DIAGNOSTIC LABORATORY MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

SK .TABISH [RA2211003011087]

K. YASWANTH [RA2211003011104]

T. SANSKAR [RA2211003011130]

Under the Guidance of

Dr. J. KALAIVANI

Assistant Professor

Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING

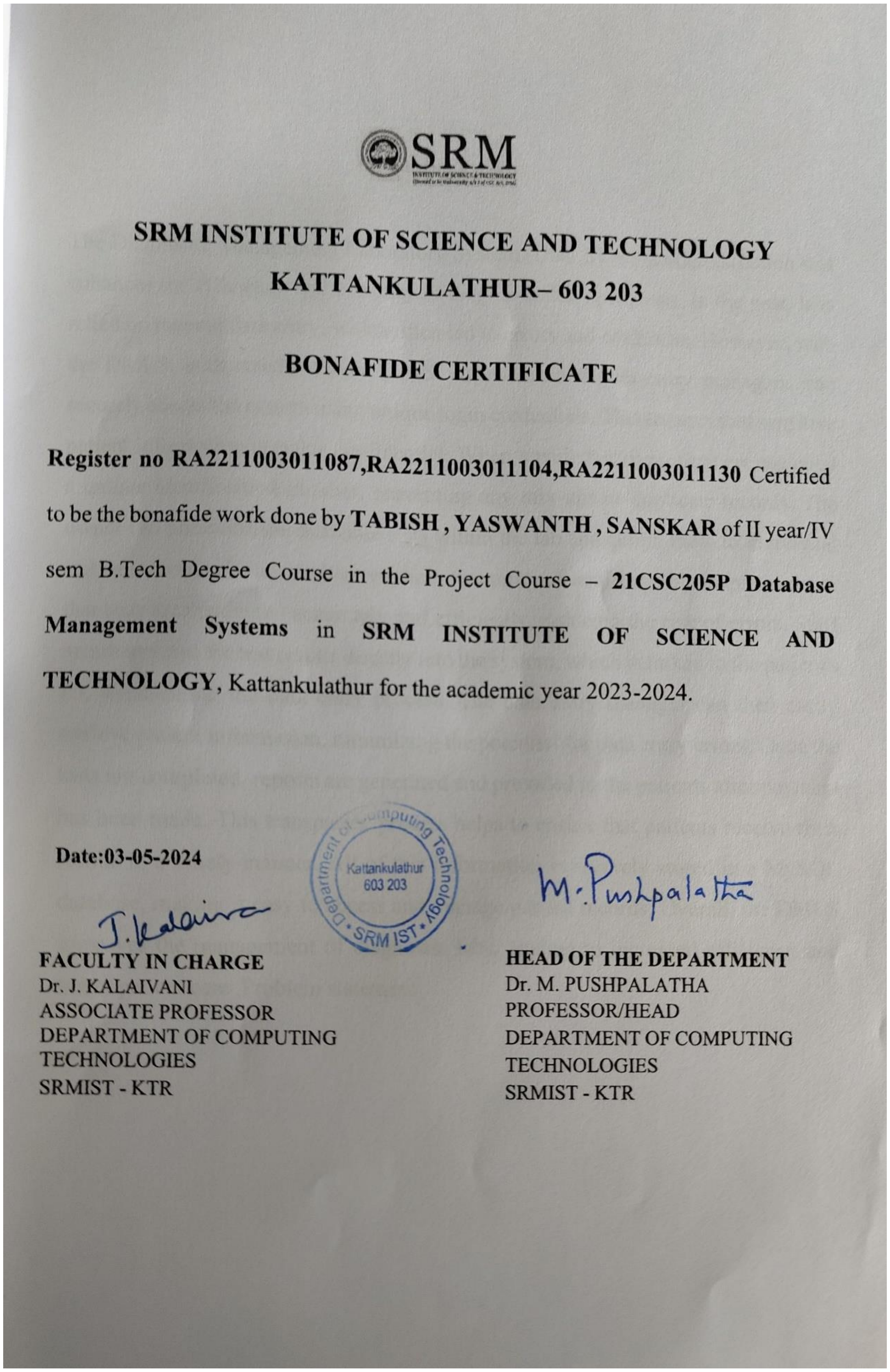
DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR– 603 203

BONAFIDE CERTIFICATE

Register no RA2211003011087,RA2211003011104,RA2211003011130 Certified to be the bonafide work done by **TABISH , YASWANTH , SANSKAR** of II year/IV sem B.Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date:03-05-2024

Department of Computing Technology
Kattankulathur
603 203
SRM IST

[illegible]
FACULTY IN CHARGE
Dr. J. KALAIVANI
ASSOCIATE PROFESSOR
DEPARTMENT OF COMPUTING TECHNOLOGIES
SRMIST - KTR

M.Pushpalatha
HEAD OF THE DEPARTMENT
Dr. M. PUSHPALATHA
PROFESSOR/HEAD
DEPARTMENT OF COMPUTING TECHNOLOGIES
SRMIST - KTR

ABSTRACT

The Diagnostic Management Laboratory System (DMLS) is a modern solution that enhances the efficiency and accuracy of diagnostic lab operations. In the past, labs relied on manual data entry, which often led to errors and confusion. However, with the DMLS, authorized personnel, such as owners and data entry managers, can securely access the system using unique login credentials. This ensures that sensitive patient information remains confidential. When a patient arrives, they are assigned a unique identification number, preventing any mix-ups or duplicate records. The owner can predefine the available tests within the lab and assign them to individual staff members, who are each given their own unique identifier. This helps to ensure that tests are conducted accurately and efficiently, reducing the risk of errors. Staff members input the test results directly into the system, which is linked to the patient's ID, streamlining the data entry process. The data entry manager can then easily retrieve patient information, minimizing the potential for data entry errors. Once the tests are completed, reports are generated and provided to the patients after payment has been made. This transparent process helps to ensure that patients receive their results in a timely manner. All of this information is securely stored in a MySQL database, making it easy to access and manage patient records. Overall, the DMLS improves the management of diagnostic labs, leading to increased efficiency and better patient care ,Problem statement

PROBLEM STATEMENT

The current manual system for managing diagnostic laboratory operations faces significant challenges including errors in data entry, data redundancy, lack of coordination, and security vulnerabilities, all of which impede efficiency, accuracy, and patient confidentiality. To address these issues, there is a pressing need for a modern Diagnostic Management Laboratory System (DMLS) that automates key processes, centralizes data management, and enhances security measures. By implementing a digital platform for managing patient information, test assignments, and staff activities, the DMLS can minimize errors, ensure consistency in records, and protect sensitive data from unauthorized access or breaches. Features such as unique patient identification numbers, staff identifiers, and secure login credentials for authorized personnel help prevent data redundancy, ensure accountability in test execution, and safeguard patient confidentiality. Overall, the adoption of a modern DMLS promises to significantly improve the efficiency, accuracy, and security of diagnostic laboratory operations, ultimately enhancing patient care and workflow efficiency for lab personnel.

TABLE OF CONTENTS

3.	Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.	**11 - 21**
4.	Analyzing the pitfalls, identifying the dependencies, and applying normalizations	**22 – 30**
5.	Implementation of concurrency control and recovery mechanisms	**31 - 33**
6.	Code for the project	**34 - 91**
7.	Result and Discussion (Screen shots of the implementation with front end.	**92 - 95**
8.	Attach the Real Time project certificate / Online course certificate	**96 -**

CHAPTER – 1

1.1 Introduction

The Diagnostic Management Laboratory System project introduces a transformative approach to managing diagnostic laboratory operations. Traditionally, diagnostic labs have relied on manual processes for recording patient information, conducting tests, and generating reports, leading to inefficiencies, errors, and security vulnerabilities. However, the DMLS project aims to address these challenges by implementing a modern, automated system that streamlines key processes, enhances data integrity, and improves security measures. By leveraging technology to centralize data management, introduce unique identifiers for patients and staff, and implement secure login credentials for authorized personnel, the DMLS project promises to revolutionize diagnostic lab operations, ultimately enhancing efficiency, accuracy, and patient confidentiality. This introduction sets the stage for understanding the significance and objectives of the DMLS project in transforming traditional diagnostic lab practices into a more efficient, secure, and patient-centric model.

1.2 Problem Understanding

The Diagnostic Management Laboratory System (DMLS) project aims to revolutionize the conventional methods prevalent in diagnostic laboratories by introducing a modernized approach to managing operations. The current reliance on manual processes, such as handwritten records and disparate data entry systems, poses significant challenges, including errors, redundancies, coordination issues, and security vulnerabilities. These outdated practices hinder operational efficiency, compromise the accuracy of test results, and jeopardize patient confidentiality. To address these shortcomings, the DMLS project proposes the implementation of an automated solution that centralizes patient information, test assignments, and staff activities within a digital platform. By standardizing processes and consolidating data management, the DMLS seeks to streamline operations and minimize the risk of errors associated with manual data entry. Furthermore, the system will enhance security measures through encrypted data storage and user authentication protocols, thereby safeguarding patient privacy and confidentiality.

In addition to improving accuracy and security, the DMLS project aims to enhance efficiency and coordination within diagnostic laboratories. By providing a centralized platform for managing tasks and tracking progress, the system will facilitate better coordination among staff members and enable real-time visibility into lab activities. Clear task assignments, automated reminders, and streamlined communication channels will help optimize workflow efficiency and ensure timely completion of tasks. Moreover, the DMLS will enable managers to monitor performance metrics and identify areas for improvement, thereby promoting continuous enhancement of operational processes. Through its comprehensive approach to modernizing diagnostic lab operations, the DMLS project endeavors to optimize efficiency, accuracy, and security while ultimately enhancing the quality of patient care.

1.3 Modules Involved

1. Login
2. Admin_view
3. Test
4. Add_test
5. Update_test
6. Patient
7. Add_patient_info
8. Delete_patient_info
9. Update_patient_info
10. Staff
11. Add_staff
12. Delete_staff
13. Report
14. Payment
15. Bill_generation

1.4 Identification Of Entities

Based on the modules involved in our project we are taking around six entities namely

1. Admin
2. Staff
3. Test
4. Report
5. Patients
6. Payment

1.5 Construction Of ER Diagram

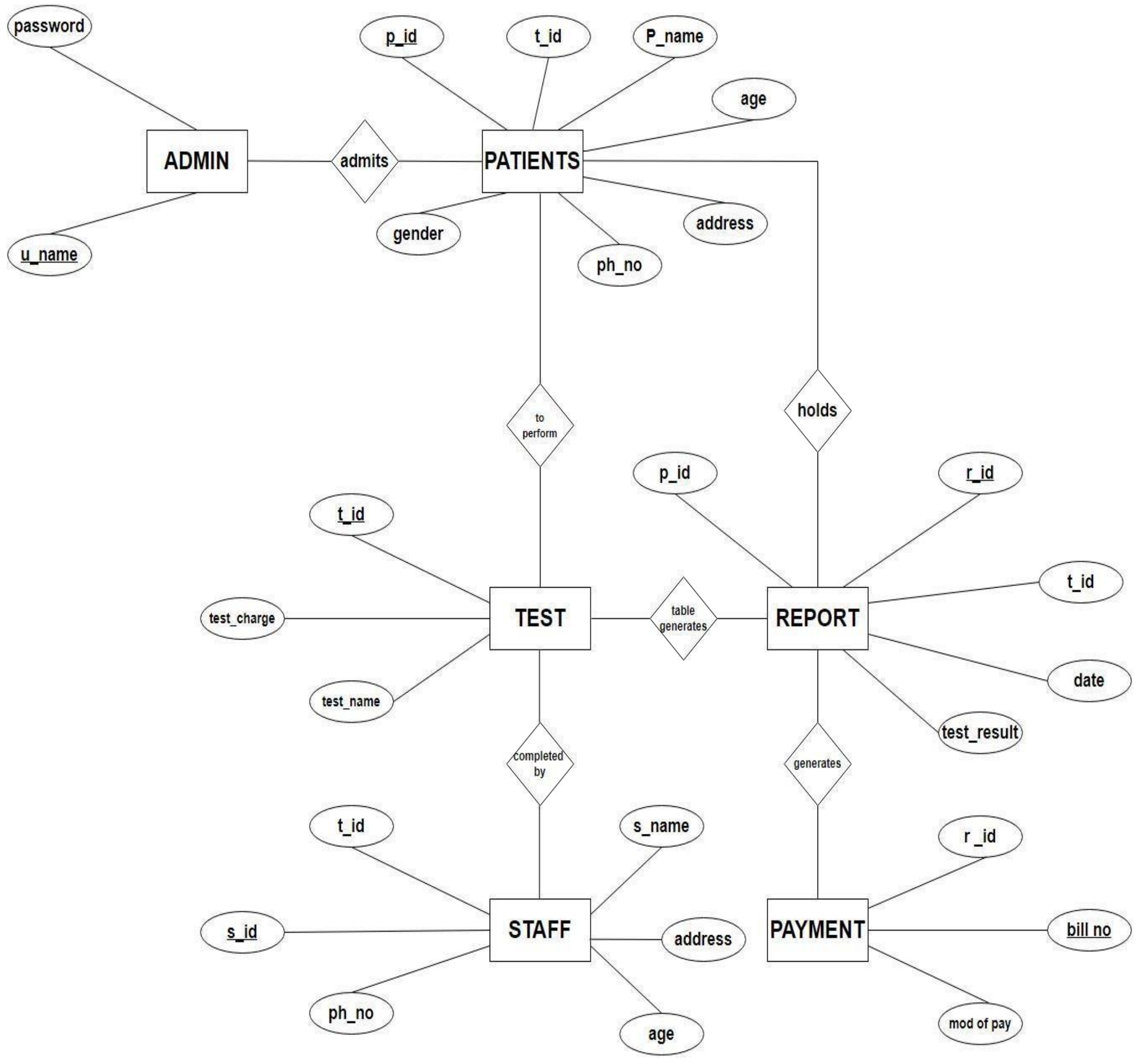


Fig 1 ER Diagram

1.6 Construction of a database

Using the entities we have and through following the guidelines to develop a database. We constructed a database named dlm in MYSQL

1.6.1 Rules followed

In our project we have only strong entity set so if we wanted to convert that ER diagram into tables we should follow these rules

1. Entity becomes table
2. Single valued attribute /simple attributes becomes columns

3. Ignore derived attributes
4. Simple attributes of composite attributes are considered but ignore composite attributes
5. Multivalued attributes are represented by a separate table
6. Ket attribute becomes primary key
7. Strong entity – entity which uses primary key is called as strong entity

CHAPTER – 2

2.1 Architecture

The architecture of our project is as follows:

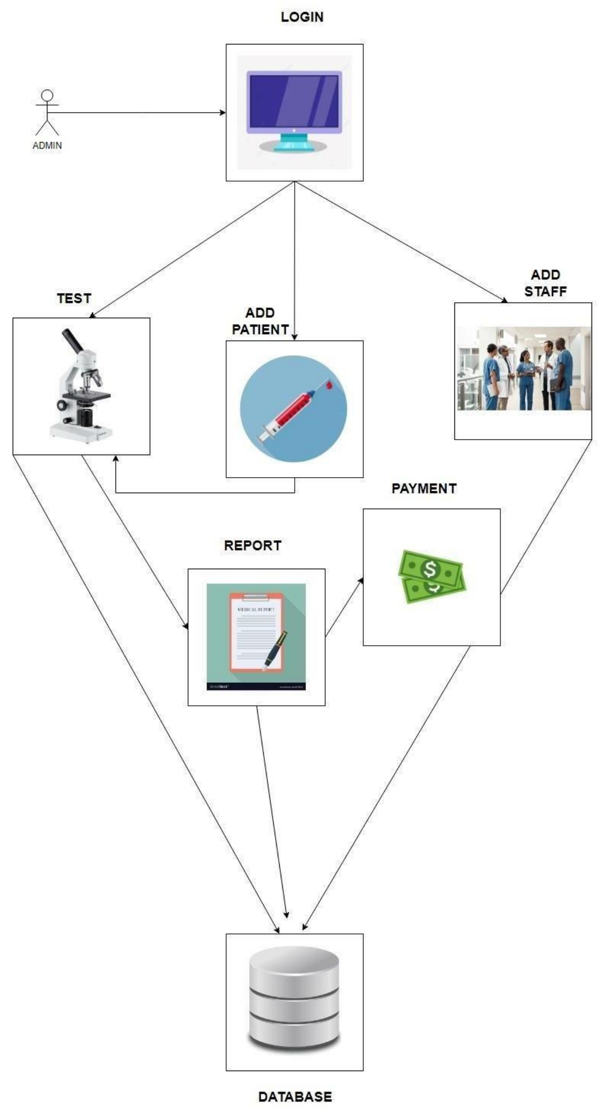


Fig2 : Architecture Diagram

It shows that only admin can login using their credentials and then the API gives about three options to add test, add patients, add staff once we select them we will be moved to new interface where after performing test we will move to report page once the data entered it will be taking to payment and after paying the bill the report will be generated all these data will be stored in a database that we created dlm.

2.2 Design Of Relational Schemas

On the basis of ER diagram we are going to convert the ER diagram into tables Schema is as below:

1. adm(u_name,pwd);
2. admits(p_id,u_name);
3. completed_by(t_id,s_id);
4. generates(r_id,bill_no);
5. holds(p_id,r_id);
6. patient(p_id,t_id,p_name,age,address,ph_no,gender);
7. payment(r_id,bill_no,mod_of_pay);
8. report(p_id,r_id,t_id,dat,test_result);
9. staff(s_id,t_id,addrtess,age,ph_no);
10. t_generates(r_id,t_id);
11. test(t_id,test_name,test_charge);

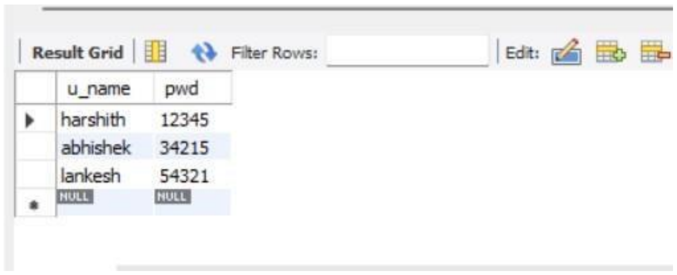
2.3 Tables Involved

1. adm
2. admits
3. completed_by
4. generates
5. holds
6. patient
7. payment
8. report
9. staff
10. t_generates
11. test

2.3.1 Queries used to create tables

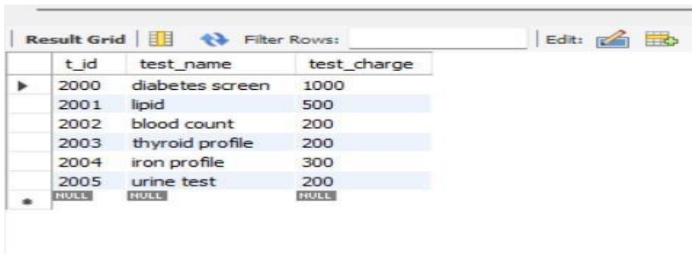
```
create table adm(
u_name varchar(10) not null unique, pwd
varchar(10)not null unique, primary
key(u_name)
```

);



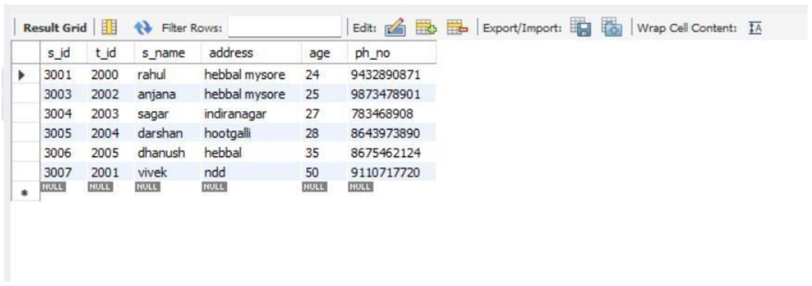
u_name	pwd
harshith	12345
abhishek	34215
lankesh	54321
NULL	NULL

Fig 3 : table adm create table test(t_id int not null unique,
test_name varchar(10), test_charge int, primary key(t_id)
) ;



t_id	test_name	test_charge
2000	diabetes screen	1000
2001	lipid	500
2002	blood count	200
2003	thyroid profile	200
2004	iron profile	300
2005	urine test	200
NULL	NULL	NULL

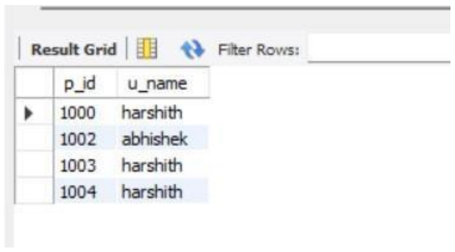
Fig 4 :table test create table patient(p_id int not null, t_id
int ,
p_name varchar(20) not null, age int, address varchar(50), ph_no bigint, gender varchar(10),
primary key(p_id) , constraint fk_tid foreign key(t_id) references test(t_id) on update cascade
on delete cascade
);



s_id	t_id	s_name	address	age	ph_no
3001	2000	rahul	hebbal mysore	24	9432890871
3003	2002	anjana	hebbal mysore	25	9873478901
3004	2003	sagar	indiranagar	27	783468908
3005	2004	darshan	hootgalli	28	8643973890
3006	2005	dhanush	hebbal	35	8675462124
3007	2001	vivek	ndd	50	9110717720
NULL	NULL	NULL	NULL	NULL	NULL

Fig 5: table patients

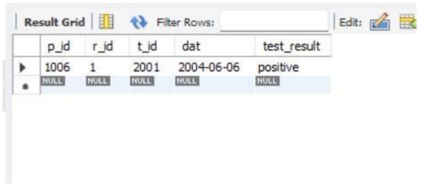
create table admits(p_id int, u_name varchar(10), constraint fkp_id foreign key(p_id) references patient(p_id) on update cascade on delete cascade, constraint fku_name foreign key(u_name) references adm(u_name)on update cascade on delete cascade);



p_id	u_name
1000	harshith
1002	abhishek
1003	harshith
1004	harshith

Fig 6: table admits

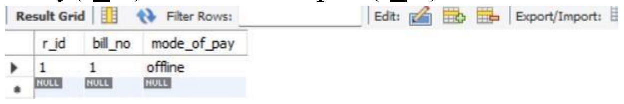
create table report(p_id int, r_id int not null , t_id int, dat date, test_result varchar(20), primary key(r_id), constraint fkt_id foreign key(t_id) references test(t_id) on delete cascade on update cascade, constraint fk_p_id_ foreign key(p_id)references patient(p_id) on update cascade on delete cascade);



p_id	r_id	t_id	dat	test_result
1006	1	2001	2004-06-06	positive
NULL	NULL	NULL	NULL	NULL

Fig 7: table report

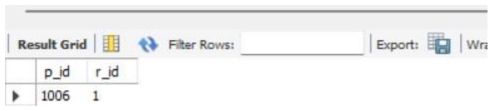
create table payment(r_id
int, bill_no int not null ,
mode_of_pay varchar(20),
primary key(bill_no),
constraint fkr_id_ foreign key(r_id) references report(r_id) on delete cascade on update cascade);



r_id	bill_no	mode_of_pay
1	1	offline
NULL	NULL	NULL

Fig 8 : table payment

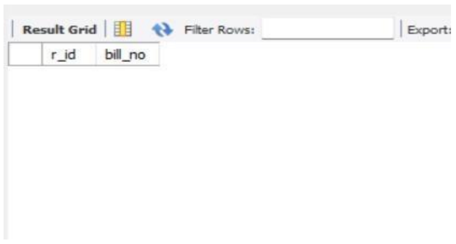
create table t_generates(r_id int, t_id int, constraint fkr_id foreign key(r_id)references report(r_id) on update cascade on delete cascade, constraint fkt______id foreign key(t_id)references test(t_id) on update cascade on delete cascade); create table holds (p_id int, r_id int, constraint fkrid foreign key(r_id) references report(r_id) on delete cascade on update cascade, constraint fkp_id foreign key(p_id) references patient(p_id)on update cascade on delete cascade);



p_id	r_id
1006	1

Fig 9 : table holds

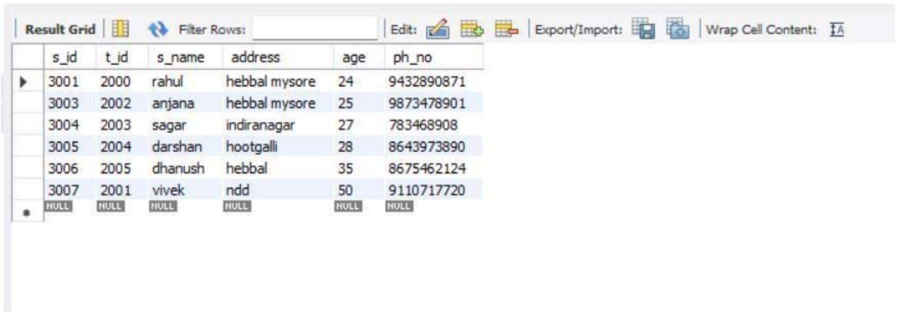
create table generates(r_id int, bill_no int, constraint fk_r_id foreign key(r_id) references report(r_id) on delete cascade on update cascade, constraint fkbill_no foreign key(bill_no) references payment(bill_no) on delete cascade on update cascade);



r_id	bill_no

Fig 10 : table generates

create table staff(s_id int not null , t_id int, s_name varchar(20), address varchar(50), age int, ph_no bigint, primary key(s_id),
constraint fkt_id__ foreign key(t_id)references test(t_id) on update cascade on delete cascade);



s_id	t_id	s_name	address	age	ph_no
3001	2000	rahul	hebbal mysore	24	9432890871
3003	2002	anjana	hebbal mysore	25	9873478901
3004	2003	sagar	indiranagar	27	783468908
3005	2004	darshan	hootgalli	28	8643973890
3006	2005	dhanush	hebbal	35	8675462124
3007	2001	vivek	ndd	50	9110717720
NULL	NULL	NULL	NULL	NULL	NULL

Fig 11: table staff

CHAPTER – 3

3.1 Types Of Queries Used

3.1.1 DDL(DATA DEFINITION LANGUAGE)

DDL is used to define the structure of a database, including creating, modifying, and deleting database objects like tables, indexes, and constraints.

Create – create database and tables

Alter – alter the structure of database

Drop - delete the table

Truncate – remove all the records of a table

3.1.2 DML(DATA MANIPULATION LANGUAGE)

DML is used to manipulate data within the database objects like inserting, updating, and deleting records.

Select – retrieve data from database

Insert – insert data into the database

Delete – delete single or multiple records

Update – update the data

3.1.3 DCL(DATA CONTROL LANGUAGE)

DCL is used to control access to data within the database.

Grant – giving privileges to the user to access a database Revoke

– Taking back the permissions given to the user

3.1.4 TCL(TRANSACTION CONTROL LANGUAGE)

TCL is used to manage transactions within the database, including committing or rolling back changes.

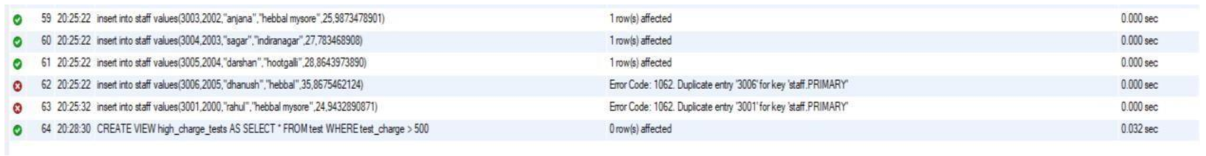
Commit – save the transaction to the database

Rollback – undo the recent transaction

3.2 Queries

1 . CREATE VIEW high_charge_tests AS

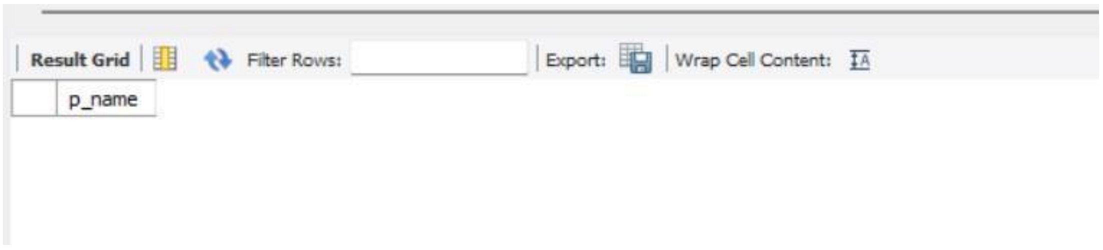
SELECT * FROM test WHERE test_charge > 500;



```
59  20:25:22  insert into staff values(3003,2002,"anjana","hebbal mysore",25,9873478901)    1 row(s) affected    0.000 sec
60  20:25:22  insert into staff values(3004,2003,"sagar","indiranagar",27,783468908)    1 row(s) affected    0.000 sec
61  20:25:22  insert into staff values(3005,2004,"darshan","hootgalli",28,8643973890)    1 row(s) affected    0.000 sec
62  20:25:22  insert into staff values(3006,2005,"dhanush","hebbal",35,8675462124)    Error Code: 1062. Duplicate entry '3006' for key 'staff.PRIMARY'    0.000 sec
63  20:25:32  insert into staff values(3001,2000,"rahul","hebbal mysore",24,9432890871)    Error Code: 1062. Duplicate entry '3001' for key 'staff.PRIMARY'    0.000 sec
64  20:28:30  CREATE VIEW high_charge_tests AS SELECT * FROM test WHERE test_charge > 500    0 row(s) affected    0.032 sec
```

Explanation: This view named high_charge_tests will display all the tests with charges greater than 500.

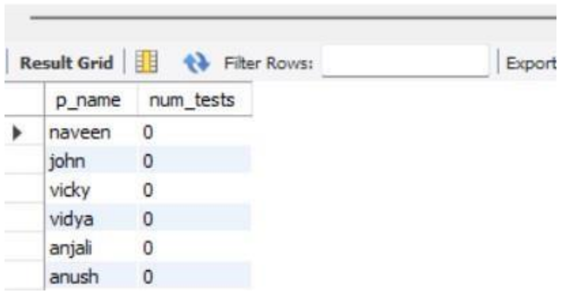
2. SELECT p_name
FROM patient
WHERE p_id IN (SELECT p_id FROM report WHERE test_result = 'Positive');



```
p_name
```

Explanation: This query retrieves the names of patients who have received positive test results.

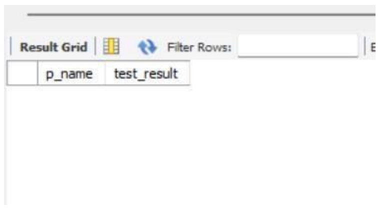
3. SELECT p.p_name, COUNT(r.r_id) AS num_tests
FROM patient p
LEFT JOIN report r ON p.p_id = r.p_id
GROUP BY p.p_name;



```
p_name   num_tests
naveen   0
john     0
vicky    0
vidya    0
anjali   0
anush    0
```

Explanation: It counts the number of tests each patient has undergone and returns the patient name along with the count.

4. SELECT p.p_name, r.test_result
FROM patient p
INNER JOIN report r ON p.p_id = r.p_id AND r.dat = (SELECT MAX(dat) FROM report WHERE p_id = p.p_id);


Result Grid Filter Rows:

	p_name	test_result

Explanation: This query fetches the latest test result for each patient.

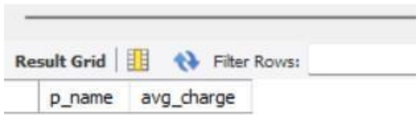
5. SELECT p.p_name, AVG(t.test_charge) AS avg_charge

FROM patient p

INNER JOIN report r ON p.p_id = r.p_id

INNER JOIN test t ON r.t_id = t.t_id

GROUP BY p.p_name;


Result Grid Filter Rows:

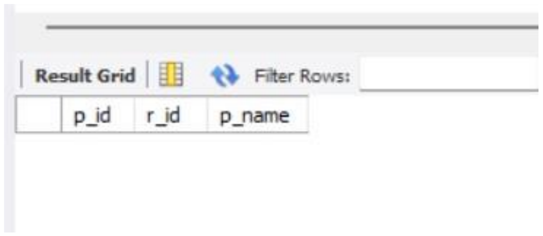
	p_name	avg_charge

Explanation: Calculates the average test charge for each patient.

6. SELECT h.p_id, h.r_id, p.p_name

FROM holds h

INNER JOIN patient p ON h.p_id = p.p_id;


Result Grid Filter Rows:

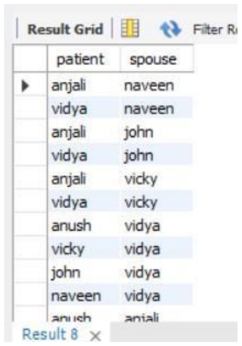
	p_id	r_id	p_name

Explanation: Retrieves the patient name along with their holds details.

7. SELECT p1.p_name AS patient, p2.p_name AS spouse

FROM patient p1

INNER JOIN patient p2 ON p1.gender != p2.gender;



Result Grid | Filter R

patient	spouse
anjali	naveen
vidya	naveen
anjali	john
vidya	john
anjali	vicky
vidya	vicky
anush	vidya
vicky	vidya
john	vidya
naveen	vidya
anush	anjali

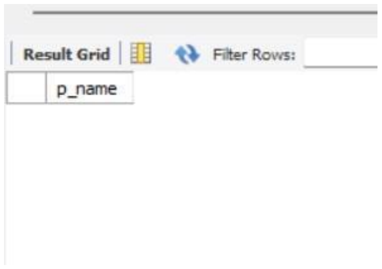
Result 8 ×

Explanation: Fetches pairs of patients who could potentially be spouses based on different genders.

8. SELECT p.p_name

FROM patient p

WHERE EXISTS (SELECT 1 FROM report WHERE p_id = p.p_id);

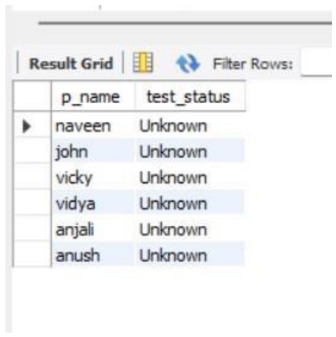


Result Grid | Filter Rows:

p_name

Explanation: Retrieves names of patients who have undergone tests.

```
9. SELECT p.p_name,
       CASE
           WHEN r.test_result = 'Positive' THEN 'Requires Attention'
           WHEN r.test_result = 'Negative' THEN 'Normal'
           ELSE 'Unknown'
       END AS test_status
FROM patient p
LEFT JOIN report r ON p.p_id = r.p_id;
```

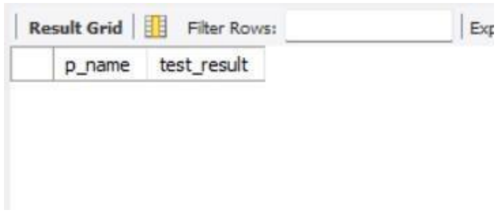


```
p_name   test_status
naveen   Unknown
john     Unknown
vicky    Unknown
vidya    Unknown
anjali   Unknown
anush    Unknown
```

Explanation: This query categorizes the test results of patients into different statuses like 'Requires Attention', 'Normal', or 'Unknown' using a CASE statement.

10 .

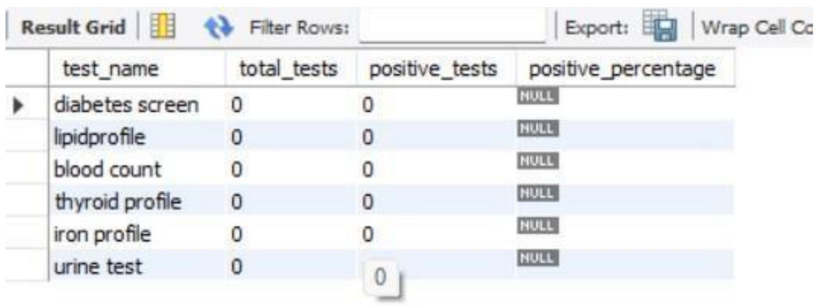
```
SELECT p.p_name, r.test_result
FROM patient p
INNER JOIN LatestReport lr ON p.p_id = lr.p_id
INNER JOIN report r ON lr.p_id = r.p_id AND lr.latest_date = r.dat;
```



```
p_name   test_result
```

Explanation: Retrieves the names of patients who have undergone more than 2 tests.

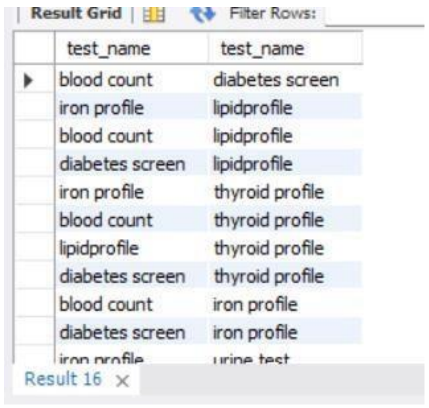
11.
```
SELECT t.test_name,
    (SELECT COUNT(*) FROM report WHERE t_id = t.t_id) AS total_tests,
    (SELECT COUNT(*) FROM report WHERE t_id = t.t_id AND test_result = 'Positive') AS
positive_tests,
    (SELECT COUNT(*) FROM report WHERE t_id = t.t_id AND test_result = 'Positive') /
    (SELECT COUNT(*) FROM report WHERE t_id = t.t_id) * 100 AS positive_percentage
FROM test t;
```



test_name	total_tests	positive_tests	positive_percentage
diabetes screen	0	0	NULL
lipidprofile	0	0	NULL
blood count	0	0	NULL
thyroid profile	0	0	NULL
iron profile	0	0	NULL
urine test	0	0	NULL

Explanation: This query calculates the percentage of positive tests for each test type.

```
12. SELECT t1.test_name, t2.test_name
FROM test t1
CROSS JOIN test t2
WHERE t1.test_name < t2.test_name;
```

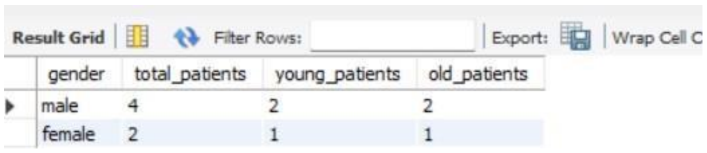


test_name	test_name
blood count	diabetes screen
iron profile	lipidprofile
blood count	lipidprofile
diabetes screen	lipidprofile
iron profile	thyroid profile
blood count	thyroid profile
lipidprofile	thyroid profile
diabetes screen	thyroid profile
blood count	iron profile
diabetes screen	iron profile
iron profile	urine test

Result 16

Explanation: This query generates combinations of test pairs, avoiding duplicates by ensuring that the first test name is less than the second one.

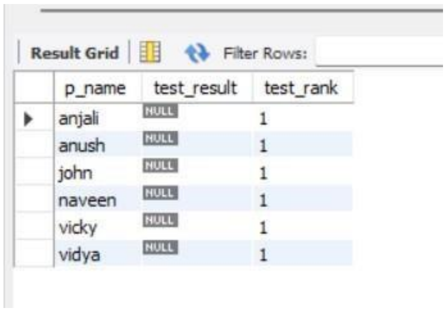
```
13. SELECT p.gender,
    COUNT(*) AS total_patients,
    SUM(CASE WHEN p.age < 30 THEN 1 ELSE 0 END) AS young_patients,
    SUM(CASE WHEN p.age >= 30 THEN 1 ELSE 0 END) AS old_patients
FROM patient p
GROUP BY p.gender;
```



gender	total_patients	young_patients	old_patients
male	4	2	2
female	2	1	1

Explanation: This query counts the total number of patients and categorizes them into young and old based on age using CASE statements.

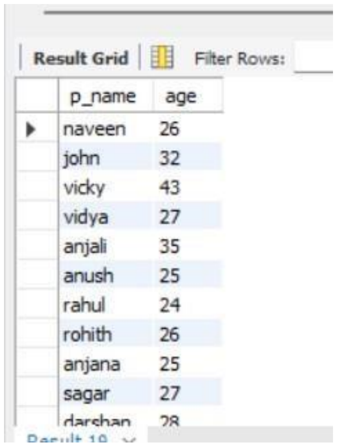
14. SELECT p.p_name, r.test_result,
    ROW_NUMBER() OVER (PARTITION BY p.p_name ORDER BY r.dat DESC) AS test_rank
FROM patient p
LEFT JOIN report r ON p.p_id = r.p_id;



p_name	test_result	test_rank
anjali	NULL	1
anush	NULL	1
john	NULL	1
naveen	NULL	1
vicky	NULL	1
vidya	NULL	1

Explanation: This query ranks the test results for each patient based on the test date.

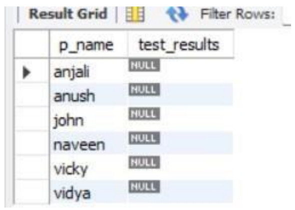
15. (SELECT p_name, age FROM patient) UNION ALL
(SELECT s_name, age FROM staff);



p_name	age
naveen	26
john	32
vicky	43
vidya	27
anjali	35
anush	25
rahul	24
rohith	26
anjana	25
sagar	27
darshan	28

Explanation: This query combines the names and ages of patients and staff members.

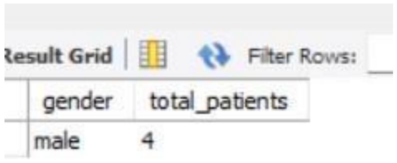
16. SELECT p.p_name, GROUP_CONCAT(r.test_result SEPARATOR ', ') AS test_results
FROM patient p
LEFT JOIN report r ON p.p_id = r.p_id
GROUP BY p.p_name;



```
Result Grid | Filter Rows:
p_name    test_results
anjali    NULL
anush     NULL
john      NULL
naveen    NULL
vicky     NULL
vidya     NULL
```

Explanation: This query concatenates all the test results of each patient into a single string separated by commas.

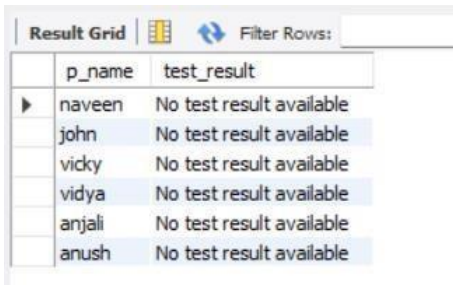
17. SELECT p.gender, COUNT(*) AS total_patients
FROM patient p
GROUP BY p.gender
HAVING COUNT(*) > 2;



```
Result Grid | Filter Rows:
gender    total_patients
male      4
```

Explanation: This query retrieves the gender distribution of patients but only includes genders with more than 2 occurrences.

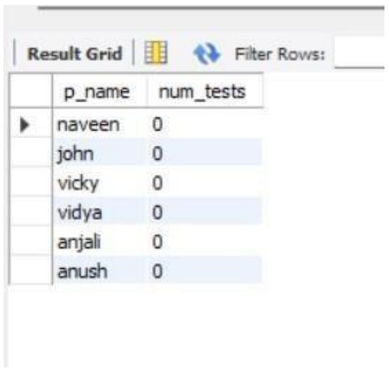
18 . SELECT p.p_name, COALESCE(r.test_result, 'No test result available') AS test_result
FROM patient p
LEFT JOIN report r ON p.p_id = r.p_id;



p_name	test_result
naveen	No test result available
john	No test result available
vicky	No test result available
vidya	No test result available
anjali	No test result available
anush	No test result available

Explanation: This query uses COALESCE to replace NULL test results with a default message.

19. SELECT p.p_name, (SELECT COUNT(*) FROM report WHERE p_id = p.p_id) AS num_tests FROM patient p;



p_name	num_tests
naveen	0
john	0
vicky	0
vidya	0
anjali	0
anush	0

Explanation: This query uses a scalar subquery in the SELECT clause to fetch the number of tests for each patient.

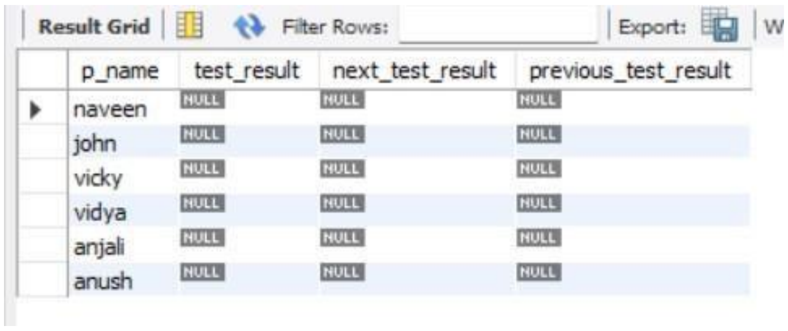
20 . SELECT p.p_name, r.test_result,

LEAD(r.test_result) OVER (PARTITION BY p.p_id ORDER BY r.dat) AS next_test_result,

LAG(r.test_result) OVER (PARTITION BY p.p_id ORDER BY r.dat) AS previous_test_result

FROM patient p

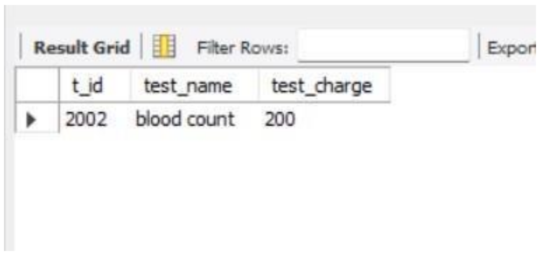
LEFT JOIN report r ON p.p_id = r.p_id;



p_name	test_result	next_test_result	previous_test_result
naveen	NULL	NULL	NULL
john	NULL	NULL	NULL
vicky	NULL	NULL	NULL
vidya	NULL	NULL	NULL
anjali	NULL	NULL	NULL
anush	NULL	NULL	NULL

Explanation: This query uses LEAD() and LAG() functions to fetch the next and previous test results for each patient.

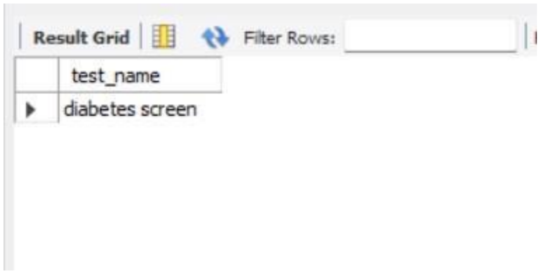
21 . (SELECT * FROM test WHERE test_name LIKE 'blood%')



t_id	test_name	test_charge
2002	blood count	200

Explanation: This query uses EXCEPT to find tests related to blood but not specifically blood count.

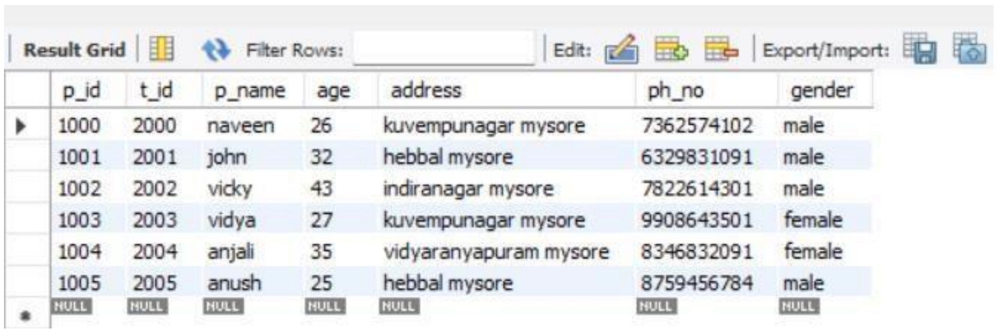
22. SELECT test_name
FROM test
WHERE test_charge > ALL (SELECT test_charge FROM test WHERE test_name LIKE '%profile%');



test_name
diabetes screen

Explanation: This query finds the test(s) with the highest charge compared to all tests with 'profile' in their name.

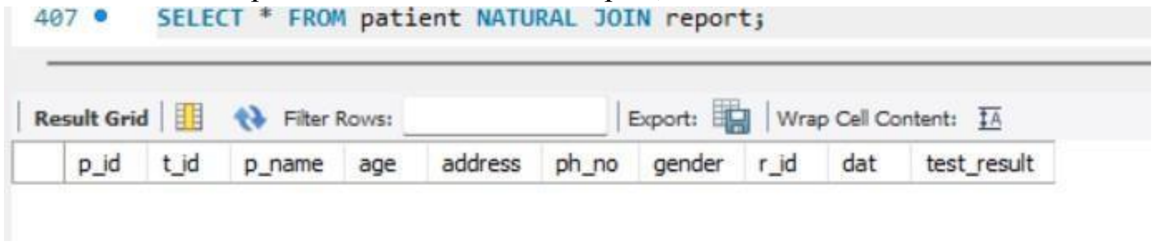
23. SELECT * FROM patient FOR UPDATE;



p_id	t_id	p_name	age	address	ph_no	gender
1000	2000	naveen	26	kuvempunagar mysore	7362574102	male
1001	2001	john	32	hebbal mysore	6329831091	male
1002	2002	vicky	43	indiranagar mysore	7822614301	male
1003	2003	vidya	27	kuvempunagar mysore	9908643501	female
1004	2004	anjali	35	vidyaranyapuram mysore	8346832091	female
1005	2005	anush	25	hebbal mysore	8759456784	male
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Explanation: This query locks rows in the patient table for update operations.

24. SELECT * FROM patient NATURAL JOIN report;

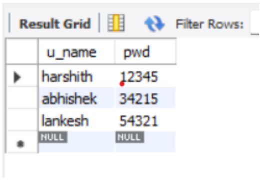


```
SELECT * FROM patient NATURAL JOIN report;
```

p_id	t_id	p_name	age	address	ph_no	gender	r_id	dat	test_result

Explanation: This query performs a natural join between the patient and report tables based on columns with the same names.

25. select * from adm;



u_name	pwd
harshith	12345
abhishek	34215
lankesh	54321
NULL	NULL

It gives the whole admin table

CHAPTER – 4

4.1 Functional Dependencies Identification

Based on rules to find functional dependencies we attained as:

Rule : functional dependency(if tuple1.a = tuple2.a

Then

tuple1.b = tuple2.b)

1. Adm{ u_name -> pwd}
2. admits{p_id, u_name} -> {p_id}, {u_name}
3. completed_by{s_id -> t_id}
4. generates{r_id -> bill_no}
5. holds{p_id -> r_id}
6. patient{p_id ->p_name,age,address,ph_no,gender},{p_id,p_name->age,gender,ph_no}{ph_no ->p_id,p_name,age,address}
7. Payment{r_id ->bill_no,mod_of_pay},{bill_no ->r_id},{r_id,bill_no - >mod_of_pay}
8. Report{r_id ->p_id,t_id,date,test_result},{p_id ->t_id}
9. T_generates{r_id ->t_id}
10. Test{t_id ->test_name,test_charge},{test_name -> test_charge}
11. Staff{s_id ->t_id,address,age,ph_no},{ph_no ->s_id,t_id,address,age}

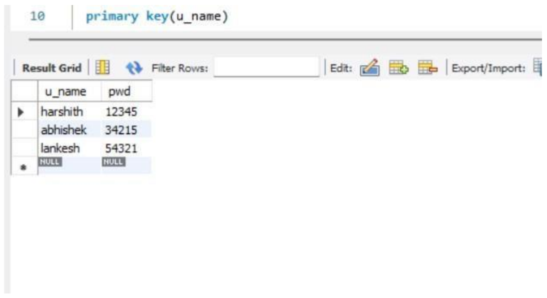
4.2 Analysing The pitfalls and Applying Normalization

Table 1: adm

FUNCTIONAL DEPENDENCY IS: Adm{ u_name -> pwd}

1 ST NORMAL FORM(1NF)

Since its having all atomic values we can say that its on 1 st normal form

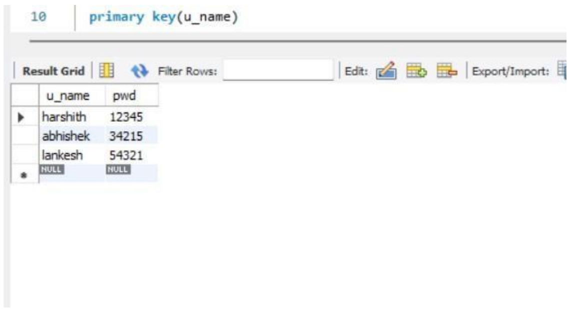


```
primary key(u_name)
```

u_name	pwd
harshith	12345
abhishek	34215
lankesh	54321
NULL	NULL

2 ND NORMAL FORM (2NF)

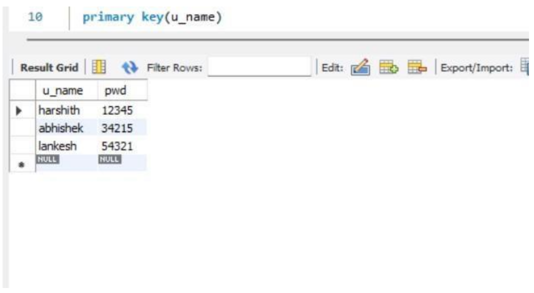
Since its satisfying the conditions like it has no partial functional dependency its directly is in 2 nd normal form


```
primary key(u_name)
```

u_name	pwd
harshith	12345
abhishek	34215
lankesh	54321
NULL	NULL

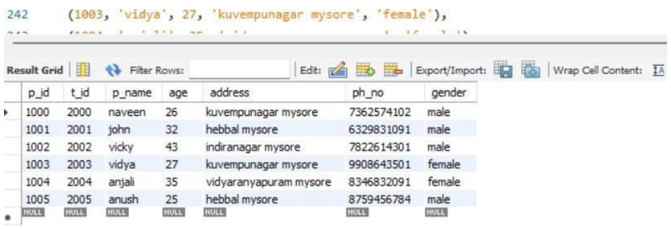
3 RD NORMAL FORM(3NF)

Since its satisfying the conditions like it has no transitive functional dependency its directly is in 3 nd normal form


```
primary key(u_name)
```

u_name	pwd
harshith	12345
abhishek	34215
lankesh	54321
NULL	NULL

Table 2: PATIENT patient{p_id ->p_name,age,address,ph_no,gender},{p_id,p_name ->age,gender,ph_no} this table is in normal form 2 (NF2)

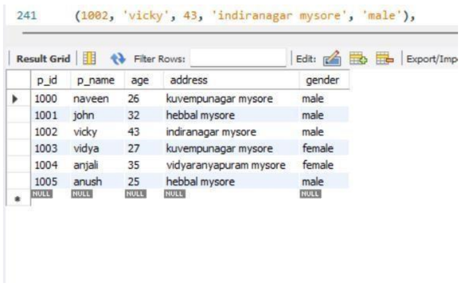

```
(1003, 'vidya', 27, 'kuvempunagar mysore', 'female'),
```

p_id	t_id	p_name	age	address	ph_no	gender
1000	2000	naveen	26	kuvempunagar mysore	7362574102	male
1001	2001	john	32	hebbal mysore	6329831091	male
1002	2002	vicky	43	indiranagar mysore	7822614301	male
1003	2003	vidya	27	kuvempunagar mysore	9908643501	female
1004	2004	anjali	35	vidyaranyapuram mysore	8346832091	female
1005	2005	anush	25	hebbal mysore	8759456784	male
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Now we are changing this table to normal form 3 as it unsatisfy the condition of transitive dependency

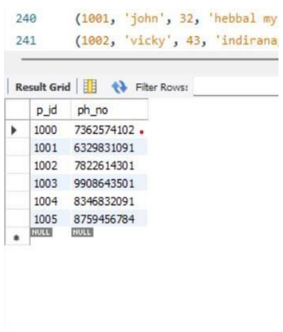
NORMAL FORM 3(NF3)

TABLE 1


```
(1002, 'vicky', 43, 'indiranagar mysore', 'male'),
```

p_id	p_name	age	address	gender
1000	naveen	26	kuvempunagar mysore	male
1001	john	32	hebbal mysore	male
1002	vicky	43	indiranagar mysore	male
1003	vidya	27	kuvempunagar mysore	female
1004	anjali	35	vidyaranyapuram mysore	female
1005	anush	25	hebbal mysore	male
NULL	NULL	NULL	NULL	NULL

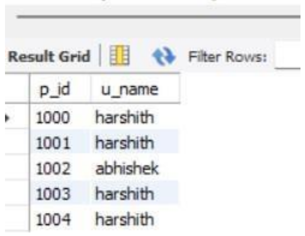
TABLE 2



```
(1001, 'john', 32, 'hebbal my
(1002, 'vicky', 43, 'indirana
```

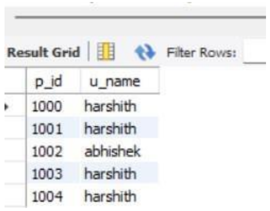
p_id	ph_no
1000	7362574102
1001	6329831091
1002	7822614301
1003	9908643501
1004	8346832091
1005	8759456784
NULL	NULL

Table 3: admits admits{p_id, u_name} ->
{p_id}, {u_name}
since this functional dependency represents no transitive dependency therefore it represents (NF3)



p_id	u_name
1000	harshith
1001	harshith
1002	abhishek
1003	harshith
1004	harshith

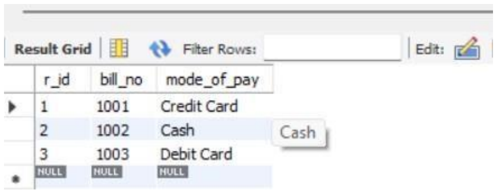
We are going to convert this table into boyce cott normal form(BCNF)
Since the determinant should be super key rather than candidate key although we change to BCNF the output will be the same



p_id	u_name
1000	harshith
1001	harshith
1002	abhishek
1003	harshith
1004	harshith

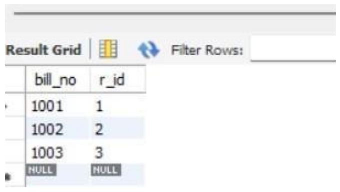
Table 4: payment
Payment{r_id ->bill_no,mod_of_pay},{bill_no ->r_id},{r_id,bill_no ->mod_of_pay} All the non-prime attributes are fully functionally dependent on the candidate keys.
There are no partial dependencies.
There are no transitive dependencies.

Since the table satisfies all the conditions of the Boyce-Codd Normal Form (BCNF), we can conclude that it exists in BCNF.



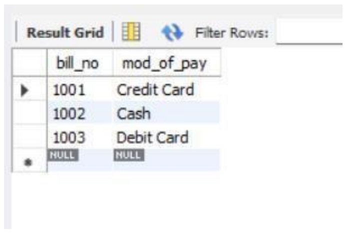
r_id	bill_no	mode_of_pay
1	1001	Credit Card
2	1002	Cash
3	1003	Debit Card
NULL	NULL	NULL

AFTER NORMALIZATION TO 4NF TABLE 1:



bill_no	r_id
1001	1
1002	2
1003	3
NULL	NULL

TABLE 2:



bill_no	mod_of_pay
1001	Credit Card
1002	Cash
1003	Debit Card
NULL	NULL

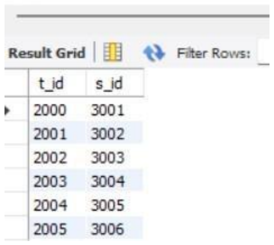
Table 5: completed by completed_by{s_id -> t_id}

Reflexivity: {s_id} -> {s_id} (Trivial)

Augmentation: {s_id} -> {t_id, s_id} (Augmenting with s_id)

Transitivity: {s_id} -> {t_id} (From {s_id} -> {t_id, s_id} and {s_id} -> {s_id})

Since it has atomic values each and converted to 2NF it has no partital functional dependency so it satisfies the concept of 2NF



Result Grid | Filter Rows:

t_id	s_id
2000	3001
2001	3002
2002	3003
2003	3004
2004	3005
2005	3006

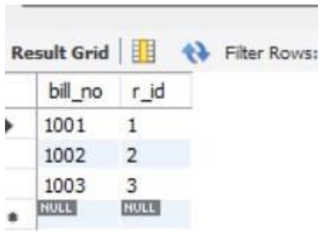
Table 6: generates generates{r_id
-> bill_no}
Reflexivity: {r_id} -> {r_id} (Trivial)
Augmentation: {r_id} -> {bill_no, r_id} (Augmenting with r_id)
Transitivity: {r_id} -> {bill_no} (From {r_id} -> {bill_no, r_id} and {r_id} -> {r_id})
Since it has atomic values each and converted to 2NF it has no partital functional dependency so it satisfies the concept of 2NF Third Normal Form (3NF):
There are no transitive dependencies in either table. Each non-key attribute is directly dependent on the primary key.
Boyce-Codd Normal Form (BCNF):
In both tables, the only determinant is the primary key, which is also a superkey.
Fourth Normal Form (4NF):
Neither table has multivalued dependencies.
Fifth Normal Form (5NF):
There are no join dependencies present in either table
Table 7 : holds holds{p_id
-> r_id}
Reflexivity: {p_id} -> {p_id} (Trivial)
Augmentation: {p_id} -> {r_id, p_id} (Augmenting with p_id)
Transitivity: {p_id} -> {r_id} (From {p_id} -> {r_id, p_id} and {p_id} -> {p_id})
Since it has atomic values each and converted to 2NF it has no partital functional dependency so it satisfies the concept of 2NF Third Normal Form (3NF):
There are no transitive dependencies in either table. Each non-key attribute is directly dependent on the primary key.
Boyce-Codd Normal Form (BCNF):
In both tables, the only determinant is the primary key, which is also a superkey.

Fourth Normal Form (4NF):

Neither table has multivalued dependencies.

Fifth Normal Form (5NF):

There are no join dependencies present in either table



Result Grid | Filter Rows:

bill_no	r_id
1001	1
1002	2
1003	3
NULL	NULL

Table 8: report

Report{r_id ->p_id,t_id,date,test_result},{p_id ->t_id} Functional Dependency Analysis:

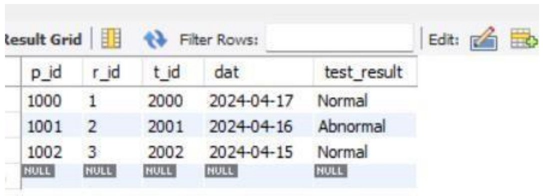
{r_id} -> {p_id, t_id, date, test_result}`: This dependency indicates that all attributes are fully functionally dependent on the candidate key {r_id}.

{p_id} -> {t_id}: This dependency indicates a partial dependency where {p_id}determines {t_id}.

Based on the above analysis:

the Report table satisfies the criteria of the Second Normal Form (2NF) because it contains no partial dependencies and all non-prime attributes are fully functionally dependent on the candidate key {r_id}

BEFORE NORMALIZATION:



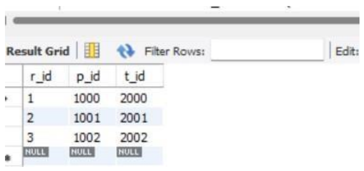
Result Grid | Filter Rows: | Edit:

p_id	r_id	t_id	dat	test_result
1000	1	2000	2024-04-17	Normal
1001	2	2001	2024-04-16	Abnormal
1002	3	2002	2024-04-15	Normal
NULL	NULL	NULL	NULL	NULL

AFTER NORMALIZATION:

We observe a transitive dependency where {p_id} -> {t_id} and {t_id} -> {date, test_result}. To remove this transitive dependency, we create a new table for Test_Details with t_id as its primary key, and another table for Report_Details with r_id as its primary key. This ensures lossless decomposition while eliminating the transitive dependency.

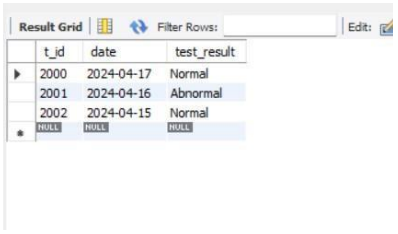
TABLE1:



Result Grid | Filter Rows: | Edit:

r_id	p_id	t_id
1	1000	2000
2	1001	2001
3	1002	2002
NULL	NULL	NULL

TABLE 2:



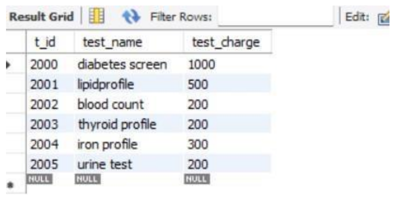
Result Grid | Filter Rows: | Edit:

t_id	date	test_result
2000	2024-04-17	Normal
2001	2024-04-16	Abnormal
2002	2024-04-15	Normal
NULL	NULL	NULL

TABLE 9: TEST

Test{t_id ->test_name,test_charge},{test_name -> test_charge}

BEFORE NORMALIZATION



Result Grid | Filter Rows: | Edit:

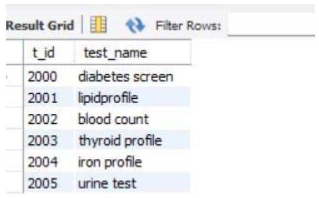
t_id	test_name	test_charge
2000	diabetes screen	1000
2001	lipidprofile	500
2002	blood count	200
2003	thyroid profile	200
2004	iron profile	300
2005	urine test	200
NULL	NULL	NULL

Actually it satisfies the condition of no partial functional dependency therefore it is in NF2

Therefore we are converting to higher NF3 where it satisfies transistive dependency

AFTER NORMALIZATION

To convert the Test table to Third Normal Form (3NF), we need to remove the transitive dependency where {test_name} determines {test_charge}. We can achieve this by decomposing the table into two tables: one for test details and another for test charges.



Result Grid | Filter Rows:

t_id	test_name
2000	diabetes screen
2001	lipidprofile
2002	blood count
2003	thyroid profile
2004	iron profile
2005	urine test

Table 10 : staff

Staff{s_id ->t_id,address,age,ph_no},{ph_no ->s_id,t_id,address,age} Let's analyze the normal form:

Candidate Keys:

Since {s_id} uniquely determines all attributes, it serves as a candidate key.

Functional Dependency Analysis:

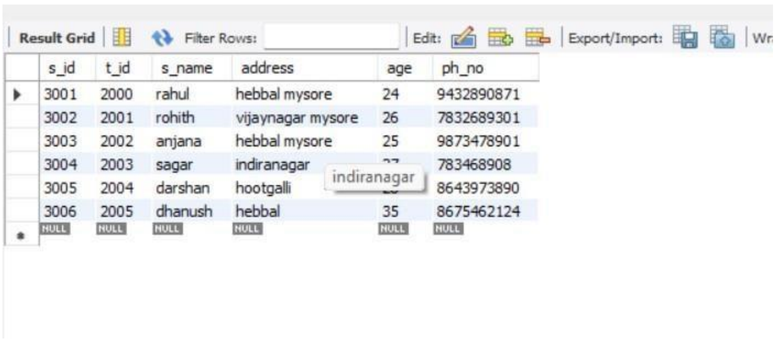
{s_id} -> {t_id, address, age, ph_no}: This dependency indicates that all attributes are fully functionally dependent on the candidate key {s_id}.

{ph_no} -> {s_id, t_id, address, age}: This dependency also indicates that all attributes are fully functionally dependent on the candidate key {ph_no}.

Based on the above analysis:

The Staff table satisfies the criteria of the Second Normal Form (2NF) because it contains no partial dependencies and all non-prime attributes are fully functionally dependent on the candidate keys {s_id} and {ph_no}.

However, it does not meet the criteria for Third Normal Form (3NF) as it contains transitive dependencies.



s_id	t_id	s_name	address	age	ph_no
3001	2000	rahul	hebbal mysore	24	9432890871
3002	2001	rohith	vijaynagar mysore	26	7832689301
3003	2002	anjana	hebbal mysore	25	9873478901
3004	2003	sagar	indiranagar	[illegible]	783468908
3005	2004	darshan	hootgalli	[illegible]	8643973890
3006	2005	dhanush	hebbal	35	8675462124
NULL	NULL	NULL	NULL	NULL	NULL

CHAPTER – 5

5.1 Scheduling

Scheduling in Database Management Systems (DBMS) plays a crucial role in optimizing the utilization of system resources and ensuring efficient execution of queries and transactions. In a DBMS, scheduling involves the allocation of system resources such as CPU time, memory, and I/O operations to various tasks in a manner that maximizes throughput and minimizes response time. One type of scheduling commonly used in DBMS is serial scheduling. In serial scheduling, transactions are executed sequentially, one after the other. Each transaction is granted exclusive access to the resources it requires until it completes, ensuring that no conflicts or concurrency issues arise. While serial scheduling ensures simplicity and avoids concurrency-related problems such as deadlock and inconsistency, it may lead to underutilization of system resources and longer response times, especially in systems with high transaction volumes.

On the other hand, parallel scheduling in DBMS involves executing multiple transactions simultaneously, utilizing the available system resources more efficiently. Parallel scheduling can be classified into two main types: inter-query parallelism and intra-query parallelism. Inter-query parallelism involves executing multiple independent queries concurrently, thereby reducing overall query response time. Intra-query parallelism, on the other hand, involves breaking down a single complex query into smaller subtasks and executing them concurrently on multiple processors or cores. This approach can significantly speed up query processing for complex analytical queries or data-intensive operations. While parallel scheduling offers the potential for improved performance and throughput, it also introduces challenges such as the need for synchronization mechanisms to ensure data consistency and the risk of resource contention among concurrently executing transactions. Overall, both serial and parallel scheduling techniques in DBMS play important roles in optimizing system performance and ensuring efficient resource utilization, with each approach offering its own set of advantages and trade-offs depending on the specific requirements and characteristics of the database system.

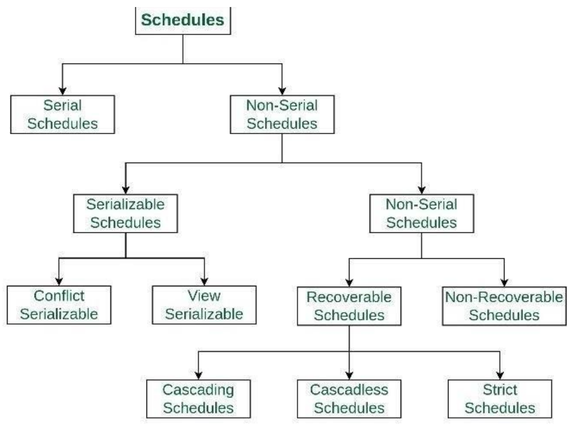


Fig 12: showing different types of scheduling

5.2 Concurrency Control

In our project, concurrency control is pivotal to ensuring data consistency and integrity within the Diagnostic Management Laboratory System (DMLS). Given the concurrent access and modification of patient records, test assignments, and staff activities, robust concurrency control mechanisms are imperative. Through techniques such as locking and timestamp-based protocols, transactions acquire exclusive access to data, preventing conflicts and maintaining accuracy. Additionally, isolation levels like Read Committed or Serializable enforce strict data visibility rules, ensuring transactions only access consistent and committed data. By implementing effective concurrency control, we safeguard against inconsistencies, enhance reliability, and optimize the efficiency of diagnostic lab operations, ensuring the seamless and accurate management of patient information.

5.3 Transaction Control and Recovery

In our project, transaction control and recovery mechanisms are fundamental components of the Diagnostic Management Laboratory System (DMLS), ensuring data consistency and reliability. Transaction control involves managing the execution of database transactions, ensuring that they are atomic, consistent, isolated, and durable (ACID properties). By adhering to these principles, we guarantee that transactions either complete successfully, leaving the database in a consistent state, or are rolled back to their original state in the event of failure. Additionally, recovery mechanisms are implemented to restore the database to a consistent state after unexpected events, such as system crashes or hardware failures. Techniques such as transaction logging, checkpoints, and database backups are utilized to facilitate recovery and minimize data loss. By integrating robust transaction control and recovery mechanisms into the DMLS, we ensure the integrity and reliability of patient records, test results, and other critical data, thereby enhancing the overall stability and resilience of the system.

CHAPTER – 6

6.1 Application Program Interface

6.1.1 Login

```
package dlm; import java.awt.EventQueue; import
javax.swing.JFrame; import javax.swing.JLabel; import
javax.swing.JOptionPane; import
javax.swing.ImageIcon; import java.awt.Color; import
javax.swing.SwingConstants; import java.awt.Font;
import javax.swing.JTextField; import
javax.swing.JPasswordField; import
javax.swing.JButton; import
java.awt.event.ActionListener; import java.sql.Statement;
import java.awt.event.ActionEvent; import java.sql.*;
public class login { private static final mysqlconnection
NULL = null; JFrame frmLoginPage; private JTextField
textFieldu_name; private JPasswordField
passwordFieldpwd; public static void main(String[] args)
{
        EventQueue.invokeLater(new Runnable() { public void run()
            { try { login window = new login();
            window.frmLoginPage.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
    Connection connection=null; public
    login() {
        initialize();
        connection=mysqlconnection.dbconnector();
```

```
} private void initialize() { frmLoginPage = new JFrame();
frmLoginPage.setTitle("LOGIN PAGE"); frmLoginPage.setBounds(100,
100, 545, 443);
frmLoginPage.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frmLoginPage.getContentPane().setLayout(null);

        JLabel lblNewLabel_1 = new JLabel("ADMIN LOGIN");
        lblNewLabel_1.setFont(new Font("Times New Roman", Font.BOLD,
        14)); lblNewLabel_1.setIcon(null);
        lblNewLabel_1.setBackground(Color.RED);
        lblNewLabel_1.setForeground(Color.BLACK);
        lblNewLabel_1.setLabelFor(lblNewLabel_1);
        lblNewLabel_1.setBounds(179, 46, 116, 48);
        frmLoginPage.getContentPane().add(lblNewLabel_1);

        JLabel lblNewLabel = new JLabel("USERNAME");
        lblNewLabel.setBackground(new Color(0, 0, 0));
        lblNewLabel.setForeground(new Color(0, 128, 128));
        lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD, 14));
        lblNewLabel.setBounds(22, 107, 94, 40);
        frmLoginPage.getContentPane().add(lblNewLabel);

        JLabel lblNewLabel_2 = new JLabel("PASSWORD");
        lblNewLabel_2.setBackground(Color.BLACK);
        lblNewLabel_2.setForeground(new Color(0, 128, 128));
        lblNewLabel_2.setFont(new Font("Times New Roman", Font.BOLD, 14));
        lblNewLabel_2.setBounds(22, 175, 94, 29);
        frmLoginPage.getContentPane().add(lblNewLabel_2);
```

```
textFieldu_name = new JTextField(); textFieldu_name.setBounds(155,
118, 148, 20); frmLoginPage.getContentPane().add(textFieldu_name);
textFieldu_name.setColumns(10);

passwordFieldpwd = new JPasswordField();
passwordFieldpwd.setBounds(155, 180, 148, 20);
frmLoginPage.getContentPane().add(passwordFieldpwd);

JButton btnNewButton = new JButton("LOGIN");
btnNewButton.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent arg0)
        { try { int count=0;
                //Statement stmt=connection.createStatement();
String query="select * from dlm.adm where u_name=? and       pwd=?";
                        PreparedStatement
pst=connection.prepareStatement(query);
                        pst.setString(1,textFieldu_name.getText());
                        pst.setString(2,passwordFieldpwd.getText());
                        ResultSet rs=pst.executeQuery();
                        while(rs.next()) { count++;
                        }
                        if(count==0)
                        {
                 JOptionPane.showMessageDialog(null,"INVALID ADMIN");
                        }
                        if(count==1)
                        {
                        frmLoginPage.dispose();
                        JOptionPane.showMessageDialog(null,"login successfull");
                        admin_view adm=new admin_view(); adm.setVisible(true);
```

```
                                        } pst.close();
                                                rs.close();
                                        }catch(Exception E) {
                                                E.printStackTrace();
                                        }
                        }
                });


                btnNewButton.setBounds(189, 252, 89, 23);
                frmLoginPage.getContentPane().add(btnNewButton);

                JLabel lblNewLabel_4 = new JLabel("");
                lblNewLabel_4.setIcon(new
ImageIcon(login.class.getResource("/dlm/images/LABIMAGE.jpg")));
                lblNewLabel_4.setBounds(0, 0, 529, 404);
                frmLoginPage.getContentPane().add(lblNewLabel_4);

        }
}
```

6.1.2 Admin_view

```
package   dlm;   import   java.sql.*;   import
javax.swing.JOptionPane;                import
java.awt.BorderLayout;                  import
java.awt.EventQueue;                    import
javax.swing.JFrame;                     import
javax.swing.JPanel;                     import
javax.swing.border.EmptyBorder;         import
net.proteanit.sql.DbUtils;              import
javax.swing.JButton;   import   java.awt.Font;
```

```
import        javax.swing.JLabel;        import
javax.swing.JOptionPane;                 import
javax.swing.ImageIcon;                   import
java.awt.event.ActionListener;           import
java.awt.event.ActionEvent;              import
java.awt.event.MouseAdapter;             import
java.awt.event.MouseEvent;    public    class
admin_view extends JFrame { private JPanel
contentPane; private JFrame frmLoginPage;
public static void main(String[] args) {
                EventQueue.invokeLater(new Runnable() {
                        public void run() { try {
                                        admin_view frame = new admin_view();
                                        frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }
Connection     connection=null;     public     admin_view()     {
        connection=mysqlconnection.dbconnector();
        setTitle("diagnostic                          laboratory");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 764, 480); contentPane = new JPanel();
        contentPane.setBorder(new   EmptyBorder(5,   5,   5,   5));
        setContentPane(contentPane); contentPane.setLayout(null);
                JButton patientbutton = new JButton("PATIENT");
                patientbutton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) {
                dispose(); patient_view patient=new patient_view();
                patient.setVisible(true);
```

```
            }
        });
        patientbutton.setFont(new Font("Times New Roman", Font.BOLD, 14));
        patientbutton.setBounds(55, 93, 185, 44);
        contentPane.add(patientbutton);

        JButton testbutton = new JButton("TEST"); testbutton.addActionListener(new
        ActionListener() { public void actionPerformed(ActionEvent e) { dispose();
        test t=new test();
                t.setVisible(true);
            }
        });
        testbutton.setFont(new Font("Times New Roman", Font.BOLD,
        14)); testbutton.setBounds(55, 180, 185, 44);
        contentPane.add(testbutton);
        JButton    staffbutton    =    new    JButton("STAFF");
        staffbutton.addActionListener(new ActionListener() {
        public    void    actionPerformed(ActionEvent    e)    {
        dispose();    staff_page    st=new    staff_page();
        st.setVisible(true);
            }
        });
        staffbutton.setFont(new Font("Times New Roman", Font.BOLD, 14));
        staffbutton.setBounds(55, 266, 185, 44); contentPane.add(staffbutton);

        JButton    return_loginbutton    =    new    JButton("LOGOUT");
        return_loginbutton.addActionListener(new ActionListener() { public void
        actionPerformed(ActionEvent        e)        {        int
        n=JOptionPane.showConfirmDialog(null, "do you want to
logout","logout",JOptionPane.YES_NO_OPTION);
                if(n==0)
```

```
                                    { dispose(); login log=new login();
                                            log.frmLoginPage.setVisible(true)
                                            ;

                                    }

                            }

                    });
                    return_loginbutton.setFont(new Font("Times New Roman", Font.BOLD, 14));
                    return_loginbutton.setBounds(259, 391, 122, 23);
                    contentPane.add(return_loginbutton);
                    JLabel lblNewLabel = new JLabel("welcome to diagnostic laboratory");
                    lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD, 23));
                    lblNewLabel.setBounds(121, 30, 344, 33);
                    contentPane.add(lblNewLabel);
                    JButton btnNewButton = new JButton("REPORT");
                    btnNewButton.addActionListener(new ActionListener()
                    { public void actionPerformed(ActionEvent e) {
                    dispose(); report r=new report(); r.setVisible(true);
//                                  try {
//                                          String n=JOptionPane.showInputDialog("patient_id");
//                  String que="select
h.p_id,p.p_name,h.r_id,r.test_result,tg.t_id,t.test_name,t.test_charge,cb.s_id,s.s_name from holds
h,report r,patient p,staff s,t_generates tg,test t,completed_by cb where h.p_id=? and
p.p_id=h.p_id and r.r_id=h.r_id and tg.r_id=r.r_id and t.t_id=tg.t_id and cb.t_id=t.t_id and
s.s_id=cb.s_id;";
//                                          PreparedStatement pst=connection.prepareStatement(que);
//                                          pst.setString(1,n);
//                                          ResultSet rs=pst.executeQuery();
//                                          dispose();
//                                          report r=new report(rs);
//                                          r.setVisible(true);
//
//
//                                  }catch(Exception ec) {
```

```
//                                              ec.printStackTrace();
//                                    }
                          }
                    });
                    btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
                    btnNewButton.setBounds(455, 180, 137, 44);
                    contentPane.add(btnNewButton);
                    JButton btnNewButton_1 = new JButton("payments");
                    btnNewButton_1.addActionListener(new ActionListener()
                    { public void actionPerformed(ActionEvent arg0) {
                    dispose(); payments p=new payments(); p.setVisible(true);
                          }
                    });
                    btnNewButton_1.setFont(new Font("Times New Roman", Font.BOLD,
                    14)); btnNewButton_1.setBounds(455, 278, 137, 32);
                    contentPane.add(btnNewButton_1); JLabel lblNewLabel_1 = new
                    JLabel(""); lblNewLabel_1.setIcon(new
ImageIcon(admin_view.class.getResource("/dlm/images/adminpage.jpg")))
          ; lblNewLabel_1.setBounds(0, 0, 748, 441);
          contentPane.add(lblNewLabel_1); }
}
```

6.1.3 patient_view

```
package              dlm;              import
java.awt.BorderLayout;                 import
java.awt.EventQueue;                   import
javax.swing.JFrame;                    import
javax.swing.JPanel;                    import
javax.swing.border.EmptyBorder;        import
net.proteanit.sql.DbUtils;             import
javax.swing.JTable; import java.awt.Color;
```

```
import      javax.swing.JButton;      import
java.sql.*;                           import
javax.swing.JOptionPane;              import
java.awt.event.ActionListener;        import
java.awt.event.ActionEvent;           import
javax.swing.JFormattedTextField;      import
java.awt.Font;                        import
javax.swing.JScrollPane;              import
javax.swing.JLabel;                   import
javax.swing.Box;                      import
javax.swing.ImageIcon;                import
javax.swing.JTextField;    public     class
patient_view   extends   JFrame   {   private
JPanel contentPane; private JTable table;
        public static void main(String[] args) {
                EventQueue.invokeLater(new Runnable() { public void run() {
                        try   {   patient_view   frame   =   new   patient_view();
                        frame.setVisible(true);

                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }
        Connection        connection=null;
        private JTextField textFieldsearch;
        public void refreshtable() { try {
                        String query="select * from patient";


                        PreparedStatement pst=connection.prepareStatement(query);
                        ResultSet rs=pst.executeQuery();
```

```
                    table.setModel(DbUtils.resultSetToTableModel(rs));

                    pst.close();
                    rs.close();
                }catch(Exception EX){
                    EX.printStackTrace();
                }
        } public void addpat(String p_id,String test_id,String patient_name, String age,
String address,
String phone_no,String gender) { try
                {
                    String query = "insert into patient (p_id,t_id,p_name, age, address, ph_no,
gender) values (?,?,?,?,?,?,?)";
                    PreparedStatement pst = connection.prepareStatement(query);
                    pst.setString(1, p_id); pst.setString(2, test_id);
                    pst.setString(3,
                    patient_name);
                    pst.setString(4, age);
                    pst.setString(5, address);
                    pst.setString(6, phone_no);
                    pst.setString(7, gender);
                    pst.execute(); pst.close();
                } catch (SQLException e1) { e1.printStackTrace();
                }
        } public void uppat(String p_id,String test_id,String patient_name, String age,
String address,
String phone_no,String gender) { try
        {
                String query = "update patient set t_id=?,p_name=?, age=?, address=?, ph_no=?,
gender=? where p_id=?";
```

```
            PreparedStatement pst = connection.prepareStatement(query);
            pst.setString(1,    test_id);    pst.setString(2,    patient_name);
            pst.setString(3, age); pst.setString(4, address); pst.setString(5,
            phone_no);  pst.setString(6,  gender);  pst.setString(7,  p_id);
            pst.executeUpdate(); pst.close();
        } catch (SQLException e1) {
        e1.printStackTrace(); }


    }
        public patient_view() { setTitle("patient_view");
            connection=mysqlconnection.dbconnector();
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
            ; setBounds(100, 100, 827, 420); contentPane = new
            JPanel(); contentPane.setBorder(new EmptyBorder(5,
            5, 5, 5)); setContentPane(contentPane);
            contentPane.setLayout(null);
            JScrollPane scrollPane = new JScrollPane();
            scrollPane.setBounds(10,   74,   791,   124);
            contentPane.add(scrollPane);  table  =  new
            JTable();
            scrollPane.setViewportView(table);
            JButton btnNewButton = new JButton("Add patientinfo");
            btnNewButton.setFont(new Font("Times New Roman", Font.BOLD,
            13)); btnNewButton.addActionListener(new ActionListener() { public
            void actionPerformed(ActionEvent arg0) { try {
                                    JTextField p_id = new JTextField(10);
                                    JTextField test_id = new JTextField(10);
                                    JTextField patient_name = new JTextField(10);
                              JTextField age = new JTextField(10);
                              JTextField address = new JTextField(10);
                              JTextField phone = new JTextField(10);
```

```
                                        JTextField gender = new JTextField(10);

                                        JPanel myPanel = new JPanel();
                                        myPanel.add(new JLabel("p_id:"));
                                        myPanel.add(p_id);
                                        myPanel.add(Box.createVerticalStrut(15));
                                        myPanel.add(new JLabel("test_id:"));
                                        myPanel.add(test_id);
                                        myPanel.add(Box.createVerticalStrut(15));
                                        myPanel.add(new JLabel("patient_name:"));
                                        myPanel.add(patient_name);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("age:"));
                                        myPanel.add(age);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("address:"));
                                        myPanel.add(address);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("Phone:"));
                                        myPanel.add(phone);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("gender:"));
                                        myPanel.add(gender); int result =
                                        JOptionPane.showConfirmDialog(null, myPanel,
                                                  "Please Enter X and Y Values",
JOptionPane.OK_CANCEL_OPTION); if (result == JOptionPane.OK_OPTION) {
                                        if(p_id.getText().matches("[0-9]+") == false) {
                                                            JOptionPane.showMessageDialog(null,
 "Enter A Valid patient_id");

                                                  } else if(test_id.getText().matches("[0-9]+") ==
false) {
```

```
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid test_id");
                                                      } else if(patient_name.getText().matches("[a-zA-
Z]+")== false) {
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid patient name");
            } else if(age.getText().matches("[0-9]+") == false) {
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid age");
            } else if(phone.getText().matches("[0-9]+")==false) {
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid phone");


                                                      }
            else if(phone.getText().length()!=10) {
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid phone");
                                                            }
            else if(gender.getText().matches("[a-zA-Z]+") == false) {
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid gender");
            }else {

      addpat(p_id.getText(),test_id.getText(),patient_name.getText(), age.getText(),
address.getText(), phone.getText(), gender.getText());
                                                JOptionPane.showMessageDialog(null, "patientinfo added
successfully");

                                                refreshtable();
                                                      }
                                          }
```

```
                                        }

                                      catch(Exception Ec) {
                                                Ec.printStackTrace();
                                      }
                            }
                  });
                  btnNewButton.setBounds(27, 227, 137, 43); contentPane.add(btnNewButton);

                  JButton btnNewButton_1 = new JButton("Delete patientinfo");
                  btnNewButton_1.addActionListener(new   ActionListener()   {
                  public void actionPerformed(ActionEvent arg0) {
//                                    contentPane.removeAll();
//                                    delete del= new delete();
//                                    del.setVisible(true);s

                                      try {

                                                String n=JOptionPane.showInputDialog("patient_id");
                                                String  q="select  p_id  from  patient  where  p_id=?";
                                                PreparedStatement pt=connection.prepareStatement(q);
                                                pt.setString(1,n);
                                                ResultSet s=pt.executeQuery();
                                                while(s.next()==false) {
in database");                                            JOptionPane.showMessageDialog(null, "patient not

                                                          break;
                                                }
//                                                        if(s!=n) {
//                                                                  JOptionPane.showMessageDialog(null,
"patient not in database");
//                                                        }
```

```
                                    String que="delete from patient where p_id=? ";
                                    PreparedStatement pst=connection.prepareStatement(que);
                                    pst.setString(1,n);

int rs=pst.executeUpdate();    if(n.isEmpty()) {
                                     JOptionPane.showMessageDialog(null, "enter valid patient

id");
                                    }

                                    else {
                                            JOptionPane.showMessageDialog(null, "patient

info deleted");

                                    }


                                    refreshtable();

                              }catch(Exception ec) { ec.printStackTrace();
                              }
                        }
                  });
                  btnNewButton_1.setFont(new Font("Times New Roman", Font.BOLD, 13));
                  btnNewButton_1.setBounds(224, 227, 137, 43);
                  contentPane.add(btnNewButton_1);
                  JLabel lblNewLabel = new JLabel("patient table");
                  lblNewLabel.setForeground(new Color(255, 255, 0));
                  lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD, 25));
                  lblNewLabel.setBounds(155, 27, 149, 36);
                  contentPane.add(lblNewLabel);
```

```
                    JButton btnNewButton_2 = new JButton("back to admin_view");
                    btnNewButton_2.setFont(new Font("Times New Roman", Font.BOLD,
                    12)); btnNewButton_2.addActionListener(new ActionListener() { public
                    void actionPerformed(ActionEvent arg0) { dispose();

                                    admin_view adm=new admin_view();

                                    adm.setVisible(true);
                            }
                    });
                    btnNewButton_2.setBounds(598, 359, 150, 23);
                    contentPane.add(btnNewButton_2);

                    JButton btnNewButton_3 = new JButton("search patient through
                    p_id"); btnNewButton_3.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent arg0) { try {
                                            String query="select * from patient where p_id=?";


                                            PreparedStatement
pst=connection.prepareStatement(query);
                                            pst.setString(1,textFieldsearch.getText());
                                            ResultSet rs=pst.executeQuery();
                                            table.setModel(DbUtils.resultSetToTableModel(rs));


                                            pst.close();
                                            rs.close();
                                    }catch(Exception EX){
                                            EX.printStackTrace();
                                    }


                            }
                    });
```

```
btnNewButton_3.setFont(new Font("Times New Roman", Font.BOLD, 13));
btnNewButton_3.setBounds(611, 235, 190, 43);
contentPane.add(btnNewButton_3);
textFieldsearch     =     new     JTextField();
textFieldsearch.setBounds(451, 250, 137, 20);
contentPane.add(textFieldsearch);
textFieldsearch.setColumns(10);

JButton btnNewButton_4 = new JButton("refresh");
btnNewButton_4.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) { refreshtable();
        }
});
btnNewButton_4.setFont(new Font("Times New Roman", Font.BOLD, 13));
btnNewButton_4.setBounds(565, 27, 137, 36);
contentPane.add(btnNewButton_4);

JButton btnNewButton_5 = new JButton("total number of
patients"); btnNewButton_5.addActionListener(new
ActionListener() { public void actionPerformed(ActionEvent arg0) {
int co=0; try {
                String query="call get_count_for_patient(@count);";

                                PreparedStatement
pst=connection.prepareStatement(query);
                ResultSet r=pst.executeQuery();
                String q="select @count";
                PreparedStatement ps=connection.prepareStatement(q);
                ResultSet n=ps.executeQuery(); while(n.next())
{ co=n.getInt(1);
}
                        JOptionPane.showMessageDialog(null,
                        co); pst.close(); r.close();
                        n.close();
                }catch(Exception EX){
                        EX.printStackTrace();
                }
        }
});
btnNewButton_5.setBounds(250, 359, 253, 23);
contentPane.add(btnNewButton_5);

JButton btnNewButton_6 = new JButton("Update
patientinfo"); btnNewButton_6.addActionListener(new
ActionListener() { public void actionPerformed(ActionEvent
arg0) { try {
                        JTextField p_id = new JTextField(10);
                        JTextField test_id = new JTextField(10);
                        JTextField patient_name = new JTextField(10);
                  JTextField age = new JTextField(10);
                  JTextField address = new JTextField(10);
                  JTextField phone = new JTextField(10);
                  JTextField gender = new JTextField(10);

                  JPanel myPanel = new JPanel();
                  myPanel.add(new JLabel("p_id:"));
                  myPanel.add(p_id);
                  myPanel.add(Box.createVerticalStrut(15));
                  myPanel.add(new JLabel("test_id:"));
                  myPanel.add(test_id);
                  myPanel.add(Box.createVerticalStrut(15));
```

```
                                        myPanel.add(new JLabel("patient_name:"));
                                        myPanel.add(patient_name);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("age:"));
                                        myPanel.add(age);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("address:"));
                                        myPanel.add(address);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("Phone:"));
                                        myPanel.add(phone);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("gender:"));
                                        myPanel.add(gender);

                                        int result = JOptionPane.showConfirmDialog(null, myPanel,
                                                  "Please Enter X and Y Values",
JOptionPane.OK_CANCEL_OPTION); if (result == JOptionPane.OK_OPTION) {
                                        if(p_id.getText().matches("[0-9]+") == false) {
                                                        JOptionPane.showMessageDialog(null,
 "Enter A Valid patient_id");
                                                  } else if(test_id.getText().matches("[0-9]+") ==
false) {
                                                        JOptionPane.showMessageDialog(null,
"Enter A Valid test_id");
                                                  } else if(patient_name.getText().matches("[a-zA-
Z]+")== false) {
                                                        JOptionPane.showMessageDialog(null,
    "Enter A Valid patient name");            } else if(age.getText().matches("[0-9]+") == false)

    {                                                 JOptionPane.showMessageDialog(null,

                                              } else if(phone.getText().matches("[0-9]+")==false)
    "Enter A Valid age");

                                                      JOptionPane.showMessageDialog(null,
    {

    "Enter A Valid phone");


                                              }
                                              else if(phone.getText().length()!=10) {
                                                      JOptionPane.showMessageDialog(null,
    "Enter A Valid phone");
                                                      } else if(gender.getText().matches("[a-zA-
false) {                                      Z]+") ==

                                                      JOptionPane.showMessageDialog(null,
    "Enter A Valid gender");

                                              }else {


            uppat(p_id.getText(),test_id.getText(),patient_name.getText(), age.getText(),
    address.getText(), phone.getText(), gender.getText());
                                      JOptionPane.showMessageDialog(null, "patientinfo added

    successfully");

                                      refreshtable();
                                              }
                                }
                                }
```

```
                    catch(Exception Ec) {
                            Ec.printStackTrace();
                    }



            }
        });
        btnNewButton_6.setFont(new Font("Times New Roman", Font.BOLD,
        14)); btnNewButton_6.setBounds(127, 299, 149, 30);
        contentPane.add(btnNewButton_6); JLabel lblNewLabel_13 = new
        JLabel(""); lblNewLabel_13.setIcon(new
ImageIcon(report.class.getResource("/dlm/images/patient_viewpage image.jpg")));
        lblNewLabel_13.setBounds(0,         0,         900,         650);
        contentPane.add(lblNewLabel_13); refreshtable();
    }
}
```

6.1.4 Payments

```
package dlm; import java.sql.*; import
javax.swing.JOptionPane;          import
java.awt.BorderLayout;            import
java.awt.EventQueue;              import
javax.swing.JFrame;               import
javax.swing.JPanel;               import
javax.swing.border.EmptyBorder; import
net.proteanit.sql.DbUtils;        import
javax.swing.JTable;               import
javax.swing.JScrollPane;
```

```
import         javax.swing.JLabel;         import
java.awt.Font;   import   javax.swing.JButton;
import  java.awt.event.ActionListener;  import
java.awt.event.ActionEvent;               import
javax.swing.ImageIcon;                    import
java.awt.Color; public class payments extends
JFrame  {  private  JPanel  contentPane;  public
static void main(String[] args) {
                EventQueue.invokeLater(new  Runnable()  {  public  void
                        run()  {  try  {  payments  frame  =  new  payments();
                        frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        } public void refreshtable()
        { try {
                        String query="select
p.p_id,p.p_name,s.s_name,t.t_id,t.test_name,t.test_charge,r.r_id,r.test_result,z.bill_no,z.mode_of
_pay from patient p,test t,staff s,completed_by cb,report r,payment z,holds h where
h.p_id=p.p_id and r.r_id=h.r_id and t.t_id=r.t_id and cb.t_id=t.t_id and s.s_id=cb.s_id ";

                        PreparedStatement
                        pst=connection.prepareStatement(query);              ResultSet
                        rs=pst.executeQuery();
                        table.setModel(DbUtils.resultSetToTableModel(rs));
                        pst.close(); rs.close();
                }catch(Exception EX){
                        EX.printStackTrace();
                }
        }
```

```
Connection connection=null; private JTable table; public
payments() { connection=mysqlconnection.dbconnector();
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(100, 100, 849, 503); contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane); contentPane.setLayout(null);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setBounds(10,    82,    813,    80);
        contentPane.add(scrollPane);

        table = new JTable(); scrollPane.setViewportView(table);

        JLabel lblNewLabel = new JLabel("payments");
        lblNewLabel.setForeground(Color.GREEN); lblNewLabel.setFont(new
        Font("Times New Roman", Font.BOLD, 18));
        lblNewLabel.setBounds(354, 25, 87, 22);
        contentPane.add(lblNewLabel);
        JButton btnNewButton = new JButton("back to admin
        page"); btnNewButton.addActionListener(new
        ActionListener() { public void actionPerformed(ActionEvent
        arg0) { dispose(); admin_view ad=new admin_view();
        ad.setVisible(true);
                }
        });
        btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 13));
        btnNewButton.setBounds(346, 373, 214, 52);
        contentPane.add(btnNewButton);
```

```
                    JLabel lblNewLabel_1 = new JLabel("");
                    lblNewLabel_1.setIcon(new
ImageIcon(payments.class.getResource("/dlm/images/staffupdateimage.jpg")))
                    ; lblNewLabel_1.setBounds(0, 0, 833, 464);
                    contentPane.add(lblNewLabel_1); refreshtable();
          }
}
```

6.1.5 report package dlm; import

```
java.sql.*;                       import
javax.swing.JOptionPane;          import
java.awt.BorderLayout;            import
java.awt.EventQueue;              import
javax.swing.JFrame;               import
javax.swing.JPanel;               import
javax.swing.border.EmptyBorder; import
net.proteanit.sql.DbUtils;        import
javax.swing.JLabel;               import
java.awt.Font;                    import
javax.swing.JTextField;           import
javax.swing.JButton;              import
java.awt.event.ActionListener;    import
java.awt.event.ActionEvent;       import
javax.swing.JTable;               import
javax.swing.JScrollPane;          import
java.awt.event.ContainerAdapter; import
java.awt.event.ContainerEvent;    import
java.awt.event.ComponentAdapter;
import  java.awt.event.ComponentEvent;
import javax.swing.JComboBox; import
javax.swing.JList;                import
```

```
java.awt.event.ItemListener;        import
java.awt.event.ItemEvent;           import
javax.swing.ImageIcon;              import
java.awt.Color;   public   class   report
extends   JFrame   {   private   JPanel
contentPane;     private     JComboBox
comboBoxpatientname;
        /**
         * Launch the application.
         */ public static void main(String[]
        args) {
                EventQueue.invokeLater(new Runnable() { public
                        void  run()  {  try  {  report  frame  =  new
                        report(); frame.setVisible(true);



                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }


        /**
         * Create the frame.
         */
        Connection     connection=null;     private
        JTextField       textFieldr_id;       private
        JTextField     textFields_name;     private
        JTextField       textFieldt_id;       private
        JTextField     textFieldt_name;     private
```

```
              JTextField    textFieldt_charge;    private
              JTextField     textFieldt_result;     private
              JTextField textFieldpatient_name; private
              JTextField        textFielddat;        private
              JTextField textFieldmode_of_pay; private
              JTextField       textFieldp_id;       private
              JTextField     textFieldbill_no;     private
              JTextField textFieldmode;
      public void fillcomboboxpatient() { try {

                          String qu="select * from patient";
                          PreparedStatement p=connection.prepareStatement(qu);
                          ResultSet r=p.executeQuery();

                          while(r.next()) { comboBoxpatientname.addItem(r.getString("p_id"));


                          }
                          r.close();
                          p.close();
                  }catch(Exception es) { es.printStackTrace();
                  }
              }


              public report() { connection=mysqlconnection.dbconnector();
                  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
                  ; setBounds(100, 100, 699, 593); contentPane = new
                  JPanel(); contentPane.setBorder(new EmptyBorder(5,
                  5, 5, 5)); setContentPane(contentPane);
                  contentPane.setLayout(null);
```

```
                    JLabel lblNewLabel = new JLabel("REPORT");
                    lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD,
                    20)); lblNewLabel.setBounds(272, 33, 90, 24);
                    contentPane.add(lblNewLabel); comboBoxpatientname = new
                    JComboBox();
                    comboBoxpatientname.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent arg0) { try {
                                            String q=" select
p.p_id,p.p_name,s.s_name,t.t_id,t.test_name,t.test_charge from patient p,test t,staff
s,completed_by cb where p.p_id=? and t.t_id=p.t_id and cb.t_id=t.t_id and s.s_id=cb.s_id ";

                                            PreparedStatement   ps=connection.prepareStatement(q);
                                    ps.setString(1,(String)comboBoxpatientname.getSelectedItem())
                                    ; ResultSet r=ps.executeQuery();

                                            while(r.next()) {

                                                    textFieldp_id.setText(r.getString("p_id"));


            textFieldpatient_name.setText(r.getString("p_name"));
                                                    textFields_name.setText(r.getString("s_name"));
                                                    textFieldt_id.setText(r.getString("t_id"));
                                                    textFieldt_name.setText(r.getString("test_name"));

            textFieldt_charge.setText(r.getString("test_charge"));



                                            }


                                    }catch(Exception ec) { ec.printStackTrace();
                                    }
```

```
        }
    });
    comboBoxpatientname.setBounds(234, 83, 128, 20);
    contentPane.add(comboBoxpatientname);

    JLabel lblNewLabel_2 = new JLabel("Choose patient id");
    lblNewLabel_2.setFont(new Font("Times New Roman", Font.BOLD, 15));
    lblNewLabel_2.setBounds(67, 71, 157, 14);
    contentPane.add(lblNewLabel_2);

    JLabel lblNewLabel_4 = new JLabel("report_id");
    lblNewLabel_4.setFont(new Font("Times New Roman", Font.BOLD, 15));
    lblNewLabel_4.setBounds(341, 154, 83, 20);
    contentPane.add(lblNewLabel_4);

    JLabel lblNewLabel_5 = new JLabel("staff name");
    lblNewLabel_5.setFont(new Font("Times New Roman", Font.BOLD, 15));
    lblNewLabel_5.setBounds(24, 245, 83, 24);
    contentPane.add(lblNewLabel_5);

    JLabel lblNewLabel_6 = new JLabel("test_id");
    lblNewLabel_6.setFont(new Font("Times New Roman", Font.BOLD, 15));
    lblNewLabel_6.setBounds(21, 289, 65, 24);
    contentPane.add(lblNewLabel_6);

    JLabel lblNewLabel_7 = new JLabel("test_name");
    lblNewLabel_7.setFont(new Font("Times New Roman", Font.BOLD, 15));
    lblNewLabel_7.setBounds(22, 324, 85, 24);
    contentPane.add(lblNewLabel_7);
```

```
JLabel lblNewLabel_8 = new JLabel("test_charge");
lblNewLabel_8.setFont(new Font("Times New Roman", Font.BOLD, 15));
lblNewLabel_8.setBounds(22, 366, 83, 24);
contentPane.add(lblNewLabel_8);

JLabel lblNewLabel_9 = new JLabel("test_result"); lblNewLabel_9.setFont(new
Font("Times New Roman", Font.BOLD, 15)); lblNewLabel_9.setBounds(341,
197, 98, 18); contentPane.add(lblNewLabel_9);

textFieldr_id = new JTextField();
textFieldr_id.setBounds(443, 155, 157, 20);
contentPane.add(textFieldr_id);
textFieldr_id.setColumns(10);

textFields_name = new JTextField();
textFields_name.setBounds(115, 248, 143, 20);
contentPane.add(textFields_name);
textFields_name.setColumns(10);

textFieldt_id = new JTextField();
textFieldt_id.setBounds(115, 292, 143, 20);
contentPane.add(textFieldt_id);
textFieldt_id.setColumns(10);

textFieldt_name = new JTextField();
textFieldt_name.setBounds(115, 327, 143, 20);
contentPane.add(textFieldt_name);
textFieldt_name.setColumns(10);

textFieldt_charge = new JTextField();
textFieldt_charge.setBounds(115, 369, 143, 20);
```

```
contentPane.add(textFieldt_charge);
textFieldt_charge.setColumns(10);

textFieldt_result      =      new      JTextField();
textFieldt_result.setBounds(443, 195, 157, 20);
contentPane.add(textFieldt_result);
textFieldt_result.setColumns(10);


JLabel lblNewLabel_10 = new JLabel("patient_name");
lblNewLabel_10.setFont(new Font("Times New Roman", Font.BOLD, 15));
lblNewLabel_10.setBounds(22, 152, 98, 24);
contentPane.add(lblNewLabel_10);

textFieldpatient_name = new JTextField();
textFieldpatient_name.setBounds(118, 155, 140, 20);
contentPane.add(textFieldpatient_name);
textFieldpatient_name.setColumns(10);

JLabel lblNewLabel_11 = new JLabel("Date"); lblNewLabel_11.setFont(new
Font("Times New Roman", Font.BOLD, 14));
lblNewLabel_11.setBounds(341, 254, 43, 14);
contentPane.add(lblNewLabel_11);

textFielddat      =      new      JTextField();
textFielddat.setBounds(443, 248, 157, 20);
contentPane.add(textFielddat);
textFielddat.setColumns(10);

textFieldp_id      =      new      JTextField();
textFieldp_id.setBounds(115, 197, 143, 20);
```

```
contentPane.add(textFieldp_id);
textFieldp_id.setColumns(10);

JLabel lblNewLabel_1 = new JLabel("Patient_id");
lblNewLabel_1.setFont(new Font("Times New Roman", Font.BOLD, 15));
lblNewLabel_1.setBounds(22, 198, 83, 17);
contentPane.add(lblNewLabel_1);

JButton  btnNewButton  =  new  JButton("PAY  BILL");
btnNewButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent arg0) { try {
                String g="insert into report values(?,?,?,?,?)";

                PreparedStatement ps=connection.prepareStatement(g);
                ps.setString(1,  textFieldp_id.getText());  ps.setString(2,
                textFieldr_id.getText());                    ps.setString(3,
                textFieldt_id.getText());                    ps.setString(4,
                textFielddat.getText()          );           ps.setString(5,
                textFieldt_result.getText());

                ps.executeUpdate();
                ps.close();
          }catch(Exception e) {
                e.printStackTrace();
          }
               try
          {
                String g="insert into payment values(?,?,?)";

                PreparedStatement
                ps=connection.prepareStatement(g);       ps.setString(1,
                textFieldr_id.getText());                    ps.setString(2,
                textFieldbill_no.getText());                 ps.setString(3,
                textFieldmode.getText());

                ps.executeUpdate();
                ps.close();
            }catch(Exception e) {
                e.printStackTrace();
            }
              try
            {
                String g="insert into holds values(?,?)";

                PreparedStatement
                ps=connection.prepareStatement(g);       ps.setString(1,
                textFieldp_id.getText());                    ps.setString(2,
                textFieldr_id.getText());



                ps.executeUpdate();
                ps.close();
            }catch(Exception e) {
                e.printStackTrace();
            }
            JOptionPane.showMessageDialog(null, "payment successfully");
            int n=JOptionPane.showConfirmDialog(null, "do you want to go
back to admin page","admin",JOptionPane.YES_NO_OPTION);
            if(n==0) {
                dispose();
```

```
        admin_view log=new admin_view();          log.setVisible(true); }
else {
                                        report r=new report();

                                        r.setVisible(true);
                                }
                        dispose();



                        }
                });
                btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
                btnNewButton.setBounds(312, 465, 157, 38);
                contentPane.add(btnNewButton);

                JLabel lblNewLabel_3 = new JLabel("Bill_no");
                lblNewLabel_3.setFont(new Font("Times New Roman", Font.BOLD, 15));
                lblNewLabel_3.setBounds(341, 299, 66, 20);
                contentPane.add(lblNewLabel_3);

                textFieldbill_no    =    new    JTextField();
                textFieldbill_no.setBounds(443, 292, 157, 20);
                contentPane.add(textFieldbill_no);
                textFieldbill_no.setColumns(10);

                JLabel lblNewLabel_12 = new JLabel("mode_of_payment");
                lblNewLabel_12.setFont(new Font("Times New Roman", Font.BOLD, 15));
                lblNewLabel_12.setBounds(341, 347, 128, 24);
                contentPane.add(lblNewLabel_12);

                textFieldmode    =    new    JTextField();
                textFieldmode.setBounds(479, 350, 121, 20);
                contentPane.add(textFieldmode);
                textFieldmode.setColumns(10);

                JButton btnNewButton_1 = new JButton("back to admin");
                btnNewButton_1.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) { dispose();
                                admin_view ad=new admin_view();
                                ad.setVisible(true);

                        }
                });
                btnNewButton_1.setFont(new Font("Times New Roman", Font.BOLD, 13));
                btnNewButton_1.setBounds(80, 471, 121, 29);
                contentPane.add(btnNewButton_1);

                JLabel lblNewLabel_13 = new JLabel("");
                lblNewLabel_13.setIcon(new
ImageIcon(report.class.getResource("/dlm/images/LABIMAGE.jpg")));
                lblNewLabel_13.setBounds(0, 0, 683, 554); contentPane.add(lblNewLabel_13);

                fillcomboboxpatient();


        }
}
6.1.6  staff_page  package  dlm;  import
java.sql.*;                       import
javax.swing.JOptionPane;          import
java.awt.BorderLayout;            import
java.awt.EventQueue;              import
javax.swing.JFrame;               import
javax.swing.JPanel;               import
javax.swing.border.EmptyBorder;   import
```

```
net.proteanit.sql.DbUtils;            import
javax.swing.JTable;                   import
javax.swing.JTextField;               import
javax.swing.JLabel;                   import
javax.swing.JScrollPane;              import
javax.swing.JButton;
import          java.awt.Font;             import
java.awt.event.ActionListener;             import
java.awt.event.ActionEvent;                import
javax.swing.Box;                           import
javax.swing.ImageIcon;       public       class
staff_page extends JFrame { private JPanel
contentPane; public static void main(String[]
args) {
                EventQueue.invokeLater(new Runnable() { public void
                        run() { try { staff_page frame = new staff_page();
                        frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }


        /**
         * Create the frame.
         *
         */Connection
         connection=null; public void
         refreshtable() { try {
                                String query="select * from staff";
                PreparedStatement
                pst=connection.prepareStatement(query);          ResultSet
                rs=pst.executeQuery();
                table.setModel(DbUtils.resultSetToTableModel(rs));
                pst.close(); rs.close();
            }catch(Exception EX){
                EX.printStackTrace();
            }
        } public void addstaff(String staff_id,String test_id, String
    staff_name, String
address,String age, String phone_no) {

            try {


                String query = "insert into staff (s_id,t_id,s_name,
address,age,ph_no) values (?,?,?,?,?,?)";
                PreparedStatement pst =
                connection.prepareStatement(query); pst.setString(1,
                staff_id); pst.setString(2, test_id); pst.setString(3,
                staff_name); pst.setString(4, address); pst.setString(5, age);
                pst.setString(6, phone_no);

                pst.execute();

                pst.close();

            } catch (SQLException e1) { e1.printStackTrace();
            }

        } public void upstaff(String staff_id,String test_id, String
    staff_name, String address,String age, String phone_no) {
```

```
try {

                                        String query = "update staff set t_id=?,s_name=?,
    address=?,age=?,ph_no=? where s_id=?";
                                        PreparedStatement pst = connection.prepareStatement(query);

                                        pst.setString(1, test_id);
                                        pst.setString(2, staff_name);
                                        pst.setString(3, address);
                                        pst.setString(4, age);
                                        pst.setString(5, phone_no);

                                        pst.setString(6, staff_id);

                                        pst.executeUpdate();

                                        pst.close();

                               } catch (SQLException e1) { e1.printStackTrace();
                               }

                      } public void completed_by(String test_id, String
              staff_id) {

                               try {

                                        String query = "insert into completed_by (t_id,s_id) values (?,?)";
                                        PreparedStatement pst = connection.prepareStatement(query);
                                        pst.setString(1, test_id); pst.setString(2,
                                        staff_id);
```

```
                        pst.execute();

                        pst.close();

                    } catch (SQLException e1) { e1.printStackTrace();
                    }


            } private JTable table; public staff_page() {
setTitle("staff info");
connection=mysqlconnection.dbconnector();
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(100, 100, 759, 477); contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane); contentPane.setLayout(null);

            JScrollPane scrollPane = new JScrollPane();
            scrollPane.setBounds(10,   82,   626,   123);
            contentPane.add(scrollPane);

            table = new JTable(); scrollPane.setViewportView(table);

            JLabel lblNewLabel = new JLabel("Staffinfo");
            lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD,
            22)); lblNewLabel.setBounds(204, 29, 100, 27);
            contentPane.add(lblNewLabel); refreshtable();
            JButton btnNewButton = new JButton("Add staff_info");
            btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) { try {
                            JTextField staff_id = new JTextField(10);
                            JTextField test_id = new JTextField(10); JTextField
                        staff_name = new JTextField(10);
```

```
                                        JTextField address = new JTextField(10);
                                        JTextField age = new JTextField(10);
                                        JTextField phone = new JTextField(10);

                                        JPanel myPanel = new JPanel(); myPanel.add(new
                                        JLabel("s_id:")); myPanel.add(staff_id);
                                        myPanel.add(Box.createVerticalStrut(15));
                                        myPanel.add(new JLabel("t_id:"));
                                        myPanel.add(test_id);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("s_name:"));
                                        myPanel.add(staff_name);
                                        myPanel.add(Box.createVerticalStrut(15)); // a
                                        spacer myPanel.add(new JLabel("address:"));
                                        myPanel.add(address);
                                        myPanel.add(Box.createVerticalStrut(15)); // a spacer
                                        myPanel.add(new                    JLabel("age:"));
                                        myPanel.add(age);
                                        myPanel.add(Box.createVerticalStrut(15)); // a spacer
                                        myPanel.add(new                 JLabel("ph_no:"));
                                        myPanel.add(phone);

                                        int result = JOptionPane.showConfirmDialog(null, myPanel,
                                                  "Please Enter X and Y Values",
JOptionPane.OK_CANCEL_OPTION);
                                        if (result == JOptionPane.OK_OPTION) {
                                            if(staff_id.getText().matches("[0-9]+") == false) {
                                                        JOptionPane.showMessageDialog(null,
"Enter A Valid staff_id");

                                                  } else if(test_id.getText().matches("[0-9]+") ==
false) {
```

```
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid test_id");
                                                    } else if(staff_name.getText().matches("[a-zA-
Z]+")== false) {
                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid staff name");
                                                     } else if(age.getText().matches("[0-9]+")==false) {

                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid age");


                                                    } else if(phone.getText().matches("[0-9]+")==false)
{
                                                            JOptionPane.showMessageDialog(null,

"Enter A Valid phone");


                                                    } else if(phone.getText().length()!=10) {

                                                            JOptionPane.showMessageDialog(null,
"Enter A Valid phone");
                                                    }else {
                                            addstaff(staff_id.getText(),test_id.getText(),
staff_name.getText(), address.getText(), age.getText(), phone.getText()); refreshtable();
                                            JOptionPane.showMessageDialog(null, "staffinfo added
successfully");

                                            completed_by(test_id.getText(),staff_id.getText());
                                        }       }



                                    }catch(Exception Ec) {
                                            Ec.printStackTrace();
```

```
                    }
                }
            });
            btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 15));
            btnNewButton.setBounds(20, 218, 213, 43);
            contentPane.add(btnNewButton);
            JButton btnNewButton_1 = new JButton("Delete
            staff_info"); btnNewButton_1.addActionListener(new
            ActionListener() { public void actionPerformed(ActionEvent
            e) { try {
                        String n=JOptionPane.showInputDialog("staff_id");
                        String q="select s_id from staff where s_id=?";

                        PreparedStatement pt=connection.prepareStatement(q);
                        pt.setString(1,n);

                        ResultSet s=pt.executeQuery();

                        while(s.next()==false) {
JOptionPane.showMessageDialog(null, "staffinfo not in database");
                            break;
                        }
                        String que="delete from staff where s_id=? ";
                        PreparedStatement pst=connection.prepareStatement(que);

                        pst.setString(1,n); int
                        rs=pst.executeUpdate();
                        if(n.isEmpty()) {
                            JOptionPane.showMessageDialog(null, "enter valid

  staff id");
                            } else
                            {
```

```
                                                    JOptionPane.showMessageDialog(null,
"staff info deleted");
                                                    }

                                    refreshtable();

                                    pst.close();

                            }catch(Exception ec) { ec.printStackTrace();
                            }
                    }
            });
            btnNewButton_1.setFont(new Font("Times New Roman", Font.BOLD, 15));
            btnNewButton_1.setBounds(285, 216, 213, 43);
            contentPane.add(btnNewButton_1);

            JButton btnNewButton_2 = new JButton("back to Admin page");
            btnNewButton_2.addActionListener(new    ActionListener()    {
            public void actionPerformed(ActionEvent e) {

                            dispose();

  admin_view ad=new admin_view();            ad.setVisible(true);
                    }
            });
            btnNewButton_2.setFont(new Font("Times New Roman", Font.BOLD, 15));
            btnNewButton_2.setBounds(273, 350, 213, 43);
            contentPane.add(btnNewButton_2);

            JButton btnNewButton_3 = new JButton("Update
            Staff_info"); btnNewButton_3.addActionListener(new
```

```
                    ActionListener() { public void actionPerformed(ActionEvent
                    arg0) { try {
                                        JTextField staff_id = new JTextField(10);
                                        JTextField test_id = new JTextField(10);
                                JTextField staff_name = new JTextField(10);
                                JTextField address = new JTextField(10);
                                JTextField age = new JTextField(10);
                                JTextField phone = new JTextField(10);

                                JPanel myPanel = new JPanel();
                                myPanel.add(new JLabel("s_id:"));
                                myPanel.add(staff_id);
                                myPanel.add(Box.createVerticalStrut(15));
                                myPanel.add(new JLabel("t_id:"));
                                myPanel.add(test_id);
                                myPanel.add(Box.createVerticalStrut(15)); // a spacer
                                myPanel.add(new JLabel("s_name:"));
                                myPanel.add(staff_name);
                                myPanel.add(Box.createVerticalStrut(15)); // a spacer
                                myPanel.add(new JLabel("address:")); myPanel.add(address);
                                myPanel.add(Box.createVerticalStrut(15)); // a spacer
                                myPanel.add(new JLabel("age:")); myPanel.add(age);
                                myPanel.add(Box.createVerticalStrut(15)); // a spacer
                                myPanel.add(new JLabel("ph_no:")); myPanel.add(phone);

                                int result = JOptionPane.showConfirmDialog(null, myPanel,
                                            "Please Enter X and Y Values",
            JOptionPane.OK_CANCEL_OPTION);
                                if (result == JOptionPane.OK_OPTION) {
                                    if(staff_id.getText().matches("[0-9]+") == false) {
                                                JOptionPane.showMessageDialog(null,
"Enter A Valid staff_id");
                                                    } else if(test_id.getText().matches("[0-9]+") ==
false) {
                                                             JOptionPane.showMessageDialog(null,
"Enter A Valid test_id");
                                                     } else if(staff_name.getText().matches("[a-zA-
Z]+")== false) {
                                                             JOptionPane.showMessageDialog(null,
"Enter A Valid staff name");
                                                      } else if(age.getText().matches("[0-9]+")==false) {

                                                             JOptionPane.showMessageDialog(null,
"Enter A Valid age");


                                                     } else if(phone.getText().matches("[0-9]+")==false)
{
                                                             JOptionPane.showMessageDialog(null,
"Enter A Valid phone");



                                                     } else if(phone.getText().length()==10) {

                                                             JOptionPane.showMessageDialog(null,
"Enter A Valid phone");


                                                     }else {
                                            addstaff(staff_id.getText(),test_id.getText(),
staff_name.getText(), address.getText(), age.getText(), phone.getText()); refreshtable();
                                            JOptionPane.showMessageDialog(null, "staffinfo added
successfully");

                                            completed_by(test_id.getText(),staff_id.getText());
```

```
                                        }       }


                                    }catch(Exception Ec) {
                                            Ec.printStackTrace();
                                    }
                            }
                    });
                    btnNewButton_3.setFont(new Font("Times New Roman", Font.BOLD, 14));
                    btnNewButton_3.setBounds(542, 216, 165, 43);
                    contentPane.add(btnNewButton_3);


                    JLabel lblNewLabel_13 = new JLabel("");
                    lblNewLabel_13.setIcon(new
ImageIcon(report.class.getResource("/dlm/images/staffinfobackgroundimage.jpg")));
                    lblNewLabel_13.setBounds(0, 0, 900, 900); contentPane.add(lblNewLabel_13);
            }
}   6.1.7   test   package   dlm;   import
java.sql.*;                        import
javax.swing.JOptionPane;           import
java.awt.BorderLayout;             import
java.awt.EventQueue;               import
javax.swing.JFrame;                import
javax.swing.JPanel;                import
javax.swing.border.EmptyBorder; import
net.proteanit.sql.DbUtils;         import
javax.swing.JLabel;                import
javax.swing.JOptionPane;           import
java.awt.Font;                     import
javax.swing.JTable;                import
javax.swing.JTextField;              import
javax.swing.JButton;                 import
javax.swing.JScrollPane;             import
java.awt.event.ActionListener;       import
java.awt.event.ActionEvent;          import
javax.swing.Box;                     import
javax.swing.ImageIcon; public class test
extends   JFrame   {   private   JPanel
contentPane; private JTable table;
        /**
         * Launch the application.
         */ public static void main(String[]
        args) {
                EventQueue.invokeLater(new Runnable() {
                        public void run() { try { test frame =
                                new test();
                                        frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }

        /**
         * Create the frame.
         */

        Connection connection=null;
        public void refreshtable() {
        try {
```

```
                              String query="select * from test";
                              PreparedStatement pst=connection.prepareStatement(query);
                              ResultSet rs=pst.executeQuery(); String
                              q="";
                              table.setModel(DbUtils.resultSetToTableModel(rs));

pst.close(); rs.close();
                    }catch(Exception EX){
                              EX.printStackTrace();
                    }
          } public void addtest(String test_id,String test_name, String
   test_charge) { try {


                              String query = "insert into test (t_id,test_name,test_charge) values (?,?,?)";
                              PreparedStatement pst = connection.prepareStatement(query);
                              pst.setString(1, test_id); pst.setString(2, test_name); pst.setString(3,
                              test_charge); pst.execute(); pst.close();
                    } catch (SQLException e1) { e1.printStackTrace();
                    }
   }
   public void uptest(String test_id,String test_name, String test_charge) {

try {
                    String query = "update test set test_name=?,test_charge=? where t_id=?";
                    PreparedStatement pst = connection.prepareStatement(query);
                    pst.setString(1, test_name); pst.setString(2, test_charge); pst.setString(3,
                    test_id); pst.executeUpdate(); pst.close();
          } catch (SQLException e1) {
                    e1.printStackTrace();
          }
```

```
}

        public test() {
                setTitle("test view");
                connection=mysqlconnection.dbconnector();

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
                ; setBounds(100, 100, 551, 400); contentPane = new
                JPanel(); contentPane.setBorder(new EmptyBorder(5,
                5, 5, 5)); setContentPane(contentPane);
                contentPane.setLayout(null);

                JLabel lblNewLabel = new JLabel("TEST"); lblNewLabel.setFont(new
                Font("Times New Roman", Font.BOLD, 19));
                lblNewLabel.setBounds(252, 27, 59, 23);
                contentPane.add(lblNewLabel);

                JScrollPane scrollPane = new JScrollPane();
                scrollPane.setBounds(10,   117,   516,   127);
                contentPane.add(scrollPane);

                table = new JTable(); scrollPane.setViewportView(table);

                JButton btnNewButton_2 = new JButton("back to admin
                page"); btnNewButton_2.addActionListener(new
                ActionListener() { public void actionPerformed(ActionEvent
                arg0) { dispose(); admin_view ad=new admin_view();
                ad.setVisible(true);
                        }
                });
                btnNewButton_2.setFont(new Font("Times New Roman", Font.BOLD, 15));
                btnNewButton_2.setBounds(360, 327, 165, 23);
                contentPane.add(btnNewButton_2);

                JButton btnNewButton = new JButton("Add Test");
                btnNewButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) { try {
                                JTextField test_id = new JTextField(10);
                                JTextField test_name = new JTextField(10);
                            JTextField test_charge = new JTextField(10);



                            JPanel myPanel = new JPanel(); myPanel.add(new
                            JLabel("test_id:")); myPanel.add(test_id);
                            myPanel.add(Box.createVerticalStrut(15));
                            myPanel.add(new JLabel("test_name:"));
                            myPanel.add(test_name);
                            myPanel.add(Box.createVerticalStrut(15)); // a
                            spacer myPanel.add(new JLabel("test_charge:"));
                            myPanel.add(test_charge);


                            int result = JOptionPane.showConfirmDialog(null, myPanel,
                                    "Please Enter X and Y Values",
JOptionPane.OK_CANCEL_OPTION);
                            if (result == JOptionPane.OK_OPTION) {

                                    if(test_id.getText().matches("[0-9]+") == false) {

    JOptionPane.showMessageDialog(null, "Enter A Valid test_id");
                                        } else if(test_name.getText().matches("[a-
zA-Z]+") == false) {


        JOptionPane.showMessageDialog(null, "Enter A Valid test_name");
                                            } else if(test_charge.getText().matches("[0-
9]+")== false) {


        JOptionPane.showMessageDialog(null, "Enter A Valid test_charge"); }
                                            else {
                                    addtest(test_id.getText(), test_name.getText(),
test_charge.getText());
                                    JOptionPane.showMessageDialog(null, "testinfo added
successfully");

                                    refreshtable();
                                            }


                                }
                            }catch(Exception Ec) {
                                    Ec.printStackTrace();
                            }
                    }
                });
                btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 13));
                btnNewButton.setBounds(57, 286, 105, 23);
                contentPane.add(btnNewButton);

                JButton btnNewButton_1 = new JButton("Delete Test");

 btnNewButton_1.addActionListener(new   ActionListener()   {     public   void
actionPerformed(ActionEvent arg0) { try {
        String n=JOptionPane.showInputDialog("test_id");
```

```
            String q="select t_id from test where t_id=?";
            PreparedStatement pt=connection.prepareStatement(q);
            pt.setString(1,n);
            ResultSet s=pt.executeQuery();
            while(s.next()==false) {
                JOptionPane.showMessageDialog(null, "testinfo not in database");
                break;
            }
                                        String que="delete from test where t_id=? ";
                                        PreparedStatement pst=connection.prepareStatement(que);


                                        pst.setString(1,n); int
                                        rs=pst.executeUpdate();
                                        if(n.isEmpty()) {
                                            JOptionPane.showMessageDialog(null, "enter valid
test id");

                                            }
                                            else
                                            {
                                                JOptionPane.showMessageDialog(null,  "test

  info deleted");
                                            }

                                        refreshtable();


                                    }catch(Exception ec) { ec.printStackTrace();
                                    }
                                }
                    });
                    btnNewButton_1.setFont(new  Font("Times  New  Roman",  Font.BOLD,  13));
                    btnNewButton_1.setBounds(195,          286,          105,          23);
                    contentPane.add(btnNewButton_1);
                    JButton btnNewButton_3 = new JButton("Update Test");
                    btnNewButton_3.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent arg0) { try {
                                        JTextField test_id = new JTextField(10);
                                        JTextField test_name = new JTextField(10);
                                   JTextField test_charge = new JTextField(10);


                                   JPanel myPanel = new JPanel(); myPanel.add(new
                                   JLabel("test_id:")); myPanel.add(test_id);
                                   myPanel.add(Box.createVerticalStrut(15));
                                   myPanel.add(new JLabel("test_name:"));
                                   myPanel.add(test_name);
                                   myPanel.add(Box.createVerticalStrut(15)); // a
                                   spacer myPanel.add(new JLabel("test_charge:"));
                                   myPanel.add(test_charge);


                                   int result = JOptionPane.showConfirmDialog(null, myPanel,
                                             "Please Enter X and Y Values",
JOptionPane.OK_CANCEL_OPTION); if (result == JOptionPane.OK_OPTION) { if (result
                                   == JOptionPane.OK_OPTION) {
                                   if(test_id.getText().matches("[0-9]+") == false) {

       JOptionPane.showMessageDialog(null, "Enter A Valid patient_id");
                                                  } else if(test_name.getText().matches("[a-
zA-Z]+") == false) {

       JOptionPane.showMessageDialog(null, "Enter A Valid test_id");
                                                  } else if(test_charge.getText().matches("[0-
9]+")== false) {
```

```
          JOptionPane.showMessageDialog(null, "Enter A Valid patient name"); }
                                                            else {
                                          uptest(test_id.getText(), test_name.getText(),
test_charge.getText());
                                          JOptionPane.showMessageDialog(null, "testinfo added
successfully");

                                          refreshtable();
                                                            }
                                    }
                                    }
                                 }catch(Exception Ec) {
                                          Ec.printStackTrace();
                                 }
                       }
                 });
                 btnNewButton_3.setFont(new Font("Times New Roman", Font.BOLD, 13));
                 btnNewButton_3.setBounds(360, 286, 105, 23);
                 contentPane.add(btnNewButton_3);

                 JLabel lblNewLabel_13 = new JLabel("");
                 lblNewLabel_13.setIcon(new
ImageIcon(report.class.getResource("/dlm/images/image.jpg")));
                 lblNewLabel_13.setBounds(0, 0, 650, 554); contentPane.add(lblNewLabel_13);
                 refreshtable();
                 }
          }
```

6.2 Connection To Database package

```
dlm;
import java.sql.*;

import javax.swing.JOptionPane;

public class mysqlconnection {


        Connection    con=null;    public    static
        Connection dbconnector(){ try {
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/dlm","root","12345678");
//                      JOptionPane.showMessageDialog(null, "connected");


                        return con;
        } catch(Exception e) {
                        JOptionPane.showMessageDialog(null, e);
                        return null;
                }


}
}
```

CHAPTER – 7

7.1 Results and Discussion

We have completed the project diagnostic laboratory management system successfully. We created graphical user interfaced application model in java using abstract window toolkit package and we used Swing packages and we handled the events using event handling package in our project. And we connected this API to database named dlm which is created In mysql workbench we used jdbc drivers and jar files to connect them to API. It was a great experience working on this project.

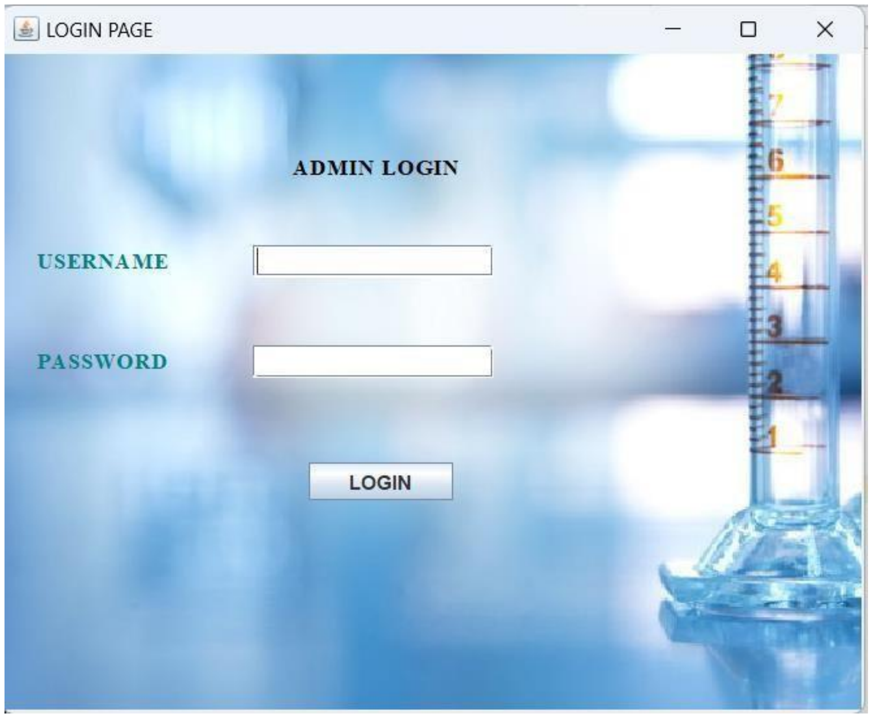


Fig 13 : admin page

Using this API we can login to work with this project. We may had a doubt that why we mentioned only login not register only reason is to increase security because if all the people working in the lab can register then their can be a chance of manipulation therefore we will be giving unique id and password to the owner and he mentioned people In the lab such that only that people can login

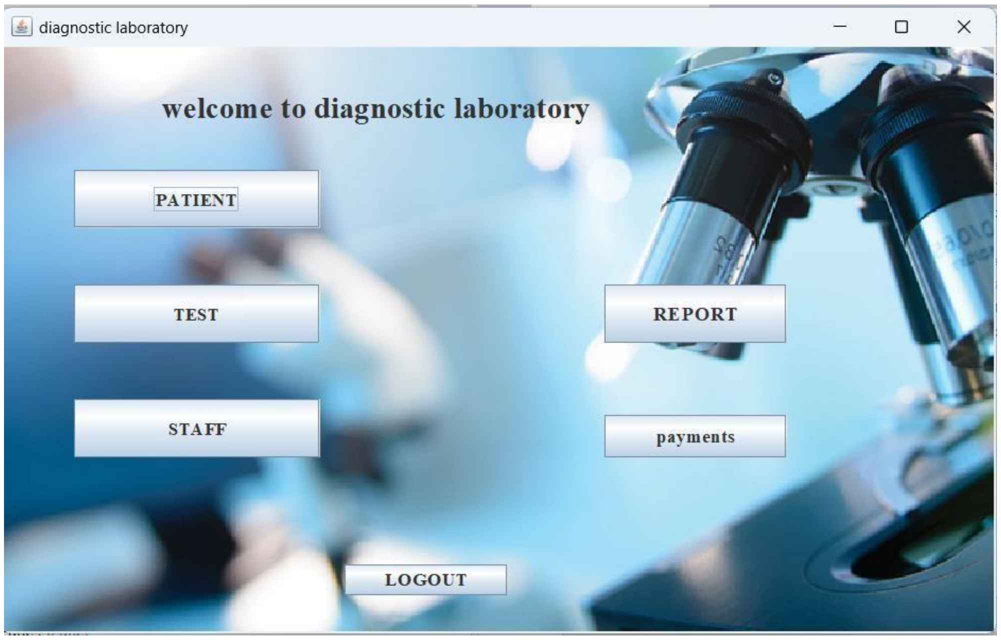
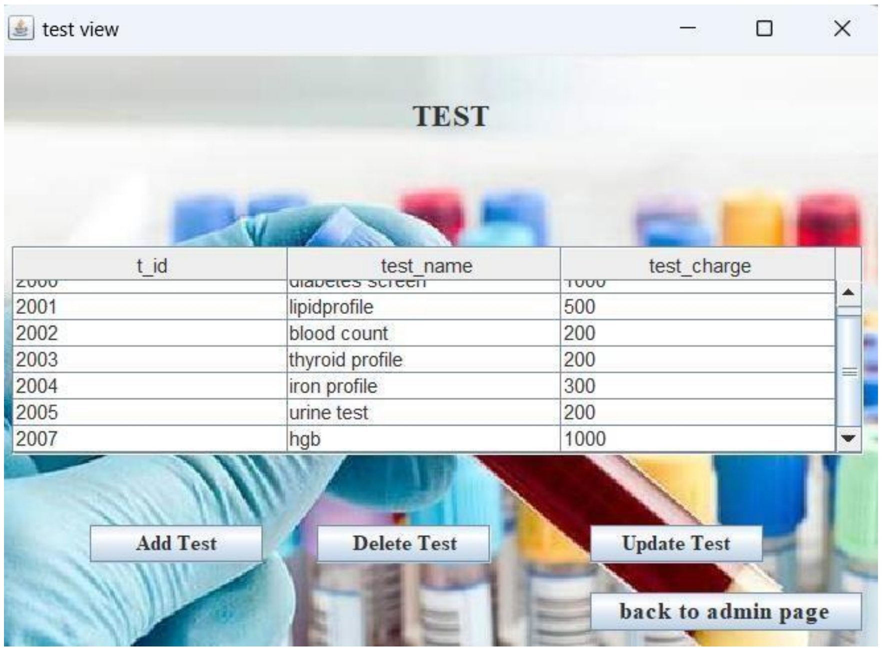


Fig 14 : admin view page

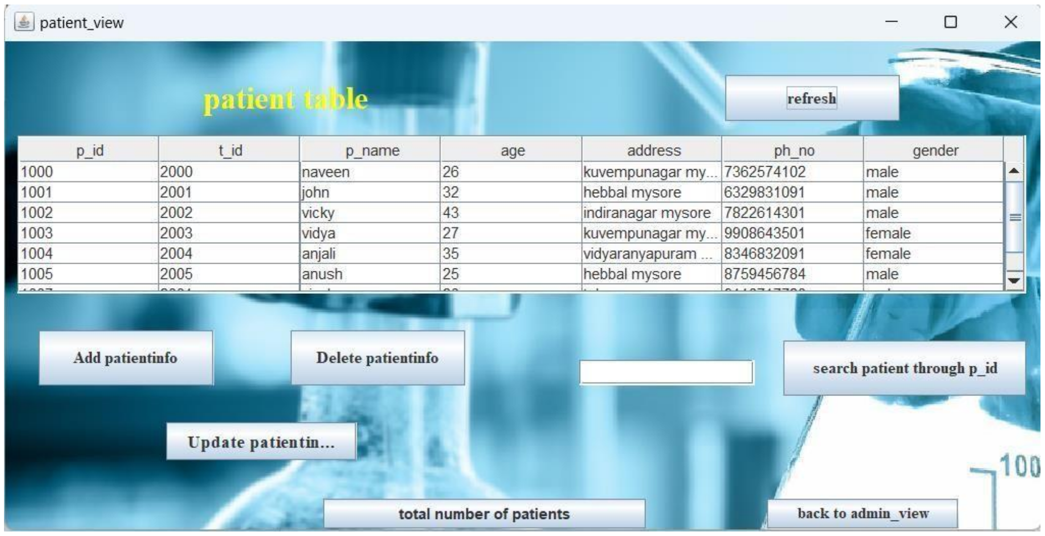
Here we can observe that in this API we have buttons like patient,test,report,staff,payments from all these we will be sent to different pages to work on. If we choose patients it asks to add patients,delete patients , update patients. Followed by test gives add test , update test.



t_id	test_name	test_charge
2000	diabetes screen	1000
2001	lipidprofile	500
2002	blood count	200
2003	thyroid profile	200
2004	iron profile	300
2005	urine test	200
2007	hgb	1000

Fig 15 : test page

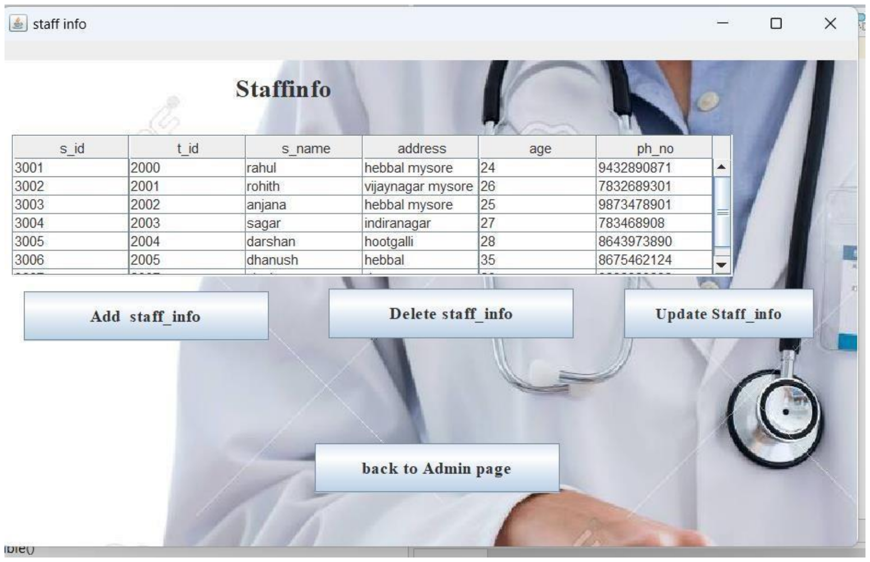
This page get evolved if we go to test window and here we can perform transactions like add test, delete test,update test. It was the main step to enter into project first of all admin add all the test available in their diagnostic lab then only we can go to further steps.



p_id	t_id	p_name	age	address	ph_no	gender
1000	2000	naveen	26	kuvempunagar my...	7362574102	male
1001	2001	john	32	hebbal mysore	6329831091	male
1002	2002	vicky	43	indiranagar mysore	7822614301	male
1003	2003	vidya	27	kuvempunagar my...	9908643501	female
1004	2004	anjali	35	vidyaranyapuram ...	8346832091	female
1005	2005	anush	25	hebbal mysore	8759456784	male

Fig 16 : patients page

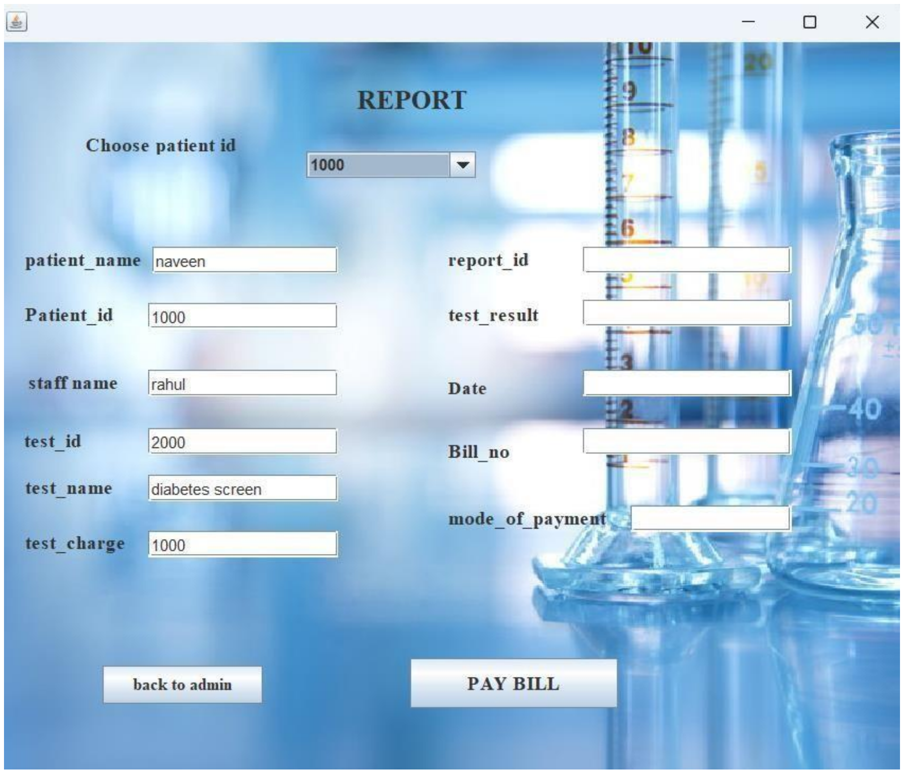
In this page we have options to add patient info and to work with patient info we will be getting this page from admin view if we go to patients button all the patients here will be given with a unique id such that no redundancy and mistakes happen from admin side while entering the report



s_id	t_id	s_name	address	age	ph_no
3001	2000	rahul	hebbal mysore	24	9432890871
3002	2001	rohith	vijaynagar mysore	26	7832689301
3003	2002	anjana	hebbal mysore	25	9873478901
3004	2003	sagar	indiranagar	27	783468908
3005	2004	darshan	hootgalli	28	8643973890
3006	2005	dhanush	hebbal	35	8675462124

Fig 17 : staff page

In this page their will be options to add,delete,update staff info and this page is obtained from admin view page by going to patient button


REPORT

Choose patient id 1000

patient_name naveen

Patient_id 1000

staff name rahul

test_id 2000

test_name diabetes screen

test_charge 1000

report_id

test_result

Date

Bill_no

mode_of_payment

Fig 17 : report page

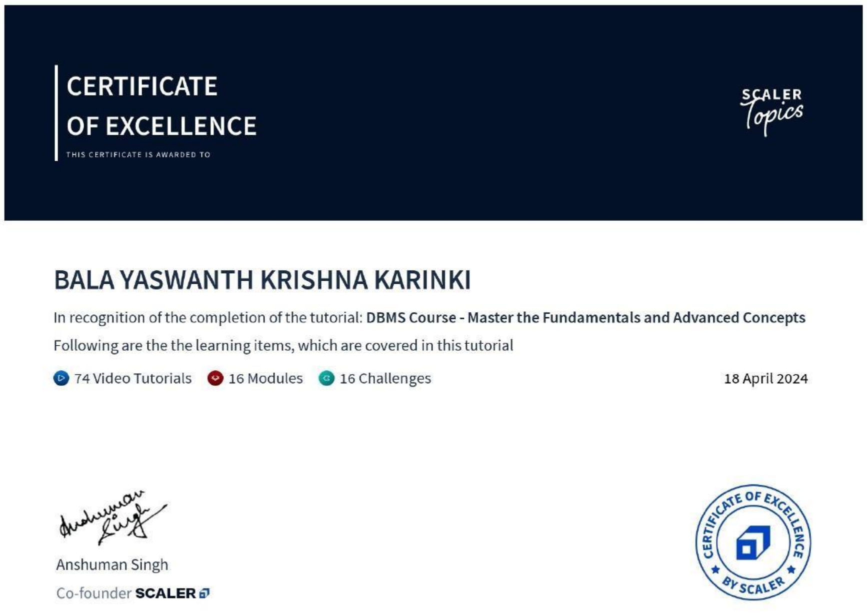
In this page we will be having option to give unique report id and the admin will be giving the test result,date,bill no,mode of payment here the unique feature is only we should choose p_id their details will be taken automatically. After this we will go for paying the bill .

7.2 conclusion

We conclude that the project diagnostic laboratory management completed successfully. With all requirements we need such that we got a good dbms project without redundancy of data.

CHAPTER – 8

8.1 Course Completion Certificate



CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

BALA YASWANTH KRISHNA KARINKI

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

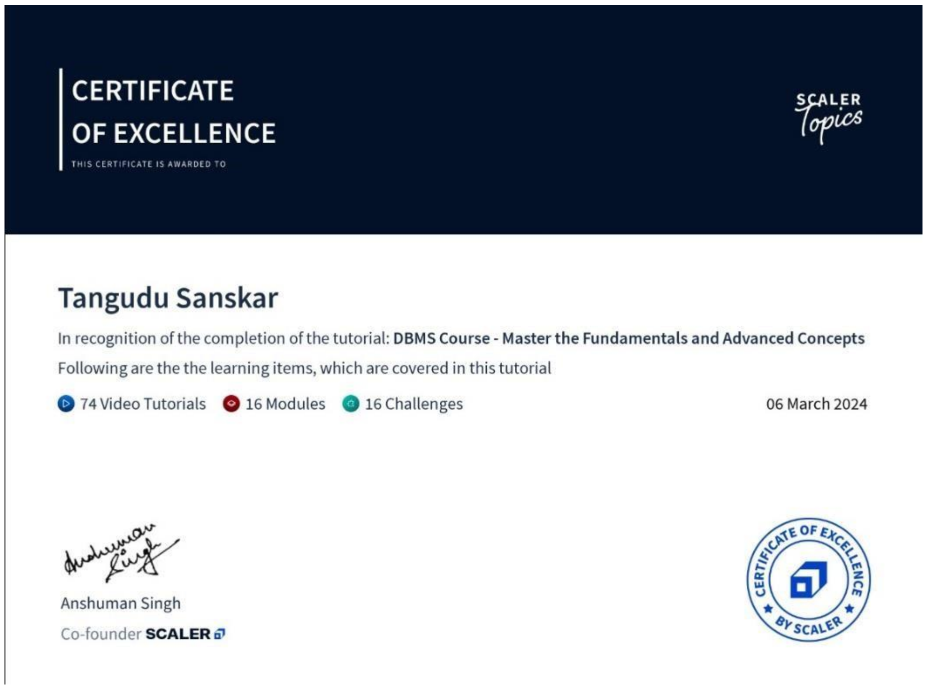
74 Video Tutorials 16 Modules 16 Challenges 18 April 2024

[illegible]

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE BY SCALER



CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

Tangudu Sanskar

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

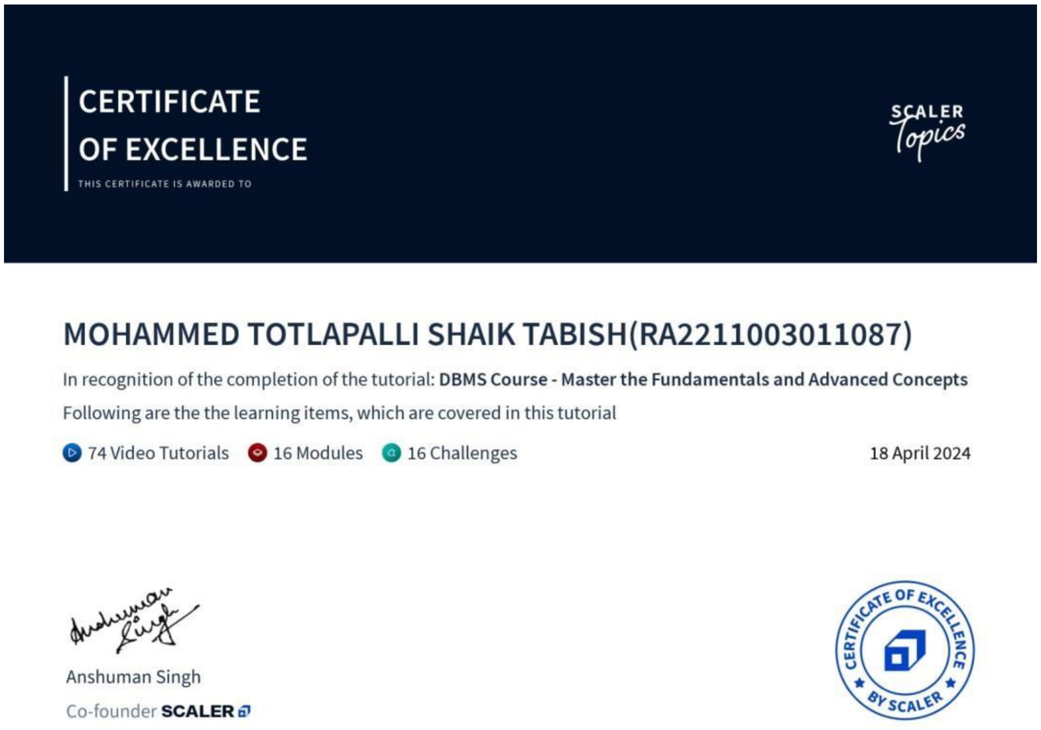
74 Video Tutorials 16 Modules 16 Challenges 06 March 2024

[illegible]

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE BY SCALER



CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

MOHAMMED TOTLAPALLI SHAIK TABISH(RA2211003011087)

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

● 74 Video Tutorials ● 16 Modules ● 16 Challenges

18 April 2024

[illegible]

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE ★ BY SCALER ★