UCS312
DATABASE MANAGEMENT SYSTEMS

Student Management System-Project

Submitted by

SAMBHAVYA SETHI

102104073

3EE3

Submitted to:

Dr. Ravinder Kaur

ASSISTANT PROFESSOR

Computer Science and Engineering Department

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB INDIA

Jan-May 2024

Table Of Contents

1.INTRODUCTION

1.1 OBJECTIVES:

- The main objective of the project is to design and develop a user friendly-system
- Easy to use and an efficient computerized system.
- To develop an accurate and flexible system, it will eliminate data redundancy.
- To study the functioning of Students management System.
- To make a software fast in processing, with good user interface.
- To make software with good user interface so that user can change it and it should be used for a long time without error and maintenance.
- To provide synchronized and centralized farmer and seller database.
- Computerization can be helpful as a means of saving time and money.
- To provide better Graphical User Interface (GUI).
- Less chances of information leakage.
- Provides Security to the data by using login and password method.
- To provide immediate storage and retrieval of data and information.
- Improving arrangements for students coordination.
- Reducing paperwork.

2.PROPOSED SYSTEM

While there has been no consensus on the definition of Students Management in the literature, people have proposed that researchers adopt the below definition to allow for the coherent development of theory in the colleges. In order to have a successful students management, we need to make many decisions related to the flow of marks, attendance, and data. Each records should be added in a way to increase the scalability. Student management is more complex in colleges and other universities because of the impact on people's number requiring adequate and accurate information of students need.

3.DATABASE DESIGN

3.1 software Used:

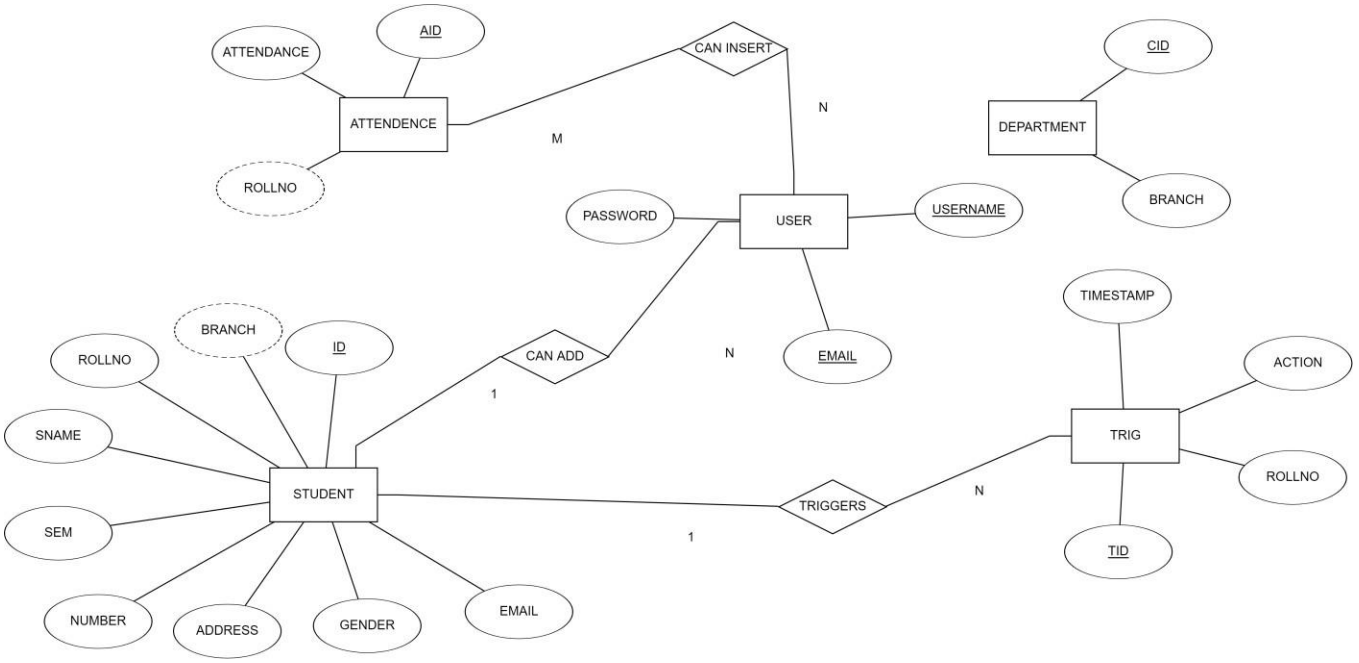
Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.7) , SQLAlchemy,

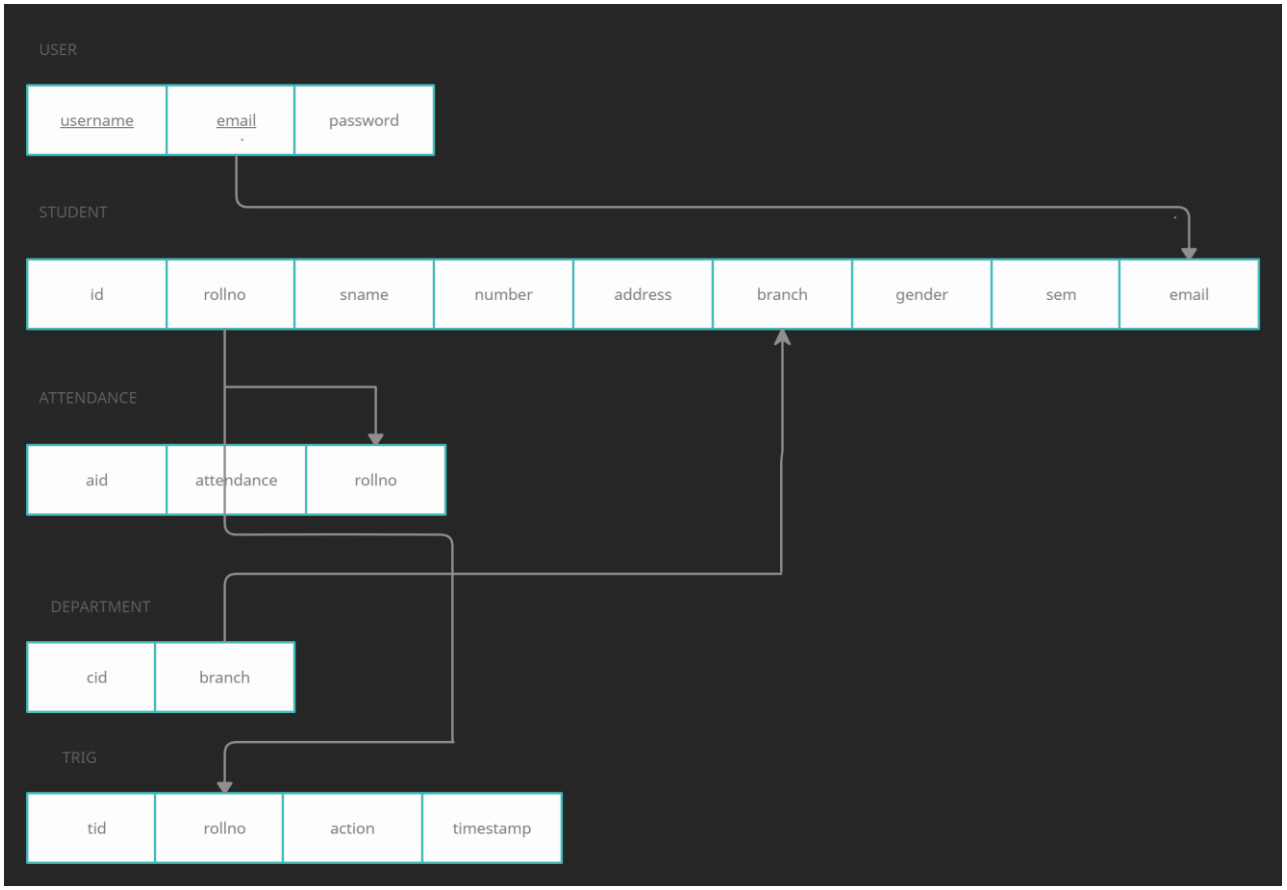
- Operating System: Windows 11
- Google Chrome
- XAMPP (Version-3.3.0)
- Python main editor (user interface): PyCharm Community
- workspace editor: Vs Code
- Cmd for creating a virtual environment

3.2 CONCEPTUAL DESIGN:

3.2.1 E-R DIAGRAM:



3.2.2 SCHEMA DIAGRAM:



3.3 Backend (MYSQL) DATABASE

3.3.1 Overview

A Database Management System (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems. Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

- A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
- The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.
- Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory).A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.

✓ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).

- It also maintains the integrity of the data in the database.
- The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. to the Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

SQL:

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

- In the relational model, data is stored in structures called relations or tables.

SQL statements are issued for the purpose of:

- Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes)

4. Database Normalization

4.1 Introduction to Normalization

Normalization is a process used in database design to organize data tables in such a way that redundancy and dependency issues are minimized, leading to efficient data storage and maintenance. It involves breaking down complex data structures into smaller, more manageable units while ensuring data integrity and reducing anomalies.

4.2 Application of Normalization in Student Management System

In my Student Management System project, normalization principles I applied to the database schema design to achieve the following:

Reduction of Redundancy: By breaking down the database into smaller, related tables, redundancy in data storage was minimized. For example, instead of storing department information repeatedly for each student, a separate "Departments" table was created and linked to the "Students" table, reducing redundant data.

Prevention of Anomalies: Normalization helped prevent insertion, update, and deletion anomalies by structuring the database properly. For instance, by ensuring that each piece of data is stored in only one place and is not duplicated across multiple records, we avoided inconsistencies that could arise from redundant data.

Improvement of Data Integrity: With normalized tables, data integrity constraints were enforced more effectively. For example, by separating data into smaller tables based on their functional dependencies, we ensured that each table adheres to specific rules, leading to improved data consistency and accuracy.

Simplified Data Maintenance: Normalization made the database easier to maintain and update. Changes to the database structure were localized, and modifications were made in only the relevant tables, reducing the risk of errors and inconsistencies.

By applying normalization principles to the Student Management System project, we achieved a well-structured database design that optimized data storage, ensured data integrity, and simplified data maintenance tasks.

5. Code (Frontend and Backend)

5.1 BACKEND PYHTON WITH MYSQL CODE

```
from flask import Flask,render_template,request,session,redirect,url_for,flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from werkzeug.security import generate_password_hash,check_password_hash
from flask_login import login_user,logout_user,login_manager,LoginManager
from flask_login import login_required,current_user
import json

# MY db connection
local_server= True
app = Flask(__name__)
app.secret_key='kusumachandashwini'


# this is for getting unique user access
login_manager=LoginManager(app)
login_manager.login_view='login'

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/databas_table_name
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/studentdbms'
db=SQLAlchemy(app)

# here we will create db models that is tables
class Test(db.Model):
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(100))
    email=db.Column(db.String(100))

class Department(db.Model):
    cid=db.Column(db.Integer,primary_key=True)
    branch=db.Column(db.String(100))

class Attendence(db.Model):
    aid=db.Column(db.Integer,primary_key=True)
    rollno=db.Column(db.String(100))
    attendance=db.Column(db.Integer())
class Trig(db.Model):
    tid=db.Column(db.Integer,primary_key=True)
    rollno=db.Column(db.String(100))
    action=db.Column(db.String(100))
    timestamp=db.Column(db.String(100))
class User(UserMixin,db.Model):
    id=db.Column(db.Integer,primary_key=True)
    username=db.Column(db.String(50))
    email=db.Column(db.String(50),unique=True)
    password=db.Column(db.String(1000))
class Student(db.Model):
    id=db.Column(db.Integer,primary_key=True)
```

```
    rollno=db.Column(db.String(50))
    sname=db.Column(db.String(50))
    sem=db.Column(db.Integer)
    gender=db.Column(db.String(50))
    branch=db.Column(db.String(50))
    email=db.Column(db.String(50))
    number=db.Column(db.String(12))
    address=db.Column(db.String(100))
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/studentdetails')
def studentdetails():
    # query=db.engine.execute(f"SELECT * FROM `student`")
    query=Student.query.all()
    return render_template('studentdetails.html',query=query)
@app.route('/triggers')
def triggers():
    # query=db.engine.execute(f"SELECT * FROM `trig`")
    query=Trig.query.all()
    return render_template('triggers.html',query=query)
@app.route('/department',methods=['POST','GET'])
def department():
    if request.method=="POST":
        dept=request.form.get('dept')
        query=Department.query.filter_by(branch=dept).first()
        if query:
            flash("Department Already Exist","warning")
            return redirect('/department')
        dep=Department(branch=dept)
        db.session.add(dep)
        db.session.commit()
        flash("Department Addes","success")
    return render_template('department.html')
@app.route('/addattendance',methods=['POST','GET'])
def addattendance():
    # query=db.engine.execute(f"SELECT * FROM `student`")
    query=Student.query.all()
    if request.method=="POST":
        rollno=request.form.get('rollno')
        attend=request.form.get('attend')
        print(attend,rollno)
        atte=Attendence(rollno=rollno,attendance=attend)
        db.session.add(atte)
        db.session.commit()
        flash("Attendance added","warning")
return render_template('attendance.html',query=query)
@app.route('/search',methods=['POST','GET'])
def search():
    if request.method=="POST":
        rollno=request.form.get('roll')
        bio=Student.query.filter_by(rollno=rollno).first()
        attend=Attendence.query.filter_by(rollno=rollno).first()
        return render_template('search.html',bio=bio,attend=attend)
return render_template('search.html')
@app.route("/delete/<string:id>",methods=['POST','GET'])
@login_required
```

```
def delete(id):
    post=Student.query.filter_by(id=id).first()
    db.session.delete(post)
    db.session.commit()
    # db.engine.execute(f"DELETE FROM `student` WHERE `student`.`id`={id}")
    flash("Slot Deleted Successful","danger")
    return redirect('/studentdetails')
@app.route("/edit/<string:id>",methods=['POST','GET'])
@login_required
def edit(id):
    # dept=db.engine.execute("SELECT * FROM `department`")
    if request.method=="POST":
        rollno=request.form.get('rollno')
        sname=request.form.get('sname')
        sem=request.form.get('sem')
        gender=request.form.get('gender')
        branch=request.form.get('branch')
        email=request.form.get('email')
        num=request.form.get('num')
        address=request.form.get('address')
        # query=db.engine.execute(f"UPDATE `student` SET
`rollno`='{rollno}',`sname`='{sname}',`sem`='{sem}',`gender`='{gender}',`branch`='{branch}',`email`='{email}'
,`number`='{num}',`address`='{address}'")
        post=Student.query.filter_by(id=id).first()
        post.rollno=rollno
        post.sname=sname
        post.sem=sem
        post.gender=gender
        post.branch=branch
        post.email=email
        post.number=num
        post.address=address
        db.session.commit()
        flash("Slot is Updates","success")
        return redirect('/studentdetails')
    dept=Department.query.all()
    posts=Student.query.filter_by(id=id).first()
    return render_template('edit.html',posts=posts,dept=dept)
@app.route('/signup',methods=['POST','GET'])
def signup():
    if request.method == "POST":
        username=request.form.get('username')
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).first()
        if user:
            flash("Email Already Exist","warning")
            return render_template('/signup.html')
        # encpassword=generate_password_hash(password)
        # new_user=db.engine.execute(f"INSERT INTO `user` (`username`,`email`,`password`) VALUES
('{username}','{email}','{encpassword}')")
        # this is method 2 to save data in db
        newuser=User(username=username,email=email,password=password)
        db.session.add(newuser)
        db.session.commit()
        flash("Signup Succes Please Login","success")
        return render_template('login.html')
```

```
    return render_template('signup.html')

@app.route('/login',methods=['POST','GET'])
def login():
    if request.method == "POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).first()
        # if user and check_password_hash(user.password,password):
        if user and user.password == password:
            login_user(user)
            flash("Login Success","primary")
            return redirect(url_for('index'))
        else:
            flash("invalid credentials","danger")
            return render_template('login.html')
    return render_template('login.html')
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash("Logout SuccessFul","warning")
    return redirect(url_for('login'))
@app.route('/addstudent',methods=['POST','GET'])
@login_required
def addstudent():
    # dept=db.engine.execute("SELECT * FROM `department`")
    dept=Department.query.all()
    if request.method=="POST":
        rollno=request.form.get('rollno')
        sname=request.form.get('sname')
        sem=request.form.get('sem')
        gender=request.form.get('gender')
        branch=request.form.get('branch')
        email=request.form.get('email')
        num=request.form.get('num')
        address=request.form.get('address')
        # query=db.engine.execute(f"INSERT INTO `student`
(`rollno`,`sname`,`sem`,`gender`,`branch`,`email`,`number`,`address`) VALUES
('{rollno}','{sname}','{sem}','{gender}','{branch}','{email}','{num}','{address}')")
query=Student(rollno=rollno,sname=sname,sem=sem,gender=gender,branch=branch,email=email,number=num,
address=address)
        db.session.add(query)
        db.session.commit()
        flash("Booking Confirmed","info")
    return render_template('student.html',dept=dept)
@app.route('/test')
def test():
    try:
        Test.query.all()
        return 'My database is Connected'
    except:
        return 'My db is not Connected'
app.run(debug=True)
```

5.2 FRONT END CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
<title>{% block title %}
{% endblock title %}</title>
 <meta content="" name="description">
 <meta content="" name="keywords">
{% block style %}
{% endblock style %}
<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,700,700i|Raleway:3
00,400,500,700,800" rel="stylesheet">
  <!-- Vendor CSS Files -->
  <link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="static/assets/vendor/venobox/venobox.css" rel="stylesheet">
  <link href="static/assets/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet">
  <link  href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css"  rel="stylesheet">
  <link href="static/assets/vendor/aos/aos.css" rel="stylesheet">
  <!-- Template Main CSS File -->
  <link href="static/assets/css/style.css" rel="stylesheet">
</head>
<body>
  <!-- ======= Header ======= -->
  <header id="header">
    <div class="container">
   <div id="logo" class="pull-left">
       <a href="/" class="scrollto">S.M.S</a>
     </div>
     <nav id="nav-menu-container">
       <ul class="nav-menu">
         <li class="{% block home %}
         {% endblock home %}"><a href="/">Home</a></li>
 <li><a href="/addstudent">Students</a></li>
 <li><a href="/addattendance">Attendance</a></li>
 <li><a href="/department">Department</a></li>
 <li><a href="/triggers">Records</a></li>
```

```
<li><a href="/studentdetails">Student Details</a></li>

<li><a href="/search">Search</a></li>
        <li><a href="/about">About</a></li>
   {% if current_user.is_authenticated  %}

        <li class="buy-tickets"><a href="">Welcome</a></li>

          <li class="buy-tickets"><a href="/logout">Logout</a></li>

        {% else %}

        <li class="buy-tickets"><a href="/signup">Signin</a></li>
        {% endif %}

       </ul>

     </nav><!-- #nav-menu-container -->

     </div>

 </header><!-- End Header -->
 <!-- ======= Intro Section ======= -->

<section id="intro">

   <div class="intro-container" data-aos="zoom-in" data-aos-delay="100">

     <h1 class="mb-4 pb-0">STUDENT MANAGEMENT SYSTEM </span> </h1>

     <p class="mb-4 pb-0">DBMS Mini Project Using Flask & MYSQL</p>


     <a href="" class="about-btn scrollto">View More</a>

   </div>

 </section><!-- End Intro Section -->

 <main id="main">
 {% block body %}
{% with messages=get_flashed_messages(with_categories=true) %}

{% if messages %}

{% for category, message in messages %}
<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">

   {{message}}
</div>
 {% endfor %}

 {% endif %}

 {% endwith %}

 {% endblock body %}
 <a href="#" class="back-to-top"><i class="fa fa-angle-up"></i></a>
```

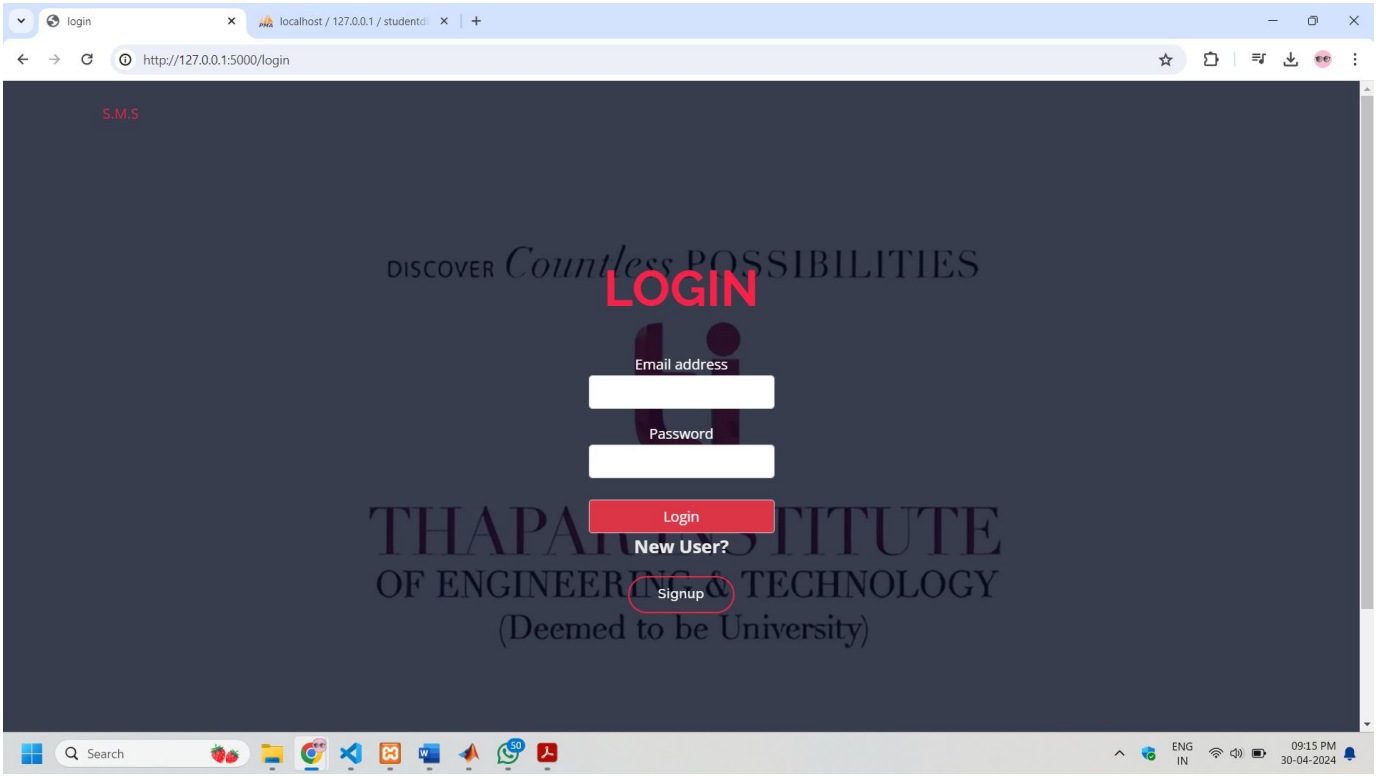
```
<!-- Vendor JS Files -->
  <script src="static/assets/vendor/jquery/jquery.min.js"></script>
  <script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>
  <script src="static/assets/vendor/php-email-form/validate.js"></script>
  <script src="static/assets/vendor/venobox/venobox.min.js"></script>
  <script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>
  <script src="static/assets/vendor/superfish/superfish.min.js"></script>
  <script src="static/assets/vendor/hoverIntent/hoverIntent.js"></script>
  <script src="static/assets/vendor/aos/aos.js"></script>
  <!-- Template Main JS File -->
  <script src="static/assets/js/main.js"></script>

</body>
</html> 2.Students.html
{% extends 'base.html' %}
{% block title %}
Add Students
{% endblock title %}
{% block body %}
 <h3 class="text-center"><span>Add Student Details</span> </h3>
{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}
<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
   {{message}}
</div>
  {% endfor %}
  {% endif %}
  {% endwith %}
<br>
<div class="container">
<div class="row">
<div class="col-md-4"></div>
<div class="col-md-4">
<form action="/addstudent" method="post">
<div class="form-group">
```

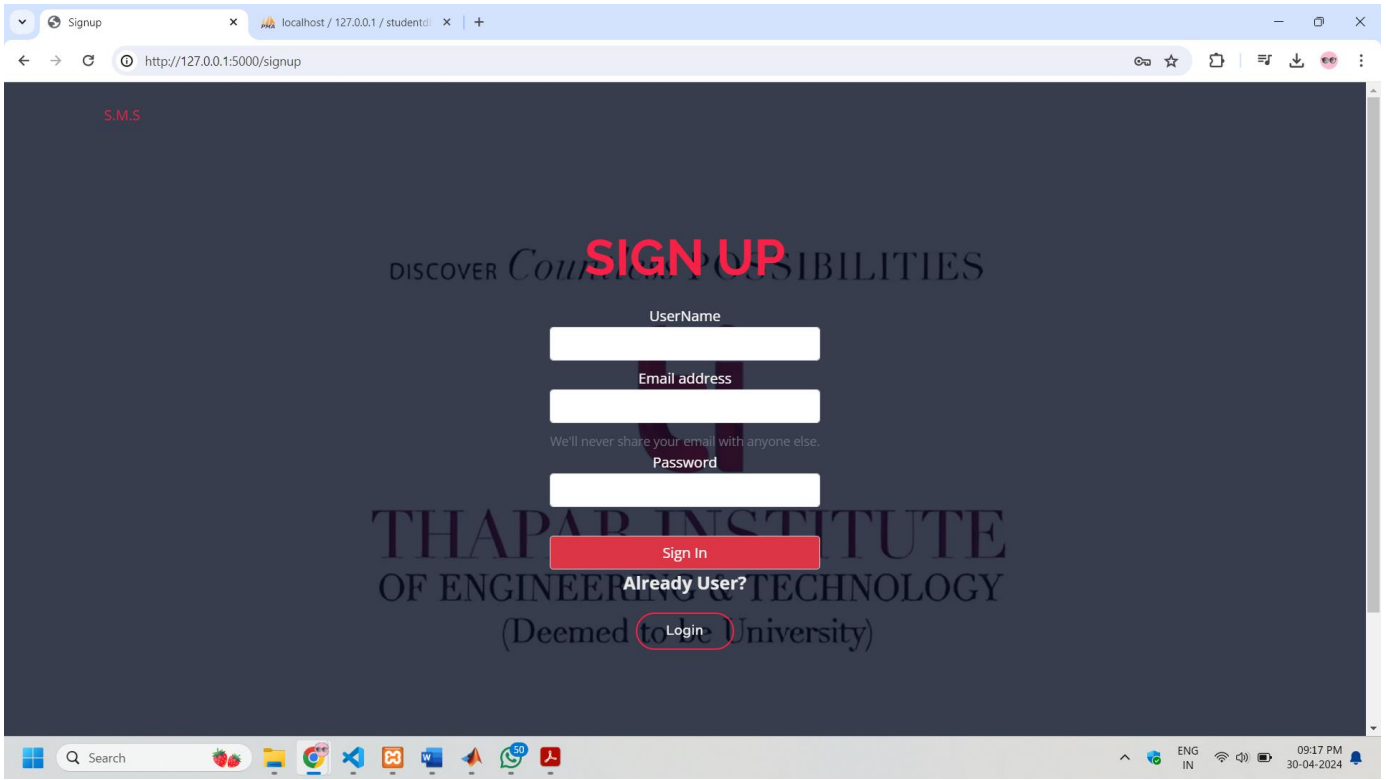
```
<label for="rollno">Roll Number</label>
<input type="text" class="form-control" name="rollno" id="rollno">
</div>
<br>
<div class="form-group">
<label for="sname">Student Name</label>
<input type="text" class="form-control" name="sname" id="sname">
</div>
<br>
<div class="form-group">
<label for="sem">Sem</label>
<input type="number" class="form-control" name="sem" id="sem">
</div>
<br>
<div class="form-group">
<select  class="form-control" id="gender" name="gender"  required>
        <option selected>Select Gender</option>
       <option value="male">Male</option>
        <option value="female">Female</option>
      </select>
 </div>
<br>
<div class="form-group">
<select class="form-control" id="branch" name="branch"  required>
        <option selected>Select Branch</option>
        {% for d in dept  %}
        <option value="{{d.branch}}">{{d.branch}}</option>
        {% endfor %}
      </select>
 </div>
<br>
<div class="form-group">
<label for="email">Email</label>
<input type="email" class="form-control" name="email" id="email">
</div>
<br>
<div class="form-group">
```

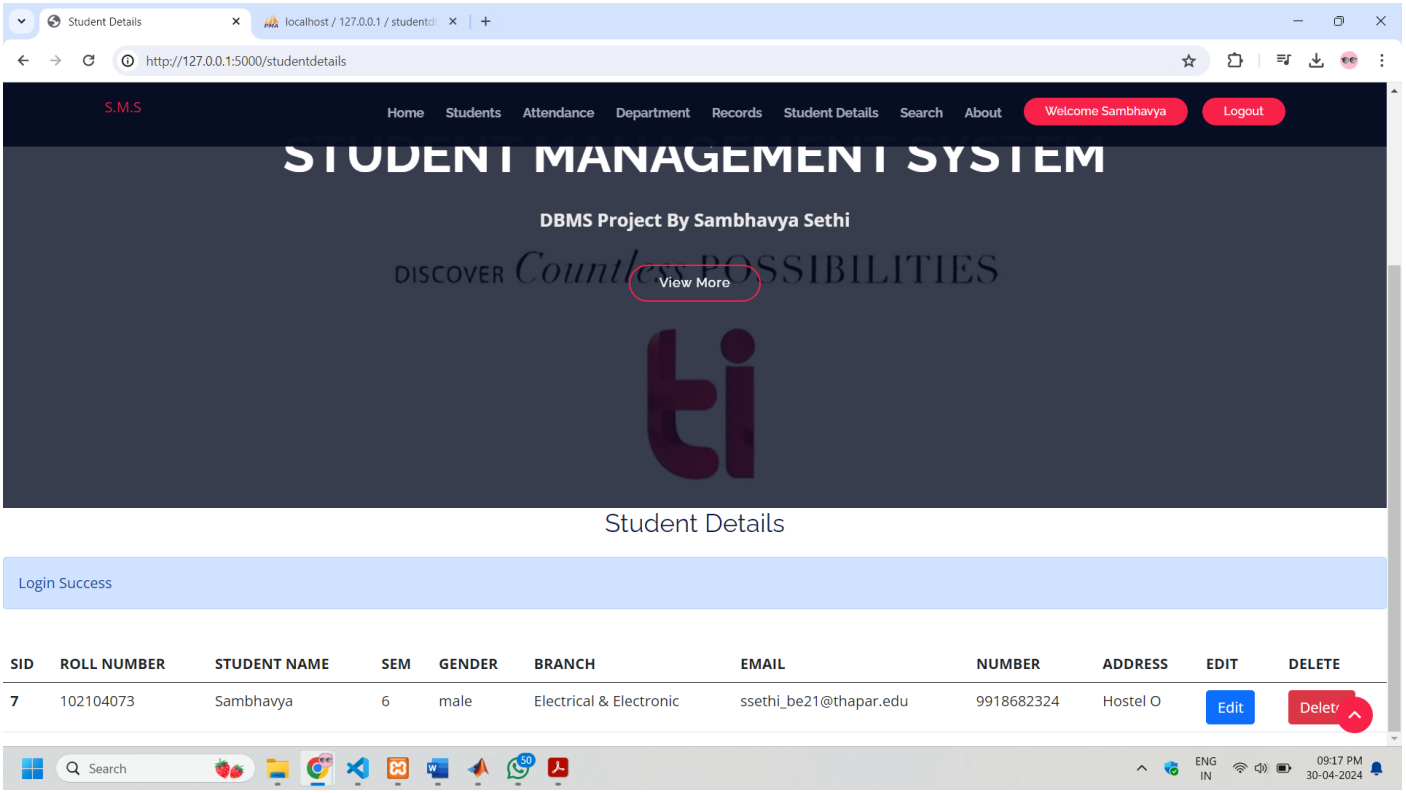
```
<label for="num">Phone Number</label>
<input type="number" class="form-control" name="num" id="num">
</div>
<br>
<div class="form-group">
<label for="address">Address</label>
<textarea class="form-control" name="address" id="address"></textarea>
</div>
<br>
<button type="submit" class="btn btn-danger btn-sm btn-block">Add Record</button>
</form>
<br>
<br>
</div>
<div class="col-md-4"></div>
</div></div>
{% endblock body %}
```

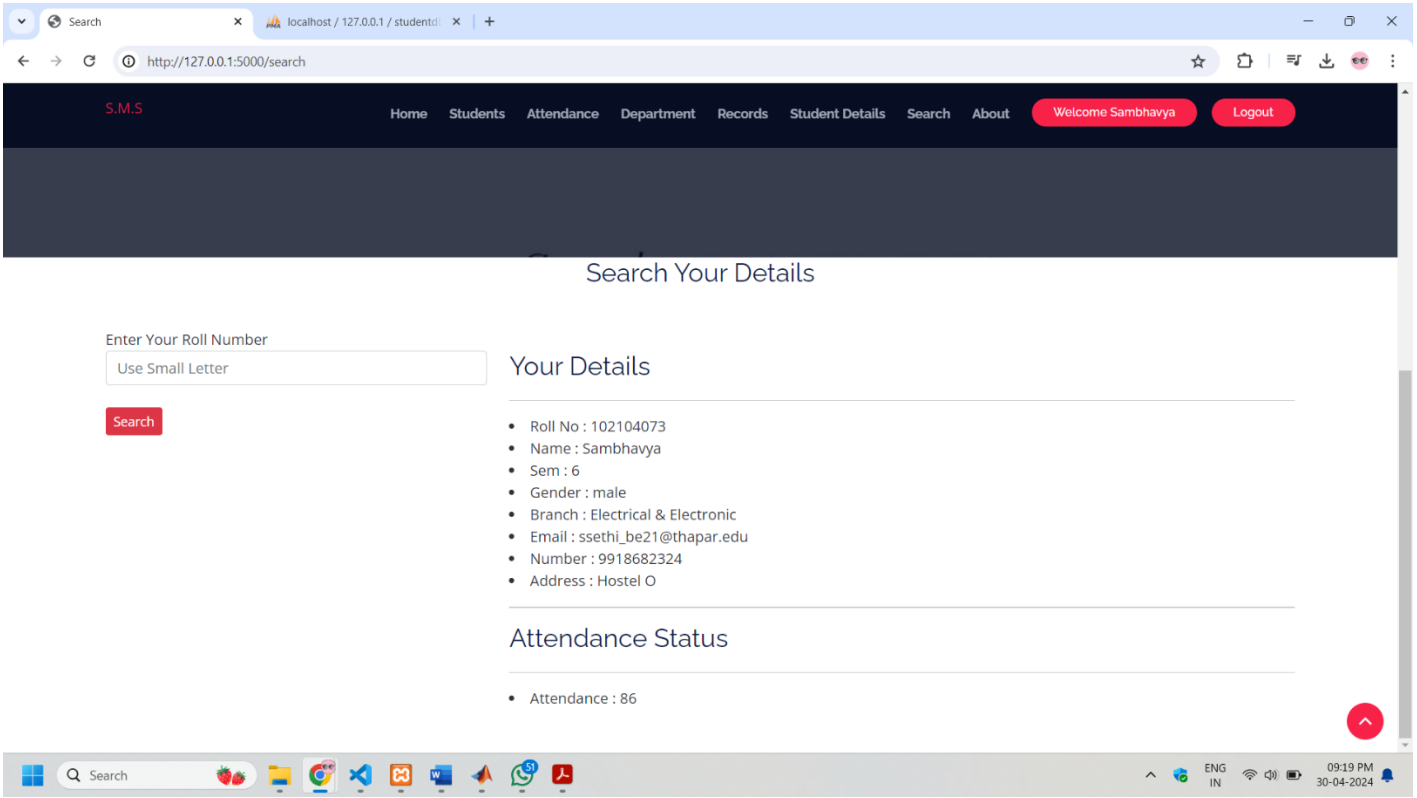
6.USER INTERFACE

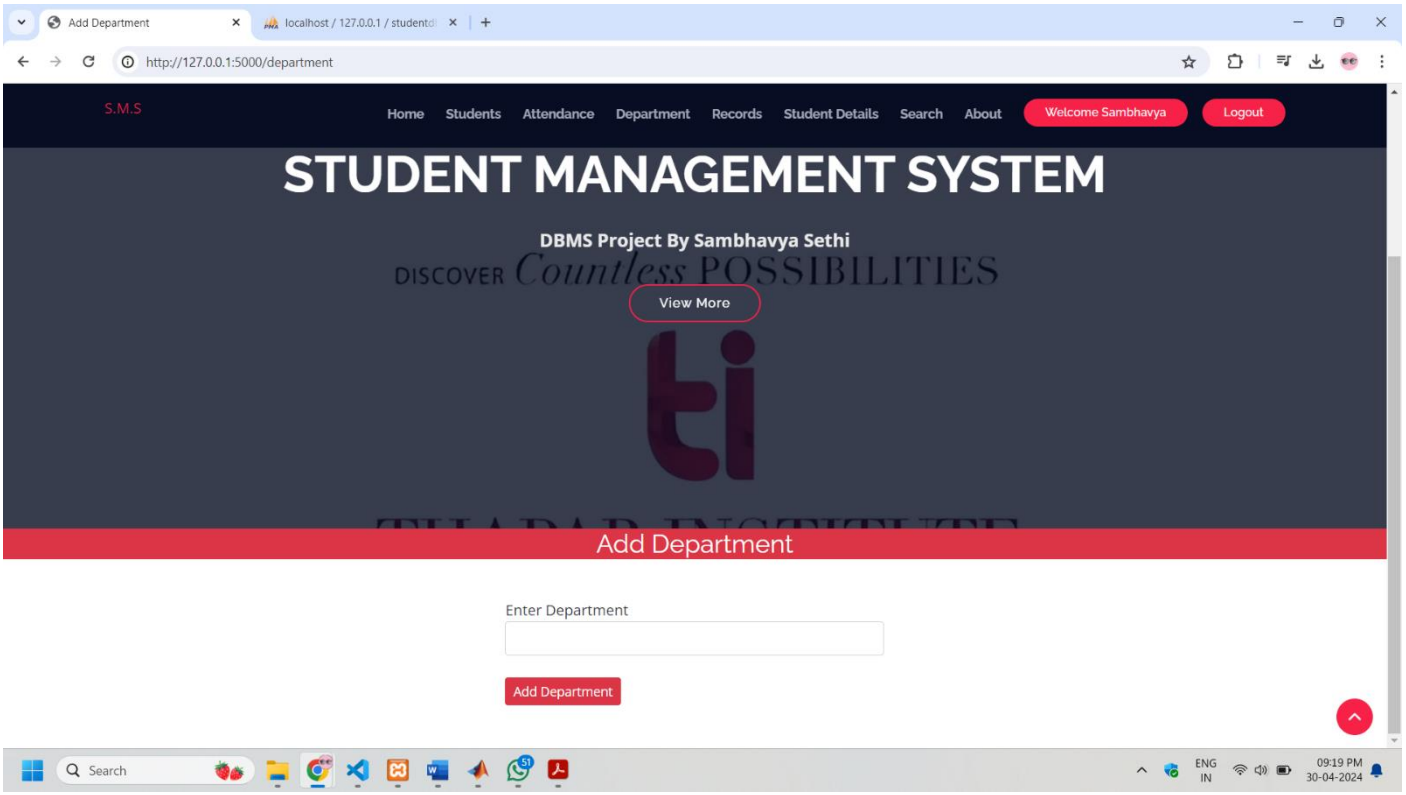
6.1 SCREEN SHOTS

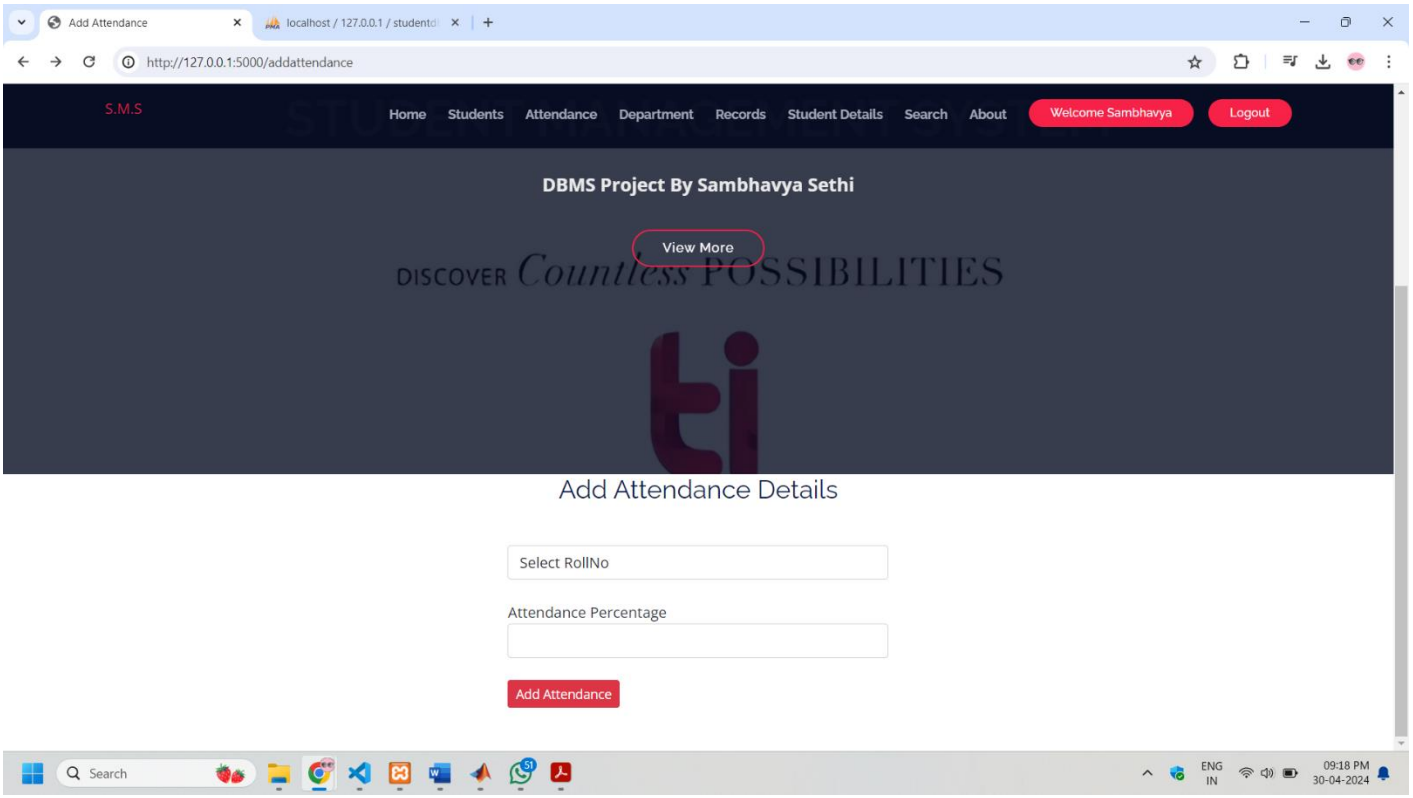


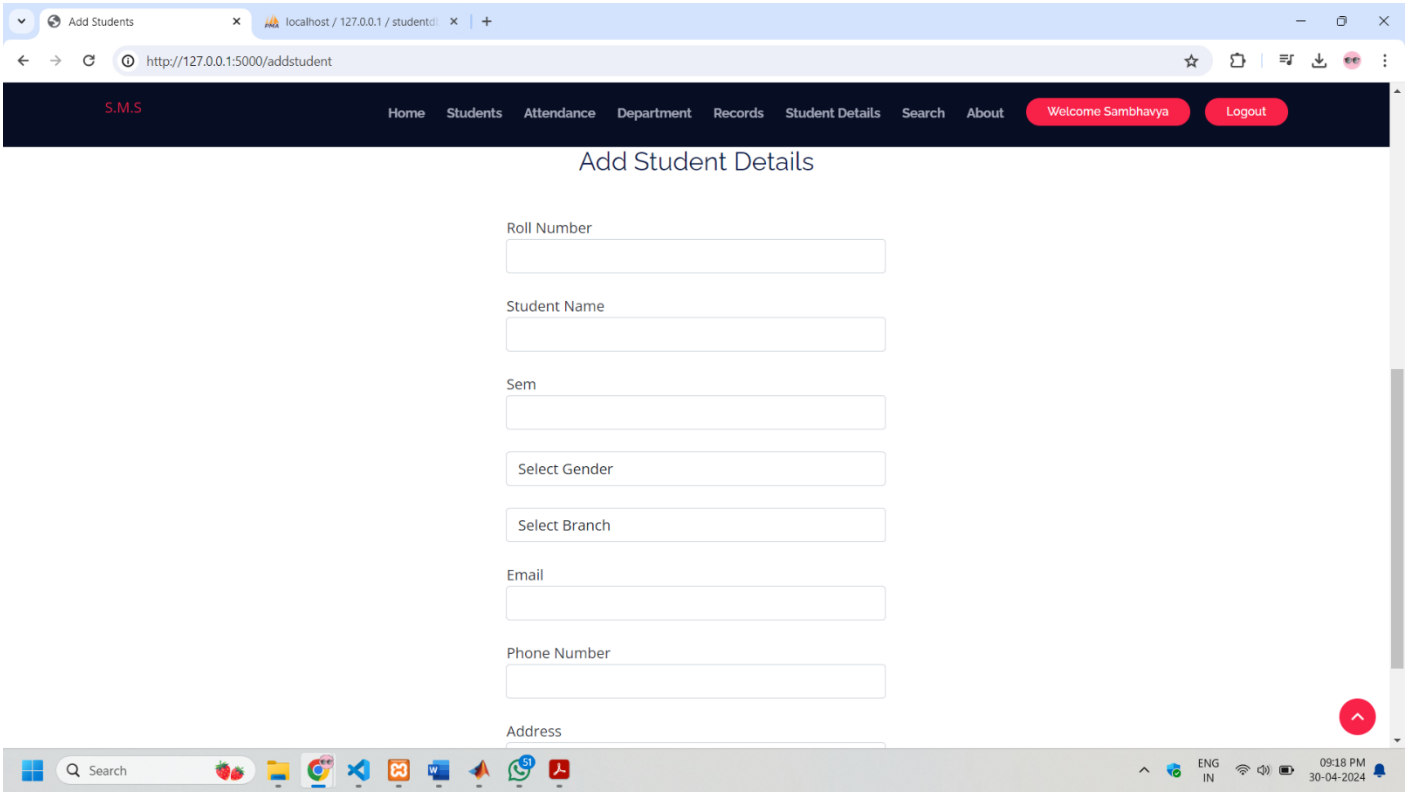


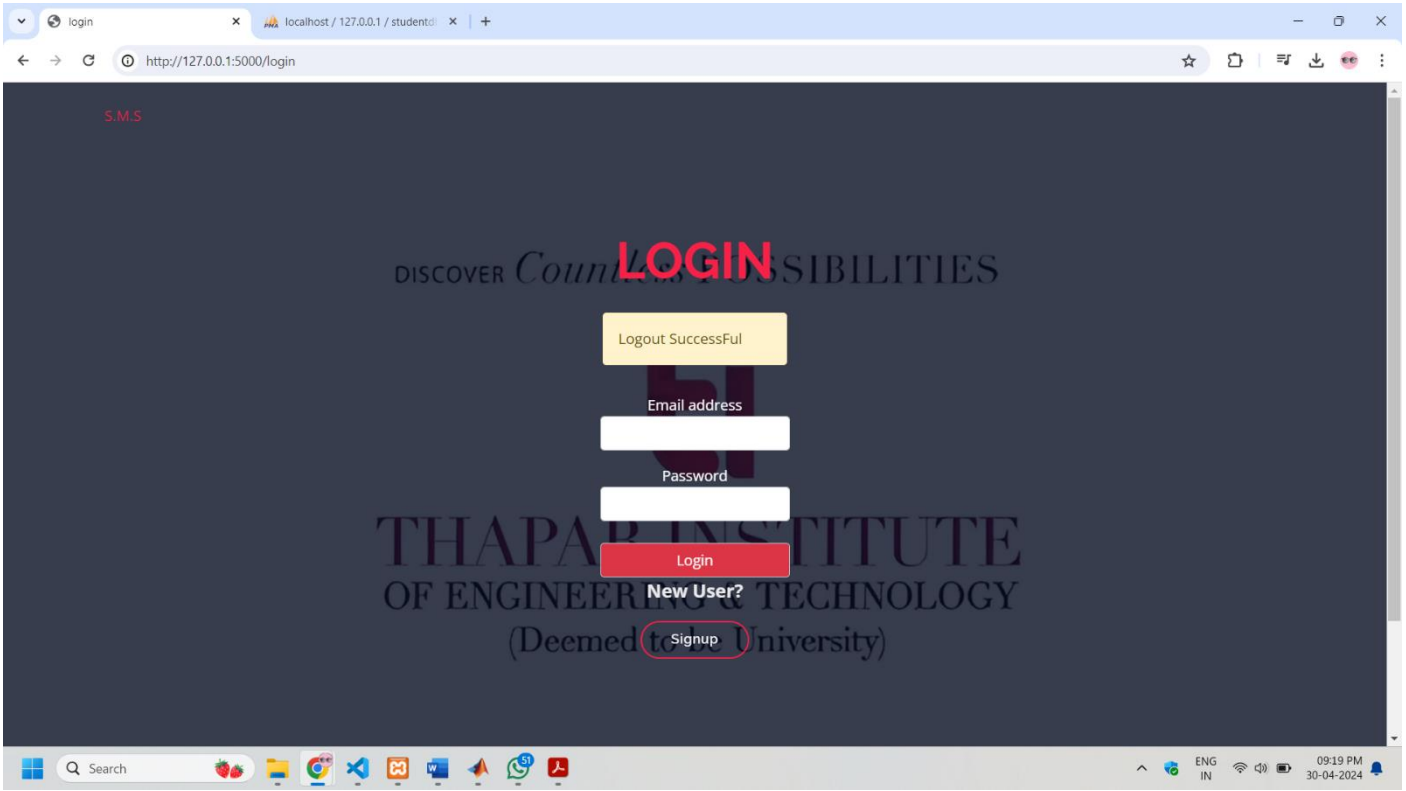

Student Details
http://127.0.0.1:5000/studentdetails
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome Sambhavya
Logout
STUDENT MANAGEMENT SYSTEM
DBMS Project By Sambhavya Sethi
DISCOVER
POSSIBILITIES
View More
Student Details
Login Success
SID
ROLL NUMBER
STUDENT NAME
SEM
GENDER
BRANCH
EMAIL
NUMBER
ADDRESS
EDIT
DELETE
7
102104073
Sambhavya
6
male
Electrical & Electronic
ssethi_be21@thapar.edu
9918682324
Hostel O
Edit
Delete
Search
ENG
IN
09:17 PM
30-04-2024


Search
http://127.0.0.1:5000/search
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome Sambhavya
Logout
Search Your Details
Enter Your Roll Number
Use Small Letter
Search
Your Details
Roll No : 102104073
Name : Sambhavya
Sem : 6
Gender : male
Branch : Electrical & Electronic
Email : ssethi_be21@thapar.edu
Number : 9918682324
Address : Hostel O
Attendance Status
Attendance : 86
Search
ENG
IN
09:19 PM
30-04-2024


Add Department
http://127.0.0.1:5000/department
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome Sambhavya
Logout
STUDENT MANAGEMENT SYSTEM
DBMS Project By Sambhavya Sethi
DISCOVER Countless POSSIBILITIES
View More
Add Department
Enter Department
Add Department

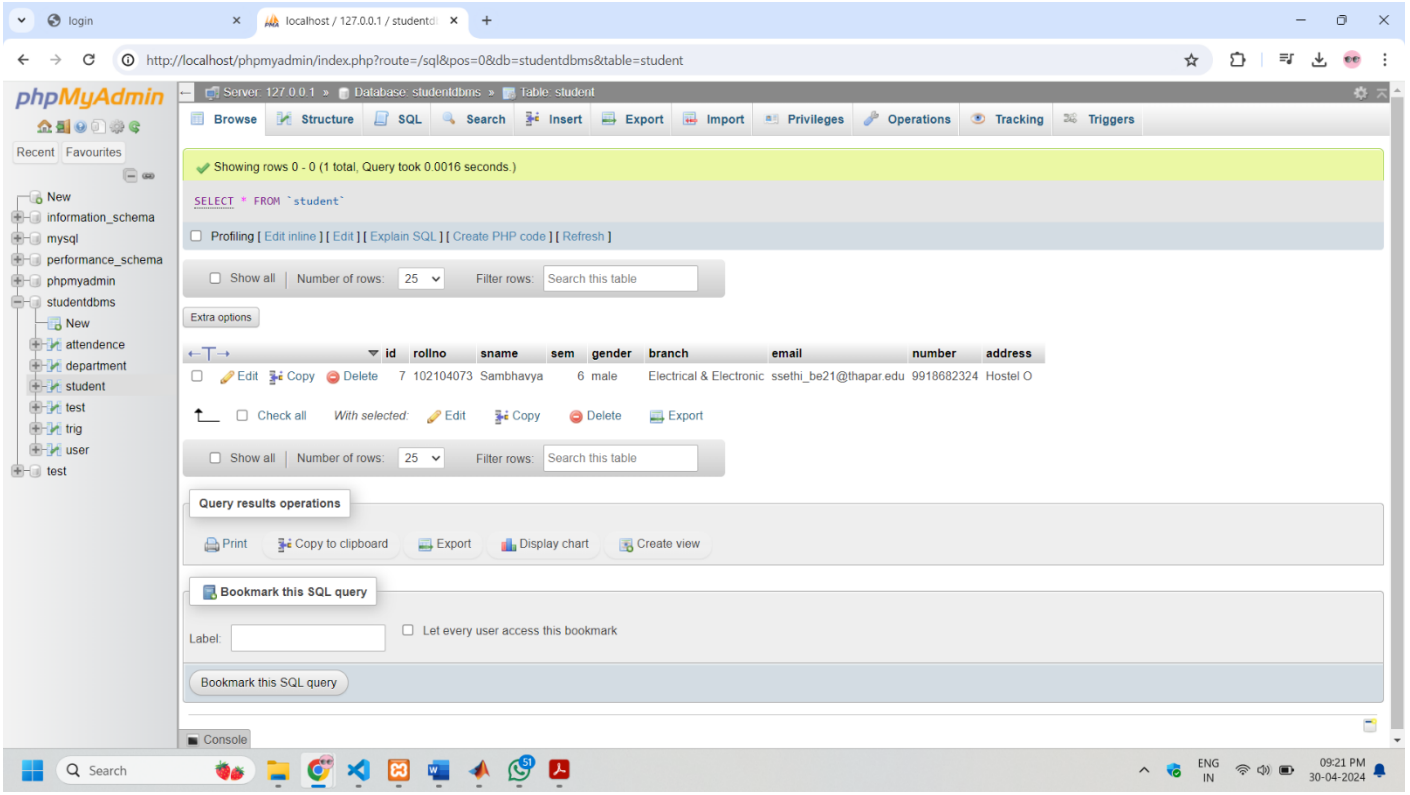

Add Attendance
http://127.0.0.1:5000/addattendance
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome Sambhavya
Logout
DBMS Project By Sambhavya Sethi
View More
DISCOVER Countless POSSIBILITIES
Add Attendance Details
Select RollNo
Attendance Percentage
Add Attendance

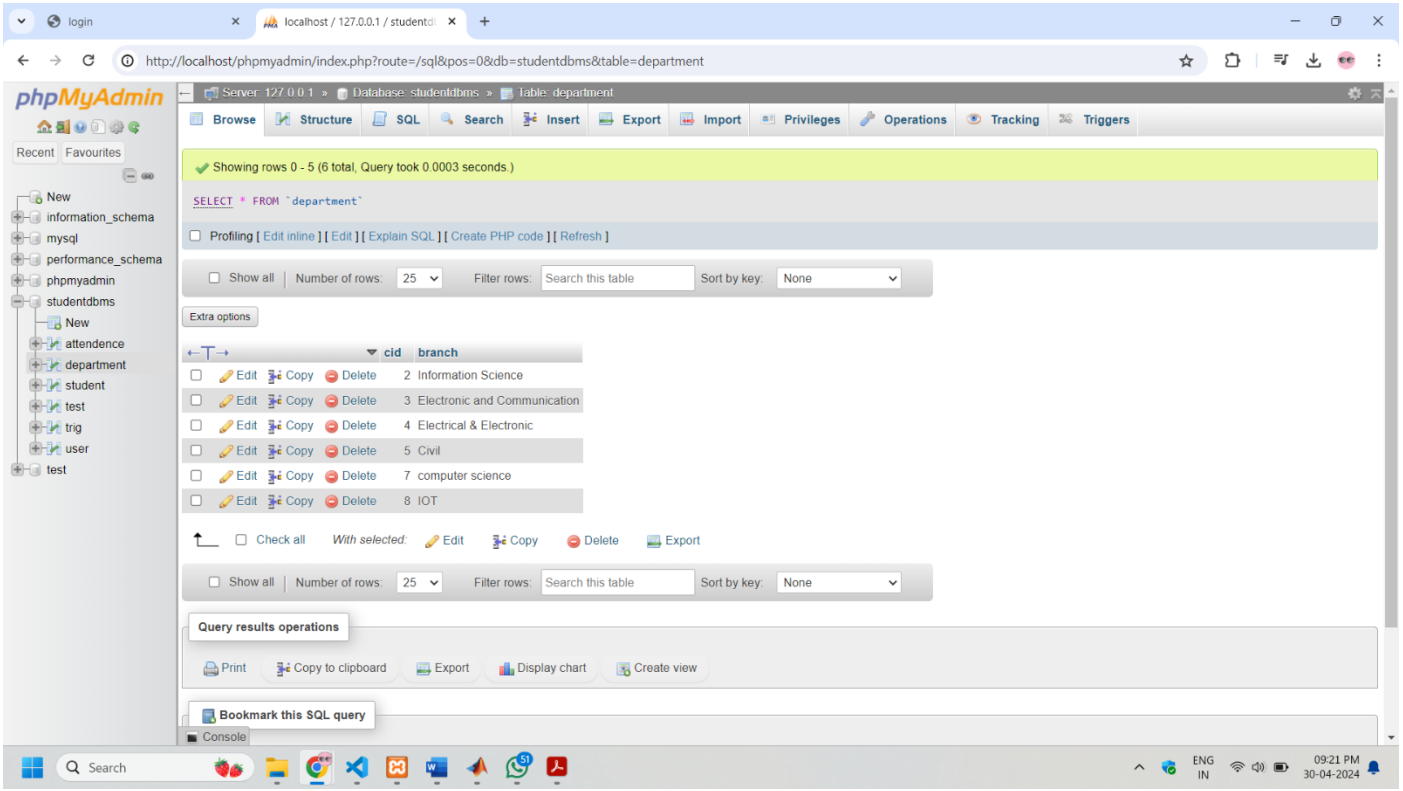

Add Students
localhost / 127.0.0.1 / studentdb
http://127.0.0.1:5000/addstudent
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome Sambhavya
Logout
Add Student Details
Roll Number
Student Name
Sem
Select Gender
Select Branch
Email
Phone Number
Address
Search
ENG
IN
09:18 PM
30-04-2024

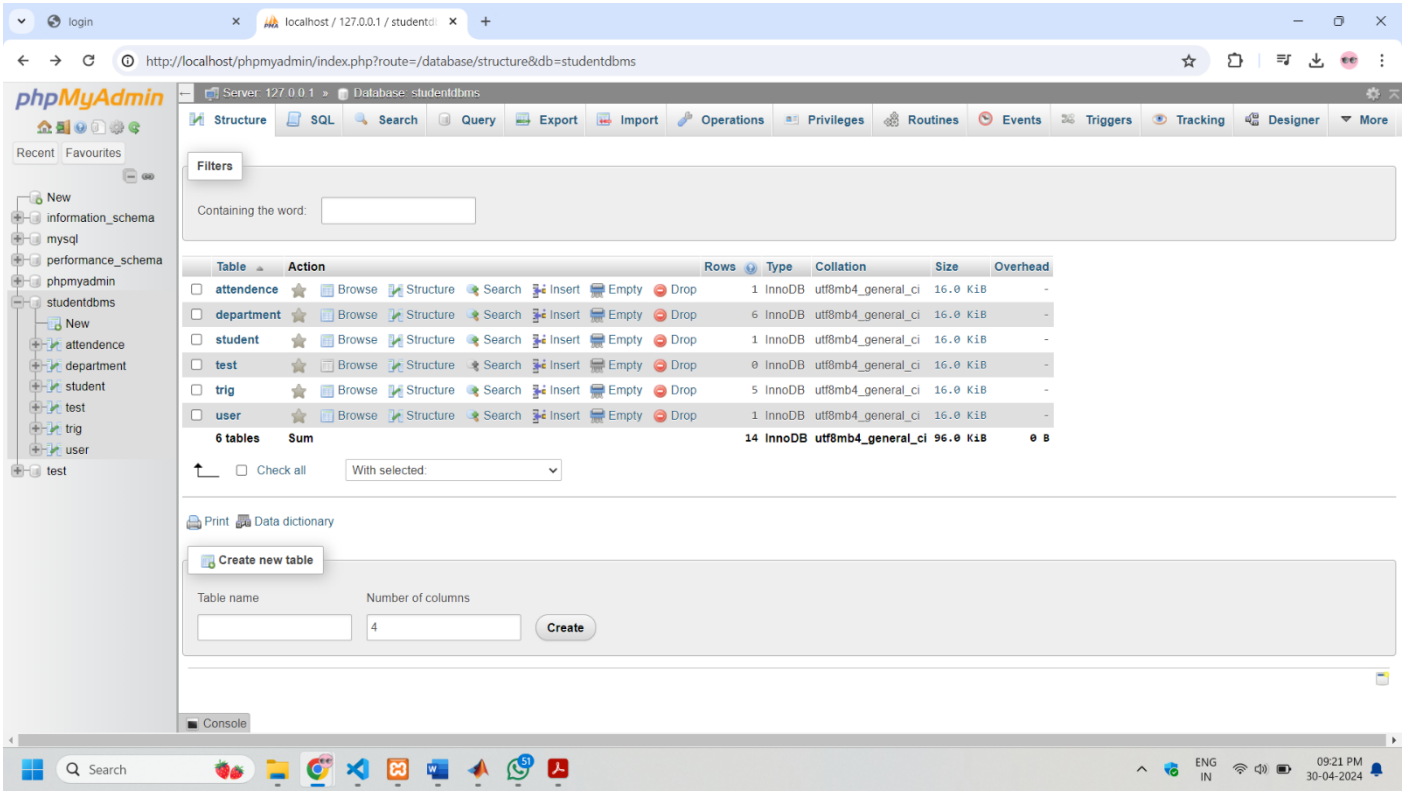

login
localhost / 127.0.0.1 / studentdb
http://127.0.0.1:5000/login
S.M.S
LOGIN
Logout SuccessFul
Email address
Password
Login
New User?
Signup
Search
ENG
IN
09:19 PM
30-04-2024

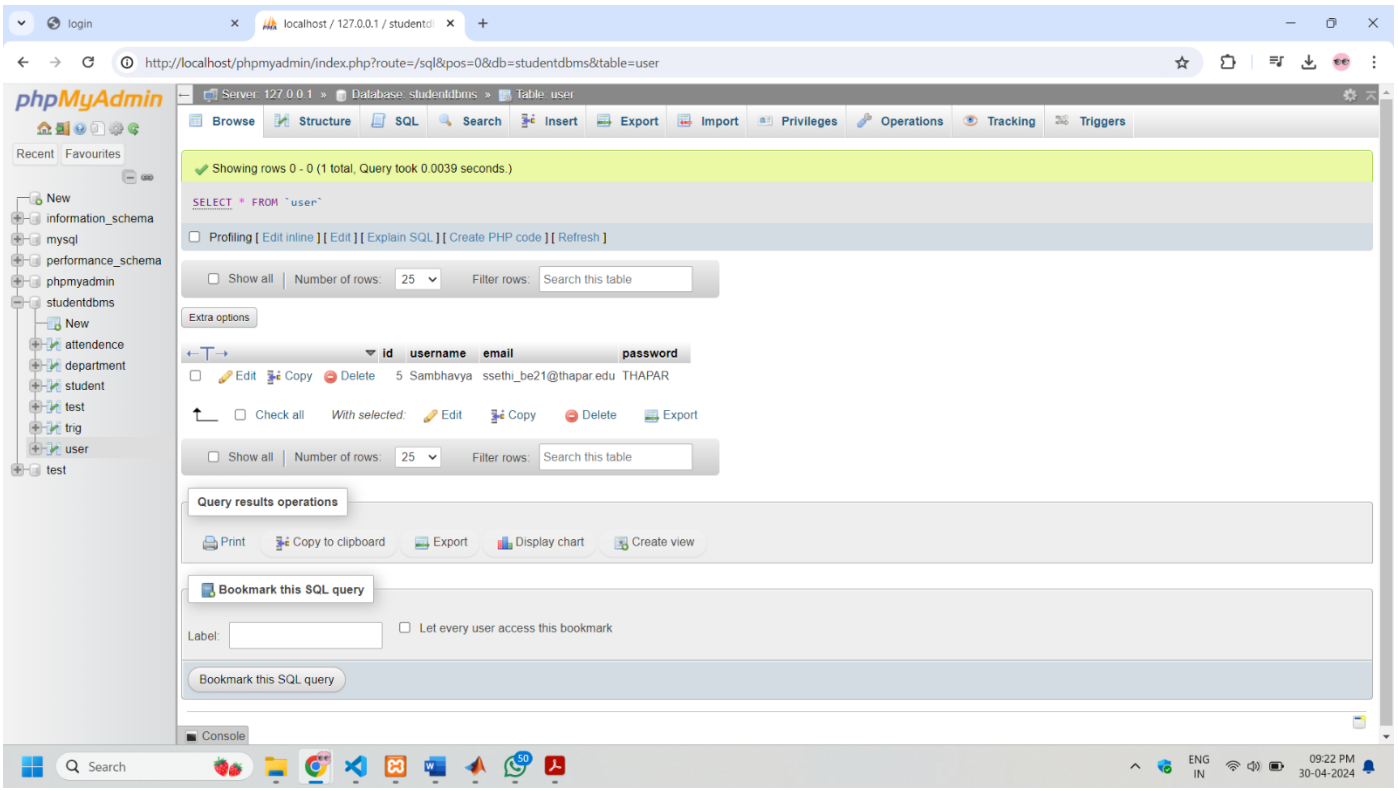
DATABASE LOCALHOST

PhpMyAdmin






phpMyAdmin
Server: 127.0.0.1 » Database: studentdbms
Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tracking Designer More
Filters
Containing the word:
Table Action Rows Type Collation Size Overhead
attendence Browse Structure Search Insert Empty Drop 1 InnoDB utf8mb4_general_ci 16.0 KiB -
department Browse Structure Search Insert Empty Drop 6 InnoDB utf8mb4_general_ci 16.0 KiB -
student Browse Structure Search Insert Empty Drop 1 InnoDB utf8mb4_general_ci 16.0 KiB -
test Browse Structure Search Insert Empty Drop 0 InnoDB utf8mb4_general_ci 16.0 KiB -
trig Browse Structure Search Insert Empty Drop 5 InnoDB utf8mb4_general_ci 16.0 KiB -
user Browse Structure Search Insert Empty Drop 1 InnoDB utf8mb4_general_ci 16.0 KiB -
6 tables Sum 14 InnoDB utf8mb4_general_ci 96.0 KiB 0 B
Check all With selected:
Print Data dictionary
Create new table
Table name
Number of columns
4
Create
Console


phpMyAdmin
Server: 127.0.0.1 » Database: studentdbms » Table: user
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Showing rows 0 - 0 (1 total, Query took 0.0039 seconds.)
SELECT * FROM `user`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all Number of rows: 25 Filter rows: Search this table
Extra options
id username email password
Edit Copy Delete 5 Sambhavya ssethi_be21@thapar.edu THAPAR
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query
Console

7.CONCLUSION

STUDENT MANAGEMENT SYSTEM successfully implemented based on online data filling which helps us in administrating the data user for managing the tasks performed in students. The project successfully used various functionalities of Xampp and python flask and also create the fully functional database management system for online portals.

Using MySQL as the database is highly beneficial as it is free to download, popular and can be easily customized. The data stored in the MySQL database can easily be retrieved and manipulated according to the requirements with basic knowledge of SQL.

With the theoretical inclination of our syllabus, it becomes very essential to take the utmost advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project “Students Management System” was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

• The planning that goes into implementing a project.

• The importance of proper planning and an organized methodology.

8.References

- https://chat.openai.com/c/576f95bf-1b96-488e-9f71-9c9351dce118
- https://www.w3schools.com/
- https://github.com/
- https://youtube.com/
- Silberschatz, Abraham, Henry F. Korth, and S. Sudarshan. "Database System Concepts." McGraw-Hill Education, 2019.
- GeeksforGeeks. "Database Normalization | Normal Forms." https://www.geeksforgeeks.org/normal-forms-in-dbms