

---

# Chapter 1

## INTRODUCTION

### 1.1 Project overview

The purpose of developing exam hall seating arrangement system is to computerized the traditional way of conducting exams and help staffs in allocating exam hall easily without any burden. Another purpose of developing this software is to generate the report automatically during exams at the end of the session or in between the session. This project also allocate particular invigilator for particular hall. It is also very useful for the college where the software may generate the hall separation and seat allocation. Hence the hall and seat is allocated to the students automatically based on their departments and register numbers

➤ The major modules in this application are

- Student details
- Invigilator details
- Room details
- Exam schedule
- Room allocation
- Branch details

### 1.2 Database management system (DBMS):

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.

A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified – and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

The DBMS perhaps most useful for providing a centralized view of data that can be accessed by multiple users, for multiple locations, in a controlled manner. A DBMS can limit what data the end user sees, as well as how that end user can view the data, providing many views of single database schema. End users can software programs are free from having to understand where the data is physically located or on what type of storage media it resides because the DBMS handles all requests.

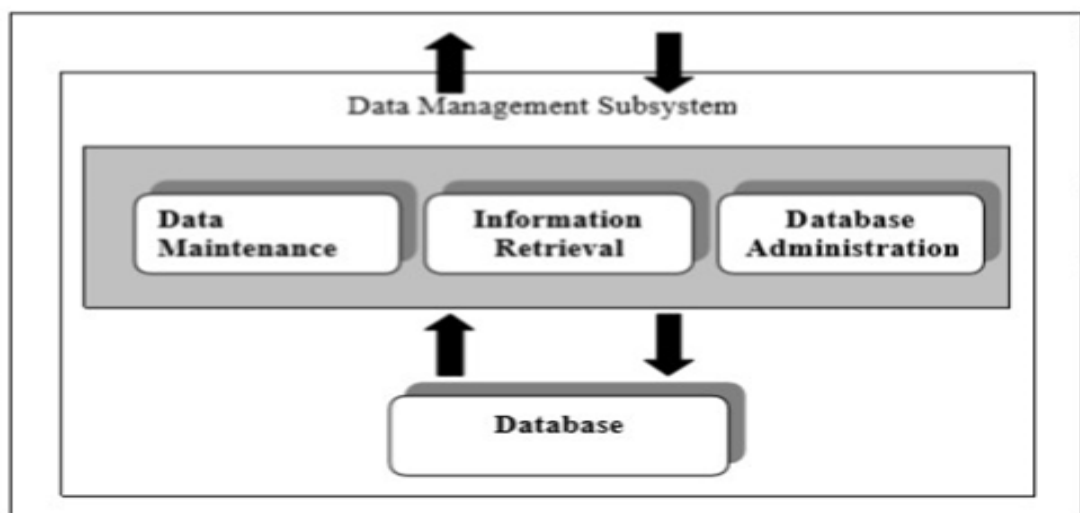


Fig 1.1 Facilities of a DBMS

### 1.3 Structured Query Language (SQL):

SQL was one of the first commercial Languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, 'A Relational Model of Data for Large Shared Data Banks'. Despite not entirely adhering to the relation model as described by Codd it becomes the most widely used database language.

SQL become a standard of the American National Standard Institute (ANSI) in 1986,

And of the International Organization for Standardization(ISO) in 1987. Since then, the standard has been revised to include a large set of features. Despite the existing of such standards, most SQL code is not completely portable among different database systems without adjustments.

## **1.4 Front end development using Net Beans:**

Front-end and Back-end are terms used to characterize program interfaces and services relative to the initial user of these interfaces and services. A “front-end” application is the one application users interact with directly. A “back-end” application serves in directly in support of the front-end services, usually by being closer to the required resource or having the capability to communicate with required resource. The back-end application may interact directly with front-end or, perhaps more typically, is a program called from an intermediate program that mediates front-end and back-end activities.

Net Beans is a software development platform written in Java. The Net Beans Platform allows application to be developed from a set of modular components called modules. The Net Beans IDE is primarily intended for development in Java, but also supports other languages, in a particular XAMPP, PHP, and C/C++ etc. Net Beans is cross platform and runs on Microsoft Windows, Mac OS, Linux and other platforms.

The entire project has been developed keeping in view of the distributed client server computing technology. The user interfaces are browser specific to give distributed accessibility for the overall system. The database connectivity was planned using the latest “XAMPP” technology provided by Apache consisting mainly of the Apache HTTP Server, and interpreters for scripts written in PHP and Perl programming languages. The authentication and authorization was cross checked at all the relevant stages.

## **1.5 Three Schema Architecture:**

Following are the three levels of database architecture:

- Physical Level
- Conceptual Level
- External Level

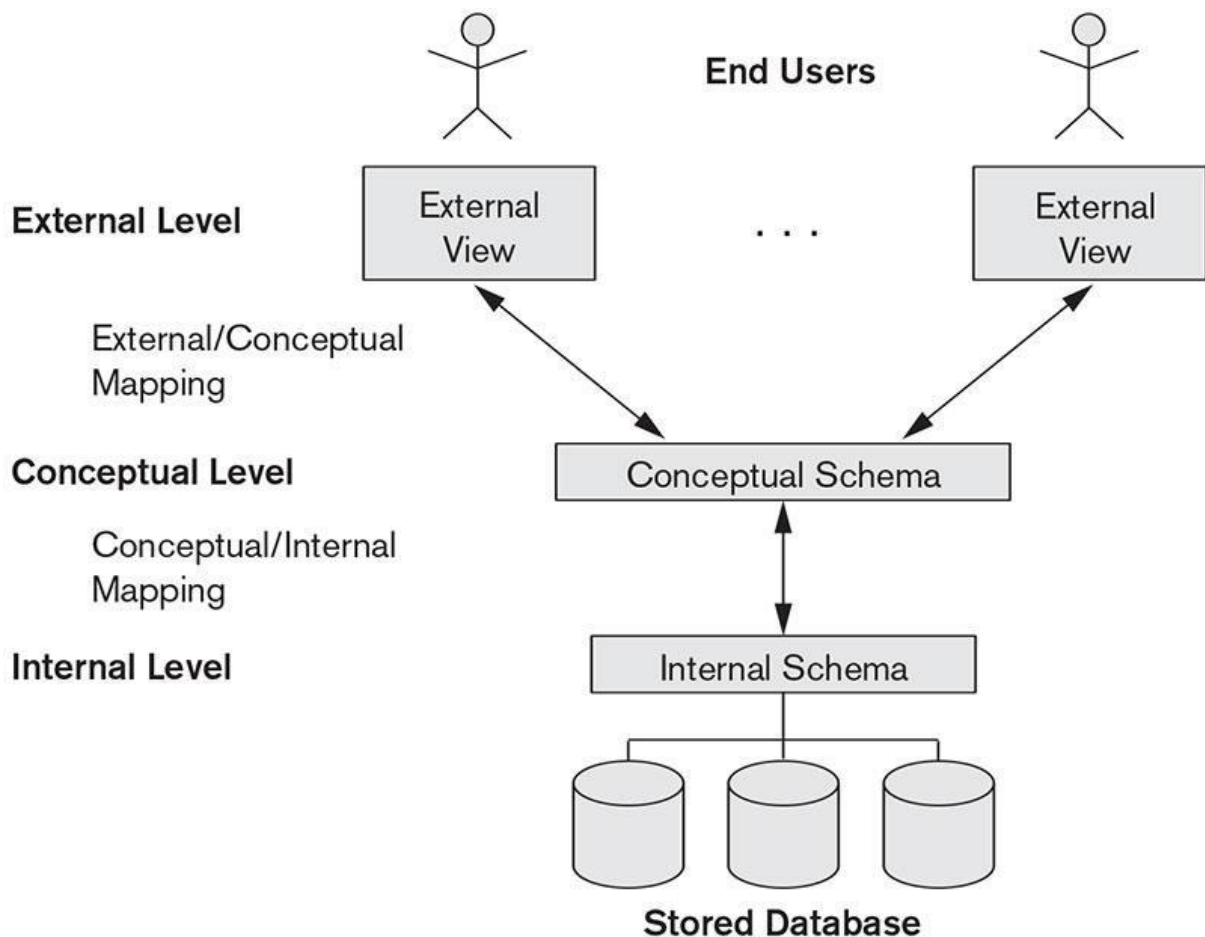


Fig 1.2 The Three Schema Architecture

In the above diagram,

- It shows the architecture of DBMS.
- Mapping is the process of transforming request response between various database levels of architecture.
- Mapping is not good for small database, because it takes more times.
- In External/Conceptual mapping, DBMS transforms a request on an external schema against the conceptual schema.
- In Conceptual/Internal mapping, it is necessary to transform the request form the conceptual to internal levels.

### 1. Physical Level:

- Physical level describes the physical storage structure of data in database.
- It is also known as Internal Level.

- This level is very close to physical storage of data.
- At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device.
- At highest level, it can be viewed in the form of files.
- The internal schema defines the various stored data types. It uses a physical data model.

## **2. Conceptual Level:**

- Conceptual level describes the structure of the whole database for a group of users.
- It is also known as the data model.
- Conceptual schema is a representation of the entire content of the database.
- This schema contain all the information to build relevant external records.
- It hides the internal details of physical storage.

## **3. External Level:**

- External level is related to the data which is viewed by individual end users.
- This level includes a no. of user views or external schemas.
- This level is closest to the user.
- External view describe the segment of the database that is required for a particular user group and hides the rest of the database from that user group.

## **1.6 NORMALIZATION:**

Here are the most commonly used normal forms:

### **1). First normal form (1NF):**

- As per the rule of first normal form, an attribute(column) of a table cannot hold multiple values. It should hold only atomic values.

**2). Second normal form (2NF):**

- A table is said to be in 2NF if both the following conditions hold:
- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.
- An attribute that is not part of any candidate key is known as non-prime attribute.

**3). Third normal form (3NF):**

- A table design is said to be in 3NF if both the following conditions hold:
- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed
- An attribute that is not part of any candidate key is known as non-prime attribute.
- In other words 3NF can be explained like this : A table is in 3NF if it is in 2NF and for each functional dependency  $X \rightarrow Y$  at least one of the following condition hold:
- X is a super key of table
- Y is a prime attribute of table
- An attribute that is a part of one of the candidate keys is known as prime attribute.

**Fourth Normal Form(4NF):** is a normal form used in normalization. Introduced by Ronald Fagin in 1997, 4NF is the next level of normalization after Boyce-Codd normal form (BCNF). Whereas the second, third, and Boyce-Codd normal forms are concerned with functional dependencies, 4NF is concerned with a more general type of dependency known as multivalued dependency. A table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies  $X \twoheadrightarrow Y$ , X is a super-key that is, X is either a candidate key or a super set.

**Fifth Normal Form(5NF) :** A database is said to be decomposes in 5<sup>th</sup> Normal Form, if and only if, it is in 4<sup>th</sup> normal form and if the table further to eliminate redundancy and

anomaly and when we join the decomposed tables by means of candidate keys we should not be losing the original data or any new record set should not arise. In simple words joining two or more decomposed tables should not lose records or new records.

**Boyce and Codd normal form (BCNF):**

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency  $X \rightarrow Y$ ,  $X$  should be the super key of the table.

---

## Chapter 2

### REQUIREMENT SPECIFICATION

#### 2.1 Software Requirement:

- OPERATING SYSTEM : Windows 10
- FRONT END : Java / Net beans
- BACK END : My SQL
- DATABASE : MY SQL Server/ Xampp
- SOFTWARE DEVELOPMENT KIT : Java JDK
- BROWSER : Google Chrome

#### 2.2 Hardware Requirement:

- Intel P4 1.5 GHz Processor or Above
- RAM 512MB and Above
- HDD 20 GB Hard Disk Space or Above

#### 2.3 About Connectivity:

##### Creating a Database Using MySQL:

- Let create a MySQL database using MySQL Command Line Client.
- Provide the password for root user login.
- Create a database named examseat and then exit.

Ex: CREATE DATABASE examseat ;

##### Making a Connection Using Net Beans:

- Open Net Beans IDE. In the Services pane, expand Database node > expand the Drivers node > select MySQL (Connector/J driver) > right click mouse > select Connect Using. ☐ Fill in the database URL as shown below and the root password. Click OK.



- The following Figure shows that the connection to the examseat MySQL database was established.
- You can re-confirm the connection from the following Figure.

**Create a new project in the IDE:**

- Start Net Beans IDE and choose File > New Project (Ctrl-Shift-N) from the main menu.

Under Categories select Web; under Projects select Web Application. Click Next.

- From the Server drop-down list, select the server you plan work with. Leave all other settings at their defaults and click Finish. The IDE creates a project template for the entire application, and opens an empty JSP page (index.jsp) in the Source Editor. index.jsp serves as the entry point for the application. The new project is structured according to Sun Java Blue Prints guidelines.

**Preparing the Interface:**

Begin by preparing a simple interface for the two pages. Both index.jsp and response.jsp implement an HTML table to display data in a structured fashion. index.jsp also requires an HTML form that includes a drop-down list.

---

## CHAPTER 3

### SYSTEM DESIGN AND ANALYSIS

#### 3.1 ER-to- Relational Mapping Algorithms:

##### Step 1: Mapping of Regular Entity Types

For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of E as the primary key for R. If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R.

##### Step 2: Mapping of Weak Entity Types

For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes(or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s). The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

##### Step 3: Mapping of Binary 1:1 Relationship Type

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

##### Step 4: Mapping of Binary 1: N Relationship Types

For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represent the other entity type participating in R. Include any simple attributes of the 1:N relationship type as attributes of S.

**Step 5: Mapping of Binary M: N Relationship Types**

For each regular binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attribute in S the primary key of the relations that represent the participating entity type; their combination will form the primary key of S. Also Include any simple attributes of the M:N relationship type as attributes of S.

**Step 6: Mapping of Multi valued attributes**

For each multi valued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R- of the relation that represents the entity type or relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is a composite, we include its simple component.

**Step 7: Mapping of N-array Relationship Types**

For each n-array relationship type R, where  $n \geq 2$ , create a new relation S to represent R. Include as foreign key attribute in S the primary key of the relations that represent the participating entity types. So Include any simple attribute of the n-ary relationship type as attribute of S.

### 3.2 ER Diagram:

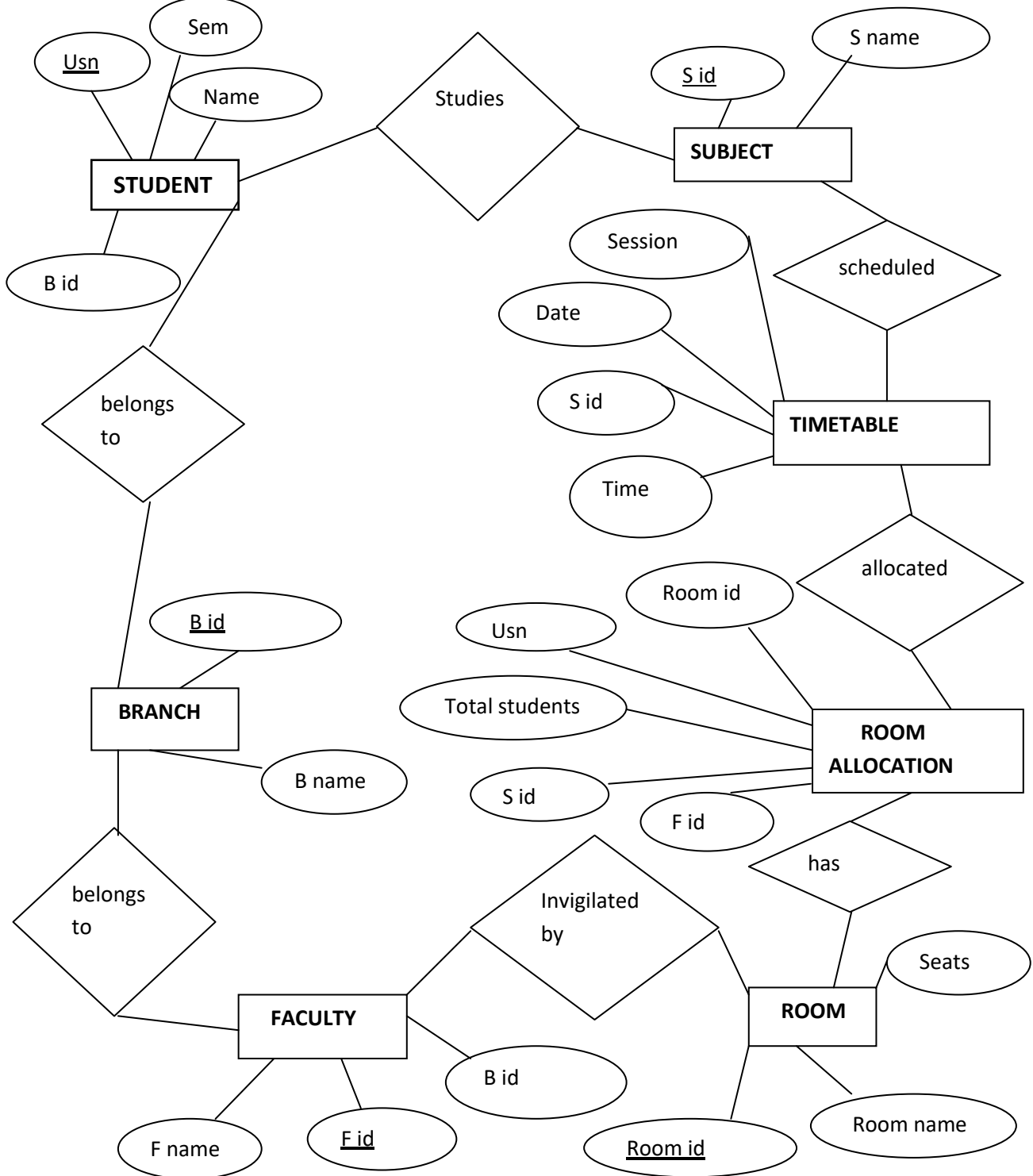


Fig.3.1 ER diagram of Exam Seat Arrangement System

### 3.3 Schema Diagram:

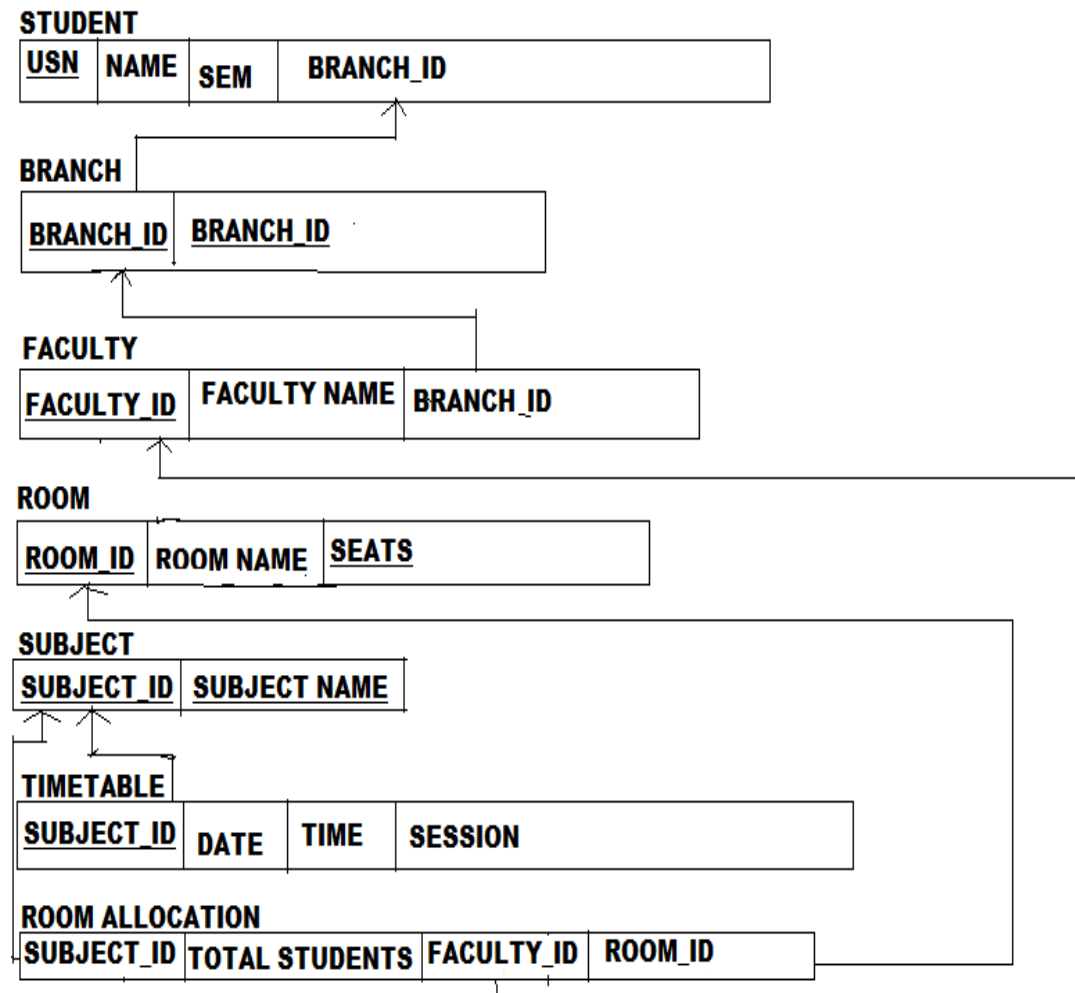


Fig.3.2 Schema diagram of Exam Seat Arrangement System

---

## Chapter 4

### IMPLEMENTATION

A database management system handles the requests generated from the SQL interface, producing or modifying data in response to these requests. This involves a multilevel processing system. level structure processes the SQL submitted by the user or application

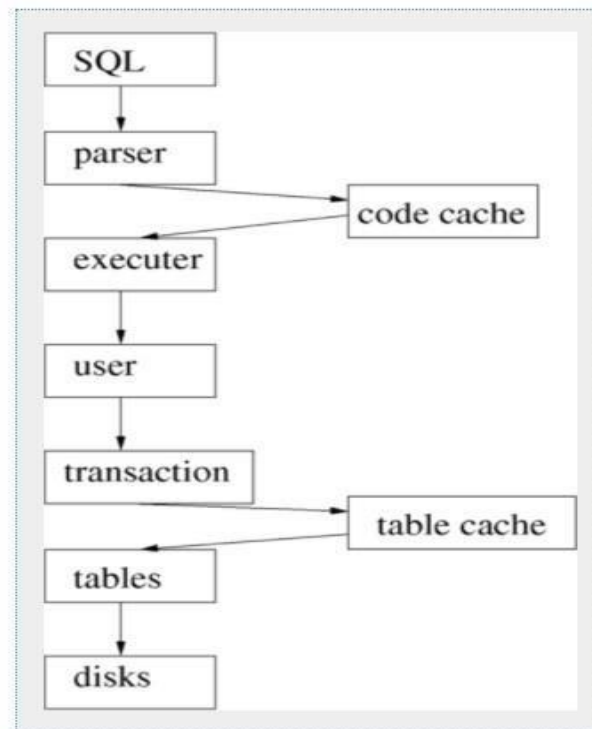


Fig .4.1 DBMS Execution and Parsing

**Parser:** The SQL must be parsed and tokenized. Syntax errors are reported back to the user. Parsing can be time consuming, so good quality DBMS implementations cache queries after they have been parsed so that if the same query is submitted again the cache copy can be used instead. To make the best use of this most systems use placeholders in queries, like :

**Executer:** This takes the SQL tokens and basically translate it into relational algebra. Each relational algebra fragment is optimized, and the passed down the levels to be acted on.

**User:** The concept of the user is required at this stage. This gives the query context, and also allows security to be implemented on a per-user basis.

**Transaction:** The queries are executed in the transaction model. The same query from the same user can be executing multiple times in different transactions. Each transaction is quite separate.

**Tables:** The idea of the table structure is controlled at low level.

**Table cache:** Disks are slow, yet a disk is the best way of storing long-term data. Memory is much faster, so it makes sense to keep as much table information as possible in memory. The disk remains synchronized to memory as part of the transaction control system.

**Disks:** Underlying almost all the database system is the disk storage system. This provides storage for the DBMS system tables, user information, schema definition, and the user data itself. It also provides the means for transaction logging.

## 4.1 Create table:

BRANCH :

```
SQL> Create table Branch(b_id varchar(10),branch_name varchar(10),constraint pk1  
primary key(b_id));
```

STUDENT:

```
SQL> Create table Student(usn varchar(15),name varchar(20),sem varchar(10),branch_id  
varchar(10),constraint pk2 primary key(usn),constraint fk1 foreign key(b_id) references  
branch(branch_id)on delete cascade);
```

ROOM:

```
SQL> Create table Room(r_id varchar(10),room_name varchar(10),seats varchar(10),  
constraint pk4 primary key(r_id));
```

FACULTY:

```
SQL> Create table Faculty(f_id varchar(05),fname varchar(10),b_id varchar(05),constraint  
pk3 primary key(f_id),constraint fk2 foreign key(b_id) references Branch(b_id) on delete  
cascade);
```

**SUBJECT:**

```
SQL> Create table Subject(s_id varchar(10),subject_name varchar(10),constraint pk5  
primary key(s_id));
```

**TIMETABLE:**

```
SQL> Create table TimeTable(s_id varchar(10),date date,time varchar(10),session  
varchar(20),constraint fk3 foreign key(s_id)references Subject(s_id)on delete cascade);
```

**ROOM ALLOCATION:**

```
SQL> Create table RoomAllocation(s_id varchar(10),totalstud numeric(10),r_id  
varchar(10),f_id varchar(05),usn varchar(15),constraint fk4 foreign key(r_id) references  
Room(r_id)on delete cascade,constraint fk5 foreign key(f_id) references Faculty(f_id)on  
delete cascade,constraint fk6 foreign key(usn) references Student(usn)on delete cascade);
```

**4.2 Insert table:****BRANCH :**

```
Insert into Branch values(&b_id,&branch_name);
```

```
Insert into Branch values(10,'cs');
```

```
Insert into Branch values(12,'is');
```

```
Insert into Branch values(11,'me');
```

```
Insert into Branch values(14,'ec');
```

**STUDENT:**

```
Insert into student values(&usn,&name,&sem,&b_id);
```

```
Insert into student values('1vk17cs001',abhay,5,10);
```

```
Insert into student values('1vk17is012','akshaya',4,12);
```

```
Insert into student values('1vk17me003','akash',4,11);
```

```
Insert into student values('1vk17ec001','bharath',4,14);
```

**FACULTY:**

```
Insert into Faculty values(&f_id,&fname,b_id);
```

```
Insert into Faculty values(11,'amrutha',10);
```

```
Insert into Faculty values(12,'ram',11);
```

```
Insert into Faculty values(13,'nandini',12);
```



Insert into Faculty values(14,'manya',14);

**ROOM:**

Insert into Room values(&r\_id,&room\_name,&seats);

Insert into Room values(111,'exam hall-1',50);

Insert into Room values(112,'exam hall-2',50);

Insert into Room values(113,'exam hall-3',50);

Insert into Room values(114,'exam hall-4',50);

**SUBJECT:**

Insert into Subject values(&s\_id,&subject\_name);

Insert into Subject values('cs01','mp');

Insert into Subject values('cs02','daa');

Insert into Subject values('cs10','dbms');

Insert into Subject values('cs12','me');

**TIMETABLE:**

Insert into TimeTable values(&s\_id,&date,&time,&session);

Insert into TimeTable values('cs01','09-dec-2019',9:30,'morning');

Insert into TimeTable values('cs02','14-dec-2019',8:30,'morning');

Insert into TimeTable values('cs10','15-dec-2019',9:30,'morning');

Insert into TimeTable values('cs12','18-dec-2019',9:30,'morning');

**ROOM ALLOCATION:**

Insert into RoomAllocation values(&s\_id,&totalstud,&r\_id,&f\_id,&usn);

Insert into RoomAllocation values('cs01',30,111,11,'1vk17cs001');

Insert into RoomAllocation values('cs02',30,112,12,'1vk17is012');

Insert into RoomAllocation values('cs10',30,113,13,'1vk17me003');

Insert into RoomAllocation values('cs12',30,114,14,'1vk17ec001');

## 4.4 Source Code:

### Code for LOGIN:

```
private void loginbuttonActionPerformed(java.awt.event.ActionEvent evt)

{

String PASSWORD=passwordtextfield.getText();

String USERNAME=usernametextfield.getText();

if(PASSWORD.contains("123")&& USERNAME.contains("exam"))

{

usernametextfield.setText("");

passwordtextfield.setText("");

close();

mainscreen m= new mainscreen();

m.setVisible(true);

}

else{

JOptionPane.showMessageDialog(null,"This password is wrong!\n click ok and try again.", "wrong pass",JOptionPane.ERROR_MESSAGE);

passwordtextfield.setText("");

usernametextfield.setText("");

}

}
```

**Code for INSERT:**

```
private void addActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String u,n,s,b;  
  
    u=txtusn.getText();  
  
    n=txtname.getText();  
  
    s=txtsem.getText();  
  
    b=txtbid1.getText();  
  
    try{  
  
        Class.forName("com.mysql.jdbc.Driver");  
  
        Connection  
        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/examseat","root","");  
  
        Statement st=con.createStatement();  
  
        String q;  
  
        q="insert into student values('"+u+"','"+n+"','"+s+"','"+b+"')";  
  
        st.executeUpdate(q);  
  
        JOptionPane.showMessageDialog(null,"Student added");  
  
    }  
  
    catch(Exception e)  
  
    {  
  
        JOptionPane.showMessageDialog(null,e);  
  
    }  
}
```

**Code for DELETE:**

```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {  
  
    try  
  
    {  
  
        Class.forName("com.mysql.jdbc.Driver");  
  
        Connection  
        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/examseat","root","");  
  
        Statement st=con.createStatement();  
  
        String q,u;  
  
        u=txtusn.getText();  
  
        q="delete from student where usn='"+u+"' ";  
  
        st.executeUpdate(q);  
  
        JOptionPane.showMessageDialog(this,"Student details deleted");  
  
    }  
  
    catch(Exception e)  
  
    {  
  
    }  
  
    }
```

**Code for VIEW:**

```
public class vstudent extends javax.swing.JFrame {  
  
    public vstudent() {  
  
        initComponents();  
  
        DisplayTable();  
  
    }
```

```
}

private void DisplayTable()

{

try{

Class.forName("com.mysql.jdbc.Driver");

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/examseat","root","");

String sql ="select *from student" ;

PreparedStatement pstmt = con.prepareStatement(sql);

ResultSet rs = pstmt.executeQuery();

t1.setModel(DbUtils.resultSetToTableModel(rs));

}

catch(Exception e){

JOptionPane.showMessageDialog(null, e);

}

}
```

**Code for UPDATE:**

```
private void updateActionPerformed(java.awt.event.ActionEvent evt) {

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/examseat","root","");
```

```
Statement st=con.createStatement();
```

```
String q,u,n,s,b;
```

```
u=txtusn.getText();
```

```
n=txtname.getText();
```

```
s=txtsem.getText();
```

```
b=txtbid1.getText();
```

```
q="update student set usn='"+u+"',name='"+n+"',sem='"+s+"',branch_id='"+b+"' where  
usn='"+u+"';";
```

```
st.executeUpdate(q);
```

```
JOptionPane.showMessageDialog(this," details updated successfully");
```

```
}
```

```
catch(Exception e)
```

```
{
```

```
}
```

```
}
```

**Code for BACK:**

```
private void txtbackActionPerformed(java.awt.event.ActionEvent evt)
```

```
{
```

```
dispose();
```

```
mainscreen m=new mainscreen();
```

```
m.setVisible(true);
```

---

## CHAPTER 5

# TESTING

### 5.1 Introduction:

The purpose of testing is to discover errors. Testing is the process of trying to every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies, and/or finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner.

### 5.2 LEVELS OF TESTING:

#### 5.2.1 UNIT TESTING:

Unity testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures and Operating procedures .For unit testing first we adopted the code testing strategy, which examined the logic of program

#### 5.2.2 USER ACCEPTANNCE TESTING:

User acceptance testing of the system is the key factor for the success of the system. A system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system at the time of development and making change whenever required. This is done with regard to the input screen design and output screen design.

#### 5.2.3 GUI Testing:

GUI testing is use to ensure the visual clarity of the system, flexibility of the system, user friendliness of the system. The various component which are to be tested are: i. Relative layout ii. Various Link and Buttons.

**5.2.4 Validation testing:**

At the culmination of black box testing, software is completely assembled is a package. Interfacing errors have uncovered and the correct and final states of tests i.e. validation is defined with a simple definition that validation succeeds when the software function in a manner that can be reasonably accepted by the customer.

**5.2.5 Output Testing:**

After performing validation testing, the next step is output testing of the proposed system. Since the system cannot be useful if it does not procedure the required output. Asking the user about the user about its required format in which the system is required tests the output displayed or generated by the system under consideration.

**Test Cases**

<b>Case Id</b>	<b>Test Case</b>	<b>Test Condition</b>	<b>Excepted Output</b>	<b>Actual Output</b>	<b>Pass / Fail</b>
1	Validation Test Case	Required Field Validation	Mandatory field should not be blank	You have to enter value in mandatory Field	Pass
		Regular Expression	A predefine format should be follow	Check proper format	Pass
		Compare Validation	Check with predefine control	Compare with control	Pass



2	Login on	Username	Username format Must be Input	Username format must be input.	Pass
		Confirmation Password	Password & confirm password match	Password & Confirm password match.	Pass
		Text field	All information must be input	All information must be input	Pass

.

## CHAPTER 6

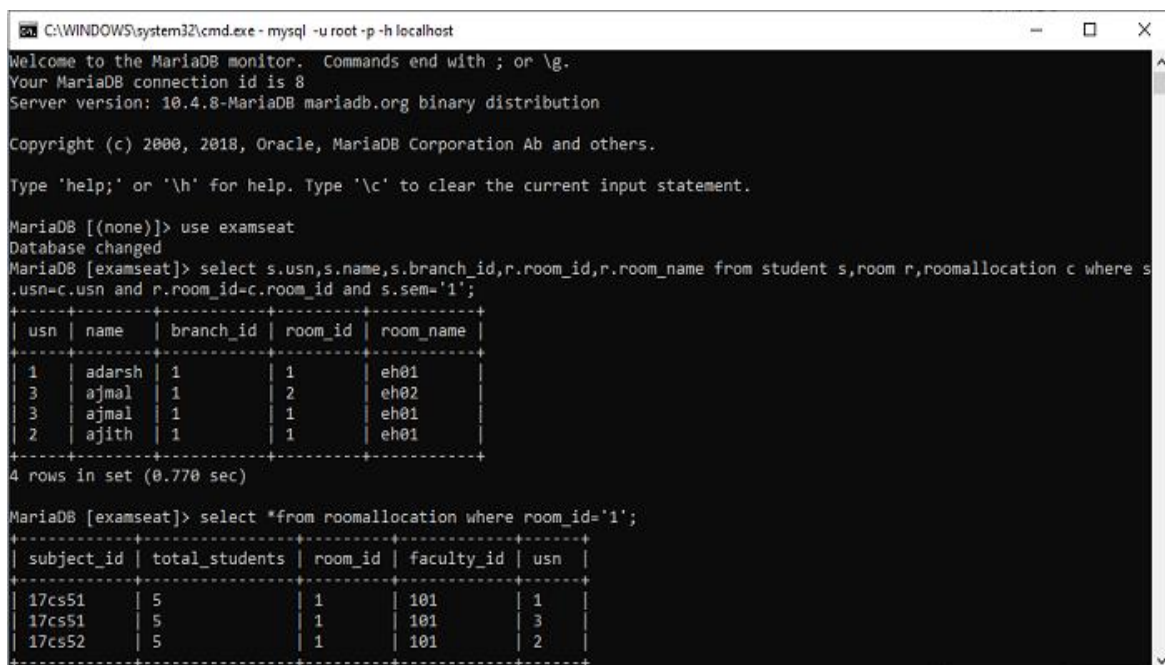
### RESULTS

A functional exam seat arrangement system was developed using NETBEANS and MYSQL as the side-server and JAVA as the client-side. The database used in the Exam Seat Arrangement System was designed with MYSQL Database Management System (DBMS). All information pertaining to the Exam Seat Arrangement System are stored in the system database. The sample outputs/allotments are shown in various snapshots.

The system is accessed by single admin. Admin can login and allot seat according to the timetable and allot invigilator for the particular room. Admin can add students, faculty, new room and subjects. Admin allocates the room in room allocation table. By using this miniproject manual work can be reduced.

#### 6.1 Queries

1. select s.usn,s.name,s.branch\_id,r.room\_id,r.room\_name from student s,room r,roomallocation c where s.usn=c.usn and r.room\_id=c.room\_id and s.sem='1';



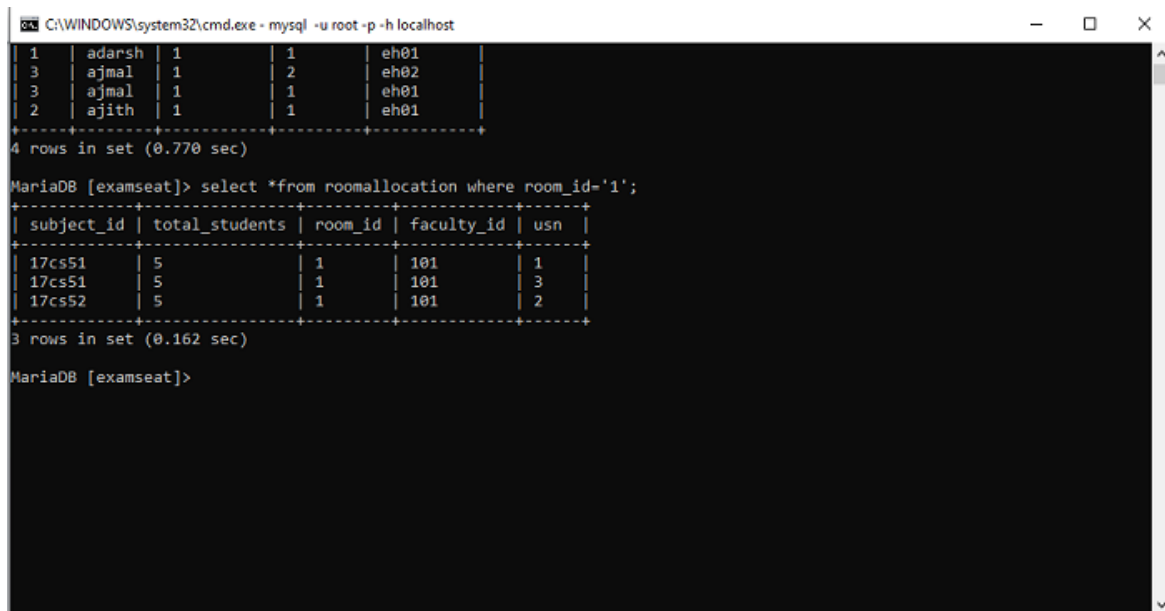
```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p -h localhost
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.8-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use examseat
Database changed
MariaDB [examseat]> select s.usn,s.name,s.branch_id,r.room_id,r.room_name from student s,room r,roomallocation c where s.usn=c.usn and r.room_id=c.room_id and s.sem='1';
+-----+-----+-----+-----+-----+
| usn | name | branch_id | room_id | room_name |
+-----+-----+-----+-----+-----+
| 1 | adarsh | 1 | 1 | eh01 |
| 3 | ajmal | 1 | 2 | eh02 |
| 3 | ajmal | 1 | 1 | eh01 |
| 2 | ajith | 1 | 1 | eh01 |
+-----+-----+-----+-----+-----+
4 rows in set (0.770 sec)

MariaDB [examseat]> select *from roomallocation where room_id='1';
+-----+-----+-----+-----+-----+
| subject_id | total_students | room_id | faculty_id | usn |
+-----+-----+-----+-----+-----+
| 17cs51 | 5 | 1 | 101 | 1 |
| 17cs51 | 5 | 1 | 101 | 3 |
| 17cs52 | 5 | 1 | 101 | 2 |
+-----+-----+-----+-----+-----+
```

Fig:6.1 Query 1

2. select \*from roomallocation where room\_id='1';



```

C:\WINDOWS\system32\cmd.exe - mysql -u root -p -h localhost
+----+-----+-----+-----+-----+
| 1 | adarsh | 1 | 1 | eh01 |
| 3 | ajmal | 1 | 2 | eh02 |
| 3 | ajmal | 1 | 1 | eh01 |
| 2 | ajith | 1 | 1 | eh01 |
+----+-----+-----+-----+
4 rows in set (0.770 sec)

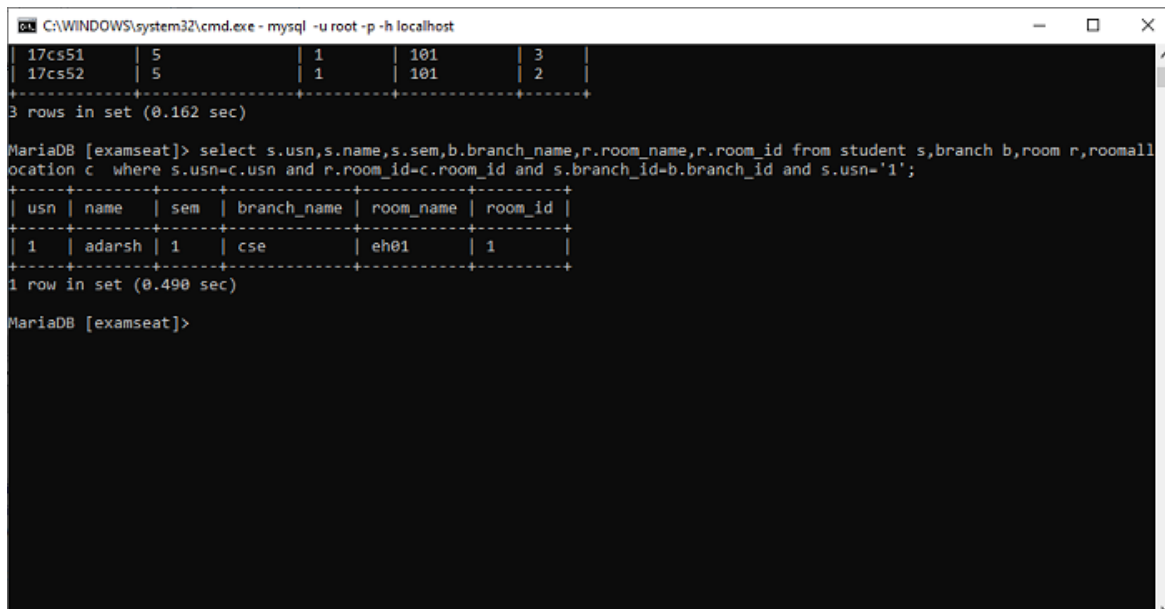
MariaDB [examseat]> select *from roomallocation where room_id='1';
+-----+-----+-----+-----+-----+
| subject_id | total_students | room_id | faculty_id | usn |
+-----+-----+-----+-----+-----+
| 17cs51 | 5 | 1 | 101 | 1 |
| 17cs51 | 5 | 1 | 101 | 3 |
| 17cs52 | 5 | 1 | 101 | 2 |
+-----+-----+-----+-----+
3 rows in set (0.162 sec)

MariaDB [examseat]>

```

Fig:6.2 Query 2

3. select s.usn,s.name,s.sem,b.branch\_name,r.room\_name,r.room\_id from student s,branch b,room r,roomallocation c where s.usn=c.usn and r.room\_id=c.room\_id and s.branch\_id=b.branch\_id and s.usn='1';



```

C:\WINDOWS\system32\cmd.exe - mysql -u root -p -h localhost
+-----+-----+-----+-----+-----+
| 17cs51 | 5 | 1 | 101 | 3 |
| 17cs52 | 5 | 1 | 101 | 2 |
+-----+-----+-----+-----+
3 rows in set (0.162 sec)

MariaDB [examseat]> select s.usn,s.name,s.sem,b.branch_name,r.room_name,r.room_id from student s,branch b,room r,roomall
ocation c where s.usn=c.usn and r.room_id=c.room_id and s.branch_id=b.branch_id and s.usn='1';
+-----+-----+-----+-----+-----+
| usn | name | sem | branch_name | room_name | room_id |
+-----+-----+-----+-----+-----+
| 1 | adarsh | 1 | cse | eh01 | 1 |
+-----+-----+-----+-----+
1 row in set (0.490 sec)

MariaDB [examseat]>

```

Fig:6.3 Query 3

---

## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **7.1 CONCLUSION.**

This project shows the details of table which is made using Netbeans 8.2 to display the details and operations like insert ,delete, update and query. It provides an easy way for user to handle the database it as it as a very good user Interface. User can directly insert ,delete,update to the table without using any SQL codes .Miniproject is used for easy way of doing operation in the database..

#### **7.2 FUTURE ENHANCEMENT**

Exam seat arrangement system is basically concerned with managing the seat allocation for examination purpose. The main objective of this is to reduce the effort of manual work for allotting seats for exam by this exam seat arrangement system manual work can be reduced.

---


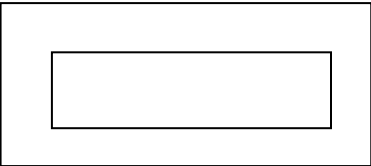
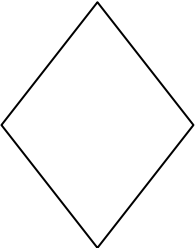
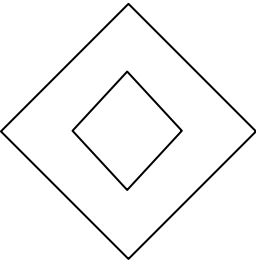
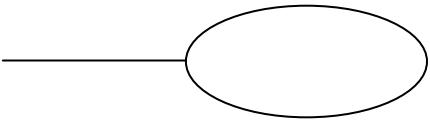
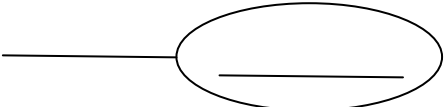
## BIBLIOGRAPHY

- [1]. Ramez Elmasri, Shamkant B Navathe “Fundamentals of Database Systems”, 7TH, Pearson Edition.
- [2]. Raghu Ramakrishna “Database Management System”.
- [3]. R. M. Menon “Expert Oracle JDBC Programming”.
- [4]. [http:// www. Google.com](http://www.Google.com)
- [5]. <http://www.stackoverflow.com>
- [6]. <http://www.javacode.com>

---

## APPENDIX A

### A. ER Diagram Notations:

Meaning	Symbol
Entity	
Weak Entity	
Relationship	
Identifying Relationship	
Attribute	
KeyAttribute	

**Fig:A Notations**

---

## APPENDIX B

### B. Snapshots

#### B.1 LOGIN PAGE

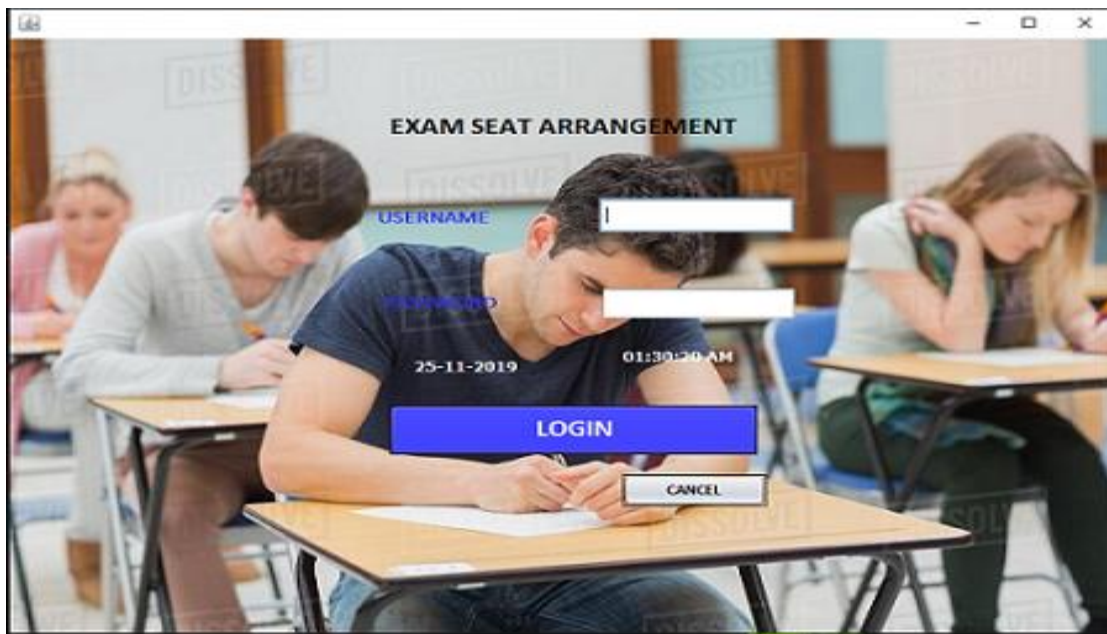


Fig:B.1 Login Page

#### B.2 HOME PAGE



Fig:B.2 Home Page



## B.3 STUDENT TABLE

A screenshot of a web application window titled "STUDENT" with a dark, patterned background and a wooden floor at the bottom. The interface includes four input fields for "USN", "NAME", "SEM", and "BRANCH ID", each with a red label. To the right of these fields is a "CLEAR" button. Below the input fields are three blue buttons: "ADD", "VIEW", and "BACK".

Fig:B.3Student table

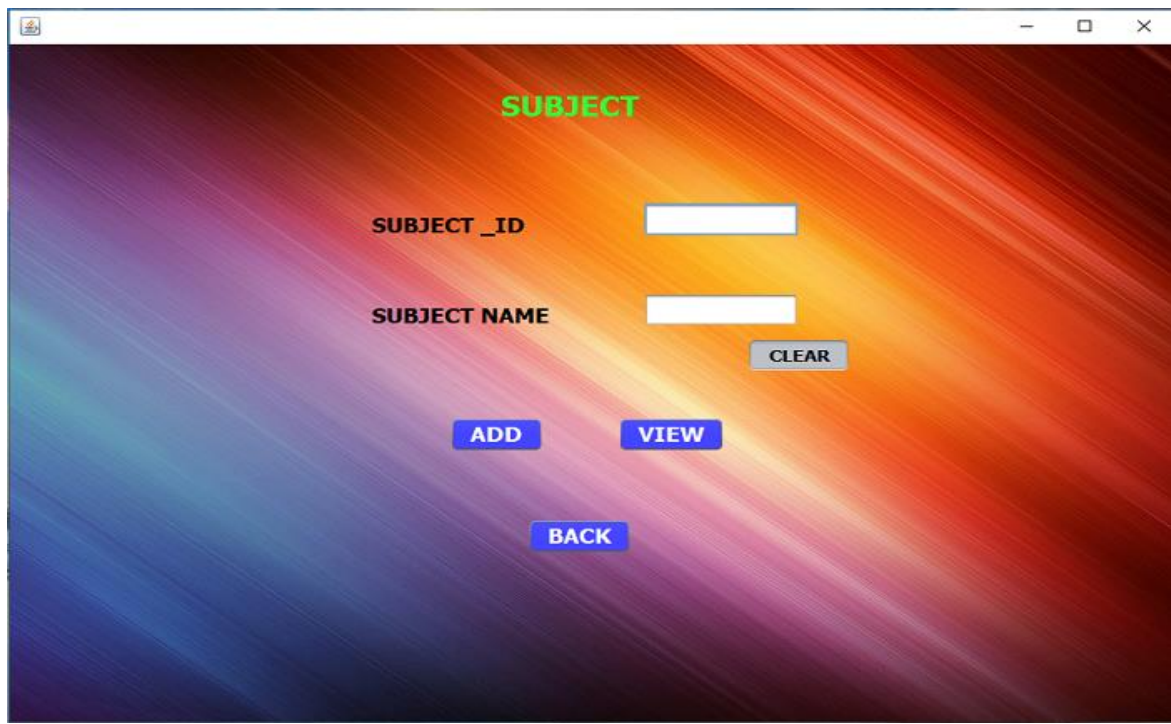
## B.4 ROOM TABLE

A screenshot of a web application window titled "ROOM" with a background image of a classroom. The interface includes three input fields for "ROOM\_ID", "ROOM\_NAME", and "SEATS", each with a cyan label. To the right of these fields is a "CLEAR" button. Below the input fields are three blue buttons: "ADD", "VIEW", and "BACK".

Fig:B.4 Room table



## B.5 SUBJECT TABLE



A screenshot of a web application window titled "SUBJECT". The window has a colorful, abstract background with diagonal streaks of purple, blue, and orange. The title "SUBJECT" is displayed in green text at the top center. Below the title, there are two input fields: "SUBJECT\_ID" and "SUBJECT NAME". To the right of the "SUBJECT NAME" field is a "CLEAR" button. Below these fields are three buttons: "ADD", "VIEW", and "BACK", arranged in a triangular pattern. The "ADD" and "VIEW" buttons are blue, while the "BACK" button is also blue and positioned below them.

Fig:B.5 Subject table

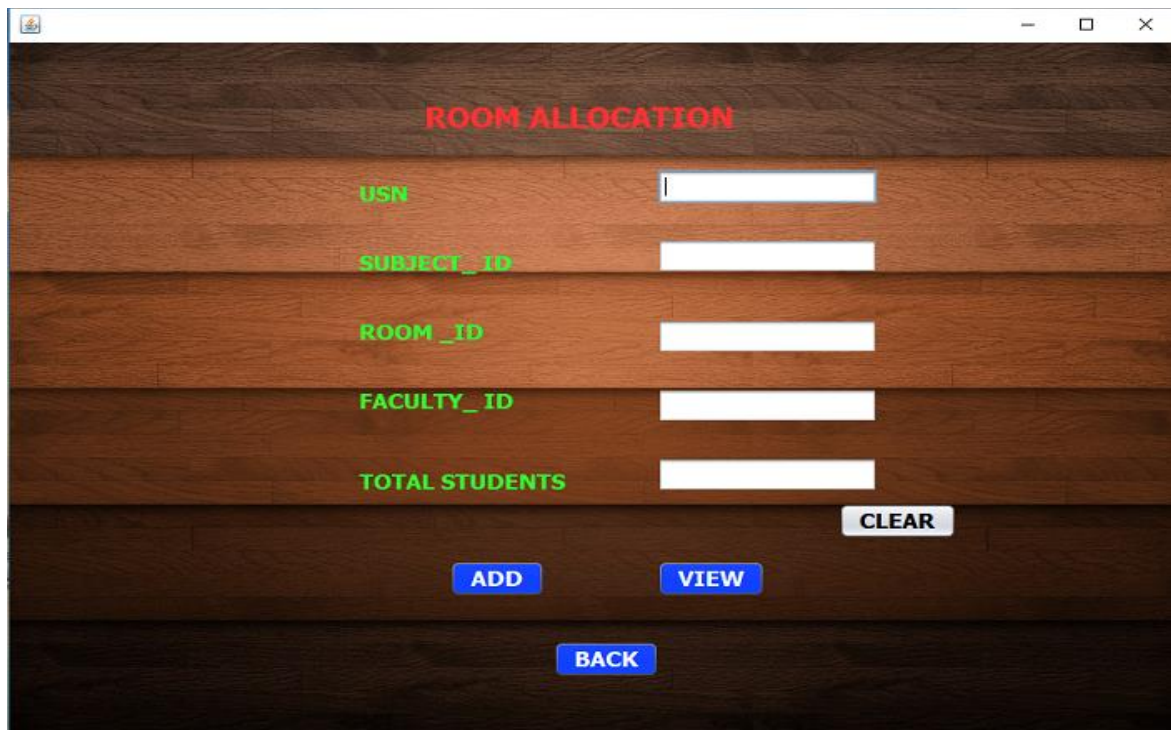
## B.6 TIMETABLE TABLE



A screenshot of a web application window titled "TIME TABLE". The window has a dark wood-grain background with warm, glowing light effects at the top. The title "TIME TABLE" is displayed in blue text at the top center. Below the title, there are four input fields: "SUBJECT\_ID", "DATE", "TIME", and "SESSION". The "DATE" field has a calendar icon to its right. The "SESSION" field has a dropdown arrow to its right. To the right of the "SESSION" field is a "CLEAR" button. Below these fields are three buttons: "ADD", "VIEW", and "BACK", arranged in a triangular pattern. The "ADD" and "VIEW" buttons are blue, while the "BACK" button is also blue and positioned below them.

Fig:B.6 Timetable table

## B.7 ROOM ALLOCATION TABLE



The screenshot shows a web application window titled "ROOM ALLOCATION" in red text. The background is a dark wood texture. The form contains five input fields with green labels: "USN", "SUBJECT\_ID", "ROOM\_ID", "FACULTY\_ID", and "TOTAL STUDENTS". To the right of each label is a white input box. Below the input fields is a "CLEAR" button. At the bottom are three blue buttons: "ADD", "VIEW", and "BACK".

Fig:B.7 Room allocation table

## B.8 VIEW TABLE



The screenshot shows a web application window titled "STUDENT DETAILS" in blue text. The background is a dark wood texture. A table with 4 columns (usn, name, sem, branch\_id) and 22 rows of student data is displayed. To the right of the table are two buttons: a green "PRINT" button and a blue "BACK" button.

usn	name	sem	branch_id
1	adarsh	1	1
10	jagdish	3	6
11	jaswanth	3	6
12	kotresh	4	5
13	madhu	4	5
14	manoj	4	5
15	moni	4	5
16	shashi	3	4
17	raj	3	4
18	rajkumar	5	3
19	kumar	5	3
2	ajith	1	1
20	naveen	6	2
21	nandeesh	6	2
22	naveenkumar	7	4
3	ajmal	1	1
4	anil	1	1
5	ankith	1	1
8	chetan	8	1
9	gokul	3	6

Fig:B.8 View table