switch to the Form Menu

A screenshot of a "LOGIN" window. The window has a title bar with "LOGIN" and standard window controls. The main content area features the logo of the "Diabetic Association of Bangladesh" on the left, which includes a circular emblem with a traffic light and the text "DISCIPLINE IS LIFE" and "DIABETIC ASSOCIATION OF BANGLADESH". To the right of the logo, the text "Diabetic Association of Bangladesh" is displayed. Below this, there is a login form with two rows of input fields. The first row has a dropdown menu with "Accountant" selected and a text box containing "Accountant". The second row has a label "Password" and a text box with five dots, indicating a masked password. At the bottom of the window, there are two buttons: "Login" and "Cancel".

Fig: Now the Accountant option is selected and password is entered.



Fig: The password is matched



Fig: When the password is matched we can switch to the Form Menu



LOGIN

DISCIPLINE IS LIFE
DIABETIC ASSOCIATION OF BANGLADESH

Receptionist

Receptionist

Password

.....

Login Cancel

Fig: Now Receptionist option is selected and password is entered



LOGIN

DISCIPLINE IS LIFE
DIABETIC ASSOCIATION OF BANGLADESH

Receptionist

Receptionist

Password

.....

Login Cancel

Password Matched

OK

Fig: The password Matched.



Fig: When the password is matched we can switch to the Form Menu.



Fig: If the password does not match Error Message is showed.

❖ Security Code:

```
int password = Convert.ToInt32(passwordtext.Text);

    if (DesignationText.Text == "Administrator" && password
== 62413)
    {
        MessageBox.Show("Password Matched");
        frmMenubar f = new frmMenubar();
        f.Show();
    }

    else if (DesignationText.Text == "Accountant" && password
== 62436)
    {
        MessageBox.Show("Password Matched");
        frmMenubar f = new frmMenubar();
        f.Show();
    }
```

```

        else if (DesignationText.Text == "Receptionist" &&
password == 62444)
    {
        MessageBox.Show("Password Matched");
        frmMenubar f = new frmMenubar();
        f.Show();
    }
else
{
    MessageBox.Show("Invalid Password");
}
}

```

We design the security part of our Software by following a Point which is taken from the User Guidelines Interfaces. It is very much helpful for us to think and design the interface of our software in this respect. The point is given below:-

❖ **Provide selectable areas to allow users to access information**

- Some possible selectable areas to consider are buttons and hot text within a text field. The location of these

elements on the screen will depend on the available screen real estate and the function of the selectable areas.

- ✓ Here we use the 'Login' and 'Cancel' as a 'Button' and also use group box, where we include combo box, textbox and label.
- It is recommended that the placement of selectable areas be tested with users to find out what is the optimal location for them.
- The selectable area will be a control element for users to access information. The control chosen will depend on the task to be done. Be consistent in implementing particular controls for particular functions. [10-15]

To control and work with the Database Security we do some tasks which are given below ->

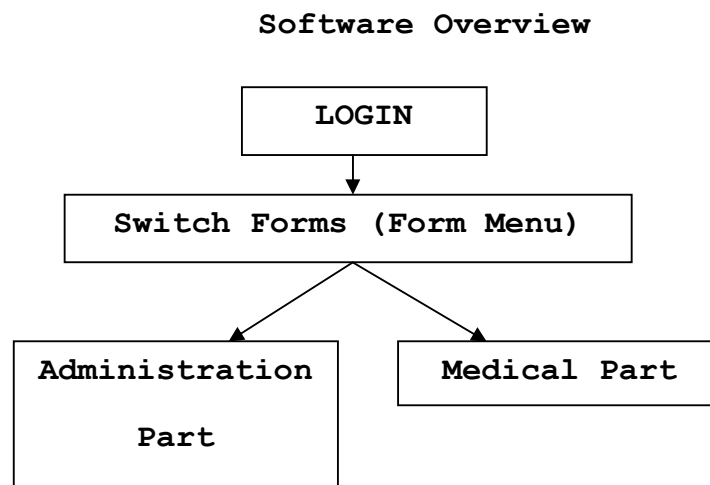
- ✓ Determine ways to store, organize, analyze, use, and present data.
- ✓ Identify user needs and set up new computer databases,
- ✓ Ensure privacy, to protect personal data by Testing and coordinate modifications to the system when we need so.

❖ **Summary:**

Security is very important in software development. We apply security in our software so that any user cannot access the information, entered by the input users. We control the security from the front end. It works efficiently.

3.4 Implementation of Insert, Delete, Update buttons and Search Option

In our software save, delete and update buttons are very common features and search option is a special feature. These buttons carry out the actions as their names imply, Search option helps to search info according to selections of id and table name.



Login Form

Action:

- ✓ When the Designation and Password matched we can go to the next step "Form Menu".
- ® Refer to the codes and Description in the Appendix.

Form Menu

Action:

- ✓ We can easily switch to the different forms of our software.
- ® Refer to the codes and Description in the Appendix.

Administration part:

- a. Room Form
- b. Bill Form
- c. Accountant Form
- d. Receptionist Form
- e. Driver Form
- f. Ambulance Form
- g. Carriers Form
- h. Nurse Form
- i. WardBoy Form
- j. Admission Service Form
- k. Appointment Service Form
- l. Ambulance Service Form
- m. Nursing Service Form
- n. Cleaning Service Form
- o. Carrying Service Form
- p. Payment Form

Medical Part:

- i. Doctor Form
- ii. OT Form
- iii. Medicine From
- iv. Department Form
- v. Test Form
- vi. Patient Form
- vii. DFD Form
- viii. Prescription Form
- ix. Assist Form
- x. Operation Form

Save, Delete, Update and Search codes are similar for all the forms.

So we are describing the codes of the Room form as an example.

Save Action for Room Form

Code

```
private void btSave_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Save();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
    ButtonControl(false);
}
```

When Save button is clicked these codes are executed.

We can see three functions

a. Validation()

This function checks all the insert data in the form is valid or not.

```
private bool Validation()
{
    if (textBox1.Text == "")
    {
        MASICEIU.MessageShow.Information("Select Item from room list.");
        return false;
    }
    else if (textBox2.Text == "")
    {
        MASICEIU.MessageShow.Information("Room No");
        textBox2.Focus();
        return false;
    }
    else if (textBox3.Text == "")
    {
        MASICEIU.MessageShow.Information("Room Type");
        textBox3.Focus();
        return false;
    }
    else if (textBox4.Text == "")
    {
        MASICEIU.MessageShow.Information("Room Cost");
        textBox4.Focus();
        return false;
    }
    return true;
}
```

b. SetRoomInstant();

This function sets instances and convert variables to string if necessary.

```
private void SetRoomInstant()
{
    roomInstant.Room_id1 = Convert.ToInt16(textBox1.Text);
    roomInstant.Room_no1 = textBox2.Text;
    roomInstant.Room_type1 = textBox3.Text;
    roomInstant.Room_cost1 = Convert.ToInt16(textBox4.Text);
}
```

c. ClearTextBox()

This function clears all the textbox of the form after Save button is clicked.

```
private void ClearTextBox()
{
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
}
```

Delete Action for Room Form

Code

```
private void btDelete_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Delete();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
}
```

We can also see three functions

- a. Validation()
- b. SetRoomInstant();
- c. ClearTextBox()

® The descriptions of these functions have been described earlier.

Update Action for Room Form

Code

```
private void btUpdate_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Update();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
}
```

We can also see three functions

- a. `Validation()`
- b. `SetRoomInstant();`
- c. `ClearTextBox()`

® The descriptions of these functions have been described earlier.

Search Action

In the search form combobox2 we can select a form's data grid view as shown as page 104. Then we can search id from the combobox1 as shown as page 104. Accordingly single row is displayed. The code is given below:

Code

```
private void Search_Click(object sender, EventArgs e)
{
    if (comboBox2.Text != "" && comboBox1.SelectedIndex > -1)
    {
        dataGridView1.DataSource =
CommonDataAccess.GetData(comboBox1.Text, comboBox2.Text);
    }
}
```

In order to do the Save, Delete, Update and Search we use 3 helping files

- RoomDataAccess.cs
- RoomDataObject.cs
- RoomService.cs

® The description of these classes and codes are described in the Appendix part.

3.5 Usage of DLL file

❖ DLL File

Dynamic-link library (also written without the hyphen), or DLL, is Microsoft's implementation of the shared library concept in the Microsoft Windows and OS/2 operating systems. These libraries usually have the file extension DLL, OCX (for libraries containing ActiveX controls), or DRV (for legacy system drivers). The file formats for DLLs are the same as for Windows EXE files — that is, Portable Executable (PE) for 32-bit and 64-bit Windows, and New Executable (NE) for 16-bit Windows. As with EXEs, DLLs can contain code, data, and resources, in any combination. [22]



Fig: DLL (Dynamic-link library) Details

Key	Value
FILEVERSION	1, 0, 0, 0
PRODUCTVERSION	1, 0, 0, 0
FILEFLAGSMASK	0x3fL
FILEFLAGS	0x8L
FILEOS	VOS_WINDOWS32
FILETYPE	VFT_DLL
FILESUBTYPE	VFT2_UNKNOWN
Block Header	Language Neutral (000004b0)
Assembly Version	1.0.0.0
Comments	This Software is made for Thesis Implementation of CSE -4-2 (Ahsanullah University of Science & Technology)
CompanyName	AUSTCSE18
FileDescription	masiceiu
FileVersion	1.0.0.0
InternalName	masiceiu.dll
LegalCopyright	Copyright © austcse4-2 2010
LegalTrademarks	@13364442
OriginalFilename	masiceiu.dll
PrivateBuild	Fully Private
ProductName	masiceiu
ProductVersion	1.0.0.0
SpecialBuild	

Fig: the DLL file used in our Software

Source: [22]

❖ The purpose of using the DLL file

- Using DLL file we can easily carry our database with our software.
- We don't need to load the database first.
- The software becomes more efficient and user friendly.
- After using DLL file we do not need to load the database to interface with the front end in different PC s.

CONCLUSION AND FUTURE WORK

4.1 Conclusion

By the grace of Allah, the Almighty we have come to the end of our thesis report. It is not the work of one day. In fact it took us a year to complete. The group members worked hard to make it a good and improvised thesis.

Summing up, we worked on a case study of BIRDEM Hospital Management, designing and storing its information in a sample database of our creation. We designed ER models, Relational Models and Normalized tables of the relational model and finally implemented the SQL Server Diagram, filled the server tables with data values and queried different useful information from the database.

The second part of the thesis involved developing a user friendly and efficient interface to the backend database in SQL Server. We researched User Interface Guidelines and applied some of those to our front end forms design. We have taken into account issues of security too.

4.2 Future Work

While an efficient user friendly interface to SQL-based backend database has been successfully developed, we have in mind some scope for future work involving Guideline View Features and Trigger Features. These are explained as follows.

4.2.1 Data GridView:

The DataGridView control provides a customizable table for displaying data. The DataGridView class allows customization of cells, rows, columns, and borders through the use of properties such as DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyle, and GridColor.

We can use a DataGridView control to display data with or without an underlying data source. Without specifying a data source, we can create columns and rows that contain data and add them directly to the DataGridView using the Rows and Columns properties. You can also use the Rows collection to access DataGridViewRow objects and the DataGridViewRow.Cells property to read or write cell values directly. The Item indexer also provides direct access to cells.

As an alternative to populating the control manually, we can set the DataSource and DataMember properties to bind the DataGridView to a data source and automatically populate it with data.

When working with very large amounts of data, you can set the VirtualMode property to true to display a subset of the available data. Virtual mode requires the implementation of a data cache from which the DataGridView control is populated. [23]

❖ Use Data Gridview in .NET FRAMEWORK

1. Retrieve Data from the Database:

The screenshot shows a Windows application window titled "Operation Theature". Inside the window, there is a form with two input fields labeled "Operation_Id" and "Ot_room_no". Below these fields is a DataGrid. The DataGrid has two columns: "Ot_id" and "Ot_room_no". The first row of the DataGrid is highlighted with a yellow background and contains an asterisk (*) in the "Ot_id" column. At the bottom of the window, there are two buttons labeled "Close" and "Save".

	Ot_id	Ot_room_no
*		

Fig: When A DataGridView is loaded in a form.

	Ot_id	Ot_room_no	Edit	Delete
▶	1	R-100	Edit	Delete
	2	R-200	Edit	Delete
	3	R-300	Edit	Delete
	4	R-400	Edit	Delete
	5	R-500	Edit	Delete
	6	R-605	Edit	Delete

Fig: When we run the form, GridView retrieves data from the database.

Here we can see Operation Theater Information where Ot_id and Ot_room_no are the information. We manage to add two more columns named Edit and Delete. Edit Column contain Edit Button and Delete Column Contains Delete Button.

❖ See the recently entered data:

The screenshot shows a window titled "Operation Theature". Inside, there is a form with two input fields: "Operation_id" containing the value "7" and "Ot_room_no" containing the value "R-700". Below the form is a table with the following data:

	Ot_id	Ot_room_no	Edit	Delete
▶	1	R-100	Edit	Delete
	2	R-200	Edit	Delete
	3	R-300	Edit	Delete
	4	R-400	Edit	Delete
	5	R-500	Edit	Delete
	6	R-605	Edit	Delete

At the bottom of the window, there are two buttons: "Close" and "Save". A red arrow points to the "Save" button.

Fig: Inserting New Data in the form and clicking the *Save* Button.

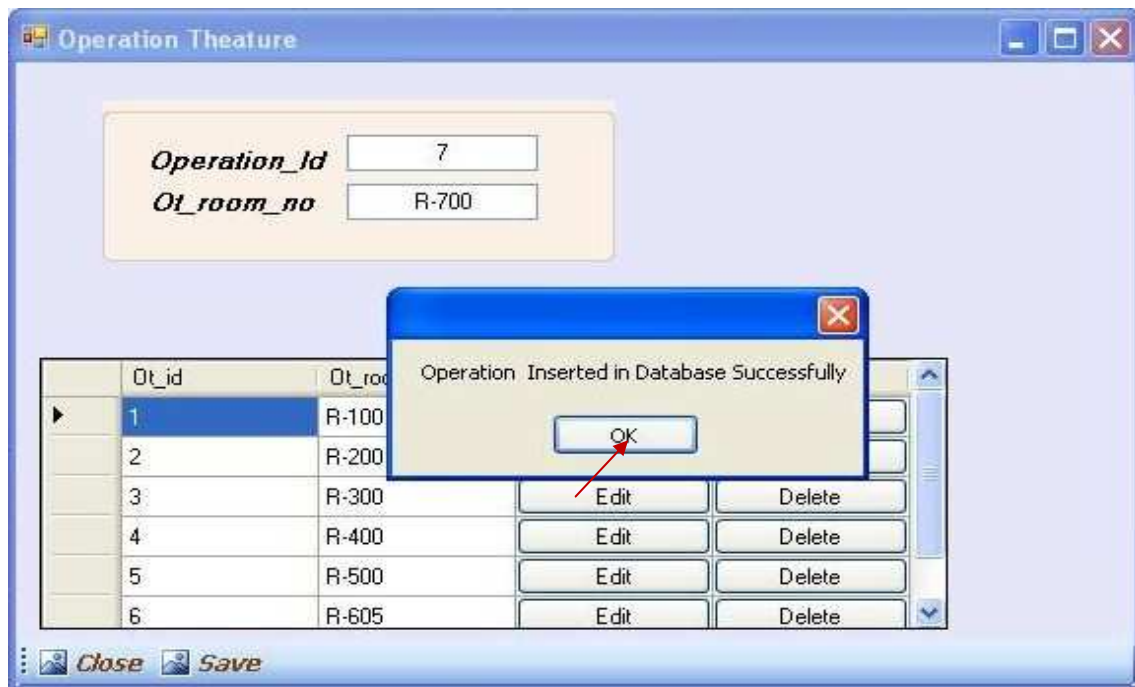


Fig: Confirmation of Data Insertion in the Database

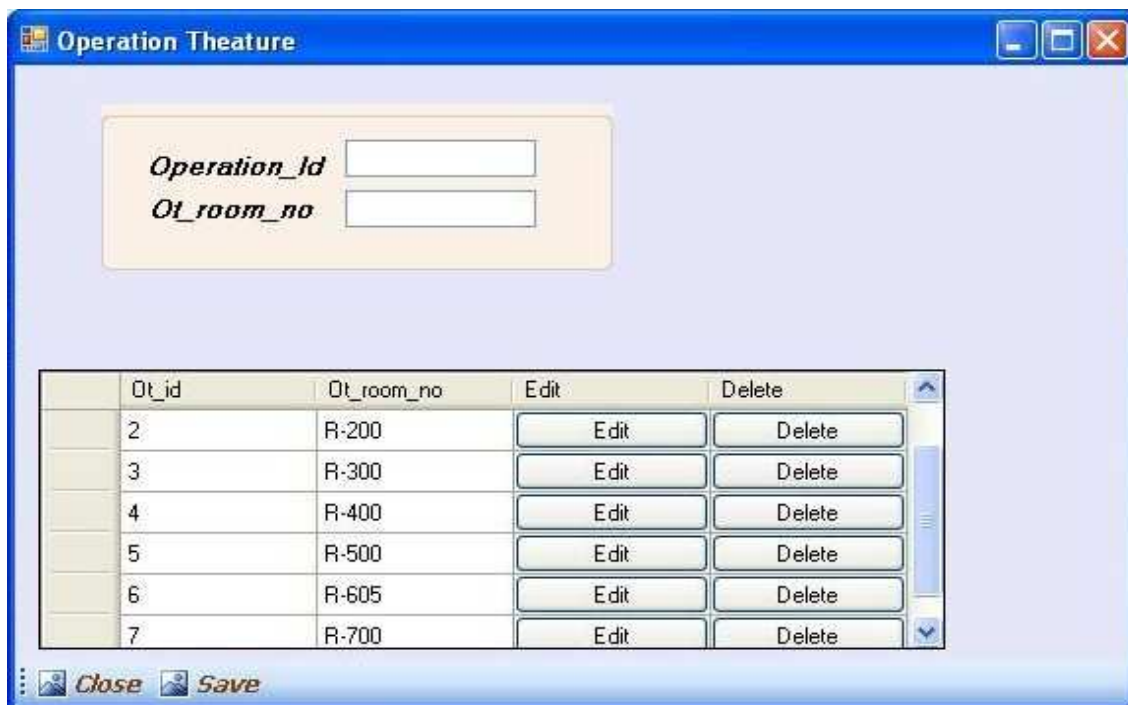


Fig: Recently inserted data is seen in the DataGridView.

❖ Delete Data from the DataGridView:

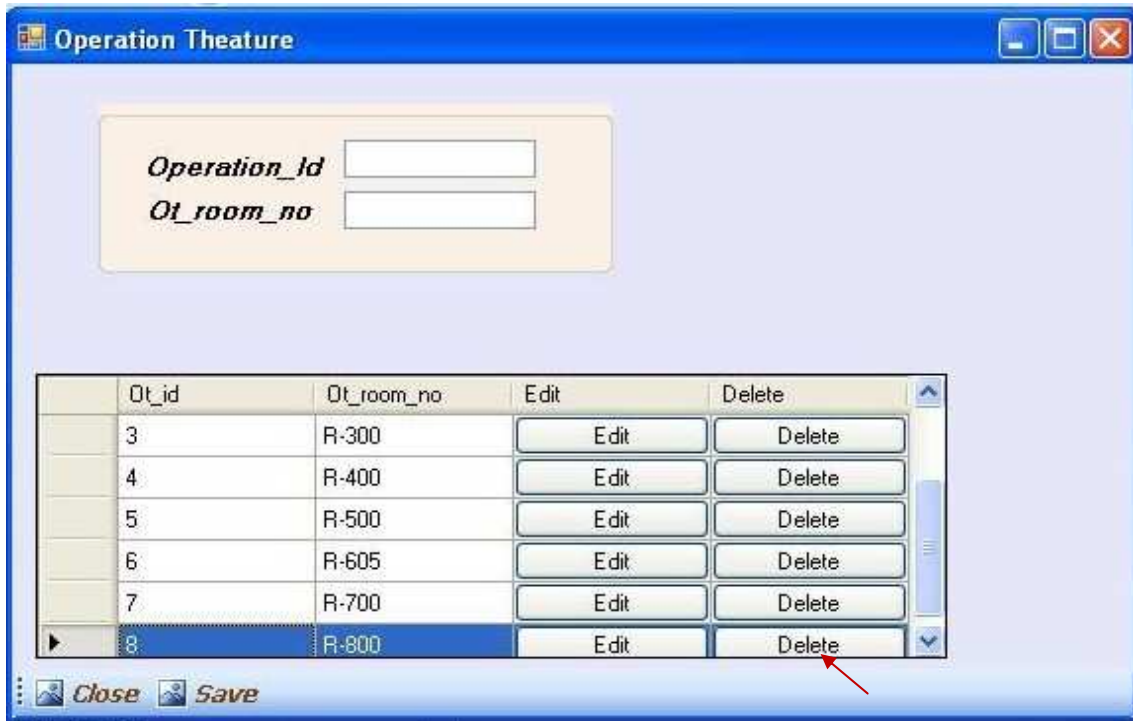


Fig: A Column is selected to Delete.

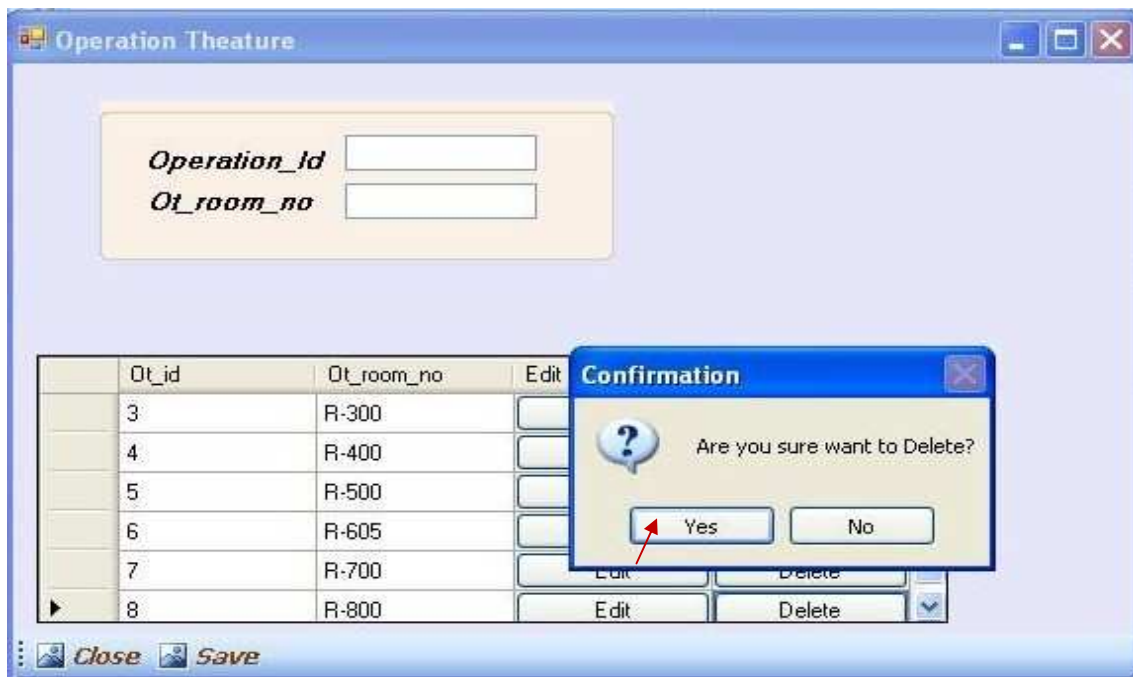


Fig: A Message Box is Displayed for the User Confirmation

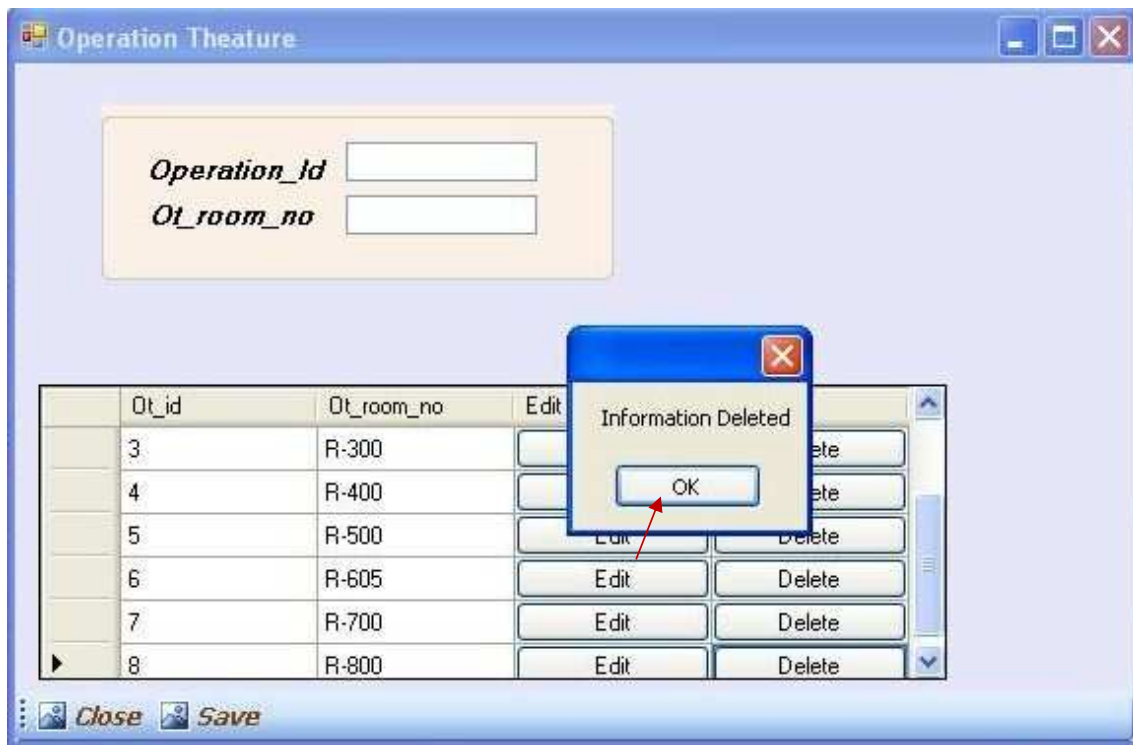


Fig: Confirmation that the information or data is deleted successfully.

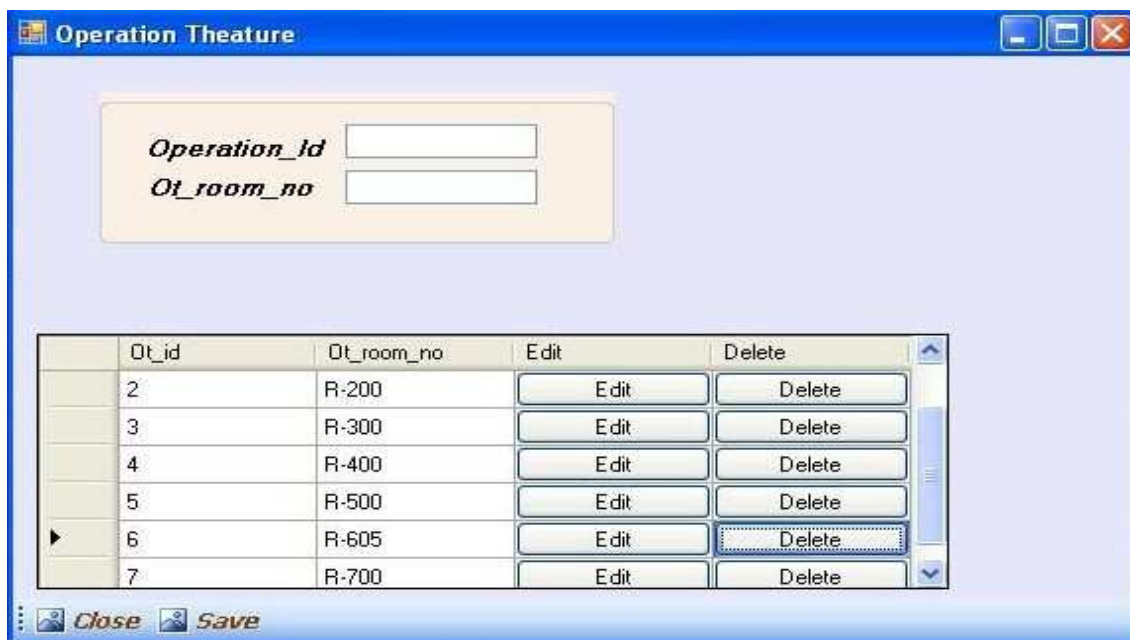


Fig: The Picture of Grid View after the data is deleted.

Edit and Update Data from the DataGridView:

The screenshot shows a Windows application window titled "Operation Theature". At the top, there are two input fields: "Operation_Id" with the value "7" and "Ot_room_no" with the value "R-700". Below these is a DataGridView with the following data:

	Ot_id	Ot_room_no	Edit	Delete
	2	R-200	Edit	Delete
	3	R-300	Edit	Delete
	4	R-400	Edit	Delete
	5	R-500	Edit	Delete
	6	R-605	Edit	Delete
	7	R-700	Edit	Delete

At the bottom of the window, there are two buttons: "Close" and "Update". A red arrow points to the "Edit" button in the last row of the DataGridView, and a white arrow points to the "Update" button at the bottom.

Fig: When the Edit Button is clicked the data is seen in the form. The *Save* Button is changed to *Update* Button.

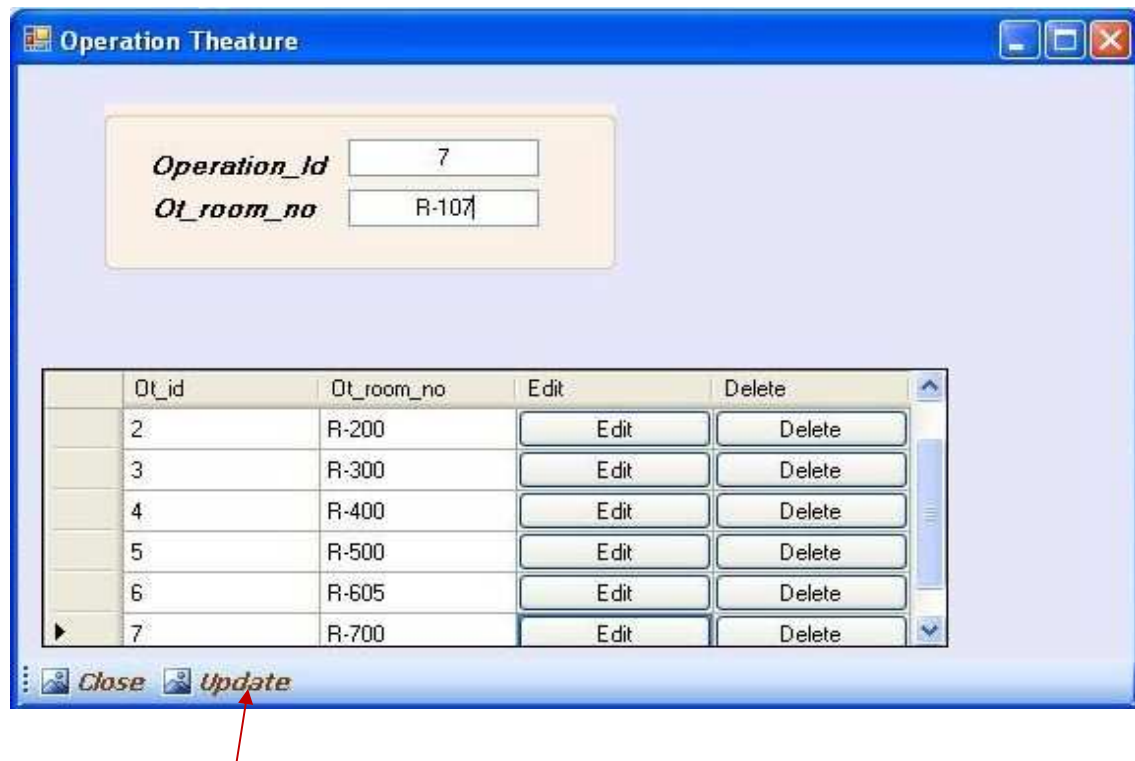


Fig: After Edit the data *Update* button is clicked.

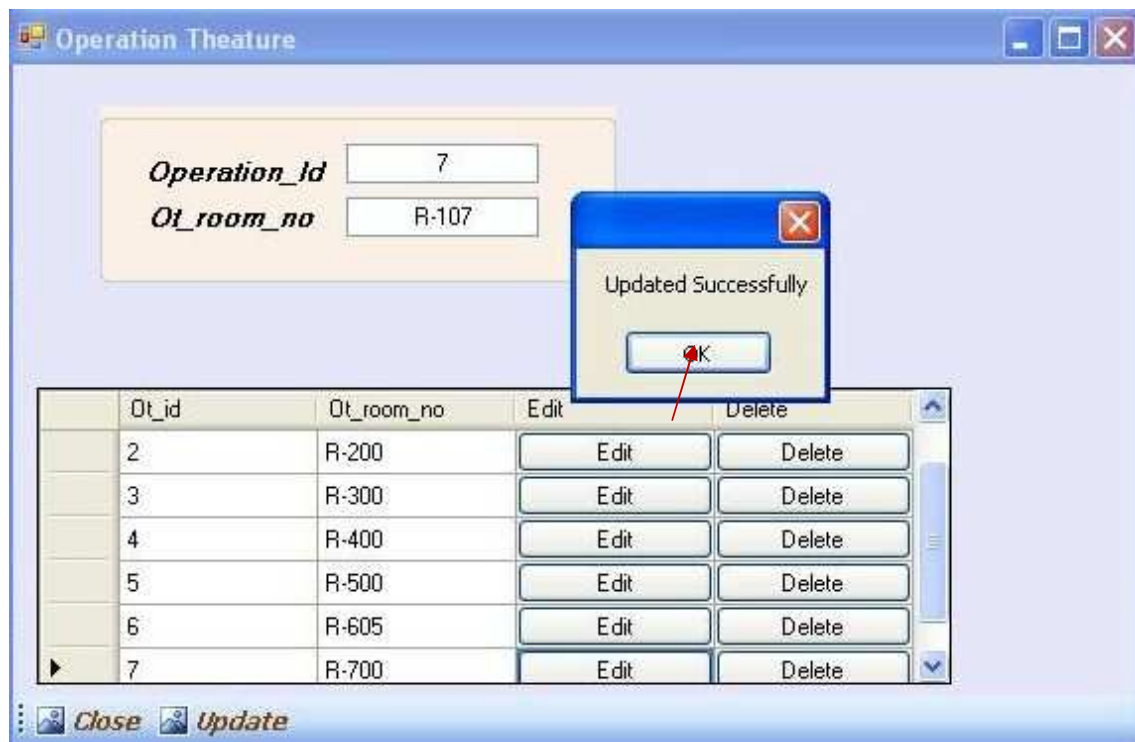


Fig: Confirmation that the data is updated.

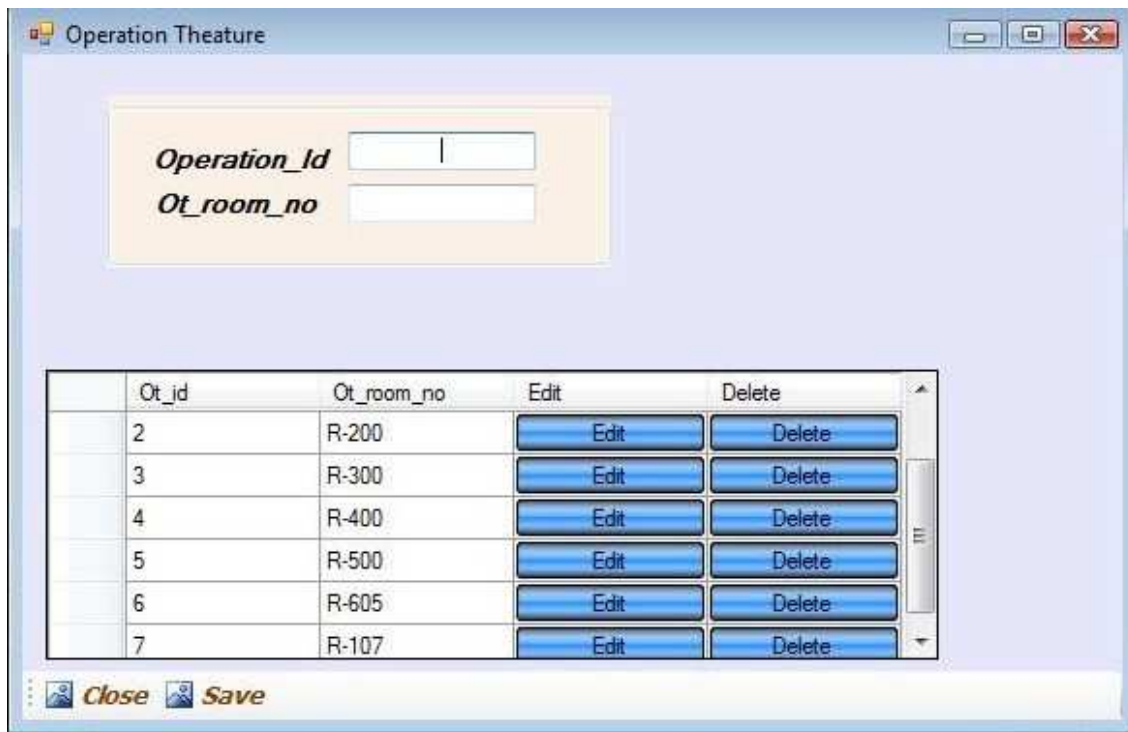


Fig: The Picture of the DataGridView after data update.

#Codes for Data Gridline View:

Code_OT Class:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace OperationTheature
{
    public partial class OT : Form
    {
        public OT()
        {
            InitializeComponent();
        }
    }
}
```



```

public int otid;
OperationBasic ob = new OperationBasic();
OTGateway og = new OTGateway();
OTManager om = new OTManager();
bool isTrue = false;
public string msg = null;

private void OT_Load(object sender, EventArgs e)
{
    this.tbl_OTTableAdapter1.Fill(this.db_PatientDataSet1.tbl_OT);

    this.AddColumns();
}

private void LoadInitializes()
{
    OTManager om = new OTManager();
    OTDataGridview.DataSource = om.ShowOperation();
}

private void AddColumns()
{
    DataGridViewButtonColumn EditCol = new DataGridViewButtonColumn();
    EditCol.Name = "Edit";
    EditCol.Text = "Edit";
    EditCol.UseColumnTextForButtonValue = true;
    this.OTDataGridview.Columns.Add(EditCol);
    DataGridViewButtonColumn DeleteCol = new DataGridViewButtonColumn();
    DeleteCol.Name = "Delete";
    DeleteCol.Text = "Delete";
    DeleteCol.UseColumnTextForButtonValue = true;
    this.OTDataGridview.Columns.Add(DeleteCol);
}

public void Clear()
{
    operationidText.Text = null;
    otroomnoText.Text = null;
    this.Save_Button.Text = "Save";
}

private void Save_Button_Click(object sender, EventArgs e)
{
    OTManager om = new OTManager();
    OperationBasic ob = new OperationBasic();

```

```

ob.ot_id = Convert.ToInt32(operationidText.Text);
ob.ot_room_no = oroomnoText.Text;

if (this.otid == 0)
{
    msg = om.SaveOperation(ob);

    MessageBox.Show(msg);
    this.OTDataGridview.Columns.Remove("Edit");
    this.OTDataGridview.Columns.Remove("Delete");
}
else
{
    OTGateway og = new OTGateway();
    og.UpdateOperation(ob);
    MessageBox.Show("Updated Successfully");
    this.OTDataGridview.Columns.Remove("Edit");
    this.OTDataGridview.Columns.Remove("Delete");
}
Clear();
OT_Load(null, null);
}

private void DeleteButton_Click(object sender, EventArgs e)
{
    ob = new OperationBasic();
    om = new OTManager();
    int operationcode = ob.ot_id =
Convert.ToInt32(OTDataGridview.Rows[OTDataGridview.SelectedCells[0].RowIndex].Cells["Ot
_id"].Value.ToString());

OTDataGridview.Rows[OTDataGridview.SelectedCells[0].RowIndex].Cells["Ot_id"].Value.ToStri
ng();

    msg = om.DeleteOperation(operationcode);
    Close();
    MessageBox.Show(msg);
}

private void OTDataGridview_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == this.OTDataGridview.Columns["Edit"].Index)
    {
        EditAction(e);
    }
    else if (e.ColumnIndex == this.OTDataGridview.Columns["Delete"].Index)
    {
        DeleteAction(e);
    }
}

```

```

    }
}

private void EditAction(DataGridViewCellEventArgs e)
{
    otid=int.Parse(this.OTDataGridView.Rows[e.RowIndex].Cells[0].Value.ToString());

    this.operationidText.Text=otid.ToString();

    this.otroomnoText.Text=this.OTDataGridView.Rows[e.RowIndex].Cells[1].Value.ToString();
    this.Save_Button.Text = "Update";
}

private void DeleteAction(DataGridViewCellEventArgs e)
{
    if (MessageBox.Show("Are you sure want to Delete?", "Confirmation",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {
        otid = int.Parse(this.OTDataGridView.Rows[e.RowIndex].Cells[0].Value.ToString());
        og.DeleteOperation(otid);
        MessageBox.Show("Information Deleted");
        this.OTDataGridView.Columns.Remove("Edit");
        this.OTDataGridView.Columns.Remove("Delete");

        OT_Load(null, null);
    }
}
}
}
}

```

Difficulties:

- ✓ This process works well but some times changes of commands make forms disable and invalid.
- ✓ If we give more time and afford we can complete the software using Grid view control in the future.

4.2.2 TRIGGER Features:

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for keeping the integrity of the information on the database. For example, when a new record (representing a new worker) is added to the employees table, new records should be created also in the tables of the taxes, vacations, and salaries.

We can write triggers that fire whenever one of the following operations occurs:

1. DML statements (INSERT, UPDATE, DELETE) on a particular table or view, issued by any user.
2. DDL statements (CREATE or ALTER primarily) issued either by a particular schema/user or by any schema/user in the database.
3. Database events, such as logon/logoff, errors, or startup/shutdown, also issued either by a particular schema/user or by any schema/user in the database.

Triggers are similar to stored procedures. A trigger stored in the database can include SQL and PL/SQL or Java statements to run as a unit and can invoke stored procedures. However, procedures and triggers differ in the way that they are invoked. A procedure is explicitly run by a user, application, or trigger. Triggers are implicitly fired by Oracle SQL server when a triggering event occurs, no matter which user is connected or which application is being used. [24, 25]

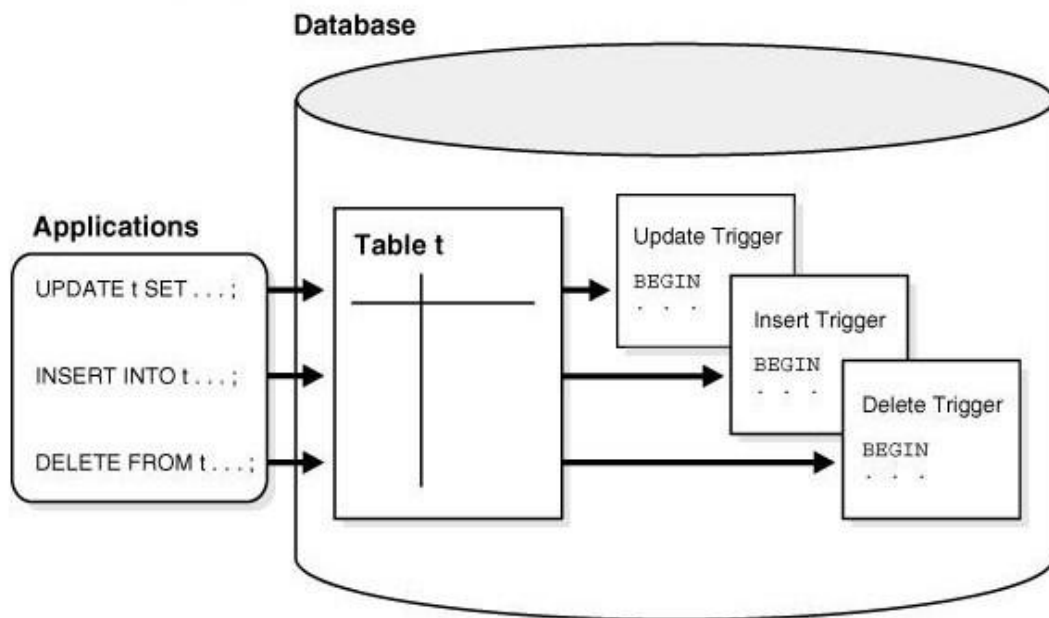
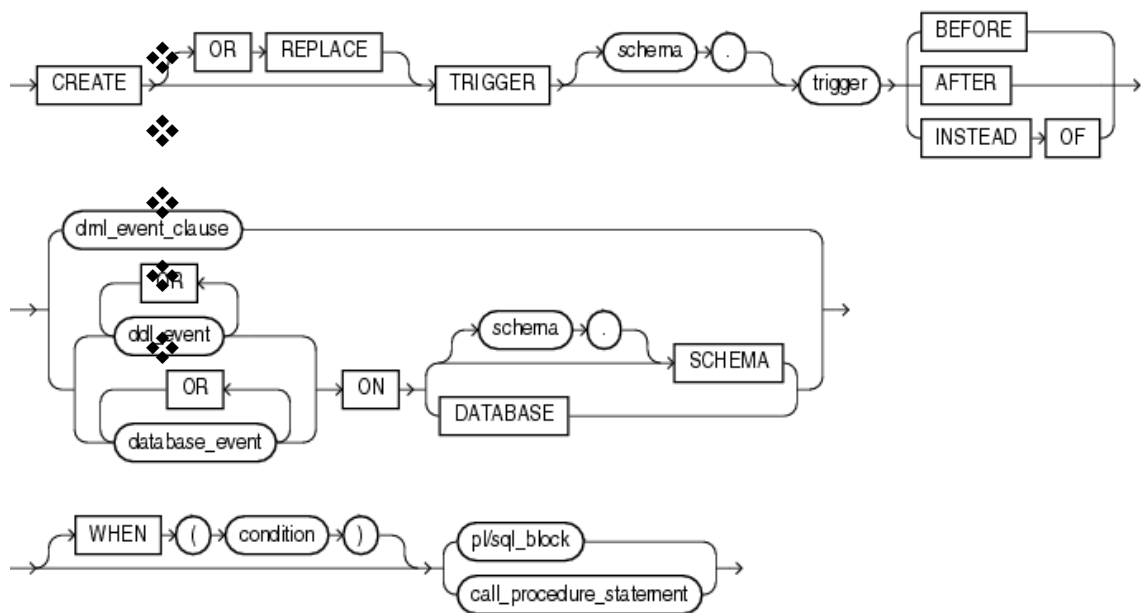


Fig: Triggers

Trigger Structure:



Sample Code:

```
create trigger overdraft-trigger after update on account
referencing new row as nrow
for each row
when nrow.balance < 0
begin atomic
    insert into borrower
        (select customer-name, account-number
         from depositor
         where nrow.account-number =
              depositor.account-number);
    insert into loan values
        (n.row.account-number, nrow.branch-name,
         - nrow.balance);

    update account set balance = 0
        where account.account-number = nrow.account-number
end
```

Source: [25]

Applying Triggers in our Database:

The trigger we may apply in our database is similar for all tables. So trigger applied on Room Table and Admission Table can be given as an example:

Create or replace trigger Admission after insert on Room

for each row

begin

insert into Admission

(select * from Admission

```
where Admission.Room_no=:new.Room_no);
```

```
end;
```

Source: [25]

Difficulties:

- ✓ The triggers are created but when we insert values it does not work properly.
- ✓ This is left as a part of future work.

Summary

We can say that Data Grid view is very essential in .NET Framework. We can do a lot of things easily and efficiently using Data Grid view. Though the coding is not so easy but it will help us to make user friendly software. On the other hand trigger is a very essential approach in database. We can make a database for functional and efficient using Triggers.

REFERENCES

1. <http://www.blurtit.com/q959542.html>
2. <http://en.wikipedia.org/wiki/Database#Applications>
3. http://www.cl500.net/pros_cons.html
4. <http://en.wikipedia.org/wiki/Entity>
5. http://en.wikipedia.org/wiki/Attribute_%28computing%29
6. http://en.wikipedia.org/wiki/Data_type
7. Prof. Dorothee Koch, Lecture Notes : HfT Stuttgart-Normalization.e.fm
8. Source: SOFTWARE ENGINEERING A Practical Approach 6th Edition
McGRAW-HILL INTERNATIONAL EDITION by ROGER S. PRESSMAN
(Chapter-26)
9. <http://ergo.human.cornell.edu/ahtutorials/interface.html>
10. Jones, M. K. (1989). Human-computer interaction: A design guide. Englewood Cliffs, NJ: Educational Technology Publications.
11. Nicol, A. (1990). Interfaces for learning: What do good teachers know that don't? In B. Laurel (Ed.), the art of human-computer interface design. (pp. 113-123). Maidenhead Birkshire: Pergammon Infotech Limited.
12. Reingold, H. (1990). An interview with Don Norman. In B. Laurel (Ed.), the art of human-computer interface design. (pp. 113-123). Maidenhead Birkshire: Pergammon Infotech Limited.

13. Laurel, B. (1991). *Computer as theatre*. Menlo Park, CA: Addison Wesley;
Laurel, B. (Ed.). (1991). *the art of human-computer interface design*. Menlo Park, CA: Addison Wesley.
14. Laurel, B, Oren, T., & Don, A. (1992). Issues in multimedia design: Media integration and interface agents. In M. M. Blattner & R. B. Dannenberg (Eds.), *Multimedia interface design*. (pp. 53-64), ACM Press.
15. Jones, M.G. (1993). *Guidelines for screen design and user-interface design in computer-based learning environments*. (Doctoral Dissertation, The University of Georgia, 1993). *Dissertation Abstracts International*, 54 (9), 308a - 309a.
16. Presentation Slide (Judical Database System) by Kipp Scott and Michael Sinks
17. http://en.wikipedia.org/wiki/Front_and_back_ends
18. <http://en.wikipedia.org/wiki/Security>
19. http://en.wikipedia.org/wiki/Computer_Security
20. http://en.wikipedia.org/wiki/Database_Security
21. <http://en.wikipedia.org/wiki/dba>
22. http://en.wikipedia.org/wiki/Dynamic-link_library
23. <http://msdn.microsoft.com/enus/library/system.windows.forms.datagridview.aspx>
24. http://en.wikipedia.org/wiki/Database_trigger
25. http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/triggers.htm#CNCPT017

APPENDIX

Login Form:

Code for Login Form:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using WindowsFormsbirdem.UI;
namespace WindowsFormsbirdem
{
    public partial class LOGIN : Form
    {
        public void dis()
        {
            LOGIN l = new LOGIN();
            l.WindowState = FormWindowState.Maximized;
        }
        public enum Designation
        {
            Administrator,
            Accountant,
            Receptionist
        }

        public LOGIN()
        {
            InitializeComponent();
            this.DesignationCombo.DataSource =
Enum.GetNames(typeof(Designation));
        }

        private void btAddNew_Click(object sender, EventArgs e)
        {
            int password = Convert.ToInt32(passwordtext.Text);

            if (DesignationText.Text == "Administrator" && password ==
62413)
            {
                MessageBox.Show("Password Matched");
                frmMenubar f = new frmMenubar();
                f.Show();
            }
        }
    }
}
```

```

62436)         else if (DesignationText.Text == "Accountant" && password ==
{
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();

}

62444)         else if (DesignationText.Text == "Receptionist" && password ==
{
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();

}

        else
        {
            MessageBox.Show("Invalid Password");
        }

        //this.Close();
    }

    private void DesignationCombo_SelectedIndexChanged(object sender,
EventArgs e)
    {
        Designation des = (Designation)Enum.Parse(typeof(Designation),
DesignationCombo.Text);
        switch (des)
        {
            case Designation.Administrator:
                DesignationText.Text = "Administrator";
                break;

            case Designation.Accountant:
                DesignationText.Text = "Accountant";
                break;

            case Designation.Receptionist:
                DesignationText.Text = "Receptionist";
                break;

        }
    }
}

```

Form Menu:

Codes for the Form Menu:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Forms;

namespace WindowsFormsbirdem.UI
{
    public partial class frmMenubar : Form
    {
        private int childFormNumber = 0;

        public frmMenubar()
        {
            InitializeComponent();
        }

        private void ShowNewForm(object sender, EventArgs e)
        {
            Form childForm = new Form();
            childForm.MdiParent = this;
            childForm.Text = "Window " + childFormNumber++;
            childForm.Show();
        }

        private void OpenFile(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            openFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files
(*.*)|*.*";
            if (openFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = openFileDialog.FileName;
            }
        }

        private void SaveAsToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            saveFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files
(*.*)|*.*";
            if (saveFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = saveFileDialog.FileName;
            }
        }
    }
}
```

```

        }
    }

    private void ExitToolsStripMenuItem_Click(object sender, EventArgs
e)
    {
        this.Close();
    }

    private void ToolBarToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        toolStrip.Visible = toolBarToolStripMenuItem.Checked;
    }

    private void StatusBarToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        statusStrip.Visible = statusBarToolStripMenuItem.Checked;
    }

    private void CascadeToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        LayoutMdi (MdiLayout.Cascade);
    }

    private void TileVerticalToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        LayoutMdi (MdiLayout.TileVertical);
    }

    private void TileHorizontalToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        LayoutMdi (MdiLayout.TileHorizontal);
    }

    private void ArrangeIconsToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        LayoutMdi (MdiLayout.ArrangeIcons);
    }

    private void CloseAllToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        CloseAllChildForm();
    }

    private void CloseAllChildForm()
    {
        foreach (Form childForm in MdiChildren)
        {
            childForm.Close();
        }
    }

    private void aboutToolStripMenuItem_Click(object sender, EventArgs
e)
    {

```

```

        Show(new frmAboutDeveloper());
    }

e) private void roomToolStripMenuItem_Click(object sender, EventArgs
    {
        Show(new frmRoom());
    }

    private void Show(Form frm)
    {
        CloseAllChildForm();
        frm.WindowState = FormWindowState.Maximized;

        frm.MdiParent = this;
        frm.Show();
    }

e) private void roomToolStripMenuItem1_Click(object sender, EventArgs
    {
        Show(new Search_info());
    }

e) private void billToolStripMenuItem_Click(object sender, EventArgs
    {
        Show(new frmBill());
    }

e) private void driverToolStripMenuItem_Click(object sender, EventArgs
    {
        Driver d = new Driver();
        d.Show();
    }

    private void frmMenubar_Load(object sender, EventArgs e)
    {
        BackColor = Color.Lavender;

        Show(new Form1(this.menuStrip, this.toolStrip));
        BackColor = Color.Lavender;
    }

    private void accountantToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmAccountant());
    }

    private void receptionistToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmReceptionist());
    }

    private void ambulanceToolStripMenuItem_Click(object sender,
EventArgs e)
    {

```

```

        Show(new frmAmbulance());
    }

    private void carriersToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmCarriers());
    }

    private void nurseToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        Show(new frmNurse());
    }

    private void wardboyToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmWardboy());
    }

    private void doctorToolStripMenuItem1_Click(object sender,
EventArgs e)
    {
        Show(new frmDoctor());
    }

    private void oTToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Show(new frmOT());
    }

    private void medicineToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmMedicine());
    }

    private void departmentToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmDepartment());
    }

    private void testToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        Show(new frmTest());
    }

    private void patientToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        Show(new frmPatient());
    }

    private void doctorToolStripMenuItem_Click(object sender, EventArgs
e)
    {

```

```

        private void dFDToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Show(new frmDFD());
        }
    }
}

```

Room Form:

Codes for the Room Form:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using WindowsFormsbirdem.DAL;

namespace WindowsFormsbirdem.UI
{
    public partial class frmRoom : Form
    {
        public static RoomDataObject roomInstant = new RoomDataObject();
        public frmRoom()
        {
            InitializeComponent();
            dataGridView1.DataSource = roomInstant.GetAllData();
        }
        private void toolStripButton4_Click(object sender, EventArgs e)
        {
            ClearTextBox();
            ButtonControl(false);
        }

        private void dataGridView1_MouseDoubleClick(object sender,
MouseEventArgs e)
        {
            textBox1.Text =
dataGridView1.Rows[dataGridView1.SelectedCells[0].RowIndex].Cells[0].Value.
ToString();
            textBox2.Text =
dataGridView1.Rows[dataGridView1.SelectedCells[0].RowIndex].Cells[1].Value.
ToString();
            textBox3.Text =
dataGridView1.Rows[dataGridView1.SelectedCells[0].RowIndex].Cells[2].Value.
ToString();
            textBox4.Text =
dataGridView1.Rows[dataGridView1.SelectedCells[0].RowIndex].Cells[3].Value.
ToString();
            ButtonControl(false);
        }
        private bool Validation()
        {
            if (textBox1.Text == "")

```



```

        {
            MASICEIU.MessageShow.Information("Select Item from room
list.");
            return false;
        }
        else if (textBox2.Text == "")
        {
            MASICEIU.MessageShow.Information("Room No");
            textBox2.Focus();
            return false;
        }
        else if (textBox3.Text == "")
        {
            MASICEIU.MessageShow.Information("Room Type");
            textBox3.Focus();
            return false;
        }
        else if (textBox4.Text == "")
        {
            MASICEIU.MessageShow.Information("Room Cost");
            textBox4.Focus();
            return false;
        }
        return true;
    }

    private void ButtonControl(bool boolValue)
    {
        btSave.Enabled = boolValue;
        btUpdate.Enabled = !boolValue;
        btDelete.Enabled = !boolValue;
        btAddNew.Enabled = !boolValue;
    }

    private void ClearTextBox()
    {
        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
    }

    private void SetRoomInstant()
    {
        roomInstant.Room_id1 = Convert.ToInt16(textBox1.Text);
        roomInstant.Room_no1 = textBox2.Text;
        roomInstant.Room_type1 = textBox3.Text;
        roomInstant.Room_cost1 = Convert.ToInt16(textBox4.Text);
    }

    private void toolStripButton5_Click(object sender, EventArgs e)
    {
        textBox1.Text = new RoomDataAccess().NextID("Select
max(Room_id)from tbl_Room").ToString();
        ButtonControl(true);
    }

    private void toolStripButton1_Click(object sender, EventArgs e)
    {
        if (!Validation()) return;
        SetRoomInstant();
        roomInstant.Save();
        dataGridView1.DataSource = roomInstant.GetAllData();
        ClearTextBox();
        ButtonControl(false);
    }

```

```

    }

    private void toolStripButton2_Click(object sender, EventArgs e)
    {
        if (!Validation()) return;
        SetRoomInstant();
        roomInstant.Update();
        dataGridView1.DataSource = roomInstant.GetAllData();
        ClearTextBox();
    }

    private void toolStripButton3_Click(object sender, EventArgs e)
    {
        if (!Validation()) return;
        SetRoomInstant();
        roomInstant.Delete();
        dataGridView1.DataSource = roomInstant.GetAllData();
        ClearTextBox();
    }

    private void frmRoom_Load(object sender, EventArgs e)
    {
        btSave.Enabled = false;
    }
}

```

Actions:

- Save, Delete, Update, Addnew buttons are controlled in this form.
- DataGridView is controlled from this form.
- Helping file RoomDataObject is called from this class.

Code of RoomDataObject.cs class

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MASICEIU.BaseDataLayer;
using WindowsFormsbirdem.DAL.DOL;

namespace WindowsFormsbirdem.DAL
{

```

```

public class RoomDataObject:DataObject
{
    RoomService service = new RoomService();
    private int Room_id;
    private string Room_no;
    private string Room_type;
    private int Room_cost;

    public int Room_id1 { get { return Room_id; } set { Room_id =
value; } }
    public string Room_no1 { get { return Room_no; } set { Room_no =
value; } }
    public string Room_type1 { get { return Room_type; } set {
Room_type = value; } }
    public int Room_cost1 { get { return Room_cost; } set { Room_cost =
value; } }

    public override List<object> GetAllData()
    {
        return service.GetAllData();
    }
    public override void Save()
    {
        service.Save(WindowsFormsbirdem.UI.frmRoom.roomInstant);
    }
    public void Update()
    {
        service.Update(WindowsFormsbirdem.UI.frmRoom.roomInstant);
    }
    public override void Delete()
    {
        service.Delete(WindowsFormsbirdem.UI.frmRoom.roomInstant.Room_id1.ToString(
));
    }
}

```

Actions:

- ✓ All Private variables are used as public variables using set and get methods.
- ✓ Helping file RoomService is called.
- ✓ Save, Delete, Update instances are called.

Code of RoomService.cs Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MASICEIU.BaseDataLayer;
using MASICEIU;
using System.Data;

namespace WindowsFormsbirdem.DAL.DOL
{
    public class RoomService : Service
    {
        RoomDataAccess dataAccess = new RoomDataAccess();

        public override List<object> GetAllData()
        {
            DataTable dt = dataAccess.GetAllData();
            return MapObject(dt);
        }
        public override List<object> MapObject(System.Data.DataTable
dataTable)
        {
            List<object> list = new List<object>();
            foreach (DataRow row in dataTable.Rows)
            {
                RoomDataObject roomDataObject = new RoomDataObject();
                roomDataObject.Room_id1 =
NullHandler.GetInt(row["Room_id"]);
                roomDataObject.Room_no1 =
NullHandler.GetString(row["Room_no"]);
                roomDataObject.Room_type1 =
NullHandler.GetString(row["Room_type"]);
                roomDataObject.Room_cost1 =
NullHandler.GetInt(row["Room_cost"]);
                list.Add(roomDataObject);
            }
            return list;
        }
        public override void Save(object objectValue)
        {
            dataAccess.Save(objectValue);
        }
        public void Update(object objectValue)
        {
            dataAccess.Update(objectValue);
        }
        public override bool Delete(string query)
        {
            dataAccess.Delete(query);
            return true;
        }
    }
}
```

Actions:

- ✓ All variable types are set.
- ✓ Helping class RoomDataAccess is called
- ✓ Save, Delete and Update are ensured.

Code of RoomDataAccess.cs Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MASICEIU.BaseDataLayer;
using System.Data;
using System.Data.SqlClient;

namespace WindowsFormsbirdem.DAL
{
    public class RoomDataAccess: DataAccess
    {
        public override DataTable GetAllData()
        {
            return ConnectionManager.DatabaseInstant.GetTable("Select *from
tbl_Room");
        }

        public override void Save(object objectValue)
        {
            RoomDataObject obj = (RoomDataObject)objectValue;
            string query = "insert into tbl_Room values(" + obj.Room_id1 +
",'" + obj.Room_no1 + "','" + obj.Room_type1 + "','" + obj.Room_cost1 + ")";
            ConnectionManager.DatabaseInstant.Insert(query);
        }
        public override int NextID(string query)
        {
            return ConnectionManager.DatabaseInstant.NextID(query);
        }
        public void Update(object objectValue)
        {
            RoomDataObject obj = (RoomDataObject)objectValue;
            string query = "Update tbl_Room set Room_no='" + obj.Room_no1 +
"',Room_type='" + obj.Room_type1 + "',Room_cost=" + obj.Room_cost1 + "
where Room_id='" + obj.Room_id1 + "'";
            ConnectionManager.DatabaseInstant.Update(query);
        }
        public override bool Delete(string query)
        {
            ConnectionManager.DatabaseInstant.Delete("delete from tbl_Room
where Room_id='" + query + "'");
            return true;
        }
    }
}
```

```
}  
}  
}
```

Actions:

- ✓ All sql queries like Save, Delete, Update are done here.
- ✓ Data from Sql Server is controlled using this class.

THE END