VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

A PROJECT REPORT
ON

# Grievances Application Portal for Sahyadri

ARJUN RAGHUVIR BHANDARI | KUMAR RISHAV
4SF20CS017 4SF20CS054

In the partial fulfillment of the requirement for V Sem. B. E. (CSE)

DBMS LABORATORY WITH MINI PROJECT
Under the guidance of

**Ms. Vanishree B S**
Assistant Professor, Dept. of CSE

Department of Computer Science & Engineering
SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
An Autonomous Institution
Adyar, Mangalore-575007
2021-22

SAHYADRI COLLEGE OF ENGINEERING & MANAGEMENT

**An Autonomous Institution**

(Affiliated to Visvesvaraya Technological University, BELAGAVI)

Adyar, Mangalore – 575007

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

# CERTIFICATE

This is to certify that the project entitled "Grievances Application Portal for Sahyadri" is submitted in partial fulfillment for the requirement of V sem. B. E. (Computer Science & Engineering), "DBMS LABORATORY WITH MINI PROJECT" during the year 2022 – 23 is a result of bonfire work carried out by

| | |
|---|---|
| ARJUN RAGHUVIR BHANDARI | 4SF20CS017 |
| KUMAR RISHAV | 4SF20CS054 |

………………………………

Ms. Vanishree B S
Asst. Prof. Dept. of CS&E
SCEM, Mangaluru

………………………………

Dr. Nagesh H R
HOD, Dept. of CS&E
SCEM, Mangaluru

Signature of the Examiners

1. ………………………….

2. ………………………….

# ABSTRACT

Database is a collection of interrelated data which helps inefficient retrieval, insertion, and deletion of data from database and organizes data in the form of tables, views, schemas, reports, etc. The general-purpose software used to manage a database is called Database Management System (DBMS)

This project aims to develop an online Grievances Application Portal for Sahyadri. The portal will provide an efficient way for people in the Sahyadri region to register their grievances and complaints. It will provide an easy to use interface for users to enter their grievances, complaints, and suggestions, which can then be tracked and managed efficiently by the Grievance Officer. The portal will also provide a system for tracking the status of the grievance and a feedback system for users.

The portal will also provide a platform for the Grievance Officer to respond to the grievances and take necessary action. The portal will also provide an interface for the Grievance Officer to monitor and analyze the grievances, complaints, and suggestions received from the users. This project will help to streamline the grievance management process in the Sahyadri region, making it more efficient and effective.

## ACKNOWLEDGEMENT

It is with great satisfaction and emporia that we are submitting the Mini Project report on "Grievances Application Portal for Sahyadri". We have completed the part of 5th semester DBMS Laboratory with a Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi..

We are profoundly indebted to our guides, **Ms. Vanishree B S**, Assistant Professor, Department of Computer Science & Engineering for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Nagesh H R**, Head & Professor, Department of Computer Science & Engineering for her invaluable support and guidance.

We sincerely thank Dr. Rajesha. S, Principal, Sahyadri College of Engineering & Management who have always been a great source of inspiration.
Finally, yet importantly, we express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

ARJUN RAGHUVIR BHANDARI
4SF20CS017
V Sem, B.E., CSE
SCEM, Mangaluru

KUMAR RISHAV
4SF20CS054
V Sem, B.E., CSE
SCEM, Mangaluru

# PAGE INDEX

CHAPTER 1

# INTRODUCTION

Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS as a collection of interrelated data and set of programs to store & access those data in an easy and effective manner.

Database systems are basically developed for large amounts of data. When dealing with huge amounts of data, there are two things that require optimization: Storage of data and retrieval of data. A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data.

## 1.1 INTRODUCTION TO MYSQL

MySQL is an open-source, fast, reliable, and flexible relational database management system, typically used with PHP. It is a database system used for developing web-based software applications.

It was developed by Michael Widenius and David Axmark in 1994. It is presently developed, distributed, and supported by Oracle Corporation. It was written in C, C++.It is fully multithreaded by using kernel threads. It can handle multiple CPUs if they are available. It provides transactional and non-transactional storage engines.

## 1.2 JAVA

Java is a popular programming language. It is owned by Oracle, and more than 3 billion devices run Java. Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).It is opensource secure, fast, free and powerful. As Java is close to C++ and C#, it makes it easy for programmers to switch to Java.

## 1.3 INTRODUCTION TO NET BEANS

NetBeans IDE is a free, open source, integrated development environment (IDE) that enables you to develop desktop, mobile and web applications. The IDE supports application development in various languages, including Java, HTML5, PHP and C++.
The IDE provides integrated support for the complete development cycle, from project creation through debugging, profiling and deployment. The IDE runs on Windows, Linux, Mac OS X, and other UNIX-based systems.

## 1.4 NECESSITY OF PROJECT

This project "Grievances Application Portal for Sahyadri" is necessary in college because it provides a platform for students to voice their grievances and concerns in a secure and effective manner. It also gives the college administration a way to easily track, monitor and address student complaints. This helps the college to ensure a better and more positive learning environment for all students. Additionally, having a centralized system for grievances allows for better transparency and accountability in the college's decision-making processes.

## 1.5 INTRODUCTION TO PROJECT

In a college or any organization; finding the right and responsible department for your communication, grievance or deliverance can be challenging. A centralized portal to sort this confusion and haste the process of information flow and resolution in an organization is crucial. Tracking the complaint, request or communication all the way from lodging on the portal to tracking its acknowledgement can be tedious to do manually and a lot of this can be overlooked. A portal with tracking is important while this also keeps the head of the department in the loop so that status and satisfaction of the department is kept in question of competence allowing more transparency and efficient supervision.

CHAPTER 2

# REQUIREMENT SPECIFICATION

## 2.1 HARDWARE REQUIREMENTS

• Processor : Intel Core i5-8250U (8th Gen)

• RAM : 8 GB

• Hard Disk : 1 TB

• Input Device : Standard keyboard and Mouse

• Output Device : Monitor

## 2.2 SOFTWARE REQUIREMENTS

• Operating System : Windows 10

• User Interface : JAVA

• Programming Language : JAVA

• Workbench : MySQL Version 19c

• Database : MySQL

• Application used for Frontend : NetBeans IDE 12.2

• JDBC API for Database connection

**Research Method :**

- Functionalities mixed from g-pay, bank systems and other ticket based transaction services
- Requirement variables from Personal experience
- Investigation method : Interviews and peer-peer conversations
- Standardization references : UX for Edutainment Portal Enhancement -Universitas Multimedia Nusantara Tangerang, Indonesia

CHAPTER 3

# DESIGN

## 3.1 E-R DIAGRAM

An entity-relationship model describes interrelated things of interest in a specific domain of knowledge. The ER Diagram of our project is shown in Figure: 3.1.1

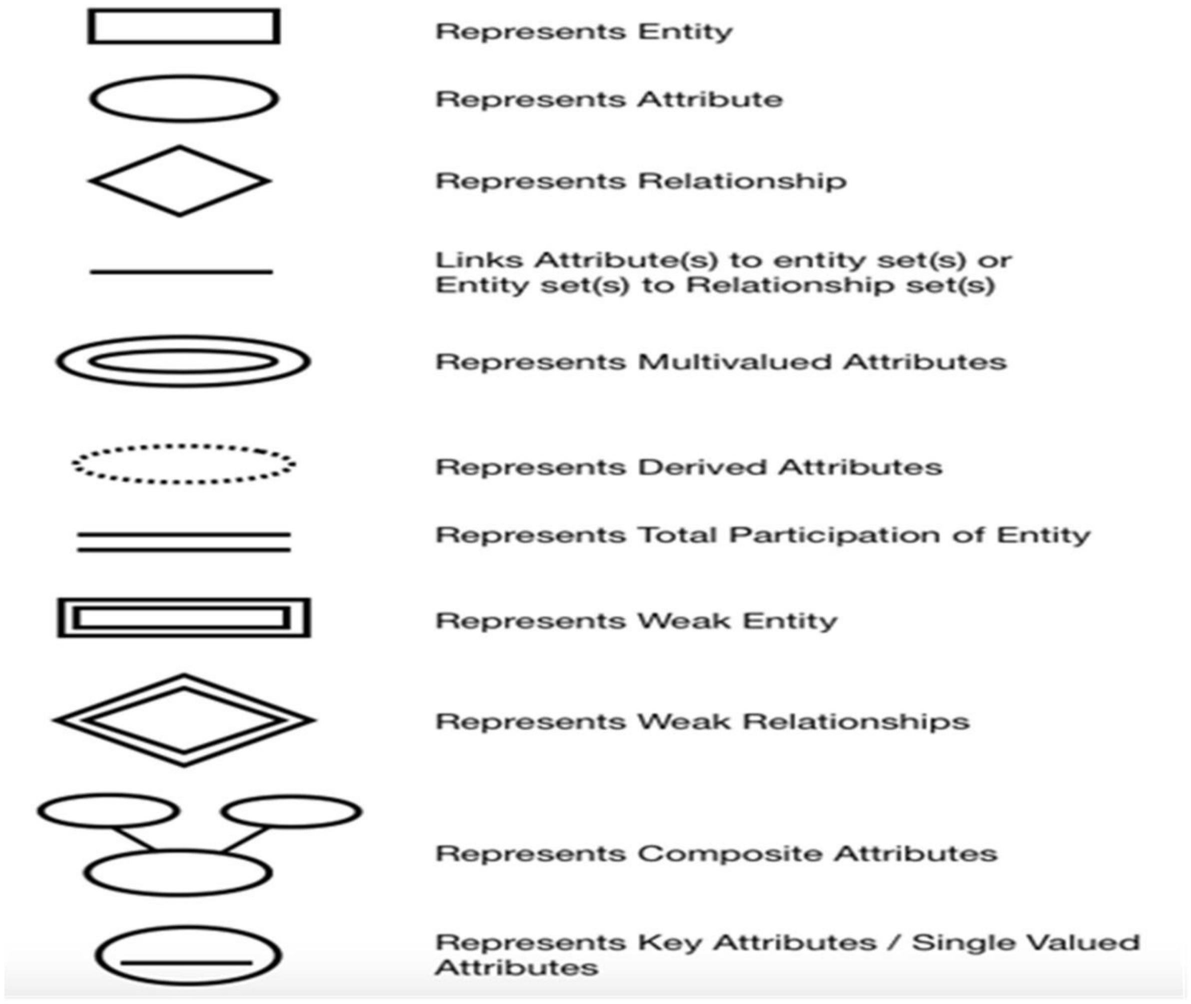

Represents Entity
Represents Attribute
Represents Relationship
Links Attribute(s) to entity set(s) or Entity set(s) to Relationship set(s)
Represents Multivalued Attributes
Represents Derived Attributes
Represents Total Participation of Entity
Represents Weak Entity
Represents Weak Relationships
Represents Composite Attributes
Represents Key Attributes / Single Valued Attributes

Fig:- 3.1.1

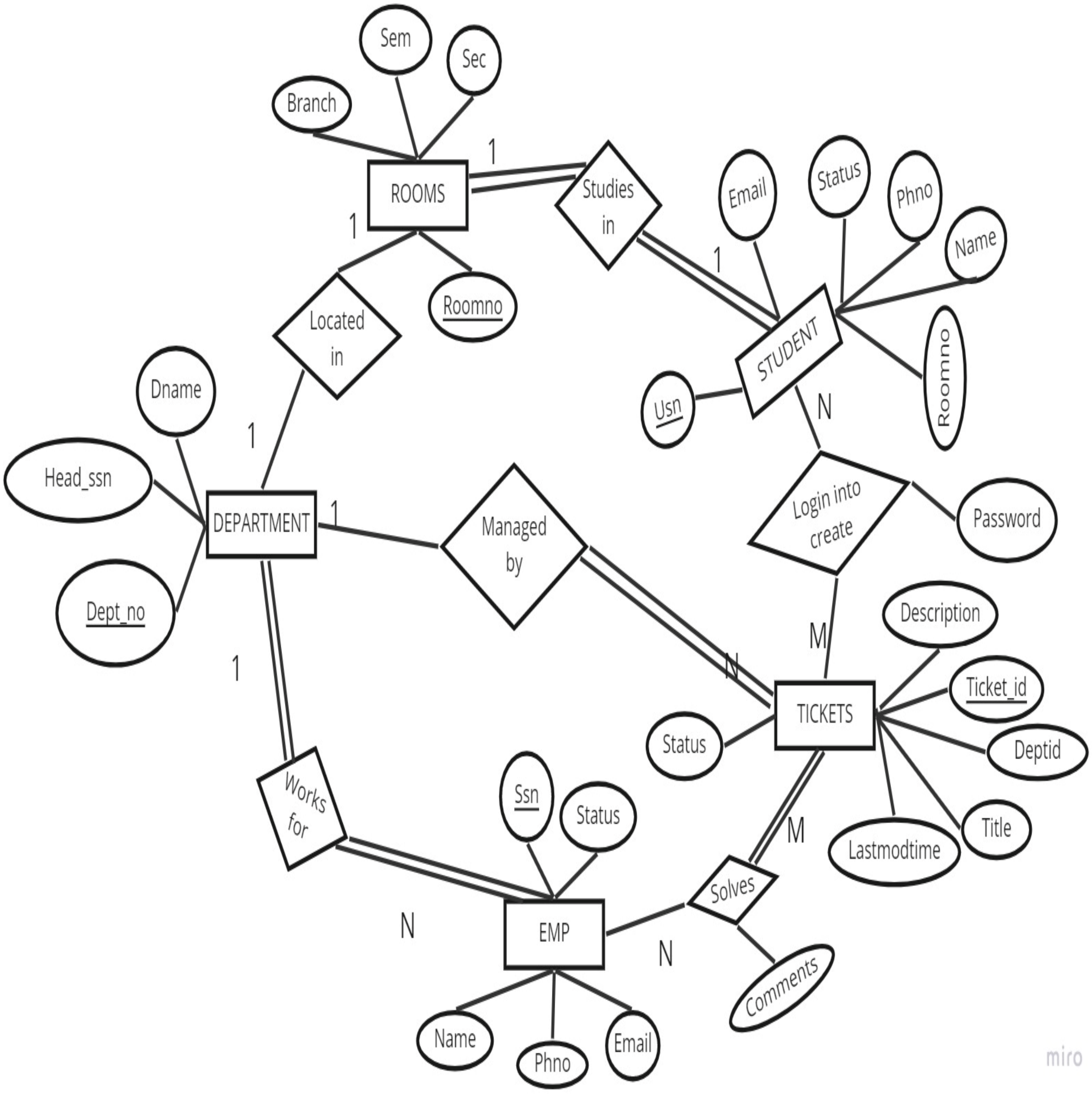


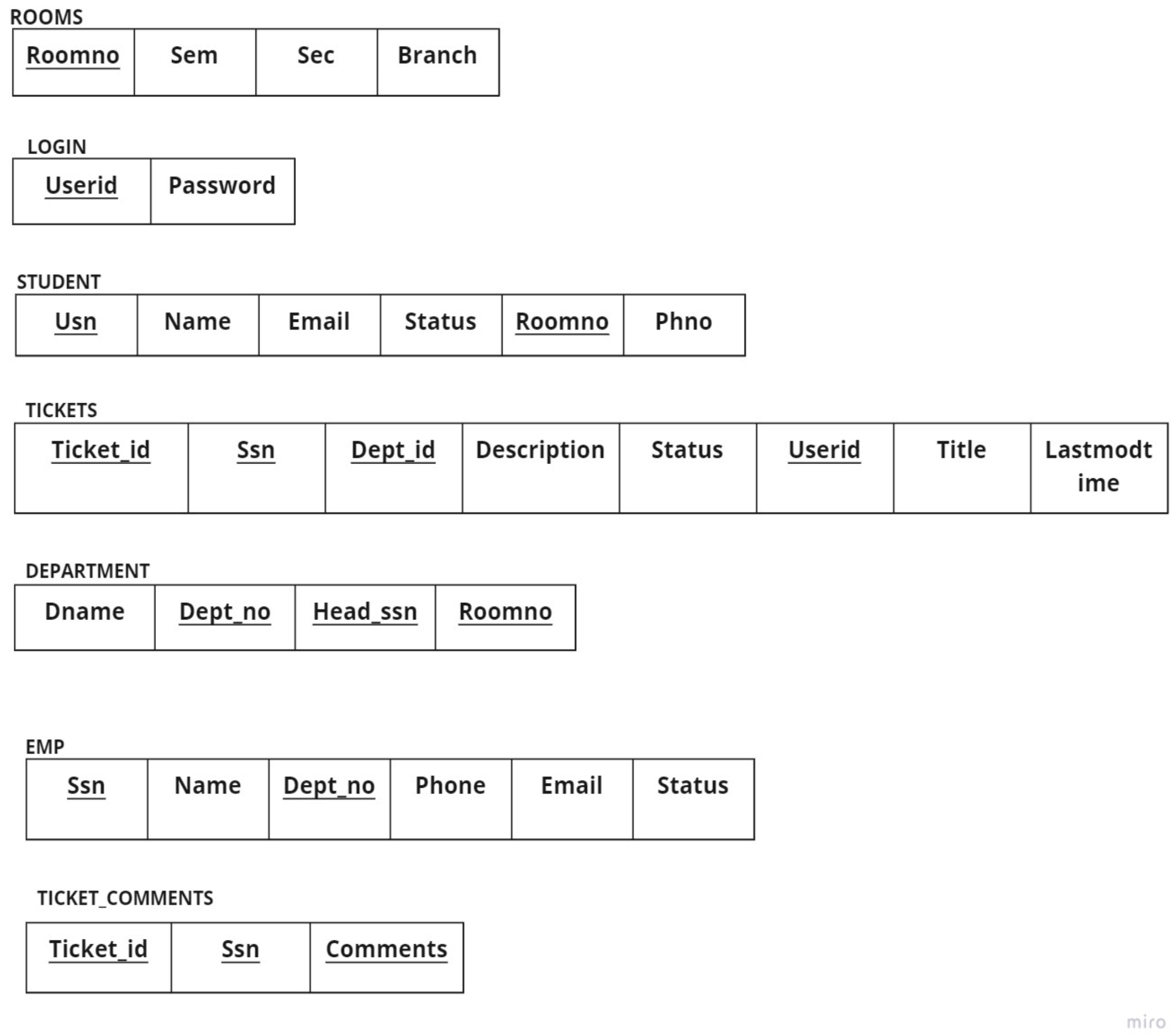
FIGURE: 3.1.2 ER Diagram

## 3.2 RELATIONAL SCHEMA

A relational schema is a blueprint for a relational database. It defines the structure of the database, including the names and data types of the tables, the names and data types of the columns within each table, and the relationships between the tables.

### Step 1:Mapping of Regular Entity Types

For every regular entity in our entity-relationship diagram, we have created a separate relation. These created relations contain the respected attributes and respected primary key.



ROOMS

| Roomno | Sem | Sec | Branch |
|---|---|---|---|

LOGIN

| Userid | Password |
|---|---|

STUDENT

| Usn | Name | Email | Status | Roomno | Phno |
|---|---|---|---|---|---|

TICKETS

| Ticket_id | Ssn | Dept_id | Description | Status | Userid | Title | Lastmodtime |
|---|---|---|---|---|---|---|---|

DEPARTMENT

| Dname | Dept_no | Head_ssn | Roomno |
|---|---|---|---|

EMP

| Ssn | Name | Dept_no | Phone | Email | Status |
|---|---|---|---|---|---|

TICKET_COMMENTS

| Ticket_id | Ssn | Comments |
|---|---|---|

**Figure: 3.2.1.1 Mapping of Regular Entity Type**

STEP 2: Mapping of weak entity Types

A weak entity set is one which does not have any primary key associated with it. The primary key of the parent entity i.e. TICKET_ID,SSN,USER_ID entities are added to the TICKETS_COMMENT,LOGIN.

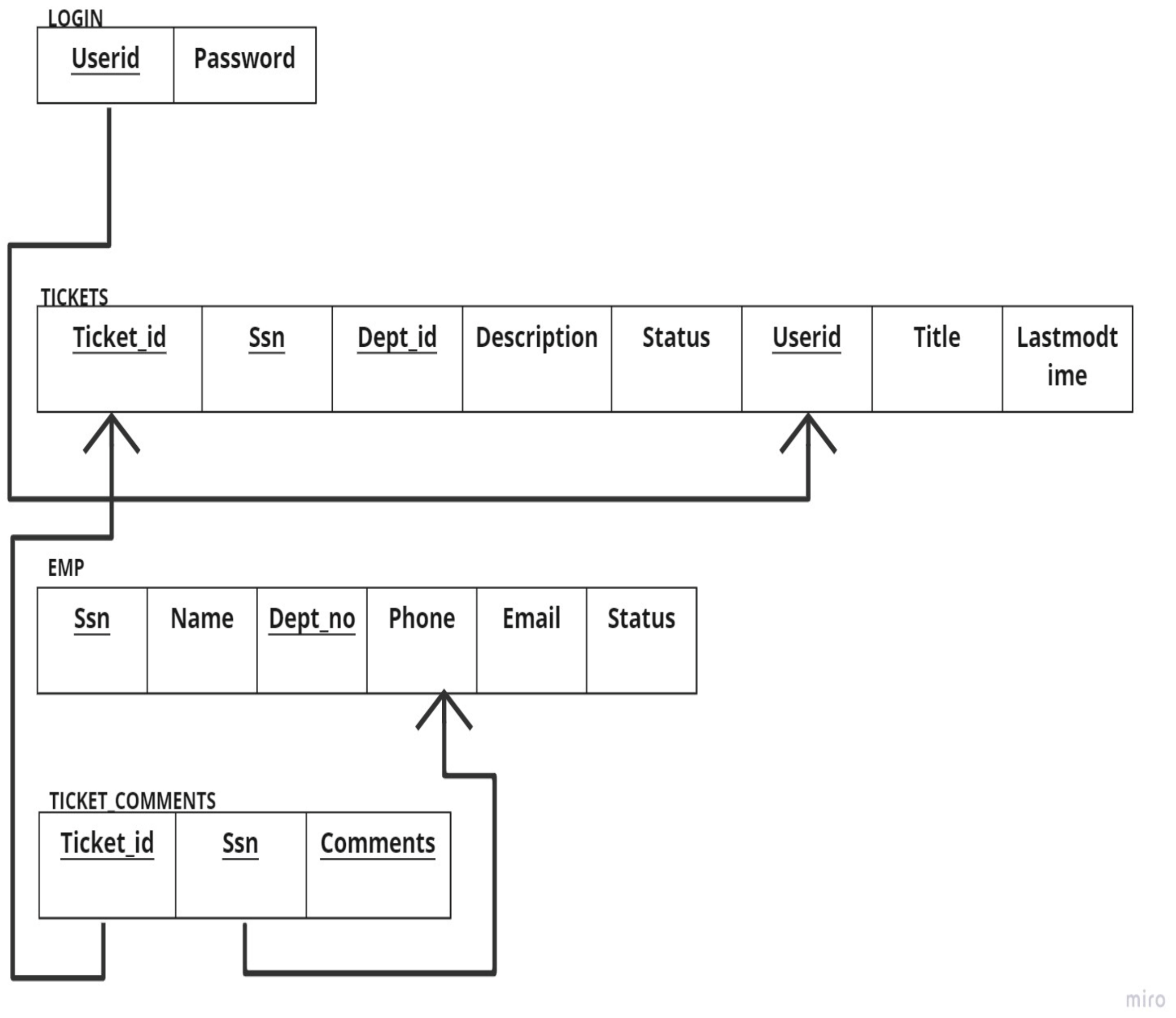


**Fig:-3.2.2.2**

## STEP 3: Mapping of 1:1 relation type

For each binary 1:1 relationship type R in ER Schema identifies the relation S and T that are participating in the relationship R. Here we add the primary key of one side of the relation to the other side as a foreign key.

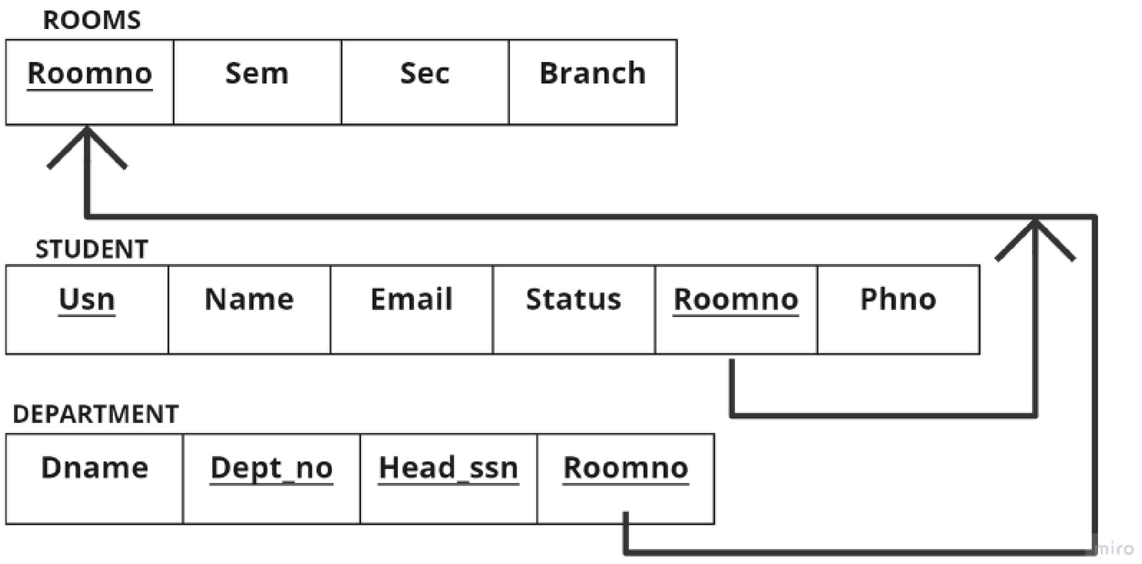


**Fig:-3.2.2.3**

STEP 4: Mapping of 1: N relation types

For every 1: N relation types identify the entity on the N side. Make primary key entity Which is participation in 1 side as foreign key of the entity which is the N side. If there are any attributes for relationships add to n side.Since there is only one 1:N mapping and the primary key of the 1th side is already present at the Nth side of the relation we ignore this step.

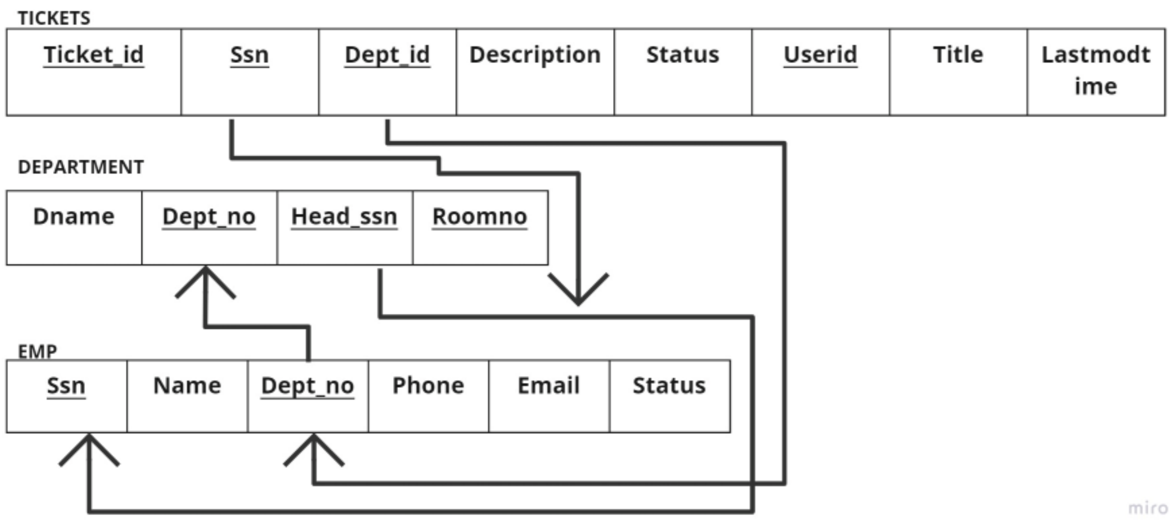


Fig:-3.2.2.4

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

The ER Diagram contains  N:M relationship type.

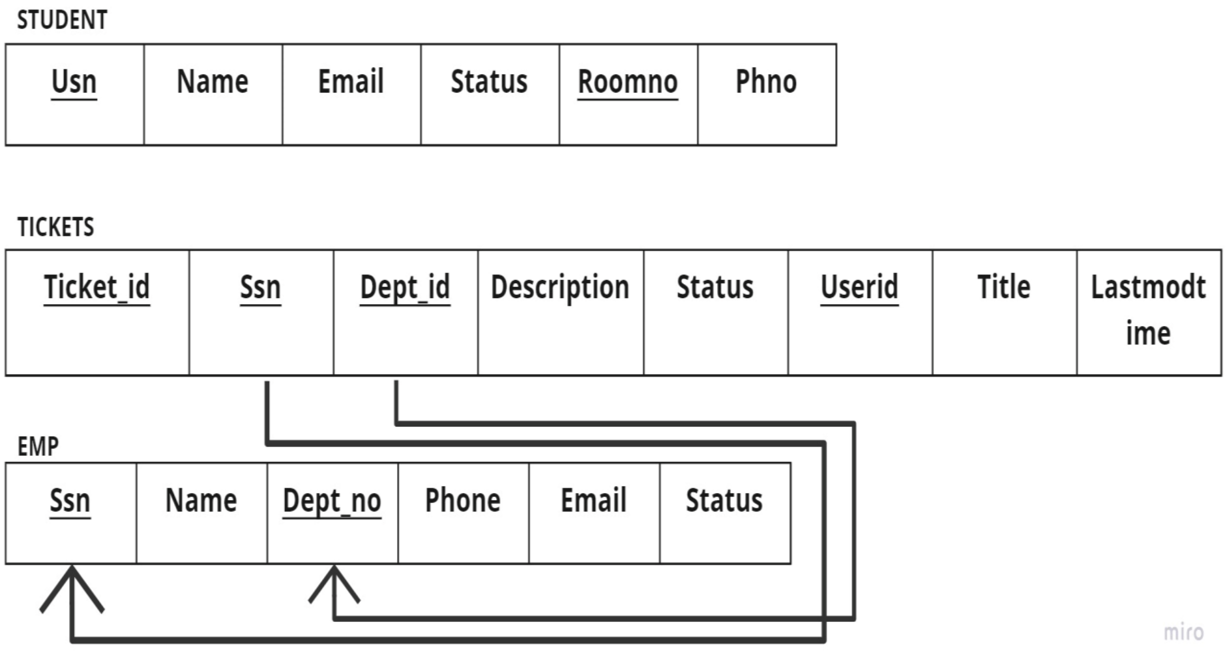


Fig:-3.2.2.5

## Step 6: Mapping of Multivalued Attributes

The ER Diagram contains no M: N relationship type. Therefore this step is ignored in the project.

## Step 7: Mapping of Binary N-ary Relationship Types

The ER Diagram contains no N-ary relationship type. Therefore this step is ignored in the project.

## SCHEMA:

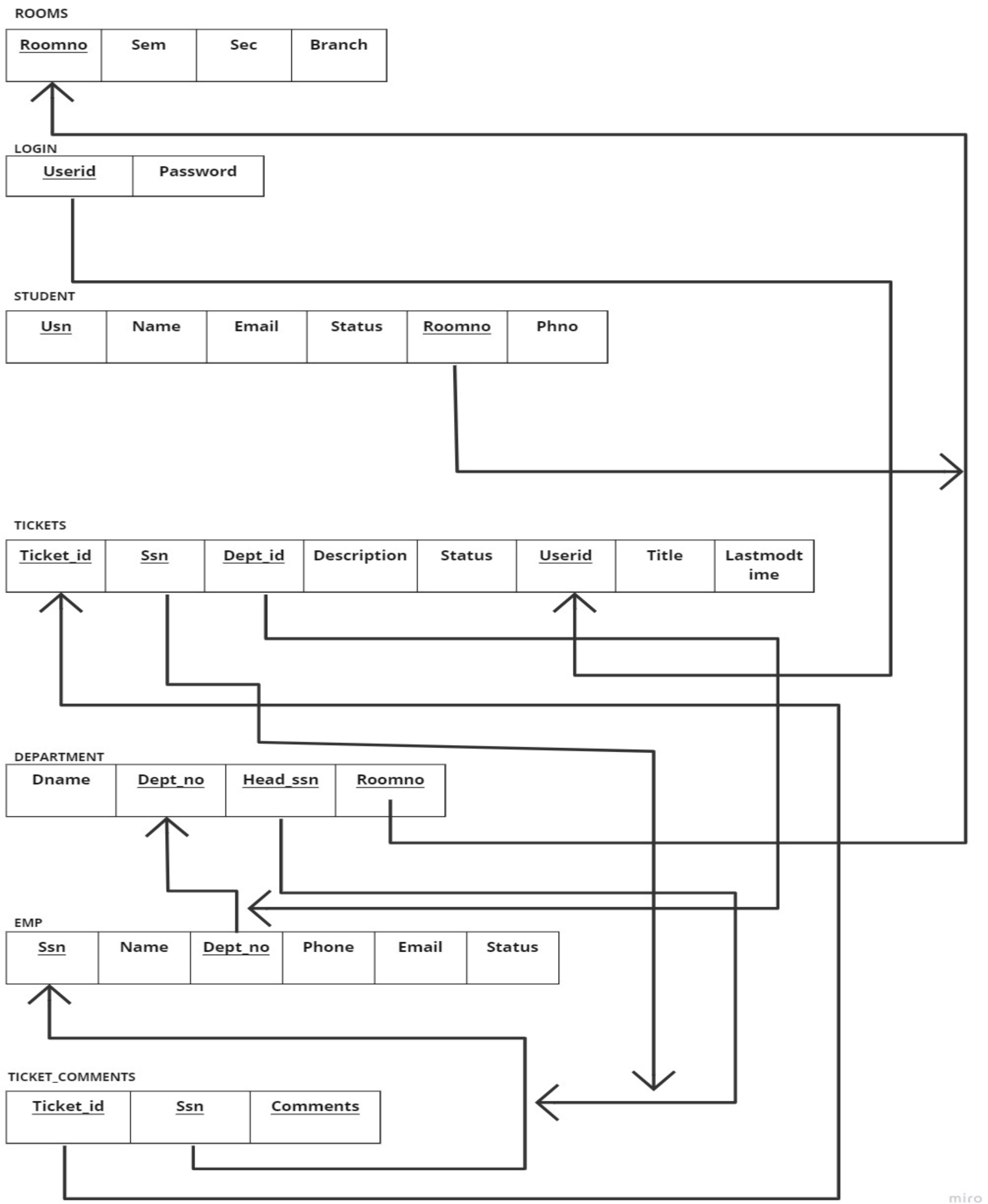


**Fig:-3.2.2.6**

## 3.3 NORMALIZATION

In relational database design, the process of organizing data to minimize redundancy is Normalization.
Normalization usually involves dividing a database into two or more tables and defining relationships between the tables.

### FIRST NORMAL FORM

A table is said to be in 1NF if and only if each attribute of the relation is atomic, Each row in a table should be identified by primary key. No rows of data should have replacing group of column values.

### SECOND NORMAL FORM

A table is said to be in 2NF if both the following conditions hold

- Table is in INF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of the table.

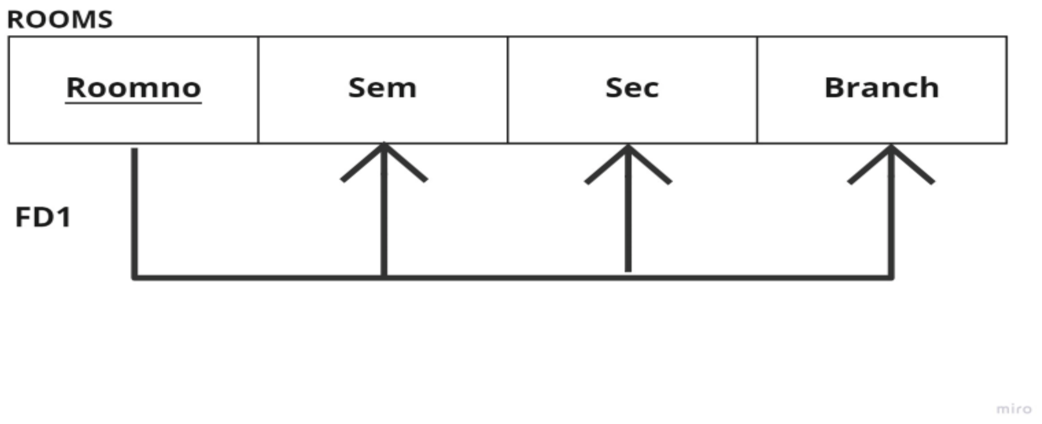
### THIRD NORMAL FORM

A table design is said to be in 3NF if both the following conditions hold

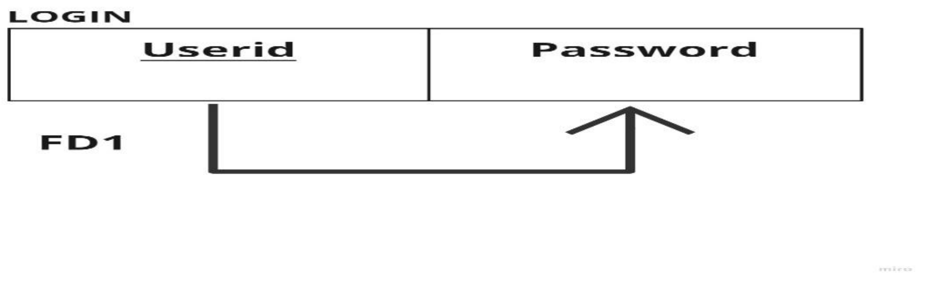
- Table must be in 2NF
- No non-prime attribute is transitively dependent on primary key

## 3.3 NORMALIZATION

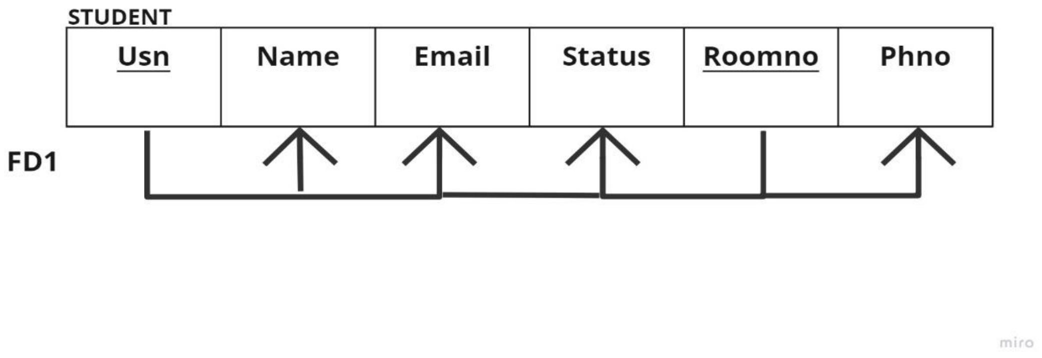
Normalization in database management systems (DBMS) refers to the process of organizing data in a database into separate tables in order to eliminate redundancy and improve data integrity. The goal of normalization is to minimize data duplication and ensure that data is stored in a consistent, structured format.



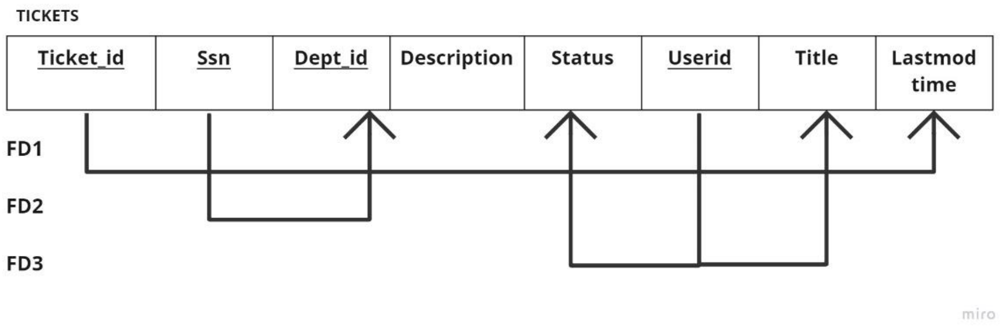
- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key.



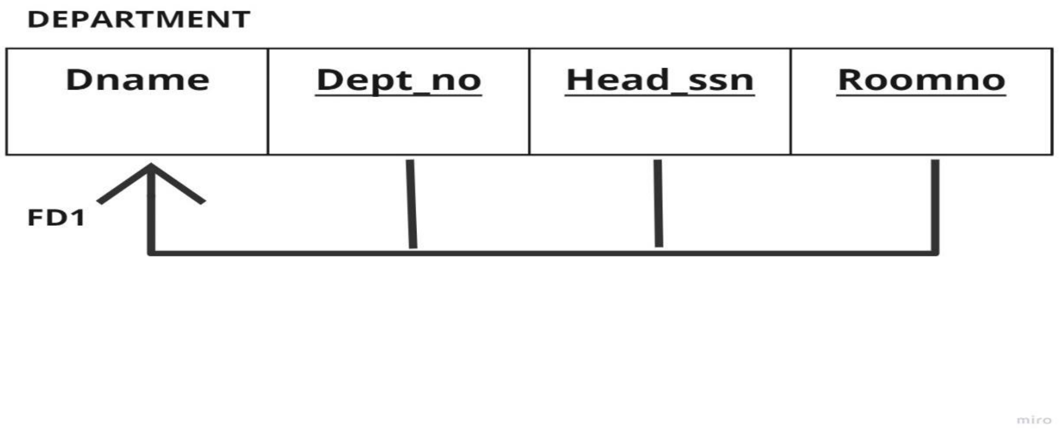
- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key



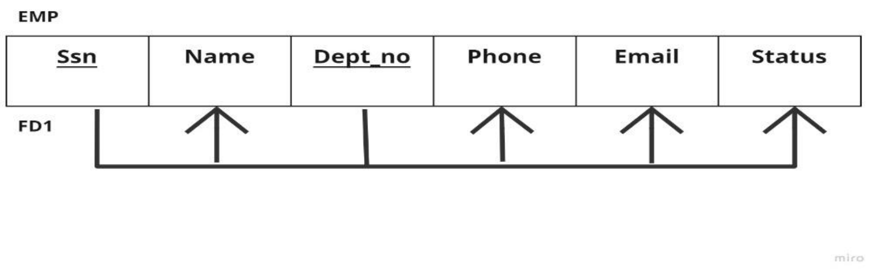
- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key



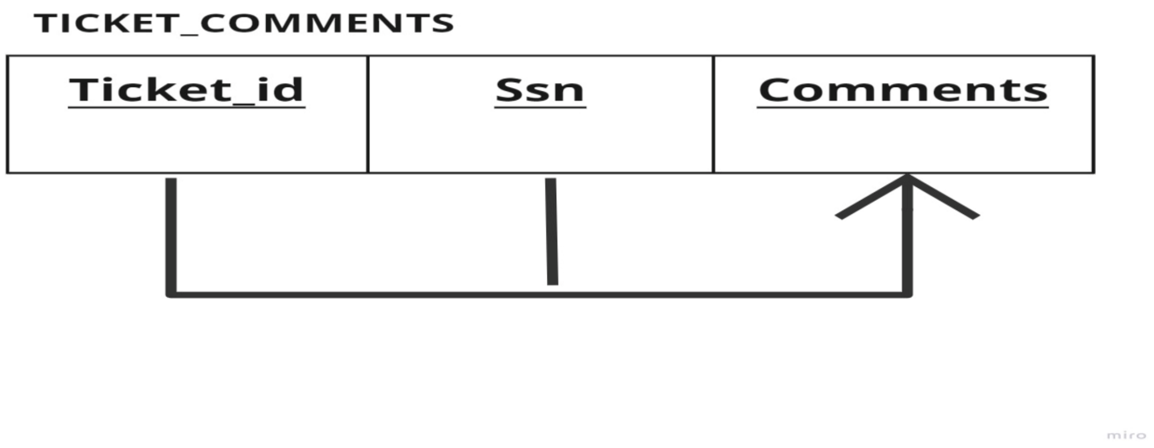
- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key



- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key



- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key



- It is in 1NF because there are no multivalued and composite attributes and their combinations.
- It is in 2NF because it satisfies 1NF and there is only one primary key.
- It is in 3NF because it satisfies 2NF and there are no non-prime attributes transitively dependent on the primary key

CHAPTER 4

# IMPLEMENTATION

## 4.1 TABLE STRUCTURE

### STUDENT

```
CREATE TABLE STUDENT (
USN VARCHAR(10) PRIMARY KEY,
NAME VARCHAR(20),
phno number(10),
roomno number(3),
email varchar(25),
status char(1));
ALTER TABLE student add foreign key (roomno) references ROOMS(roomno);
```

| ATTRIBUTES | TYPE |
|---|---|
| USN | VARCHAR(10) |
| NAME | VARCHAR(20) |
| EMAIL | VARCHAR(25) |
| PHNO | INT(10) |
| ROOMNO | VARCHAR(15) |
| STATUS | CHAR(1) |

## ROOMS

```
CREATE TABLE ROOMS (
roomno number(3) primary key,
sem char(1),
sec char(1),
branch varchar(10));
```

| ATTRIBUTES | TYPE |
|---|---|
| ROOMNO | NUMBER(3) |
| SEM | CHAR(1) |
| SEC | CHAR(1) |
| BRANCH | VARCHAR(10) |

## EMPLOYEE

```
CREATE TABLE EMP (
SSN VARCHAR(8) PRIMARY KEY,
NAME VARCHAR(20),
phno number(10),
dept_no number(2),
email varchar(25),
status char(1));
ALTER TABLE emp add foreign key (dept_no) references dept(dept_no);
```

| ATTRIBUTES | TYPE |
|---|---|
| SSN | VARCHAR(8) |
| NAME | VARCHAR(20) |
| PHNO | NUMBER(10) |
| DEPT_NO | NUMBER(2) |
| EMAIL | VARCHAR(25) |
| STATUS | CHAR(1) |

## DEPARTMENT

```
CREATE TABLE DEPT (
dept_no number(2) primary key,
head_ssn varchar(8),
roomno number(3),
dName varchar(255));
ALTER TABLE dept add foreign key (head_ssn) references emp(ssn);
```

| ATTRIBUTES | TYPE |
|---|---|
| HEAD_SSN | VARCHAR(8) |
| DNAME | VARCHAR(255) |
| DEPT_NO | NUMBER(2) |
| ROOMNO | NUMBER(3) |

## LOGIN

```
CREATE TABLE login (
userid varchar(10) primary key,
password varchar(20));
```

| ATTRIBUTES | TYPE |
|---|---|
| USERID | VARCHAR(10) |
| PASSWORD | VARCHAR(20) |

## TICKET

```
CREATE TABLE ticket(
ticketid number(5) GENERATED BY DEFAULT ON NULL AS
IDENTITY,
userid varchar(13),
deptid references dept,ssn varchar (10),
title varchar(255),
status varchar(20),
description varchar(265),
primary key(ticketid),
LastModTime DATE DEFAULT CURRENT_DATE);
```

| ATTRIBUTES | TYPE |
|---|---|
| TICKET_ID | NUMBER(5) |
| USER_ID | VARCHAR(13) |
| DEPT_ID | VARCHAR(10) |
| TITLE | VARCHAR(255) |
| STATUS | VARCHAR(20) |
| DESCRIPTION | VARCHAR(265) |
| LATMODTIME | DATE |

## TICKET_COMMENTS

```
create table ticket_comments(
ticketid references ticket,
ssn references emp,
comments varchar(256),
primary key(ticketid,ssn,comments));
```

| ATTRIBUTES | TYPE |
|---|---|
| TICKET_ID | NUMBER(5) |
| SSN | VARCHAR(8) |
| COMMENTS | VARCHAR(256) |

## USER

CREATE USER username IDENTIFIED BY password is used to
create a user for a database with certain special privileges,
like notSYS in this case is a DBA who is not the SYS login,
which means notSys user can create triggers unlike the sys logon for a database.

## 4.2 FUNCTIONALITY

### 4.2.1 CONNECTING TO DATABASE

```
Connection conn = null;
     try
       {
          Class.forName("oracle.jdbc.OracleDriver");
          String dbURL = "jdbc:oracle:thin:@localhost:1521:orcl";
          String username = "sys as SYSDBA";
          String password = "GAPS";
          conn = DriverManager.getConnection(dbURL, username, password);
       }
       catch (ClassNotFoundException ex)
       {
          ex.printStackTrace();
       }
       catch (SQLException ex)
       {
          ex.printStackTrace();
       }
```

## 4.2.2 SELECTION

```
Connection conn = null;
        try
        {
            Class.forName("oracle.jdbc.OracleDriver");
            String dbURL = "jdbc:oracle:thin:@localhost:1521:orcl";
            String username = "sys as SYSDBA";
            String password = "GAPS";
            conn = DriverManager.getConnection(dbURL, username, password);
        }
        catch (ClassNotFoundException ex)
             ex.printStackTrace();
        catch (SQLException ex)
            ex.printStackTrace();
        finally
        {
            try
            {
                if (conn != null && !conn.isClosed())
                {
                    Statement stmt=conn.createStatement();
                    // String tid,String title, String desc, String status
                    ResultSet rs=stmt.executeQuery("select ticketid,title,description,status from Ticket
where ssn='"+u+"'");
                    return rs;
                }
            }
            catch (SQLException ex)
{
                ex.printStackTrace();
            }
            catch (Exception ex)
            {
                ex.printStackTrace();
            }
        }
        return null;
```

## 4.2.3 INSERTING VALUES INTO TABLE

Collection System Table

```
Assign.setBackground(new java.awt.Color(102, 255, 204));

        Assign.setText("Assign");

        Assign.addMouseListener(new java.awt.event.MouseAdapter() {

            public void mouseClicked(java.awt.event.MouseEvent evt) {

                if(TicketSelector.getSelectedItem().toString().equals("Ticket IDs") ||
EMPSelector.getSelectedItem().toString().equals("Available EMPS"))

                {

                    new ErrorScreen("<html>You need to select<br>1. A Ticket ID<br>2. An Employee");

                    return;

                }
                RunQueriesHead.createTicket("update ticket set
ssn='"+EMPSelector.getSelectedItem().toString()+"',status='Assigned' where
ticketid="+TicketSelector.getSelectedItem().toString());
                new ErrorScreen("Assign SUCCESS!");
            }
        });


public static void createTicket(String q) {
```

```
Connection conn = null;
try
{
    Class.forName("oracle.jdbc.OracleDriver");
    String dbURL = "jdbc:oracle:thin:@localhost:1521:orcl";
    String username = "sys as SYSDBA";
    String password = "GAPS";
    conn = DriverManager.getConnection(dbURL, username, password);
}
catch (ClassNotFoundException ex)
    ex.printStackTrace();
catch (SQLException ex)
    ex.printStackTrace();
finally
{
    try
    {
        if (conn != null && !conn.isClosed())
        {
            Statement stmt=conn.createStatement();
            stmt.executeQuery(q);
        }
    }
    catch (SQLException ex)
        ex.printStackTrace();
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
```

## 4.2.4 DELETE

```
 if (conn != null && !conn.isClosed())
                {
                    if(!isValidPassword(n))
                        return "<html>Invalid, use:<br>1. 8-20 Letters<br>2. Capital and Small
letters<br>3. Digits";
                    Statement stmt=conn.createStatement();
                    ResultSet rs=stmt.executeQuery("select password from login where
userid='"+u+"'");
                    rs.next();
                    if((c.toString()).equals(rs.getString("password")))
                    {
                        stmt=conn.createStatement();
                        stmt.executeQuery("alter table login delete where userid='"+u+"'");
                        return "ACCOUNT DELETED!!";
                    }
                    else
                    {
                        return "<html>Incorrect <br>Current Password";
                    }
        }
```

## 4.2.5 UPDATE

```
try
{
    if (conn != null && !conn.isClosed())
    {
        if(!isValidPassword(n))
            return "<html>Invalid, use:<br>1. 8-20 Letters<br>2. Capital and Small letters<br>3.
                    Digits";
        Statement stmt=conn.createStatement();
        ResultSet rs=stmt.executeQuery("select password from login where userid='"+u+"'");
        rs.next();
        if((c.toString()).equals(rs.getString("password")))
        {
            stmt=conn.createStatement();
            stmt.executeQuery("update login set password ='"+n+"' where userid='"+u+"'");
            return "UPDATED!!";
        }
        else
        {
            return "<html>Incorrect <br>Current Password";
        }
    }
}
catch (SQLException ex)
{
    ex.printStackTrace();
}
catch (Exception ex)
{
    ex.printStackTrace();
}
```

## 4.2.6 TRIGGER

```
CREATE OR REPLACE TRIGGER update_timestamp
  BEFORE INSERT OR UPDATE ON ticket
  FOR EACH ROW
BEGIN
  :NEW.LastModTime := systimestamp;
END;
```

# CHAPTER 5

# RESULTS

## 5.1 Home Page

The user will be able to see the interface of home page or welcome page

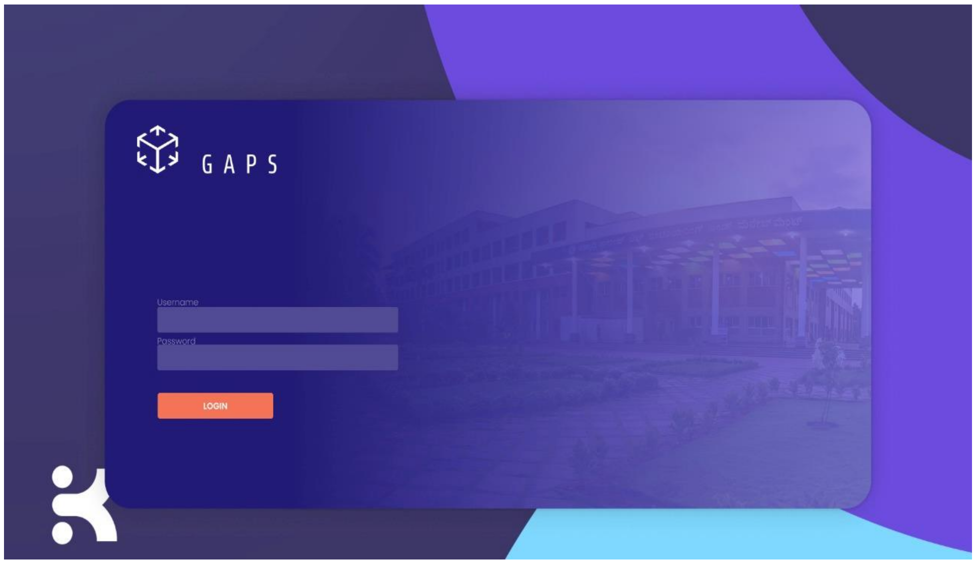


**Fig:5.1.1**

## 5.2 Click on the LOGIN button on Home page

In Fig 5.1.1 :When a STUDENT clicks on LOGIN button on our home page, he/she will be directed to our CREATE TICKET page (Fig:5.2.1)

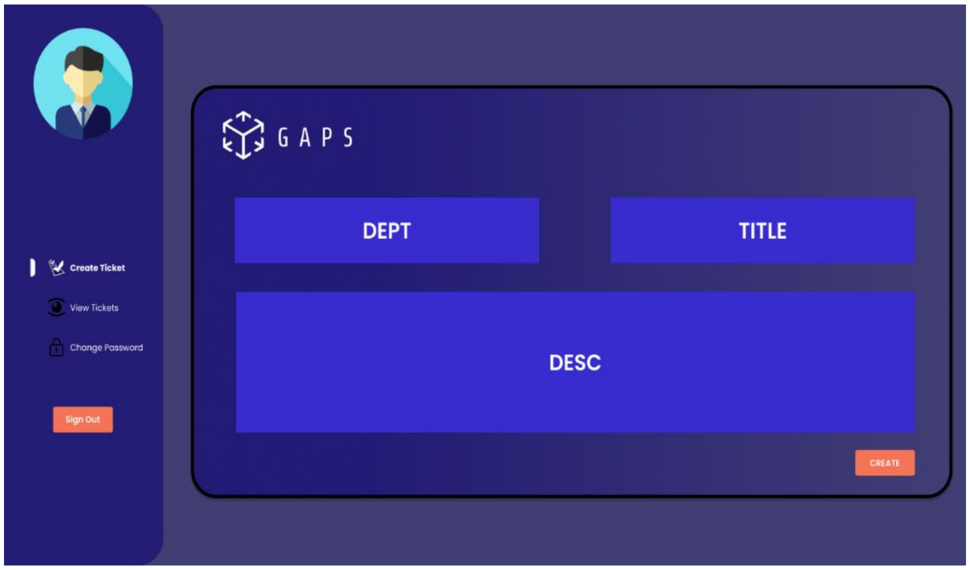


**Fig:-5.2.1**

OR When a HEAD_SSN/,EMPLOYEE clicks on LOGIN button on our home page, he/she will be directed to our CREATED TICKET(FIG:-5.2.2)/ASSIGNED TICKET page(FIG:-5.2.3)

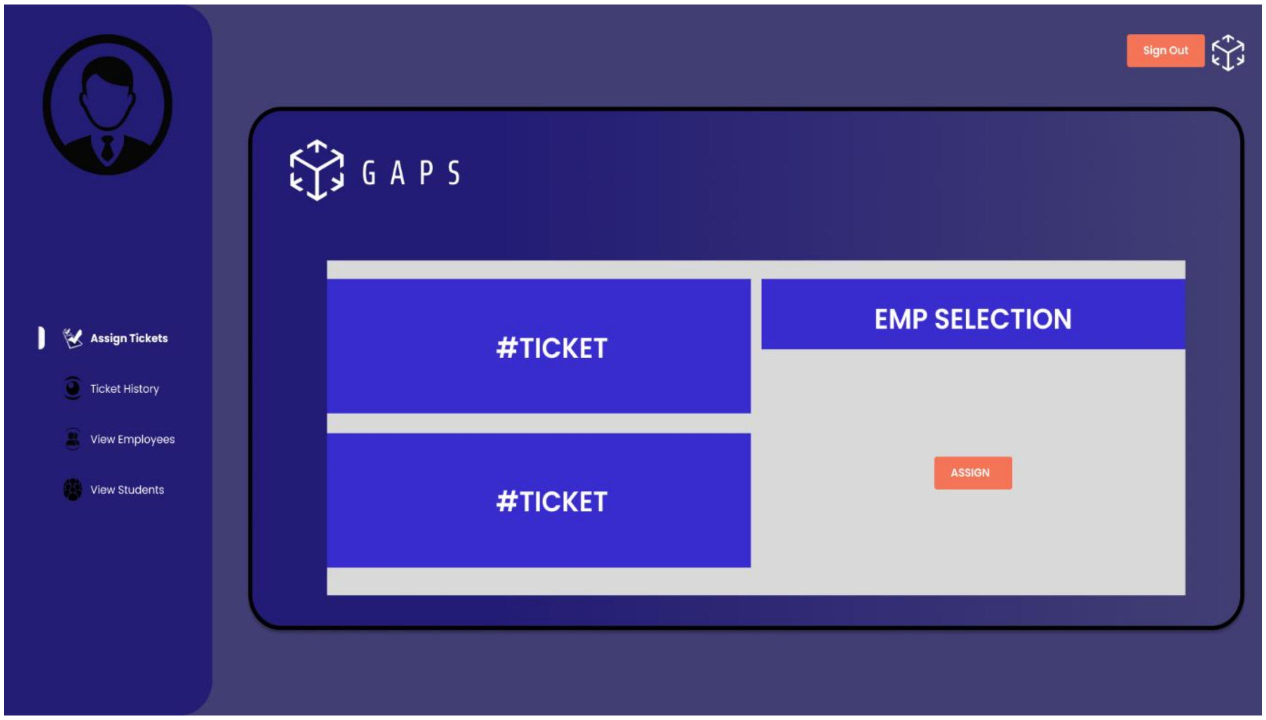


**Fig:-5.2.2**

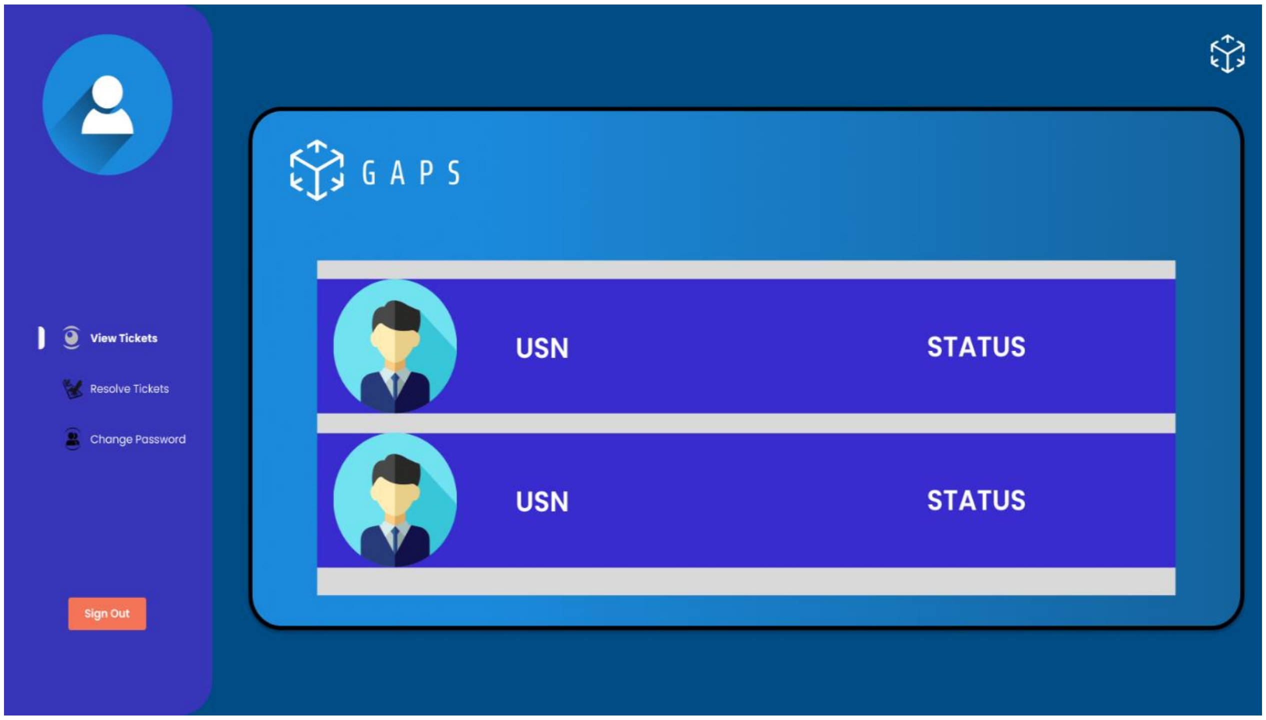


**Fig:-5.2.3**

## 5.3 STUDENT page

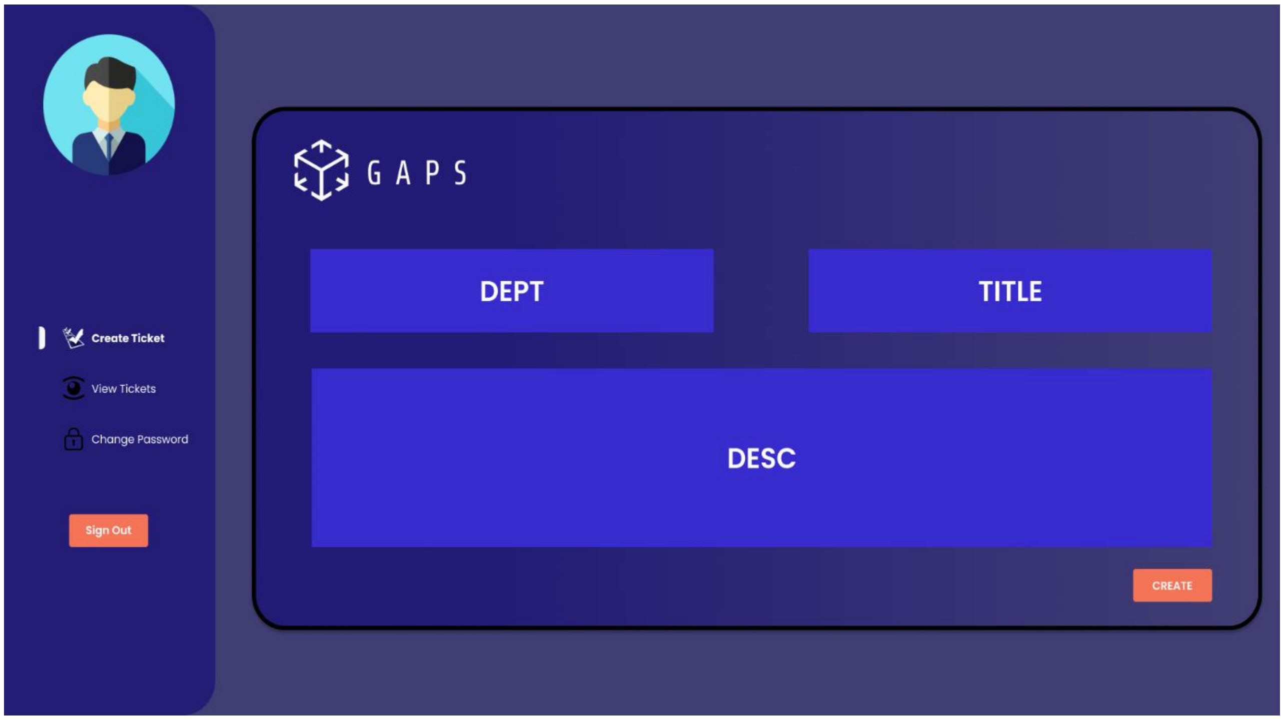


**Fig:-5.3.1**

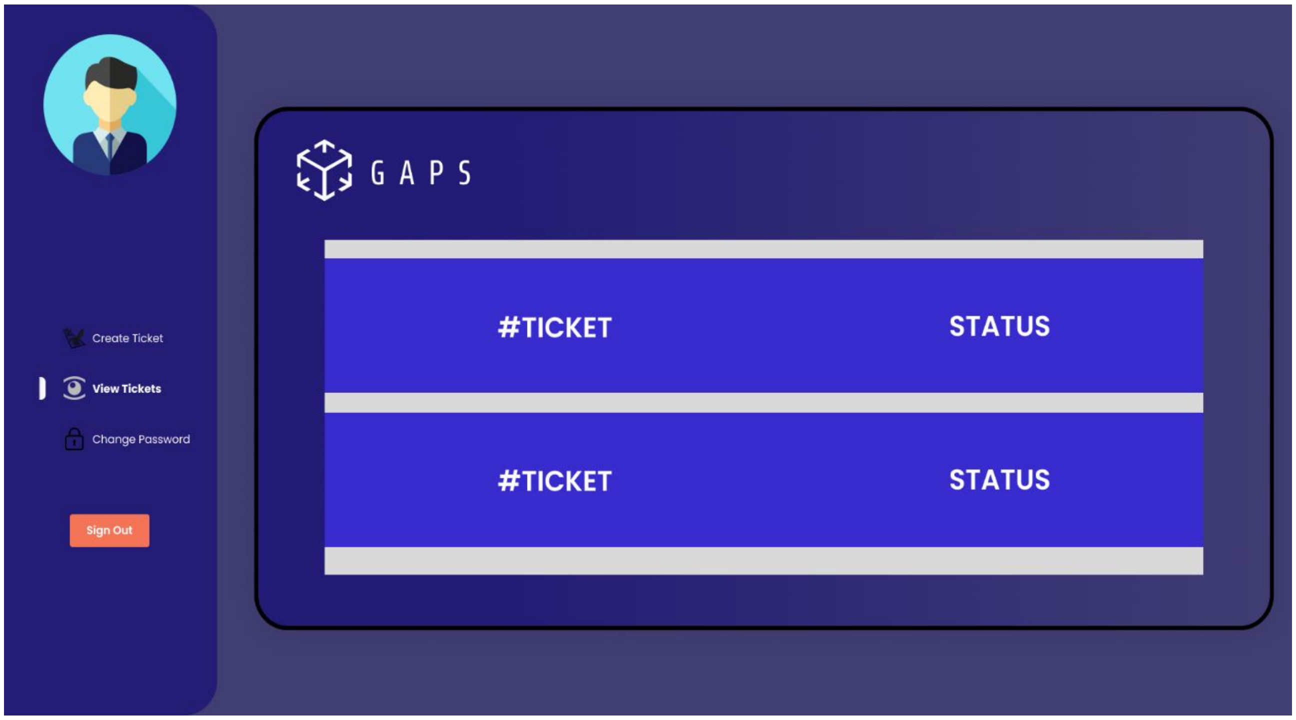


**Fig:-5.3.2**

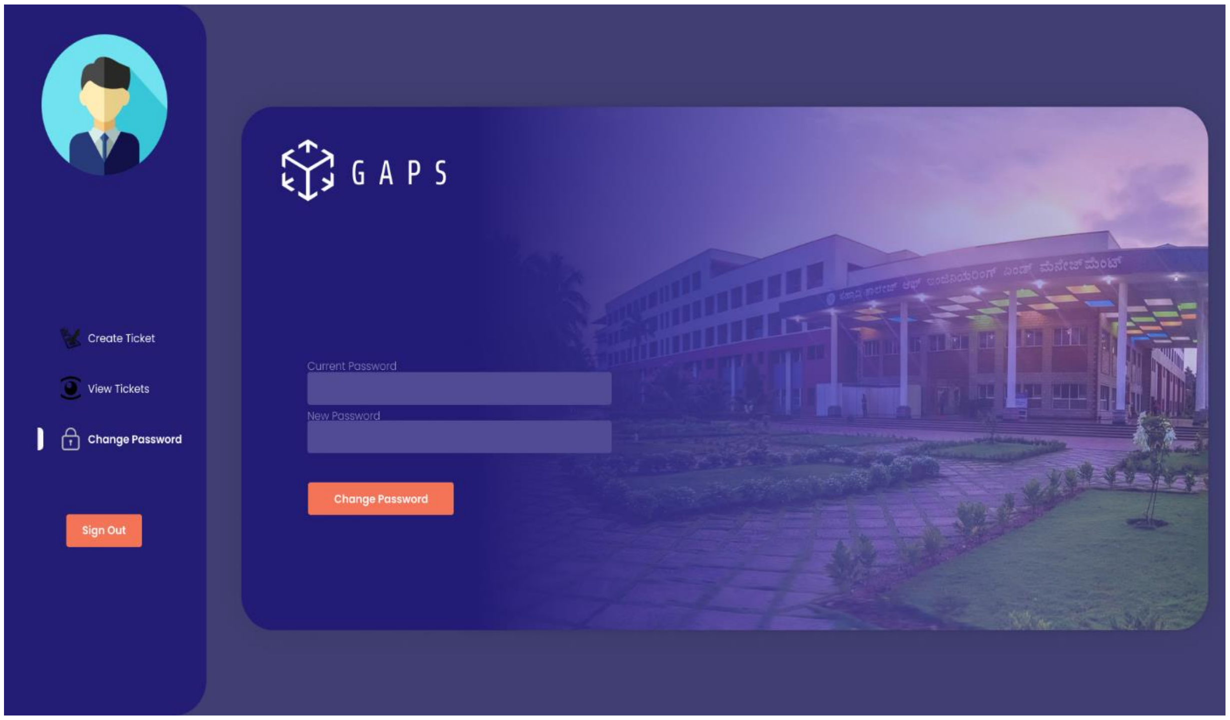


**Fig:-5.3.**3

In fig:5.3.1 The STUDENT will be able to create the ticket.  Then he'll/she'll be able to view the tickets in his/her view tickets section as you can see above in fig: 5.3.2

## 5.4 HEAD_SSN Page

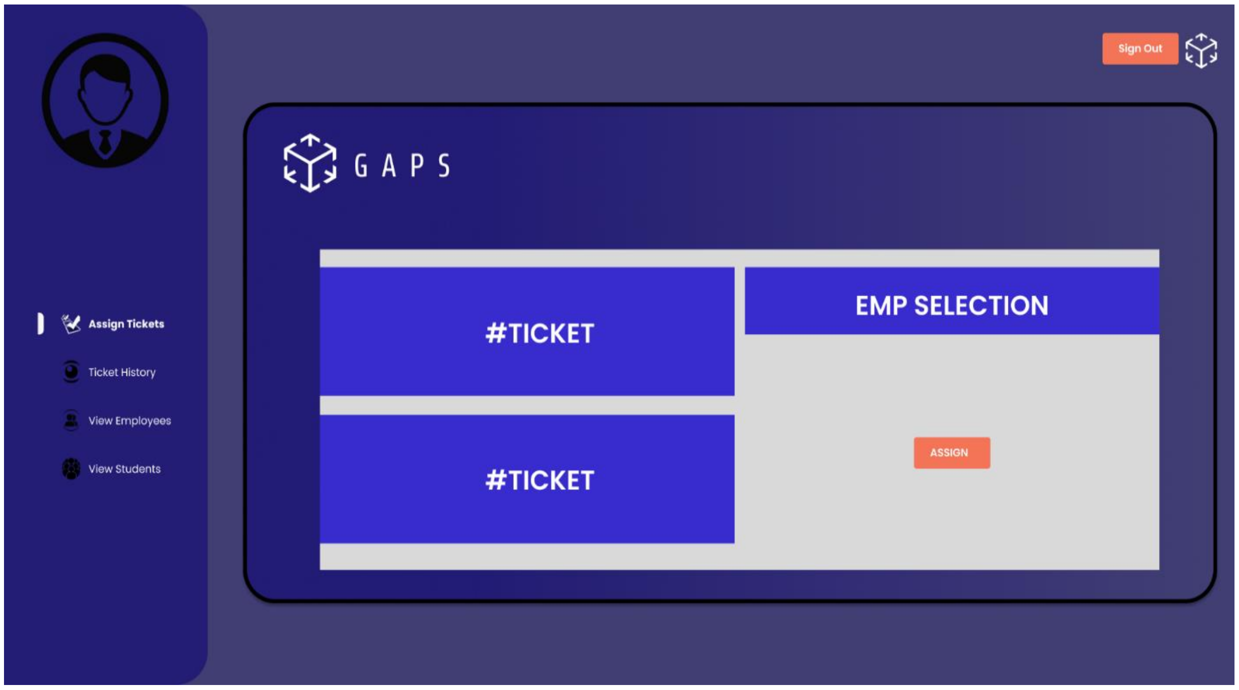


**Fig:-5.4.1**

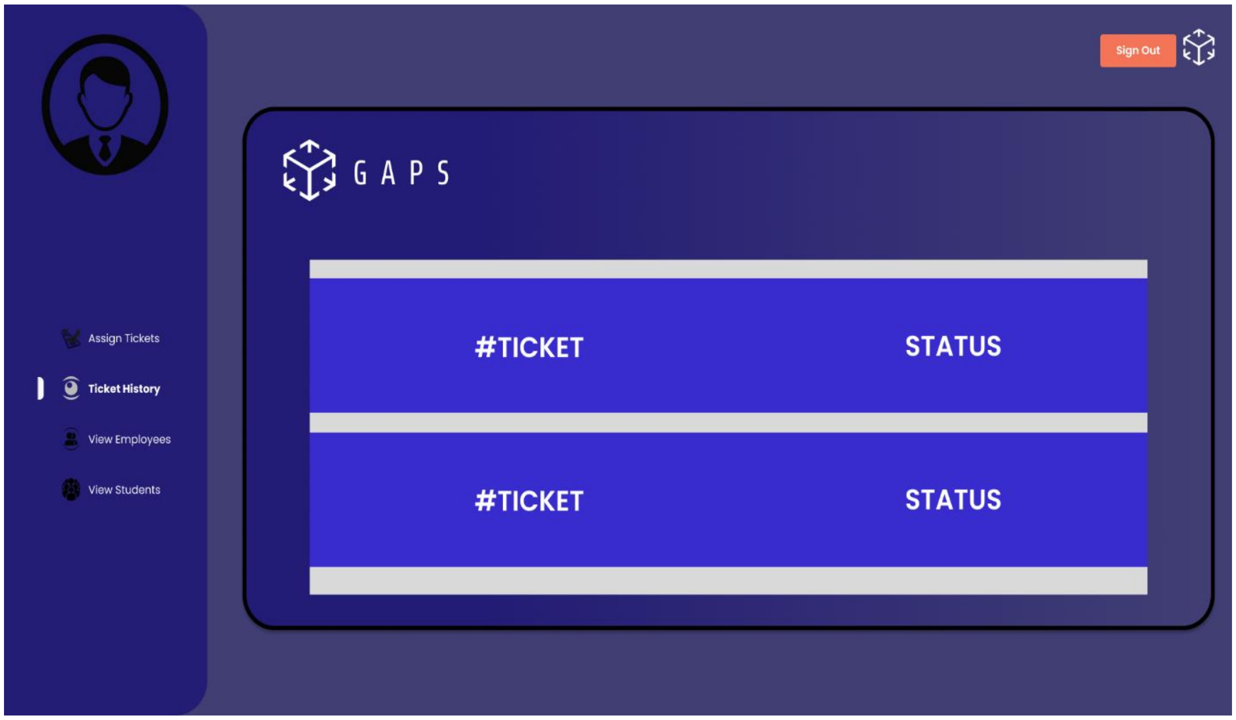


**Fig:-5.4.2**

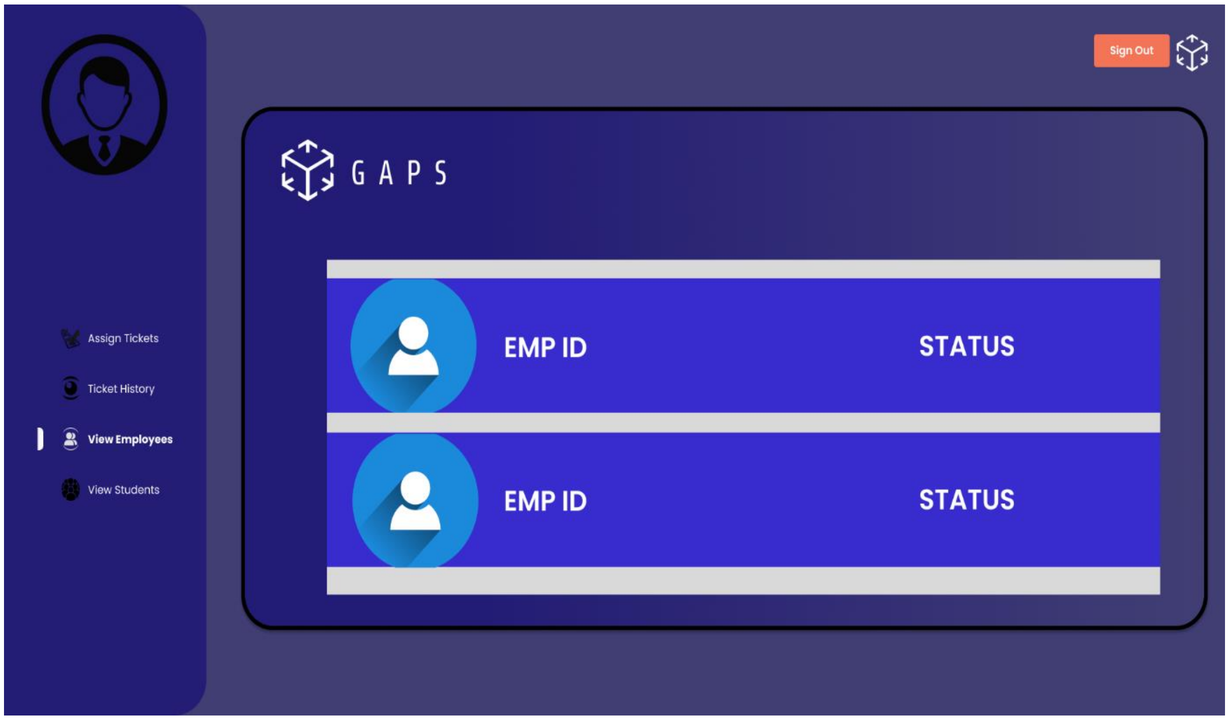


**Fig:-5.4.3**

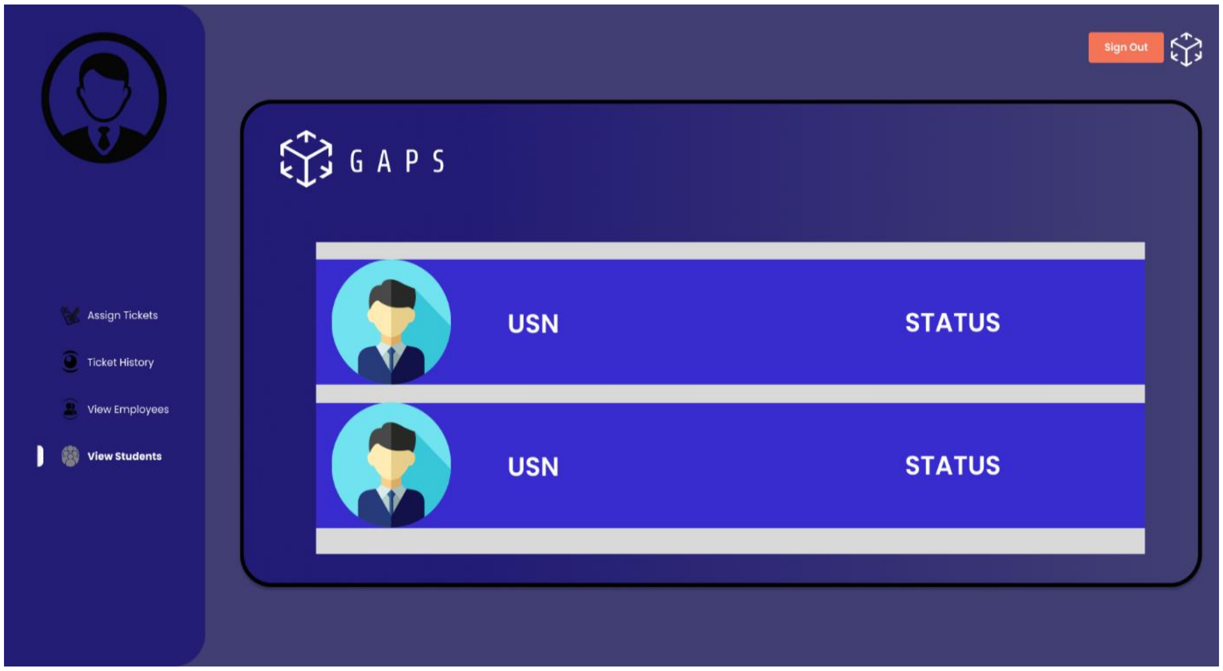


**Fig:-5.4.4**

Now, HEAD_SSN can view the student ticket details in his/her login page (fig:-5.4.1)and then he/she has to select the ticket from the given list of tickets as shown in the table above Fig;5.4.1 and then assign to the employee(fig:-5.4.3)

## 5.5 SUPER/EMPLOYEE pages

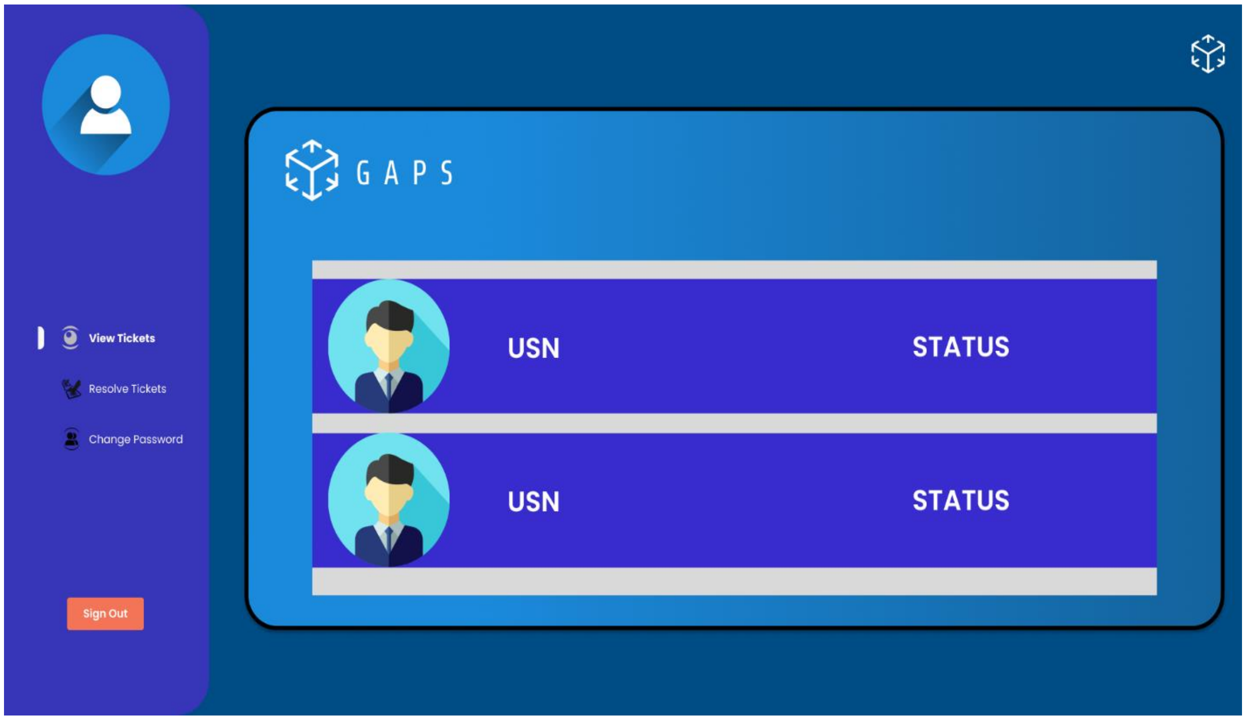


**Fig:-5.5.1**

The EMPLOYEE will be able to view the tickets that is assigned by head_ssn(fig:-5.5.1)

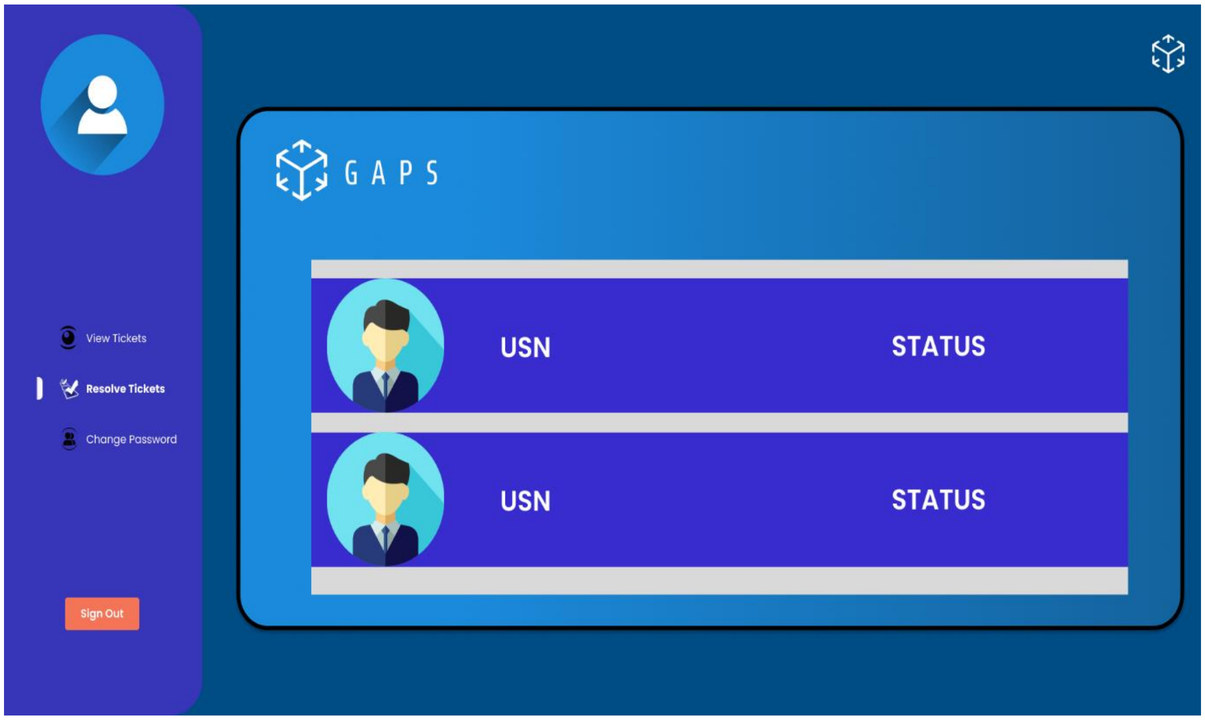


**Fig:-5.5.2**

Now, employee will resolve these tickets and time to time update the status the as shown in above fig 5.5.2

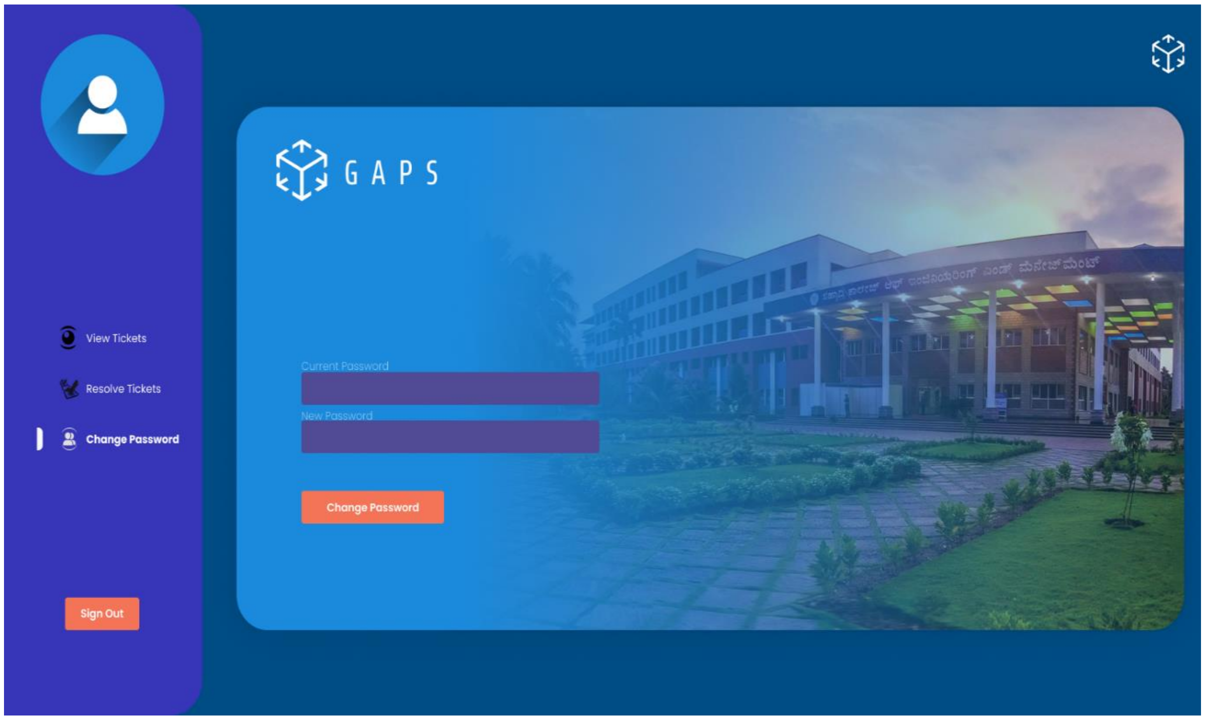


**Fig:-5.5.3**

## OPERATIONS (INSERTION , DELETION, UPDATION & SELECTION, LOGOUT)

DIFFERENT OPERATIONS :

Insertion : For Insertion, the student to insert the details of the problem in the create section on the left hand side of Dashboard

Deletion: For Deletion, the student has to select the ticket if it is resolved from the table mention in the Dashboard in Figure:5.8.1 And then clicking the accept button if he is satisfied

Updation and Selection: For Updation, the Head_ssn has to select an employee and assign for a ticket table in the dashboard and then employee can insert the updated information of ticket into the database by clicking on the update button and add comments button

Log-out : For Logging-out from the Dashboard, the student,head_ssn,employee has to click the log-out button

CHAPTER 6

# CONCLUSION

As a resolution to the problem stated above; we propose a multi-viewed app with 3 views;

- An User view to lodge and track complains or resolutions
- An Admin view based on the selected “type” allow assortment to departments and assorting the right person from the department who is responsible for the issue in consideration of
- A TicketSuper view who gets assigned the problems, can add comments and update status of the ticket live time.

The relations required for the same would be

- An user table (Students in the particular implementation) to prevent spam queries and unnecessary tickets
- A ticket table and a sub comment table to store an auto generated ticket id against all history of comments, the type of ticket, dept assigned, status etc
- An employee table to decide the type of access (Admin or TicketSuper) to see assigned problems, and details for contact
- A department table with concerned heads to be notified about tickets pending to be assigned to a super or about tickets that have been pending for a long time and to let the user know about the departments in charge of their issue currently