A Mini Project Report

on

Student Management System

by

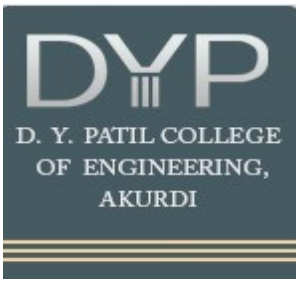
Yash Varpe(TEAIDB15)

Vedant Dhore (TEAIDB38)

Sourabh Kapse (TEAIDB47)

Under the guidance of

Prof. Manasi Karajgar



Department of Artificial Intelligence and Data Science

D. Y. PATIL COLLEGE OF ENGINEERING, AKURDI, PUNE

SAVITRIBAI PHULE PUNE UNIVERSITY

2022-2023

Department of Artificial Intelligence and Data Science

D. Y. Patil College of Engineering

Date:

CERTIFICATE

This is to certify that,

Yash Varpe(TEAIDB15)

Vedant Dhore (TEAIDB38)

Sourabh Kapse (TEAIDB47)

of class T.E; have successfully completed their mini project work on "Student Management System'' at D. Y. Patil College of Engineering in the partial fulfillment of the Graduate Degree course in T.E at the depart- ment of **Artificial Intelligence and Data Science** in the academic Year 2022-2023 Semester – I as prescribed by the Savitribai Phule Pune University.

Prof. Manasi Karajgar
Guide

Dr. Vinayak Kottawar
Head of the Department
(Department of AI & DS)

INTRODUCTION

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.

LIMITATIONS:

- Time consumption in data entry as the records are to be manually maintained faculties a lot of time.
- Lot of paper work is involved as the records are maintained in the files and registers.
- Storage Requires as files and registers are used the storage space requirement is increased.
- Less Reliable use of papers for storing valuable data information is not at all reliable.
- Aadhar linkage with the official aadhar database has not been done.

STUDY OF EXISTING SYSTEM

CASE STUDY

The success of any organization such as School of Public Health, University of Ghana hinges on its ability to acquire accurate and timely data about its operations, to manage this data effectively, and to use it to analyze and guide its activities. Integrated student database system offer users (Student, Registrar, HOD) with a unified view of data from multiple sources. To provide a single consistent result for every object represented in these data sources, data fusion is concerned with resolving data inconsistency present in the heterogeneous sources of data. The main objective of this project is to build a rigid and robust integrated student database system that will track and store records of students. This easy-to-use, integrated database application is geared towards reducing time spent on administrative tasks. The system is intended to accept process and generate report accurately and any user can access the system at any point in time provided internet facility is available. The system is also intended to provide better services to users, provide meaningful, consistent, and timely data and information and finally promotes efficiency by converting paper processes to electronic form. The system was developed using technologies such as, HTML, CSS ,JS and MySQL. PYTHON- FLASK, HTML and CSS are used to build the user interface and database was built using MySQL. The system is free of errors and very efficient and less time consuming due to the care taken to develop it. All the phases of software development cycle are employed and it is worthwhile to state that the system is very robust. Provision is made for future development in the system.

PROPOSED SYSTEM

While there has been no consensus on the definition of Students Management in the literature, they 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.

DATABASE DESIGN

SOFTWARE REQUIREMENTS:

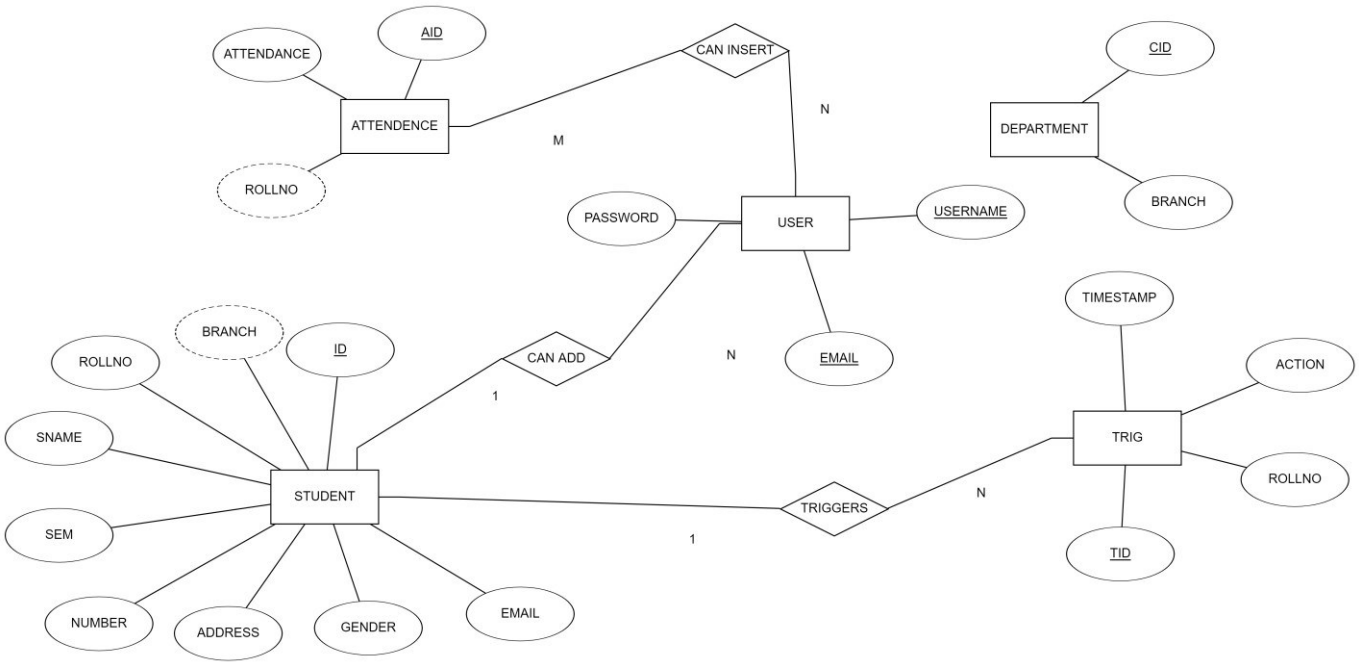
Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.9) , SQL

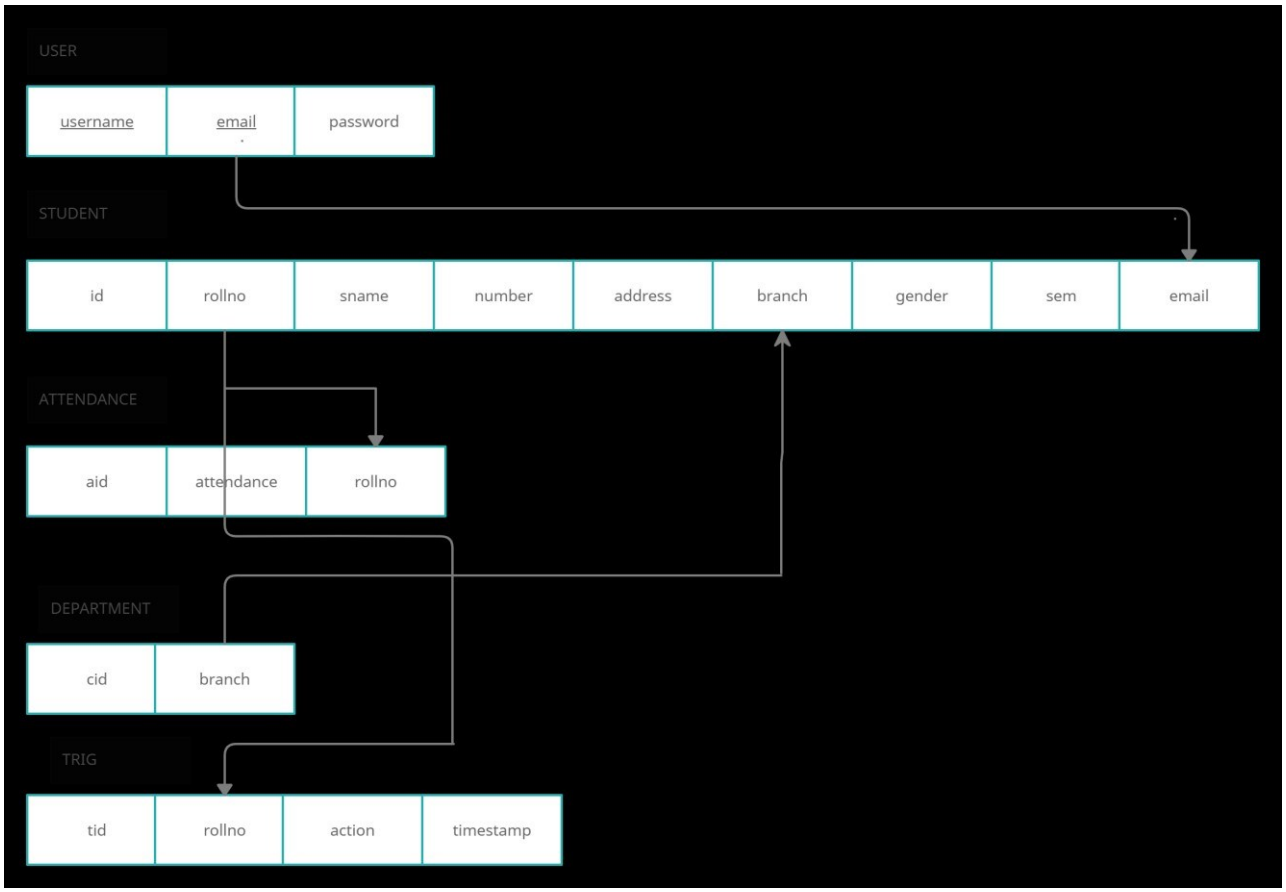
- Operating System: Windows 11
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- Python main editor (user interface): VS Code
- workspace editor: VS Code

CONCEPTUAL DESIGN:

E-R DIAGRAM:



SCHEMA DIAGRAM:



IMPLEMENTATION

An "implementation" of Python should be taken to mean a program or environment which provides support for the execution of programs written in the Python language, as represented by the CPython reference implementation.

There have been and are several distinct software packages providing of what we all recognize as Python, although some of those are more like distributions or variants of some existing implementation than a completely new implementation of the language.

Back End (MySQL) Database:

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.

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/database_table
_ name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/dyp_students'
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`")
return render_template('studentdetails.html',query=query)

@app.route('/triggers') def
triggers():
    query=db.engine.execute(f"SELECT * FROM `trig`")
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`")
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):
    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`")
posts=Student.query.filter_by(id=id).first()     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`='{em
ail}',`number`='{num}',`address`='{address}'")
        flash("Slot is Updates","success")
return redirect('/studentdetails')

    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=encpassword)
    # 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):
        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`")
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}')")          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)
```

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>
```

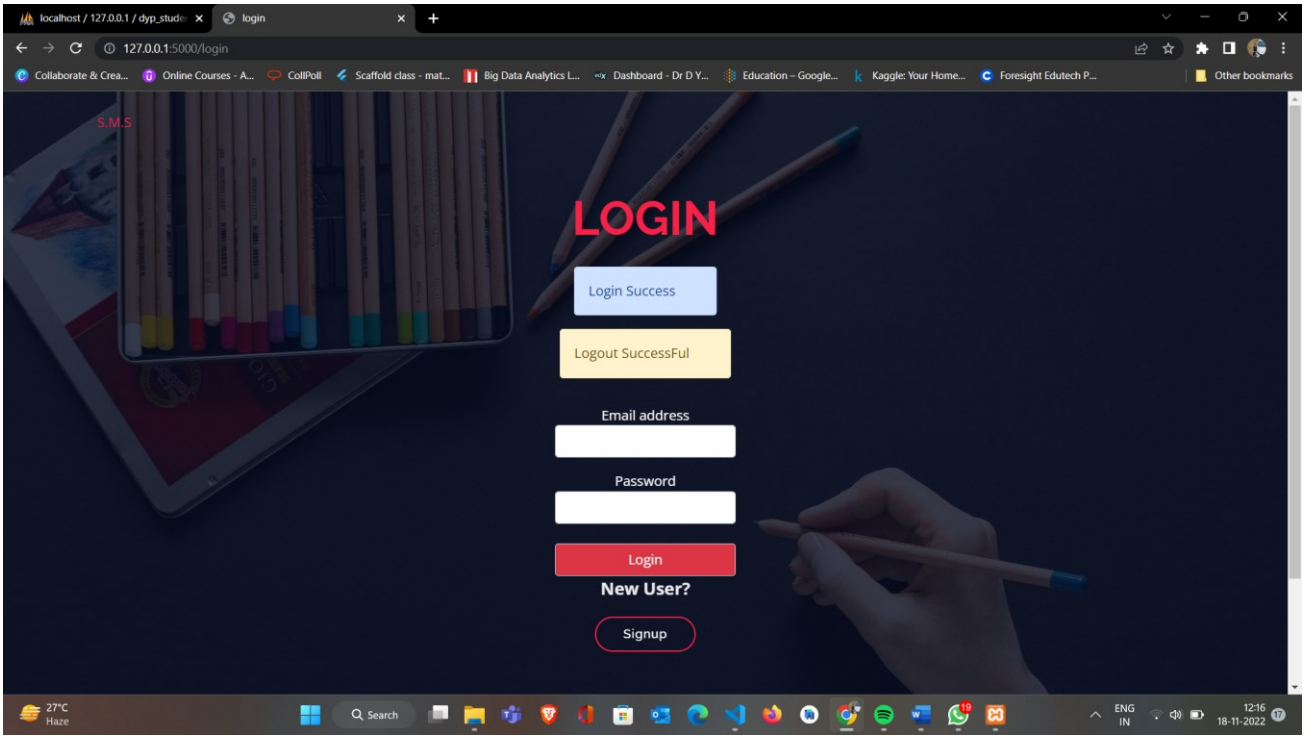
```
<div class="form-group">
<select  class="form-control" id="gender" name="gender"  required>
      <option selected>Select Gender</option>

      <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>
```

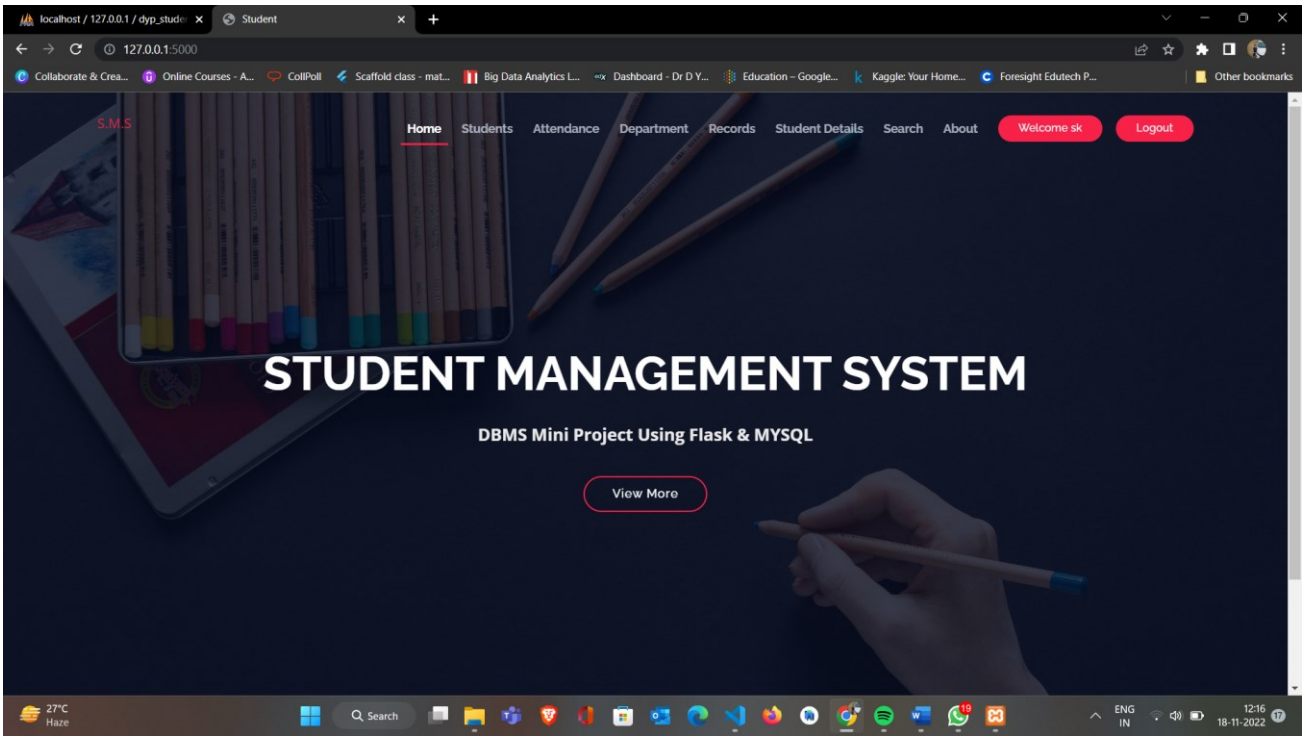
```
<br>

  <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 %}
```

USER INTERFACE

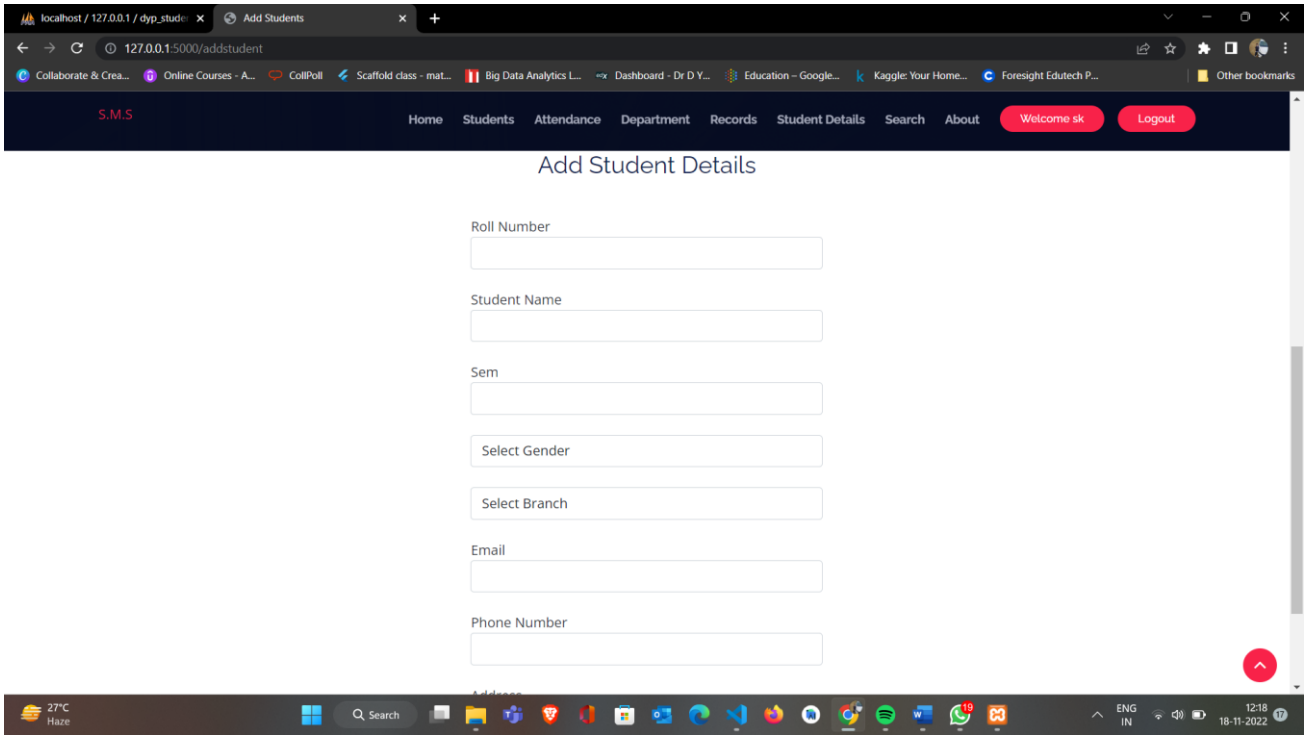
SCREEN SHOTS

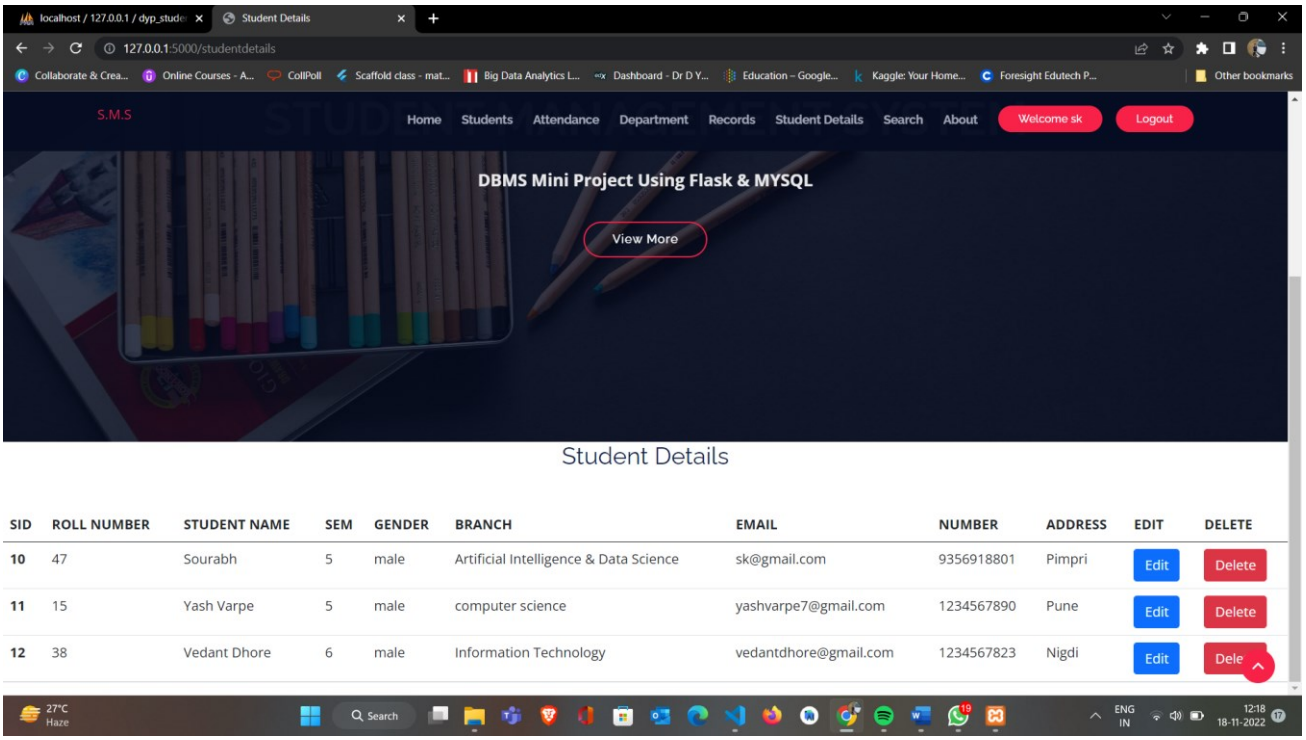
LOGIN PAGE:



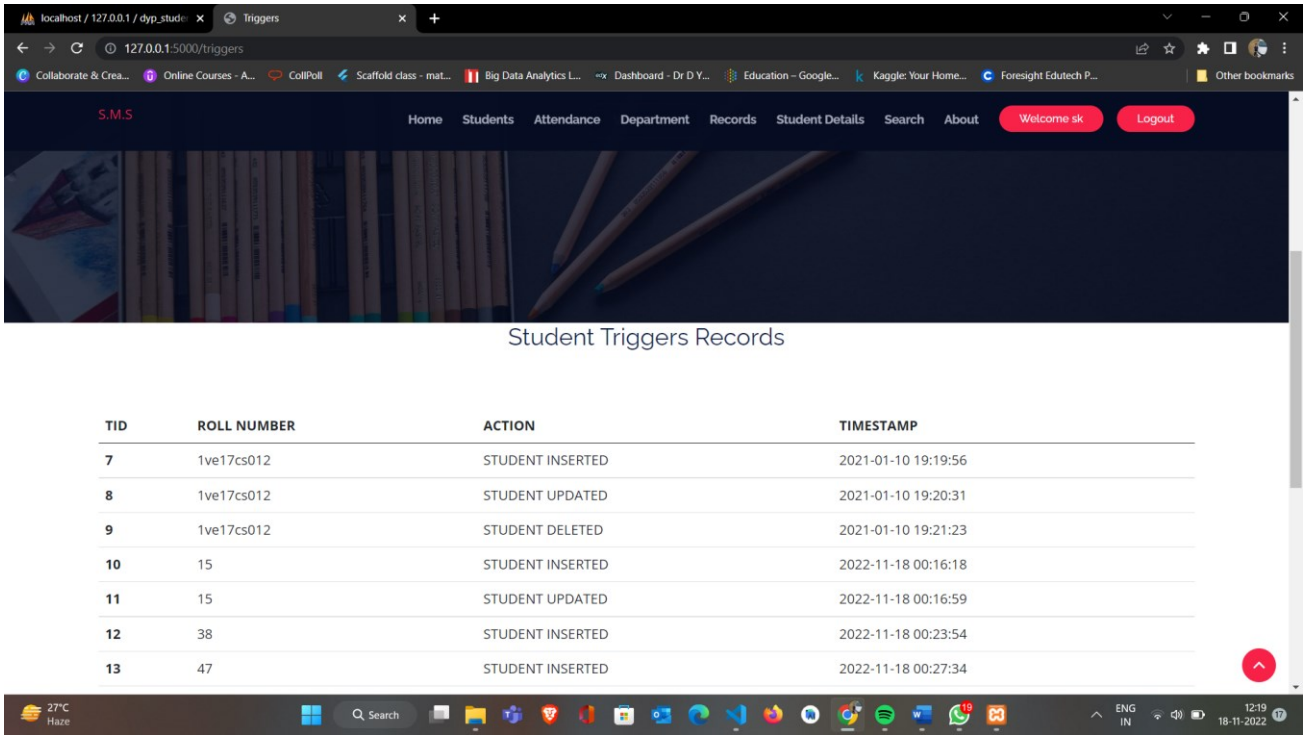


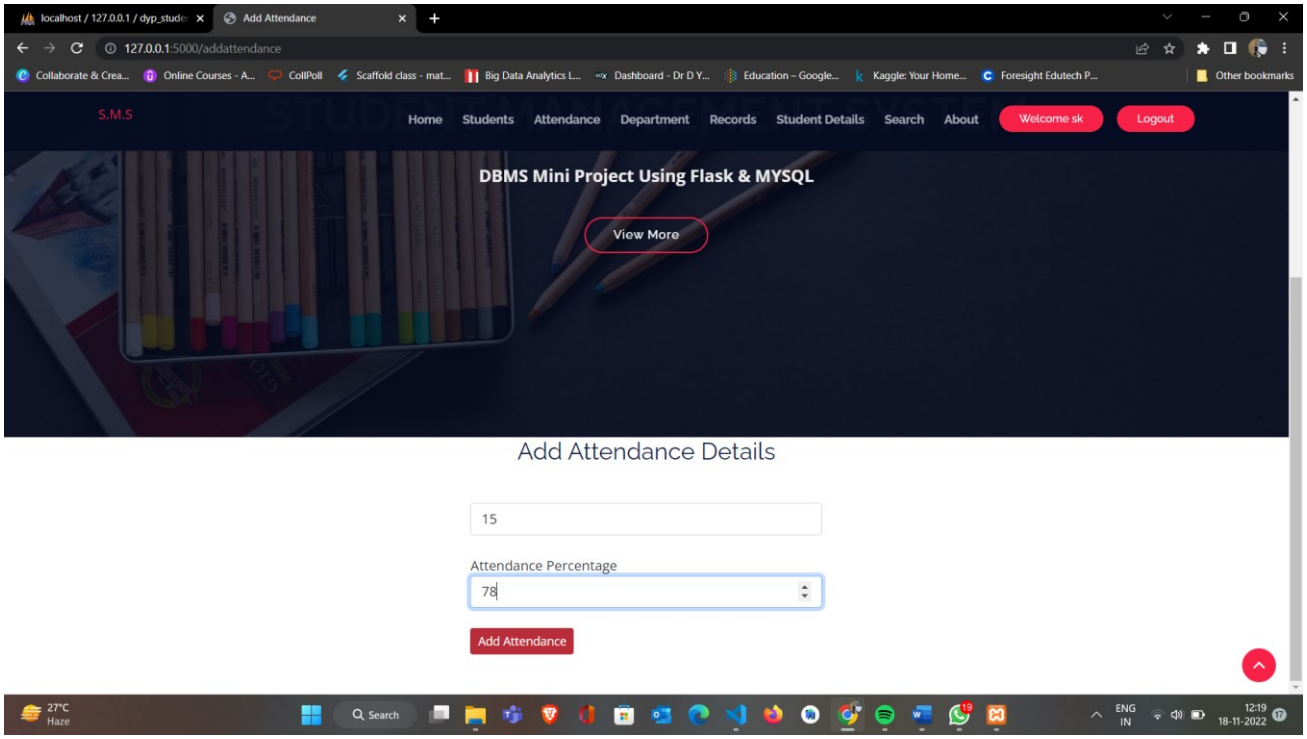
ADDING STUDENTS INFO

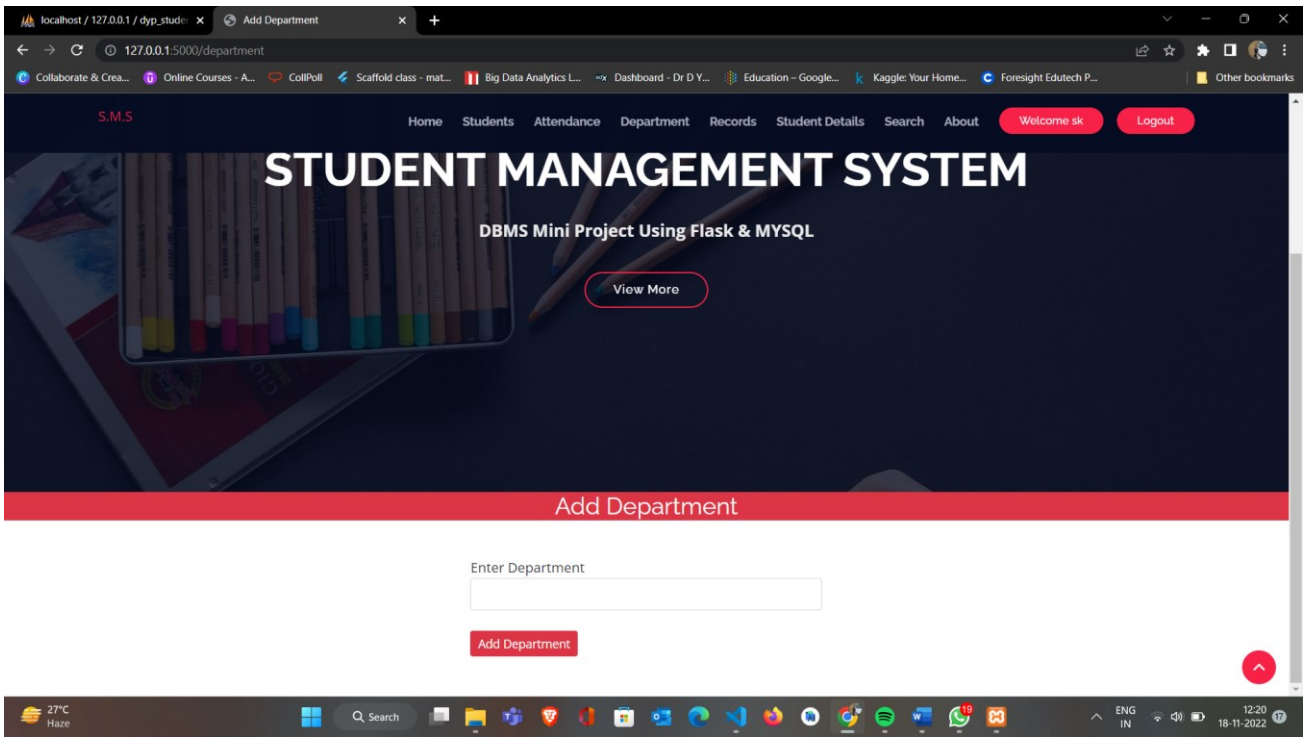


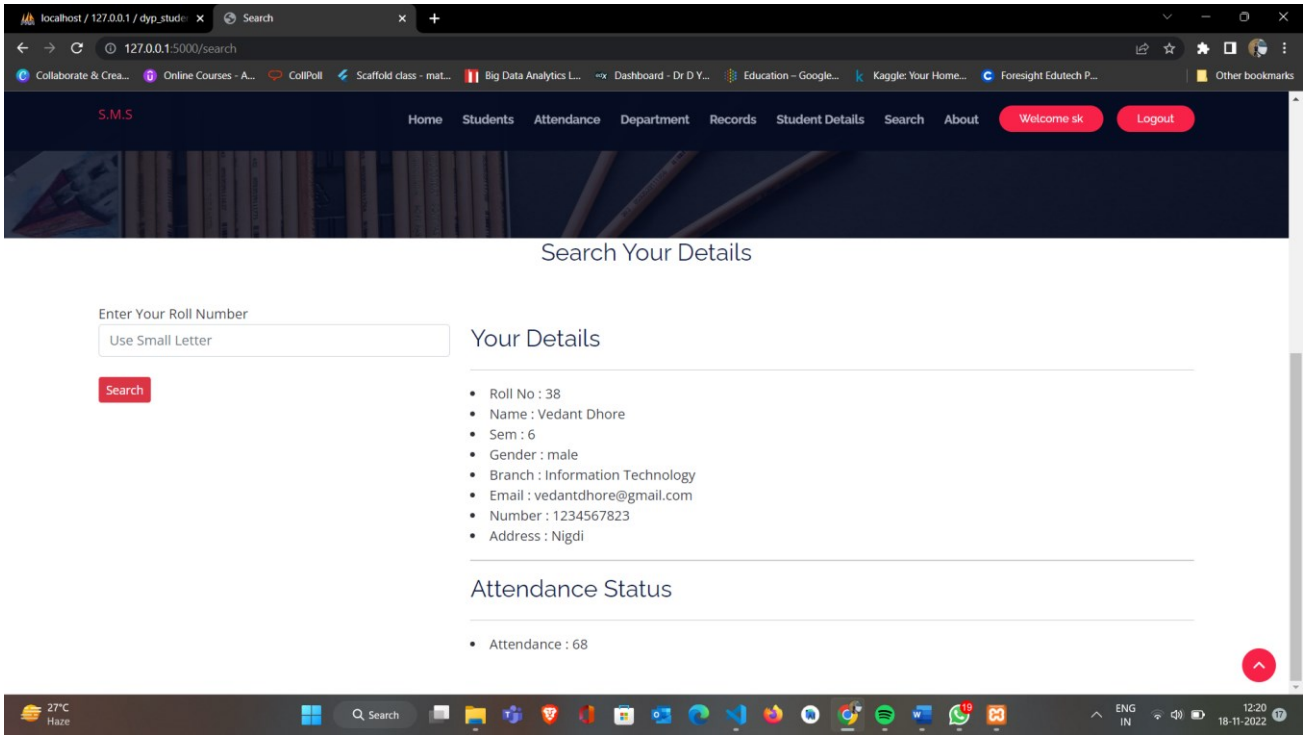


TRIGGERS RECORDS

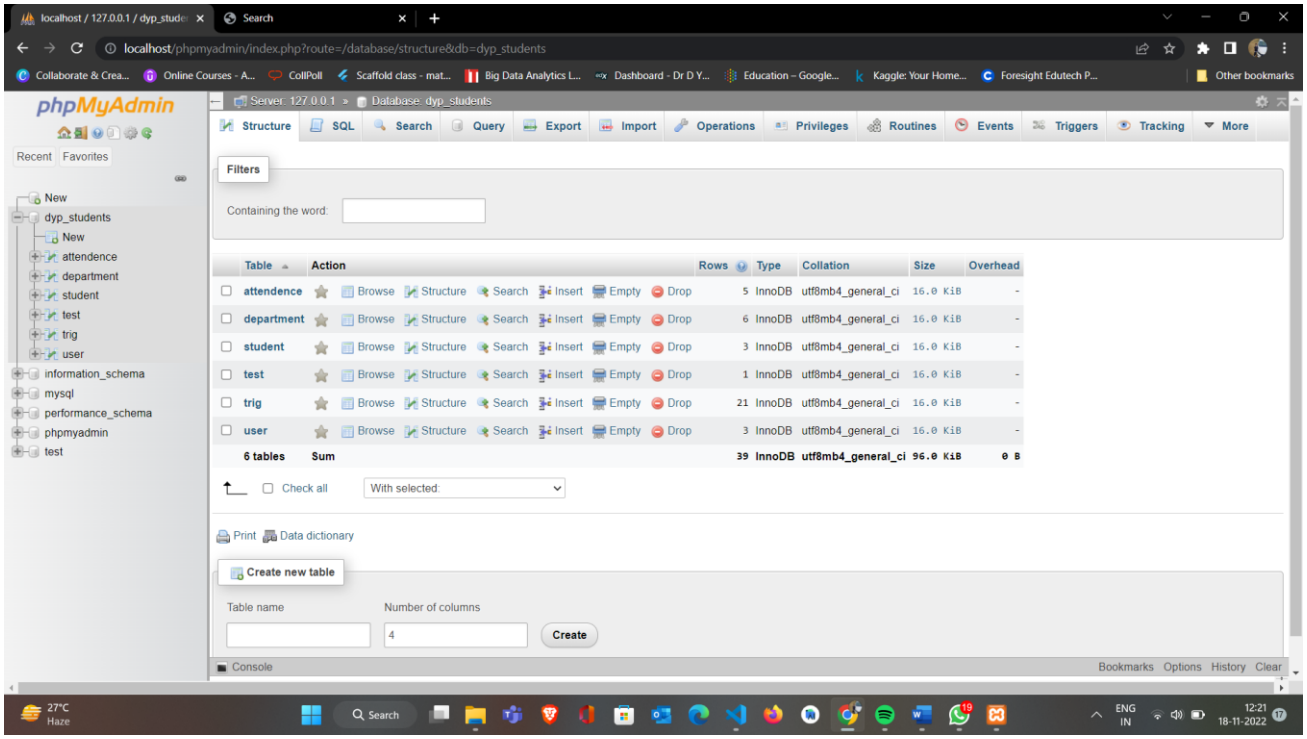


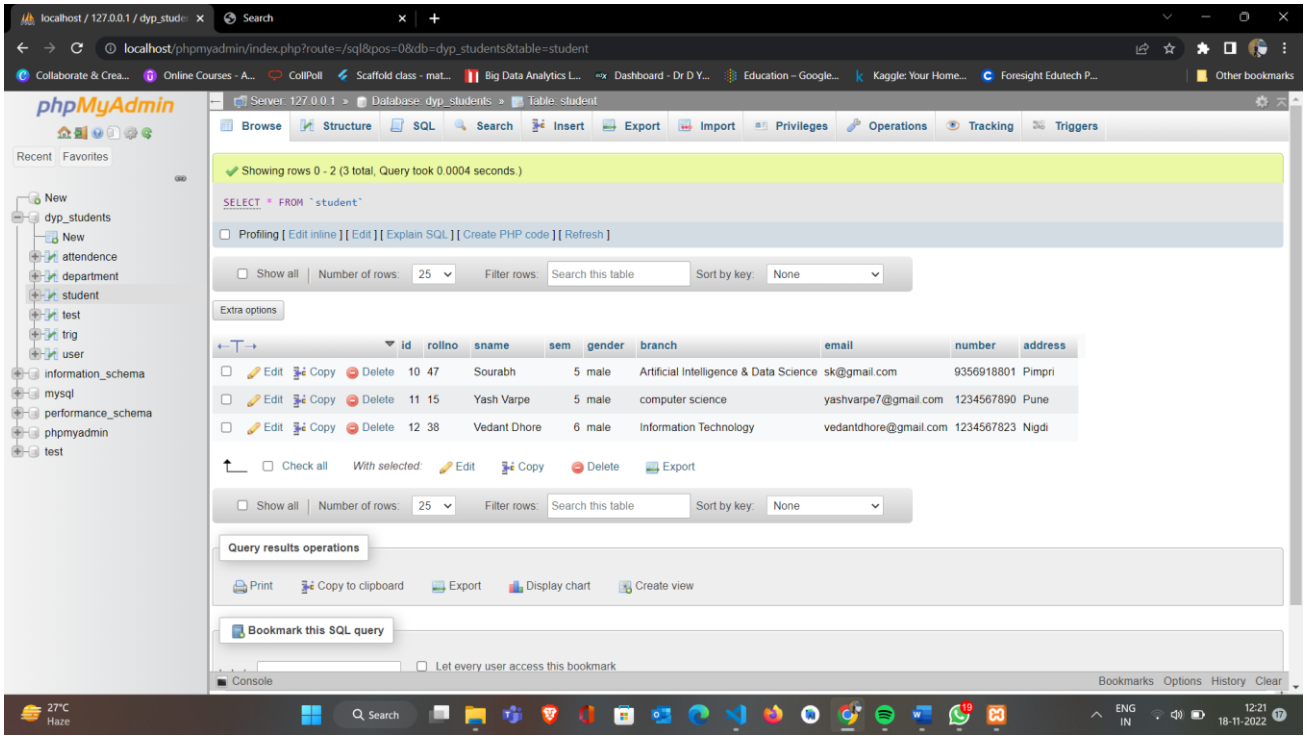


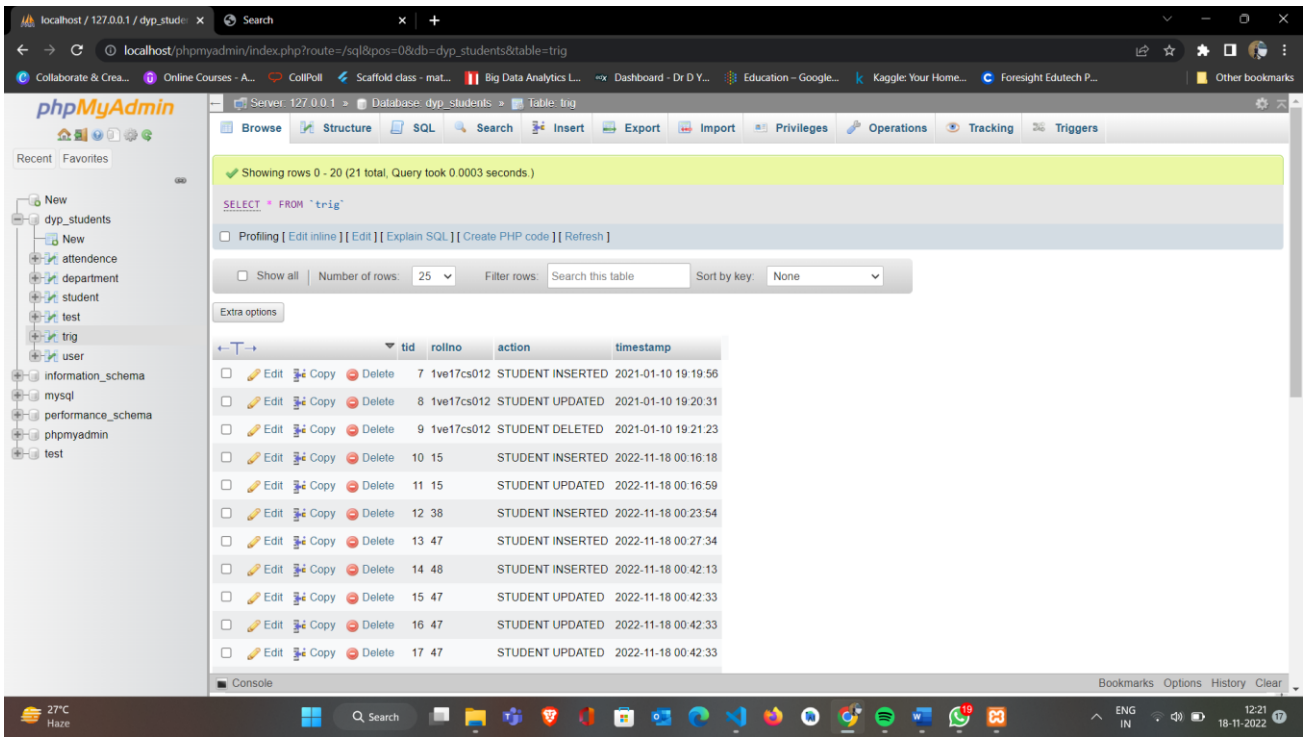

localhost / 127.0.0.1 / dyp_stude
Add Department
127.0.0.1:5000/department
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome sk
Logout
STUDENT MANAGEMENT SYSTEM
DBMS Mini Project Using Flask & MYSQL
View More
Add Department
Enter Department
Add Department

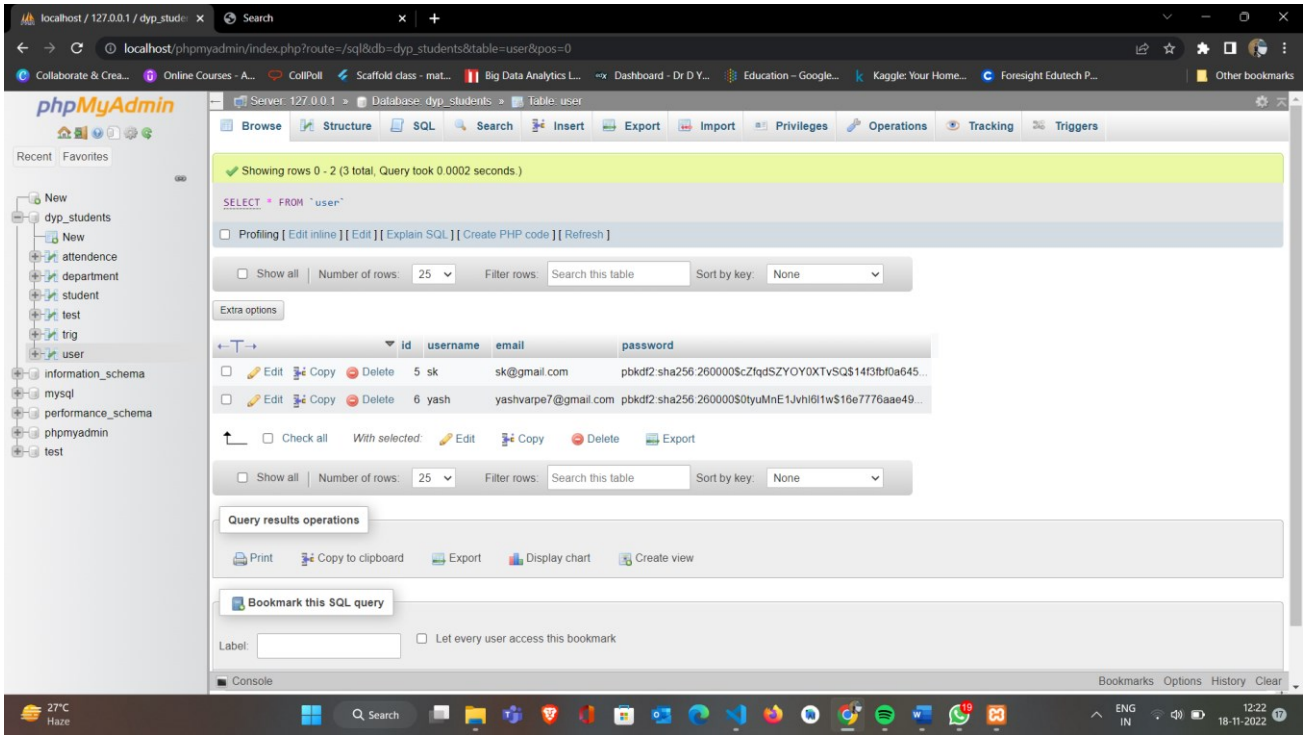

localhost / 127.0.0.1 / dyp_stude
Search
127.0.0.1:5000/search
S.M.S
Home
Students
Attendance
Department
Records
Student Details
Search
About
Welcome sk
Logout
Search Your Details
Enter Your Roll Number
Use Small Letter
Search
Your Details
Roll No : 38
Name : Vedant Dhore
Sem : 6
Gender : male
Branch : Information Technology
Email : vedantdhore@gmail.com
Number : 1234567823
Address : Nigdi
Attendance Status
Attendance : 68

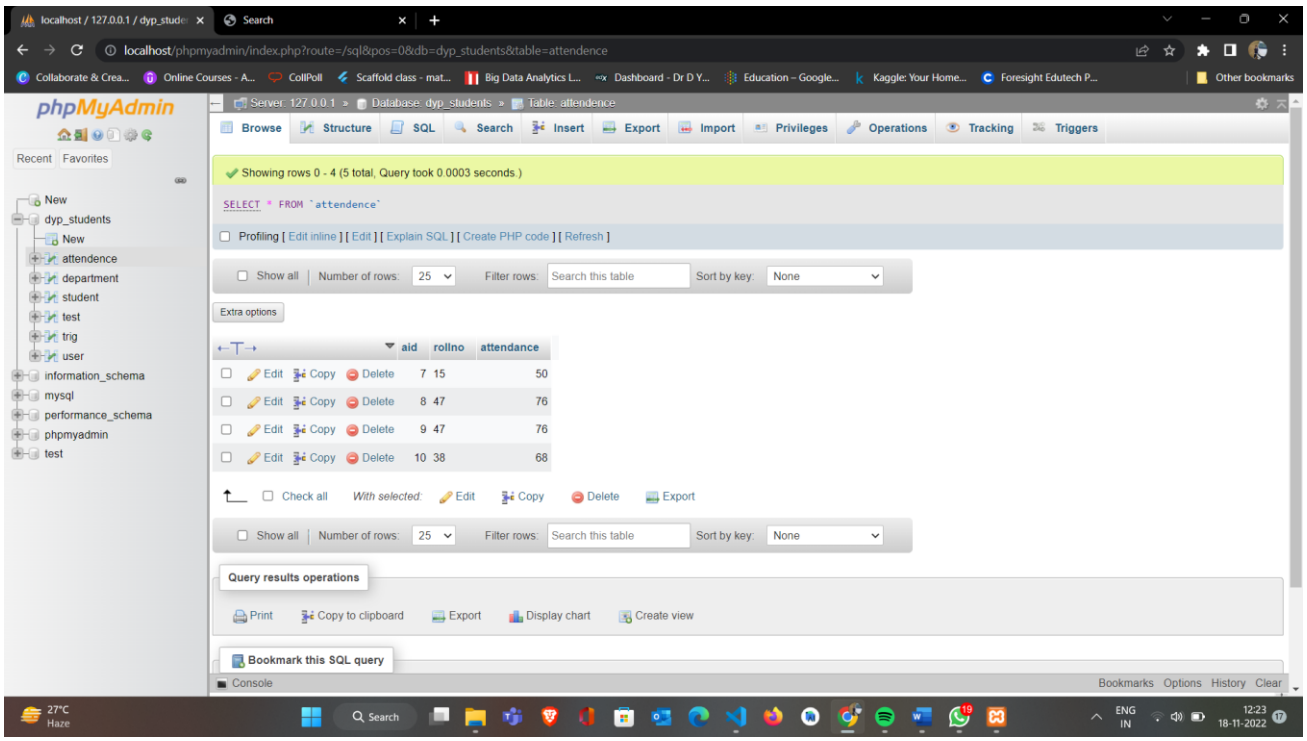
DATABASE LOCALHOST






localhost / 127.0.0.1 / dyp_stude
Search
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=dyp_students&table=trig
phpMyAdmin
Server: 127.0.0.1 » Database: dyp_students » Table: trig
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Showing rows 0 - 20 (21 total, Query took 0.0003 seconds.)
SELECT * FROM `trig`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
tid rollno action timestamp
7 1ve17cs012 STUDENT INSERTED 2021-01-10 19:19:56
8 1ve17cs012 STUDENT UPDATED 2021-01-10 19:20:31
9 1ve17cs012 STUDENT DELETED 2021-01-10 19:21:23
10 15 STUDENT INSERTED 2022-11-18 00:16:18
11 15 STUDENT UPDATED 2022-11-18 00:16:59
12 38 STUDENT INSERTED 2022-11-18 00:23:54
13 47 STUDENT INSERTED 2022-11-18 00:27:34
14 48 STUDENT INSERTED 2022-11-18 00:42:13
15 47 STUDENT UPDATED 2022-11-18 00:42:33
16 47 STUDENT UPDATED 2022-11-18 00:42:33
17 47 STUDENT UPDATED 2022-11-18 00:42:33
Console
Bookmarks Options History Clear


localhost / 127.0.0.1 / dyp_stude
Search
localhost/phpmyadmin/index.php?route=/sql&db=dyp_students&table=user&pos=0
phpMyAdmin
Server: 127.0.0.1 » Database: dyp_students » Table: user
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Showing rows 0 - 2 (3 total, Query took 0.0002 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 Sort by key: None
Extra options
id username email password
5 sk sk@gmail.com pbkdf2:sha256:260000$cZfqdSZYOY0XTvSQ$14f3fbf0a645...
6 yash yashvarpe7@gmail.com pbkdf2:sha256:260000$0tyuMnE1Jvhl6l1w$16e7776aae49...
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Label: Let every user access this bookmark
Console
Bookmarks Options History Clear


localhost / 127.0.0.1 / dyp_stude
Search
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=dyp_students&table=attendence
phpMyAdmin
Recent Favorites
New
dyp_students
New
attendence
department
student
test
trig
user
information_schema
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: dyp_students » Table: attendence
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)
SELECT * FROM `attendence`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
aid rollno attendance
Edit Copy Delete 7 15 50
Edit Copy Delete 8 47 76
Edit Copy Delete 9 47 76
Edit Copy Delete 10 38 68
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Console
Bookmarks Options History Clear
27°C Haze
Search
ENG IN
12:23 18-11-2022

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 atmost 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.

• The key element of team spirit and co-ordination in a successful project.

REFERENCES

- https://www.youtube.com
- https://www.google.com
- http://www.getbootstrap.com
- http://www.python.org